Simultaneous Localization and Mapping—A Unifying Optimization-Based Framework

Chih-Yuan (Frank) Chiu

Spring 2023

Outline

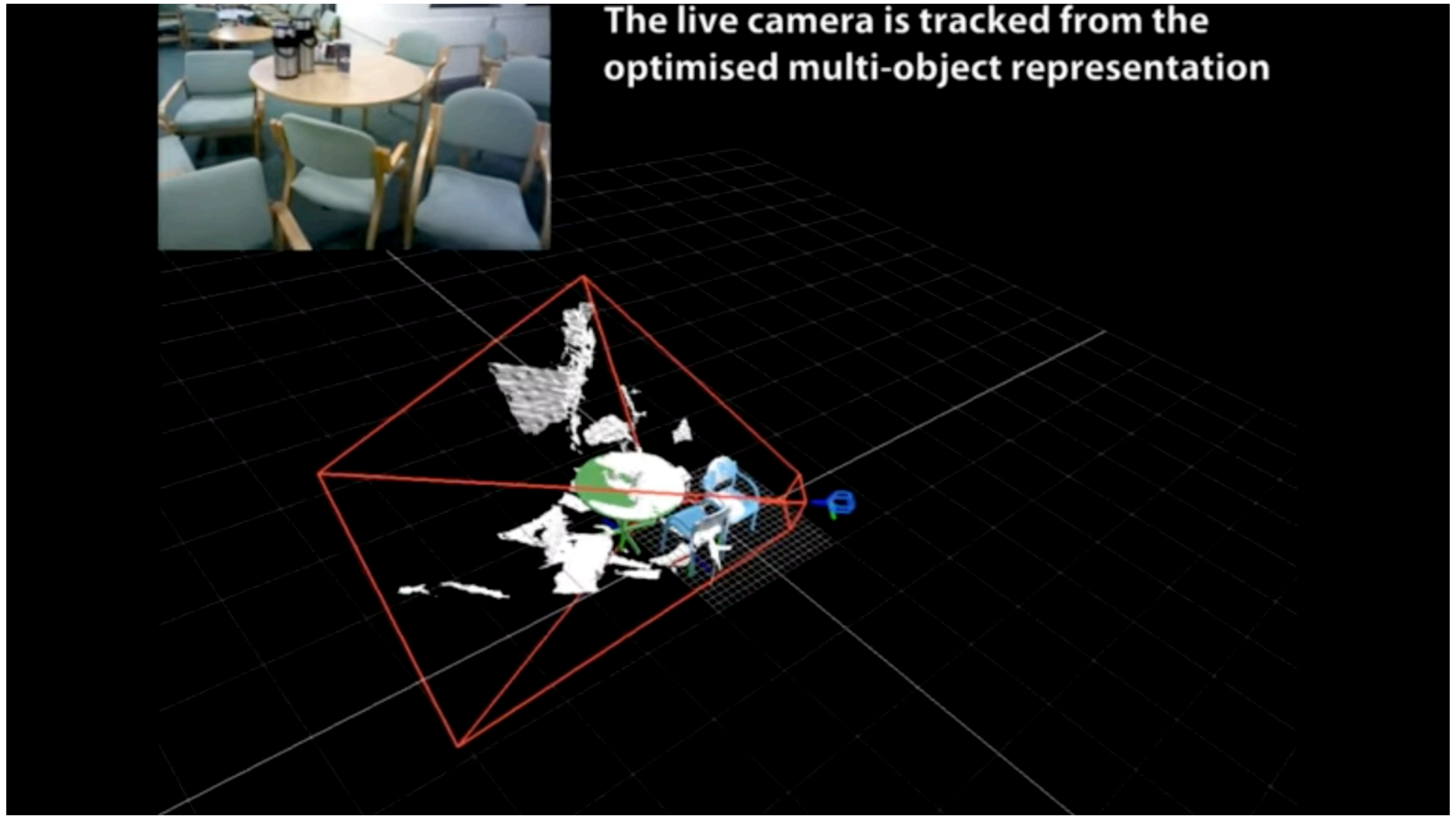
- **Introduction, FAQs**
- **Front End**
  - Feature Extraction
  - Data Association, with Outlier Rejection
- **Back End**
  - Goal, Setup
  - Unifying Framework: 3 Steps
  - Aside: Kalman Filtering, Basics
  - Example: EKF SLAM
  - State-of-the-art Algorithms
  - Experiments
  - Loop Closures
- **Next Steps**
  - Active Perception: Dynamic, Semantic SLAM

Outline

- **Introduction, FAQs**
- **Front End**
  - Feature Extraction
  - Data Association, with Outlier Rejection
- **Back End**
  - Goal, Setup
  - Unifying Framework: 3 Steps
  - Aside: Kalman Filtering, Basics
  - Example: EKF SLAM
  - State-of-the-art Algorithms
  - Experiments
  - Loop Closures
- **Next Steps**
  - Active Perception: Dynamic, Semantic SLAM

Intro: What is SLAM?

- **Simultaneous Localization and Mapping: (SLAM)**
  - Localization — Estimate robot state (pose, velocity, etc.).
  - Mapping — Construct a map (landmarks, features, etc.) of its surroundings.



Video Credit: "SLAM++: Simultaneous Localisation and Mapping at the Level of Objects," (https://www.youtube.com/watch?v=tmrAh1CqCRo)
Cadena et al, "Past, Present, and Future of Simultaneous Localization and Mapping: Toward the Robust-Perception Age," IEEE Transactions on Robotics, 2016.

Intro: Why SLAM, and When?

- **Why SLAM is useful:**
  - SLAM research has produced the visual-inertial odometry 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)

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

Intro: 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.

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

Intro: 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.

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

Intro: Terminology

- **Robot Pose:**
  - Position and orientation of the robot camera.
  - Described by a translational (e.g., vector) component and a rotational (e.g., SO(3) or quaternion) component.
- **Feature:**
  - Positions (2D or 3D) of notable attributes in 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).

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

Intro: Terminology

- **Image Measurement:**
  - 2D measurements of the surroundings, periodically captured by robot sensors (e.g., cameras)
  - Provides metric information for robot poses and features
- **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

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

Intro: 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).

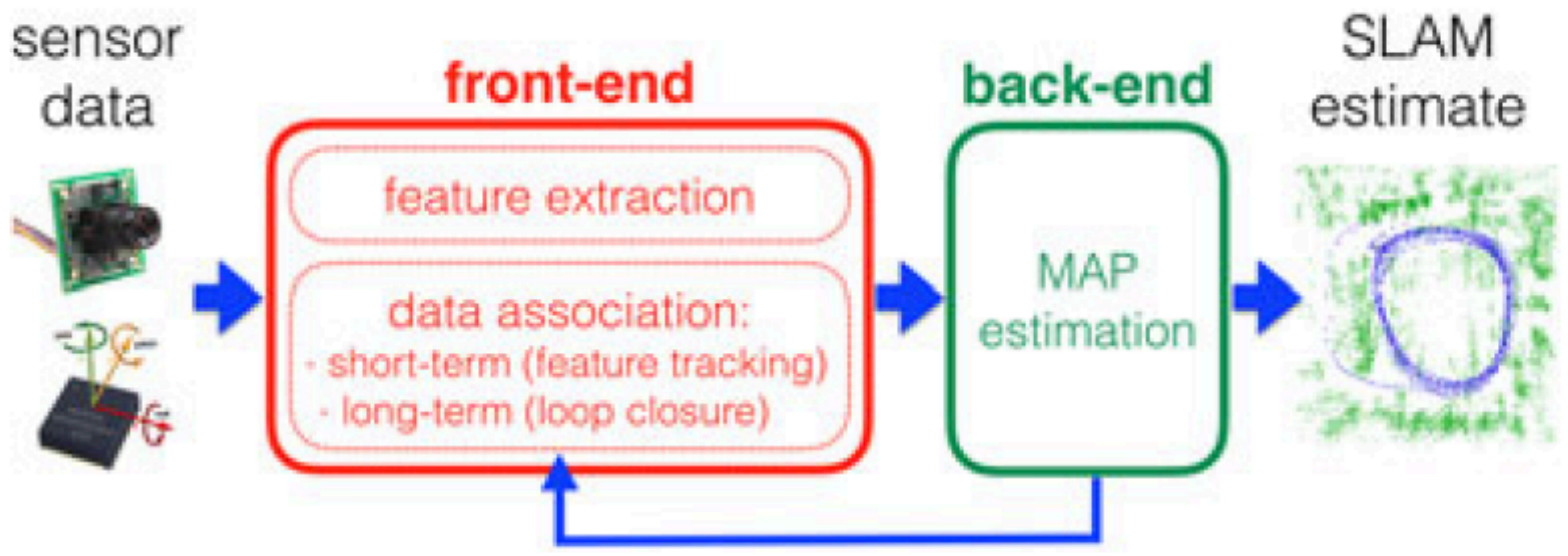


Fig. 2. Front end and back end in a typical SLAM system. The back end can provide feedback to the front end for loop closure detection and verification.

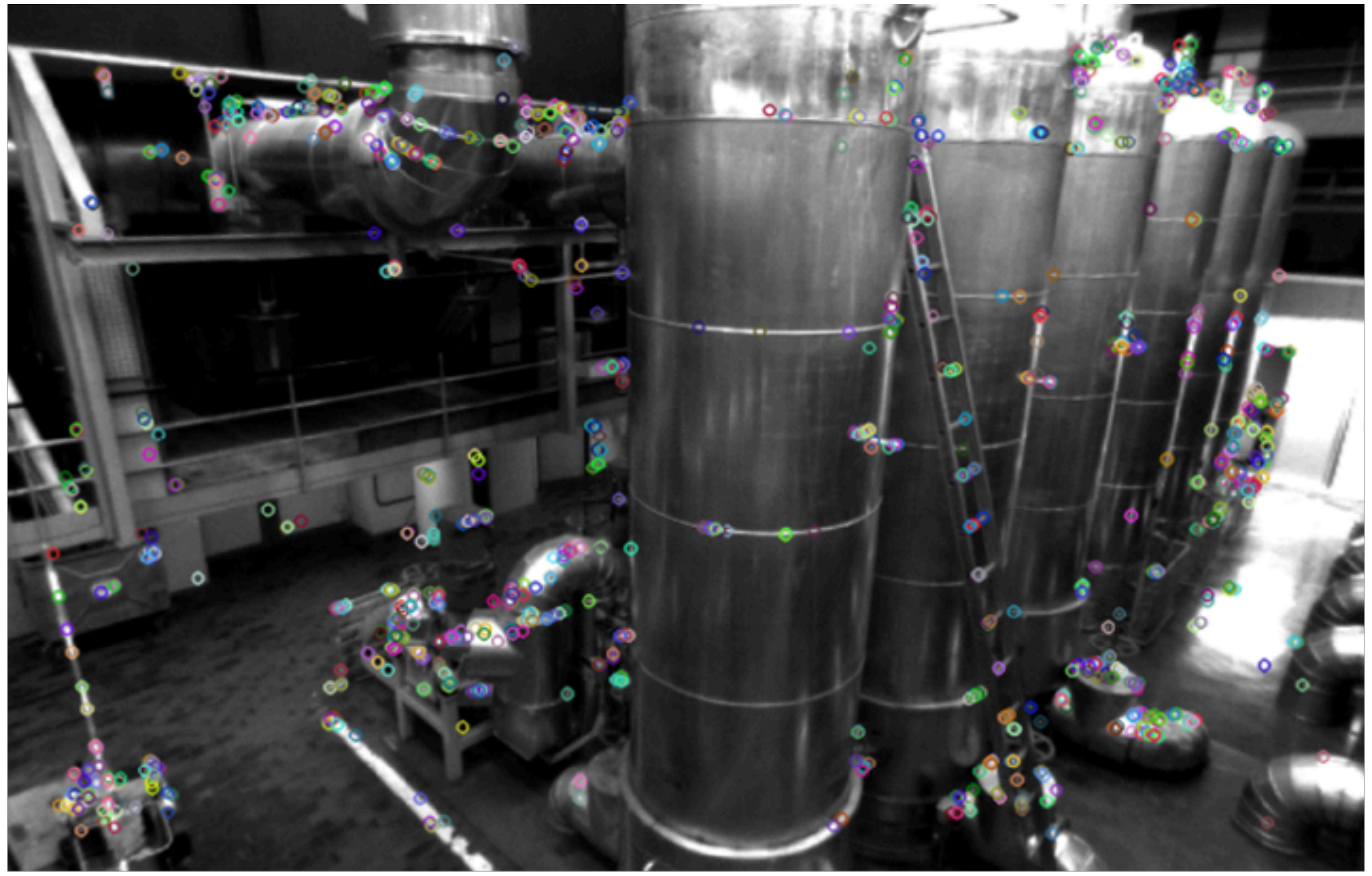
Cadena et al, "Past, Present, and Future of Simultaneous Localization and Mapping: Toward the Robust-Perception Age," IEEE Transactions on Robotics, 2016.

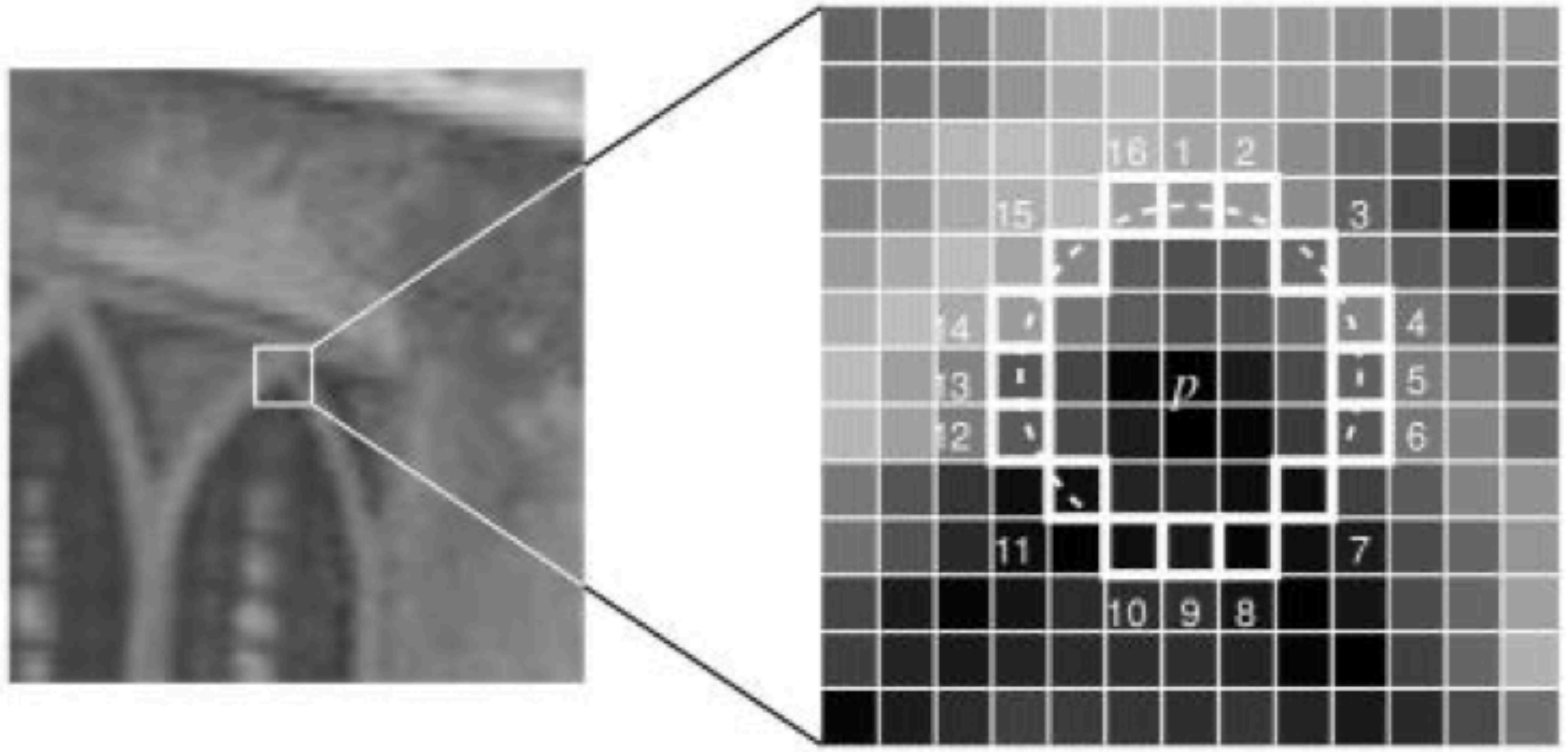
Outline

- **Introduction, FAQs**
- **Front End**
  - Feature Extraction
  - Data Association, with Outlier Rejection
- **Back End**
  - Goal, Setup
  - Unifying Framework: 3 Steps
  - Aside: Kalman Filtering, Basics
  - Example: EKF SLAM
  - State-of-the-art Algorithms
  - Experiments
  - Loop Closures
- **Next Steps**
  - Active Perception: Dynamic, Semantic SLAM

Front End: Feature Extraction

- **Goal:** Extract repeatably detected features from raw images.
- **Key:** Find distinctive “image patches” detectable from multiple views.
  - Corner points work well.
  - Corner detectors: FAST, Harris, DoG

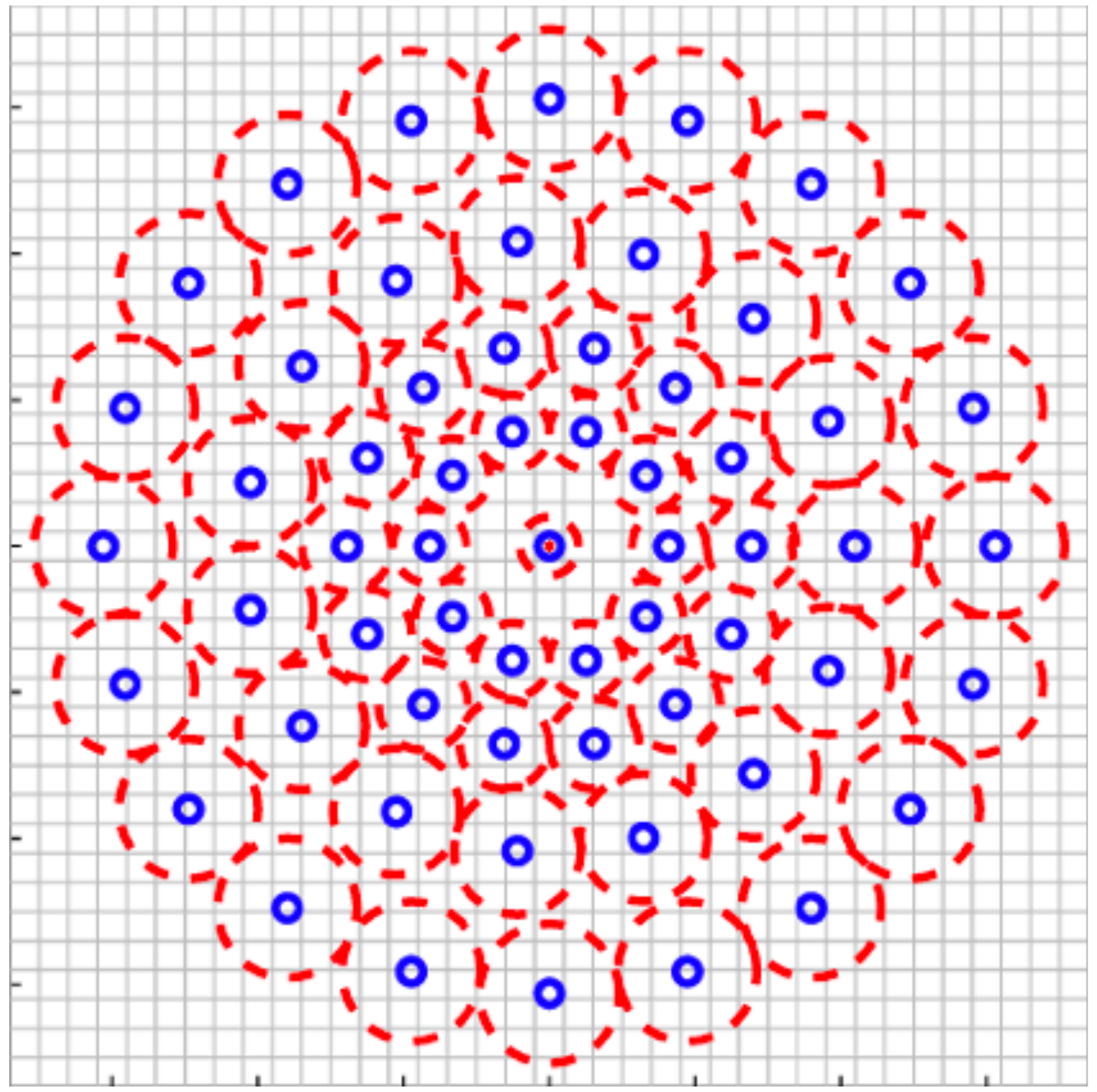


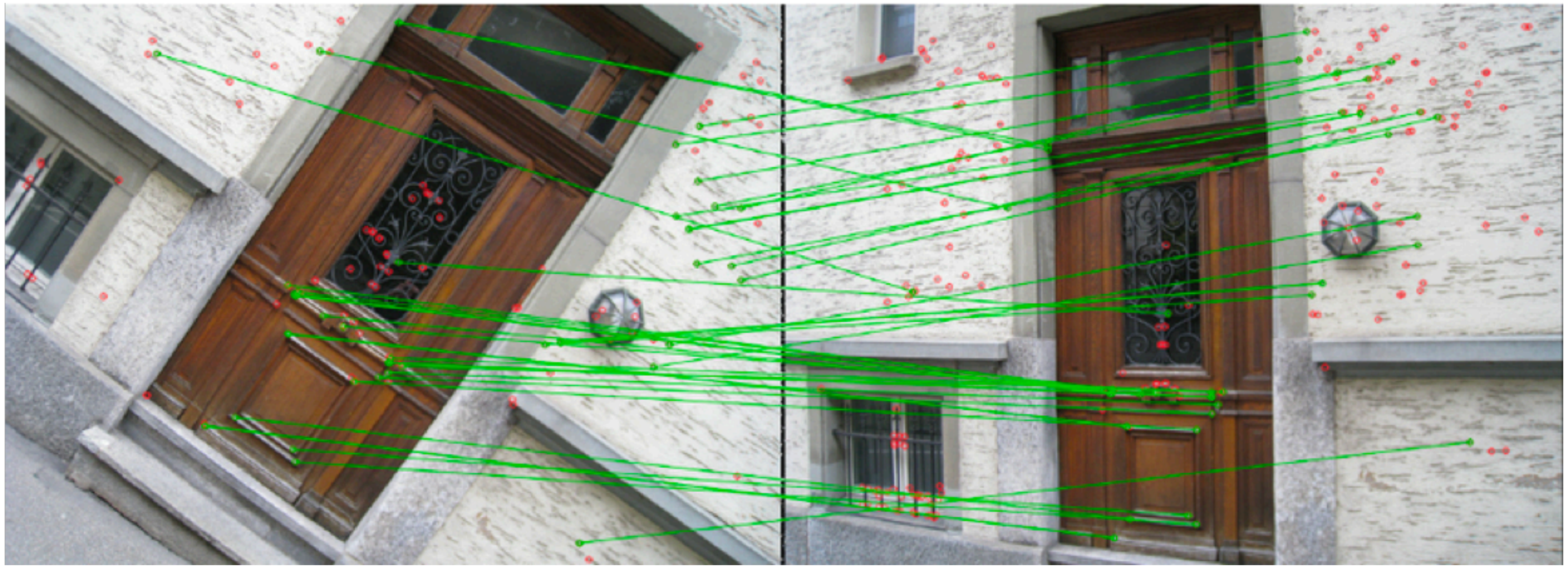


Rosten, Edward, and Tom Drummond. "Machine learning for high-speed corner detection." European conference on computer vision. Springer, Berlin, Heidelberg, 2006.

Front End: Data Association

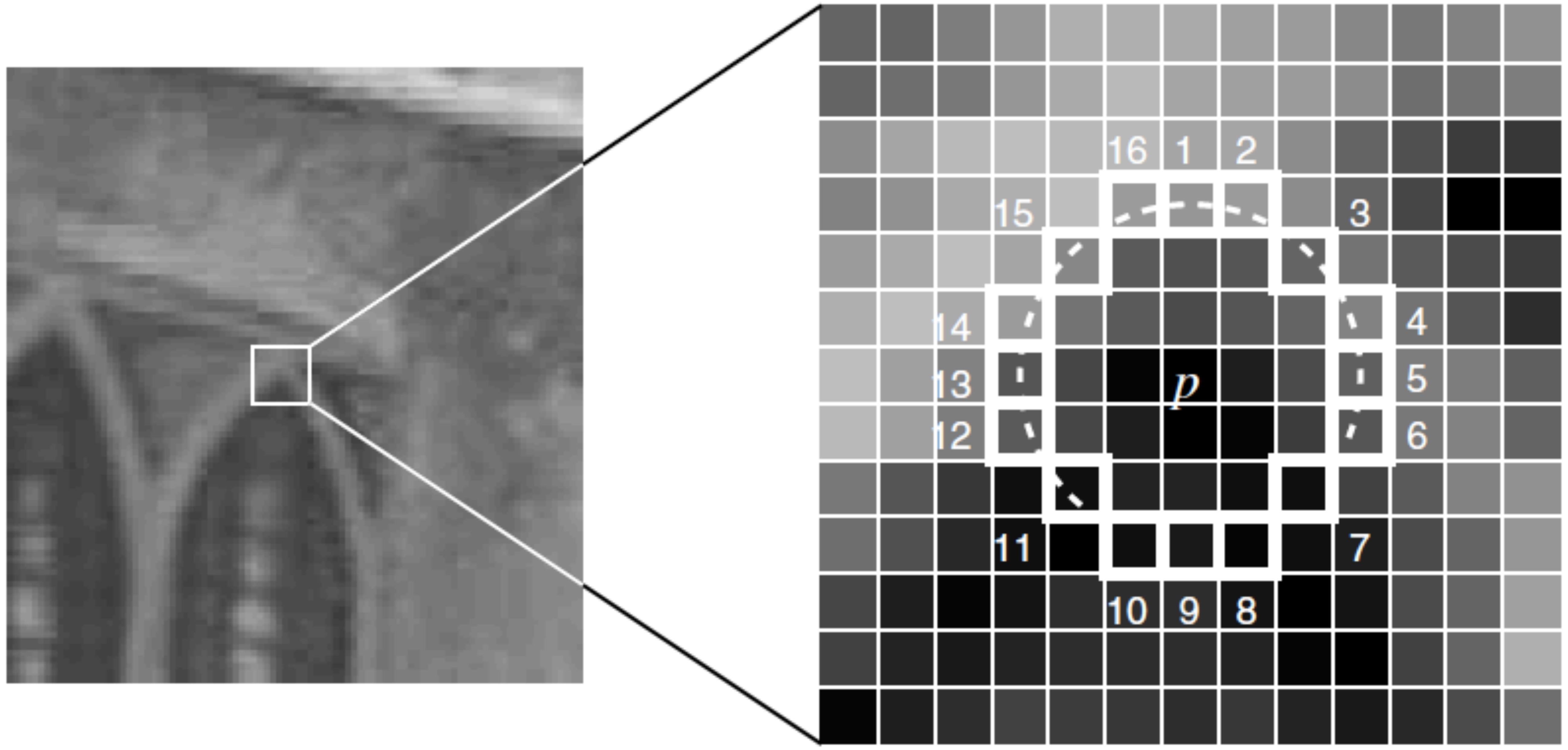
- To reliably detect the same point in multiple views of a scene:
  - Describe the image patch around a feature point in a way that is comparable, informative, and invariant to camera orientation.
  - Given a camera pose estimate, triangulate feature location from multiple views.
  - Data association methods: SURF, SIFT, FAST, BRIEF, ORB





Front End: Data Association

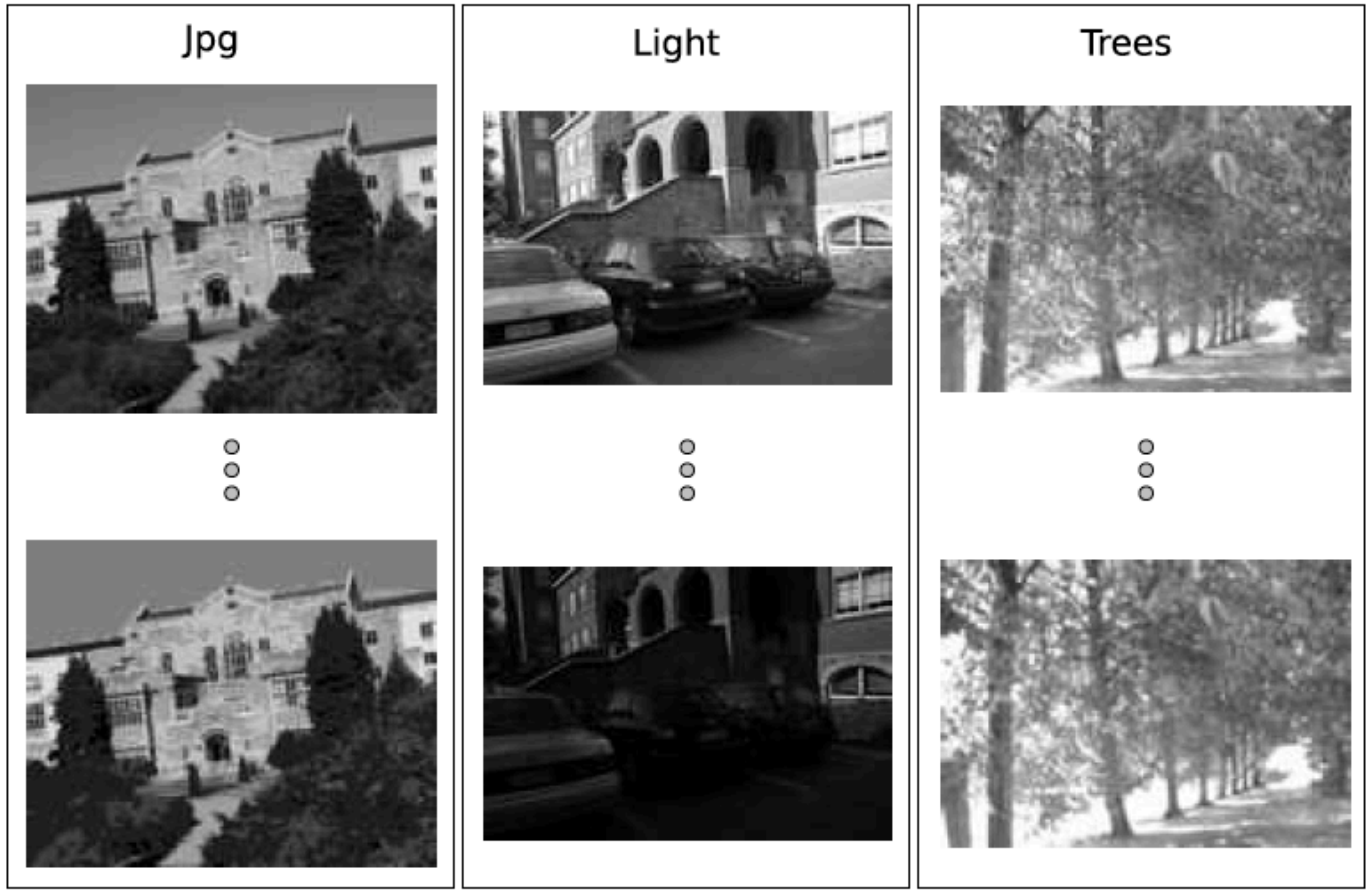
- **FAST: (Features from Accelerated Segment Test)**
  - Compares each pixel with its neighboring pixels
  - If the pixels is "sufficiently" different from "most" of its pixels, it is labeled as a potential corner



Rosten, Drummond. "Machine learning for high-speed corner detection." European conference on computer vision. Springer, Berlin, Heidelberg, 2006.
FAST Feature Detection, OpenCV code: https://docs.opencv.org/3.4/df/d0c/tutorial_py_fast.html

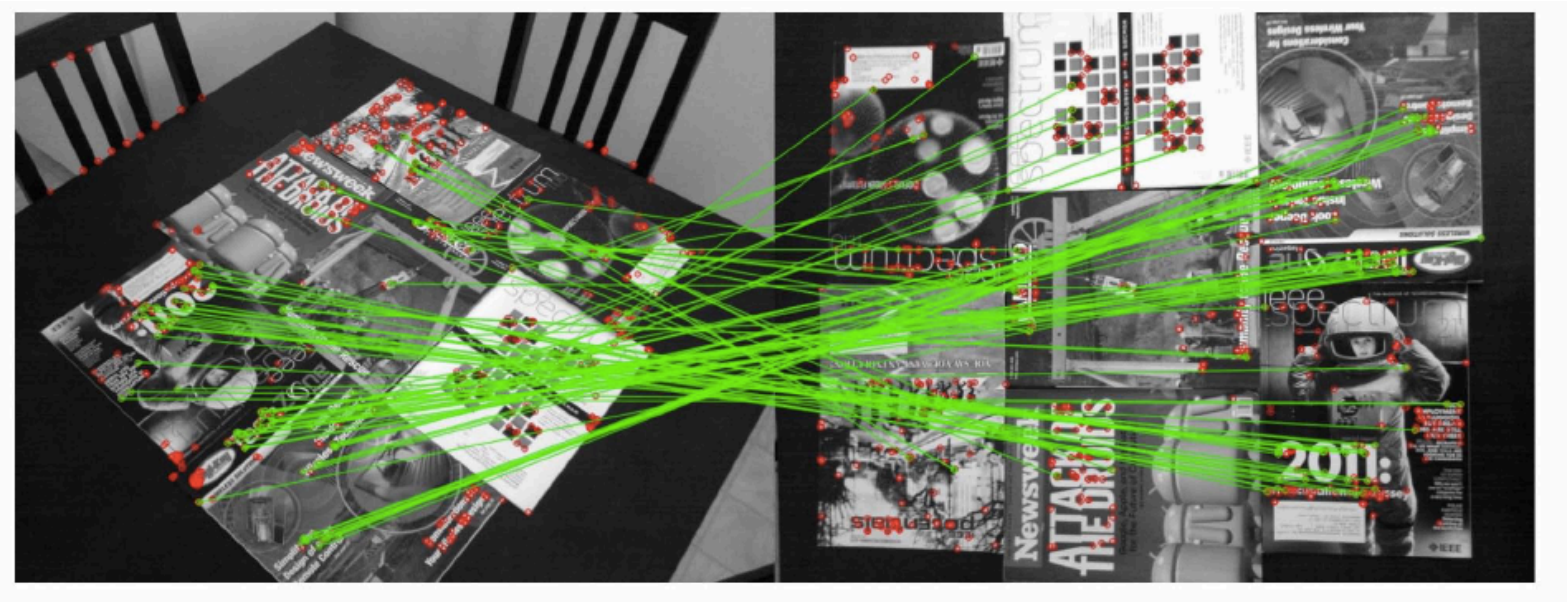
Front End: Data Association

- **BRIEF: (Binary Robust Independent Elementary Features)**
  - Uses binary strings as feature descriptors
  - Randomly samples pairs of pixels, and compares pixel intensity within each pair
-



Calondar et al. “BRIEF: Binary Robust Independent Elementary Features,” ECCV 2010.
BRIEF feature descriptor, OpenCV code: https://docs.opencv.org/3.4/dc/d7d/tutorial_py_brief.html

Front End: Data Association

- **ORB: (Oriented FAST and Rotated BRISK)**
  - Drawback of FAST — Not scale invariant, cannot record "orientation" of corner
  - Drawback of BRISK — Performs poorly with rotations
  - ORB — FAST + BRISK, with above issues addressed
- 

Green lines:
Correct matches

Red dots:
Incorrect matches

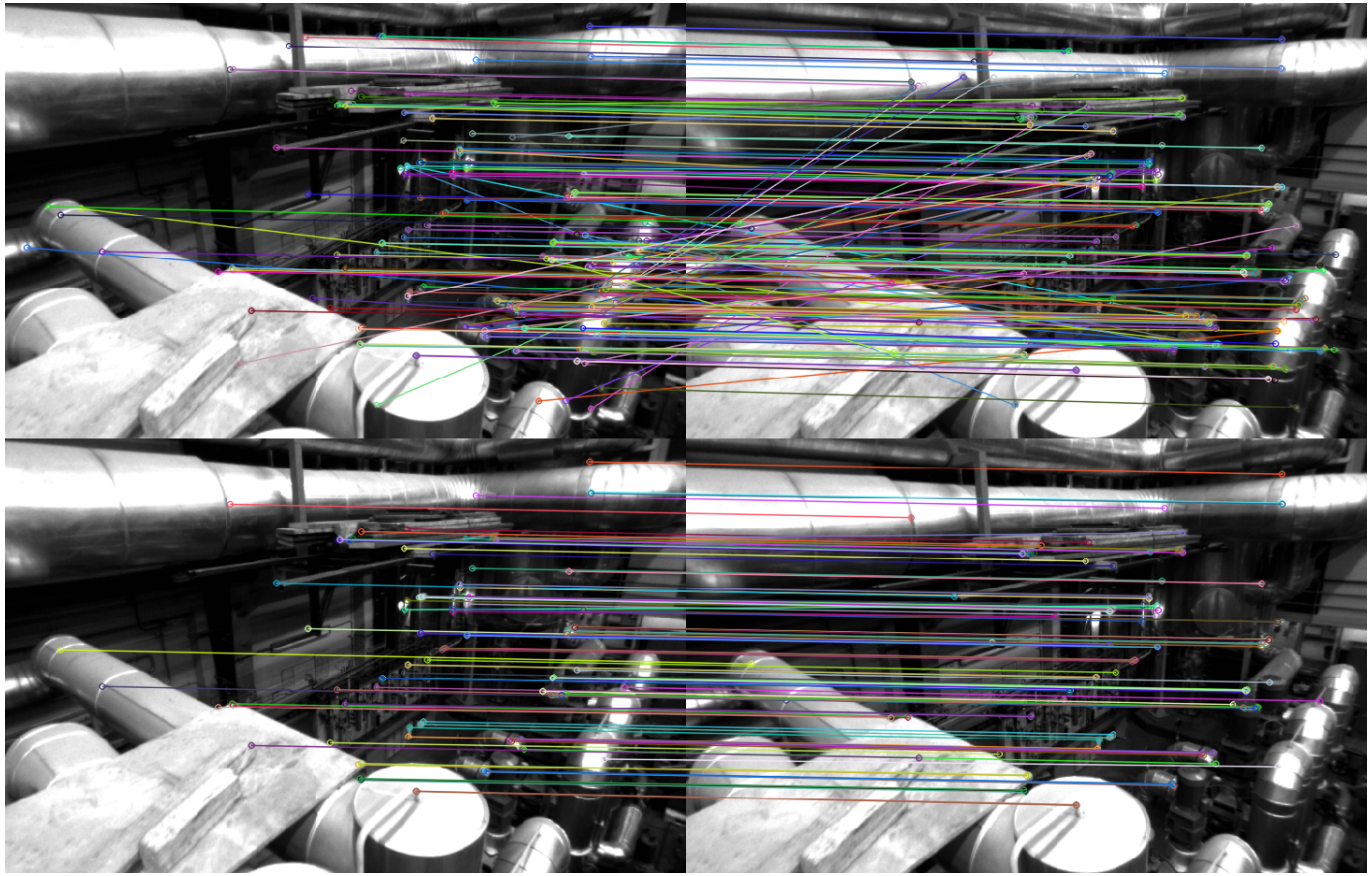
Rublee et al., "ORB: An efficient alternative to SIFT or SURF," ICCV 2011.
ORB, OpenCV code: https://docs.opencv.org/3.4/d1/d89/tutorial_py_orb.html

Front End: Outlier Rejection

- Nearest-neighbor feature matching: “local”, can be error-prone.
- Outlier feature matches should be rejected.
- **Method 1: RANSAC:** (Random Sampling and Consensus)
  - Estimate fundamental matrix, reject matches that violate epipolar constraints.
  - Estimate Perspective-N-Point solution, reject matches with high reprojection error.
-

Python implementation: https://en.wikipedia.org/wiki/Random_sample_consensus#Example_code

Front End: Outlier Rejection



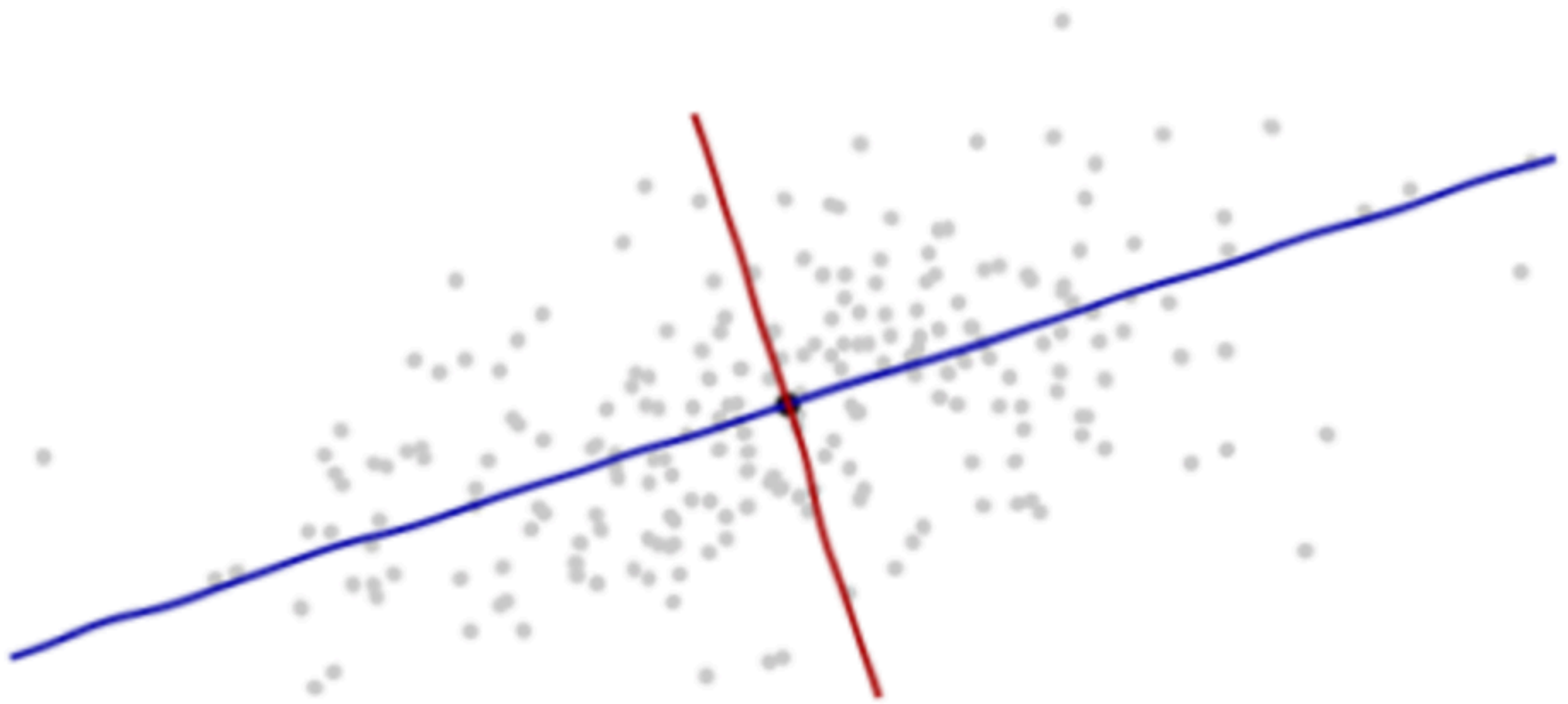
Top: Raw matches

Bottom: After outlier rejection using RANSAC

Image at time $t - 1$ **Image at time t**

Front End: Outlier Rejection

- Nearest-neighbor feature matching: “local”, can be error-prone.
- Outlier feature matches should be rejected.
- **Method 2: Mahalanobis distance test:** (Chi-squared rejection)
  - Accept a feature match only if the location of the new image feature is within 3 standard deviations of the expected location given the current best estimate.



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

Outline

- **Introduction, FAQs**
- **Front End**
  - Feature Extraction
  - Data Association, with Outlier Rejection
- **Back End**
  - Goal, Setup
  - Unifying Framework: 3 Steps
  - Aside: Kalman Filtering, Basics
  - Example: EKF SLAM
  - State-of-the-art Algorithms
  - Experiments
  - Loop Closures
- **Next Steps**
  - Active Perception: Dynamic, Semantic SLAM

Back End: Setup

- **Dynamics model:** $g : \mathbb{R}^{d_x} \to \mathbb{R}^{d_x}$
  - 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: $x_{t+1} - g(x_t)$
- **Example:**
  - Discrete time robot model with associated input u_t and IMU measurements.
  - (See Appendix for a simple, 2D example)

Back End: Setup

- **Measurement model:** $h : \mathbb{R}^{d_x} \times \mathbb{R}^{d_f} \to \mathbb{R}^{d_z}$
  - 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: $z_{t,j} - h(x_t, f_{t,j})$

Measurement Model

- **Example — Pinhole camera:**

$$x_t \leftarrow \text{ pose of camera at time } t$$

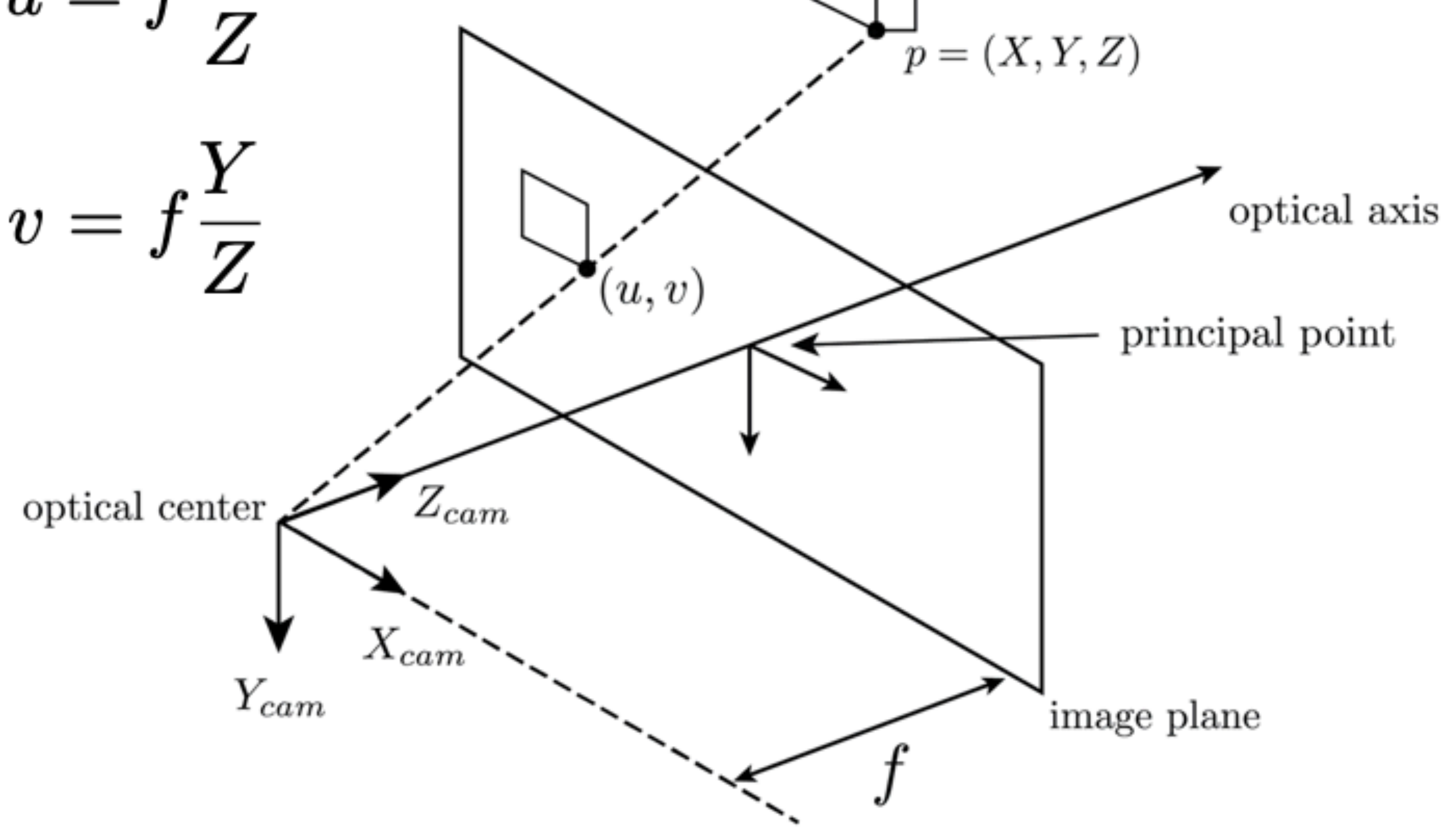
$$f_j = (f_j^x, f_j^y, f_j^z) \leftarrow \text{ location of feature } j$$

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

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

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

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



Back End: Setup

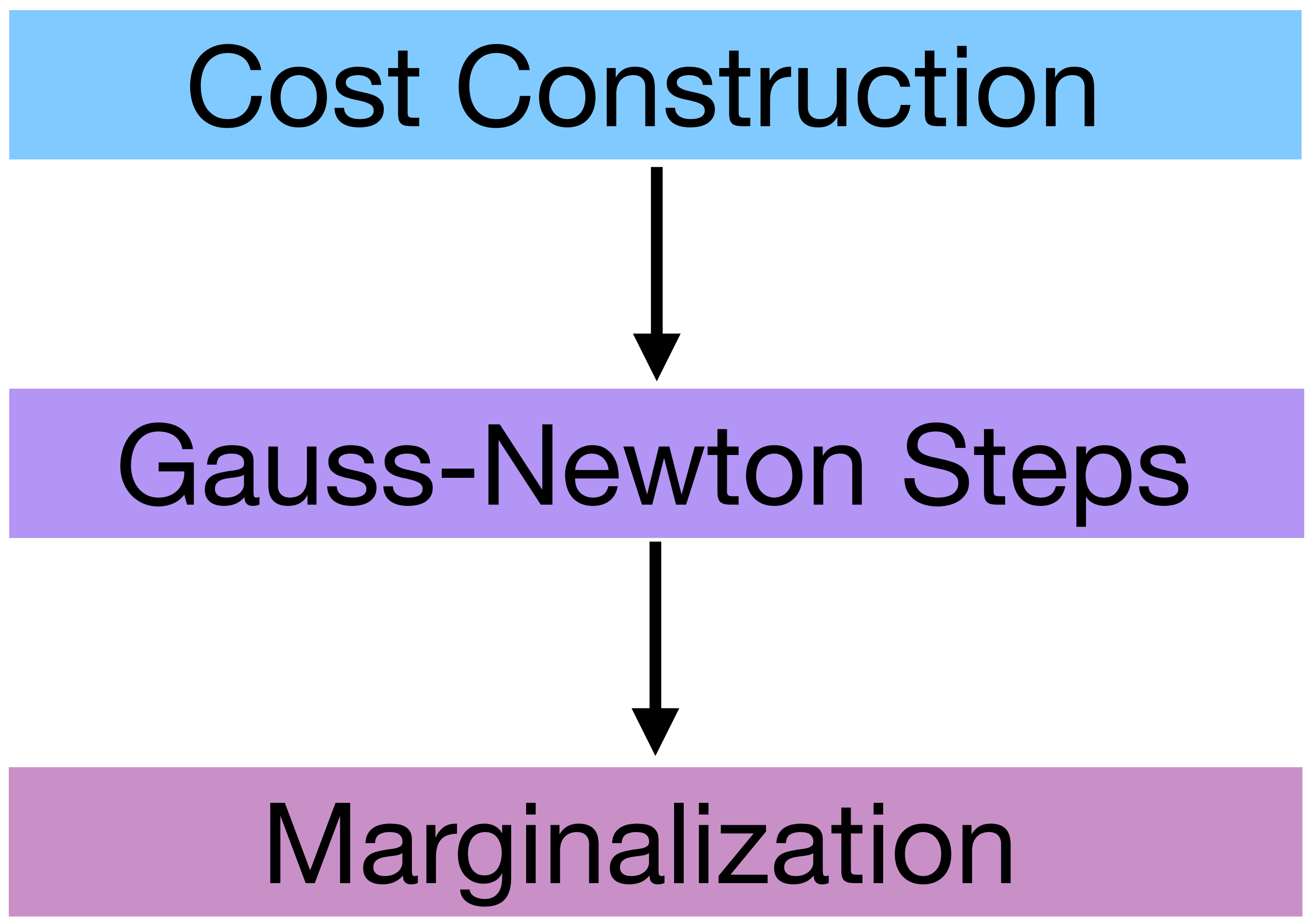
- **State Vector:** (Variables to estimate)
  - Camera pose, at time t: $x_t \in \mathbb{R}^{d_x}, \forall t \geq 0.$
  - Position of feature j at time t in global frame: $f_{t,j} \in \mathbb{R}^{d_f}, \forall t \geq 0, j \geq 1.$
  - Full state, at time t: $\tilde{x}_t \in \mathbb{R}^d, \forall t \geq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

    (The full state consists of the concatenation of multiple poses and features)
  -
- **State Vector 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: Unifying Framework

- **State-of-the-art SLAM Back End Algorithms:**
    - Involve different design choices, trades off computational speed and accuracy
    - Different algorithms are often described very differently in the literature
- **3 Main Modules in Back End:**



Saxena, **Chiu**, et al, "Simultaneous Localization and Mapping: Through the Lens of Nonlinear Optimization," ICRA / R-AL, 2022.

Back End: Unifying Framework

- **Cost Construction:**
  - **Q:** What variables and measurement data should we take into consideration?
  - Residuals — Error terms derived from dynamics and measurement models:

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

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

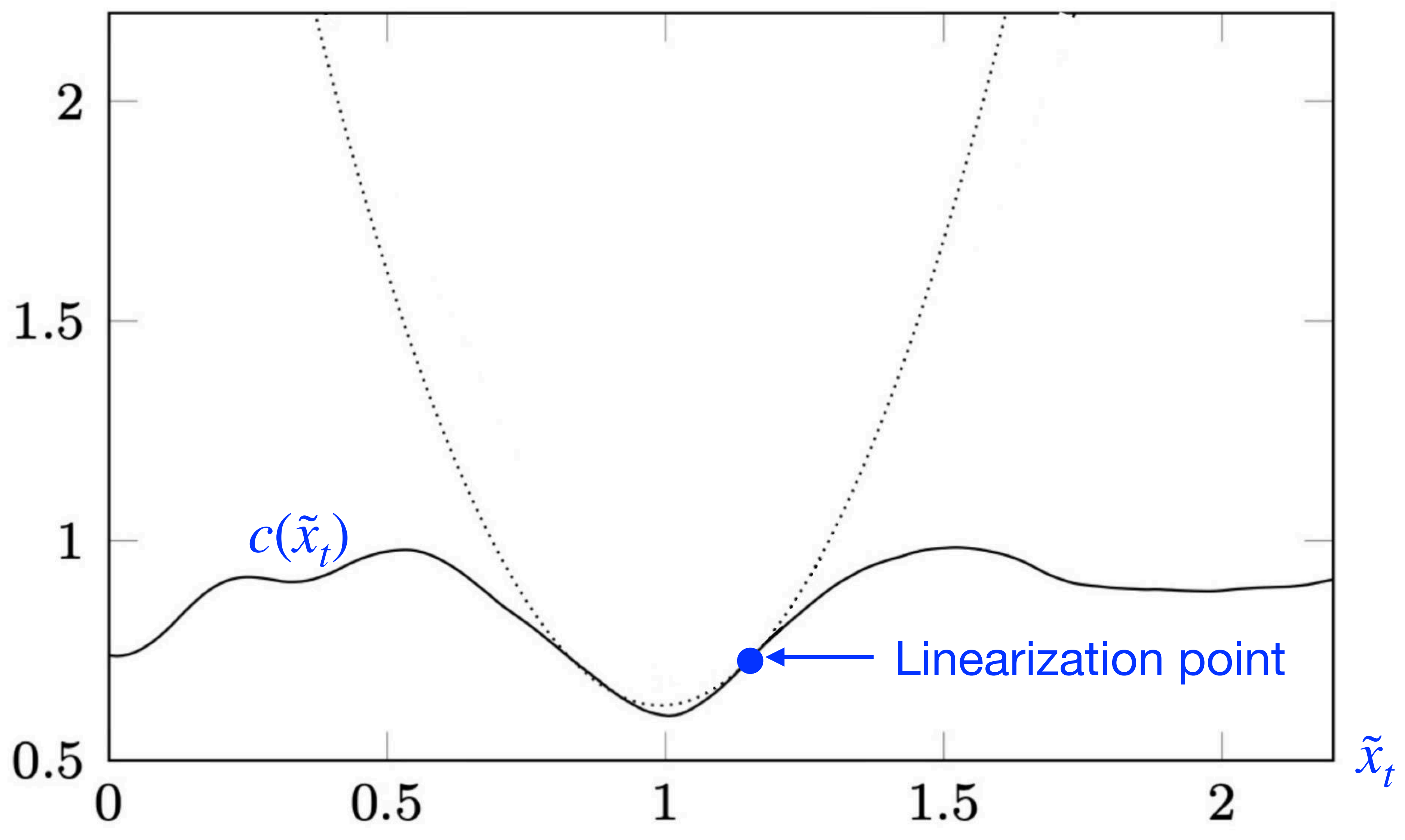
where $x_t :=$ poses, $f_j :=$ features, $z_{t,j} :=$ measurements.

  - **Idea** —If models and measurements are consistent, then all residuals are small.
  - Define — cost $c(\tilde{x}_t)$ = sum of weighted 2-norm squared of residuals.
  - **Goal—Find $\tilde{x}_t$ that minimizes $c(\tilde{x}_t)$.** (← Nonlinear least squares problem)

Saxena, **Chiu**, et al, "Simultaneous Localization and Mapping: Through the Lens of Nonlinear Optimization," ICRA / R-AL, 2022.

Back End: Unifying Framework

- **Gauss-Newton Update:**
  - **Q:** How do we compute a minimizer of the cost quickly and accurately?
  - Linearization Point — Current estimate of $\tilde{x}_t$, denoted μ_t
  - Approximate $c(\tilde{x}_t)$ about μ_t as a convex quadratic, and find the minimum.
  - Levenberg-Marquardt Update — L_2-Regularized Gauss-Newton Update



Saxena, **Chiu**, et al, "Simultaneous Localization and Mapping: Through the Lens of Nonlinear Optimization," ICRA / R-AL, 2022.

Photo credits: https://fgs-2019.sciencesconf.org/data/diehl.pdf

Back End: Unifying Framework

- **Marginalization:**
  - **Q:** What variables should we disregard to increase computation speed, and when?
  - Split full state into parts to keep, and parts to marginalize — $\tilde{x}_t = (\tilde{x}_{t,K}, \tilde{x}_{t,M})$.
  - **Goal** — Replace $c(\tilde{x}_t)$ with a cost function that depends only on $\tilde{x}_{t,K}$:

$$\min_{\tilde{x}_t} c(\tilde{x}_t) = \min_{\tilde{x}_{t,K}} \left(\underline{\min_{\tilde{x}_{t,M}} c(\tilde{x}_{t,K}, \tilde{x}_{t,M})} \right)$$

(1) Depends only on $\tilde{x}_{t,K}$,
(2) Computed via Gauss-Newton steps

Saxena, **Chiu**, et al, "Simultaneous Localization and Mapping: Through the Lens of Nonlinear Optimization," ICRA / R-AL, 2022.

Back End: Unifying Framework

- **Summary — Steps as Design Choices:**
  - **Cost Construction** — What variables and measurement data should we take into consideration?
  - **Gauss-Newton Update** — How do we compute a minimizer of the cost quickly and accurately?
  - **Marginalization** — What variables should we disregard to increase computation speed, and when?
- **This is reminiscent of how humans process information.**
  - The above steps provide a sound framework for representing scenes accurately and efficiently.

Saxena, **Chiu**, et al, "Simultaneous Localization and Mapping: Through the Lens of Nonlinear Optimization," ICRA / R-AL, 2022.

Optimization on Manifolds

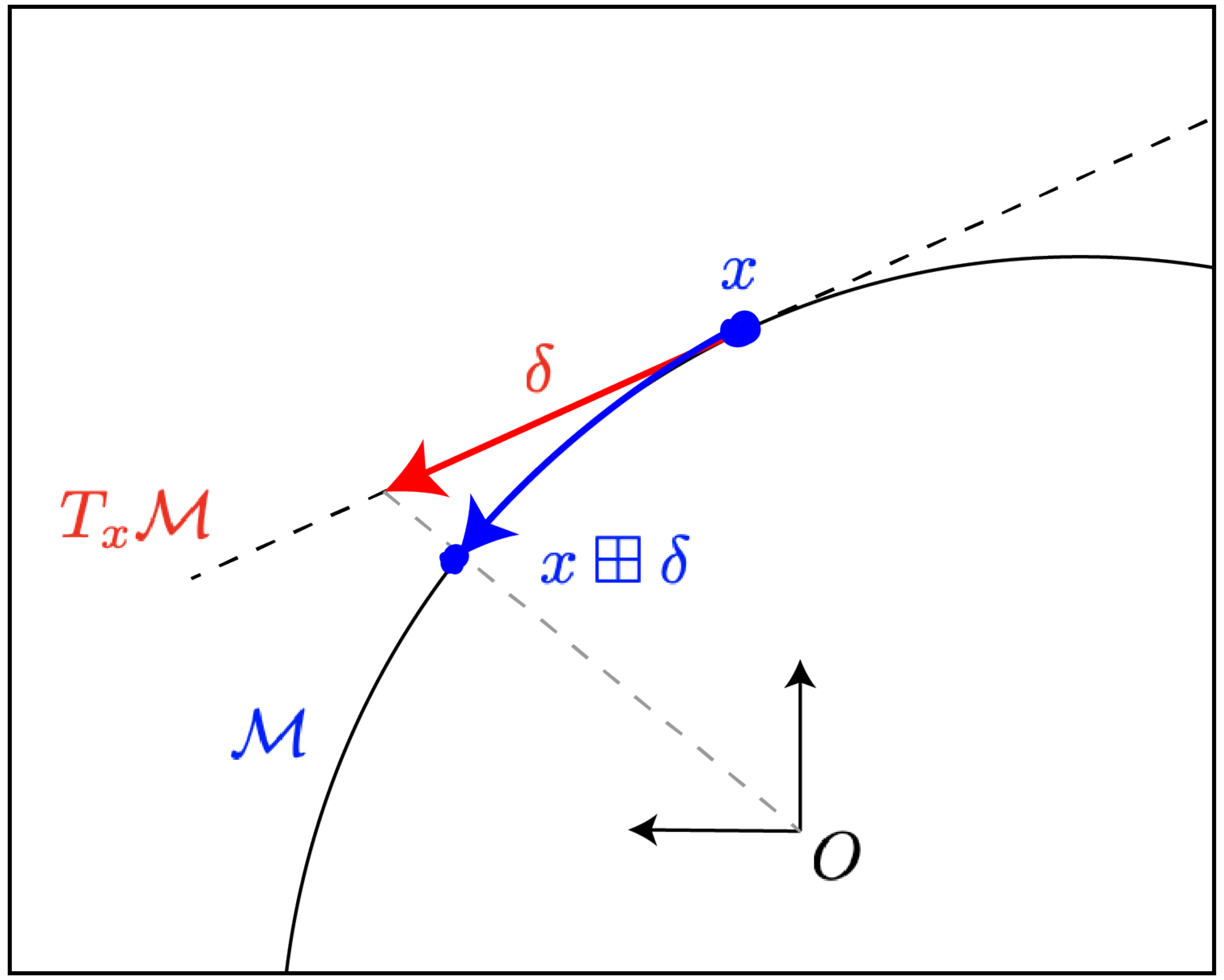
- So far — Taylor expansion on an objective function $f : \mathbb{R}^n \to \mathbb{R}^k$, with $\delta \in \mathbb{R}^n$.

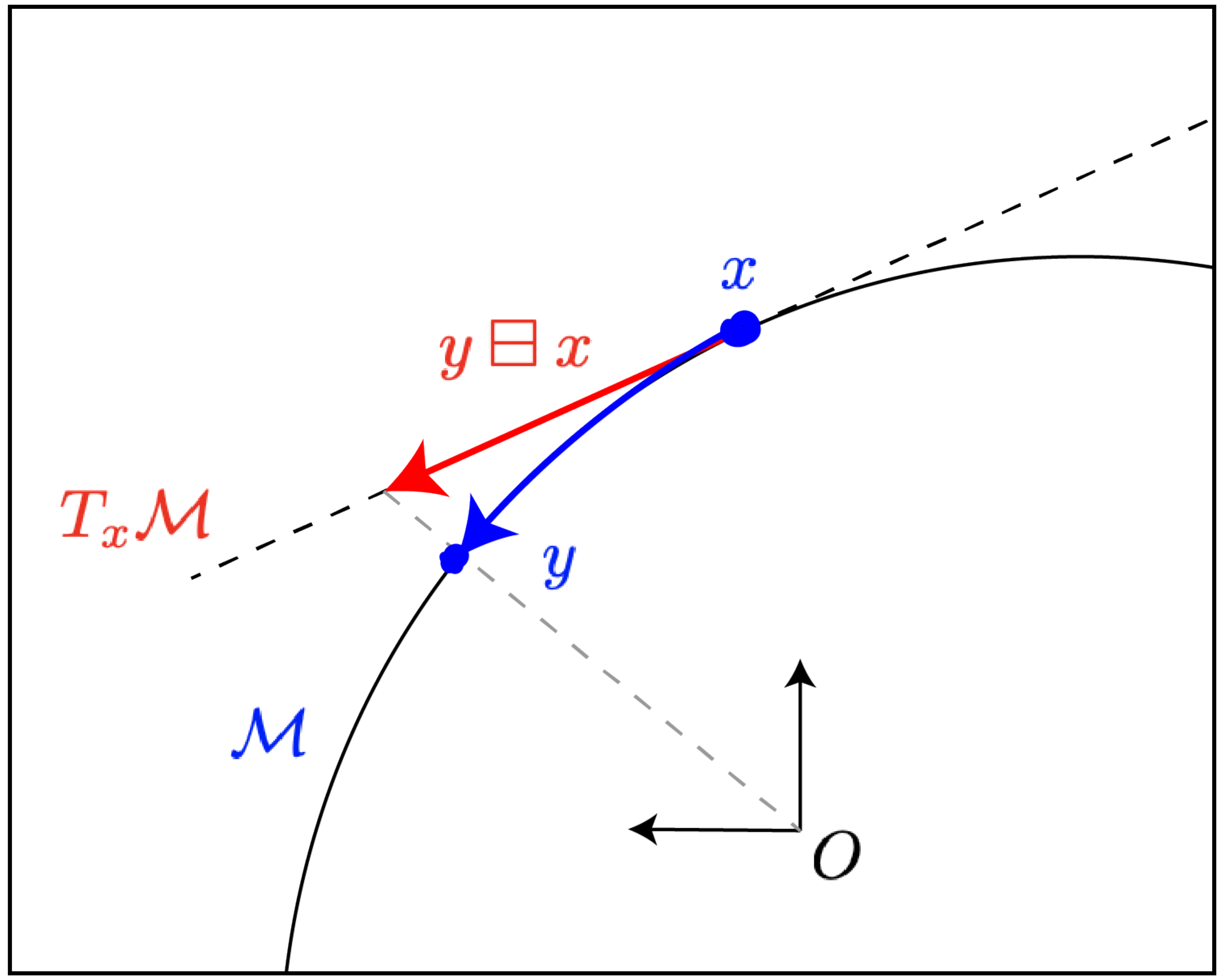
$$f(x+\delta) \approx f(x) + \frac{df}{dx}(x) \cdot \delta$$

- Problem — In SLAM, "x" contains *poses* $(R, T) \in SE(n)$, where R: rotation matrix T: translation vector. How would one define "$R + \delta$"?
- Solution — We require a notion of "differential change" other than simple addition.

Optimization on Manifolds

- **Setup** — $f : M \to \mathbb{R}^k$, where M: smooth manifold of dimension n
- **Idea** — Use the tangent space $T_x M$ at a given point $x \in M$, to locally parameterize small changes from that point.
- 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

- **Example** — SO(3) and SE(3) in 3D SLAM
- Use the exponential and logarithmic maps to define $\boxplus : M \times \mathbb{R}^n \rightarrow M$ and $\boxminus : M \times M \rightarrow \mathbb{R}^n$ by:

$$
\begin{aligned}
x \boxplus \delta &= x \cdot \exp(\delta) \\
x \boxminus y &= \log(y^{-1} x)
\end{aligned}
$$

- Covariances — Redefined to characterize the tangent-space deviation from the mean.

Optimization on Manifolds

- **Taylor expansion, on Manifolds:**

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

where $f : M \to \mathbb{R}^k$, $\delta \in \mathbb{R}^n$, and $J : \mathbb{R}^n \to \mathbb{R}^k$:

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

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

Optimization on Manifolds

- **New (correct) update rules on manifolds:**
  - Gauss-Newton descent:

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

  - 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

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

Back End: EKF SLAM (Upcoming Slides)

- **Main Focus — The simplest example of SLAM:**
  - Extended Kalman Filter — Recursive, filtering-based algorithm for updating a full state vector, using dynamics and measurement models.
- **Brief Outline:**
  - Introduce the Kalman Filter (KF) for linear systems
  - Introduce the 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

Saxena, **Chiu**, et al, "Simultaneous Localization and Mapping: Through the Lens of Nonlinear Optimization," ICRA / R-AL, 2022.

Outline

- **Introduction, FAQs**
- **Front End**
  - Feature Extraction
  - Data Association, with Outlier Rejection
- **Back End**
  - Goal, Setup
  - Unifying Framework: 3 Steps
  - Aside: Kalman Filtering, Basics
  - Example: EKF SLAM
  - State-of-the-art Algorithms
  - Experiments
  - Loop Closures
- **Next Steps**
  - Active Perception: Dynamic, Semantic SLAM

Aside: Kalman Filtering

- **Kalman Filter (KF) for a Linear System:**
  - **Linear System with additive Gaussian noise:**

$$\begin{aligned} x_0 &\sim N(\mathbf{0}_n, \Sigma_0), \\ x_{t+1} &= A_t x_t + w_t, \qquad && w_t \sim N(\mathbf{0}_n, \Sigma_w) \\ y_{t+1} &= C_t x_t + v_t, \qquad && v_t \sim N(\mathbf{0}_{n_o}, \Sigma_v) \end{aligned}$$

  - where $x_t, w_t \in \mathbb{R}^n$, $u_t \in \mathbb{R}^{n_i}$, $y_t \in \mathbb{R}^{n_o}$.
  - **Conditional mean and covariance of state x_t:**

$$\begin{aligned} \mu_{t'|t} &:= \mathbb{E}[x_{t'}|y_{1:t}], \\ \Sigma_{t'|t} &:= Cov[x_{t'}, x_{t'}|y_{1:t}] = \mathbb{E}[(x_{t'} - \mu_{t'|t})(x_{t'} - \mu_{t'|t})^\top | y_{1:t}] \end{aligned}$$

  - where $y_{1:t} = (y_1, \cdots, y_t)$.

Aside: Kalman Filtering

- **Kalman Filter (KF) for a Linear System:**
  - **2 Main Steps**
  - **(A) Propagation:** $(x_t \mid y_{1:t} \rightarrow x_{t+1} \mid y_{1:t})$
    - Given: (1) An estimate of the current state (x_t) conditioned on all observations up to the present ($y_{1:t}$),
    - Given: (2) The dynamics model $x_{t+1} = A_t x_t + w_t$
    - Find: An estimate of the next state x_{t+1} conditioned on the current observation set $y_{1:t}$
  -

Aside: Kalman Filtering

- **Kalman Filter (KF) for a Linear System:**
  - **2 Main Steps**
  - **(B) Update:** $(x_{t+1} | y_{1:t} \to x_{t+1} | y_{1:t+1})$
    - Given: (1) An estimate of the new state (x_{t+1}) conditioned on all observations up to the recent past ($y_{1:t}$),
    - Given: (2) A new observation y_{t+1}
    - Given: (3) The observation model $y_{t+1} = C_{t+1}x_{t+1} + w_{t+1}$
    - Find: An estimate of the new state x_{t+1} conditioned on the new observation set $(y_{1:t}, y_{t+1}) = y_{1:t+1}$.

Aside: Kalman Filtering

- **Kalman Filter (KF) for a Linear System:**
  - **Propagation Step:** $(x_t \mid y_{1:t} \rightarrow x_{t+1} \mid y_{1:t})$
  -

$$
\begin{aligned}
\mu_{t+1|t} &= \mathbb{E}[x_{t+1}|y_{1:t}] = \mathbb{E}[A_t x_t + w_t|y_{1:t}] = A_t\,\mathbb{E}[x_t|y_{1:t}] + 0 \\
&= A_t \mu_{t|t}, \\
\Sigma_{t+1|t} &= \mathbb{E}\Big[(x_{t+1} - \mu_{t+1|t})(x_{t+1} - \mu_{t+1|t})^\top |y_{1:t}\Big] \\
&= \mathbb{E}\Big[\big(A_t(x_t - \mu_{t|t}) + w_t\big)\big(A_t(x_t - \mu_{t|t}) + w_t\big)^\top\Big] \\
&= A_t \mathbb{E}\Big[(x_t - \mu_{t|t})(x_t - \mu_{t|t})^\top\Big] A_t^\top + 0 + \mathbb{E}[w_t w_t^\top] \\
&= A_t \Sigma_{t|t} A_t^\top + \Sigma_w
\end{aligned}
$$

Thrun, Burgard, Fox. “Probabilistic Robotics,” MIT Press, 2005.

Aside: Kalman Filtering

- **Kalman Filter (KF) for a Linear System:**
  - **Update Step:** $(x_{t+1} \mid y_{1:t} \rightarrow x_{t+1} \mid y_{1:t+1})$
  - Idea:
    - (1) $y_{1:t}$ is "old data" on which estimates of x_{t+1}, y_{t+1} are based
    - (2) So, characterize $x_{t+1} \mid y_{1:t}$ and $y_{t+1} \mid y_{1:t}$
    - (3) Then, condition $x_{t+1} \mid y_{1:t}$ on $y_{t+1} \mid y_{1:t}$ to characterize:

$$(x_{t+1} \mid y_{1:t}) \mid (y_{t+1} \mid y_{1:t}) \longleftrightarrow x_{t+1} \mid y_{1:t+1}$$

Thrun, Burgard, Fox. "Probabilistic Robotics," MIT Press, 2005.

Aside: Kalman Filtering

- **Kalman Filter (KF) for a Linear System:**
  - **Update Step:** $(x_{t+1} \mid y_{1:t} \to x_{t+1} \mid y_{1:t+1})$
  - Idea — (2) Characterize $x_{t+1} \mid y_{1:t}$ and $y_{t+1} \mid y_{1:t}$

$$
\begin{aligned}
Cov[x_{t+1}, y_{t+1}|y_{1:t}] &= \mathbb{E}[(x_{t+1} - \mu_{t+1|t})(y_{t+1} - C_{t+1}\mu_{t+1|t})^\top|y_{1:t}] \\
&= \mathbb{E}[(x_{t+1} - \mu_{t+1|t})\big((x_{t+1} - \mu_{t+1|t})^\top C_{t+1}^\top + v_{t+1}^\top\big)|y_{1:t}] \\
&= \mathbb{E}[(x_{t+1} - \mu_{t+1|t})(x_{t+1} - \mu_{t+1|t})^\top|y_{1:t}] \cdot C_{t+1}^\top \\
&= \Sigma_{t+1|t}C_{t+1}^\top, \\
Cov[y_{t+1}, y_{t+1}|y_{1:t}] &= \mathbb{E}\big[(y_{t+1} - C_{t+1}\mu_{t+1|t})(y_{t+1} - C_{t+1}\mu_{t+1|t})^\top|y_{1:t}\big] \\
&= \mathbb{E}\big[\big(C_{t+1}(x_{t+1} - \mu_{t+1|t}) + v_{t+1}\big)\big((x_{t+1} - \mu_{t+1|t})^\top C_{t+1}^\top + v_{t+1}^\top\big)|y_{1:t}\big] \\
&= C_{t+1} \cdot \mathbb{E}[(x_{t+1} - \mu_{t+1|t})(x_{t+1} - \mu_{t+1|t})^\top|y_{1:t}] \cdot C_{t+1}^\top + \mathbb{E}[v_{t+1}v_{t+1}^\top] \\
&= C_{t+1}\Sigma_{t+1|t}C_{t+1}^\top + \Sigma_v.
\end{aligned}
$$

Thrun, Burgard, Fox. “Probabilistic Robotics,” MIT Press, 2005.

Aside: Kalman Filtering

- **Kalman Filter (KF) for a Linear System:**
  - **Update Step:** $(x_{t+1} \mid y_{1:t} \rightarrow x_{t+1} \mid y_{1:t+1})$
  - Idea — (3) Condition $x_{t+1} \mid y_{1:t}$ on $y_{t+1} \mid y_{1:t}$ to characterize $x_{t+1} \mid y_{1:t+1}$
  - **Lemma** — Given Gaussian random variables X, Y with joint distribution:

$$(X, Y) \sim N\left(\begin{bmatrix}\mu_X \\ \mu_Y\end{bmatrix}, \begin{bmatrix}\Sigma_X & \Sigma_{XY} \\ \Sigma_{XY}^\top & \Sigma_Y\end{bmatrix}\right)$$

    - The conditional distribution $X \mid Y$ is likewise Gaussian, with:

$$X|Y \sim N\big(\mu_X + \Sigma_{XY}\Sigma_{YY}^{-1}(Y - \mu_Y), \Sigma_{XX} - \Sigma_{XY}\Sigma_{YY}^{-1}\Sigma_{XY}^\top\big)$$

    - We wish to apply this lemma, with $X = x_{t+1} \mid y_{1:t}$ and $Y = y_{t+1} \mid y_{1:t}$.

Thrun, Burgard, Fox. “Probabilistic Robotics,” MIT Press, 2005.

Aside: Kalman Filtering

- **Kalman Filter (KF) for a Linear System:**
  - **Update Step:** $(x_{t+1} \,|\, y_{1:t} \rightarrow x_{t+1} \,|\, y_{1:t+1})$
  - Idea — (3) Condition $x_{t+1} \,|\, y_{1:t}$ on $y_{t+1} \,|\, y_{1:t}$ to characterize $x_{t+1} \,|\, y_{1:t+1}$
  - Applying the lemma gives us:

$$\boxed{\begin{aligned}
\mu_{t+1|t+1} &= \mathbb{E}[x_{t+1}|y_{t+1}] \\
&= \mu_{t+1|t} + \Sigma_{t+1|t} C_{t+1}^{\top} \left(C_{t+1} \Sigma_{t+1|t} C_{t+1}^{\top} + \Sigma_v\right)^{-1} (y_{t+1} - C_{t+1} \mu_{t+1|t}), \\
\bar{\Sigma}_{t+1|t+1} &= Cov[x_{t+1}, x_{t+1}|y_{t+1}] \\
&= \Sigma_{t+1|t} - \Sigma_{t+1|t} C_{t+1}^{\top} \left(C_{t+1} \Sigma_{t+1|t} C_{t+1}^{\top} + \Sigma_v\right)^{-1} C_{t+1} \Sigma_{t+1|t}
\end{aligned}}$$

Thrun, Burgard, Fox. “Probabilistic Robotics,” MIT Press, 2005.

Aside: Kalman Filtering

- **Kalman Filter (KF) for a Linear System:**
  - **Summary:**
  - **Propagation Step:**

$$\boxed{\begin{aligned} \mu_{t+1|t} &= A_t \mu_{t|t}, \\ \Sigma_{t+1|t} &= A_t \Sigma_{t|t} A_t^\top + \Sigma_w \end{aligned}}$$

  - **Update Step:**

$$\boxed{\begin{aligned} \mu_{t+1|t+1} &= \mu_{t+1|t} + \Sigma_{t+1|t} C_{t+1}^\top \left(C_{t+1} \Sigma_{t+1|t} C_{t+1}^\top + \Sigma_v \right)^{-1} (y_{t+1} - C_{t+1} \mu_{t+1|t}), \\ \bar{\Sigma}_{t+1|t+1} &= \Sigma_{t+1|t} - \Sigma_{t+1|t} C_{t+1}^\top \left(C_{t+1} \Sigma_{t+1|t} C_{t+1}^\top + \Sigma_v \right)^{-1} C_{t+1} \Sigma_{t+1|t} \end{aligned}}$$

Thrun, Burgard, Fox. "Probabilistic Robotics," MIT Press, 2005.

Aside: Kalman Filtering

- **Extended Kalman Filter (EKF) for a Nonlinear System:**
  - Instead of the linear system we considered before:

$$
\begin{aligned}
x_0 &\sim N(\mathbf{0}_n, \Sigma_0), \\
x_{t+1} &= A_t x_t + w_t, && w_t \sim N(\mathbf{0}_n, \Sigma_w) \\
y_t &= C_t x_t + v_t, && v_t \sim N(\mathbf{0}_{n_o}, \Sigma_v)
\end{aligned}
$$

  - Let us consider the following *nonlinear* system:

$$
\begin{aligned}
x_0 &\sim N(\mathbf{0}_n, \Sigma_0), \\
x_{t+1} &= g_t(x_t) + w_t, && w_t \sim N(\mathbf{0}_n, \Sigma_w) \\
y_t &= h_t(x_t) + v_t, && v_t \sim N(\mathbf{0}_{n_o}, \Sigma_v)
\end{aligned}
$$

  - where $g : \mathbb{R}^n \rightarrow \mathbb{R}^n$ and $h : \mathbb{R}^n \rightarrow \mathbb{R}^{n_o}$ are nonlinear maps.

Thrun, Burgard, Fox. “Probabilistic Robotics,” MIT Press, 2005.

Aside: Kalman Filtering

- **Extended Kalman Filter (EKF) for a Nonlinear System:**
  - We wish to apply Kalman filtering-type techniques to the nonlinear system
  - Apply Jacobian linearization to approximate the nonlinear system as:

$$
\begin{aligned}
x_0 &\sim N(\mathbf{0}_n, \Sigma_0), \\
x_{t+1} &\approx g_t(\mu_{t|t}) + G_t(x_t - \mu_{t|t}) + w_t, \qquad && w_t \sim N(\mathbf{0}_n, \Sigma_w) \\
y_t &\approx h_t(\mu_{t|t}) + H_t(x_t - \mu_{t|t}) + v_t, \qquad && v_t \sim N(\mathbf{0}_{n_o}, \Sigma_v)
\end{aligned}
$$

  - where $\mu_{t'|t} := \mathbb{E}[x_{t'} \,|\, y_{1:t}]$, as before.
  - Then, apply the Kalman Filtering equations.
    - Means — Use true nonlinear maps g_t, h_t.
    - Covariances — Use linearization G_t, H_t.

Thrun, Burgard, Fox. “Probabilistic Robotics,” MIT Press, 2005.

Aside: Kalman Filtering

- **Extended Kalman Filter (EKF) for a Nonlinear System:**
  - **Summary:**
  - **Propagation Step:**

$$\boxed{\begin{aligned} \mu_{t+1|t} &= g_t(\mu_{t|t}), \\ \Sigma_{t+1|t} &= G_t \Sigma_{t|t} G_t^\top + \Sigma_w \end{aligned}}$$

  - **Update Step:**

$$\boxed{\begin{aligned} \mu_{t+1|t+1} &= \mu_{t+1|t} + \Sigma_{t+1|t} H_{t+1}^\top \left(H_{t+1} \Sigma_{t+1|t} H_{t+1}^\top + \Sigma_v\right)^{-1} (y_{t+1} - h_{t+1}(\mu_{t+1|t})), \\ \bar{\Sigma}_{t+1|t+1} &= \Sigma_{t+1|t} - \Sigma_{t+1|t} H_{t+1}^\top \left(H_{t+1} \Sigma_{t+1|t} H_{t+1}^\top + \Sigma_v\right)^{-1} H_{t+1} \Sigma_{t+1|t} \end{aligned}}$$

Thrun, Burgard, Fox. “Probabilistic Robotics,” MIT Press, 2005.

Outline

- **Introduction, FAQs**
- **Front End**
  - Feature Extraction
  - Data Association, with Outlier Rejection
- **Back End**
  - Goal, Setup
  - Unifying Framework: 3 Steps
  - Aside: Kalman Filtering, Basics
  - Example: EKF SLAM
  - State-of-the-art Algorithms
  - Experiments
  - Loop Closures
- **Next Steps**
  - Active Perception: Dynamic, Semantic SLAM

Back End: EKF SLAM

- **EKF SLAM, Setup:**
  - State Vector $\tilde{x}_t$—In Extended Kalman Filter (EKF) SLAM: Most current pose, and all features ever detected (e.g., p features):

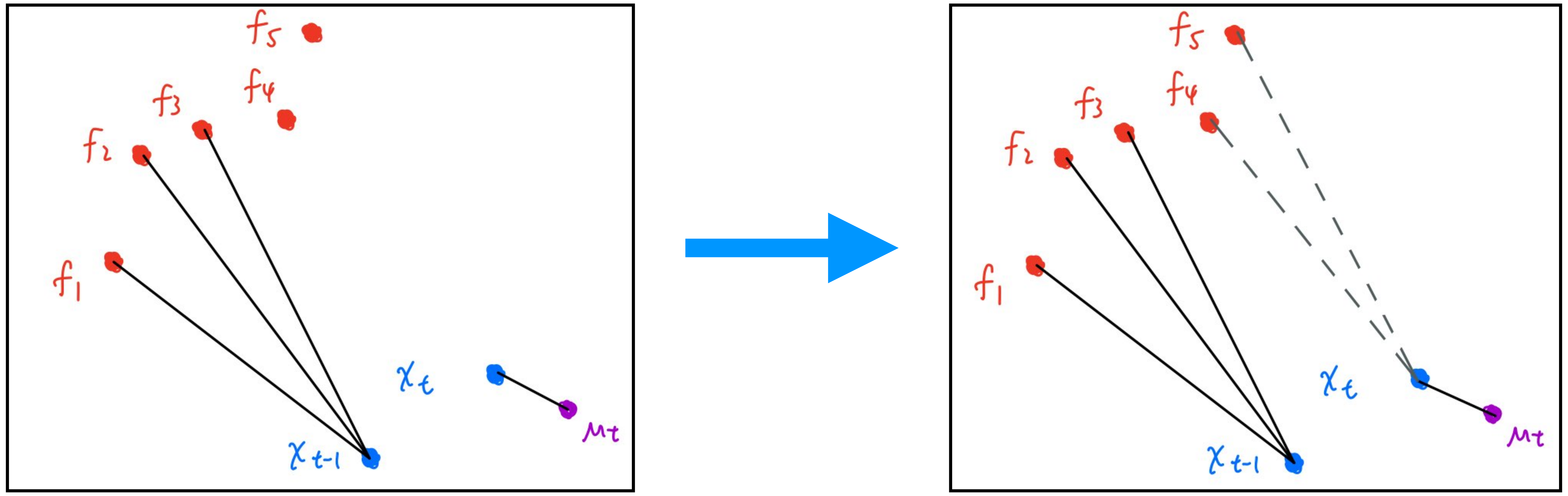
$$\tilde{x}_t = (x_t, f_{t,1}, \cdots, f_{t,p}) \in \mathbb{R}^{d_x + p d_f} := \mathbb{R}^d$$

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

Saxena, **Chiu**, et al, "Simultaneous Localization and Mapping: Through the Lens of Nonlinear Optimization," ICRA / R-AL, 2022.

Back End: EKF SLAM

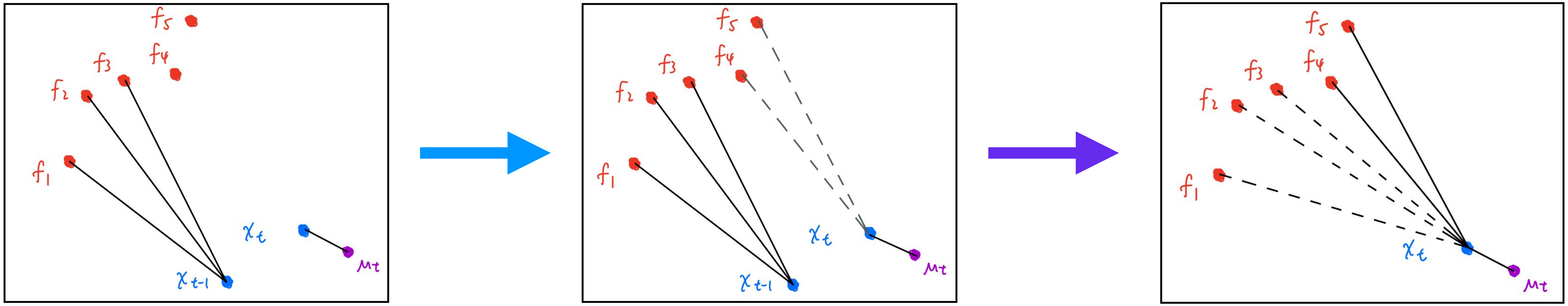
- **Step 1 — Feature Augmentation:**
  - Augment $\tilde{x}_t$ with position estimates of newly detected features



Saxena, **Chiu**, et al, "Simultaneous Localization and Mapping: Through the Lens of Nonlinear Optimization," ICRA / R-AL, 2022.

Back End: EKF SLAM

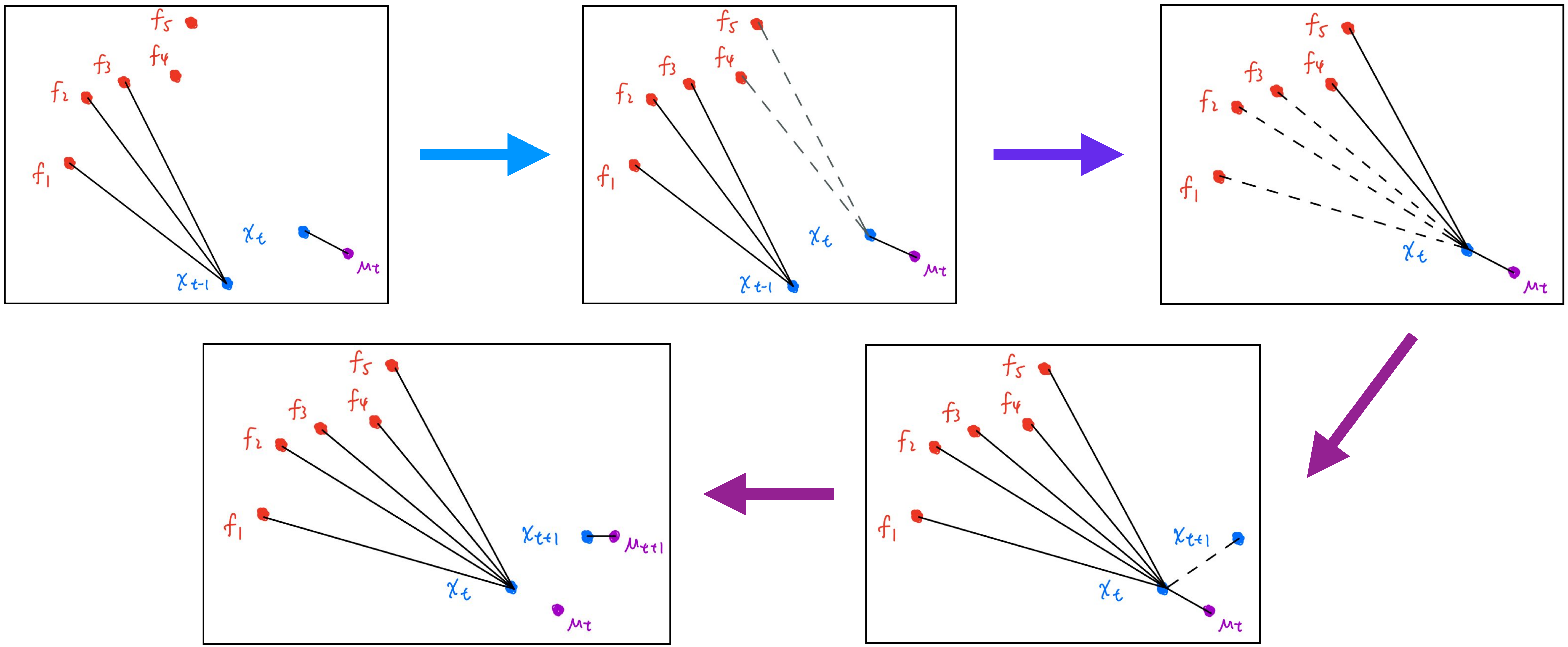
- **Step 2 — Feature Update:**
  - Update $\tilde{x}_t$ with position estimates of features already described in $\tilde{x}_t$



Saxena, **Chiu**, et al, “Simultaneous Localization and Mapping: Through the Lens of Nonlinear Optimization,” ICRA / R-AL, 2022.

Back End: EKF SLAM

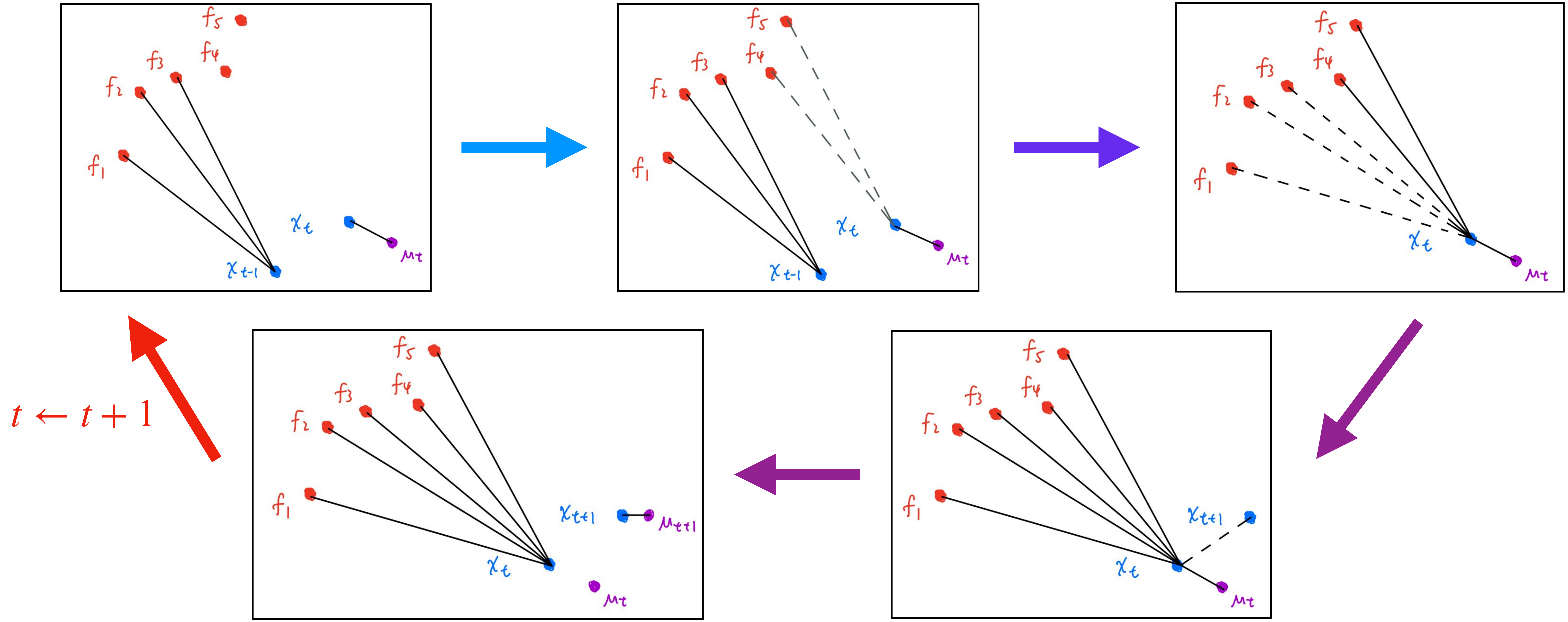
- **Step 3 — State Propagation:**
  - In $\tilde{x}_t$, replace the current pose x_t with the new pose x_{t+1}



Saxena, **Chiu**, et al, "Simultaneous Localization and Mapping: Through the Lens of Nonlinear Optimization," ICRA / R-AL, 2022.

Back End: EKF SLAM

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



Saxena, **Chiu**, et al, "Simultaneous Localization and Mapping: Through the Lens of Nonlinear Optimization," ICRA / R-AL, 2022.

Back End: EKF SLAM

- **EKF SLAM, Standard Formulation:**

For loop running through the finite time horizon

Feature Augmentation Step (Cost Construction)

Feature Update Step (Gauss-Newton)

State Propagation Step (Marginalization)

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 $t \leq T$.

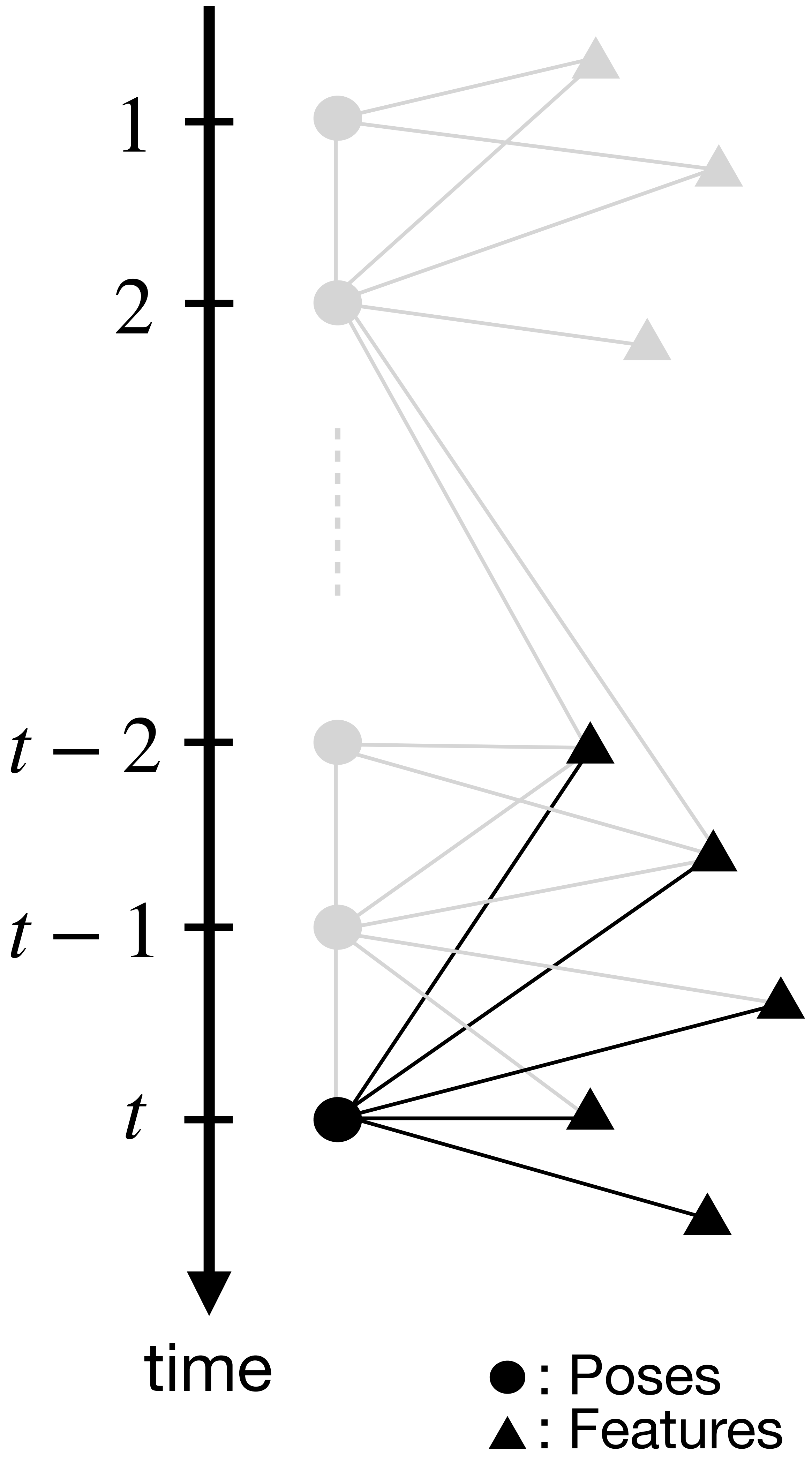
for $t = 0, \cdots, T$ **do**
    if detect new feature measurements $z_{t,p+1:p+p'} := (z_{t,p+1}, \cdots, z_{t,p+p'}) \in \mathbb{R}^{p'd_z}$ **then**
        $\boxed{\mu_t, \Sigma_t, p \leftarrow \text{Alg. 2, EKF feature augmentation } \left(\mu_t, \Sigma_t, p, z_{t,p+1:p+p'}, h(\cdot)\right)}$
    end
    $z_{t,1:p} := (z_{t,1}, \cdots, z_{t,p}) \in \mathbb{R}^{pd_z} \leftarrow$ New measurements of existing features.
    $\boxed{\overline{\mu_t}, \overline{\Sigma_t} \leftarrow \text{Alg. 3, EKF feature update } \left(\overline{\mu_t}, \overline{\Sigma_t}, z_{t,1:p}, h(\cdot)\right).}$
    if $t < T$ **then**
        $\boxed{\mu_{t+1}, \Sigma_{t+1} \leftarrow \text{Alg. 4, EKF state propagation } \left(\mu_t, \Sigma_t, g(\cdot)\right)}$
    end
end
return $\hat{x}_0, \cdots \hat{x}_T \in \mathbb{R}^{d_x}$.

Saxena, **Chiu**, et al, "Simultaneous Localization and Mapping: Through the Lens of Nonlinear Optimization," ICRA / R-AL, 2022.

Outline

- **Introduction, FAQs**
- **Front End**
  - Feature Extraction
  - Data Association, with Outlier Rejection
- **Back End**
  - Goal, Setup
  - Unifying Framework: 3 Steps
  - Aside: Kalman Filtering, Basics
  - Example: EKF SLAM
  - State-of-the-art Algorithms
  - Experiments
  - Loop Closures
- **Next Steps**
  - Active Perception: Dynamic, Semantic SLAM

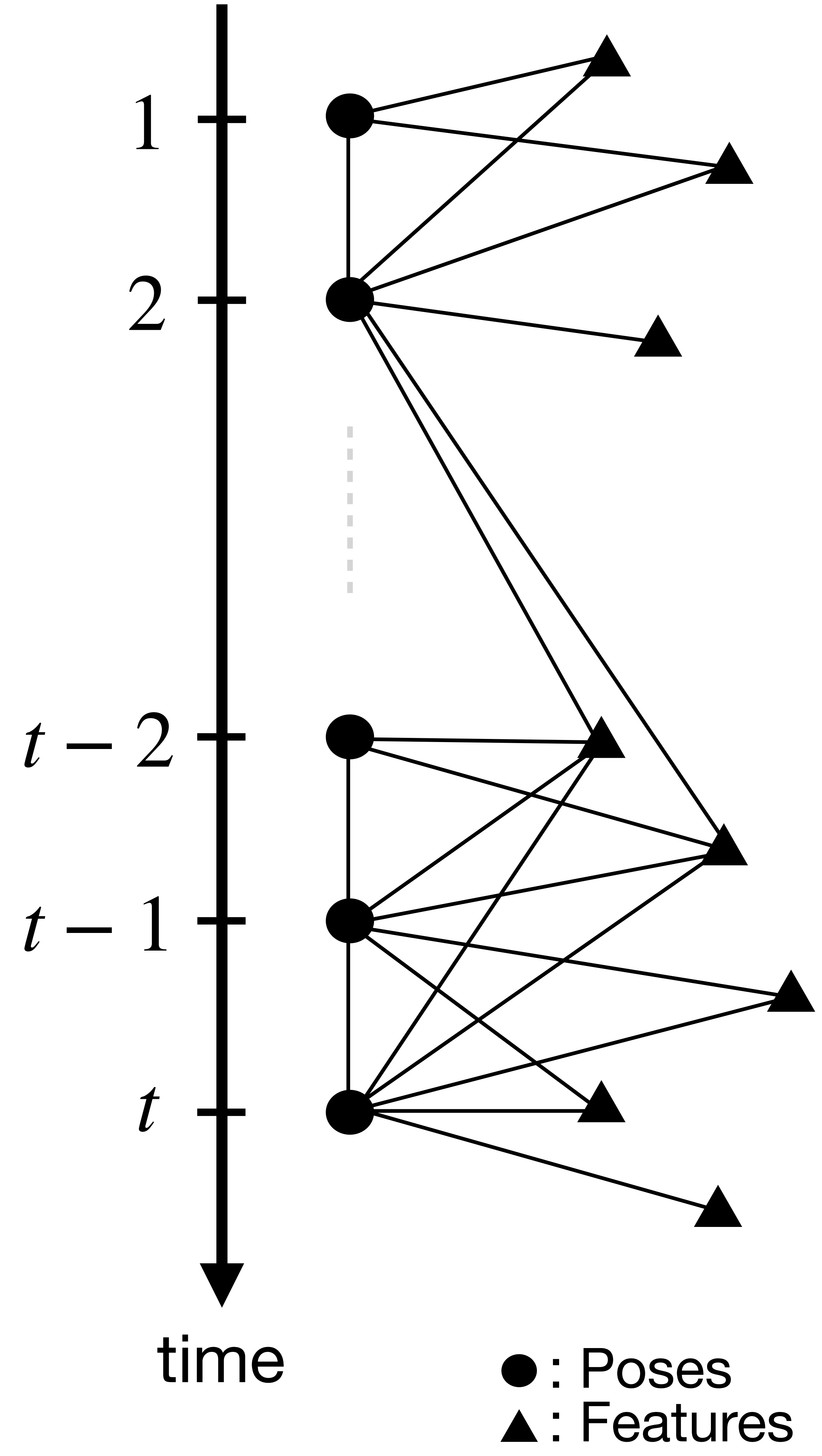
Back End: State-of-the-art Algorithms



Algorithm	Cost Construction	Gauss-Newton	Marginalization
EKF	Current pose + All current and past features	One	All past poses
iEKF	Current pose + All current and past features	Multiple	All past poses
MSCKF	Current IMU state + n ($\leq N_{\max}$) past poses, evenly spaced in time	One	All other poses
SWF	Current IMU state + n ($\leq N_{\max}$) most recent poses	One	All other poses
OKVis	Keyframe poses in sliding window + associated features	Multiple	Keyframe poses leaving sliding window + associated features
PGO	Current and all past poses, with no features	Multiple, until convergence	None
BA	Current and all past poses and features, with no pairwise pose costs	Multiple, until convergence	None

Saxena, **Chiu**, et al, “Simultaneous Localization and Mapping: Through the Lens of Nonlinear Optimization,” ICRA / R-AL, 2022.

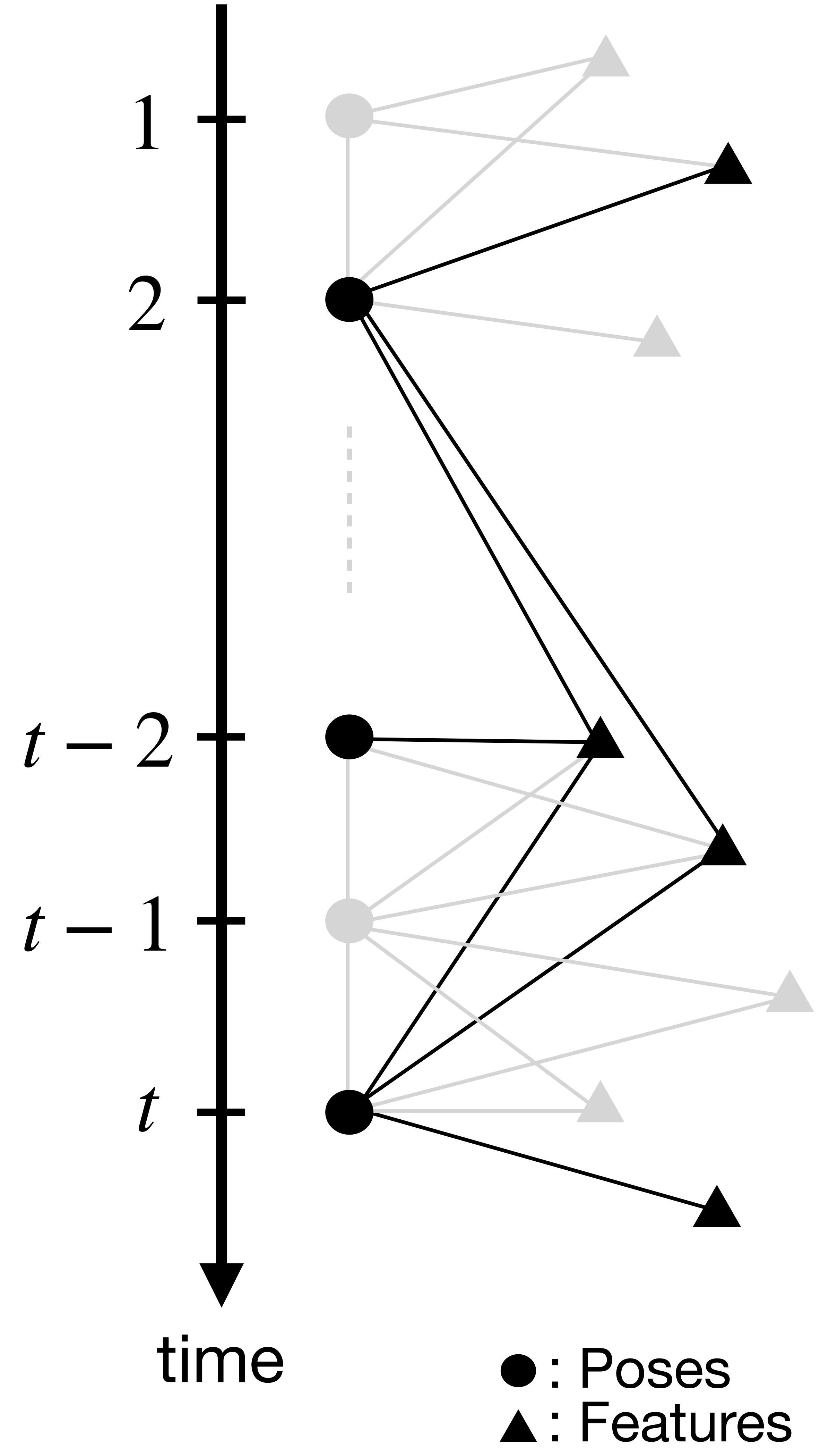
Back End: State-of-the-art Algorithms



Algorithm	Cost Construction	Gauss-Newton	Marginalization
EKF	Current pose + All current and past features	One	All past poses
iEKF	Current pose + All current and past features	Multiple	All past poses
MSCKF	Current IMU state + n ($\leq N_{\max}$) past poses, evenly spaced in time	One	All other poses
SWF	Current IMU state + n ($\leq N_{\max}$) most recent poses	One	All other poses
OKVis	Keyframe poses in sliding window + associated features	Multiple	Keyframe poses leaving sliding window + associated features
PGO	Current and all past poses, with no features	Multiple, until convergence	None
BA	Current and all past poses and features, with no pairwise pose costs	Multiple, until convergence	None

Saxena, **Chiu**, et al, "Simultaneous Localization and Mapping: Through the Lens of Nonlinear Optimization," ICRA / R-AL, 2022.

Back End: State-of-the-art Algorithms



Algorithm	Cost Construction	Gauss-Newton	Marginalization
EKF	Current pose + All current and past features	One	All past poses
iEKF	Current pose + All current and past features	Multiple	All past poses
MSCKF	Current IMU state + n ($\leq N_{\max}$) past poses, evenly spaced in time	One	All other poses
SWF	Current IMU state + n ($\leq N_{\max}$) most recent poses	One	All other poses
OKVis	Keyframe poses in sliding window + associated features	Multiple	Keyframe poses leaving sliding window + associated features
PGO	Current and all past poses, with no features	Multiple, until convergence	None
BA	Current and all past poses and features, with no pairwise pose costs	Multiple, until convergence	None

Saxena, **Chiu**, et al, "Simultaneous Localization and Mapping: Through the Lens of Nonlinear Optimization," ICRA / R-AL, 2022.

Back End: State-of-the-art Algorithms

Small number of states, aggressive marginalization

Medium number of states, some marginalization

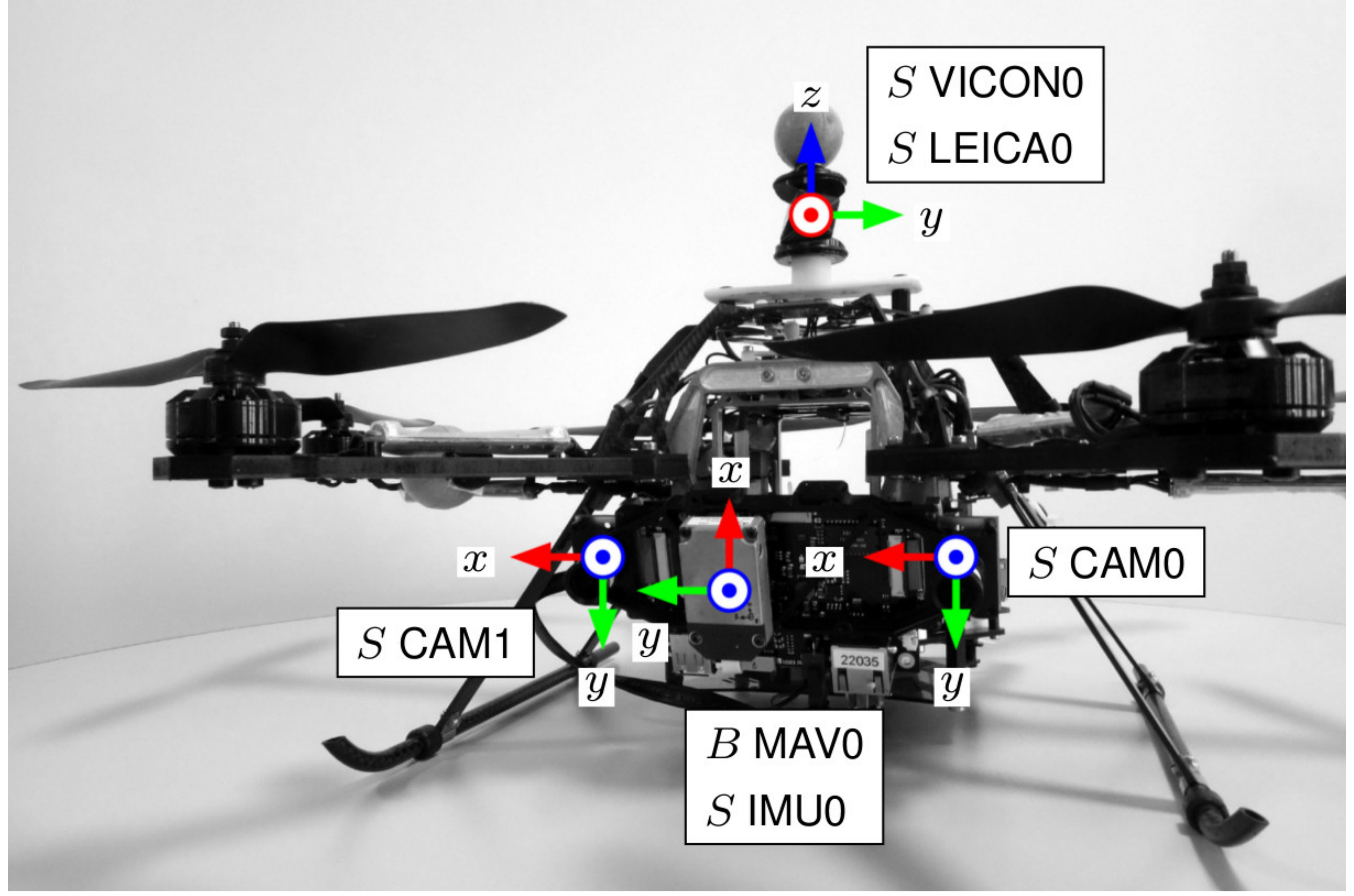
Large number of states, little or no marginalization

Algorithm	Cost Construction	Gauss-Newton	Marginalization
EKF	Current pose + All current and past features	One	All past poses
iEKF	Current pose + All current and past features	Multiple	All past poses
MSCKF	Current IMU state + n ($\leq N_{\max}$) past poses, evenly spaced in time	One	All other poses
SWF	Current IMU state + n ($\leq N_{\max}$) most recent poses	One	All other poses
OKVis	Keyframe poses in sliding window + associated features	Multiple	Keyframe poses leaving sliding window + associated features
PGO	Current and all past poses, with no features	Multiple, until convergence	None
BA	Current and all past poses and features, with no pairwise pose costs	Multiple, until convergence	None

Saxena, **Chiu**, et al, "Simultaneous Localization and Mapping: Through the Lens of Nonlinear Optimization," ICRA / R-AL, 2022.

Back End: Experiments

- **Goal** — Use our optimization framework to compare state-of-the-art SLAM back-end algorithms
- **Dataset** — EuRoC MAV, Vicon Room 2



Saxena, **Chiu**, et al, "Simultaneous Localization and Mapping: Through the Lens of Nonlinear Optimization," ICRA / R-AL, 2022.
Leutenegger et al. "Keyframe-based Visual-Inertial Odometry using Nonlinear Optimization," The International Journal of Robotics Research, 2015.
Mourikis, Roumeliotis. "A Multi-State Constraint Kalman Filter for Vision-aided Inertial Navigation," ICRA, 2007.

Back End: Experiments

- **Front End:**
  - Standardized across all experiments
- **Back Ends:**
  - Multi-State Constrained Kalman Filter **(MSCKF)**,
  - Sliding Window Filter **(SWF)**,
  - Open-Keyframe Visual Inertial SLAM **(OKVIS)**, etc.

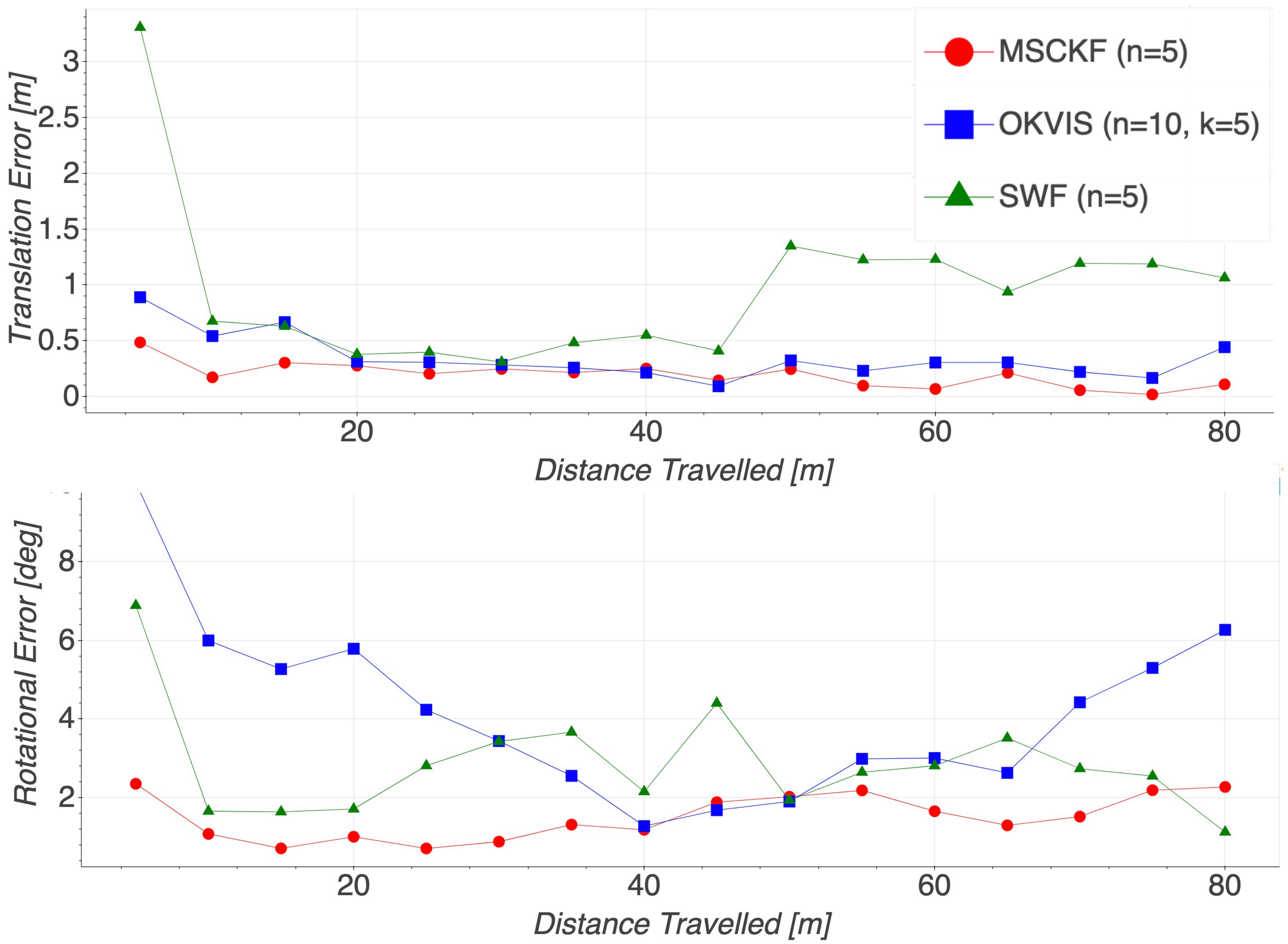


Fig. 1. Localization results the Vicon Room 2 (medium) dataset. Drift from the ground-truth location is plotted against the distance travelled along the ground-truth trajectory, sampled at 5 meter intervals. Note that the curves for MSCKF and iMSCKF are almost completely on top of each other.

Saxena, **Chiu**, et al, “Simultaneous Localization and Mapping: Through the Lens of Nonlinear Optimization,” ICRA / R-AL, 2022.
Leutenegger et al. “Keyframe-based Visual-Inertial Odometry using Nonlinear Optimization,” The International Journal of Robotics Research, 2015.
Mourikis, Roumeliotis. “A Multi-State Constraint Kalman Filter for Vision-aided Inertial Navigation,” ICRA, 2007.

Back End: Experiments

- **Conclusions:**
  - **MSCKF** outperforms **OK-VIS** and **SWFs**.
  - **MSCKF** recovers more easily from localization errors, by marginalizing in a manner that usually maintains some poses arbitrarily far in the past.
  - (Older poses supply better localization information.)

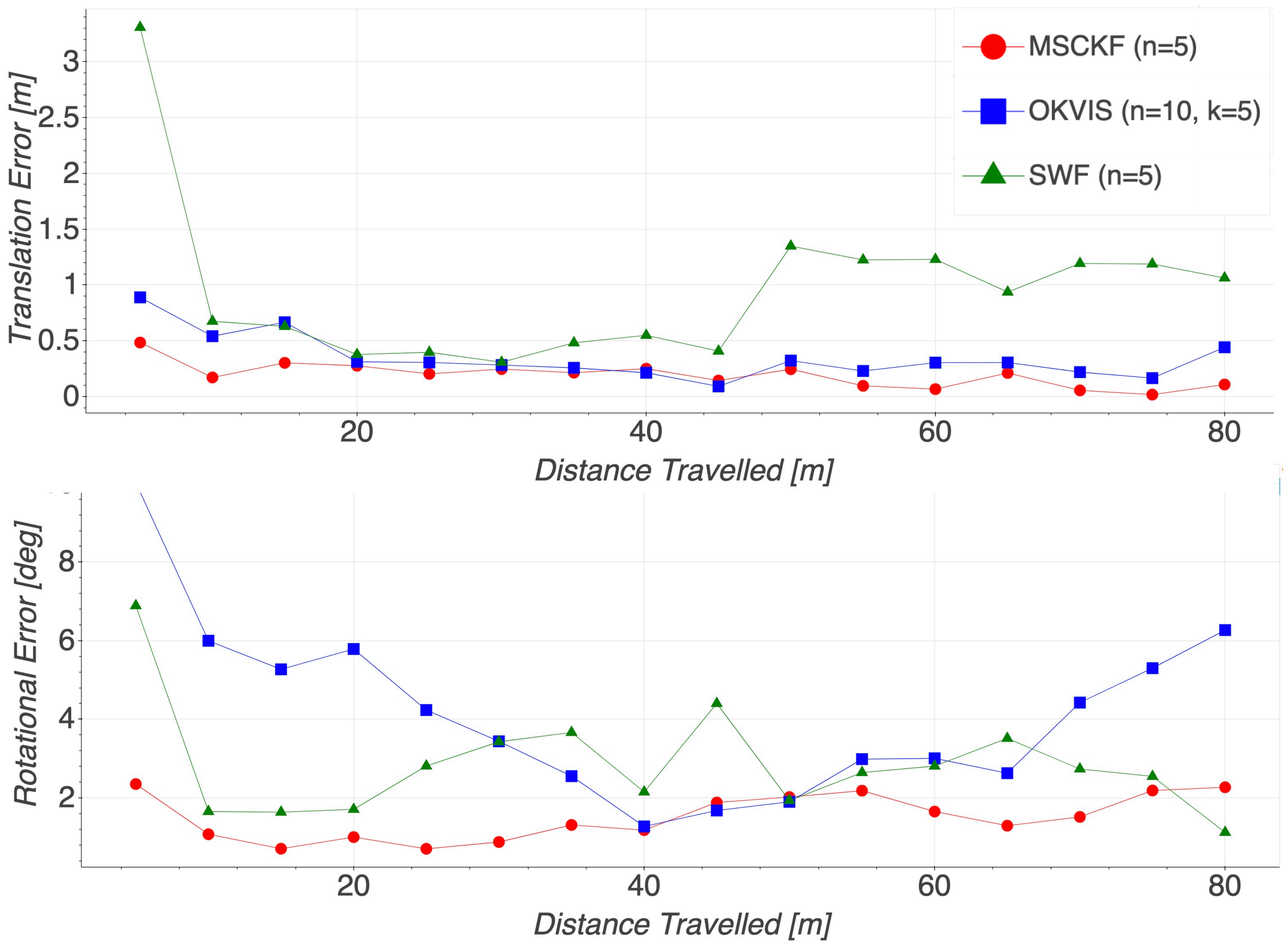
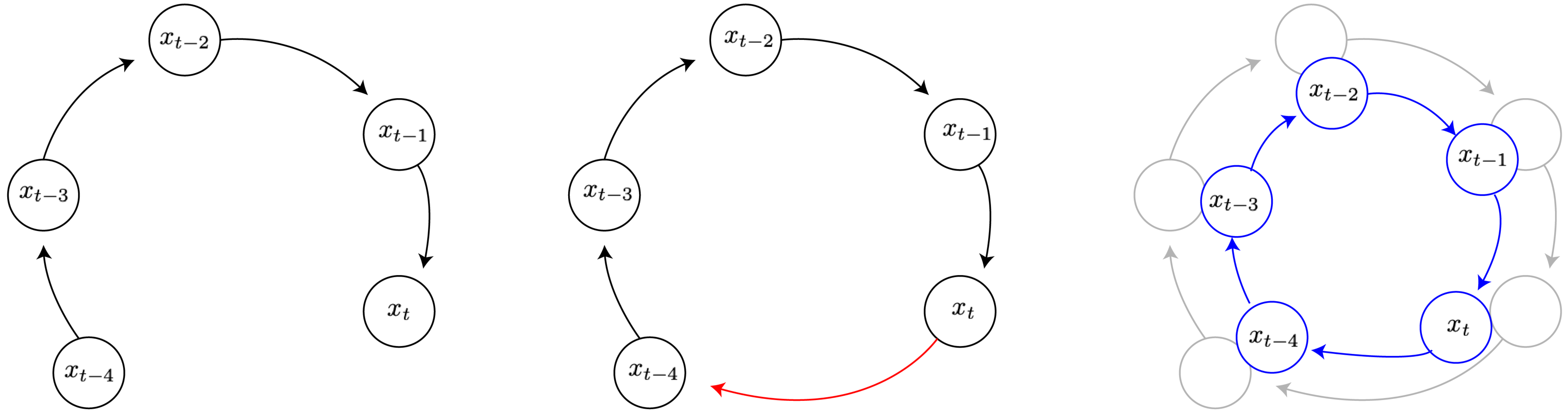


Fig. 1. Localization results the Vicon Room 2 (medium) dataset. Drift from the ground-truth location is plotted against the distance travelled along the ground-truth trajectory, sampled at 5 meter intervals. Note that the curves for MSCKF and iMSCKF are almost completely on top of each other.

Saxena, **Chiu**, et al, “Simultaneous Localization and Mapping: Through the Lens of Nonlinear Optimization,” ICRA / R-AL, 2022.
Leutenegger et al. “Keyframe-based Visual-Inertial Odometry using Nonlinear Optimization,” The International Journal of Robotics Research, 2015.
Mourikis, Roumeliotis. “A Multi-State Constraint Kalman Filter for Vision-aided Inertial Navigation,” ICRA, 2007.

Loop Closures: Global Optimization

- **What separates SLAM from odometry:**
  - **In addition to “local” real-time tracking:** Maintain a "global" optimization problem over all poses, even ones marginalized out of the “local” problem.
  - **In addition to incremental pose constraints:** Introduce "loop closure" constraints, activated when the robot re-visits parts of the map 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

Loop Closure Detection

- **To establish loop closures:**
  - Detect when the camera is looking at the same place as some time in the past.
  - **Naive method:** Compare every detected feature to every feature we have seen so far. Terrible.
  - **Bag-of-words approach:** Assigns a global, 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 (2012)

Outline

- **Introduction, FAQs**
- **Front End**
  - Feature Extraction
  - Data Association, with Outlier Rejection
- **Back End**
  - Goal, Setup
  - Unifying Framework: 3 Steps
  - Aside: Kalman Filtering, Basics
  - Example: EKF SLAM
  - State-of-the-art Algorithms
  - Experiments
  - Loop Closures
- **Next Steps**
  - Active Perception: Dynamic, Semantic SLAM

Next Steps: Active Perception

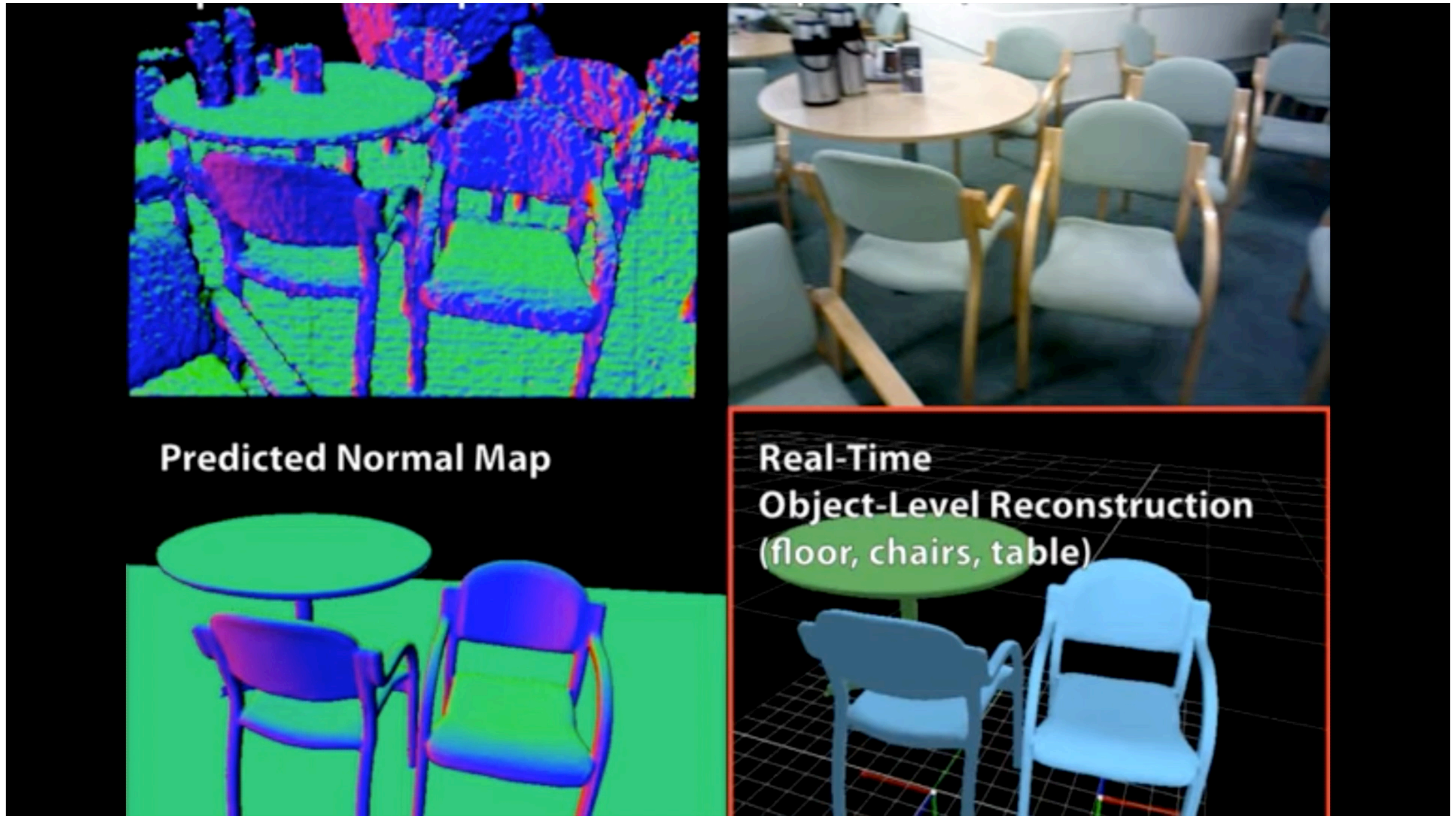
- “Controlling robot motion to minimize localization and map reconstruction uncertainty.” (Cadena et al.)

Video credits: https://www.youtube.com/watch?v=dQ6bY0XrZeg&t=165s

Cadena et al. “Past, Present, and Future of Simultaneous Localization and Mapping: Toward the Robust-Perception Age.” IEEE Transactions on Robotics, 2016.
R. Bajcsy, “Active perception,” Proc. IEEE, vol. 76, no. 8, pp. 966–1005, Aug. 1988.
Davison et al., “FutureMapping: The Computational Structure of Spatial AI Systems.”

Next Steps: Semantic SLAM

- Learning-based feature labeling + Optimization-based back end.



Video Credit: “SLAM++: Simultaneous Localisation and Mapping at the Level of Objects,“ (https://www.youtube.com/watch?v=tmrAh1CqCRo)
Cadena et al. “Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age.” IEEE Transactions on Robotics, 2016.
Davison et al., “FutureMapping: The Computational Structure of Spatial AI Systems.”

Next Steps: Dynamic SLAM

- **Goal:** Use SLAM to track moving features.
- This enhances intent inference → motion predictions of other agents → safety and efficiency of planned trajectories.

Cadena et al. “Past, Present, and Future of Simultaneous Localization and Mapping: Toward the Robust-Perception Age.” IEEE Transactions on Robotics, 2016.
Zhang, Henein et al., “VDO-SLAM: A Visual Dynamic Object-aware SLAM System,” ArXiv, 2020.

Next Steps: Dynamic SLAM

- **Goal:** Use SLAM to track moving features.
- **Recall:** 3-step back-end procedure for Static SLAM
  - Feature Augmentation (Cost Construction Step)
  - Feature Update (Gauss-Newton Step)
  - State Propagation (Marginalization Step)
- Missing pieces for Dynamic SLAM:
  - Identifying which features belong to a single moving object (front-end)
  - Optimizing a motion model for moving objects (back-end)
  - Ensuring motion model smoothness for moving objects (back-end)

Chiu, "SLAM Backends with Objects in Motion: A Unifying Framework and Tutorial," ACC 2023.

Next Steps: Dynamic SLAM

- **Goal:** Use SLAM to track moving features.
- **Roadmap:** 5-step procedure for Static SLAM
  - Feature Augmentation (Cost Construction Step)
  - Moving Object Pose Augmentation (Gauss-Newton Step)
  - Feature Update (Gauss-Newton Step)
  - Smoothing Factor Augmentation (Gauss-Newton Step)
  - State Propagation (Marginalization Step)
-

Chiu, "SLAM Backends with Objects in Motion: A Unifying Framework and Tutorial," ACC 2023.

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

Front End: Data Association

- **SIFT: (Scale-Invariant Feature Transform)**
  - Feature matching algorithm, invariant to scale and rotations
  - Uses high-dimensional feature descriptors in each object for identification
  - Heavy computational burden — Too slow for SLAM
  - Subject to licensing restrictions (in contrast, ORB is open-source)

Lowe. “Object Recognition from Local Scale-Invariant Features,” ICCV 1999.
SIFT, OpenCV Code: https://docs.opencv.org/4.x/da/df5/tutorial_py_sift_intro.html

Front End: Data Association

- **SURF: (Speeded-up Robust Features)**
  - Designs feature descriptors to have descriptive power and computation speed
  - Has poorer performance compared to ORB
  - Subject to licensing restrictions (in contrast, ORB is open-source)
-

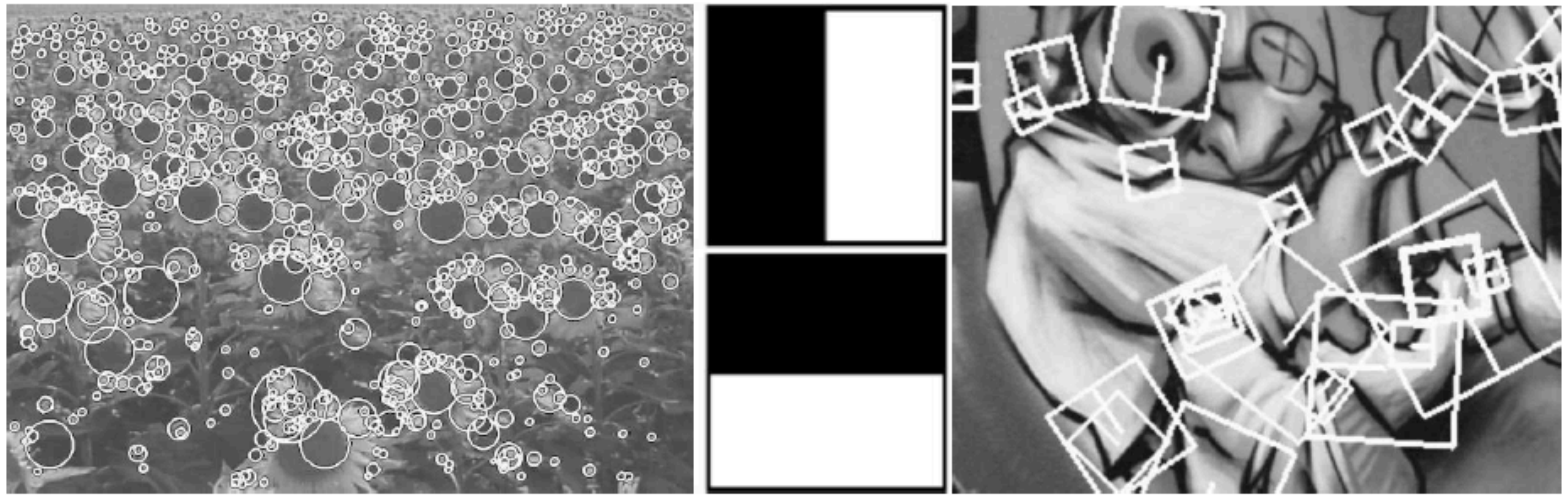


Fig. 2. Left: Detected interest points for a Sunflower field. This kind of scenes shows clearly the nature of the features from Hessian-based detectors. Middle: Haar wavelet types used for SURF. Right: Detail of the Graffiti scene showing the size of the descriptor window at different scales.

Bay et al. "SURF: Speeded up Robust Features," ECCV 2006.
SURF, OpenCV Code: https://docs.opencv.org/3.4/df/dd2/tutorial_py_surf_intro.html

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.

Back End: EKF SLAM

- **Theorem: Feature Augmentation = Cost Construction + 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:**
  - **Filtering approach**—Linearize inverse measurement model (image → feature position), then perform a MAP estimate update using this linearized function
  - **Optimization approach**—Use measurement model to form a nonlinear least-squares cost, then perform one Gauss-Newton step on this cost
  - Theorem 6.1 shows that these two approaches are equivalent.

Back End: EKF SLAM

- **Theorem: 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:**
  - **Filtering approach**—Linearize measurement model, then perform a MAP estimate update using this linearized function
  - **Optimization approach**—Use measurement model to form a nonlinear least-squares cost, then perform one Gauss-Newton step on this cost
  - Theorem 6.2 shows that these two approaches are equivalent.

Back End: EKF SLAM

- **Theorem: 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:**
  - **Filtering approach**—Linearize dynamics model, then perform a MAP estimate update using this linearized function
  - **Optimization approach**—Use dynamics model to form a nonlinear least-squares cost, then perform one marginalization step on this cost
  - Theorem 6.3 shows that these two approaches are equivalent.

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.

Back End: Setup and Terminology

- **Dynamics model 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 model with an example:**
  - General form: $h : \mathbb{R}^{d_x} \times \mathbb{R}^{d_f} \to \mathbb{R}^{d_z}$

    $z_{t,j} = h(x_t, f_{t,j}) + v_{t,j}$, 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: $f_{t,j} := (f_{t,j}^1, f_{t,j}^2) \in \mathbb{R}^2$
  - Image measurements: $z_{t,j} := (z_{t,j}^1, z_{t,j}^2) \in \mathbb{R}^2$
  - Noise: $v_t := (v_t^1, v_t^2) \in \mathbb{R}^2$
  - Measurement model:

$$z_{t,j}^1 = f_{t,j}^1 - x_t^1 + v_t^1$$
$$z_{t,j}^2 = f_{t,j}^2 - x_t^2 + v_t^2$$

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:

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

where:

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

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:

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

  - 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:

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

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

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:

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

  - 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:

$$
\boxed{
\begin{array}{ll}
\mathbf{5} & \overline{\mu_t} \leftarrow \mu_t + \Sigma_t H_t^T (H_t \Sigma_t H_t^T + \tilde{\Sigma}_v)^{-1} \big(z_{t,1:p} - \tilde{h}(\mu_t, f_{t,1:p})\big) \in \mathbb{R}^{d_x+pd_f}. \\
\mathbf{6} & \overline{\Sigma}_t \leftarrow \Sigma_t - \Sigma_t H_t^T (H_t \Sigma_t H_t^T + \tilde{\Sigma}_v)^{-1} H_t \Sigma_t \in \mathbb{R}^{(d_x+pd_f)\times(d_x+pd_f)}.
\end{array}
}
$$

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:

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

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

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

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