SLAM-related Notes

Charlie Li

2018.05.28

Table of Contents

Notes on SLAM Tutorial Part I (Durrant-Whyte and Bailey 2006) (paper)

- · Problem:
  - · Whether a mobile robot placed in an unknown location of an unknown environment can incrementally **build a consistent map** of this environment while simultaneously **determining its location** within this map

1. History

- · As a mobile robot moves through an unknown environment taking relative observations of landmarks, the estimates of these landmarks are all necessarily correlated with each other because of the common error in estimated vehicle location
  - · Implication: consistent full solution of SLAM problem would require **a joint state** composed of the **vehicle pose** and **every landmark position**, to be updated following **each landmark observation**
    - · Pose: combination of position & orientation
- · Once SLAM is formulated as a single **estimation problem**, the estimated map errors are **actually convergent**
- · The more these correlations between landmarks grow, the better the solution

2. Formulation & Structure of the SLAM Problem

- · SLAM: a process by which a mobile robot can **build a map of an environment** and at the same time **use this map to deduce its location**
- · Both the **trajectory of the platform** and the **location of all landmarks** can be estimated online without any **a priori knowledge of location**

2.1 Preliminaries

· SLAM problem illustration:
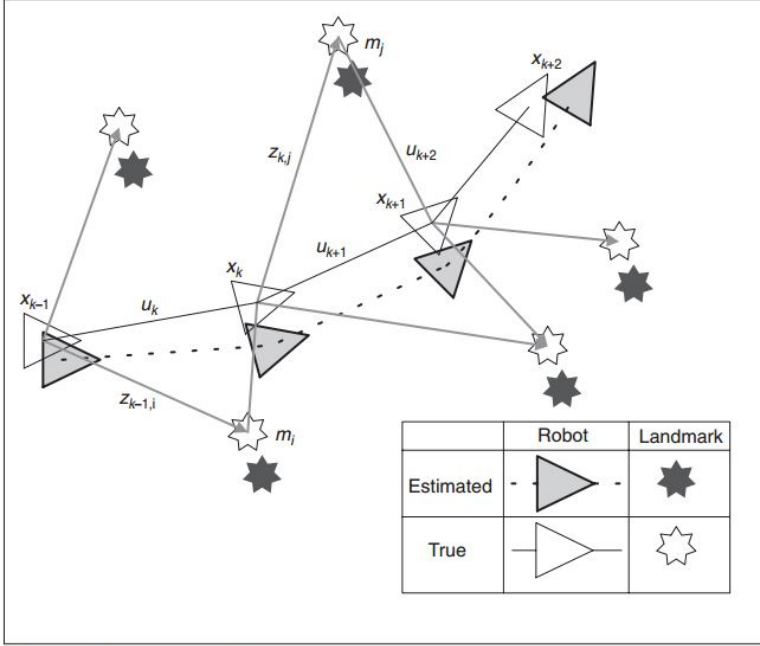
Figure 1. *The essential SLAM problem. A simultaneous estimate of both robot and landmark locations is required. The true locations are never known or measured directly. Observations are made between true robot and landmark locations.*

· $\mathbf{x}_k$: the state vector indicating vehicle location & orientation at time k
· $\mathbf{u}_k$: the control vector, applied at time $k-1$ to drive the vehicle to state vector $\mathbf{x}_k$ at time k
· $\mathbf{m}_i$: a vector indicating the location of ith landmark whose true location is assumed to be time invariant
· $\mathbf{z}_{ik}$: an observation of ith landmark taken from the vehicle at time k
    · May be abbreviated as $\mathbf{z}_k$ indicating multiple observations at time k or if the specific landmark is not revelant to the discussion
· $\mathbf{X}_{0:k} = \{\mathbf{x}_0, \mathbf{x}_1, ..., \mathbf{x}_k\} = \{\mathbf{X}_{0:k-1}, \mathbf{x}_k\}$: the history of previous vehicle locations
· $\mathbf{U}_{0:k} = \{\mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_k\} = \{\mathbf{U}_{0:k-1}, \mathbf{u}_k\}$: the history of control inputs
· $\mathbf{m} = \{\mathbf{m}_1, \mathbf{m}_2, ..., \mathbf{m}_n\}$: the set of all landmarks
· $\mathbf{Z}_{0:k} = \{\mathbf{z}_1, \mathbf{z}_2, ..., \mathbf{z}_k\} = \{\mathbf{Z}_{0:k-1}, \mathbf{z}_k\}$: the set of all landmark observations

2.2 Probabilistic SLAM

· SLAM problem requires the computation of $P(\mathbf{x}_k, \mathbf{m}|\mathbf{Z}_{0:k}, \mathbf{U}_{0:k}, \mathbf{x}_0)$ at all times k
    · Joint posterior density of the *vehicle state* and the *landmark locations* (at time k)
    · Recursive computation: compute joint posterior density at time k by $P(\mathbf{x}_{k-1}, \mathbf{m}|\mathbf{Z}_{0:k-1}, \mathbf{U}_{0:k-1}, \mathbf{x}_0)$ following control $\mathbf{u}_k$ and observation $\mathbf{z}_k$
    · Requirement: the definition of a ***state transition model*** and an ***observation model*** describing the control input and observation respectively
        · ***Observation model***: $P(\mathbf{z}_k|\mathbf{x}_k, \mathbf{m})$
            · The probability of making an observation at time k given the *vehicle location* and *landmark locations*
            · The observations are ***conditionally independent*** of current vehicle state $(\mathbf{x}_k)$ and the map $(\mathbf{m})$ once the vehicle location and map are defined

· **Motion model**: $P(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{u}_k)$
   · Probability distribution on state transitions
   · Assume state transition is a **Markov process**:
      · The next state $\mathbf{x}_k$ depends only on the previous state $\mathbf{x}_{k-1}$ and the applied control $\mathbf{u}_k$
      · State transition is independent of both the observations and the map
· SLAM Algorithm: two-step recursive (*sequential*) prediction (*time-update*) correction (*measurement-update*) form
   · Step 1: Time-update

$$P(\mathbf{x}_k, \mathbf{m}|\mathbf{Z}_{0:k-1}, \mathbf{U}_{0:k}, \mathbf{x}_0) = \int P(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{u}_k)P(\mathbf{x}_{k-1}, \mathbf{m}|\mathbf{Z}_{0:k-1}, \mathbf{U}_{0:k-1}, \mathbf{x}_0)\mathrm{d}\mathbf{x}_{k-1}$$

   · Notes:
      · Only vehicle location $\mathbf{x}_k$ is updated? Map data $\mathbf{m}$ should be of time $k-1$
      · Law of total probability: $P(A) = \sum_n P(A|B_n)P(B_n)$
         · Event A: get prediction of current vehicle location & measurement of *?previous?* landmark locations
         · Event B: get measurement of previous vehicle location & landmark locations
   · Step 2: Measurement-update

$$P(\mathbf{x}_k, \mathbf{m}|\mathbf{Z}_{0:k}, \mathbf{U}_{0:k}, \mathbf{x}_0) = \frac{P(\mathbf{z}_k|\mathbf{x}_k, \mathbf{m})P(\mathbf{x}_k, \mathbf{m}|\mathbf{Z}_{0:k-1}, \mathbf{U}_{0:k}, \mathbf{x}_0)}{P(\mathbf{z}_k|\mathbf{Z}_{0:k-1}, \mathbf{U}_{0:k})}$$

   · Notes:
      · Bayes' theorem: $P(A|B, C) = \dfrac{P(B|A, C)P(A|C)}{P(B|C)}$
      · Event A: get current vehicle location & landmark locations
         · $\mathbf{x}_k, \mathbf{m}$
      · Event B: get current observation
         · $\mathbf{z}_k$
      · Event C: get historical observations & control inputs
         · $\mathbf{Z}_{0:k-1}, \mathbf{U}_{0:k-1}, \mathbf{x}_0$
      · Current observation is only related to current vehicle state and map
         · $P(B|A, C) = P(\mathbf{z}_k|(\mathbf{x}_k, \mathbf{m}), (\mathbf{Z}_{0:k-1}, \mathbf{U}_{0:k-1}, \mathbf{x}_0)) = P(\mathbf{z}_k|\mathbf{x}_k, \mathbf{m}) = P(B|A)$

2.3 Structure of Probabilistic SLAM

· Observations are **dependent** on both vehicle & landmark locations from the *observation model* $P(\mathbf{z}_k|\mathbf{x}_k, \mathbf{m})$, therefore:

$$P(\mathbf{x}_k, \mathbf{m}|\mathbf{Z}_{0:k}, \mathbf{U}_{0:k}, \mathbf{x}_0) \neq P(\mathbf{x}_k|\mathbf{Z}_{0:k}, \mathbf{U}_{0:k}, \mathbf{x}_0)P(\mathbf{m}|\mathbf{Z}_{0:k}, \mathbf{U}_{0:k}, \mathbf{x}_0)$$

· Errors in landmark locations are highly correlated
   · Relative location between any 2 landmarks $\mathbf{m}_i$, $\mathbf{m}_j$, may be **known with high accuracy**, even when the absolute location of a landmark $\mathbf{m}_i$ is quite uncertain
      · Meaning in probabilistic form: joint probability density for the pair of landmarks

$P(\mathbf{m}_i, \mathbf{m}_j)$ is highly peaked even when the marginal densities $P(\mathbf{m}_i)$ may be quite dispersed

- Correlations between landmark estimates increase monotonically as more and more observations are made
  - Knowledge of relative landmark locations always improves and never diverges, regardless of robot motion
    - $P(\mathbf{m})$ becomes monotonically peaked as more observations are made
  - All known relative landmark locations are updated at each time segment even though some of the landmarks are not seen by the vehicle at that time

3. Solutions to the SLAM Problem

- Solutions involve *finding an appropriate representation* for both the **observation model** and the **motion model**
  - Representation 1: a state-space model with additive Gaussian noise
    - Use **extended Kalman filter** (EKF) to solve the SLAM problem (**EKF-SLAM**)
  - Representation 2: describe vehicle motion model $P(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{u}_k)$ as a set of samples of a more general non-Gaussian probability distribution
    - Use **Rao-Blackwellized particle filter** or **FastSLAM** algorithm to solve the SLAM problem

3.1 Prerequisites: Kalman Filter & EKF

- Tutorials on Kalman filter & extended Kalman filter:
  - Kalman Filter: #1 | *#2* & extra | #3 (Faragher 2012) | #4 (Wiki)
  - Extend Kalman Filter: #1 | #2 | #3 (Wiki)

3.1.1 Kalman Filter

- Models
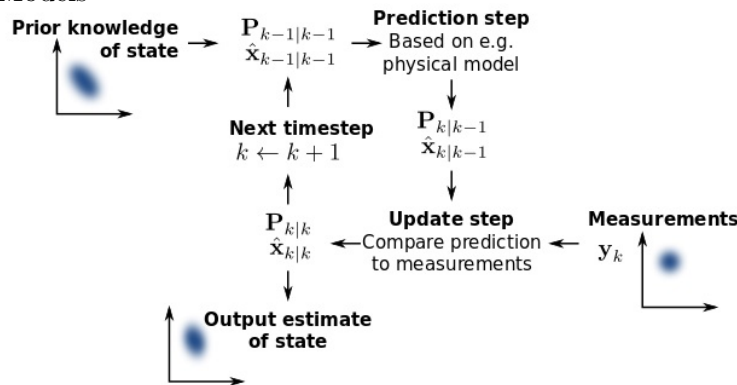
- **State model** ($P(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{u}_k)$):

$$\mathbf{x}_k = \mathbf{F}_k\mathbf{x}_{k-1} + \mathbf{B}_k\mathbf{u}_k + \mathbf{w}_k$$

- $\mathbf{x}_k$: the state vector containing the parameters for the system at time k

- $\mathbf{F}_k$: state-transition model to transfer the state vector $\mathbf{x}_{k-1}$ (such as the position & velocity of a robot) at time $k-1$ to the next state $\mathbf{x}_k$ at time k
- $\mathbf{B}_k$: control-input model - control matrix applying on the control vector $\mathbf{u}_k$ (external force to lead the robot to the next state) at time k
- $\mathbf{w}_k$: process noise at time k which assumes to have a multivariate Gaussian distribution with zero mean and a covariance of $\mathbf{Q}_k$
  - $\mathbf{w}_k \sim \mathcal{N}(0, \mathbf{Q}_k)$
- ***Observation model*** ($P(\mathbf{z}_k|\mathbf{x}_k)$):

$$\mathbf{z}_k = \mathbf{H}_k\mathbf{x}_k + \mathbf{v}_k$$

  - $\mathbf{z}_k$: observation/measurement at time k made by true state $\mathbf{x}_k$ (not an estimation)
  - $\mathbf{H}_k$: observation model (transformation matrix) which maps the state space/domain into the observation/measurement space/domain
  - $\mathbf{v}_k$: observation noise which assumes to have a multivariate Gaussian distribution of zero mean and a covariance of $\mathbf{R}_k$
    - $\mathbf{v}_k \sim \mathcal{N}(0, \mathbf{R}_k)$
- 2 steps: prediction & update
  - Step 1: Kalman filter produces estimates of the *current state variables*, along with their uncertainties

$$\hat{\mathbf{x}}_{k|k-1} = \mathbf{F}_k\hat{\mathbf{x}}_{k-1|k-1} + \mathbf{B}_k\mathbf{u}_k$$

$$\mathbf{P}_{k|k-1} = \mathbf{F}_k\mathbf{P}_{k-1|k-1}\mathbf{F}_k^T + \mathbf{Q}_k$$

  - General notation $\mathbf{X}_{a|b}$: $\mathbf{X}$ at time a given observations up to and including at time $b \leq a$
  - State prediction:
    - $\hat{\mathbf{x}}_{k|k-1}$: the ***estimate*** of state vector $\mathbf{x}$ at time k given observations up to and including at time $k-1$
  - Covariance prediction:
    - $\mathbf{P}_{k|k-1}$: covariance matrix at time k (describes the correlations between any two parameters inside the state vector $\mathbf{x}$)
    - $\mathbf{Q}_k$: covariance matrix of the process noise (additive Gaussian noise) at time k
    - Derivation (notice the uncorrelation between state estimation errors and process noise):

$$
\begin{aligned}
\mathbf{P}_{k|k-1} &= \mathrm{E}[(\mathbf{x}_k - \hat{\mathbf{x}}_{k|k-1})(\mathbf{x}_k - \hat{\mathbf{x}}_{k|k-1})^T] \\
&= \mathrm{E}[(\mathbf{F}_k(\mathbf{x}_{k-1} - \hat{\mathbf{x}}_{k|k-1}) + \mathbf{w}_k)(\mathbf{F}_k(\mathbf{x}_{k-1} - \hat{\mathbf{x}}_{k|k-1}) + \mathbf{w}_k)^T] \\
&= \mathrm{E}[(\mathbf{F}_k(\mathbf{x}_{k-1} - \hat{\mathbf{x}}_{k|k-1})(\mathbf{x}_{k-1} - \hat{\mathbf{x}}_{k|k-1})^T\mathbf{F}_k^T)] + \mathrm{E}[\mathbf{F}_k(\mathbf{x}_{k-1} - \hat{\mathbf{x}}_{k|k-1})\mathbf{w}_k^T] \\
&\quad + \mathrm{E}[\mathbf{w}_k(\mathbf{x}_{k-1} - \hat{\mathbf{x}}_{k|k-1})^T\mathbf{F}_k^T] + \mathrm{E}[\mathbf{w}_k\mathbf{w}_k^T] \\
&= \mathbf{F}_k\mathrm{E}[(\mathbf{x}_{k-1} - \hat{\mathbf{x}}_{k|k-1})(\mathbf{x}_{k-1} - \hat{\mathbf{x}}_{k|k-1})^T]\mathbf{F}_k^T + 0 + 0 + \mathrm{E}[\mathbf{w}_k\mathbf{w}_k^T] \\
&= \mathbf{F}_k\mathbf{P}_{k-1|k-1}\mathbf{F}_k^T + \mathbf{Q}_k
\end{aligned}
$$

- Step 2: estimates are updated using a *weighted average* once the outcome of the *next*

measurement is observed

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k \tilde{\mathbf{y}}_k$$

$$\mathbf{P}_{k|k} = (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k)\mathbf{P}_{k|k-1}$$

where

$$\mathbf{K}_k = \mathbf{P}_{k|k-1} \mathbf{H}_k^T \mathbf{S}_k^{-1}$$

$$\tilde{\mathbf{y}}_k = \mathbf{z}_k - \mathbf{H}_k \hat{\mathbf{x}}_{k|k-1}$$

$$\mathbf{S}_k = \mathbf{H}_k \mathbf{P}_{k|k-1} \mathbf{H}_k^T + \mathbf{R}_k$$

· $\tilde{\mathbf{y}}_k$ & $\mathbf{S}_k$: innovation / measurement residual and its covariance at time k
· Both *estimated observation* ($\mathbf{z}_{k,\text{estimated}} = \mathbf{H}_k \hat{\mathbf{x}}_{k|k-1}$) and *actual observation* ($\mathbf{z}_k$) have uncertainties (a range of possible observation data (assuming Gaussian distributions))
· $\mathbf{z}_{k,\text{estimated}} \sim \mathcal{N}(\mathbf{H}_k \hat{\mathbf{x}}_{k|k-1}, \mathbf{H}_k \mathbf{P}_{k|k-1} \mathbf{H}_k^T) = \mathcal{N}(\mathbf{H}_k \boldsymbol{\mu}_0, \mathbf{H}_k \boldsymbol{\Sigma}_0 \mathbf{H}_k^T)$
· $\mathbf{z}_{k,\text{actual}} \sim \mathcal{N}(\mathbf{z}_k, \mathbf{R}_k) = \mathcal{N}(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)$
· Choose the **overlapped** region of the 2 ranges as the best guess of the real observation:
    · Also a Gaussian distribution (multiplying the 2 densities): $\mathbf{z}_{k,\text{fused}} \sim \mathcal{N}(\boldsymbol{\mu}_{\text{fused}}, \boldsymbol{\Sigma}_{\text{fused}})$
    · $\boldsymbol{\mu}_{\text{fused}} = \boldsymbol{\mu}_0 + \mathbf{K}_k(\boldsymbol{\mu}_1 - \mathbf{H}_k \boldsymbol{\mu}_0)$
    · $\boldsymbol{\Sigma}_{\text{fused}} = \boldsymbol{\Sigma}_0 - \mathbf{K}_k \mathbf{H}_k \boldsymbol{\Sigma}_0$
    · $\mathbf{K}_k = \boldsymbol{\Sigma}_0 \mathbf{H}_k^T (\mathbf{H}_k \boldsymbol{\Sigma}_0 \mathbf{H}_k^T + \boldsymbol{\Sigma}_1)^{-1}$ ($\mathbf{K}_k$: **Kalman gain** at time k)

3.1.2 Extended Kalman Filter (EKF)

Notes: mathematical notations are almost the same with those in KF

· KF limitations:
    · Linear function/model/system
    · Gaussian distribution for observed variables
· EKF: Nonlinear version of KF
    · Use **estimated** mean & covariance of observed variables for filter response computation
    · Mean & covariance is estimated using linear approximations through Taylor Series (multivariate Taylor Series)
        · Linear because only the first order (partial) derivatives are computed (using the Jacobian matrix)
· Models: use **differentiable** functions (f & h) instead of linear functions (i.e., matrices $\mathbf{F}_k$, $\mathbf{B}_k$, and $\mathbf{H}_k$)
    · **State model**:

$$\mathbf{x}_k = f(\mathbf{x}_{k-1}, \mathbf{u}_k) + \mathbf{w}_k$$

    · **Observation model**:

$$\mathbf{z}_k = h(\mathbf{x}_k) + \mathbf{v}_k$$

· 2 steps: prediction & update
    · Step 1: prediction

$$\hat{\mathbf{x}}_{k|k-1} = f(\hat{\mathbf{x}}_{k-1|k-1}, \mathbf{u}_k)$$

10

$$\mathbf{P}_{k|k-1} = \mathbf{F}_k \mathbf{P}_{k-1|k-1} \mathbf{F}_k^T + \mathbf{Q}_k$$

- $\mathbf{F}_k$: Jacobian matrix whose elements are the linear approximations (partial derivatives) of state transition function f defined as follows (assume state vector has a dimension of $n \times 1$)

$$\mathbf{F}_{k,n \times n} = \frac{\partial f_{n \times 1}}{\partial \mathbf{x}_{n \times 1}}(\hat{\mathbf{x}}_{k-1|k-1}, \mathbf{u}_k)$$

- Step 2: update

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k \tilde{\mathbf{y}}_k$$

$$\mathbf{P}_{k|k} = (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k)\mathbf{P}_{k|k-1}$$

where

$$\mathbf{K}_k = \mathbf{P}_{k|k-1}\mathbf{H}_k^T \mathbf{S}_k^{-1}$$

$$\tilde{\mathbf{y}}_k = \mathbf{z}_k - h(\hat{\mathbf{x}}_{k|k-1})$$

$$\mathbf{S}_k = \mathbf{H}_k \mathbf{P}_{k|k-1}\mathbf{H}_k^T + \mathbf{R}_k$$

- $\mathbf{H}_k$: Jacobian matrix whose elements are the linear approximations (partial derivatives) of measurement-to-state mapping function h defined as follows (assume state vector has a dimension of $n \times 1$ and observation vector has a dimension of $m \times 1$)

$$\mathbf{H}_{k,m \times n} = \frac{\partial h_{m \times 1}}{\partial \mathbf{x}_{n \times 1}}(\hat{\mathbf{x}}_{k|k-1})$$

3.2 EKF-SLAM

- Models
  - Motion model:

$$P(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{u}_k) \iff \mathbf{x}_k = f(\mathbf{x}_{k-1}, \mathbf{u}_k) + \mathbf{w}_k$$

  - $f(\cdot)$: models the vehicle kinematics
  - $\mathbf{w}_k$: *additive, uncorrelated* Gaussian motion disturbances, and $\mathbf{w}_k \sim \mathcal{N}(0, \mathbf{Q}_k)$
  - Observation model:

$$P(\mathbf{z}_k|\mathbf{x}_k, \mathbf{m}) \iff \mathbf{z}_k = h(\mathbf{x}_k, \mathbf{m}) + \mathbf{v}_k$$

  - $h(\cdot)$: models the geometry of the observation
  - $\mathbf{v}_k$: *additive, uncorrelated* Gaussian observation errors, and $\mathbf{v}_k \sim \mathcal{N}(0, \mathbf{R}_k)$
- Applying EKF to SLAM problem by modeling joint posterior distribution $P(\mathbf{x}_k, \mathbf{m}|\mathbf{Z}_{0:k}, \mathbf{U}_{0:k}, \mathbf{x}_0)$ as $\mathbf{z}_{k,\text{fused}}$

$$\mathbf{z}_{k,\text{fused}} \sim \mathcal{N}(\boldsymbol{\mu}_{\text{fused}}, \boldsymbol{\Sigma}_{\text{fused}})$$

$$\boldsymbol{\mu}_{\text{fused}} = \begin{bmatrix} \hat{\mathbf{x}}_{k|k} \\ \hat{\mathbf{m}}_k \end{bmatrix} = \mathrm{E}\left[\begin{array}{c} \mathbf{x}_k \\ \mathbf{m} \end{array} \bigg| \mathbf{Z}_{0:k}\right]$$

$$\boldsymbol{\Sigma}_{\text{fused}} = \begin{bmatrix} \mathbf{P}_{xx} & \mathbf{P}_{xm} \\ \mathbf{P}_{xm}^T & \mathbf{P}_{mm} \end{bmatrix}_{k|k} = \mathrm{E}\left[\begin{pmatrix} \mathbf{x}_k - \hat{\mathbf{x}}_{k|k} \\ \mathbf{m} - \hat{\mathbf{m}}_k \end{pmatrix}\begin{pmatrix} \mathbf{x}_k - \hat{\mathbf{x}}_{k|k} \\ \mathbf{m} - \hat{\mathbf{m}}_k \end{pmatrix}^T\right] = \mathrm{E}\left[\begin{pmatrix} \mathbf{x}_k - \hat{\mathbf{x}}_k \\ \mathbf{m} - \hat{\mathbf{m}}_k \end{pmatrix}\begin{pmatrix} \mathbf{x}_k - \hat{\mathbf{x}}_k \\ \mathbf{m} - \hat{\mathbf{m}}_k \end{pmatrix}^T \bigg| \mathbf{Z}_{0:k}\right]$$

- EKF applying: ***time-update*** (prediction step) & ***observation-update*** (update step)

- Time-update is only for state vectors (except when landmarks can move)
- Observation-update: $\hat{\mathbf{x}}_{k|k} \rightarrow \begin{bmatrix} \hat{\mathbf{x}}_{k|k} \\ \hat{\mathbf{m}}_k \end{bmatrix}$
- $\mathbf{F}_k = \nabla f = \dfrac{\partial f}{\partial \mathbf{x}}(\hat{\mathbf{x}}_{k-1|k-1}, \mathbf{u}_k)$
- $\mathbf{H}_k = \nabla h = \dfrac{\partial h}{\partial \mathbf{x}}(\hat{\mathbf{x}}_{k|k-1}, \hat{\mathbf{m}}_{k-1})$
- $\mathbf{W}_k$: Kalman gain ($\mathbf{K}_k$)
- Map convergence: map convergence matrix $\mathbf{P}_{mm,k}$ and all landmark pair submatrices are monotonically convergent toward zero
- Computation grows **quadratically** with **the number of landmarks** because all landmarks and joint covariance matrix will be updated every time an observation is made
- Standard formulation of EKF-SLAM solution is fragile to incorrect association of observations to landmarks (**loop-closure** problem)
- Linearization of nonlinear motion & observation models (**E**KF) leads to inconsistency in solutions

3.3 Rao-Blackwellized Filter (FastSLAM)

- Basis: recursive Monte Carlo sampling / particle filtering
- Applying Rao-Blackwellization (R-B) on sample-space (state-space)
  - Partition joint state according to the product rule: $P(\mathbf{x}_1, \mathbf{x}_2) = P(\mathbf{x}_2|\mathbf{x}_1)P(\mathbf{x}_1)$
    - Only $P(\mathbf{x}_1)$ needs be sampled ($\mathbf{x}_1^{(i)} \sim P(\mathbf{x}_1)$) if $P(\mathbf{x}_2|\mathbf{x}_1)$ can be represented analytically
  - Joint distribution is represented by the set $\{\mathbf{x}_1^{(i)}, P(\mathbf{x}_2|\mathbf{x}_1^{(i)})\}$
  - Statistics can be obtained by sampling over the joint space
    - E.g., the marginal:
    $$P(\mathbf{x}_2) \approx \frac{1}{N} \sum_i^N P(\mathbf{x}_2|\mathbf{x}_1^{(i)})$$
- Joint SLAM state may be factored into a **vehicle component** and a **conditional map component**:

$$P(\mathbf{X}_{0:k}, \mathbf{m}|\mathbf{Z}_{0:k}, \mathbf{U}_{0:k}, \mathbf{x}_0) = P(\mathbf{X}_{0:k}|\mathbf{Z}_{0:k}, \mathbf{U}_{0:k}, \mathbf{x}_0)P(\mathbf{m}|\mathbf{X}_{0:k}, \mathbf{Z}_{0:k}, \mathbf{U}_{0:k}, \mathbf{x}_0)$$
$$= P(\mathbf{X}_{0:k}|\mathbf{Z}_{0:k}, \mathbf{U}_{0:k}, \mathbf{x}_0)P(\mathbf{m}|\mathbf{X}_{0:k}, \mathbf{Z}_{0:k})$$

  - Note: $P(A, B|C) = P(A|C)P(B|A, C)$
  - Map landmarks **become independent** if the distribution is conditioned on the trajectory $\mathbf{X}_{0:k}$ rather than a single pose $\mathbf{x}_k$

$$P(\mathbf{m}|\mathbf{X}_{0:k}, \mathbf{Z}_{0:k}) = \prod_{j=1}^n P(\mathbf{m}_j|\mathbf{X}_{0:k}, \mathbf{Z}_{0:k})$$

  - Conditional independence of landmarks:
    - As landmark status is dependent with history of observations & robot poses, if all the history info are known, any landmark status in the landmark set can be extracted using the same history info
      - Thus there's no dependence between any two of the landmarks given the history info (observation history is manifestly known)

- $P(\mathbf{m}_i|\mathbf{m}_j, \mathbf{X}_{0:k}, \mathbf{Z}_{0:k}) = P(\mathbf{m}_i|\mathbf{X}_{0:k}, \mathbf{Z}_{0:k})$
    - The map is represented as a set of ***independent Gaussians*** in FastSLAM with linear complexity rather than a joint map of covariance with quardratic complexity (n^2 elements in the covariance matrix at any given time k)
- Rao-Blackwellized joint distribution (N-sample set) at time k: $\{w_k^{(i)}, \mathbf{X}_{0:k}^{(i)}, P(\mathbf{m}|\mathbf{X}_{0:k}^{(i)}, \mathbf{Z}_{0:k})\}_i^N$
    - $w_k^{(i)}$: weight for trajectory particle/sample $\mathbf{X}_{0:k}^{(i)}$
    - Rao-Blackwellized conditional map component for each particle in the above set
    $$P(\mathbf{m}|\mathbf{X}_{0:k}^{(i)}, \mathbf{Z}_{0:k}) = \prod_{j=1}^{n} P(\mathbf{m}_j|\mathbf{X}_{0:k}^{(i)}, \mathbf{Z}_{0:k})$$
- Particle filter theory is derived from a recursive form of sampling known as ***sequential importance sampling*** (SIS)
    - Sampling recursively via the product rule rather than directly sampling the joint state

    $$P(\mathbf{x}_0, \mathbf{x}_1, \dots, \mathbf{x}_T|\mathbf{Z}_{0:T}) = P(\mathbf{x}_0|\mathbf{Z}_{0:T})P(\mathbf{x}_1|\mathbf{x}_0, \mathbf{Z}_{0:T})\dots P(\mathbf{x}_T|\mathbf{X}_{0:T-1}, \mathbf{Z}_{0:T})$$

- General form of R-B particle filter for SLAM
    - Assume the joint state is represented as $\{w_{k-1}^{(i)}, \mathbf{X}_{0:k-1}^{(i)}, P(\mathbf{m}|\mathbf{X}_{0:k-1}^{(i)}, \mathbf{Z}_{0:k-1})\}_i^N$ at time $k-1$
    - Step 1: draw a sample from a proposal distribution conditioned on specific particle history

    $$\mathbf{x}_k^{(i)} \sim \pi(\mathbf{x}_k|\mathbf{X}_{0:k-1}^{(i)}, \mathbf{Z}_{0:k}, \mathbf{u}_k)$$

    - Implicitly update the particle history: $\mathbf{X}_{0:k}^{(i)} \triangleq \{\mathbf{X}_{0:k-1}^{(i)}, \mathbf{x}_k^{(i)}\}$
    - Step 2: compute sample weight

    $$w_k^{(i)} = w_{k-1}^{(i)} \frac{P(\mathbf{z}_k|\mathbf{X}_{0:k}^{(i)}, \mathbf{Z}_{0:k-1})P(\mathbf{x}_k^{(i)}|\mathbf{x}_{k-1}^{(i)}, \mathbf{u}_k)}{\pi(\mathbf{x}_k^{(i)}|\mathbf{X}_{0:k}^{(i)}, \mathbf{Z}_{0:k}^{(i)})}$$

    - Numerator terms: ***observation model*** (map info being marginalized) & ***motion model***
        - Map marginalization for *observation model*:

    $$P(\mathbf{z}_k|\mathbf{X}_{0:k}, \mathbf{Z}_{0:k-1}) = \int P(\mathbf{z}_k|\mathbf{x}_k, \mathbf{m})P(\mathbf{m}|\mathbf{X}_{0:k-1}\mathbf{Z}_{0:k-1})\mathrm{d}\mathbf{m}$$

    - Step 3: perform resampling if necessary
        - Select particles with replacement from the set $\{\mathbf{X}_{0:k}^{(i)}\}_i^N$ and set uniform weight $w_k^{(i)} = 1/N$
    - Step 4: update observed landmarks using EKF as a simple mapping function with known vehicle pose

Notes on SLAM Tutorial Part II (Bailey and Durrant-Whyte 2006) (paper)

1. Focus

- · Tutorial focus: recursive Bayesian formulation of the SLAM problem where probability distributions/estimates of absolute/relative landmark locations & vehicle poses are obtained
- · 3 Key areas in SLAM research in this tutorial:
    - a. Computational complexity
        - · Approaches to reduce the computational complexity
            - · Linear-time state augmentation
            - · Sparsification in information form
            - · Partitioned updates
            - · Submapping methods
    - b. Data association
        - · Methods to correctly associate landmark observations with landmarks in the map
            - · Batch-validation methods
            - · Appearance-based methods
            - · Multihypothesis techniques
    - c. Environment representation
        - · Related to appearance-based models of landmarks and maps
            - · Delayed mapping
            - · Use of nongeometric landmarks
            - · Trajectory estimation methods
- · Use mathematical notations defined in SLAM Part I tutorial

2. Computational Complexity

- · Exploit the structure to limit computational complexity of SLAM algorithm
    - · Process/Motion model $P(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{u}_k)$: only affect vehicle pose states
    - · Observation model $P(\mathbf{z}_k|\mathbf{x}_k, \mathbf{m})$: only make reference to single vehicle-landmark pair
- · Category of techniques:

· **_Optimal_**: _reduce_ required computation and get estimates & covariances that are **_equal to_** the full-form SLAM algorithm
· **_Conservative_**: computationally more efficient but **_less accurate_**
  · Termed **_inconsistent_** & regarded as **_invalid solutions_** to the SLAM (or any estimation) problem
· Methods:
  · State augmentation:
    · Re-formulating time-update equations
    · Optimal
  · Partitioned updates:
    · Re-formulating observation-update equations
    · Optimal
  · Sparsification in information form
    · Re-formulating standard state-space SLAM representation into information form
    · Usually conservative
  · Submapping methods
    · Provide estimates in global frame
    · Conservative

2.1 State Augmentation

· SLAM state vector $\mathbf{x}_k = [\mathbf{x}_{vk}^T, \mathbf{m}^T]^T$ (where $\mathbf{x}_{vk}$ denotes the vehicle state vector at time k)
· Covariance computation in a naive implementation of EKF-SLAM:

$$\mathbf{P}_{k|k-1} = \nabla\mathbf{f_x}\mathbf{P}_{k-1|k-1}\nabla\mathbf{f_x}^T + \nabla\mathbf{f_u}\mathbf{U}_k\nabla\mathbf{f_u}^T$$

where $\nabla\mathbf{f_x} = \partial\mathbf{f}/\partial\mathbf{x}_{k-1}$, $\nabla\mathbf{f_u} = \partial\mathbf{f}/\partial\mathbf{u}_k$, and $\mathbf{U}_k$ is the covariance characterising uncertainty on control vector
· Exploit the fact that motion model only affects vehicle pose states:

$$\mathbf{P}_{k|k-1} = \begin{bmatrix} \mathbf{P}_{vv} & \mathbf{P}_{vm} \\ \mathbf{P}_{vm}^T & \mathbf{P}_{mm} \end{bmatrix} = \begin{bmatrix} \nabla\mathbf{f}_{v_\mathbf{x}}\mathbf{P}_{vv}\nabla\mathbf{f}_{v_\mathbf{x}}^T + \nabla\mathbf{f}_{v_\mathbf{u}}\mathbf{U}_k\nabla\mathbf{f}_{v_\mathbf{u}}^T & \nabla\mathbf{f}_{v_\mathbf{x}}\mathbf{P}_{vm} \\ \mathbf{P}_{vm}^T\nabla\mathbf{f}_{v_\mathbf{x}}^T & \mathbf{P}_{mm} \end{bmatrix}$$

where $\nabla\mathbf{f}_{v_\mathbf{x}} = \partial\mathbf{f}_v/\partial\mathbf{x}_{v(k-1)}$ and $\nabla\mathbf{f}_{v_\mathbf{u}} = \partial\mathbf{f}_u/\partial\mathbf{u}_k$
· Adding new landmarks to the SLAM state vector at a new state k:

$$\mathbf{x}_k^+ = \begin{bmatrix} \mathbf{x}_{vk} \\ \mathbf{m} \\ \mathbf{m}_{new} \end{bmatrix} = \begin{bmatrix} \mathbf{x}_{vk} \\ \mathbf{m} \\ g(\mathbf{x}_{vk}, \mathbf{z}_k) \end{bmatrix}$$

where the new map landmark(s) are initialized by a function of vehicle pose $\mathbf{x}_{vk}$ and the current observation $\mathbf{z}_k$

· Apply state augmentation whenever new states are a function of a subset of existing states:

$$\mathbf{x}^+ = \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ f(\mathbf{x}_2, \mathbf{q}) \end{bmatrix}$$

$$\mathbf{P}^+ = \begin{bmatrix} \mathbf{P}_{11} & \mathbf{P}_{12} & \mathbf{P}_{12}\nabla\mathbf{f}_{\mathbf{x}_2}^T \\ \mathbf{P}_{12}^T & \mathbf{P}_{22} & \mathbf{P}_{22}\nabla\mathbf{f}_{\mathbf{x}_2}^T \\ \nabla\mathbf{f}_{\mathbf{x}_2}\mathbf{P}_{12}^T & \nabla\mathbf{f}_{\mathbf{x}_2}\mathbf{P}_{22} & \nabla\mathbf{f}_{\mathbf{x}_2}\mathbf{P}_{22}\nabla\mathbf{f}_{\mathbf{x}_2}^T + \nabla\mathbf{f}_{\mathbf{q}}\mathbf{Q}\nabla\mathbf{f}_{\mathbf{q}}^T \end{bmatrix}$$

where the dimensions are:

$$\dim \mathbf{x}^+ = \begin{bmatrix} a \times 1 \\ b \times 1 \\ (b+c) \times 1 \end{bmatrix}$$

$$\dim \mathbf{P}^+ = \begin{bmatrix} a \times a & a \times b & a \times (b+c) \\ b \times a & b \times b & b \times (b+c) \\ (b+c) \times a & (b+c) \times b & (b+c) \times (b+c) \end{bmatrix}$$

2.2 Partitioned Updates

· Instead of updating **all vehicle & map states** every time when a new measurement is made, only update a **small local region** of states and update the global map only at a **much slower frequency**
· 2 basic types:
    · Type 1: operating on a local region of the global map and maintains globally referenced coordinates
        · Key methods:
            · Compressed EKF (CEKF)
            · Postponement algorithm
    · Type 2: generating a short-term submap with its own local coordinate frame
        · Key methods:
            · Constrained local/relative submap filter (CLSF/CRSF)
            · Local map sequencing algorithm
        · Simpler & more numerically stable & less affected by linearization errors
· CLSF/CRSF:
    · Maintain 2 independent SLAM estimates at all times:

$$\mathbf{x}_G = \begin{bmatrix} \mathbf{x}_F^G \\ \mathbf{m}_G \end{bmatrix}, \quad \mathbf{x}_R = \begin{bmatrix} \mathbf{x}_v^R \\ \mathbf{m}_R \end{bmatrix}$$

        · $\mathbf{x}_G$: map composed of global reference pose of a submap coordinate frame $\mathbf{x}_F^G$ and a set of global referenced landmarks $\mathbf{m}_G$
        · $\mathbf{x}_R$: local submap with a locally referenced vehicle pose $\mathbf{x}_v^R$ and locally referenced landmarks $\mathbf{m}_R$
    · Conventional SLAM updates are performed entirely within the local submap

- Only those landmarks held in the local submap will be updated
- Obtain global vehicle pose estimate at any time by **vector summation** of *locally referenced pose* and the *global estimate* of the submap coordinate frame
- Optimal global estimate is obtained periodically by **registering the submap with the global map** and **applying constraint updates on any common features of both maps**
- Advantages:
  - At any one instance, the number of landmarks that must be updated is limited to those described in the local submap coordinate frame only
    - Observation-rate update is independent of the total map size
    - Full update can be performed in a background task at a much lower rate as long as the observation-rate update can be performed
  - Uncertainty is lower in a locally referenced frame
    - Linearization error is reduced
  - Association robustness can be improved by using *batch-validation gating* in submap registration

2.3 Sparsification

- Represent Gaussian probability density in canonical/information form rather than using the 1st & 2nd moment $\hat{\mathbf{x}}_k$ & $\mathbf{P}_k$
  - Information matrix $\mathbf{Y}_k = \mathbf{P}_k^{-1}$
  - Information vector $\hat{\mathbf{y}}_k = \mathbf{Y}_k \hat{\mathbf{x}}_k$
- Many off-diagonal components of the *normalized* information matrix are very close to 0
- An exactly sparse solution can be obtained by augmenting the state vector with new vehicle pose estimate at every time step and retaining all past robot poses:
  - $\mathbf{x}_k = [\mathbf{x}_{vk}^T, \mathbf{x}_{vk-1}^T, ..., \mathbf{x}_{v1}^T, \mathbf{m}^T]^T$
  - Nonzero off-diagonal components: only for poses & landmarks that are directly related by measurement data
  - Control sparsity by utilizing **marginalization** to remove past states
- Caveat: necessary to recover mean & covariance of the state at every time step
  - Mean estimate is required to perform linearization of process & observation models
    - Efficient recovery by conjugate gradient method
  - Both mean & covariance are required to compute validation gate for data association
    - Robust batch gating methods require the full recovery of covariance matrix, which has a **cubic** complexity in the number of landmarks

2.4 Submapping Methods

- Another category of methods to deal with the **quadratically scaling computation complexity** with the number of landmarks during measurement updates
- 2 categories: globally (a) & locally referenced submaps (b)
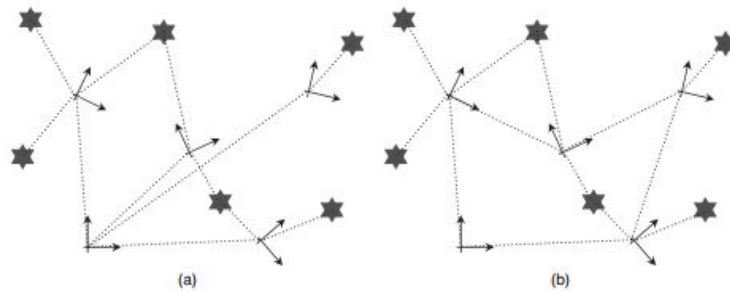
· Global submaps
   · Estimate global locations of submap coordinate frames relative to a common base frame
   · Cannot alleviate linearization issues arising from large pose uncertainties
· Local/Relative submaps
   · Location of any given submap is recorded only by its neighboring submaps
   · Submaps are connected in a graphical network
   · Obtain global estimate by vector summation along a path in the network
   · Can avoid linearization issues because global-level data fusion is eschewed

3. Data Association

· *New measurements* are associated with *existing map landmarks* before fusing data into the map
   · The associations **cannot be revised** after fusion
· Any single incorrect data association can lead to failure of the SLAM algorithm

3.1 Batch Validation

· Individual gating
   · Consider each measurement-to-landmark association individually
   · Test whether an observed landmark is close to a predicted location
   · Extremely unreliable if the vehicle pose is very uncertain
· Batch gating
   · Multiple associations are considered simultaneously
   · Exploit geometric relationship between landmarks

3.2 Appearance Signatures

· Extract landmark appearance features as its signature
· Use **similarity metrics** such as an image similarity metric to predict a possible association

3.3 Multihypothesis Data Association

· Essential for robust target tracking in cluttered environment
· Generate separate track estimate for each association hypothesis, and create an ever-branching tree of tracks over time
· Prune low-likelihood tracks from the hypothesis tree

4. Environment Representation

· Assumption in early SLAM works: the world can be reasonably modeled as ***a set of simple discrete landmarks*** described by ***geometric primitives*** such as points, lines or circles
  · Assumption does not hold for more complex & unstructured environments

4.1 Partial Observability and Delayed Mapping

· Limitations on sensing modality
  · Sonar: produce accurate range measurement but unusable bearing estimate
  · Vision (camera): provide bearing info without accurate range indication
· Single measurement (range-only / bearing-only sensors) is ***insufficient*** to constrain a landmark location
  · Need be observed from multiple vantage points (e.g., the intersection of regions mapped by multiple measurements)
  · Single measurement can only generate a non-Gaussian distribution over landmark location
    · Example for a Gaussian landmark estimate: central position & radius
· Delayed mapping: obtaining Gaussian landmark estimate from range-only / bearing-only sensors by ***delaying initialization*** & ***accumulating raw measurement data***
  · Record vehicle pose for each deferred measurement
    · Augment SLAM states with recorded vehicle pose estimates $(\mathbf{x}_k = [\mathbf{x}_{v_k}^T, \mathbf{x}_{v_{k-1}}, ..., \mathbf{x}_{v_{k-n}}^T, \mathbf{m}^T]^T)$
    · Store corresponding measurements in an auxiliary list $(\{\mathbf{z}_k, ..., \mathbf{z}_{k-n}\})$
  · Initialize a landmark by a batch update once sufficient information is accumulated
  · Remove recorded poses that do not have any associated measurements
· Delayed fusion
  · General concept for increasing robustness by ***accumulating information*** and ***permitting delayed decision making***
  · Advantages:
    · Reduce linearization errors by performing a batch update (such as *bundle adjustment* (BA)) on accumulated data set
    · Facilitate batch validation gating & aid reliable data association

4.2 Nongeometric Landmarks

· EKF-SLAM is usually applied to geometric landmarks
· Nongeometric landmarks: landmarks with arbitrary shape
  · Described by a shape model with an embedded coordinate frame whose origin defines the landmark origin
· EKF-SLAM with nongeometric landmarks: separate shape parameter estimation from landmark locations

4.3 3-D SLAM

· 3 essential forms

    a. 2-D SLAM with additional map building ability in 3rd dimension

    b. Direct extension of 2-D SLAM

    c. SLAM with joint state containing history of past vehicle poses

       · Obtaining 3-D environment scan at each pose

4.4 Trajectory-Oriented SLAM

· Estimate ***vehicle trajectory*** instead of pose + landmark

· Type 1:

    · Joint state $\mathbf{x}_k$: $[\mathbf{x}_{vk}^T, \mathbf{x}_{v_{k-1}}^T, \dots, \mathbf{x}_{v_1}^T]$

    · Particularly suited to environments where

       · Discrete landmarks are not easily discerned

       · Direct alignment of sensed data is easier / more reliable

    · Map is not part of the state but forms an auxiliary data set

       · Map is formed by aligning associated scan of sensed data from each pose estimate

· Type 2 (Consistent Pose Estimation (CPE)):

    · Connect pose in a graphical network instead of forming a joint state vector

· Type 3:

    · Sparse-information-form SLAM with sparse estimation of joint state vector of type 1

· Caveats:

    · Unbounded state-space & quantity of stored measurement data

       · Necessary to coalesce data to bound storage costs

4.5 Embedded Auxiliary Information

· Soil salinity, humidity, temperature, terrain characteristics, etc…

· May be used to assist mapping, aid data association or for tasks unrelated to mapping (path planning / data gathering)

· More difficult to be incorporated within traditional SLAM framework

4.6 Dynamic Environments

· Dynamic landmarks:

    · Moving objects such as people

    · Temporary structures that appear static but later moved such as chairs / parked cars

· Property:

    · Landmarks can be removed from the map without loss of consistency

       · Large number of landmarks can be removed without change of convergence rate

· Ideas:

    · Removing landmarks that are obsolete (due to environment changes)

    · Implement auxiliary identification routine to remove dynamic info from a data scan before sending it to SLAM algorithm

    · Tracking both stationary & moving targets

       · Reduce cost by implementing stationary SLAM update followed by separate tracking of moving objects

Notes on SLAM Survey (Cadena et al. 2016) (site|paper)

1. Introduction

- · SLAM development:
    - · Classical age (1986 - 2004): covered in SLAM tutorial I (Durrant-Whyte and Bailey 2006) & II (Bailey and Durrant-Whyte 2006)
    - · Algorithmic-analysis age (2004 - 2015): covered in (Dissanayake et al. 2011 (link))
    - · Robust-perception age (2015 -)
        - · Robust performance
        - · High-level understanding
        - · Resource awareness
        - · Task-driven inference
- · 2 questions:
    - a. Do autonomous robots really need SLAM?
        - · Visual-inertial Navigation (VIN) *is* SLAM
            - · A *reduced* SLAM system without loop closure / place recognition module
        - · Advantage of SLAM: getting real topology of the environment by enabling loop closure
            - · Offers a natural defense against wrong data association & perceptual aliasing, which will deceive place recognition
            - · Key to robust operation
        - · Many applications implicitly/explicitly require a globally consistent map
    - b. Is SLAM solved?
        - · Need to specify the robot/environment/performance combination
            - · Robot
                - · Type of motion
                - · Available sensors
                - · Available computational resources
            - · Environment
                - · Dimension: planar or three-dimensional
                - · Presence of natural/artificial landmarks
                - · Amount of dynamic elements
                - · Amount of symmetry

· Risk of perceptual aliasing

· May depend on the sensor-environment pair

· Performance requirements

· Desired accuracy of the robot state estimation

· Accuracy & type of representation of the environment (landmark-based or dense)

· Success rate

· Estimation latency

· Maximum operation time

· Maximum size of mapped area

2. Anatomy of a Modern SLAM System

· 2 parts of SLAM system
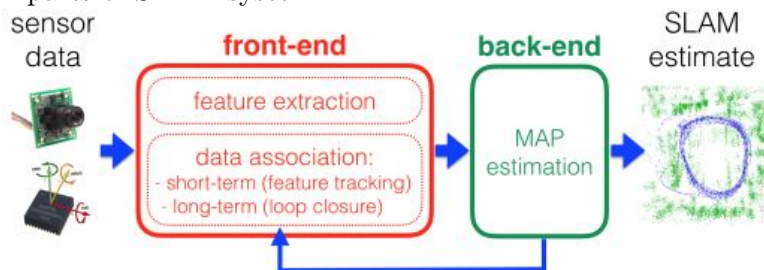
· Front-end: abstract sensor data into estimation models

· Data association module

· Short-term data block: associate corresponding features in consecutive sensor measurements

· Long-term data block (loop closure): associate new measurements to older landmarks

· Pre-processing: sensor dependent

· Back-end: perform inference on abstracted data from front-end

· Feed feedback info to the front-end to support loop closure detection & validation

2.1 SLAM Back-end

2.1.1 Fisher Information Matrix

· Abbreviated as information matrix

- Fisher information $\mathcal{I}(\theta)$ over parameter θ:

$$\mu = \mathrm{E}[\frac{\partial}{\partial\theta}\log f(X;\theta)|\theta]$$
$$= \int \frac{\partial}{\partial\theta}(\log f(X;\theta))f(X;\theta)\mathrm{d}X$$
$$= \int \frac{\frac{\partial}{\partial\theta}f(X;\theta)}{f(X;\theta)}f(X;\theta)\mathrm{d}X$$
$$= \frac{\partial}{\partial\theta}\int f(X;\theta)\mathrm{d}X$$
$$= \frac{\partial}{\partial\theta}1$$
$$= 0$$
$$\mathcal{I}(\theta) = \mathrm{E}[(\frac{\partial}{\partial\theta}\log f(X;\theta) - \mu)^2|\theta]$$
$$= \mathrm{E}[(\frac{\partial}{\partial\theta}\log f(X;\theta))^2|\theta]$$

- μ: 1st moment / expected value of the "score" (i.e., the partial derivative with respect to θ of the log-likelihood function $f(X;\theta)$)
- Fisher information: 2nd central moment / variance of log-likelihood function $f(X;\theta)$
- If $\log f(X;\theta)$ is twice differentiable w.r.t. θ, then under certain regularity conditions

$$\mathcal{I}(\theta) = \mathrm{E}[(\frac{\partial}{\partial\theta}\log f(X;\theta))^2|\theta] = -\mathrm{E}[(\frac{\partial^2}{\partial\theta^2}\log f(X;\theta)|\theta]$$

- Fisher information matrix: Fisher information of a $N{\times}N$ vector of parameter $\theta = [\theta_1, \theta_2, ..., \theta_N]^T$
  - Element (i,j) of matrix $\mathcal{J}$

$$[\mathcal{J}(\theta)]_{i,j} = \mathrm{E}[(\frac{\partial}{\partial\theta_i}\log f(X;\theta))(\frac{\partial}{\partial\theta_j}\log f(X;\theta))|\theta]$$

  - dim $\mathcal{J} = N \times N$
  - Example

2.1.2 Maximum-a-posteriori Estimation

- Maximum-a-posteriori (MAP) estimation as SLAM formulation
  - Formulate SLAM as MAP estimation problem
  - Use *factor graphs* to reason about the interdependence among variables
    - Example: $g(X_1, X_2, X_3) = f_1(X_1)f_2(X_1, X_2)f_3(X_1, X_2)f_4(X_2, X_3)$
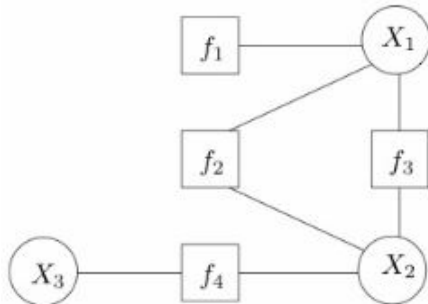
- Math notations:

- X: robot trajectory (discrete set of poses) & landmark positions in the environment
- $Z = \{z_k : k = 1, \dots, m\}$: set of measurements
- $z_k = h_k(X_k) + \epsilon_k$, where $X_k \subseteq X$, $h_k(\cdot)$ being the observation/measurement model (known *nonlinear* function)
- MAP estimation: estimate X by computing variables X^* that attains the maximum of the posterior $P(X|Z)$ (the *belief* over X given the measurements)

$$X^* \triangleq \arg\max_X P(X|Z) = \arg\max_X \frac{P(Z|X)P(X)}{\int_X P(Z|X)P(X)dX} = \arg\max_X P(Z|X)P(X)$$

  - The denominator of the posterior distribution (marginal likelihood) plays no role in the optimization because it is always positive & does not depend on X
- Assume the measurements are independent (i.e., the noise terms are not correlated with the measurements)

$$X^* = \arg\max_X P(X) \prod_{k=1}^m P(z_k|X) = \arg\max_X P(X) \prod_{k=1}^m P(z_k|X_k) \qquad \text{(3-1)}$$

- Assume ϵ_k is a zero-mean Gaussian noise with information matrix Ω_k, and the prior is normally distributed

$$P(z_k|X_k) \propto \exp(-\frac{1}{2}\|h_k(X_k) - z_k\|^2_{\Omega_k}) \qquad \text{(3-2)}$$

$$P(X) \propto \exp(-\frac{1}{2}\|h_0(X) - z_0\|^2_{\Omega_k}) \qquad \text{(3-3)}$$

  - $P(z_k|X_k)$: measurement likelihood
  - $P(X)$: the prior
  - $\|e\|^2_\Omega = e^T \Omega e$
- According to (3-2) and (3-3), (3-1) becomes the cost function below (nonlinear (generalized) least squares problem):

$$
\begin{aligned}
X^* &= \arg\min_X (-\log(P(X) \prod_{k=1}^m P(z_k|X_k))) \\
&= \arg\min_X \sum_{k=0}^m \frac{1}{2}\|h(X_k) - z_k\|^2_{\Omega_k}
\end{aligned}
\qquad \text{(3-4)}
$$

  - Derived from the assumption of normally distributed noise
  - Different assumptions for the noise distribution lead to different cost functions
- Similarity & differences between (3-4) & bundle adjustment (BA):
  - Similarity: both stem from MAP formulation
  - Differences:
    - Factors in (3-4) are not constrained to model projective geometry in BA but include a broad variety of sensor models
    - (3-4) is designed to be solved *incrementally* because new measurements are made available at each time step as the robot moves
- Minimization problem of (3-4) is solved via successive *linearizations* using Gauss-Newton (GN) or Levenberg-Marquardt or other modern methods

· GN approximate (3-4) as

$$\delta_X^* = \arg\min_{\delta_X} \frac{1}{2}\sum_{k=0}^{m} \|A_k\delta_X - b_k\|_{\Omega_k}^2 = \arg\min_{\delta_X} \frac{1}{2}\|A\delta_X - b\|_{\Omega_k}^2 \qquad (3\text{-}5)$$

· δ_X: a small "correction" w.r.t. the linearization point $\hat{X}$

· $A_k = \dfrac{\partial h_k(X)}{\partial X}$: the Jacobian of the measurement function $h_k(\cdot)$ w.r.t. X

· $b_k \triangleq z_k - h_k(\hat{X})$: the *residual error* at $\hat{X}$

· A & b are obtained by stacking A_k & b_k

  · dim X = dim $\delta_X \triangleq n \times 1$

  · dim $A_k = n \times n$

  · dim b_k = dim $\delta_X = n \times 1$

  · Stacking:

    · dim $A = ((m+1) \times n) \times n$

    · dim $b = ((m+1) \times n) \times 1$

· Ω_k: measurement information matrix

  · dim $\Omega_k = n \times n$

· Ω: block diagonal matrix obtained by stacking Ω_k as its diagonal blocks

  · dim $\Omega = ((m+1) \times n) \times ((m+1) \times n)$

· Optimal correction δ_X^* to minimize (3-5)

$$\delta_X^* = (A^T\Omega A)^{-1} A^T\Omega b \qquad (3\text{-}6)$$

· $A^T\Omega A$: the *Hessian*

· Update of linearization point: at each iteration,

  · $\hat{X} \leftarrow \hat{X} + \delta_X^*$ if X belongs to a vector space

  · $\hat{X} \leftarrow \hat{X} \oplus \delta_X^*$ if X includes variables belonging to a smooth manifold

  · δ_X^* belongs to the tangent space of X

  · Operator $\oplus$: retraction operator that maps the vector δ_X^* in the tangent space of manifold at $\hat{X}$ to an element of the manifold

    · The mapping from one tangent space to another one

    · Example: move the point on a sphere by a small amount, and the result point will have a different tangent space (2D plane) with the original point

    · Tangent space illustration where the sphere can be seen as a 2D manifold
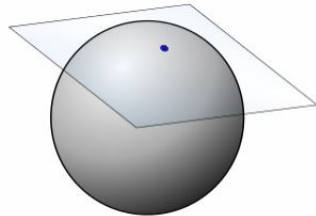
2.1.3 Summary

· Key insights behind SLAM solvers

· Jacobian matrix A in (3-6) is **sparse** and its sparsity structure is dictated by ***the topology***

        of underlying factor graph
- · Enable the use of fast linear solvers to compute δ_X^*
- · Allow the designing of *incremental/online* solvers (update the estimate of X as new observations are acquired)
- · Current SLAM libraries:
  - · GTSAM, g2o, Ceres, iSAM, and SLAM++
- · Modern SLAM formulation (variant names):
  - · Maximum-a-posteriori estimation, factor graph optimization, graph-SLAM, full smoothing, or smoothing and mapping (SAM)
- · Nonlinear filtering (covered in Section 1 & Section 2) vs. MAP estimation
  - · MAP estimation is more accurate & efficient than Nonlinear filtering
  - · Performance mismatch between Nonlinear filtering & MAP estimation gets smaller for nonlinear filtering methods if:
    - · Linearization point for EKF becomes more accurate
    - · Sliding-window filters are used
    - · Potential sources of inconsistency of EKF are taken care of
- · Why referred to as SLAM ***back-end***?
  - · MAP estimation is performed on pre-processed sensor data

2.2 SLAM Front-end

- · Motivation
  - · Hard to directly write the sensor measurements as an analytic state function required in MAP estimation
  - · Unable to design a sufficiently general yet tractable environment representation
- · Sensor-dependent SLAM front-end
  - · Set up a module called ***front-end*** before the SLAM back-end
  - · Features
    - · Extract relevant features from sensor data
    - · Associate each measurement to a specific landmark (*data association*)
    - · Provide an initial guess for the variables in the nonlinear optimization (3-4)

3. Long-term Autonomy I: Robustness

- · SLAM system failure
  - · Algorithmic
    - · Limitation of current SLAM algorithms
  - · Hardware-related
    - · Sensor/actuator degradation
  - · Software-related
- · Algorithmic robustness challenges
  - · Data association failures
    - · Feature-based visual SLAM: *perceptual aliasing* $\rightarrow$ wrong measurement-state matches (false positives) $\rightarrow$ wrong estimates from the back-end

- Unmodeled dynamics in the environment → deceive short/long-term data association
  - Typical SLAM system: system based on *static world assumption*
  - Examples:
    - Drastic illumination changes → failure in system relying on the repeatability of visual features
    - Disappearances of old structures → failure in system relying on methods leveraging the geometry of the environment
- Harsh environment with limited visibility & constantly changing conditions & impossibility of using conventional sensors
- Solutions for data association failures
  - Make front-end reliable for data association establishment
    - Short-term data association: ensure sensors are of high sampling rate (high framerate)
      - Viewpoint of the sensor does not change significantly from time t to $t + 1$
    - Long-term data association
      - Involving loop closure *detection* and *validation*
      - Loop closure detection
        - Detect features in environment and then try to match them with all previously detected features
        - Use searching algorithms with low computational complexity: ***Bag-of-words***
          - Quantizing feature space for more efficient search
        - Improve bag-of-words based techniques to handle severe illumination variations
      - Loop closure validation
        - Ascertain the quality of loop closures using additional geometric steps
        - Vision-based application: *RANSAC* is used for geometric verification & outlier rejection
        - Laser-based application: check residual error of laser scan matching
  - Make back-end resilient against spurious measurements
    - Spurious measurements (wrong loop closures): caused by perceptual aliasing
    - Methods:
      - During optimization: reason on the loop closure constraints validity by looking at the residual error induced by the constraints
      - Before optimization: identify incorrect loop closures that are not supported by the odometry
  - Make SLAM system better deal with dynamic environments
    - Challenge 1: SLAM system has to detect, discard/track changes
      - Include dynamic elements as part of the model
    - Challenge 2: SLAM system has to model permanent/semi-permanent changes & understand how and when to update the map accordingly
      - Maintain multiple (time-dependent) maps of the same location
- Open problems
  a. Failsafe SLAM and recovery
    - Current SLAM solvers (based on iterative optimization of nonconvex costs) are vulner-

able to outliers

· Goal: *fail-safe* & *failure-aware*

· Be aware of imminent failures & provide recovery mechanisms that can re-establish proper operation

b. Robustness to HW failure

· Goal: detect degraded sensor operation & adjust sensor noise statistics (covariances & biases) accordingly

c. Metric relocalization: estimate relative pose w.r.t. previously built map

· Hard/impossible for feature-based methods (rely on visual sensors) to detect loop closures among different times (day/night/diff. season/...)

· Localizing from significantly different viewpoints with heterogeneous sensing

d. Time varying and deformable maps

· Limitations on mainstream SLAM methods: rigid & static world assumption

· Ideal SLAM method: be able to deal with environment dynamics

e. Automatic parameter tuning

· SLAM systems require extensive parameter tuning for a specific scenario

· Especially for data association module

· Parameters:

· Thresholds to control feature matching

· RANSAC parameters

· Criteria to decide when to add new factors to the graph or when to trigger a loop closing algorithm to search for matches

· Goal: automatic parameter tuning for arbitrary scenarios

4. Long-term Autonomy II: Scalability

· SLAM methods should be designed to be scalable to the specific scenario

· Different scenarios require different computational time & memory footprint

· Computational & memory complexity should remain bounded

· Approaches to control/reduce growth of SLAM problem size

· ***Sparsification*** methods: trade-off information loss for memory & computational efficiency

· Node & edge sparsification

· ***Reducing*** number of nodes *added* to the graph, or ***pruning*** less "informative" nodes & factors to address scalability

· ***Out-of-core*** & ***multi-robot*** methods: split computation among many robots/processors

· Out-of-core (parallel) SLAM

· Split computation & memory load of factor graph optimization among multiple processors

· Key idea: global factor graph optimization $\rightarrow$ localized subgraph optimization + global refinement

· Called *submapping* algorithms

· Distributed multi robot SLAM

· Divide the scenario in smaller areas by deploying multiple robots

  · Map each area by a different robot

 · Two main variants

  · *Centralized* method: robots build submaps & transfer local info to a central station that performs inference

  · *Decentralized* method: no central data fusion & agents leverage local communication to reach consensus on a common estimate

· Open problems

 a. Map maintenance

  · Localization against a compressed known map

  · Memory-efficient dense reconstruction

 b. Robust distributed mapping

  · Outliers are ignored in multi-robot SLAM

  · Challenges

   · Robots may not share a common reference frame which makes it harder to detect & reject wrong loop closures

   · Robots have to detect outliers from very partial/local information

 c. Learning, forgetting, remembering

  · Question for long-term mapping: determine the followings

   · How often to update the map information

   · How to decide when the information becomes outdated & can be discarded

   · When is it fine to forget

   · What can be forgotten & what is essential to maintain

 d. Resource-constrained platforms

  · How to adapt existing SLAM algorithms for robot platforms with severe computational constraints

  · How to guarantee reliable operation for multi robot teams given tight bandwidth constraints & communication dropout

5. Representation I: Metric Reasoning

· ***Metric representation*** (metric map): symbolic structure that encodes the geometry of the environment

· Geometric modeling in SLAM in 2D case

 · *Landmark-based maps*: model the environment as a sparse set of landmarks

 · *Occupancy grid maps*: discretize the environment in cells and assign a probability of occupation to each cell

· Geometric modeling in SLAM in 3D case

 a. Landmark-based sparse representations

  · Represent scene as a set of sparse 3D landmarks corresponding to discrimitive features (lines/corners/…) in the environment

  · Referred to as landmark-based / feature-based representations

  · Application:

   · Localization & mapping in mobile robotics

- · *Structure from motion* in computer vision
  - · Assumption: landmarks are distinguishable
    - · Sensor data measure geometric aspect of a landmark & provide a descriptor for data association between measurement & landmark
  - · Features: point (mainly), lines, segments, or arcs
- b. Low-level raw dense representations
  - · Provide high-resolution models for 3D geometry
  - · Application
    - · Obstacles avoidance & path planning in robotics
    - · Visualization & rendering
  - · Raw representations
    - · Large unstructured set of points: **point clouds**
      - · In conjunction with stereo & RGB-D cameras, and 3D laser scanners
    - · Polygons: **polygon soup**
    - · **Surfels maps**: encode the geometry as a set of disks
  - · Application in monocular SLAM: in conjunction with the use of *direct methods*
    - · Estimate robot trajectory & 3D model directly from image pixel intensity values
  - · Pros & cons
    - · Pro: visually pleasant
    - · Cons:
      - · Cumbersome (require storing of large amount of data)
      - · Low-level geometry description which neglects obstacle topology
- c. Boundary and spatial-partitioning dense representations
  - · Attempt to explicitly represent surfaces/*boundaries* and volumes
  - · Boundary representations (b-reps): define 3D objects in terms of their surface boundary
    - · Plane-based models
    - · *Curve-based representations*: tensor product of NURBS or B-splines
    - · *Surface mesh models*: connected set of polygons
    - · *Implicit surface representations*
      - · Specify the surface of a solid as the zero crossing of a function defined on $\mathbb{R}^3$
      - · Function examples
        - · Radial-basis functions
        - · Signed-distance function
        - · **Truncated signed-distance function (TSDF)**
  - · Spatial-partitioning representations
    - · Spatial-occupancy enumeration: decompose 3D space into identical cubes (voxels) arranged in a regular 3D grid
    - · Octree: used for 3D mapping
      - · Occupancy grid maps: probabilistic variants of space-partitioning representations
    - · Polygonal Map octree
    - · Binary Space-partitioning tree
    - · 2.5D elevation maps

d. High-level object-based representations
- · Features of object & solid shape representations
  - · Model objects as 3D objects rather than 1D (points) & 2D (surfaces)
  - · Associate physical notions (volume, mass, …) to each object
- · Solid representations in CAD & computer graphics (no applications in SLAM yet) (intro)
  - · *Parameterized Primitive Instancing*
    - · Define a families of objects (cylinder, sphere, …)
    - · Define a set of parameters (radius, height, …) to each member of the family
  - · *Sweep representations* (see the intro above)
    - · Define a solid as the sweep of a 2D/3D object along a trajectory through space
  - · *Constructive solid geometry* (see the intro above)
- · Feature-based models in CAD
- · Dictionary-based representations
  - · Define a solid as a combination of atoms in a dictionary
  - · Dictionary source:
    - · Learned from data
    - · Existing repositories of object models
- · Affordance-based models
- · Generative & procedural models
- · Scene graphs
· Sparse (feature-based) vs. dense representations/algorithms in visual SLAM
- · Disadvantages of sparse representations
  - · Dependence of feature type
  - · Reliance on numerous detection & matching thresholds
  - · Necessity for robust estimation methods to deal with wrong correspondences
  - · Most feature detectors are optimized for speed rather than precision
- · Dense/direct methods
  - · Exploit all image information
  - · Outperform feature-based methods in scenes with poor texture, defocus & motion blur
  - · Require high computing power for real-time performance
  - · Currently unable to jointly estimate dense structure & motion
- · Alternatives to feature-based methods
  - · *Semi-dense* methods: only exploit pixels with strong gradients
  - · *Semi-direct* methods: leverage both sparse features & direct methods
    - · Most efficient alternative
    - · Allow joint estimation of structure & motion
· Open problems
a. High-level, expressive representations in SLAM
- · Point clouds or TSDF are mainly used to model 3D geometry
  - · Drawbacks
    - · Wasteful for environment modeling
    - · No high-level understanding of 3D geometry

- Advantages of higher-level representations
  - Compact representations → compressed map in large-scale mapping
  - Higher-level object description facilitates data association, place recognition & semantic understanding
  - Enable interactions with existing modern building construction & management standards
- Current representations in SLAM are only low-level point clouds, mesh models, surfels models, and TSDFs

b. Optimal representations
  - Lack of criteria to choose an appropriate representation for specific environment

c. Automatic & adaptive representations
  - Choice of representation is entrusted to SLAM system designer
    - Drawbacks
      - Representation design is time-consuming & require an expert
      - No flexibility: representation cannot be changed once the system is designed

6. Representation II: Semantic Reasoning

- Features of semantic mapping
  - Associate semantic concepts to geometric entities in robot's surroundings
  - Enhance robot's autonomy & robustness
  - Facilitate more complex tasks (e.g. avoid muddy-road while driving)
  - Move from path-planning to task-planning
  - Enable advanced human-robot interaction
  - Typical semantic parsing: formulated as a classification problem
    - Simple mapping between sensory data & semantic concepts
- Semantic vs. topological SLAM
  - Semantic: interested in classifying the places according to semantic labels
  - Topological: do place recognition only for topological mapping (disregard the actual function of a place (kitchen/corridor/...)
- Design of a semantic representation
  - Level/detail of semantic concepts: determined by specific tasks
  - Organization of semantic concepts
    - Flat/hierarchical organization
    - Sharing/not sharing properties
- Semantic mapping in SLAM
  - SLAM helps semantics
    - Segmenting the map built from classical SLAM systems into semantic concepts
  - Semantics helps SLAM
    - Improve map estimation by recognizing semantic concepts in the map
  - Joint SLAM & semantics inference
    - Perform monocular SLAM & map segmentation within a joint formulation
- Open problems

a. Semantic SLAM lacks cohesive formulation

b. Semantic mapping is much more than a categorization problem
  · How to represent interrelationships among semantic concepts

c. Ignorance, awareness & adaption
  · Given prior knowledge, a robot should be able to reason about new concepts & their semantic representations
    · Discover new objects
    · Learn new properties
    · Adapt representations to slow/abrupt changes in the environment

d. Semantic-based reasoning
  · Robots are currently unable to effectively localize & continuous map using semantic concepts in the environment
    · Unable to update & refine map by detecting new semantic concepts

7. New Theoretical Tools for SLAM

· Focus
  · Factor graph optimization approaches
  · *Algorithmic properties*
· MAP estimation → MLE if no prior is given
  · SLAM inherits all properties of maximum likelihood estimators
  · MLE properties
    · Consistent
    · Asymptotically Gaussian & asymptotically efficient
    · Invariant to transformations in the Euclidean space
  · Estimator will no longer be invariant if priors are given
· SLAM: *nonconvex* problem
  · Iterative optimization can only guarantee local convergence
  · Local minimum → completely wrong estimation & unsuitable for navigation
  · State-of-the-art iterative solvers fail to converge to global minimum for relatively small noise levels
· Progress on theoretical analysis of SLAM problem
  · Rotation estimation can be solved uniquely in closed form in 2D
  · Convergence improvement approaches
    · (Convex) dual ***semidefinite programming (SDP)***: under certain conditions often encountered in practice (strong duality), ML estimation is unique & pose graph optimization can be solved globally via this approach
    · Convex relaxation: avoid local minima convergence
    · Compute a suitable initialization for iterative nonlinear optimization
      · Effective idea: solve for rotations first, and then use the resulting estimate to bootstrap nonlinear iteration
  · Verification techniques to judge whether the quality of given SLAM solution is optimal or not

· Based on Lagrangian duality in SLAM
· Current progress: perform verification by solving a sparse linear system and provide a correct answer as long as strong duality holds
· Open problems
    a. Generality, guarantees, verification
        · Can guaranteed solution & verification techniques in context of pose graph optimization be generalized to arbitrary factor graphs?
    b. Weak or strong duality?
        · SLAM can be solved globally when strong duality holds
        · Given a set of sensors & a factor graph, does the strong duality hold?
    c. Resilience to outliers
        · Design of global techniques & verification techniques that are resilient to outliers remains open

8. Active SLAM

· Active SLAM
    · Control robot's motion to minimize the uncertainty of its map representation & localization
    · Other names:
        · Simultaneous Planning Localization and Mapping (SPLAM)
        · Autonomous/Active/Integrated Exploration
· General frameworks for decision making in active SLAM
    · Goal: solve for exploration-exploitation decisions
    · Frameworks
        · Theory of Optimal Experimental Design (TOED)
            · Allow selecting future robot motion based on predicted map uncertainty
        · Information theoretic approaches
            · Decision making is guided by *information gain*
        · Control theoretic approaches
            · Use Model Predicative Control
        · Partially Observably Markov Decision Process (POMDP)
            · Computationally intractable
        · Bayesian Optimization
        · Efficient Gaussian beliefs propagation
· Framework based on selecting future robot motion among a finite set of alternatives
    · Step 1 - *selecting vantage points*: robot identifies possible locations (vantage locations) to explore/exploit
        · Only a small subset of locations are selected as evaluation grows exponentially with the search place
        · Related approaches only guarantee local optimal policies
    · Step 2 - *computing the utility of an action*: robot computes the utility of visiting each vantage point and select the action with highest utility
        · Computationally intractable for precise information gain of each action

- · Compute approximations: utility defined as linear combination of metrics that quantify robot & map uncertainties
  - · Drawback: scale of numerical uncertainties not comparable
  - · TOED can also be applied
- · Step 3 - *executing actions or terminate exploration*: robot carries out selected action and decides if necessary to continue/terminate the task
  - · Open challenge for the decision whether exploration task is done
- · Open problems
  - a. Fast and accurate predictions of future states
    - · Necessary because each robot action should reduce the uncertainty in map & improve localization accuracy in active SLAM
    - · Loop closure is important: no efficient methods exist
    - · Predicting effects of future actions is computationally expensive
  - b. Enough is enough: when to stop active SLAM?
    - · Active SLAM is computationally expensive, need to stop and focus resources on other tasks
  - c. Performance guarantees
    - · Need mathematical guarantees for active SLAM & near-optimal policies
      - · Exact solution is computationally intractable
      - · Desirable for approximation algorithms with clear performance bounds

9. New Sensors for SLAM

- · Progress in SLAM are partly triggered by novel sensors
  - · 2D laser range finders: enable very robust SLAM algorithms
  - · 3D lidars: main thrust behind applications such as autonomous cars
  - · Vision sensors: used in augmented reality & vision-based navigation
- · Limitation of current vision sensors
  - · High latency & temporal discretization
    - · Latency: time to capture image & time to process it
    - · Temporal discretization: time to wait until the next frame arrives
  - · Prevent the advent of high-agility autonomous robot
- · ***Event-based cameras***
  - · May substitute current vision sensors
  - · Typical cameras:
    - · Dynamic Vision Sensor (DVS)
    - · Asynchronous Time-based Image Sensor (ATIS)
  - · Event cameras vs. standard frame-based cameras
    - · Standard frame-based cameras: send entire images at fixed frame rates
    - · Event cameras: only send local pixel-level changes caused by movement in a scene at the time it they occur
      - · Event is triggered if relative intensity change exceeds a threshold
  - · *Microsecond time* resolution for event cameras

- · 5 key advantages over standard frame-based cameras
    - a. Temporal latency: 1 microsecond
    - b. Measurement rate: up to 1MHz
    - c. Dynamic range: up to 140 dB (vs. 60-70 dB of standard cameras)
    - d. Power consumption: 20 mW (vs. 1.5W of standard cameras)
    - e. Bandwidth for data transfer: kilobytes/s scale (vs. Megabytes/s)
- · Disadvantages
    - · Traditional frame-based computer-vision algorithms are not applicable
        - · Sensor output composes of a sequence of asynchronous events instead of image frame
    - · Require a *paradigm shift* from traditional computer vision approaches
- · Progress on event camera applications
    - · Algorithmic adaptions:
        - · Feature detection & tracking
        - · LED tracking
        - · Epipolar geometry
        - · Image reconstruction
        - · Motion estimation
        - · Optical flow
        - · 3D reconstruction
        - · Iterative closest points
    - · Event-based visual odometry, localization & SLAM
        - · Design goal: each incoming event can asynchronously change estimated system state
        - · Adaptions of continuous-time trajectory-estimation methods
            - · Pose trajectory can be *approximated* by a smooth curve in rigid-body motion space using basis functions (cubic splines) & *optimized* according to the observed events
- · Open problems
    - a. Modeling, sensitivity and resolution
        - · Events in event-based cameras are highly susceptible to noise
        - · Lack of full characterization of sensor noise & sensor non idealities
        - · Event pipelines may be failed for scenes with smooth edges as events are triggered by brightness changes
        - · Low spatial resolution (128x128 pixels)
    - b. Event-based or standard camera for SLAM?
        - · No criteria for choosing algorithm-sensor pair

Notes on Multiple View Geometry Book (Hartley and Zisserman 2004) (site)

Part 0 - The Background: Projective Geometry, Transformations and Estimation

2. Projective Geometry and Transformations of 2D

2.2 The 2D Projective Plane

2.2.1 Points and Lines

- · Homogeneous representation
    - · Line in a plane: $\mathbf{l} : ax + by + c = 0 \rightarrow$ vector (a, b, c)
        - · $(ka)x + (kb)y + (kc) = 0$ are the same line for $k \neq 0$
    - · Point: $\mathbf{x} = (x, y) \rightarrow (x, y, 1)$
        - · $(x_1, x_2, x_3) \rightarrow (x_1/x_3, x_2/x_3)$ in $\mathbb{R}^2$ for $x_3 \neq 0$
- · Point on a line: $\mathbf{x} \cdot \mathbf{l} = 0$ or $\mathbf{x}^T \mathbf{l} = 0$
- · Degrees of freedom (dof): both point & line are 2
- · (Point) Intersection of lines: $\mathbf{x} = \mathbf{l} \times \mathbf{l}'$
    - · $\mathbf{l} \cdot (\mathbf{l} \times \mathbf{l}') = \mathbf{l}' \cdot (\mathbf{l} \times \mathbf{l}') = 0$
        - · $\mathbf{l}^T \mathbf{x} = \mathbf{l}'^T \mathbf{x} = 0$
- · Line passing through 2 points $\mathbf{x}$ & $\mathbf{x}'$: $\mathbf{l} = \mathbf{x} \times \mathbf{x}'$
    - · $\mathbf{x} \cdot \mathbf{l} = \mathbf{x}' \cdot \mathbf{l} = 0$

2.2.2 Ideal Points and the Line at Infinity

- · Intersection of parallel lines $\mathbf{l} = (a, b, c)$ & $\mathbf{l}' = (a, b, c')$
    - · $\mathbf{l} \times \mathbf{l}' = (c' - c)(b, -a, 0)$
        - · Intersection: point $(b, -a, 0)$
- · Ideal point & the line at infinity
    - · Homogeneous vectors $\mathbf{x} = (x_1, x_2, x_3)$
        - · Vectors with $x_3 \neq 0$: finite points in $\mathbb{R}^2$
        - · Vectors with any x_3: set of all homogeneous 3-vectors, i.e., projective space $\mathbb{P}^2$

· Ideal point: $(b, -a, 0)$

　　· Set of ideal points: $(x_1, x_2, 0)$ specified by the ratio $x_1 : x_2$

· Line at infinity: $\mathbf{l}_\infty = (0, 0, 1)$

· Inhomogeneous notation $(b, -a)$: represent line's direction

· Projective plane model
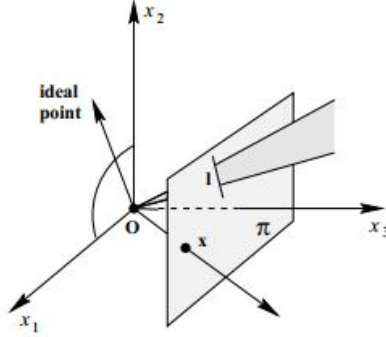
Fig. 2.1. **A model of the projective plane.** *Points and lines of* $\mathbb{P}^2$ *are represented by rays and planes, respectively, through the origin in* $\mathbb{R}^3$. *Lines lying in the* x_1x_2-*plane represent ideal points, and the* x_1x_2-*plane represents* $\mathbf{l}_\infty$.

· $\mathbb{P}^2 \to \mathbb{R}^3$

· $k(x_1, x_2, x_3)$

　　· Point in $\mathbb{P}^2$: ray through the origin

　　· Line in $\mathbb{P}^2$: plane passing through the origin

· 2 points form 1 line → 2 rays are on exactly one plane

· 2 lines intersect at 1 point → 2 planes intersect in one ray

· Duality

　　· Symmetry between line & point in homogeneous representation

　　　　· $\mathbf{l}^T \mathbf{x} = 0 \leftrightarrow \mathbf{x}^T \mathbf{l} = 0$

　　· ***Duality principle***: To any theorem of 2-dimensional projective geometry there corresponds a dual theorem, which may be derived by interchanging the roles of points and lines in the original theorem

　　　　· Concepts of incidence must be appropriately translated

　　　　· Example: line through 2 points $\leftrightarrow$ point through (point of intersection of) 2 lines

2.3 Projective Transformations

· Projectivity: an invertible mapping $h : \mathbb{P}^2 \to \mathbb{P}^2$ such that 3 points $\mathbf{x}_1$, $\mathbf{x}_2$, and $\mathbf{x}_3$ lie on the same line if and only if $h(\mathbf{x}_1)$, $h(\mathbf{x}_2)$, and $h(\mathbf{x}_3)$ do

　　· Synonym: *collineation, projective transformation, homography*

　　　　· Collineation: collinear points remain collinear after the mapping

· A mapping $h : \mathbb{P}^2 \to \mathbb{P}^2$ is a ***projectivity*** if and only if there exists a non-singular matrix $\mathbf{H}$ such that for any point in $\mathbb{P}^2$ represented by a vector $\mathbf{x}$ it is true that $h(\mathbf{x}) = \mathbf{H}\mathbf{x}$

　　· $\mathbf{x}' = \mathbf{H}\mathbf{x}$ where $\mathbf{x}$ and $\mathbf{x}'$ are homogeneous 3-vectors

$$\begin{pmatrix} x_1' \\ x_2' \\ x_3' \end{pmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}$$

　　· $\mathbf{H}$: *homogeneous* matrix (only ratio of matrix elements is significant)

　　　　· DOF: 8 because there are 8 ratios among 9 elements of $\mathbf{H}$

2.3.1 Transformations of Lines and Conics

· Transformation of lines under mapping $\mathbf{H}$

$$\mathbf{l}' = \mathbf{H}^{-T}\mathbf{l}$$

    · $\mathbf{x}$ lies on line $\mathbf{l}$ and $\mathbf{x}' = \mathbf{H}\mathbf{x}$ lies on line $\mathbf{l}'$
      · $\mathbf{x}'^T\mathbf{l}' = \mathbf{x}^T\mathbf{l} = 0$
        $\mathbf{x}^T\mathbf{H}^T\mathbf{l}' = \mathbf{x}^T\mathbf{l}$
        $\mathbf{l}' = \mathbf{H}^{-T}\mathbf{l}$

2.4 A Hierarchy of Transformations

· Projective transformations form a group called *projective linear group*
· $GL(n)$: (real) general linear group on n dimensions
    · Group of invertible $n \times n$ matrices with real elements
· $PL(n)$: projective linear group
    · Quotient group of $GL(n)$ (matrices related by a scalar multiplier)
· Important subgroups of $PL(3)$:
    · ***Affine group***: matrices whose last row is $(0, 0, 1)$
    · ***Euclidean group***: subgroup of affine group
      · Upper left hand 2×2 matrix has determinant 1
· Subsequent 4 classes of transformations are from most specialized *class I* to most generalized *class IV* transformations

2.4.1 Class I: Isometries

· Isometries are transformations of plane $\mathbb{R}^2$
· Euclidean distance is preserved
· Representation:

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{bmatrix} \epsilon\cos\theta & -\sin\theta & t_x \\ \epsilon\sin\theta & \cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

    · $\epsilon = \pm 1$
    · When $\epsilon = 1$:
      · Isometry is orientation-preserving
      · Isometry is an Euclidean transformation (composition of translation & rotation)
    · When $\epsilon = -1$: orientation is reversed
· Euclidean transformations model the motion of a *rigid object*

$$\mathbf{x}' = \mathbf{H}_E\mathbf{x} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \mathbf{x}$$

    · $\mathbf{R}$: 2×2 rotation matrix (orthogonal matrix: $\mathbf{R}^T\mathbf{R} = \mathbf{R}\mathbf{R}^T = \mathbf{I}$)
    · $\mathbf{t}$: translation 2-vector
    · $\mathbf{0}$: null 2-vector

· Special cases
  · Pure translation ($\mathbf{R} = \mathbf{I}$)
  · Pure rotation ($\mathbf{t} = \mathbf{0}$)
· Also known as ***displacement***
· DOF for planar Euclidean transformation: 3 (1 + 2)
  · 1 for rotation & 2 for translation
  · Transformation can be computed from 2 point correspondences
· Invariants
  · Length: distance between 2 points
  · Angle: angle between 2 lines
  · Area
· Orientation-preserving transformations form a group while orientation-reversing ones do not

2.4.2 Class II: Similarity Transformations

· Similarity transformation (*similarity*): isometry composed with isotropic scaling
· Representation in case of Euclidean composed with scaling (no reflection) (matrix & block matrix form)

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{bmatrix} s\cos\theta & -s\theta & t_x \\ s\sin\theta & s\theta & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \quad \mathbf{x}' = \mathbf{H}_s\mathbf{x} = \begin{bmatrix} s\mathbf{R} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \mathbf{x}$$

  · *s*: isotropic scaling
· Also known as *equi-form* transformation as it preserves "shape" (form)
· DOF for planar similarity transformation: 4 (1 + (1 + 2))
  · 1 for scaling and 3 for isometry
· Invariants
  · Angle: parallel lines are mapped to parallel lines
  · *Ratio* of 2 lengths (before & after the transformation)
  · *Ratio* of 2 areas
· *Metric structure*: the structure is defined up to a similarity

2.4.3 Class III: Affine Transformations

· Affine transformation (*affinity*): non-singular linear transformation followed by a translation
· Representation (matrix & block matrix form)

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{bmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \quad \mathbf{x}' = \mathbf{H}_A\mathbf{x} = \begin{bmatrix} \mathbf{A} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \mathbf{x}$$

  · $\mathbf{A}$: 2×2 non-singular matrix
· DOF for planar affine transformation: 6
  · 4 for transformation $\mathbf{A}$ & 2 for translation
  · Transformation can be computed from 3 point correspondences

· Decomposition of **A** (using SVD decomposition):

$$\mathbf{A} = \mathbf{U}\mathbf{D}\mathbf{V}^T$$
$$= \mathbf{U}\mathbf{V}^T\mathbf{V}\mathbf{D}\mathbf{V}^T$$
$$= (\mathbf{U}\mathbf{V}^T)(\mathbf{V}\mathbf{D}\mathbf{V}^T)$$
$$= (\mathbf{R}(\theta))(\mathbf{R}(-\phi)\mathbf{D}\mathbf{R}(\phi))$$
$$= (\text{rotation})(\text{deformation})$$

· $\mathbf{R}(\theta)$ & $\mathbf{R}(\phi)$: rotations by θ & ϕ
· **D**: a diagonal matrix representing non-isotropic scaling

$$\mathbf{D} = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}$$

· **U** & **V** are orthogonal matrices
· Affine matrix **A** is concatenation of the followings:
  · Deformation (non-isotropic scaling)
    · A rotation by ϕ
    · A non-isotropic scaling by λ_1 & λ_2 respectively in (rotated) x & y directions
    · A rotation back by ϕ
  · Rotation
    · A rotation by θ
· Non-isotropic scaling (deformation):
  · Contribute 2 extra DOF to affinity over similarity
    · Angle ϕ specifying the scaling direction
    · Ratio of scaling parameters $\lambda_1 : \lambda_2$
· Illustration of rotation & deformation
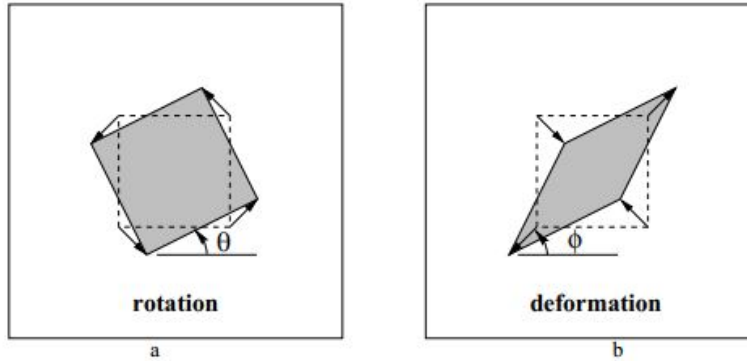
Fig. 2.7. **Distortions arising from a planar affine transformation.** *(a) Rotation by* R(θ). *(b) A deformation* R($-\phi$) D R(ϕ). *Note, the scaling directions in the deformation are orthogonal.*

· Invariants
  · Parallel lines
  · Ratio of lengths of parallel line segments
  · Ratio of areas
· $\det \mathbf{A} = \lambda_1 \lambda_2$
  · Sign of the determinant (scalings) determines whether the affinity is orientation-preserving

(positive sign) or orientation-reversing (negative sign)

2.4.4 Class IV: Projective Transformations

· Projective transformation is a general non-singular linear transformation of **homogeneous** co-ordinates

· Generalization of affinity, which is the composition of a general non-singular linear transformation of **inhomogeneous** coordinates and a translation

· Block form representation

$$\mathbf{x}' = \mathbf{H}_P \mathbf{x} = \begin{bmatrix} \mathbf{A} & \mathbf{t} \\ \mathbf{v}^T & v \end{bmatrix} \mathbf{x}$$

  · $\mathbf{v} = (v_1, v_2)^T$

· DOF: 8

  · 9 elements with only their ratio significant $\rightarrow$ 8 parameters

  · Projective transformation between 2 planes can be computed from 4 point correspondences (no 3 collinear points on either plane)

· Invariants

  · Cross ratio (ratio of ratios) of 4 collinear points

2.4.7 The Number of Invariants

· *"Number"*: counting argument for the number of *functionally independent* invariants

· number of invariants $\geq \text{DOF}_{\text{configuration}} - \text{DOF}_{\text{transformation}}$

· Summary

Group	Matrix	Distortion	Invariant properties
Projective 8 dof	$\begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix}$		Concurrency, collinearity, **order of contact**: intersection (1 pt contact); tangency (2 pt contact); inflections (3 pt contact with line); tangent discontinuities and cusps. cross ratio (ratio of ratio of lengths).
Affine 6 dof	$\begin{bmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix}$		Parallelism, ratio of areas, ratio of lengths on collinear or parallel lines (e.g. midpoints), linear combinations of vectors (e.g. centroids). The line at infinity, $\mathbf{l}_\infty$.
Similarity 4 dof	$\begin{bmatrix} sr_{11} & sr_{12} & t_x \\ sr_{21} & sr_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix}$		Ratio of lengths, angle. The circular points, $\mathbf{I}$, $\mathbf{J}$ (see section 2.7.3).
Euclidean 3 dof	$\begin{bmatrix} r_{11} & r_{12} & t_x \\ r_{21} & r_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix}$		Length, area

Table 2.1. **Geometric properties invariant to commonly occurring *planar* transformations.** *The matrix* A = [a_{ij}] *is an invertible* 2×2 *matrix,* R = [r_{ij}] *is a 2D rotation matrix, and* (t_x, t_y) *a 2D translation. The distortion column shows typical effects of the transformations on a square. Transformations higher in the table can produce all the actions of the ones below. These range from Euclidean, where only translations and rotations occur, to projective where the square can be transformed to any arbitrary quadrilateral (provided no three points are collinear).*

2.7 Recovery of Affine and Metric Properties from Images

2.7.1 The Line at Infinity

- l_∞ remains a fixed line under projective transformation $\mathbf{H}$ if and only if $\mathbf{H}$ is an affinity
- Fixed lines are fixed as a set, not fixed pointwise
  - A point on the line is mapped to another point on this line

2.9 Fixed Points and Lines

- Fixed point/line of a transformation: correspond to an ***eigenvector***
  - $\mathbf{H}\mathbf{e} = \lambda\mathbf{e}$, where $\mathbf{e}$ and $\lambda\mathbf{e}$ represents the same point (eigenvector of $\mathbf{H}$)
  - $\mathbf{H}^{-T}\mathbf{e} = \lambda\mathbf{e}$, where $\mathbf{e}$ and $\lambda\mathbf{e}$ represents the same line (eigenvector of $\mathbf{H}^T$)
- If 2 of the eigenvalues are identical, the corresponding fixed line will be fixed pointwise

3. Projective Geometry and Transformations of 3D

3.1 Points and Projective Transformations

- Point $\mathbf{X}$ in 3-space is represented in homogeneous coordinates as a 4-vector
  - $\mathbf{X} = (X_1, X_2, X_3, X_4)$ with $X_4 \neq 0$ represents the point (X, Y, Z) of $\mathbb{R}^3$ with inhomogeneous coordinates
    - $X = X_1/X_4$, $Y = X_2/X_4$, $Z = X_3/X_4$
- Projective transformation acting on $\mathbb{P}^3$: non-singular 4×4 matrix $\mathbf{H}$: $\mathbf{X}' = \mathbf{H}\mathbf{X}$
  - DOF: 15 (16 elements and less one for overall scaling)

3.2 Representing and Transforming Planes, Lines and Quadrics

- Duality in $\mathbb{P}^3$: points & *planes* are dual
  - Similar to point-line duality in $\mathbb{P}^2$
  - Lines are self-dual in $\mathbb{P}^3$

3.2.1 Planes

- A plane in 3-space may be written as $\pi_1 X + \pi_2 Y + \pi_3 Z + \pi_4 = 0$
  - Homogeneous representation: $\boldsymbol{\pi} = (\pi_1, \pi_2, \pi_3, \pi_4)$
    - DOF: 3
- $\boldsymbol{\pi}^T \mathbf{X} = 0 \rightarrow$ point $\mathbf{X}$ in on plane $\boldsymbol{\pi}$
- $\boldsymbol{\pi} = (\pi_1, \pi_2, \pi_3, \pi_4)$:
  - First 3 components: plane normal of Euclidean geometry
  - Plane equation in inhomogeneous notation: $\mathbf{n} \cdot \tilde{\mathbf{X}} + d = 0$
    - $\mathbf{n} = (\pi_1, \pi_2, \pi_3)$
    - $\tilde{\mathbf{X}} = (X, Y, Z)$ with $X_4 = 1$
    - $d = \pi_4$
    - $d/\|\mathbf{n}\|$: distance of the plane from the origin

· 3 points define a plane

$$\begin{bmatrix} \mathbf{X}_1^T \\ \mathbf{X}_2^T \\ \mathbf{X}_3^T \end{bmatrix} \boldsymbol{\pi} = \mathbf{A}\boldsymbol{\pi} = \mathbf{0}$$

  · The 3×4 matrix has a rank of 3 if the 3 points are not collinear (linearly independent)
  · Plane $\boldsymbol{\pi}$ is in the (right) nullspace (1-dimensional)
    · $\mathbf{A} : \mathbb{R}^4 \rightarrow \mathbb{R}^3$
      · $\mathbf{X}_i \in C(\mathbf{A}^T)$
      · $\boldsymbol{\pi} \in N(\mathbf{A})$
      · $\dim C(\mathbf{A}^T) + \dim N(\mathbf{A}) = 3 + 1 = 4$
    · 1-dimensional: the plane is defined uniquely (up to scale)
· 3 planes define a point

$$\begin{bmatrix} \boldsymbol{\pi}_1^T \\ \boldsymbol{\pi}_2^T \\ \boldsymbol{\pi}_3^T \end{bmatrix} \mathbf{X} = \mathbf{A}\mathbf{X} = 0$$

  · Dual to the case of 3 points define a plane
  · The point: defined in the (right) nullspace (1-dimensional)
· Projective transformation (plane)
  · $\mathbf{X}' = \mathbf{H}\mathbf{X} \rightarrow \boldsymbol{\pi}' = \mathbf{H}^{-T}\boldsymbol{\pi}$

3.2.2 Lines

· DOF of lines in $\mathbb{P}^3$: 4
    · Can be defined by two points on 2 orthogonal planes, 2 DOF for each point
    · Hard to be represented normally because it needs a homogeneous 5-vector
· Line representations
  a. Nullspace and span representation
      · Line: pencil (1-parameter family) of collinear points
        · Defined by any of these 2 points
          · 3 DOF (homogeneous 4-vector) for each point
      · Line represented as the span of row space of 2×4 matrix $\mathbf{W}$ composed of 2 transposed 4-vectors (point) $\mathbf{A}^T$ & $\mathbf{B}^T$

$$\mathbf{W} = \begin{bmatrix} \mathbf{A}^T \\ \mathbf{B}^T \end{bmatrix}$$

      1) Span of $\mathbf{W}^T$: pencil of points $\lambda\mathbf{A} + \mu\mathbf{B}$ on the line
      2) Span of 2-dimensional $N(\mathbf{W})$ (nullspace) of $\mathbf{W}$: pencil of planes with the line as axis
        · Suppose $\mathbf{P}$ & $\mathbf{Q}$ are a basis of the nullspace
        · $\mathbf{A}^T\mathbf{P} = \mathbf{B}^T\mathbf{P} = \mathbf{0}$: $\mathbf{P}$ is a plane containing the points $\mathbf{A}$ & $\mathbf{B}$
          · Similarly for $\mathbf{Q}$: a distinct plane
        · As $\mathbf{A}$ and $\mathbf{B}$ lie on both (linearly independent) plane $\mathbf{P}$ & $\mathbf{Q}$, the line defined by $\mathbf{W}$ is the plane intersection
        · Any plane of the pencil with the line as axis can be represented by the span

$$\lambda'\mathbf{P} + \mu'\mathbf{Q}$$

· Dual representation: line represented as the span (of row space) of 2×4 matrix $\mathbf{W}^*$ composed of plane $\mathbf{P}^T$ & $\mathbf{Q}^T$:

$$\mathbf{W}^* = \begin{bmatrix} \mathbf{P}^T \\ \mathbf{Q}^T \end{bmatrix}$$

1) Span of $\mathbf{W}^{*T}$: pencil of planes $\lambda'\mathbf{P} + \mu'\mathbf{Q}$ with the line as axis
2) Span of 2-dimensional $N(\mathbf{W}^*)$: pencil of points on the line

· Relation of the 2 representations: $\mathbf{W}^*\mathbf{W}^T = \mathbf{W}\mathbf{W}^{*T} = \mathbf{0}_{2 \times 2}$

  · $\mathbf{0}_{2 \times 2}$: 2×2 null matrix

b. Pluecker matrices

· Line represented by a 4×4 skew-symmetric homogeneous matrix

  · Line $\mathbf{L}$ joining 2 points $\mathbf{A}$ & $\mathbf{B}$

    · Elements: $l_{ij} = A_i B_j - B_i A_j$
    · Vector notation: $\mathbf{L} = \mathbf{AB}^T - \mathbf{BA}^T$

  · Properties of $\mathbf{L}$

  1) Rank of $\mathbf{L}$: 2

    · $\mathbf{A}^T\mathbf{P} = \mathbf{B}^T\mathbf{P} = \mathbf{A}^T\mathbf{Q} = \mathbf{B}^T\mathbf{Q} = \mathbf{0}$

      · $\mathbf{P}$ & $\mathbf{Q}$: distinct planes containing $\mathbf{A}$ & $\mathbf{B}$

    · $\mathbf{LP} = \mathbf{LQ} = \mathbf{0} \rightarrow \mathbf{L}\begin{bmatrix} \mathbf{P} & \mathbf{Q} \end{bmatrix} = \mathbf{0} \rightarrow \mathbf{LW}^{*T} = \mathbf{0}_{4 \times 2}$

      · $\mathbf{P} \in N(\mathbf{L})$ & $\mathbf{Q} \in N(\mathbf{L})$
      · $\dim N(\mathbf{L}) = 1 + 1 = 2$

  2) The representation has the 4 DOF requirement:

    · 6 independent non-zero elements in the 4×4 skew-symmetric matrix (upper/lower triangle) with 5 of their ratios significant in homogeneous representation

    · $\det\mathbf{L} = 0$ & elements satisfy a (quadric) constraint $\rightarrow$ net DOF number: 4

  3) $\mathbf{L} = \mathbf{AB}^T - \mathbf{BA}^T$: generalization to 4-space of the vector product formula $\mathbf{l} = \mathbf{x} \times \mathbf{y}$ of $\mathbb{P}^2$ for a line $\mathbf{l}$ defined by 2 points $\mathbf{x}$ & $\mathbf{y}$ represented by 3-vectors

  4) $\mathbf{L}$ is independent of the points used to define it

    · Choose another point $\mathbf{C}$ on the line containing $\mathbf{A}$ & $\mathbf{B}$ and $\mathbf{C} = \mathbf{A} + \mu\mathbf{B}$

$$\hat{\mathbf{L}} = \mathbf{AC}^T - \mathbf{CA}^T$$
$$= \mathbf{A}(\mathbf{A}^T + \mu\mathbf{B}^T) - (\mathbf{A} + \mu\mathbf{B})\mathbf{A}^T$$
$$= \mathbf{AB}^T - \mathbf{BA}^T$$
$$= \mathbf{L}$$

  5) Under point transformation $\mathbf{X}' = \mathbf{HX}$, the matrix transforms as $\mathbf{L}' = \mathbf{HLH}^T$, a valency-2 tensor (?)

· Dual Pluecker representation $\mathbf{L}^*$ as the intersection of 2 planes $\mathbf{P}$ & $\mathbf{Q}$

  · $\mathbf{L}^* = \mathbf{PQ}^T - \mathbf{QP}^T$
  · Has similar properties of $\mathbf{L}$
  · $\mathbf{X}' = \mathbf{HX} \rightarrow \mathbf{L}^{*\prime} = \mathbf{H}^{-T}\mathbf{L}^*\mathbf{H}^{-1}$

· Join & incidence properties

   1) Plane defined by the join of point $\mathbf{X}$ & line $\mathbf{L}$:

      · $\boldsymbol{\pi} = \mathbf{L}^*\mathbf{X}$

        · $\mathbf{L}^*\mathbf{X} = 0$ if and only if $\mathbf{X}$ is on $\mathbf{L}$

   2) Point defined by the intersection of line $\mathbf{L}$ & plane $\boldsymbol{\pi}$

      · $\mathbf{X} = \mathbf{L}\boldsymbol{\pi}$

        · $\mathbf{L}\boldsymbol{\pi} = \mathbf{0}$ if and only if $\mathbf{L}$ is on $\boldsymbol{\pi}$

c. Pluecker line coordinates

   · 6 non-zero elements of the 4×4 skew-symmetric Pluecker matrix $\mathbf{L}$

      · $\mathcal{L} = \{l_{12}, l_{13}, l_{14}, l_{23}, l_{42}, l_{34}\}$

        · Use l_{42} instead of l_{24} to eliminate negatives in some related formulae

   · Homogeneous 6-vector in $\mathbb{P}^5$

   · $\det\mathbf{L} = 0 \rightarrow l_{12}l_{34} + l_{13}l_{42} + l_{14}l_{23} = 0$

3.4 The Hierarchy of Transformations

· Specializations with additional properties of 3-space transformations over 2-space counterparts (check 2.4.7 for general properties)

Group	Matrix	Distortion	Invariant properties
Projective 15 dof	$\begin{bmatrix} A & t \\ v^{\mathsf{T}} & v \end{bmatrix}$		Intersection and tangency of surfaces in contact. Sign of Gaussian curvature.
Affine 12 dof	$\begin{bmatrix} A & t \\ 0^{\mathsf{T}} & 1 \end{bmatrix}$		Parallelism of planes, volume ratios, centroids. The plane at infinity, π_∞, (see section 3.5).
Similarity 7 dof	$\begin{bmatrix} sR & t \\ 0^{\mathsf{T}} & 1 \end{bmatrix}$		The absolute conic, Ω_∞, (see section 3.6).
Euclidean 6 dof	$\begin{bmatrix} R & t \\ 0^{\mathsf{T}} & 1 \end{bmatrix}$		Volume.

Table 3.2. **Geometric properties invariant to commonly occurring transformations of 3-space.** *The matrix A is an invertible 3×3 matrix, R is a 3D rotation matrix, $\mathbf{t} = (t_X, t_Y, t_Z)^{\mathsf{T}}$ a 3D translation, $\mathbf{v}$ a general 3-vector, v a scalar, and $\mathbf{0} = (0,0,0)^{\mathsf{T}}$ a null 3-vector. The distortion column shows typical effects of the transformations on a cube. Transformations higher in the table can produce all the actions of the ones below. These range from Euclidean, where only translations and rotations occur, to projective where five points can be transformed to any other five points (provided no three points are collinear, or four coplanar).*

· DOF (class I $\rightarrow$ IV) in $\mathbb{P}^3$

   · Euclidean: 6 (3 + 3)

· 3 for rotation

· 3 for translation

· Similarity: 7 (1 + 6)

· 1 for isotropic scaling

· 6 for Euclidean

· Affinity: 12 (9 + 3)

· 9 for transformation $\mathbf{A}$

· 3 for translation $\mathbf{t}$

· Projectivity: 15

· 16 elements with only their ratio significant $\rightarrow$ 15 parameters

3.5 The Plane at Infinity

· Plane at infinity: $\boldsymbol{\pi}_\infty = (0,0,0,1)$ in affine 3-space

· 2 lines are parallel if and only if their line intersection is on $\boldsymbol{\pi}_\infty$

· A line is parallel to another line/plane if the point of intersection is on $\boldsymbol{\pi}_\infty$

· $\boldsymbol{\pi}_\infty$ is a fixed plane under projective transformation $\mathbf{H}$ if and only if $\mathbf{H}$ is an affinity

· $\boldsymbol{\pi}_\infty$ is in general only fixed under an affinity as a set; it is not fixed pointwise

· There may be additional planes fixed under a particular affinity, but only $\boldsymbol{\pi}_\infty$ is fixed under any affinity

4. Estimation - 2D Projective Transformations

· Some concrete estimation problems

a. 2D homography

· Given a set of points $\mathbf{x}_i$ & corresponding $\mathbf{x}'_i$ in $\mathbb{P}^2$, compute projective transformation that maps each $\mathbf{x}_i$ to $\mathbf{x}'_i$

b. 3D to 2D camera projection

· Given a set of points $\mathbf{X}_i$ in 3D space and a set of corresponding points $\mathbf{x}_i$ in an image, find the 3D to 2D projective mapping that maps $\mathbf{X}_i$ to $\mathbf{x}_i$ carried out by a projective camera

c. Fundamental matrix computation

· Given a set of points $\mathbf{x}_i$ in one image, and corresponding set of points $\mathbf{x}'_i$ in another image, compute the fundamental matrix $\mathbf{F}$ consistent with these correspondences

· Fundamental matrix $\mathbf{F}$: a singular 3×3 matrix satisfying $\mathbf{x}_i^{T\prime}\mathbf{F}\mathbf{x}_i = 0$ for all i

d. Trifocal tensor computation

· Given a set of point correspondences $\mathbf{x}_i \leftrightarrow \mathbf{x}'_i \leftrightarrow \mathbf{x}''_i$ across 3 images, compute the trifocal tensor

· Trifocal tensor $\mathcal{T}_i^{jk}$: a tensor relating points or lines in 3 views

· Focus on 2D homography problem in this chapter

4.1 The Direct Linear Transformation (DLT) Algorithm

· $\mathbf{x}'_i = \mathbf{H}\mathbf{x}_i \rightarrow \mathbf{x}'_i \times \mathbf{H}\mathbf{x}_i = \mathbf{0}$

$$\mathbf{x}'_i \times \mathbf{H}\mathbf{x}_i = \begin{pmatrix} y'_i\mathbf{h}^{3T}\mathbf{x}_i - w'_i\mathbf{h}^{2T}\mathbf{x}_i \\ w'_i\mathbf{h}^{1T}\mathbf{x}_i - x'_i\mathbf{h}^{3T}\mathbf{x}_i \\ x'_i\mathbf{h}^{2T}\mathbf{x}_i - y'_i\mathbf{h}^{1T}\mathbf{x}_i \end{pmatrix} = \begin{pmatrix} \mathbf{0}^T\mathbf{h}^1 - w'_i\mathbf{x}_i^T\mathbf{h}^2 + y'_i\mathbf{x}_i^T\mathbf{h}^3 \\ w'_i\mathbf{x}_i^T\mathbf{h}^1 + \mathbf{0}^T\mathbf{h}^2 - x'_i\mathbf{x}_i^T\mathbf{h}^3 \\ -y'_i\mathbf{x}_i^T\mathbf{h}^1 + x'_i\mathbf{x}_i^T\mathbf{h}^2 + \mathbf{0}^T\mathbf{h}^3 \end{pmatrix}$$

$$\mathbf{A}_i\mathbf{h} = \begin{bmatrix} \mathbf{0}^T & -w'_i\mathbf{x}_i^T & y'_i\mathbf{x}_i^T \\ w'_i\mathbf{x}_i^T & \mathbf{0}^T & -x'_i\mathbf{x}_i^T \\ -y'_i\mathbf{x}_i^T & x'_i\mathbf{x}_i^T & \mathbf{0}^T \end{bmatrix} \begin{pmatrix} \mathbf{h}^1 \\ \mathbf{h}^2 \\ \mathbf{h}^3 \end{pmatrix} = \mathbf{0} \tag{4-1}$$

· $\mathbf{x}'_i = (x'_i, y'_i, w'_i)$

· Notations for (4-1)

$$\mathbf{H} = \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix} = \begin{pmatrix} \mathbf{h}^{1T} \\ \mathbf{h}^{2T} \\ \mathbf{h}^{3T} \end{pmatrix}, \quad \mathbf{h} = \begin{pmatrix} \mathbf{h}^1 \\ \mathbf{h}^2 \\ \mathbf{h}^3 \end{pmatrix} \tag{4-2}$$

· **h**: a 9-vector

· $\mathbf{h}^{jT}$: a vector denoting the j-th row of **H**

· Basic DLT for 2D homography problem (without normalization)

Objective

Given $n \geq 4$ 2D to 2D point correspondences $\{\mathbf{x}_i \leftrightarrow \mathbf{x}'_i\}$, determine the 2D homography matrix H such that $\mathbf{x}'_i = \mathbf{H}\mathbf{x}_i$.

Algorithm

(i) For each correspondence $\mathbf{x}_i \leftrightarrow \mathbf{x}'_i$ compute the matrix $\mathbf{A}_i$ from (4.1). Only the first two rows need be used in general.

(ii) Assemble the n 2×9 matrices $\mathbf{A}_i$ into a single $2n \times 9$ matrix A.

(iii) Obtain the SVD of A (section A4.4(p585)). The unit singular vector corresponding to the smallest singular value is the solution **h**. Specifically, if $\mathbf{A} = \mathbf{UDV}^T$ with D diagonal with positive diagonal entries, arranged in descending order down the diagonal, then **h** is the last column of V.

(iv) The matrix H is determined from **h** as in (4.2).

· Over-determined solution

· No exact solution $\rightarrow$ find an approximate solution:

· A vector **h** that minimizes a cost function

· Minimize $\|\mathbf{A}\mathbf{h}\|$ with constraint $\|\mathbf{h}\| = 1$

Part 1 - Camera Geometry and Single View Geometry

6. Camera Models

· Camera: a mapping between the 3D world (object space) and a 2D image

6.1 Finite Cameras

· The basic pinhole model:
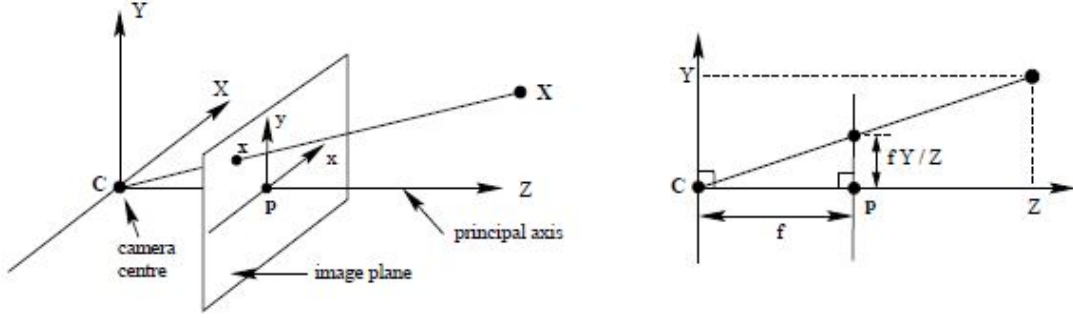
Fig. 6.1. **Pinhole camera geometry.** $\mathbf{C}$ *is the camera centre and* $\mathbf{p}$ *the principal point. The camera centre is here placed at the coordinate origin. Note the image plane is placed in front of the camera centre.*

- · $\mathbf{X}$: 3D point $\mathbf{X} = (X, Y, Z)$
- · $\mathbf{x}$: 2D point on image/focal plane $Z = f$
- · $\mathbf{p}$: *principle point* (where the principle axis meets the image plane)
- · (X, Y, Z) mapped to $(fX/Z, fY/Z, f)$ on image plane
  - · An Euclidean 3-space $\mathbb{R}^3$ to Euclidean 2-space $\mathbb{R}^2$ mapping

$$(X, Y, Z) \mapsto (fX/Z, fY/Z) \tag{6-1}$$

· Central projection using homogeneous coordinates
  · Written (6-1) as

$$\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \mapsto \begin{pmatrix} fX \\ fY \\ Z \end{pmatrix} = \begin{bmatrix} f & & & 0 \\ & f & & 0 \\ & & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{6-2}$$

  · (6-2) can be written compactly as

$$\mathbf{x} = \mathbf{P}\mathbf{X}$$

  · Camera matrix $\mathbf{P} = \mathrm{diag}(f, f, 1) \left[\, \mathbf{I} \,\middle|\, \mathbf{0} \,\right]$

· Principle point offset
  · For principle point at (p_x, p_y) in the image plane, the 3D $\rightarrow$ 2D mapping becomes $(X, Y, Z) \mapsto (fX/Z + p_x, fY/Z + p_y)$

$$\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \mapsto \begin{pmatrix} fX + Zp_x \\ fY + Zp_y \\ Z \end{pmatrix} = \begin{bmatrix} f & & p_x & 0 \\ & f & p_y & 0 \\ & & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{6-3}$$

where

$$\mathbf{K} = \begin{bmatrix} f & & p_x \\ & f & p_y \\ & & 1 \end{bmatrix}$$

is called *camera calibration matrix*

49

· (6-3) has the concise form

$$\mathbf{x} = \mathbf{K} \left[\ \mathbf{I} \ \middle| \ \mathbf{0} \ \right] \mathbf{X}_{\text{cam}} \tag{6-5}$$

  · $\mathbf{X}_{\text{cam}} = (X, Y, Z, 1)$ is in *camera coordinate frame*
· Camera rotation and translation
    · Let $\tilde{\mathbf{X}}$ & $\tilde{\mathbf{X}}_{\text{cam}}$ be inhomogeneous 3-vectors representing coordinates of a point in *world coordinate frame* and *camera coordinate frame*, respectively
      · $\tilde{\mathbf{X}}_{\text{cam}} = \mathbf{R}(\tilde{\mathbf{X}} - \tilde{\mathbf{C}})$
          · $\tilde{\mathbf{C}}$ represents the coordinate of the camera center in the world coordinate frame
        · $\mathbf{R}$: 3×3 rotation matrix representing the orientation of the camera coordinate frame
  · Homogeneous version:

$$\mathbf{X}_{\text{cam}} = \begin{bmatrix} \mathbf{R} & -\mathbf{R}\tilde{\mathbf{C}} \\ 0 & 1 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} = \begin{bmatrix} \mathbf{R} & -\mathbf{R}\tilde{\mathbf{C}} \\ 0 & 1 \end{bmatrix} \mathbf{X} \tag{6-6}$$

  · Putting (6-5) and (6-6) together:

$$\mathbf{x} = \mathbf{K}\mathbf{R} \left[\ \mathbf{I} \ \middle| \ -\tilde{\mathbf{C}} \ \right] \mathbf{X}$$

    · $\mathbf{X}$: coordinate in the world coordinate frame
    · $\mathbf{P} = \mathbf{K}\mathbf{R} \left[\ \mathbf{I} \ \middle| \ -\tilde{\mathbf{C}} \ \right]$: general pinhole camera
  · DOF for $\mathbf{P}$: 9
    · 3 for $\mathbf{K}$ (f, p_x, p_y), called *internal* camera parameters, or *internal orientation* of the camera
    · 3 for $\mathbf{R}$, and 3 for $\tilde{\mathbf{C}}$, which are called *external* parameters, or the *exterior orientation*
  · The world to image transformation can be represented as $\tilde{\mathbf{X}}_{\text{cam}} = \mathbf{R}\tilde{\mathbf{X}} + \mathbf{t}$, and then the camera matrix can be

$$\mathbf{P} = \mathbf{K} \left[\ \mathbf{R} \ \middle| \ \mathbf{t} \ \right] \tag{6-8}$$

  where $\mathbf{t} = -\mathbf{R}\tilde{\mathbf{C}}$
· CCD Cameras
    · Possibly having non-square pixels
    · General form of calibration matrix

$$\mathbf{K} = \text{diag}(m_x, m_y, 1) \begin{bmatrix} f & & p_x \\ & f & p_y \\ & & 1 \end{bmatrix} = \begin{bmatrix} \alpha_x & & x_0 \\ & \alpha_y & y_0 \\ & & 1 \end{bmatrix}$$

  where
    · m_x & m_y: number of pixels in per unit distance in image coordinates in the x and y directions
    · $\alpha_x = fm_x$ & $\alpha_y = fm_y$ represent camera focal length in terms of pixel dimensions in x and y direction respectively
    · $\tilde{\mathbf{x}}_0 = (x_0, y_0) = (m_x p_x, m_y p_y)$: principal point

50

· DOF: 10
· Finite projective camera
  · Add a *skew* parameter s to the calibration matrix

$$\mathbf{K} = \begin{bmatrix} \alpha_x & s & x_0 \\ & \alpha_y & y_0 \\ & & 1 \end{bmatrix} \tag{6-10}$$

  · Finite projective camera: a camera (6-8) with calibration matrix $\mathbf{K}$ of the form (6-10)
  · DOF: 11, same with a 3×4 matrix defined up to arbitrary scale
    · The set of camera matrices of *finite projective cameras* is **identical** with the set of *homogeneous* 3×4 *matrices* for which the left hand 3×3 submatrix is **non-singular**
  · Transformation from world coordinate $\mathbf{X} = (X, Y, Z, 1)$ to pixel coordinate $w\mathbf{x} = (wx, wy, w)$ (ref):

$$w\mathbf{x} = \mathbf{KR} \left[\, \mathbf{I} \, \middle| \, -\tilde{\mathbf{C}} \, \right] \mathbf{X}$$

· General projective camera
  · Arbitrary homogeneous 3×4 matrices with rank 3
  · DOF: 11
  · Left-hand 3×3 submatrix needs not be non-singular

Part 4 - N-View Geometry

18. N-View Computational Methods

· Topic: methods for estimating a projective/affine reconstruction from a set of images
  · Number of views is large
· Contents
  · Bundle adjustment (BA) for projective reconstruction (general case)
  · Specialization of BA to affine cameras $\rightarrow$ factorization algorithm
  · Generalization of the factorization algorithm to non-rigid scenes
  · Specialization of BA for scenes containing planes
  · Methods for obtaining point correspondences throughout image sequence and projective reconstruction from these correspondences

18.1 Projective Reconstruction - Bundle Adjustment

· Formulation
  · A set of 3D points $\mathbf{X}_j$ is viewed by a set of cameras with matrices $\mathbf{P}^i$
  · $\mathbf{x}_j^i$: coordinate of j-th point as seen by i-th camera
  · Reconstruction problem: given the set of image coordinates $\mathbf{x}_j^i$, find the set of camera matrices $\mathbf{P}^i$ and the points $\mathbf{X}_j$, such that $\mathbf{P}^i \mathbf{X}_j = \mathbf{x}_j^i$
    · Projective reconstruction if no constraints given on $\mathbf{P}^i$ or $\mathbf{X}_j$
      · $\mathbf{X}_j$ may differ by an arbitrary 3D projective transformation from true reconstruction
· Bundle adjustment

· $\mathbf{x}_j^i = \mathbf{P}^i\mathbf{X}_j$ will not be satisfied exactly if image measurements are noisy
· Maximum Likelihood (ML) solution: assume measurement noise is Gaussian
    · Estimate $\hat{\mathbf{P}}^i$ and $\hat{\mathbf{X}}_j$ such that $\hat{\mathbf{x}}_j^i = \hat{\mathbf{P}}^i\hat{\mathbf{X}}_j$
    · Minimize image distance between reprojected point $\hat{\mathbf{x}}_j^i$ and detected/measured point $\mathbf{x}_j^i$ for every view where 3D point appears

$$\min_{\hat{\mathbf{P}}^i,\hat{\mathbf{X}}_j} \Sigma_{i,j} d(\hat{\mathbf{P}}^i\hat{\mathbf{X}}_j, \mathbf{x}_j^i)^2 \tag{18-1}$$

        · $d(\mathbf{x}, \mathbf{y})$: geometric image distance between homogeneous point $\mathbf{x}$ and $\mathbf{y}$
    · Bundle adjustment: the above estimation involving minimizing the reprojection error
        · Adjusting the bundle of rays between each camera center and the set of 3D points
· Should generally be used as a final step of any reconstruction algorithm
· Pros & cons
    · + Tolerance of missing data while providing true ML estimate
    · + Allow individual covariances (or more general PDFs) assignment to each measurement
    · + May be extended to include the following estimates: priors, constraints on camera parameters, or point positions
    · - Require good initialization
    · - Can become extremely large optimization problem as the number of parameters involved increases

18.2 Affine Reconstruction - the Factorization Algorithm

· Algorithm

Objective

Given $n \geq 4$ image point correspondences over m views $\mathbf{x}_j^i$, $j = 1,\ldots,n$; $i = 1,\ldots,m$, determine affine camera matrices $\{\mathtt{M}^i, \mathbf{t}^i\}$ and 3D points $\{\mathbf{X}_j\}$ such that the reprojection error

$$\sum_{ij} \left\| \mathbf{x}_j^i - (\mathtt{M}^i \mathbf{X}_j + \mathbf{t}^i) \right\|^2$$

is minimized over $\{\mathtt{M}^i, \mathbf{t}^i, \mathbf{X}_j\}$, with $\mathtt{M}^i$ a 2×3 matrix, $\mathbf{X}_j$ a 3-vector, and $\mathbf{x}_j^i = (x_j^i, y_j^i)^{\mathsf{T}}$ and $\mathbf{t}^i$ are 2-vectors.

Algorithm

(i) **Computation of translations.** Each translation $\mathbf{t}^i$ is computed as the centroid of points in image i, namely

$$\mathbf{t}^i = \langle \mathbf{x}^i \rangle = \frac{1}{n} \sum_j \mathbf{x}_j^i.$$

(ii) **Centre the data.** Centre the points in each image by expressing their coordinates with respect to the centroid:

$$\mathbf{x}_j^i \leftarrow \mathbf{x}_j^i - \langle \mathbf{x}^i \rangle.$$

Henceforth work with these centred coordinates.

(iii) **Construct the $2m \times n$ measurement matrix** $\mathtt{W}$ from the centred data, as defined in (18.5), and compute its SVD $\mathtt{W} = \mathtt{UDV}^{\mathsf{T}}$.

(iv) Then the matrices $\mathtt{M}^i$ are obtained from the first three columns of $\mathtt{U}$ multiplied by the singular values:

$$\begin{bmatrix} \mathtt{M}^1 \\ \mathtt{M}^2 \\ \vdots \\ \mathtt{M}^m \end{bmatrix} = \begin{bmatrix} \sigma_1 \mathbf{u}_1 & \sigma_2 \mathbf{u}_2 & \sigma_3 \mathbf{u}_3 \end{bmatrix}.$$

The vectors $\mathbf{t}^i$ are as computed in step (i) and the 3D structure is read from the first three columns of $\mathtt{V}$

$$\begin{bmatrix} \mathbf{X}_1 & \mathbf{X}_2 & \ldots & \mathbf{X}_n \end{bmatrix} = \begin{bmatrix} \mathbf{v}_1 & \mathbf{v}_2 & \mathbf{v}_3 \end{bmatrix}^{\mathsf{T}}.$$

· $\hat{\mathbf{W}}$: rank 3 matrix closest to $\mathbf{W}$ in Frobenius norm
  · $\hat{\mathbf{W}} = \mathbf{U}_{2m \times 3} \mathbf{D}_{3 \times 3} \mathbf{V}_{3 \times n}^T = \hat{\mathbf{M}} \hat{\mathbf{X}}$
· Choice of $\hat{\mathbf{M}}$ & $\hat{\mathbf{X}}$ is not unique
  · E.g., $\hat{\mathbf{M}} = \mathbf{U}_{2m \times 3} \mathbf{D}_{3 \times 3}$ & $\hat{\mathbf{X}} = \mathbf{V}_{3 \times n}^T$; or $\hat{\mathbf{M}} = \mathbf{U}_{2m \times 3}$ & $\hat{\mathbf{X}} = \mathbf{D}_{3 \times 3} \mathbf{V}_{3 \times n}^T$, etc.
· Affine ambiguity: $\hat{\mathbf{W}} = \hat{\mathbf{M}} \hat{\mathbf{X}} = \hat{\mathbf{M}} \mathbf{A} \mathbf{A}^{-1} \hat{\mathbf{X}} = (\hat{\mathbf{M}} \mathbf{A})(\mathbf{A}^{-1} \hat{\mathbf{X}})$
  · Affinity in $\mathbb{P}^3$: $\begin{bmatrix} \mathbf{A}_{3 \times 3} & \mathbf{t}_{3 \times 1} \\ \mathbf{0}_{1 \times 3} & 1_{1 \times 1} \end{bmatrix}_{4 \times 4}$
  · Camera matrices $\mathbf{M}^i$ & 3D points $\mathbf{X}_j$ obtained from $\hat{\mathbf{M}}$ & $\hat{\mathbf{X}}$ (estimates) are determined up to multiplication by a common matrix $\mathbf{A}$
    · The MLE reconstruction is affine

Part 5 - Appendices

Appendix 4: Matrix Properties and Decompositions

A4.2 Symmetric and Skew-symmetric Matrices

- Cross products
    - $\mathbf{a} = (a_1, a_2, a_3) \to [\mathbf{a}]_\times$

$$[\mathbf{a}]_\times = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix}$$

    - $\mathbf{a} \times \mathbf{b} = [\mathbf{a}]_\times \mathbf{b} = (\mathbf{a}^T [\mathbf{b}]_\times)^T$

Appendix 6: Iterative Estimation Methods

A6.1 Newton Iteration

- Formulation
    - Given a hypothesized function relation $\mathbf{X} = \mathbf{f}(\mathbf{P})$
        - $\mathbf{X}$: a *measurement vector* in Euclidean space $\mathbb{R}^N$
        - $\mathbf{P}$: a *parameter vector* in Euclidean space $\mathbb{R}^M$
    - Measured value $\mathbf{X}$ approximating true value $\bar{\mathbf{X}}$ is provided
    - Find the vector $\hat{\mathbf{P}}$ satisfying $\mathbf{X} = \mathbf{f}(\hat{\mathbf{P}}) - \boldsymbol{\epsilon}$ for which $\|\boldsymbol{\epsilon}\|$ is minimized
        - Linear function $\mathbf{f}(\cdot)$: least-squares problem where $\mathbf{f}(\mathbf{P}) = \mathbf{A}\mathbf{P}$
- Solution
    - Start with an initial estimated value $\mathbf{P}_0$
        - Proceed to refine the estimate under the assumption that *function $\mathbf{f}$ is locally linear*
    - Let $\boldsymbol{\epsilon} = \mathbf{f}(\mathbf{P}_0) - \mathbf{X}$
    - Assume f is approximated at $\mathbf{P}_0$ by $\mathbf{f}(\mathbf{P}_0 + \boldsymbol{\Delta}) = \mathbf{f}(\mathbf{P}_0) + \mathbf{J}\boldsymbol{\Delta}$
        - $\mathbf{J} = \partial \mathbf{f}/\partial \mathbf{P}$ (Jacobian matrix)
    - Seek a point $\mathbf{P}_1 = \mathbf{P}_0 + \boldsymbol{\Delta}$ which minimizes $\mathbf{f}(\mathbf{P}_1) - \mathbf{X} = \mathbf{f}(\mathbf{P}_0) + \mathbf{J}\boldsymbol{\Delta} - \mathbf{X} = \boldsymbol{\epsilon}_0 + \mathbf{J}\boldsymbol{\Delta}$
        - Minimizing $\|\boldsymbol{\epsilon}_0 + \mathbf{J}\boldsymbol{\Delta}\|$ over $\boldsymbol{\Delta} \to$ linear minimization problem
    - $\boldsymbol{\Delta}$ is obtained by
        a. Solving *normal* equations

$$\mathbf{J}^T \mathbf{J} \boldsymbol{\Delta} = -\mathbf{J}^T \boldsymbol{\epsilon}_0 \qquad (A6.1)$$

        b. Using the pseudo-inverse $\boldsymbol{\Delta} = -\mathbf{J}^+ \boldsymbol{\epsilon}_0$
    - Solution vector $\hat{\mathbf{P}}$ is obtained by
        - Starting with an initial estimate $\mathbf{P}_0$ and computing successive approximations by the formula $\mathbf{P}_{i+1} = \mathbf{P}_i + \mathbf{J}\boldsymbol{\Delta}_i$
            - $\boldsymbol{\Delta}_i$: solution to the least-squares problem $\mathbf{J}\boldsymbol{\Delta}_i = -\boldsymbol{\epsilon}_i$
                - $\mathbf{J} = \dfrac{\partial \mathbf{f}}{\partial \mathbf{P}}|_{\mathbf{P}_i}$
                - $\boldsymbol{\epsilon}_i = \mathbf{f}(\mathbf{P}_i) - \mathbf{X}$
    - Disadvantage
        - The iteration procedure may converge to a local minimum or not converge at all
        - The algorithm depends very strongly on the initial estimate $\mathbf{P}_0$

- · Weighted iteration
  - · Measurement $\mathbf{X}$ satisfies a Gaussian distribution with covariance matrix $\mathbf{\Sigma_X}$
    - · $\mathbf{\Sigma_X}$: generally be an arbitrary symmetric & positive definite matrix
  - · Minimize Mahalanobis distance $\|\mathbf{f}(\mathbf{P}) - \mathbf{X}\|_{\mathbf{\Sigma_X}}$ instead of $\|\mathbf{f}(\mathbf{P}) - \mathbf{X}\|$
  - · Normal equation becomes $\mathbf{J}^T\mathbf{\Sigma_X^{-1}}\mathbf{J}\mathbf{\Delta}_i = -\mathbf{J}^T\mathbf{\Sigma_X^{-1}}\boldsymbol{\epsilon}_i$
- · Newton's method & the Hessian
  - · Finding minima of function $g(\mathbf{P})$ of many variables
    - · Expand $g(\mathbf{P})$ about $\mathbf{P}_0$ in a Taylor series
      - · $g(\mathbf{P}_0 + \mathbf{\Delta}) = g + g_\mathbf{P}\mathbf{\Delta} + \mathbf{\Delta}^T g_\mathbf{PP}\mathbf{\Delta}/2 + ...$
    - · Set derivative w.r.t. $\mathbf{\Delta}$ to 0

$$g_\mathbf{PP}\mathbf{\Delta} = -g_\mathbf{P} \tag{A6.2}$$

  - · $g_\mathbf{PP}$: the **Hessian** where the (i, j)-th entry of which is $\partial^2 g/\partial p_i \partial p_j$
  - · **Newton iteration**: starting at an initial value $\mathbf{P}_0$, and iteratively computing parameter increment $\mathbf{\Delta}$ using (A6.2) until convergence occurs
  - · Cost/error function

$$g(\mathbf{P}) = \|\boldsymbol{\epsilon}(\mathbf{P})\|^2/2 = \boldsymbol{\epsilon}(\mathbf{P})^T\boldsymbol{\epsilon}(\mathbf{P})/2$$

  - · Gradient vector: $g_\mathbf{P} = \boldsymbol{\epsilon}_\mathbf{P}^T\boldsymbol{\epsilon}$
    - · $\boldsymbol{\epsilon}_\mathbf{P}$: $\dfrac{\partial\boldsymbol{\epsilon}(\mathbf{P})}{\partial\mathbf{P}}$
    - · $\boldsymbol{\epsilon}$: $\boldsymbol{\epsilon}(\mathbf{P})$
  - · The Hessian

$$g_\mathbf{PP} = \boldsymbol{\epsilon}_\mathbf{P}^T\boldsymbol{\epsilon}_\mathbf{P} + \boldsymbol{\epsilon}_\mathbf{PP}^T\boldsymbol{\epsilon} \tag{A6.3}$$

  - · Assume $\mathbf{f}(\mathbf{P})$ is linear, 2nd term of (A6.3) becomes 0
    - · $g_\mathbf{PP} = \boldsymbol{\epsilon}_\mathbf{P}^T\boldsymbol{\epsilon}_\mathbf{P} = \mathbf{J}^T\mathbf{J}$
    - · (A6.2) becomes: $\mathbf{J}^T\mathbf{J}\mathbf{\Delta} = -\mathbf{J}^T\boldsymbol{\epsilon}$
      - · Same as (A6.1)
  - · **Gaussian-Newton method**: the iterative procedure where $\mathbf{J}^T\mathbf{J}$ is used as an approximation of the Hessian ($g_\mathbf{PP}$)
- · Gradient descent
  - · $-g_\mathbf{P} = -\boldsymbol{\epsilon}_\mathbf{P}^T\boldsymbol{\epsilon}$: negative/down-hill gradient vector defined the direction of most rapid decrease of cost function
  - · Parameter increment $\mathbf{\Delta}$: computed from the equation $\lambda\mathbf{\Delta} = -g_\mathbf{P}$
    - · λ: controls the length of the step
  - · The Hessian is approximated (somewhat arbitrarily) by matrix $\lambda\mathbf{I}$:

$$g_\mathbf{PP}\mathbf{\Delta} = -g_\mathbf{P} \quad \to \quad \lambda\mathbf{I}\mathbf{\Delta} = -g_\mathbf{P} \quad \to \quad \lambda\mathbf{\Delta} = -g_\mathbf{P}$$

  - · Not a good minimization strategy due to slow convergence by zig-zagging
- · Summary on minimization of cost function $g(\mathbf{P}) = \|\boldsymbol{\epsilon}(\mathbf{P})\|^2/2$
  - a. **Newton** $\to$ update equation:

$$g_\mathbf{PP}\mathbf{\Delta} = -g_\mathbf{P}$$

  - · $g_\mathbf{PP} = \boldsymbol{\epsilon}_\mathbf{P}^T\boldsymbol{\epsilon}_\mathbf{P} + \boldsymbol{\epsilon}_\mathbf{PP}^T\boldsymbol{\epsilon}$
  - · $g_\mathbf{P} = \boldsymbol{\epsilon}_\mathbf{P}^T\boldsymbol{\epsilon}$

b. **Gauss-Newton** $\rightarrow$ update equation:

$$\epsilon_\mathbf{P}^T \epsilon_\mathbf{P} \Delta = -\epsilon_\mathbf{P}^T \epsilon$$

c. **Gradient Descent** $\rightarrow$ update equation:

$$\lambda \Delta = -\epsilon_\mathbf{P}^T \epsilon = -g_\mathbf{P}$$

A6.2 Levenberg-Marquardt iteration

· Normal equations $\mathbf{J}^T \mathbf{J} \Delta = -\mathbf{J}^T \epsilon$ are replaced by **augmented normal equations** $(\mathbf{J}^T \mathbf{J} + \lambda \mathbf{I}) \Delta = -\mathbf{J}^T \epsilon$

  · Typical value for λ is 10^{-3}

· Levenberg-Marquardt (LM) iteration

  · Repeatedly solving augmented normal equations for different λ until an acceptable Δ is found

    · If Δ solved leads to reduction in the error

      · Accept the increment

    · If Δ solved leads to increased error

      · Multiply λ by same factor and solve new augmented normal equation

Notes on State Estimation for Robotics Book (Barfoot 2017) (link)

Part I - Estimation Machinery

2. Primer on Probability Theory

2.1 Probability Density Functions

2.1.2 Bayes' Rule and Inference

· Factorization of joint probability

$$p(\mathbf{x}, \mathbf{y}) = p(\mathbf{x}|\mathbf{y})p(\mathbf{y}) = p(\mathbf{y}|\mathbf{x})p(\mathbf{x})$$

   · $\mathbf{x} = (x_1, ..., x_N)$, $\mathbf{y} = (y_1, ..., y_M)$
· Rearranging gives *Bayes' rule*:

$$
\begin{aligned}
p(\mathbf{x}|\mathbf{y}) &= \frac{p(\mathbf{y}|\mathbf{x})p(\mathbf{x})}{p(\mathbf{y})} \\
&= \frac{p(\mathbf{y}|\mathbf{x})p(\mathbf{x})}{\int p(\mathbf{y}|\mathbf{x})p(\mathbf{x})\mathrm{d}\mathbf{x}}
\end{aligned}
$$

   · $p(\mathbf{x}|\mathbf{y})$: (*inferred*) *posterior*/likelihood of the state given the measurements
      · *Posterior* density
   · $p(\mathbf{x})$: *prior* probability density function (PDF) over the state
      · *Prior* density
   · $p(\mathbf{y}|\mathbf{x})$: sensor model

2.1.6 Normalized Product

· Normalized product $p(\mathbf{x})$ of 2 PDFs $p_1(\mathbf{x})$ & $p_2(\mathbf{x})$ for $\mathbf{x}$

$$p(\mathbf{x}) = \eta p_1(\mathbf{x})p_2(\mathbf{x})$$

   · η: normalization constant

$$\eta = (\int p_1(\mathbf{x})p_2(\mathbf{x})\mathrm{d}\mathbf{x})^{-1}$$

Part II - Three Dimensional Machinery

7. Matrix Lie Groups

· Lie group: a group that is also a **differentiable manifold**
  · Property: group operations are smooth

7.1 Geometry

· Two specific Lie groups:
  (1) $SO(3)$: special orthogonal group, which can represent rotations
  (2) $SE(3)$: special Euclidean group, which can represent poses

7.1.1 Special Orthogonal and Special Euclidean Groups

· Special orthogonal group: set of valid rotation matrices

$$SO(3) = \left\{ \mathbf{C} \in \mathbb{R}^{3\times 3} \,\middle|\, \mathbf{C}\mathbf{C}^T = \mathbf{I}, \det\mathbf{C} = 1 \right\}$$

· Special Euclidean group: set of valid transformation matrices

$$SE(3) = \left\{ \mathbf{T} = \begin{bmatrix} \mathbf{C} & \mathbf{r} \\ \mathbf{0}^T & 1 \end{bmatrix} \in \mathbb{R}^{4\times 4} \,\middle|\, \mathbf{C} \in SO(3), \mathbf{r} \in \mathbb{R}^3 \right\}$$

· Lie group properties for $SO(3)$ and $SE(3)$:

property	$SO(3)$	$SE(3)$
closure	$\mathbf{C}_1, \mathbf{C}_2 \in SO(3)$ $\Rightarrow \mathbf{C}_1\mathbf{C}_2 \in SO(3)$	$\mathbf{T}_1, \mathbf{T}_2 \in SE(3)$ $\Rightarrow \mathbf{T}_1\mathbf{T}_2 \in SE(3)$
associativity	$\mathbf{C}_1(\mathbf{C}_2\mathbf{C}_3) = (\mathbf{C}_1\mathbf{C}_2)\mathbf{C}_3$ $= \mathbf{C}_1\mathbf{C}_2\mathbf{C}_3$	$\mathbf{T}_1(\mathbf{T}_2\mathbf{T}_3) = (\mathbf{T}_1\mathbf{T}_2)\mathbf{T}_3$ $= \mathbf{T}_1\mathbf{T}_2\mathbf{T}_3$
identity	$\mathbf{C}, \mathbf{1} \in SO(3)$ $\Rightarrow \mathbf{C}\mathbf{1} = \mathbf{1}\mathbf{C} = \mathbf{C}$	$\mathbf{T}, \mathbf{1} \in SE(3)$ $\Rightarrow \mathbf{T}\mathbf{1} = \mathbf{1}\mathbf{T} = \mathbf{T}$
invertibility	$\mathbf{C} \in SO(3)$ $\Rightarrow \mathbf{C}^{-1} \in SO(3)$	$\mathbf{T} \in SE(3)$ $\Rightarrow \mathbf{T}^{-1} \in SE(3)$

7.1.2 Lie Algebras

· Lie algebra
  · Associated with every matrix Lie group
  · Consists of a vector space $\mathfrak{g}$, over some field $\mathbb{F}$, together with a binary operation $[\cdot\,,\,\cdot]$ called *Lie bracket*
    · Lie bracket: non-associative, alternating bilinear map $\mathfrak{g} \times \mathfrak{g} \to \mathfrak{g}; (x, y) \mapsto [x, y]$
    · Visualization of Lie bracket: link
  · Properties of Lie bracket for all $\mathbf{X}, \mathbf{Y}, \mathbf{Z} \in \mathfrak{g}$ and $a, b \in \mathbb{F}$
    a) Closure: $[\mathbf{X}, \mathbf{Y}] \in \mathfrak{g}$
    b) Bilinearity:
      · $[a\mathbf{X} + b\mathbf{Y}, \mathbf{Z}] = a[\mathbf{X}, \mathbf{Z}] + b[\mathbf{Y}, \mathbf{Z}]$

· $[\mathbf{Z}, a\mathbf{X} + b\mathbf{Y}] = a[\mathbf{Z}, \mathbf{X}] + b[\mathbf{Z}, \mathbf{Y}]$

c) Alternating: $[\mathbf{X}, \mathbf{X}] = \mathbf{0}$

d) Jacobi identity: $[\mathbf{X}, [\mathbf{Y}, \mathbf{Z}]] + [\mathbf{Y}, [\mathbf{Z}, \mathbf{X}]] + [\mathbf{Z}, [\mathbf{X}, \mathbf{Y}]] = \mathbf{0}$

· Lie algebra associated with $SO(3)$ (rotations)

  · Definition:

    · Vector space: $\mathfrak{so}(3) = \{\mathbf{\Phi} = \boldsymbol{\phi}^\wedge \in \mathbb{R}^{3\times 3} \,|\, \boldsymbol{\phi} \in \mathbb{R}^3\}$

    where $\boldsymbol{\phi}^\wedge$ is a skew-symmetric matrix generated from vector $\boldsymbol{\phi}$:

$$\boldsymbol{\phi}^\wedge = \begin{bmatrix} \phi_1 \\ \phi_2 \\ \phi_3 \end{bmatrix}^\wedge = \begin{bmatrix} 0 & -\phi_3 & \phi_2 \\ \phi_3 & 0 & -\phi_1 \\ -\phi_2 & \phi_1 & 0 \end{bmatrix} \in \mathbb{R}^{3\times 3}, \, \boldsymbol{\phi} \in \mathbb{R}^3$$

    · Field: $\mathbb{R}$

    · Lie bracket: $[\mathbf{\Phi}_1, \mathbf{\Phi}_2] = \mathbf{\Phi}_1 \mathbf{\Phi}_2 - \mathbf{\Phi}_2 \mathbf{\Phi}_1$

  · $\mathbf{\Phi} = \boldsymbol{\phi}^\wedge$, $\boldsymbol{\phi} = \mathbf{\Phi}^\vee$

· Lie algebra associated with $SE(3)$ (poses)

  · Definition:

    · Vector space: $\mathfrak{se}(3) = \{\mathbf{\Xi} = \boldsymbol{\xi}^\wedge \in \mathbb{R}^{4\times 4} \,|\, \boldsymbol{\xi} \in \mathbb{R}^6\}$

    where

$$\boldsymbol{\xi}^\wedge = \begin{bmatrix} \boldsymbol{\rho} \\ \boldsymbol{\phi} \end{bmatrix}^\wedge = \begin{bmatrix} \boldsymbol{\phi}^\wedge & \boldsymbol{\rho} \\ \mathbf{0}^T & 0 \end{bmatrix} \in \mathbb{R}^{4\times 4}, \, \boldsymbol{\rho}, \boldsymbol{\phi} \in \mathbb{R}^3$$

      · $\boldsymbol{\rho}$: translation component of pose

      · $\boldsymbol{\phi}$: rotation component of pose

    · Field: $\mathbb{R}$

    · Lie bracket: $[\mathbf{\Xi}_1, \mathbf{\Xi}_2] = \mathbf{\Xi}_1 \mathbf{\Xi}_2 - \mathbf{\Xi}_2 \mathbf{\Xi}_1$

  · $\mathbf{\Xi} = \boldsymbol{\xi}^\wedge$, $\boldsymbol{\xi} = \mathbf{\Xi}^\vee$

· Lie algebra: **tangent space** of the corresponding Lie group (normally, at the *identity* $\mathbf{I}$ of a Lie group)

  · Why *identity*: for Lie group G, the tangent spaces at all points of G are *isomorphic.* So we pick the identity as $\mathbf{I} \in G$ is guaranteed for every Lie group G

  · Intuitive explanation:

    Think of tangent space of a surface at a point as the space of all "directions" one can move from that point (with all velocities) *while staying on that surface.*

    The paths through the point is not on the surface, but a very small portion of the path will be very likely on it.

    Start from the identity (in this example, $\mathbf{I} \in SO(n)$):

$$(\mathbf{I} + \alpha\mathbf{t})^T (\mathbf{I} + \alpha\mathbf{t}) = \mathbf{I} + \alpha(\mathbf{t} + \mathbf{t}^T) + O(\alpha^2)$$

    where $\mathbf{t} \in \mathbb{R}^{3\times 3}$ and $\alpha \to 0$.

    If we ignore the α^2 term, $\mathbf{I} + \alpha\mathbf{t}$ will be in the Lie group if $\mathbf{t} + \mathbf{t}^T = \mathbf{0}$ for small α. That is, the skew-symmetric matrices $\mathbf{t}$ form the Lie algebra $\mathfrak{so}(n)$.

  · More generally, the Lie algebra gives the set of possible ways (directions) in which you can change the group elements while staying in the group

7.1.3 Exponential Map

· Matrix exponential

$$\exp(\mathbf{A}) = \mathbf{I} + \mathbf{A} + \frac{1}{2!}\mathbf{A}^2 + \frac{1}{3!}\mathbf{A}^3 + \cdots = \sum_{n=0}^{\infty} \frac{1}{n!}\mathbf{A}^n$$

· Matrix logarithm

$$\ln(\mathbf{A}) = \sum_{n=1}^{\infty} \frac{(-1)^{n-1}}{n}(\mathbf{A} - \mathbf{I})^n$$

· Exponential map: $\mathfrak{so}(3) \leftrightarrow SO(3)$
  · $\mathfrak{so}(3) \to SO(3)$:

$$\mathbf{C} = \exp(\boldsymbol{\phi}^\wedge) = \sum_{n=0}^{\infty} \frac{1}{n!}(\boldsymbol{\phi}^\wedge)^n$$

  where $\mathbf{C} \in SO(3)$, and $\boldsymbol{\phi} \in \mathbb{R}^3$ ($\boldsymbol{\phi}^\wedge \in \mathfrak{so}(3)$)
    · Let $\boldsymbol{\phi} = \phi\mathbf{a}$ where $\phi = |\boldsymbol{\phi}|$ is the rotation angle, and $\mathbf{a}$ is a unit-vector representing rotation axis (angle-axis representation)

$$\mathbf{C} = \exp(\boldsymbol{\phi}^\wedge) = \exp(\phi\mathbf{a}^\wedge) = \cos\phi\mathbf{I} + \sin\phi\mathbf{a}^\wedge + (1 - \cos\phi)\mathbf{a}\mathbf{a}^T$$

  (*Rodrigues' formula*)
  · $SO(3) \to \mathfrak{so}(3)$ (not uniquely):

$$\boldsymbol{\phi} = \ln(\mathbf{C})^\vee = (\boldsymbol{\phi}^\wedge)^\vee$$

· Exponential map: $\mathfrak{se}(3) \leftrightarrow SE(3)$
  · $\mathfrak{se}(3) \to SE(3)$:

$$\mathbf{T} = \exp(\boldsymbol{\xi}^\wedge)$$
$$= \sum_{n=0}^{\infty} \frac{1}{n!}(\boldsymbol{\xi}^\wedge)^n$$
$$= \sum_{n=0}^{\infty} \frac{1}{n!}\left(\begin{bmatrix} \boldsymbol{\rho} \\ \boldsymbol{\xi} \end{bmatrix}^\wedge\right)^n$$
$$= \sum_{n=0}^{\infty} \frac{1}{n!}\begin{bmatrix} \boldsymbol{\phi}^\wedge & \boldsymbol{\rho} \\ \mathbf{0}^T & 0 \end{bmatrix}^n$$
$$= \begin{bmatrix} \sum_{n=0}^{\infty} \frac{1}{n!}(\boldsymbol{\phi}^\wedge)^n & \left(\sum_{n=0}^{\infty} \frac{1}{(n+1)!}(\boldsymbol{\phi}^\wedge)^n\right)\boldsymbol{\rho} \\ \mathbf{0}^T & 1 \end{bmatrix}$$
$$= \begin{bmatrix} \mathbf{C} & \mathbf{r} \\ \mathbf{0}^T & 1 \end{bmatrix} \in SE(3)$$

  where $\boldsymbol{\xi} \in \mathbb{R}^6$, $\mathbf{r} = \mathbf{J}\boldsymbol{\rho} \in \mathbb{R}^3$, and $\mathbf{J} = \sum_{n=0}^{\infty} \frac{1}{(n+1)!}(\boldsymbol{\phi}^\wedge)^n$ ($\boldsymbol{\xi}^\wedge \in \mathfrak{se}(3)$)

- $SE(3) \to \mathfrak{se}(3)$ (not uniquely):
$$\boldsymbol{\xi} = \ln(\mathbf{T})^\vee = (\boldsymbol{\xi}^\wedge)^\vee$$

  - From $\mathbf{T} \in SE(3)$ to $\boldsymbol{\xi} \in \mathbb{R}^6$:
    a) Compute $\boldsymbol{\phi} \in \mathbb{R}^3$ from $\mathbf{C} \in SO(3)$
    b) Compute $\mathbf{J} = \sum\limits_{n=0}^{\infty} \frac{1}{(n+1)!}(\boldsymbol{\phi}^\wedge)^n$
    c) Compute $\boldsymbol{\rho} = \mathbf{J}^{-1}\mathbf{r}$
    d) Composite $\boldsymbol{\xi}$ using computed $\boldsymbol{\phi}$ and $\boldsymbol{\rho}$
- Jacobian $\mathbf{J}$: $\sum\limits_{n=0}^{\infty} \frac{1}{(n+1)!}(\boldsymbol{\phi}^\wedge)^n$
  - Close-form expressions for $\mathbf{J}$ and $\mathbf{J}^{-1}$:

$$\mathbf{J} = \frac{\sin\phi}{\phi}\mathbf{I} + (1 - \frac{\sin\phi}{\phi})\mathbf{a}\mathbf{a}^T + \frac{1-\cos\phi}{\phi}\mathbf{a}^\wedge$$

$$\mathbf{J}^{-1} = \frac{\phi}{2}\cot\frac{\phi}{2}\mathbf{I} + (1 - \frac{\phi}{2}\cot\frac{\phi}{2})\mathbf{a}\mathbf{a}^T - \frac{\phi}{2}\mathbf{a}^\wedge$$

  where $\phi = |\boldsymbol{\phi}|$ is the rotation angle and $\mathbf{a} = \boldsymbol{\phi}/\phi$ is the unit-length rotation axis
  - $\mathbf{J}\mathbf{J}^T$ and its inverse:

$$\mathbf{J}\mathbf{J}^T = \gamma\mathbf{I} + (1 - \gamma)\mathbf{a}\mathbf{a}^T$$

$$(\mathbf{J}\mathbf{J}^T)^{-1} = \frac{1}{\gamma}\mathbf{I} + (1 - \frac{1}{\gamma})\mathbf{a}\mathbf{a}^T$$

  where $\gamma = 2\dfrac{1-\cos\phi}{\phi^2}$
    - $\mathbf{J}\mathbf{J}^T$ is positive-definite
  - $\mathbf{J}$ represented by rotation matrix $\mathbf{C}$:

$$\mathbf{J} = \int_0^1 \mathbf{C}^\alpha d\alpha$$

  - Rotation matrix $\mathbf{C}$ represented by Jacobian $\mathbf{J}$:

$$\mathbf{C} = \mathbf{I} + \boldsymbol{\phi}^\wedge\mathbf{J}$$

    - Impossible to solve $\mathbf{J}$ because $\boldsymbol{\phi}$ is invertible in the foregoing expression
- Direct series expression for $\mathbf{T} \in SE(3)$

$$\mathbf{T} = \mathbf{I} + \boldsymbol{\xi}^\wedge + (\frac{1-\cos\phi}{\phi^2})(\boldsymbol{\xi}^\wedge)^2 + (\frac{\phi-\sin\phi}{\phi^3})(\boldsymbol{\xi}^\wedge)^3$$

  - Based on the identity: $(\boldsymbol{\xi}^\wedge)^4 + \phi^2(\boldsymbol{\xi}^\wedge)^2 \equiv \mathbf{0}$

7.1.4 Adjoints

- Adjoint of an element of $SE(3)$:

$$\boldsymbol{\mathcal{T}}_{6\times6} = \mathrm{Ad}(\mathbf{T}_{4\times4}) = \mathrm{Ad}\left(\begin{bmatrix} \mathbf{C} & \mathbf{r} \\ \mathbf{0}^T & 1 \end{bmatrix}\right) = \begin{bmatrix} \mathbf{C} & \mathbf{r}^\wedge\mathbf{C} \\ \mathbf{0} & \mathbf{C} \end{bmatrix}$$

where

$$\mathbf{r}^\wedge = \begin{bmatrix} r_1 \\ r_2 \\ r_3 \end{bmatrix}^\wedge = \begin{bmatrix} 0 & -r_3 & r_2 \\ r_3 & 0 & -r_1 \\ -r_2 & r_1 & 0 \end{bmatrix}$$

· Adjoints of all elements of $SE(3)$:

$$\mathrm{Ad}(SE(3)) = \{\boldsymbol{\mathcal{T}} = \mathrm{Ad}(\mathbf{T}) \,|\, \mathbf{T} \in SE(3)\}$$

· $\mathrm{Ad}(SE(3))$ is a Lie group:
  · $\boldsymbol{\mathcal{T}}_1 \boldsymbol{\mathcal{T}}_2 \in \mathrm{Ad}(SE(3))$
  · $\boldsymbol{\mathcal{T}}^{-1} \in \mathrm{Ad}(SE(3))$
· Adjoint of an element $\boldsymbol{\Xi} = \boldsymbol{\xi}^\wedge \in \mathfrak{se}(3)$:

$$\mathrm{ad}(\boldsymbol{\Xi}) = \mathrm{ad}(\boldsymbol{\xi}^\wedge) = \boldsymbol{\xi}^\curlywedge$$

where

$$\boldsymbol{\xi}^\curlywedge = \begin{bmatrix} \boldsymbol{\rho} \\ \boldsymbol{\phi} \end{bmatrix}^\curlywedge = \begin{bmatrix} \boldsymbol{\phi}^\wedge & \boldsymbol{\rho}^\wedge \\ \mathbf{0} & \boldsymbol{\phi}^\wedge \end{bmatrix} \in \mathbb{R}^{6\times 6}, \qquad \boldsymbol{\rho}, \boldsymbol{\phi} \in \mathbb{R}^3$$

· The Lie algebra associated with $\mathrm{Ad}(SE(3))$:
  · Vector space: $\mathrm{ad}(\mathfrak{se}(3)) = \{\boldsymbol{\Psi} = \mathrm{ad}(\boldsymbol{\Xi}) \in \mathbb{R}^{6\times 6} \,|\, \boldsymbol{\Xi} \in \mathfrak{se}(3)\}$
  · Field: $\mathbb{R}$
  · Lie bracket: $[\boldsymbol{\Psi}_1, \boldsymbol{\Psi}_2] = \boldsymbol{\Psi}_1 \boldsymbol{\Psi}_2 - \boldsymbol{\Psi}_2 \boldsymbol{\Psi}_1$
· Exponential map: $\mathrm{ad}(\mathfrak{se}(3)) \leftrightarrow \mathrm{Ad}(SE(3))$
  · $\mathrm{ad}(\mathfrak{se}(3)) \to \mathrm{Ad}(SE(3))$:

$$\boldsymbol{\mathcal{T}} = \exp(\mathrm{ad}(\boldsymbol{\xi}^\wedge)) = \sum_{n=0}^{\infty} \frac{1}{n!}(\boldsymbol{\xi}^\curlywedge)^n$$

  where $\boldsymbol{\mathcal{T}} \in \mathrm{Ad}(SE(3))$ and $\boldsymbol{\xi} \in \mathbb{R}^6$ ($\boldsymbol{\xi}^\curlywedge \in \mathrm{ad}(\mathfrak{se}(3))$)
  · $\mathrm{Ad}(SE(3)) \to \mathrm{ad}(\mathfrak{se}(3))$ (not uniquely):

$$\boldsymbol{\xi} = \ln(\boldsymbol{\mathcal{T}})^\curlyvee = (\boldsymbol{\xi}^\curlywedge)^\curlyvee$$

  · Relationship between various Lie groups and algebras associated with poses

$$
\begin{array}{ccc}
\text{Lie algebra} & & \text{Lie group} \\
4 \times 4 \qquad \boldsymbol{\xi}^\wedge \in \mathfrak{se}(3) & \xrightarrow{\exp} & \mathbf{T} \in SE(3) \\
\big\downarrow \mathrm{ad} & & \big\downarrow \mathrm{Ad} \\
6 \times 6 \qquad \boldsymbol{\xi}^\curlywedge \in \mathrm{ad}(\mathfrak{se}(3)) & \xrightarrow{\exp} & \boldsymbol{\mathcal{T}} \in \mathrm{Ad}(SE(3))
\end{array}
$$

· Direct series expression for $\boldsymbol{\mathcal{T}} \in \mathrm{Ad}(SE(3))$

$$\boldsymbol{\mathcal{T}} = \mathbf{I} + (\frac{3\sin\phi - \phi\cos\phi}{2\phi})\boldsymbol{\xi}^\curlywedge + (\frac{4 - \phi\sin\phi - 4\cos\phi}{2\phi^2})(\boldsymbol{\xi}^\curlywedge)^2$$
$$+ (\frac{\sin\phi - \phi\cos\phi}{2\phi^3})(\boldsymbol{\xi}^\curlywedge)^3 + (\frac{2 - \phi\sin\phi - 2\cos\phi}{2\phi^4})(\boldsymbol{\xi}^\curlywedge)^4$$

  · Based on the identity: $(\boldsymbol{\xi}^\curlywedge)^5 + 2\phi^2(\boldsymbol{\xi}^\curlywedge)^3 + \phi^4\boldsymbol{\xi}^\curlywedge \equiv \mathbf{0}$

7.1.5 Baker-Campbell-Hausdorff

· BCH formula: solution to Z in equation $e^X e^Y = e^Z$ for possibly noncommunicative X and Y in the Lie algebra of a Lie group

· General combinatorial formula by Eugene Dynkin (1947):

$$\ln(\exp(\mathbf{A})\exp(\mathbf{B})) = \sum_{n=1}^{\infty} \frac{(-1)^{n-1}}{n} \sum_{\substack{r_i+s_i>0, \\ 1\leq i \leq n}} \frac{(\sum_{i=1}^{n}(r_i+s_i))^{-1}}{\prod_{i=1}^{n} r_i! s_i!} [\mathbf{A}^{r_1}\mathbf{B}^{s_1}\mathbf{A}^{r_2}\mathbf{B}^{s_2}\cdots\mathbf{A}^{r_n}\mathbf{B}^{s_n}]$$

where

$$[\mathbf{A}^{r_1}\mathbf{B}^{s_1}\mathbf{A}^{r_2}\mathbf{B}^{s_2}\cdots\mathbf{A}^{r_n}\mathbf{B}^{s_n}] = \underbrace{[\mathbf{A},...[\mathbf{A}}_{r_1},\underbrace{[\mathbf{B},...[\mathbf{B}}_{s_1},...\underbrace{[\mathbf{A},...[\mathbf{A}}_{r_n},\underbrace{[\mathbf{B},...[\mathbf{B},\mathbf{B}],...]}_{s_n}]...]...]...]]...]$$

· Lie product formula:

$$\exp(\mathbf{A}+\mathbf{B}) = \lim_{\alpha\to\infty}(\exp(\mathbf{A}/\alpha)\exp(\mathbf{B}/\alpha))^{\alpha}$$

· BCH in $SO(3)$ case
  · General case

$$\ln(\mathbf{C}_1\mathbf{C}_2)^{\vee} = \ln(\exp(\phi_1^{\wedge})\exp(\phi_2^{\wedge}))^{\vee} = \phi_1 + \phi_2 + \frac{1}{2}\phi_1^{\wedge}\phi_2 + \frac{1}{12}\phi_1^{\wedge}\phi_1^{\wedge}\phi_2 + \frac{1}{12}\phi_2^{\wedge}\phi_2^{\wedge}\phi_1 + \cdots$$

  · Cases where either ϕ_1 or ϕ_2 is small

$$\ln(\mathbf{C}_1\mathbf{C}_2)^{\vee} = \ln(\exp(\phi_1^{\wedge})\exp(\phi_2^{\wedge}))^{\vee} \approx \begin{cases} \mathbf{J}_l(\phi_2)^{-1}\phi_1 + \phi_2 & \text{if } \phi_1 \text{ is small} \\ \phi_1 + \mathbf{J}_r(\phi_1)^{-1}\phi_2 & \text{if } \phi_2 \text{ is small} \end{cases}$$

where
$$\mathbf{J}_r(\phi)^{-1} = \sum_{n=0}^{\infty}\frac{B_n}{n!}(-\phi^{\wedge})^n = \frac{\phi}{2}\cot\frac{\phi}{2}\mathbf{I} + (1-\frac{\phi}{2}\cot\frac{\phi}{2})\mathbf{a}\mathbf{a}^T + \frac{\phi}{2}\mathbf{a}^{\wedge}$$

$$\mathbf{J}_l(\phi)^{-1} = \sum_{n=0}^{\infty}\frac{B_n}{n!}(\phi^{\wedge})^n = \frac{\phi}{2}\cot\frac{\phi}{2}\mathbf{I} + (1-\frac{\phi}{2}\cot\frac{\phi}{2})\mathbf{a}\mathbf{a}^T - \frac{\phi}{2}\mathbf{a}^{\wedge}$$

are *right* and *left Jacobians* of $SO(3)$ respectively. The B_n are the *Bernoulli numbers*. $\phi = \phi\mathbf{a}$ where $\phi = |\phi|$ and $\mathbf{a} = \phi/\phi$. $\mathbf{C} = \exp(\phi^{\wedge})$.

· More about left & right Jacobians of $SO(3)$
  · Expressions for the Jacobians

$$\mathbf{J}_r(\phi) = \sum_{n=0}^{\infty}\frac{1}{(n+1)!}(-\phi^{\wedge})^n = \int_0^1 \mathbf{C}^{-\alpha}d\alpha = \frac{\sin\phi}{\phi}\mathbf{I} + (1-\frac{\sin\phi}{\phi})\mathbf{a}\mathbf{a}^T - \frac{1-\cos\phi}{\phi}\mathbf{a}^{\wedge}$$

$$\mathbf{J}_l(\phi) = \sum_{n=0}^{\infty}\frac{1}{(n+1)!}(\phi^{\wedge})^n = \int_0^1 \mathbf{C}^{\alpha}d\alpha = \frac{\sin\phi}{\phi}\mathbf{I} + (1-\frac{\sin\phi}{\phi})\mathbf{a}\mathbf{a}^T + \frac{1-\cos\phi}{\phi}\mathbf{a}^{\wedge}$$

  · Relationship between the 2 Jacobians

$$\mathbf{J}_l(\phi) = \mathbf{C}\mathbf{J}_r(\phi)$$

$$\mathbf{J}_l(-\boldsymbol{\phi}) = \mathbf{J}_r(\boldsymbol{\phi})$$

· BCH in $SE(3)$ and $\mathrm{Ad}(SE(3))$ cases
  · General case

$$\ln(\mathbf{T}_1\mathbf{T}_2)^\vee = \ln(\exp(\boldsymbol{\xi}_1^\wedge)\exp(\boldsymbol{\xi}_2^\wedge))^\vee = \boldsymbol{\xi}_1 + \boldsymbol{\xi}_2 + \frac{1}{2}\boldsymbol{\xi}_1^\wedge\boldsymbol{\xi}_2 + \frac{1}{12}\boldsymbol{\xi}_1^\wedge\boldsymbol{\xi}_1^\wedge\boldsymbol{\xi}_2 + \frac{1}{12}\boldsymbol{\xi}_2^\wedge\boldsymbol{\xi}_2^\wedge\boldsymbol{\xi}_1 + \cdots$$

$$\ln(\boldsymbol{\mathcal{T}}_1\boldsymbol{\mathcal{T}}_2)^\curlyvee = \ln(\exp(\boldsymbol{\xi}_1^\curlywedge)\exp(\boldsymbol{\xi}_2^\curlywedge))^\curlyvee = \boldsymbol{\xi}_1 + \boldsymbol{\xi}_2 + \frac{1}{2}\boldsymbol{\xi}_1^\curlywedge\boldsymbol{\xi}_2 + \frac{1}{12}\boldsymbol{\xi}_1^\curlywedge\boldsymbol{\xi}_1^\curlywedge\boldsymbol{\xi}_2 + \frac{1}{12}\boldsymbol{\xi}_2^\curlywedge\boldsymbol{\xi}_2^\curlywedge\boldsymbol{\xi}_1 + \cdots$$

· Cases where either $\boldsymbol{\xi}_1$ or $\boldsymbol{\xi}_2$ is small

$$\ln(\mathbf{T}_1\mathbf{T}_2)^\vee = \ln(\exp(\boldsymbol{\xi}_1^\wedge)\exp(\boldsymbol{\xi}_2^\wedge))^\vee \approx \begin{cases} \boldsymbol{\mathcal{J}}_l(\boldsymbol{\xi}_2)^{-1}\boldsymbol{\xi}_1 + \boldsymbol{\xi}_2 & \text{if } \boldsymbol{\xi}_1 \text{ is small} \\ \boldsymbol{\xi}_1 + \boldsymbol{\mathcal{J}}_r(\boldsymbol{\xi}_1)^{-1}\boldsymbol{\xi}_2 & \text{if } \boldsymbol{\xi}_2 \text{ is small} \end{cases}$$

$$\ln(\boldsymbol{\mathcal{T}}_1\boldsymbol{\mathcal{T}}_2)^\curlyvee = \ln(\exp(\boldsymbol{\xi}_1^\curlywedge)\exp(\boldsymbol{\xi}_2^\curlywedge))^\curlyvee \approx \begin{cases} \boldsymbol{\mathcal{J}}_l(\boldsymbol{\xi}_2)^{-1}\boldsymbol{\xi}_1 + \boldsymbol{\xi}_2 & \text{if } \boldsymbol{\xi}_1 \text{ is small} \\ \boldsymbol{\xi}_1 + \boldsymbol{\mathcal{J}}_r(\boldsymbol{\xi}_1)^{-1}\boldsymbol{\xi}_2 & \text{if } \boldsymbol{\xi}_2 \text{ is small} \end{cases}$$

where

$$\boldsymbol{\mathcal{J}}_r(\boldsymbol{\phi})^{-1} = \sum_{n=0}^{\infty} \frac{B_n}{n!}(-\boldsymbol{\xi}^\curlywedge)^n$$

$$\boldsymbol{\mathcal{J}}_l(\boldsymbol{\phi})^{-1} = \sum_{n=0}^{\infty} \frac{B_n}{n!}(\boldsymbol{\xi}^\curlywedge)^n$$

are *right* and *left Jacobians* of $SE(3)$ respectively.

· More about left & right Jacobians of $SE(3)$
  · Expressions for the Jacobians

$$\boldsymbol{\mathcal{J}}_r(\boldsymbol{\xi}) = \sum_{n=0}^{\infty} \frac{1}{(n+1)!}(-\boldsymbol{\xi}^\curlywedge)^n = \int_0^1 \boldsymbol{\mathcal{T}}^{-\alpha}\mathrm{d}\alpha$$

$$\boldsymbol{\mathcal{J}}_l(\boldsymbol{\xi}) = \sum_{n=0}^{\infty} \frac{1}{(n+1)!}(\boldsymbol{\xi}^\curlywedge)^n = \int_0^1 \boldsymbol{\mathcal{T}}^{\alpha}\mathrm{d}\alpha$$

· Relationship between the 2 Jacobians

$$\boldsymbol{\mathcal{J}}_l(\boldsymbol{\xi}) = \boldsymbol{\mathcal{T}}\boldsymbol{\mathcal{J}}_r(\boldsymbol{\xi})$$

$$\boldsymbol{\mathcal{J}}_l(-\boldsymbol{\xi}) = \boldsymbol{\mathcal{J}}_r(\boldsymbol{\xi})$$

· Rewrite BCH approximations using *left Jacobian* ($\mathbf{J} = \mathbf{J}_l$ and $\boldsymbol{\mathcal{J}} = \boldsymbol{\mathcal{J}}_l$)
  · $SO(3)$ case:

$$\ln(\mathbf{C}_1\mathbf{C}_2)^\vee = \ln(\exp(\boldsymbol{\phi}_1^\wedge)\exp(\boldsymbol{\phi}_2^\wedge))^\vee \approx \begin{cases} \mathbf{J}(\boldsymbol{\phi}_2)^{-1}\boldsymbol{\phi}_1 + \boldsymbol{\phi}_2 & \text{if } \boldsymbol{\phi}_1 \text{ is small} \\ \boldsymbol{\phi}_1 + \mathbf{J}(-\boldsymbol{\phi}_1)^{-1}\boldsymbol{\phi}_2 & \text{if } \boldsymbol{\phi}_2 \text{ is small} \end{cases}$$

· *SE*(3) case:

$$\ln(\mathbf{T}_1\mathbf{T}_2)^\vee = \ln(\exp(\boldsymbol{\xi}_1^\wedge)\exp(\boldsymbol{\xi}_2^\wedge))^\vee \approx \begin{cases} \mathcal{J}(\boldsymbol{\xi}_2)^{-1}\boldsymbol{\xi}_1 + \boldsymbol{\xi}_2 & \text{if } \boldsymbol{\xi}_1 \text{ is small} \\ \boldsymbol{\xi}_1 + \mathcal{J}(-\boldsymbol{\xi}_1)^{-1}\boldsymbol{\xi}_2 & \text{if } \boldsymbol{\xi}_2 \text{ is small} \end{cases}$$

$$\ln(\boldsymbol{\mathcal{T}}_1\boldsymbol{\mathcal{T}}_2)^\curlyvee = \ln(\exp(\boldsymbol{\xi}_1^\curlywedge)\exp(\boldsymbol{\xi}_2^\curlywedge))^\curlyvee \approx \begin{cases} \mathcal{J}(\boldsymbol{\xi}_2)^{-1}\boldsymbol{\xi}_1 + \boldsymbol{\xi}_2 & \text{if } \boldsymbol{\xi}_1 \text{ is small} \\ \boldsymbol{\xi}_1 + \mathcal{J}(-\boldsymbol{\xi}_1)^{-1}\boldsymbol{\xi}_2 & \text{if } \boldsymbol{\xi}_2 \text{ is small} \end{cases}$$

7.1.6 Distance, Volume, Integration

· Difference of 2 rotations: right difference $\boldsymbol{\phi}_{12}$ and left difference $\boldsymbol{\phi}_{21}$

$$\boldsymbol{\phi}_{12} = \ln(\mathbf{C}_1^T\mathbf{C}_2)^\vee$$

$$\boldsymbol{\phi}_{21} = \ln(\mathbf{C}_2\mathbf{C}_1^T)^\vee$$

where $\mathbf{C}_1, \mathbf{C}_2 \in SO(3)$
· Inner product for $\mathfrak{so}(3)$:
$$\langle \boldsymbol{\phi}_1^\wedge, \boldsymbol{\phi}_2^\wedge \rangle = \frac{1}{2}\text{tr}\left(\boldsymbol{\phi}_1^\wedge\boldsymbol{\phi}_2^{\wedge^T}\right) = \boldsymbol{\phi}_1^T\boldsymbol{\phi}_2$$

· The metric *distance* between 2 rotations: (i) the square root of the inner product of difference with itself, or (ii) the Euclidean norm of the difference, or (iii) the magnitude of the angle of the rotation difference

$$\phi_{12} = \sqrt{\langle \boldsymbol{\phi}_{12}, \boldsymbol{\phi}_{12} \rangle} = \sqrt{\boldsymbol{\phi}_{12}^T\boldsymbol{\phi}_{12}} = |\boldsymbol{\phi}_{12}|$$

$$\phi_{21} = \sqrt{\langle \boldsymbol{\phi}_{21}, \boldsymbol{\phi}_{21} \rangle} = \sqrt{\boldsymbol{\phi}_{21}^T\boldsymbol{\phi}_{21}} = |\boldsymbol{\phi}_{21}|$$

· Integration on rotations
  · Let $\mathbf{C} = \exp(\boldsymbol{\phi}^\wedge) \in SO(3)$
  · Let $\mathbf{C}' = \exp((\boldsymbol{\phi} + \delta\boldsymbol{\phi})^\wedge)$ by purturbing $\mathbf{C}$ a bit
  · Right & left differences (relative to $\mathbf{C}$):

$$\ln(\delta\mathbf{C}_r)^\vee = \ln(\mathbf{C}^T\mathbf{C}')^\vee \approx \ln(\mathbf{C}^T\mathbf{C}\exp((\mathbf{J}_r\delta\boldsymbol{\phi})^\wedge))^\vee = \mathbf{J}_r\delta\boldsymbol{\phi}$$

$$\ln(\delta\mathbf{C}_l)^\vee = \ln(\mathbf{C}'\mathbf{C}^T)^\vee \approx \ln(\exp((\mathbf{J}_l\delta\boldsymbol{\phi})^\wedge)\mathbf{C}\mathbf{C}^T)^\vee = \mathbf{J}_l\delta\boldsymbol{\phi}$$

where $\mathbf{J}_r$ and $\mathbf{J}_l$ are evaluated at $\boldsymbol{\phi}$
  · BCH approximation (assume $\mathbf{J}_r(\boldsymbol{\phi})\delta\boldsymbol{\phi}$ is small enough):

$$\begin{aligned} \mathbf{C}' &= \exp((\boldsymbol{\phi} + \delta\boldsymbol{\phi})^\wedge) \\ &= \exp((\boldsymbol{\phi} + \mathbf{J}_r(\boldsymbol{\phi})^{-1}(\mathbf{J}_r(\boldsymbol{\phi})\delta\boldsymbol{\phi}))^\wedge) \\ &\approx \exp(\boldsymbol{\phi}^\wedge)\exp((\mathbf{J}_r(\boldsymbol{\phi})\delta\boldsymbol{\phi})^\wedge) \\ &= \mathbf{C}\exp((\mathbf{J}_r(\boldsymbol{\phi})\delta\boldsymbol{\phi})^\wedge) \end{aligned}$$

· Infinitestimal volume element

$$\mathrm{d}\mathbf{C} = |\det \mathbf{J}|\mathrm{d}\boldsymbol{\phi}$$
$$= \mathrm{d}\mathbf{C}_r = |\det \mathbf{J}_r|\mathrm{d}\boldsymbol{\phi}$$
$$= \mathrm{d}\mathbf{C}_l = |\det \mathbf{J}_l|\mathrm{d}\boldsymbol{\phi}$$

· $|\det(\mathbf{J})| = 2\dfrac{1 - \cos\phi}{\phi^2}$

· Integrating functions of rotations

$$\int_{SO(3)} f(\mathbf{C})\mathrm{d}\mathbf{C} \to \int_{|\phi|<\pi} f(\boldsymbol{\phi})|\det\mathbf{J}|\mathrm{d}\boldsymbol{\phi}$$

· Similar for poses ($SE(3)$ and $\mathrm{Ad}(SE(3))$)

   · Inner product defined for $\mathfrak{se}(3)$ (4×4) and $\mathrm{ad}(\mathfrak{se}(3))$ (6×6)

$$\langle \boldsymbol{\xi}_1^\wedge, \boldsymbol{\xi}_2^\wedge \rangle = -\mathrm{tr}\left(\boldsymbol{\xi}_1^\wedge \begin{bmatrix} \mathbf{I}/2 & \mathbf{0} \\ \mathbf{0}^T & \mathbf{I} \end{bmatrix} {\boldsymbol{\xi}_2^\wedge}^T \right) = \boldsymbol{\xi}_1^T \boldsymbol{\xi}_2$$

$$\langle \boldsymbol{\xi}_1^\curlywedge, \boldsymbol{\xi}_2^\curlywedge \rangle = -\mathrm{tr}\left(\boldsymbol{\xi}_1^\curlywedge \begin{bmatrix} \mathbf{I}/4 & \mathbf{0} \\ \mathbf{0}^T & \mathbf{I}/2 \end{bmatrix} {\boldsymbol{\xi}_2^\curlywedge}^T \right) = \boldsymbol{\xi}_1^T \boldsymbol{\xi}_2$$

7.1.7 Interpolation

· From typical linear interpolation scheme to interpolation of 2 rotations

$$x = (1 - \alpha)x_1 + \alpha x_2 = \alpha(x_2 - x_1) + x_1$$

$$\Downarrow$$

$$\mathbf{C} = (\mathbf{C}_2\mathbf{C}_1^T)^\alpha \mathbf{C}_1$$

where $\alpha \in [0,1]$, $\mathbf{C}, \mathbf{C}_1, \mathbf{C}_2 \in SO(3)$

   · Let $\mathbf{C} = \exp(\boldsymbol{\phi}^\wedge)$, $\mathbf{C}_1 = \exp(\boldsymbol{\phi_1}^\wedge)$, $\mathbf{C}_2 = \exp(\boldsymbol{\phi_2}^\wedge) \in SO(3)$, with $\boldsymbol{\phi}, \boldsymbol{\phi}_1, \boldsymbol{\phi}_2 \in \mathbb{R}^3$

$$\boldsymbol{\phi} = \ln(\mathbf{C})^\vee = \ln\left((\mathbf{C}_2\mathbf{C}_1^T)^\alpha\mathbf{C}_1\right)^\vee$$
$$= \ln(\exp(\alpha\boldsymbol{\phi}_{21}^\wedge)\exp(\boldsymbol{\phi}_1^\wedge))^\vee \approx \alpha\mathbf{J}(\boldsymbol{\phi}_1)^{-1}\boldsymbol{\phi}_{21} + \boldsymbol{\phi}_1$$

· Purturbed rotations ($SO(3)$)

   · Let $\mathbf{C}'$, $\mathbf{C}_1'$, $\mathbf{C}_2' \in SO(3)$ be the purturbed rotation matrices with left differences given as

$$\delta\boldsymbol{\phi} = \ln(\mathbf{C}'\mathbf{C}^T)^\vee, \quad \delta\boldsymbol{\phi}_1 = \ln(\mathbf{C}_1'\mathbf{C}_1^T)^\vee, \quad \delta\boldsymbol{\phi}_2 = \ln(\mathbf{C}_2'\mathbf{C}_2^T)^\vee$$

   · Interpolation scheme must hold for the purturbed rotation matrices

$$\mathbf{C}' = (\mathbf{C}_2'\mathbf{C}_1'^T)^\alpha\mathbf{C}_1', \quad \alpha \in [0,1]$$

   · Special case: $\mathbf{C} = \mathbf{C}_2^\alpha$, $\boldsymbol{\phi} = \alpha\boldsymbol{\phi}_2$ when $\mathbf{C}_1 = \mathbf{I}$

66

· Perturbation quantity:

$$\delta\phi = (\mathbf{I} - \mathbf{A}(\alpha, \phi_{21}))\delta\phi_1 + \mathbf{A}(\alpha, \phi_{21})\delta\phi_2$$

where

$$\mathbf{A}(\alpha, \phi_{21}) = \alpha\mathbf{J}(\alpha\phi)\mathbf{J}(\phi)^{-1}$$

$$\phi_{21} = \ln(\mathbf{C}_2\mathbf{C}_1^T)^\vee$$

- · $\mathbf{A}(\alpha, \phi) \approx \alpha\mathbf{I}$ when ϕ is small
- · $\mathbf{A}(\alpha, \phi)$ has a series expression which can be used to compute it (only use the first several terms if ϕ is small

· Similar for purturbed poses ($SE(3)$)

7.1.8 Homogeneous Points

· Represent points in $\mathbb{R}^3$ via 4×1 *homogeneous coordinates*:

$$\mathbf{p} = \begin{bmatrix} sx \\ sy \\ sz \\ s \end{bmatrix} = \begin{bmatrix} \boldsymbol{\varepsilon} \\ \eta \end{bmatrix}$$

where $\boldsymbol{\varepsilon} \in \mathbb{R}^3$, η is a scalar
· Define 2 operators for manipulating homogeneous points:

$$\begin{bmatrix} \boldsymbol{\varepsilon} \\ \eta \end{bmatrix}^\odot = \begin{bmatrix} \eta\mathbf{I}_{3\times3} & -\boldsymbol{\varepsilon}_{3\times3}^\wedge \\ \mathbf{0}_{1\times3}^T & \mathbf{0}_{1\times3}^T \end{bmatrix}_{4\times6}, \quad \begin{bmatrix} \boldsymbol{\varepsilon} \\ \eta \end{bmatrix}^\circledcirc = \begin{bmatrix} \mathbf{0}_{3\times1} & \boldsymbol{\varepsilon}_{3\times1} \\ -\boldsymbol{\varepsilon}_{3\times3}^\wedge & \mathbf{0}_{3\times1} \end{bmatrix}_{6\times4}$$

· Useful identities:
- · $\boldsymbol{\xi}^\wedge\mathbf{p} \equiv \mathbf{p}^\odot\boldsymbol{\xi}$, $\mathbf{p}^T\boldsymbol{\xi}^\wedge \equiv \boldsymbol{\xi}^T\mathbf{p}^\circledcirc$ where $\boldsymbol{\xi} \in \mathbb{R}^6$ and $\mathbf{p} \in \mathbb{R}^4$
- · $(\mathbf{Tp})^\odot \equiv \mathbf{Tp}^\odot\boldsymbol{\mathcal{T}}^{-1}$

7.1.9 Calculus and Optimization

· Rotations
- · Directly taking Jacobian of a rotated point $\mathbf{Cv}$ w.r.t. the Lie algebra vector ϕ representing the rotation $\mathbf{C}$:
$$\frac{\partial(\mathbf{Cv})}{\partial\phi} = -(\mathbf{Cv})^\wedge\mathbf{J}$$

where $\mathbf{C} = \exp(\phi^\wedge) \in SO(3)$, $\mathbf{v} \in \mathbb{R}^3$ is some arbitrary three-dimensional point, and $\mathbf{J} = \mathbf{J}_l(\phi)$
- · For scalar function $u(\mathbf{x})$ with $\mathbf{x} = \mathbf{Cv}$, applying chain rule of differentiation

$$\frac{\partial u}{\partial\phi} = \frac{\partial u}{\partial\mathbf{x}}\frac{\partial\mathbf{x}}{\partial\phi} = -\frac{\partial u}{\partial\mathbf{x}}(\mathbf{Cv})^\wedge\mathbf{J}$$

The result is the transpose of the gradient u w.r.t. ϕ.

- $\dim(\partial u_{\mathbf{1}\times 1}/\partial \boldsymbol{\phi}_{\mathbf{3}\times 1}) = 1 \times 3$

· Gradient descent on function $u(\cdot)$: update in the direction of negative gradient, evaluated at linearization point, $\mathbf{C}_{op} = \exp(\boldsymbol{\phi}_{op}^{\wedge})$

$$\boldsymbol{\phi} = \boldsymbol{\phi}_{op} - \alpha \left(\frac{\partial u}{\partial \boldsymbol{\phi}} \right)^T$$

$$= \boldsymbol{\phi}_{op} - \alpha \left(- \frac{\partial u}{\partial \mathbf{x}}\bigg|_{\mathbf{x} = \mathbf{C}_{op}\mathbf{v}} (\mathbf{C}_{op}\mathbf{v})^{\wedge} \mathbf{J} \right)^T$$

$$= \boldsymbol{\phi}_{op} + \alpha \mathbf{J}^T (-(\mathbf{C}_{op}\mathbf{v})^{\wedge}) \frac{\partial u}{\partial \mathbf{x}}\bigg|^T_{\mathbf{x} = \mathbf{C}_{op}\mathbf{v}}$$

$$= \boldsymbol{\phi}_{op} - \alpha \mathbf{J}^T \underbrace{(\mathbf{C}_{op}\mathbf{v})^{\wedge} \frac{\partial u}{\partial \mathbf{x}}\bigg|^T_{\mathbf{x} = \mathbf{C}_{op}\mathbf{v}}}_{\boldsymbol{\delta}}$$

where $\alpha > 0$ is the step size

· Cleaner way to carry out optimization: find an update step for $\mathbf{C}$ in the form of small rotation on the left rather than directly on the Lie algebra rotation vector $\boldsymbol{\phi}$

$$\mathbf{C} = \exp(\boldsymbol{\phi}^{\wedge}) = \exp \left((\boldsymbol{\phi}_{op} - \alpha \mathbf{J}^T \boldsymbol{\delta})^{\wedge} \right)$$

$$\approx \exp \left(-\alpha (\mathbf{J}\mathbf{J}^T \boldsymbol{\delta})^{\wedge} \right) \exp(\mathbf{C}_{op})$$

$$= \exp(\boldsymbol{\psi}^{\wedge}) \exp(\mathbf{C}_{op})$$

- Let $\boldsymbol{\psi} = -\alpha \mathbf{J}\mathbf{J}^T \boldsymbol{\delta}$
- To avoid computing Jacobian, drop $\mathbf{J}\mathbf{J}^T > 0$ and use $\boldsymbol{\psi} = -\alpha \boldsymbol{\delta}$ instead (slightly different direction for gradient descent)

· Compute Jacobian w.r.t $\boldsymbol{\psi}$, where the purturbation is applied on the left

$$\frac{\partial(\mathbf{C}\mathbf{v})}{\partial \boldsymbol{\phi}} \approx \frac{\partial(\exp(\boldsymbol{\psi}^{\wedge})\mathbf{C}\mathbf{v})}{\partial \boldsymbol{\psi}} = -(\mathbf{C}\mathbf{v})^{\wedge}$$

· Think optimization in terms of purturbation:
  · Purturbation scheme

$$\mathbf{C}\mathbf{v} = \exp(\boldsymbol{\psi}^{\wedge})\exp(\boldsymbol{\phi}_{op}^{\wedge})\mathbf{v}$$

$$= f(\boldsymbol{\phi}_{op} \oplus \boldsymbol{\psi})$$

$$\approx f(\boldsymbol{\phi}_{op}) + \frac{\partial f(\boldsymbol{\phi}_{op})}{\partial \boldsymbol{\psi}} \boldsymbol{\psi}$$

$$= \exp(\boldsymbol{\phi}_{op}^{\wedge})\mathbf{v} - (\exp(\boldsymbol{\phi}_{op}^{\wedge})\mathbf{v})^{\wedge} \boldsymbol{\psi}$$

$$= \mathbf{C}_{op}\mathbf{v} - (\mathbf{C}_{op}\mathbf{v})^{\wedge} \boldsymbol{\psi}$$

where $\oplus$ denotes the purturbation operation, and 1st-order Taylor series is used for the approximation

- Insert the scheme into the function to be optimized

$$
\begin{aligned}
u(\mathbf{C}\mathbf{v}) &= u(\exp(\boldsymbol{\psi}^\wedge)\mathbf{C}_{op}\mathbf{v}) \\
&\approx u(\mathbf{C}_{op}\mathbf{v} - (\mathbf{C}_{op}\mathbf{v})^\wedge\boldsymbol{\psi}) \\
&\approx u(\mathbf{C}_{op}\mathbf{v}) - \underbrace{\left.\frac{\partial u}{\partial \mathbf{x}}\right|_{\mathbf{x}=\mathbf{C}_{op}\mathbf{v}}(\mathbf{C}_{op}\mathbf{v})^\wedge}_{\boldsymbol{\delta}^T}\boldsymbol{\psi} \\
&= u(\mathbf{C}_{op}\mathbf{v}) + \boldsymbol{\delta}^T\boldsymbol{\psi}
\end{aligned}
$$

- An example perturbation vector for gradient descent:

$$
\boldsymbol{\psi} = -\alpha\mathbf{D}\boldsymbol{\delta}
$$

where $\alpha > 0$ a small step size and $\mathbf{D}$ is any positive-definite matrix
- Update step:

$$
\mathbf{C}_{op} \leftarrow \exp(-\alpha\mathbf{D}\boldsymbol{\delta}^\wedge)\mathbf{C}_{op}
$$

where $\mathbf{C}_{op} \in SO(3)$ is guaranteed at each iteration
- Purturbation applied to Gauss-Newton optimization method
  - An example general nonlinear, quadratic cost function of rotation

$$
J(\mathbf{C}) = \frac{1}{2}\sum_m (u_m(\mathbf{C}\mathbf{v}_m))^2
$$

where $u_m(\cdot)$ are scalar nonlinear functions and $\mathbf{v}_m \in \mathbb{R}^3$ are three-dimensional points
  - Purturb on an initial guess for the optimal rotation $\mathbf{C}_{op} \in SO(3)$:

$$
\mathbf{C} = \exp(\boldsymbol{\psi}^\wedge)\mathbf{C}_{op}
$$

where $\boldsymbol{\psi}$ is the purturbation
  - Apply purturbation scheme inside each $u_m(\cdot)$ to get linearized version of $u_m(\cdot)$ in terms of purturbation $\boldsymbol{\psi}$

$$
u_m(\mathbf{C}\mathbf{v}_m) \approx \underbrace{u_m(\mathbf{C}_{op}\mathbf{v}_m)}_{\beta_m} \underbrace{\left.\frac{\partial u_m}{\partial \mathbf{x}}\right|_{\mathbf{x}=\mathbf{C}_{op}\mathbf{v}_m}(\mathbf{C}_{op}\mathbf{v}_m)^\wedge}_{\boldsymbol{\delta}_m^T}\boldsymbol{\psi}
$$

and then

$$
J(\mathbf{C}) \approx \frac{1}{2}\sum_m (\boldsymbol{\delta}_m^T\boldsymbol{\psi} + \beta_m)^2
$$

- Take the derivative of J w.r.t $\boldsymbol{\psi}$

$$
\frac{\partial J}{\partial \boldsymbol{\psi}^T} = \sum_m \boldsymbol{\delta}_m(\boldsymbol{\delta}_m^T\boldsymbol{\psi} + \beta_m)
$$

· Set derivative to 0 to find the optimal purturbation $\boldsymbol{\psi}^*$ that minimizes J

$$\left(\sum_m \boldsymbol{\delta}_m \boldsymbol{\delta}_m^T\right) \boldsymbol{\psi}^* = -\sum_m \beta_m \boldsymbol{\delta}_m$$

· Apply the optimal purturbation for each iteration

$$\mathbf{C}_{op} \leftarrow \exp\left(\boldsymbol{\psi}^{*\wedge}\right) \mathbf{C}_{op}$$

where $\mathbf{C}_{op} \in SO(3)$ is guaranteed for each iteration

· Poses
  · The Jacobian of a transformed point w.r.t the Lie algebra vector $\boldsymbol{\xi}$ representing the transformation:
  $$\frac{\partial(\mathbf{Tp})}{\partial \boldsymbol{\xi}} = (\mathbf{Tp})^{\odot} \boldsymbol{\mathcal{J}}$$

  · The Jacobian w.r.t. the purturbation $\boldsymbol{\epsilon}$ where it is applied on the left of the transformation (i.e. the (left) Lie derivative)

  $$\frac{\partial(\mathbf{Tp})}{\partial \boldsymbol{\epsilon}} = (\mathbf{Tp})^{\odot} \quad \text{if} \quad \mathbf{T} \leftarrow \exp(\boldsymbol{\epsilon}^{\wedge})\mathbf{T}$$

  where the need to compute $\boldsymbol{\mathcal{J}}$ matrix is removed
  · Gauss-Newton optimization is similar to rotations by using the above purturbation scheme

70

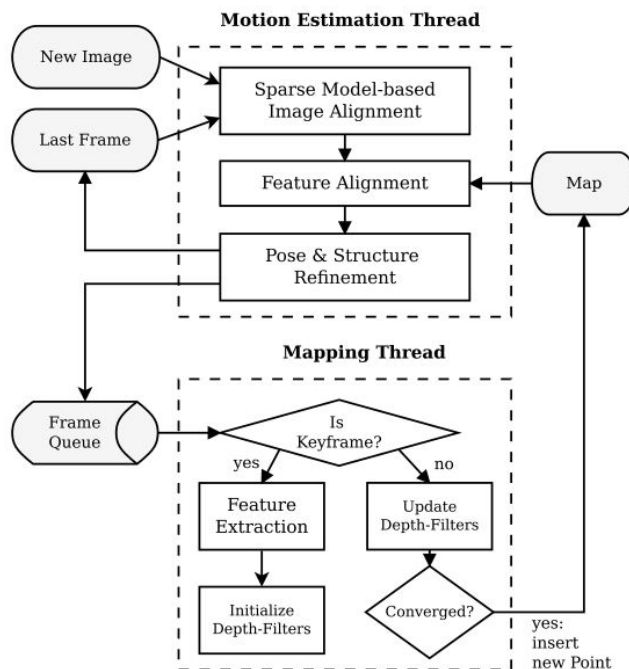# Overview of Various SLAM-related Papers

1. Classic Papers

1.1 Direct-Based Systems

1.1.1 SVO - Fast Semi-Direct Monocular Visual Odometry (Forster, Pizzoli, and Scaramuzza 2014)

· Paper

1.1.1.1 Overview

· System diagram:

· System overview:

  · Motion estimation thread
    · Pose initialization: through sparse model-based image alignment
      · Find relative pose $\mathbf{T}_{k,k-1}$
    · Pose & structure refinement: BA procedure

- · Mapping thread
  - · Depth-filter
    - · Initialized with a large uncertainty in depth
    - · Updated in a Bayesian fashion
    - · Depth is computed from the point in non-reference frame with the highest correlation (the point lying on the epipolar line) with the reference patch
  - · Add new 3D point only if the depth-filter is converged (i.e., depth uncertainty is small enough)
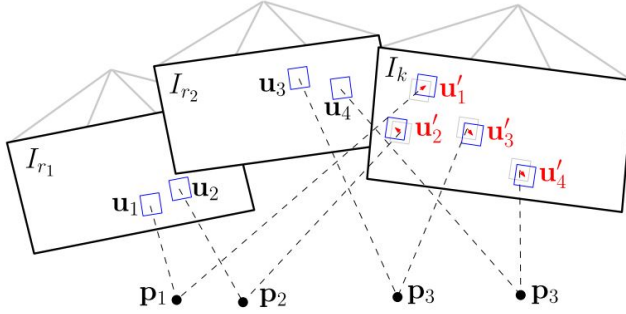
1.1.1.2 Motion Estimation

- · Procedures:

  1) Sparse Model-based Image Alignment
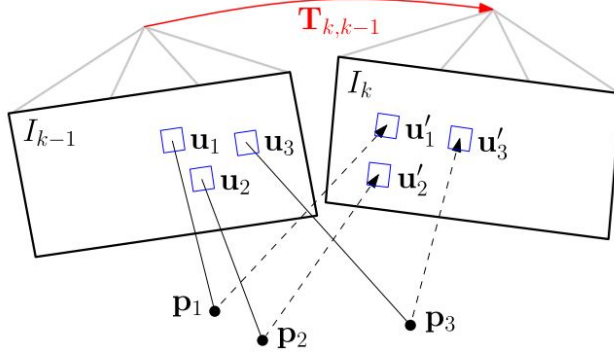
  - · Goal: compute relative pose $\mathbf{T}_{k,k-1}$ from frame $k-1$ to frame k through minimization of photometric error of image patches
  - · Minimization of reprojection error through Gauss-Newton procedure
  - · Optimize both camera pose and image patch positions in the current frame

  2) Feature Alignment

  - · Goal: further minimize photometric error of feature patches by estimating new feature positions in current frame w.r.t. reference keyframe
  - · Solved using inverse compositional Lucas-Kanade algorithm (Baker and Matthews 2004)
    - · *Inverse compositional*: apply the delta parameters on the template/reference image during an iteration of optimization, instead of the current image, and then update the parameters with the inverse of the delta parameters
  - · Camera pose is not involved in the minimization scheme
  - · Only the feature patch positions are optimized
  - · A relaxation scheme: because of the violation of epipolar constraint for higher correla-

tion between the feature patches

3) Pose & Structure Refinement
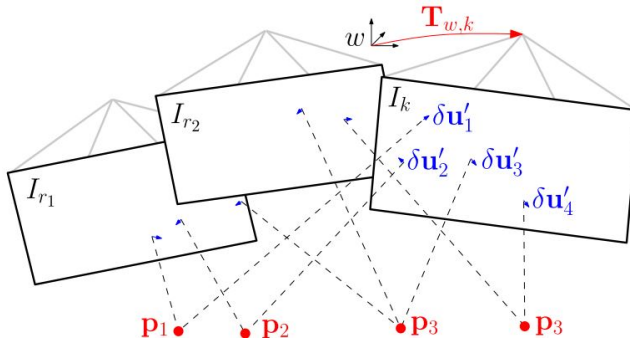
- · Step 1: motion-only BA: only optimize the camera pose
- · Step 2: structure-only BA: only optimize the 3D points
- · Step 3: (optional) local BA: optimize both local poses and 3D points
- · Minimization of reprojection error through Gauss-Newton procedure

· Advantages of the above procedures combined:

- · Advantage of the 1st procedure
  - · The procedure implicitly satisfies epipolar constraint and ensures there's no outliers
  - · Without this procedure, the tracking will be slower because of the tracking of all features over large distances, whereas this procedure only optimizes 6 parameters
- · Advantage of the combination of the 3 procedures:
  - · 2nd and 3rd procedures are necessary to reduce drift
    - · 3rd procedure: image alignment w.r.t. the map and keyframes
    - · 1st procedure only aligns image w.r.t. the previous frame

1.1.1.3 Mapping

· Depth-filter in SVO:
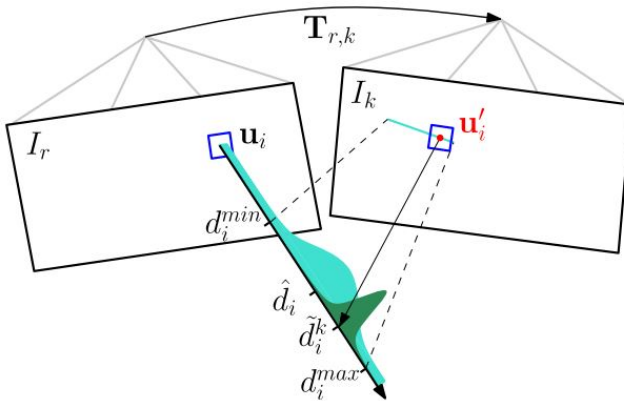
- · Modification to the original depth-filter in (Vogiatzis and Hernández 2011)
· Find depth of a image point via depth-filter associated with a keyframe
· Search for a patch on the epipolar line in the new image that has the highest correlation (most probable depth) with the reference patch
· Depth measurement is modeled with a Gaussian + uniform mixture model
- · Gaussian distribution for inlier
- · Uniform distribution for outlier

- · Weighted average of the 2 distributions
    - · Weight: inlier ratio π
- · Depth-filter update:
    - · Depth posterior: product of a **Gaussian** for the *depth Z* with a **Beta** distribution for the *inlier ratio* π as an *approximation*
        - · $Z \sim \mathcal{N}(\mu, \sigma^2)$, $\pi \sim Beta(a, b)$
    - · Bayesian update: $p(Z, \pi | x_n) = \dfrac{p(x_n | Z, \pi) p(Z, \pi)}{p(x_n)}$
        - · Prior: $p(Z, \pi) = p(Z) p(\pi)$
        - · Likelihood: $p(x_n | Z, \pi)$
        - · Posterior $p(Z, \pi | x_n)$ approximated by a $Gaussian \times Beta$ distribution $q(Z, \pi | a, b, \mu, \sigma^2)$
        - · Update step:
            - · Match the 1st and 2nd moment of $p(x | Z, \pi) q(Z, \pi | a, b, \mu, \sigma^2)$ and $q(Z, \pi | a', b', \mu', \sigma'^2)$
            - · Solve 4 equations to get a', b', μ', and σ'^2, and substitute the 4 old parameters, and also get the new measurement x

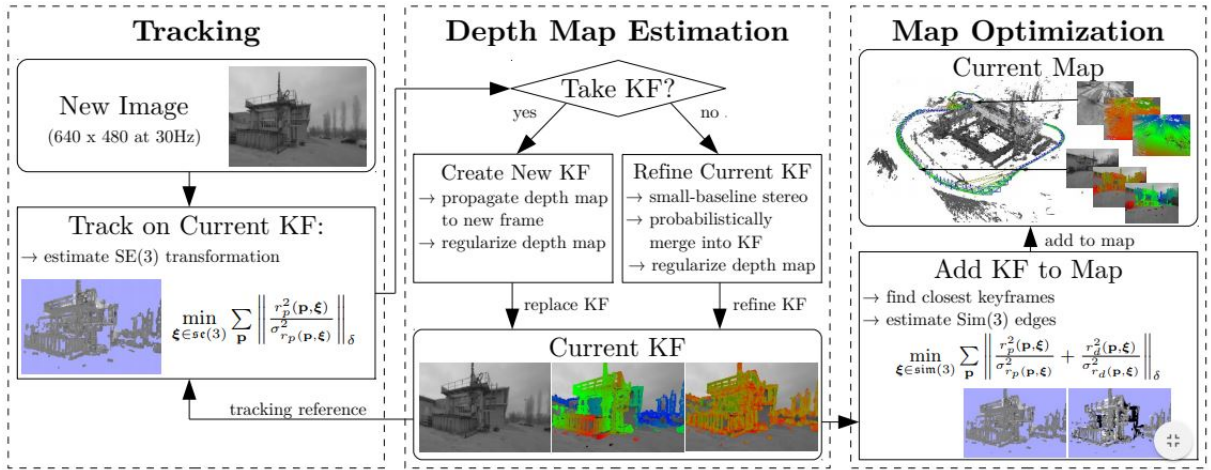### 1.1.1.4 Implementation Details

- · System initialization:
    - · Assume a locally planar scene and estimate a homography: compute 1st pose via decomposition of **H**
- · Sparse image alignment
    - · Coarse-to-fine scheme: utilize a image pyramid of 5 levels with scale factor of 2
    - · Optimization on the coarsest level for an initial solution
    - · Continue optimization on finer level and stop after convergence on the 3rd level
- · Mapping
    - · Fixed number of keyframes
        - · Queue: new keyframe added will cause the farthest keyframe to be removed
    - · Keyframe selection scheme: Euclidean distance of the new frame relative to all keyframes exceeds 12% of the average scene depth
    - · Feature extraction
        - · Divide the image in cells of fixed size (30×30 pixels)
        - · Extract FAST corners with the highest Shi-Tomasi score in each image cell at *all levels* of the pyramid
            - · Skip this step if the cell already has a 2D-to-3D correspondence

1.1.2 LSD-SLAM: Large-scale Direct Monocular SLAM (Engel, Schöps, and Cremers 2014)

- · Paper

1.1.2.1 Overview

- · System diagram:

Tracking	Depth Map Estimation	Map Optimization

Tracking

New Image
(640 x 480 at 30Hz)

Track on Current KF:
→ estimate SE(3) transformation

$$\min_{\boldsymbol{\xi} \in \mathfrak{se}(3)} \sum_{\mathbf{p}} \left\| \frac{r_p^2(\mathbf{p}, \boldsymbol{\xi})}{\sigma_{r_p(\mathbf{p}, \boldsymbol{\xi})}^2} \right\|_\delta$$

tracking reference

Depth Map Estimation

Take KF?   yes   no

Create New KF
→ propagate depth map to new frame
→ regularize depth map

Refine Current KF
→ small-baseline stereo
→ probabilistically merge into KF
→ regularize depth map

replace KF   refine KF

Current KF

Map Optimization

Current Map

add to map

Add KF to Map
→ find closest keyframes
→ estimate Sim(3) edges

$$\min_{\boldsymbol{\xi} \in \mathfrak{sim}(3)} \sum_{\mathbf{p}} \left\| \frac{r_p^2(\mathbf{p}, \boldsymbol{\xi})}{\sigma_{r_p(\mathbf{p}, \boldsymbol{\xi})}^2} + \frac{r_d^2(\mathbf{p}, \boldsymbol{\xi})}{\sigma_{r_d(\mathbf{p}, \boldsymbol{\xi})}^2} \right\|_\delta$$

· Features

  · Direct method to align 2 keyframes on $\mathfrak{sim}(3)$
    · Incorporate and detect scale-drift in a novel scale-aware image alignment algorithm
    · Detect loop closures
  · Probabilistic approach to incorporate noise on the estimated depth maps into tracking
  · A semi-dense map represented as point clouds
  · Tracking in environment with large variations in scale and large rotation

1.1.2.2 Complete Method

· 3 major components
  · **Tracking**
    · Estimates pose $\boldsymbol{\xi} \in \mathfrak{se}(3)$ w.r.t. the current keyframe, using the previous frame as initialization
  · **Depth Map Estimation**
    · Uses tracked frames to either *refine* or *replace* the current keyframe
  · **Map Optimization**
    · Detect loop closures and scale-drift by estimating a similarity transform $\boldsymbol{\xi} \in \mathfrak{sim}(3)$ to close-by existing keyframes in a scale-aware & direct $\mathfrak{sim}(3)$-image alignment scheme
· Initialization scheme
  · Initialize a first keyframe with a *random* depth map and large variance

1.1.2.3 Map Representation

· Global map as pose graph
· Vertices: keyframes $\mathcal{K}_i$
  · Camera image $I_i : \Omega \to \mathbb{R}$
  · Inverse depth map $D_i : \Omega_{D_i} \to \mathbb{R}^+$ where $\Omega_{D_i} \subset \Omega_i$
  · Variance of inverse depth $V_i : \Omega_{D_i} \to \mathbb{R}^+$
· Edges $\mathcal{E}_{ji}$
  · Relative alignment $\boldsymbol{\xi}_{ji} \in \mathfrak{sim}(3)$
  · Corresponding covariance matrix $\boldsymbol{\Sigma}_{ji}$

1.1.2.4 Tracking: Direct $\mathfrak{se}(3)$ Image Alignment

· Minimize the variance-normalized photometric error

$$E_p(\boldsymbol{\xi}_{ji}) = \sum_{p \in \Omega_{D_i}} \|\frac{r_p^2(\mathbf{p}, \boldsymbol{\xi}_{ji})}{\sigma_{r_p(\mathbf{p}, \boldsymbol{\xi}_{ji})}^2}\|_\delta$$

where

$$r_p(\mathbf{p}, \boldsymbol{\xi}_{ji}) = I_i(\mathbf{p}) - I_j(\omega(\mathbf{p}, D_i(\mathbf{p}), \boldsymbol{\xi}_{ji}))$$

being the photometric error/residual with $\omega(\cdot)$ as the mapping of pixel $\mathbf{p}$ in reference image I_i to the current image I_j, given the relative pose $\boldsymbol{\xi}_{ji} \in \mathfrak{se}(3)$, and

$$\omega_{r_p(\mathbf{p}, \boldsymbol{\xi}_{ji})}^2 = 2\omega_I^2 + (\frac{\partial r_p(\mathbf{p}, \boldsymbol{\xi}_{ji})}{\partial D_i(\mathbf{p})})^2 V_i(\mathbf{p})$$

being the variance of the residual term taking into account image noise ω_I^2 (assumed to be w Gaussian) and varying noise on depth estimates, and $\|\cdot\|_\delta$ being the Huber norm

1.1.2.5 Depth Map Estimation

· Depth map creation for new keyframe
    · Project points from the previous keyframe to the new one
    · Perform spatial regularization and outlier removal as described in (Engel, Sturm, and Cremers 2013)
    · Scale the depth map to have a mean inverse depth of one
        · The scaling factor is incorporated into the $\mathfrak{sim}(3)$ pose
    · Replace the previous keyframe and use it for tracking subsequent new frames
· Depth map refinement (Engel, Sturm, and Cremers 2013)
    · Filter over many per-pixel, small baseline stereo comparisons coupled with interleaved spatial regularization
        · For each pixel, find a previous reference frame as *old* as possible
        · Given known relative pose and the pixel, compute the epipolar line on the reference frame
        · Stereo matching: perform an exhaustive search alone the epipolar line using *SSD* error (over 5 equidistant points on the line)
        · Identify 3 major factors which determine the accuracy of such a stereo observation, and determine which pixel is worth updating the depth estimation

1.1.2.6 Constraint Acquisition: Direct $\mathfrak{sim}(3)$ Image Alignment

· Perform *direct, scale-drift aware image alignment* on $\mathfrak{sim}(3)$ to align 2 differently scaled keyframes
    · Add a depth redisual r_d to the error function defined for tracking

$$E(\boldsymbol{\xi}_{ji}) = \sum_{p \in \Omega_{D_i}} \|\frac{r_p^2(\mathbf{p}, \boldsymbol{\xi}_{ji})}{\sigma_{r_p(\mathbf{p}, \boldsymbol{\xi}_{ji})}^2} + \frac{r_d^2(\mathbf{p}, \boldsymbol{\xi}_{ji})}{\sigma_{r_d(\mathbf{p}, \boldsymbol{\xi}_{ji})}^2}\|_\delta$$

where

$$r_d(\mathbf{p}, \boldsymbol{\xi}_{ji}) = [\mathbf{p}']_3 - D_j([\mathbf{p}']_{1,2})$$

with $[\cdot]_k$ meaning the kth row of a matrix

- Constraint search (loop closure detection)
    - Use the closest 10 keyframes and a suitable candidate keyframe proposed by an appearance-based mapping algorithm (Glover et al. 2011) to detect loop closures
    - Perform a **reciprocal tracking check** for each candidate keyframe $\mathcal{K}_{j_k}$:
        - Independently track $\boldsymbol{\xi}_{j_k i}$ and $\boldsymbol{\xi}_{i j_k}$
        - Only if the 2 estimates are statistically similar are they added to the global map
- Improve convergence radius (ability to track large motion) for $\mathfrak{sim}(3)$ tracking (direct $\mathfrak{sim}(3)$ image alignment)
    - Use a small number of keypoints to compute a better initialization
    - Use **Efficient Second Order Minimization (ESM)** (Benhimane and Malis 2004)
    - Use coarse-to-fine search (image pyramids) starting at a very low resolution (such as 20×15)

1.1.2.7 Map Optimization

- Map: keyframes + tracked $\mathfrak{sim}(3)$ constraints
- Continuously optimize the map in the background using pose graph optimization (implemented using *g2o*)
    - Vertices: pose $\boldsymbol{\xi}_{Wi} \in \mathfrak{sim}(3)$ in the world coordinate frame for keyframe $\mathcal{K}_i$
        - W: world coordinate frame
    - Edges $\mathcal{E}$: relative pose $\boldsymbol{\xi}_{ji}$ from keyframe $\mathcal{K}_i$ to $\mathcal{K}_j$
    - Error function:

$$E(\boldsymbol{\xi}_{W1}, ..., \boldsymbol{\xi}_{Wn}) = \sum_{(\boldsymbol{\xi}_{ji}, \boldsymbol{\Sigma}_{ji}) \in \mathcal{E}} (\boldsymbol{\xi}_{ji} \circ \boldsymbol{\xi}_{Wi}^{-1} \circ \boldsymbol{\xi}_{Wj})^T \boldsymbol{\Sigma}_{ji}^{-1} (\boldsymbol{\xi}_{ji} \circ \boldsymbol{\xi}_{Wi}^{-1} \circ \boldsymbol{\xi}_{Wj})$$

1.2 Feature-based Systems

1.3 Systems in Kalman Filter Framework

1.3.1 A Multi-State Constraint Kalman Filter for Vision-aided Inertial Navigation (Mourikis and Roumeliotis 2007)

- Paper
- Related materials:
    - Quaternion kinematics for the error-state Kalman filter (Sola 2017): include related concepts/formulations on quaternions/rotational matrices, and their applications in Kalman filter framework
    - Why and How to Avoid the Flipped Quaternion Multiplication: details on different conventions for quaternions

1.3.1.1 Overview

- A visual inertial navigation system (VINS), **MSCKF** in short

- · Camera + inertial measurement unit (IMU)

- · Features

  - · An EKF-based algorithm in real time for VIN
  - · Main contribution:
    - · Proposed a **measurement model** for the expression of geometric constraints when observing a landmark from multiple camera poses
      - · The geometric constraints involve all related camera poses that observe the landmark
    - · The measurement model **does not require including 3D feature position** in the state vector of EKF, resulting in computational complexity only **linear** in the number of features
    - · The measurement model is **optimal** and **up to linearization errors**

- · Terms

  - · $\{I\}$: IMU-affixed frame
  - · $\{G\}$: global frame
    - · Chosen as *Earth-Centered, Earth-Fixed* (ECEF) frame:
      - · (X, Y, Z) coordinates
      - · $(0, 0, 0)$ is defined as the center mass of Earth
      - · Z-axis: extends through true north
      - · X-axis: intersects the sphere of the earth at 0° latitude (the equator) and 0° longitude (prime meridian in Greenwich)
      - · Right-handed coordinate system

- · Proposed algorithm

  Algorithm 1 Multi-State Constraint Filter

  Propagation: For each IMU measurement received, propagate the filter state and covariance (cf. Section III-B).

  Image registration: Every time a new image is recorded,
  - augment the state and covariance matrix with a copy of the current camera pose estimate (cf. Section III-C).
  - image processing module begins operation.

  Update: When the feature measurements of a given image become available, perform an EKF update (cf. Sections III-D and III-E).

- · EKF state vector at any time instant

  - · Evolving IMU state $\mathbf{X}_{IMU}$
  - · History of up to N_{max} past camera poses

1.3.1.2 Structure of EKF State Vector

- · Some references:
  - · Error-state Kalman Filter (ESKF): chapter 5 in (Sola 2017)

· Evolving IMU state

$$\mathbf{X}_{IMU} = \begin{bmatrix} {}^{I}_{G}\mathbf{q}^{T} & \mathbf{b}_{g}^{T} & {}^{G}\mathbf{v}_{I}^{T} & \mathbf{b}_{a}^{T} & {}^{G}\mathbf{p}_{I}^{T} \end{bmatrix}^{T}$$

- · ${}^{I}_{G}\mathbf{q}^{T}$: unit **q**uaternion of rotation from global frame $\{G\}$ to IMU frame $\{I\}$
  - · **WARNING**: this paper uses the **JPL convention** $(ij = -ji = -k)$ for quaternions instead of the **Hamilton convention** $(ij = -ji = k)$
- · ${}^{G}\mathbf{v}_{I}^{T}$ & ${}^{G}\mathbf{p}_{I}^{T}$: **v**elocity and **p**osition of IMU w.r.t. $\{G\}$
- · $\mathbf{b}_{g}^{T}$ & $\mathbf{b}_{a}^{T}$: **b**iases which affect the **g**yroscope and **a**ccelerometer measurements
  - · They are modeled as random walk processes, driven by white Gaussian noise vectors $\mathbf{n}_{wg}$ and $\mathbf{n}_{wa}$

· IMU error-state

$$\tilde{\mathbf{X}}_{IMU} = \begin{bmatrix} \delta\boldsymbol{\theta}_{I}^{T} & \tilde{\mathbf{b}}_{g}^{T} & {}^{G}\tilde{\mathbf{v}}_{I}^{T} & \tilde{\mathbf{b}}_{g}^{T} & {}^{G}\tilde{\mathbf{p}}_{I}^{T} \end{bmatrix}^{T}$$

- · Error definitions:
  - · Standard additive errors for *position*, *velocity*, and *biases*
    - · $\tilde{x} = x - \hat{x}$ (error = real - estimate)
  - · Error for quaternion $\mathbf{q}$: *error quaternion* $\delta\mathbf{q}$
    - · $\mathbf{q} = \delta\mathbf{q} \otimes \hat{\mathbf{q}}$ where $\otimes$ denotes quaternion multiplication

· EKF state vector at time-step k with N camera poses included:

$$\hat{\mathbf{X}}_{k} = \begin{bmatrix} \hat{\mathbf{X}}_{IMU_{k}}^{T} & {}^{C_{1}}_{G}\hat{\mathbf{q}}^{T} & {}^{G}\hat{\mathbf{p}}_{C_{1}}^{T} & ... & {}^{C_{N}}_{G}\hat{\mathbf{q}}^{T} & {}^{G}\hat{\mathbf{p}}_{C_{N}}^{T} \end{bmatrix}^{T}$$

- · ${}^{C_{i}}_{G}\hat{\mathbf{q}}^{T}$ & ${}^{G}\hat{\mathbf{p}}_{C_{i}}^{T}$: ith estimate of the camera attitude/pose and position

· EKF error-state vector at time-step k with N camera poses included:

$$\tilde{\mathbf{X}}_{k} = \begin{bmatrix} \tilde{\mathbf{X}}_{IMU_{k}}^{T} & \delta\boldsymbol{\theta}_{C_{1}}^{T} & {}^{G}\tilde{\mathbf{p}}_{C_{1}}^{T} & ... & \delta\boldsymbol{\theta}_{C_{N}}^{T} & {}^{G}\tilde{\mathbf{p}}_{C_{N}}^{T} \end{bmatrix}^{T}$$

1.3.1.3 Propagation

1.3.1.3.1 Continuous-time System Modeling

· Time evolution of IMU state (derivative w.r.t. time / increment over period dt):

$$\begin{bmatrix} {}^{I}_{G}\dot{\mathbf{q}}(t) \\ \dot{\mathbf{b}}_{g}(t) \\ {}^{G}\dot{\mathbf{v}}_{I}(t) \\ \dot{\mathbf{b}}_{a}(t) \\ {}^{G}\dot{\mathbf{p}}_{I}(t) \end{bmatrix} = \begin{bmatrix} \frac{1}{2}\boldsymbol{\Omega}(\boldsymbol{\omega}(t)){}^{I}_{G}\mathbf{q}(t) \\ \mathbf{n}_{wg}(t) \\ {}^{G}\mathbf{a}(t) \\ \mathbf{n}_{wa}(t) \\ {}^{G}\mathbf{v}_{I}(t) \end{bmatrix} \qquad \text{(MSCKF-6)}$$

where

$$\boldsymbol{\omega} = \begin{bmatrix} \omega_{x} & \omega_{y} & \omega_{z} \end{bmatrix}^{T}, \quad \boldsymbol{\Omega}(\boldsymbol{\omega}) = \begin{bmatrix} -[\boldsymbol{\omega}]_{\times} & \boldsymbol{\omega} \\ -\boldsymbol{\omega}^{T} & 0 \end{bmatrix}$$

· Ordinary differential equation (ODE) for ${}^I_G\dot{\mathbf{q}}(t)$:

$$
\begin{aligned}
{}^I_G\dot{\mathbf{q}}(t) &= \frac{1}{2}{}^I_G\mathbf{q}(t) \otimes \begin{bmatrix} \boldsymbol{\omega}^T & 0 \end{bmatrix}^T \\
&= \frac{1}{2}(0 \cdot \mathbf{I}_{4\times4} + \boldsymbol{\Omega}(\boldsymbol{\omega})){}^I_G\mathbf{q}(t) \\
&= \frac{1}{2}\boldsymbol{\Omega}(\boldsymbol{\omega}){}^I_G\mathbf{q}(t)
\end{aligned}
$$

· Gyroscope and accelerometer measurement $\boldsymbol{\omega}_m$ and $\mathbf{a}_m$ incorporating the effects of the planet's rotation $\boldsymbol{\omega}_G$:

$$
\boldsymbol{\omega}_m = \boldsymbol{\omega} + \mathbf{C}({}^I_G\mathbf{q})\boldsymbol{\omega}_G + \mathbf{b}_g + \mathbf{n}_g
$$

$$
\mathbf{a}_m = \mathbf{C}({}^I_G\mathbf{q})({}^G\mathbf{a} - {}^G\mathbf{g} + 2[\boldsymbol{\omega}_G]_\times{}^G\mathbf{v}_I + [\boldsymbol{\omega}_G]^2_\times{}^G\mathbf{p}_I) + \mathbf{b}_a + \mathbf{n}_a
$$

where $\mathbf{C}(\cdot)$ converts an input into a rotational matrix

· *Estimates* of the evolving IMU state by applying expectation operator to the equation of time evolution of IMU state in (MSCKF-6)

$$
\begin{bmatrix}
{}^I_G\dot{\hat{\mathbf{q}}} \\
\dot{\hat{\mathbf{b}}}_g \\
{}^G\dot{\hat{\mathbf{v}}}_I \\
\dot{\hat{\mathbf{b}}}_a \\
{}^G\dot{\hat{\mathbf{p}}}_I
\end{bmatrix}
=
\begin{bmatrix}
\frac{1}{2}\boldsymbol{\Omega}(\hat{\boldsymbol{\omega}}){}^I_G\hat{\mathbf{q}} \\
\mathbf{0}_{3\times1} \\
\mathbf{C}^T_{\hat{\mathbf{q}}}\hat{\mathbf{a}} - 2[\boldsymbol{\omega}]_\times{}^G\hat{\mathbf{v}}_I - [\boldsymbol{\omega}]^2_\times{}^G\hat{\mathbf{p}}_I + {}^G\mathbf{g} \\
\mathbf{0}_{3\times1} \\
{}^G\hat{\mathbf{v}}_I
\end{bmatrix}
\tag{MSCKF-9}
$$

where

$$
\mathbf{C}_{\hat{\mathbf{q}}} = \mathbf{C}({}^I_G\hat{\mathbf{q}}), \quad \hat{\boldsymbol{\omega}} = \boldsymbol{\omega}_m - \hat{\mathbf{b}}_g - \mathbf{C}_{\hat{\mathbf{q}}}\boldsymbol{\omega}_G, \quad \hat{\mathbf{a}} = \mathbf{a}_m - \hat{\mathbf{b}}_a
$$

· *Linearized* continuous-time model for IMU error-state:

$$
\dot{\tilde{\mathbf{X}}}_{IMU} = \mathbf{F}\tilde{\mathbf{X}}_{IMU} + \mathbf{G}\mathbf{n}_{IMU}
$$

where $\mathbf{n}_{IMU} = \begin{bmatrix} \mathbf{n}_g^T & \mathbf{n}_{wg}^T & \mathbf{n}_a^T & \mathbf{n}_{wa}^T \end{bmatrix}^T_{12\times1}$ is the system noise, and

$$
\mathbf{F} =
\begin{bmatrix}
-[\hat{\boldsymbol{\omega}}]_\times & -\mathbf{I}_{3\times3} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
-\mathbf{C}^T_{\hat{\mathbf{q}}}[\hat{\mathbf{a}}]_\times & \mathbf{0} & -2[\boldsymbol{\omega}_G]_\times & -\mathbf{C}^T_{\hat{\mathbf{q}}} & -[\boldsymbol{\omega}_G]^2_\times \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
\mathbf{0} & \mathbf{0} & \mathbf{I}_{3\times3} & \mathbf{0} & \mathbf{0}
\end{bmatrix}_{15\times15}
$$

and

$$
\mathbf{G} =
\begin{bmatrix}
-\mathbf{I}_{3\times3} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
\mathbf{0} & \mathbf{I}_{3\times3} & \mathbf{0} & \mathbf{0} \\
\mathbf{0} & \mathbf{0} & -\mathbf{C}^T_{\hat{q}} & \mathbf{0} \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I}_{3\times3} \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0}
\end{bmatrix}_{15\times12}
$$

1.3.1.3.2 Discrete-time Implementation

- Time-step: T
- Method for the propagation of the IMU state / EKF covariance matrix (i.e., to solve the ODE using numerical integration): 5th order Runge-Kutta method
    - Propagation: can be seen as the value of a function $f(t)$ at time $t_{base} + \delta t$ based on its value at time t_{base} after one/multiple time-step(s) δt elapsed
        - In this paper, propagation is to compute $f(t_k + T)$ given $f(t_k)$ and time-step T
- Propagation of IMU state (**not** error-state): numerical integration on the derivatives in (MSCKF-9)
- Propagation of EKF covariance matrix
    - EKF covariance matrix $\mathbf{P}_{k|k}$

$$\mathbf{P}_{k|k} = \begin{bmatrix} \mathbf{P}_{II_{k|k}} & \mathbf{P}_{IC_{k|k}} \\ \mathbf{P}_{IC_{k|k}}^T & \mathbf{P}_{CC_{k|k}} \end{bmatrix}$$

- where
    - $\mathbf{P}_{II_{k|k}}$ is the 15×15 covariance matrix of the evolving IMU error-state
    - $\mathbf{P}_{CC_{k|k}}$ is the $6N \times 6N$ covariance matrix of the camera pose estimates
    - $\mathbf{P}_{IC_{k|k}}$ is the correlation between IMU error-state and camera pose estimates

2. New Papers (2019)

2.1 IEEE International Conference on Robotics and Automation 2019 (ICRA2019)

2.1.1 Loosely-Coupled Semi-Direct Monocular SLAM (Lee and Civera 2019)

- Paper

System diagram:
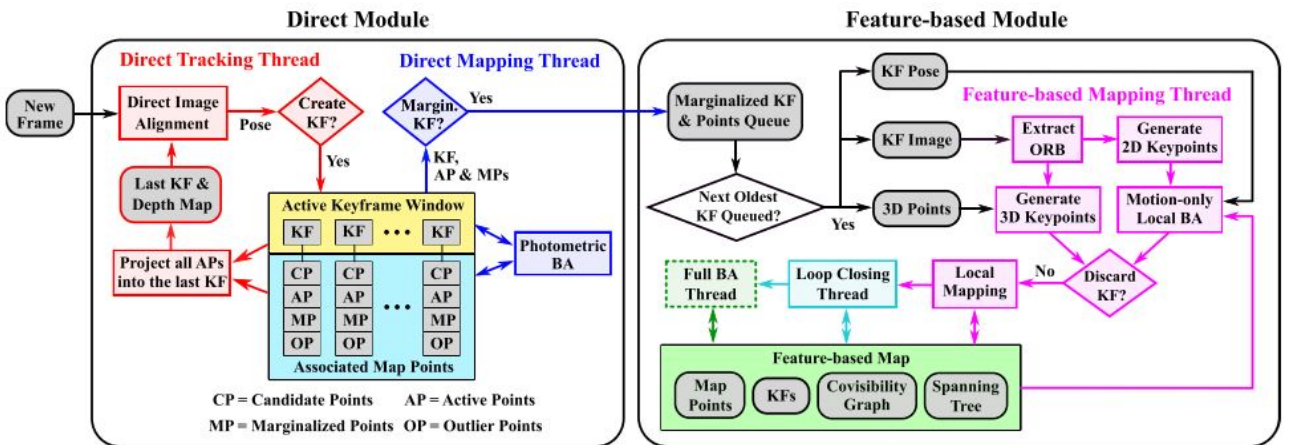
- Built upon direct sparse odometry (DSO) & ORB-SLAM

Features:

- Combine the complementary strength of direct and feature-based methods
- 2 modules running in parallel:

    1) Direct odometry module: uses direct method to track new frames based on a local semi-dense map

    2) Feature-based module: builds a sparse feature-based map based on tracked map points and poses

· 3 levels of parallel optimizations

    1) Photometric bundle adjustment (BA): jointly optimizes local structure and motion

    2) Geometric BA: refines keyframe poses and associated feature map points

    3) Pose graph optimization: achieve global map consistency in case of loop closures

· Achieve real-time by limiting feature-based operations to marginalized keyframes from the direct odometry module

2.1.2 Sparse2Dense: From Direct Sparse Odometry to Dense 3-D Reconstruction (J. Tang, Folkesson, and Jensfelt 2019)

· Paper

System diagram:
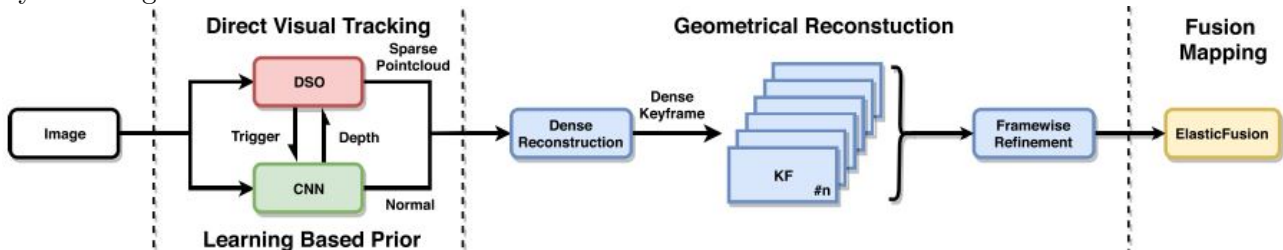
· 4 major stages:

    1) Depth/normal generation using CNN

    2) Visual tracking using direct alignment (direct sparse odometry, DSO)

    3) Geometrical sparse to dense reconstruction

    4) Fusion-based mapping

· Contribution on stage 1 & 3

Features:

· A *deep* dense monocular SLAM system

· Construct a dense 3D model via a sparse to dense mapping using learned surface normals
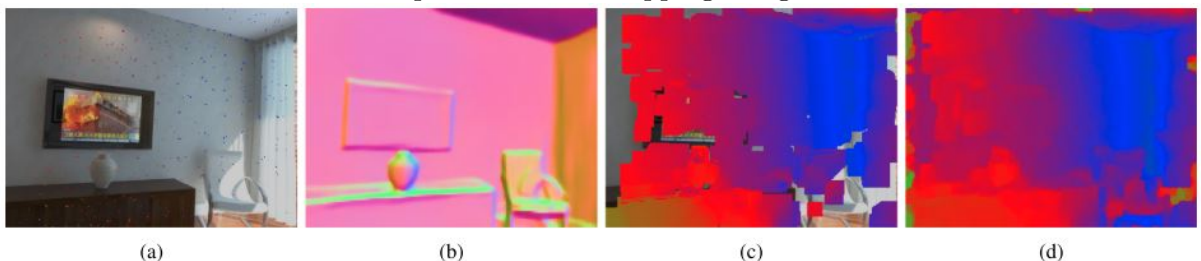
Fig. 3. The sparse to dense procedure, figures above progressively shows the intermediate outputs: (a) optimized sparse depth image using CNN depth as prior; (b) CNN normal; (c) dense reconstruction using (a) and (b); (d) after (c) has been refined with adjacent keyframes.

· Depth estimations are learned in single view as prior for monocular visual odometry (VO)

· VO with the prior above can obtain both accurate positioning and high-quality depth reconstruction

· Depth and normals are predicted by a single convoluted neural network (CNN) trained in a

tightly coupled manner

  · Depth: used in the projective geometric optimization for accurate pose estimation
  · Normal: used for a dense geometrical reconstruction with intermediate sparse point clouds
  · CNN structure
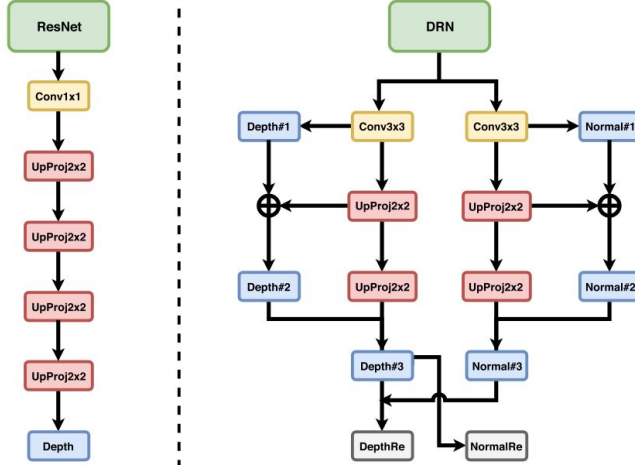
  · Left: original *fully convolutional residual network* (FCRN) with an encoder-decoder
    structure based on *ResNet-50*
  · right: FCDRN with *Resnet-50* replaced by the *Dilated Residual Network* (DRN)

2.1.3 Tightly-Coupled Visual-Inertial Localization and 3-D Rigid-Body Target Tracking (Eckenhoff et al. 2019)

  · Paper

Features:

  · Tightly-coupled estimator for visual-inertial localization and target tracking (VILTT)
    · Buiit upon visual-inertial navigation system (VINS) which is based on multi-state constraint
      Kalman filter (MSCKF) framework
    · Generalize the VINS to incorporate a 6DOF rigid-body target tracking of a 3D moving
      object
  · Key contribution: tightly-coupling of target motion estimation within VINS
  · Represent the target object as a 3D rigid-body model rather than a point with the following
    data:
    · 6 DOF target pose
    · Features attached
  · Introduced 3 stochastic target motion models for a broad range of realistic target tracking
    scenarios

2.1.4 Dense-ArthroSLAM: Dense Intra-Articular 3-D Reconstruction With Robust Localization Prior for Arthroscopy (Marmol, Banach, and Peynot 2019)
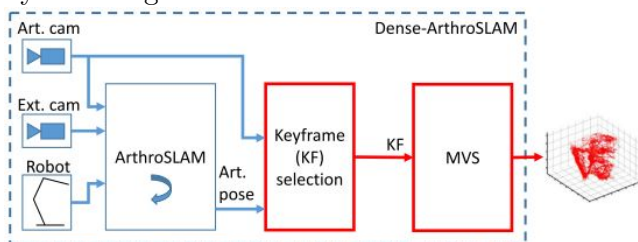
  · Paper

Features:

  · Dense-ArthroSLAM: a SLAM system based on *ArthroSLAM* for arthroscopy

- *ArthroSLAM*
    - A *sparse* SLAM system
    - 2 visual sensors are attached to the final link of a robotic arm
        - Arthroscopic camera for intra-articular tissues
        - External camera for the exterior of the patient anatomy
    - Raw camera measurements are fused with robot's odometry in an extended Kalman filter (EKF) framework
    - Role: as a robust localization prior for reliable camera pose information
- Improvement over original *ArthroSLAM*
    - Added a robust localizer with a keyframe selection strategy
        - A mix of 2 strategies:
            1) The amount of translational and rotational variation
            2) Certain number of frames
    - Added a batch multiview stereo (MVS) for 3D reconstruction
        - Re-detect SIFT features in the keyframes and do brute-force matching
        - Triangulation: 2 random keyframes having at least 5 degrees between viewing rays
        - 3D points and camera poses are refined using BA for every keyframe added to the optimization
        - Outlier removal scheme
        - Use a patch-match for per-pixel intensity similarity, and jointly optimize depth and normal of point clouds
        - Valid pixel observations are fused into a dense scene reconstruction of the tissues
            - Resulting point cloud with associated normals can be used to infer a surface mesh and approximate a reconstructed area
- Outperform state-of-the-art SLAM methods (*ORB-SLAM* for feature-based, and *LSD-SLAM* for direct-based) both in reliability and estimation accuracy

System diagram:

- Art.: Arthroscopic
- Ext.: External
- New components are in red

2.1.5 Modeling Perceptual Aliasing in SLAM via Discrete–Continuous Graphical Models (Lajoie et al. 2019)

- Paper

Features:

· A unified framework to model perceptual aliasing (wrong data association) in SLAM
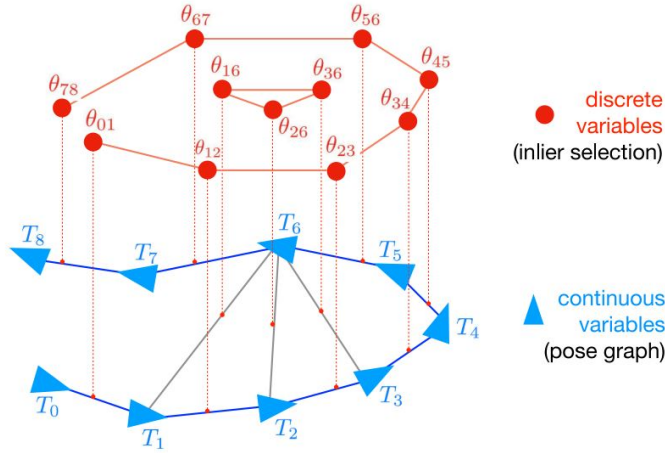   · Discrete-continuous graphical model (*DC-GM*)

Fig. 1. We introduce a *Discrete-Continuous Graphical Model* (DC-GM) to model perceptual aliasing in SLAM. The model describes the interactions between continuous variables (e.g., robot poses) and discrete variables (e.g., the binary selection of inliers and outliers), and captures the correlation between the discrete variables (e.g., due to perceptual aliasing).

    · Continuous portion: describes the standard SLAM formulation, i.e., a pose graph
    · Discrete portion: describes outlier selection and models correlation
        · A Markov random field
· Practical algorithms to cope with outliers without relying on initial guess for optimization
    · Semidefinite relaxation to perform inference in *DC-GM* and returns estimates with provable sub-optimality guarantees
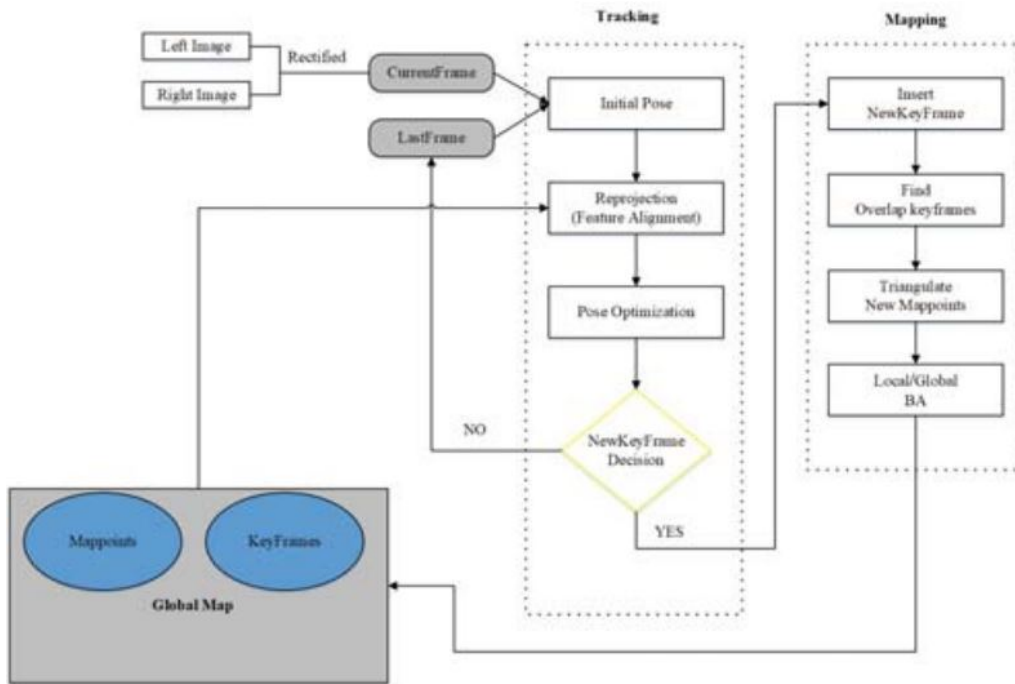
Limitations:

· MATLAB implementation: slow, and only tested for relatively small 2D SLAM problems

2.1.6 FMD Stereo SLAM: Fusing MVG and Direct Formulation Towards Accurate and Fast Stereo SLAM (F. Tang, Li, and Wu 2019)

· Paper

System diagram:

Features:

- Stereo visual SLAM system
- **F**usion of key-feature-based **M**ultiple view geometry (MVG) and **D**irect formulation (FMD)
    - Make full use of advantages from *SVO* and *ORB-SLAM*
- Front-end: make the system faster
    - Use *direct formulation* and *constant motion model* to predict an initial pose
    - Reproject local map for 3D-to-2D correspondences
    - Refine the pose by *reprojection error minimization*
- Back-end: make the system more accurate
    - Use MVG to estimate 3D structure
        - Perform triangulation for new map points when a new keyframe is inserted
        - Remove outliers in the map and maintain a global map by BA

2.1.7 GEN-SLAM: Generative Modeling for Monocular Simultaneous Localization and Mapping (Chakravarty, Narayanan, and Roussel 2019)
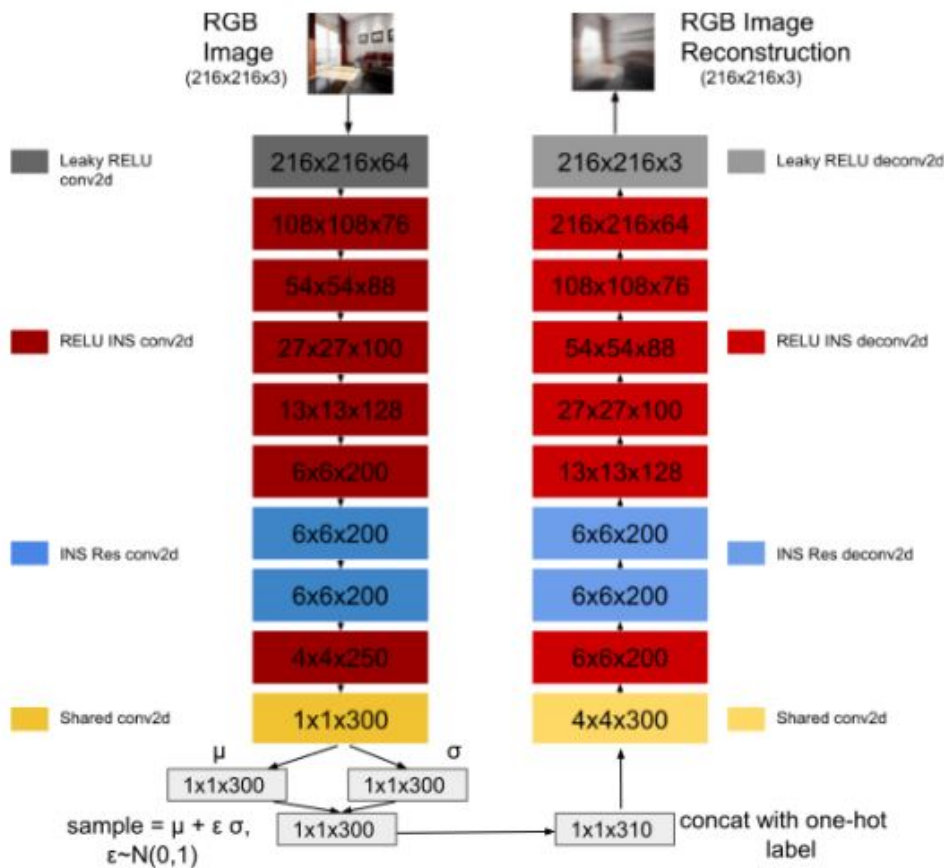
- Paper

System diagram:
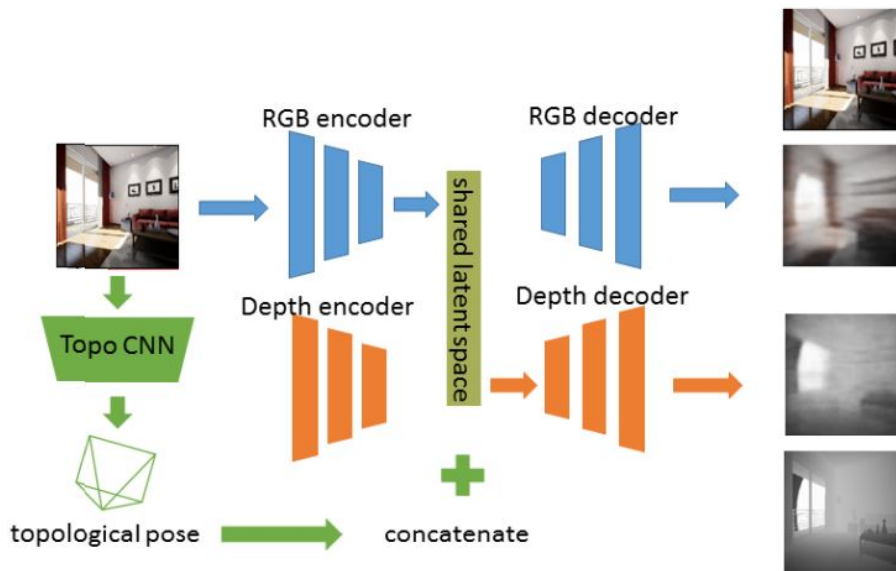
Features:

- Deep learning based visual SLAM system
- Use a *CNN* to localize in a topological map
- Generative modeling: encoder-decoder architecture with shared latent space for *variational auto encoder* (VAE)
    - 2 encoders and 2 decoders, and one each for RGB and depth respectively
    - Use a *conditional variational auto encoder* (VAE) for depth map, conditioned on the CNN-estimated pose
    - Employ 4 reconstruction losses to train the network
        - RGB -> RGB
        - Depth -> Depth
        - RGB -> Depth
        - Depth -> RGB
- RGB VAE structure:

2.1.8 RESLAM: A Real-time Robust Edge-based SLAM System (Schenk and Fraundorfer 2019)

- Paper

System diagram:
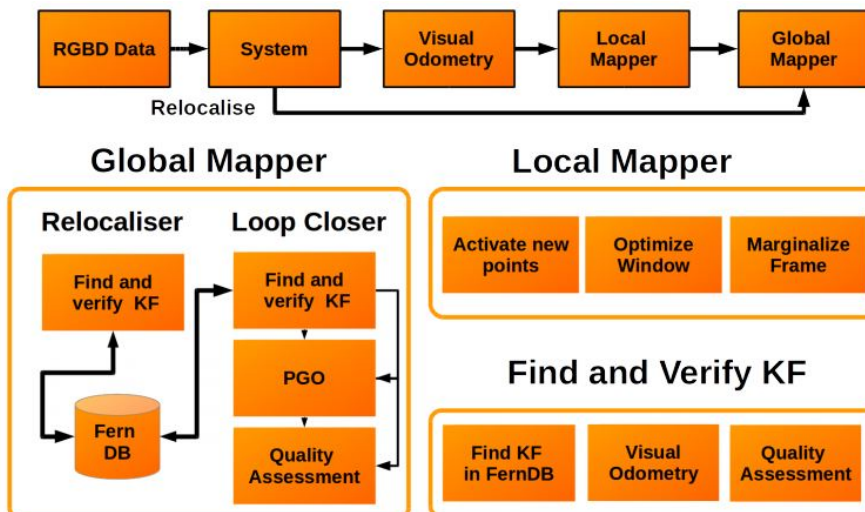
- 3 components
  - Visual odometry module: for relative camera motion estimation
  - Local mapper: manages keyframes and optimizes a local window
  - Global mapper: stores a global map and performs loop closure and relocalization

Features:

- · Edge-based SLAM system for RGBD sensors
    - · Canny edges are detected for each frame
- · Utilize edges in all the subsequent steps in the proposed SLAM pipeline:
    - · Camera pose estimation
    - · Local sliding window optimization for depth refinement
    - · Loop closure
    - · Relocalization
- · Initial depth refinement in a sliding window
- · Edge-based loop closure verification
- · Fast system that runs in real-time on a CPU
- · Open-source

2.1.9 Fast and Robust Initialization for Visual-Inertial SLAM (Campos, Montiel, and Tardós 2019)

- · Paper

Features:

- · A fast method for joint initialization of monocular-inertial SLAM built on the method proposed by (Martinelli 2014) and extended by (Kaiser et al. 2017) (the original method in the paper)
- · The original method is improved as follows:
    - · More general: allows incomplete feature tracks (features not seen by all cameras)
    - · More computationally efficient: uses the IMU preintegration method of (Scaramuzza 2015)
    - · Improved accuracy: uses 2 visual-inertial BA steps
    - · Improved robustness: uses novel observability and consensus tests to detect bad initializations
- · Scale errors decreases from up to 156% of the original method (215ms) to around 5% (1-2 seconds), or to less than 1% after performing visual-inertial BA after 10 seconds

2.1.10 Visual SLAM: Why Bundle Adjust? (Bustos et al. 2019)

- · Paper

Algorithm for the proposed system:

Algorithm 2 L-infinity SLAM.

1: **for** each keyframe step $t = 1, 2, \ldots$ **do**
2:     $s \leftarrow t - (\text{window size}) + 1$.
3:     $\{\mathbf{R}_{j,k}\}_{j,k \in \mathsf{N}_{\text{win}}} \leftarrow \text{relative\_rotation}(\mathsf{Z}_{(s-1):t})$.
4:     $\mathbf{R}_{s:t} \leftarrow \text{rotation\_averaging}\left(\{\mathbf{R}_{j,k}\}_{j,k \in \mathsf{N}_{\text{win}}}\right)$.
5:     $\mathbf{t}_{s:t}, \mathsf{X} \leftarrow \text{known\_rotation\_prob}(\mathbf{R}_{s:t}, \mathsf{Z}_{0:t})$.
6:     **if** loop is detected in Z_t **then**
7:         $\{\mathbf{R}_{j,k}\}_{j,k \in \mathsf{N}_{\text{sys}}} \leftarrow \text{relative\_rotation}(\mathsf{Z}_{0:t})$.
8:         $\mathbf{R}_{1:t} \leftarrow \text{rotation\_averaging}\left(\{\mathbf{R}_{j,k}\}_{j,k \in \mathsf{N}_{\text{sys}}}\right)$.
9:         $\mathbf{t}_{1:t}, \mathsf{X} \leftarrow \text{known\_rotation\_prob}(\mathbf{R}_{1:t}, \mathsf{Z}_{0:t})$.
10:     **end if**
11: **end for**

Features:

- · A SLAM system called L-infinity SLAM that is not based on BA
- · Solution to problems of BA-based SLAM systems
  - · Problems:
    - · The need to maintain an accurate map and camera motions at keyframe rate
    - · Cannot deal with slow motion and pure rotational motion because of the requirement on sufficient baselines for BA
  - · Solution:
    - · Conduct rotation averaging to optimize *camera orientations only* instead of BA
    - · Given the orientations, camera positions and 3D points are estimated via a quasi-convex optimization that can be *globally optimally* solved
- · Efficient loop closure module
  - · Incorporation of relative translation directions to solve camera drifts
- · Advantage of L-Infinity SLAM
  - · Many tasks of map maintenance (feature/map point selection, point culling, map updating and aggregation) can be done in a low priority thread or even offline
  - · The system can handle pure rotation motions

2.1.11 Illumination Robust Monocular Direct Visual Odometry for Outdoor Environment Mapping (Wu and Pradalier 2019)

- · Paper

Features:

- · An illumination-robust direct monocular VO system focusing on modeling outdoor scenery
- · Evaluation on state-of-the-art illumination invariant cost functions in the context of monocular joint optimization framework
- · An illumination-robust cost is proposed by combining intensity- and gradient-based costs with an adaptive weight
- · Illustration that sun glares can be modeled as local illumination changes, and its adverse effect on motion estimation can be alleviated by dealing with these local changes
- · Computational cost of the proposed system is higher than other methods

2.1.12 KO-Fusion: Dense Visual SLAM with Tightly-Coupled Kinematic and Odometric Tracking (Houseago, Bloesch, and Leutenegger 2019)

- · Paper

Features:

- · A dense SLAM system based on ElasticFusion
- · A full dense SLAM system on a mobile agent with omnidirectional motion capability where the transform between the camera frame and the agent body frame is allowed to change
- · Tightly-coupled kinematic information from the robot actuators and odometric information from the moving base

- Capability of real-time tracking and dense mapping at 30Hz while running on a GPU
- Joint optimization of scene geometry and robot actuator data to estimate the pose of both the robot base and a camera mounted on the end effector
- More robustness to fast and erratic motion with rapid motion, and a lack of significant photometric and geometric variation

2.1.13 DeepFusion: Real-Time Dense 3D Reconstruction for Monocular SLAM using Single-View Depth and Gradient Predictions (Laidlow, Czarnowski, and Leutenegger 2019)

- Paper

System diagram:
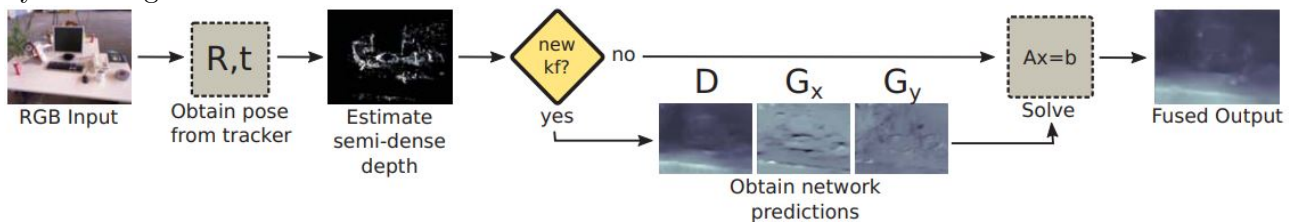
Features:

- A 3D reconstruction system that leverages the output of a CNN to produce fully dense depth maps for keyframes that include metric scale
- Capability of real-time dense reconstructions on a GPU
    - The network only needs to be run once per keyframe
- Fusion of the output of a semi-dense multiview stereo algorithm, with the depth and gradient predictions of a CNN, in a probabilistic fashion
- Pose is estimated using ORB-SLAM2 (monocular)
- Some details on the CNN in the paper:
    - It is used to predict the log-depth, log-depth gradients, and associated uncertainties from the new keyframe image
    - Log-depth gradients in the x- and y-directions are for the image planes rather than surface normals, in order to maintain the linearity of the optimization problem
    - Network structure: similar to U-Net architecture, with 3 more identical decoders added to predict log-depth uncertainties, log-depth gradients, and log-depth gradient uncertainties

2.2 Robotics: Science and Systems 2019 (RSS2019)

2.2.1 Continuous Direct Sparse Visual Odometry from RGB-D Images (Jadidi et al. 2019)

- Paper

Features:

- A direct and continuous energy-based formulation and solution for the RGB-D visual odometry problem (sensor registration problem)
    - Direct: relates to direct-based method

· Continuous: the formulation of the image registration problem, given discrete cloud measurements, is in a continuous form (based on continuous camera motion)
  · Operates from 3D space to an abstract information space such as intensity surface
  · Continuity is achieved via functional treatment of the problem, and representing the functions in a reproducing kernel Hilbert space
· Do not rely on the association between 2 measurements and the same number of measurements within each set because of the continuous formulation
· The developed framework is not limited to specific image resolution
· The framework is fully analytical and the gradient has a complete closed-form derivation
  · It is opposed to the current direct energy formulation:
    a) Computation of numerical image intensity gradient
    b) The use of the gradient with the Jacobian of the pose via the chain rule
· The proposed method can be an alternative to the core module of many modern visual odometry and tracking systems

2.2.2 PoseRBPF: A Rao-Blackwellized Particle Filter for 6D Object Pose Estimation (Deng et al. 2019)

· Paper

System diagram:
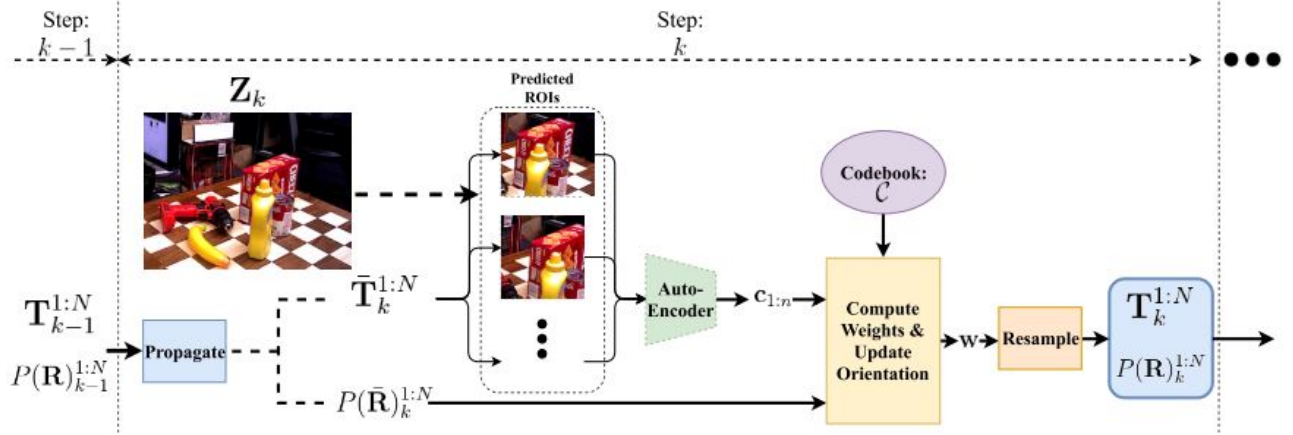
· Propagation: estimate current pose based on previous pose using motion prior
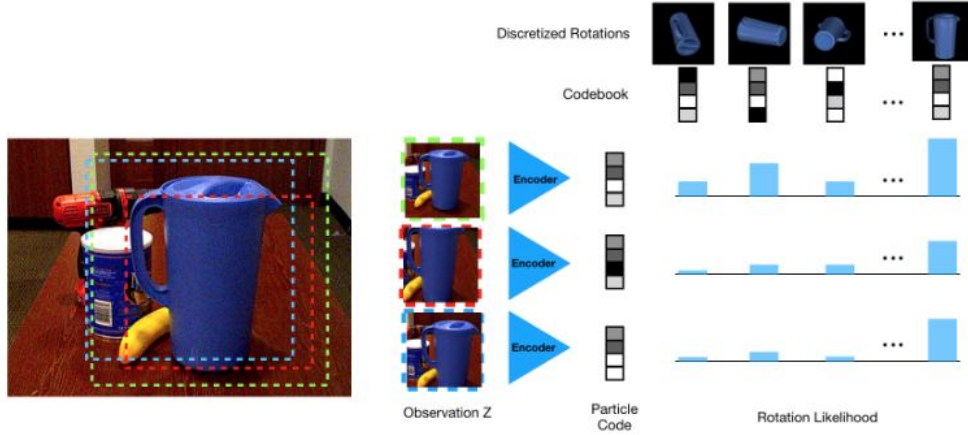
Rotation computation:

Fig. 3. Illustration of the computation for the conditional rotation likelihood by codebook matching. Left) Each particle crops the image based on its translation hypothesis. The RoI for each particle is resized and the corresponding code is computed using the encoder. Right) The rotation distribution $P(\mathbf{R}|\mathbf{Z}, \mathbf{T})$ is computed from the distance between the code for each hypothesis and those in the codebook.

Features:

- · PoseRBPF: 6D object pose tracking based on Rao-Blackwellized particle filter (RBPF) framework
- · Pose is factorized into translation and rotation parts
    - · Rotation
        - · Rotation space is discretized at 5 degree resolution ($72 \times 37 \times 72 = 191808$ bins for elevation only ranges from -90 to 90 degree)
        - · An auto-encoder network is trained as an observation model to construct a codebook of feature embeddings for the discretized rotations
- · Procedures: for each particle,
    1) Determine the center and the size of the object bounding box in the image via 3D translation data
    2) Determine embedding of the bounding box
    3) Update rotation distribution by comparing the embedding value with the pre-computed entries in the codebook using cosine distance
    4) Compute the weight of the particle from the normalization factor of the rotation distribution
    5) Update motion by sampling from a motion model over poses and a convolution over rotations

2.2.3 VIMO: Simultaneous Visual Inertial Model-Based Odometry and Force Estimation (Nisar et al. 2019)

Features:

- · VIMO: visual inertial model-based odometry with an added motion constraint
    - · Motion constraint is derived from the dynamic model including *external forces*
    - · Model dynamics and external force are combined in a preintegrated residual
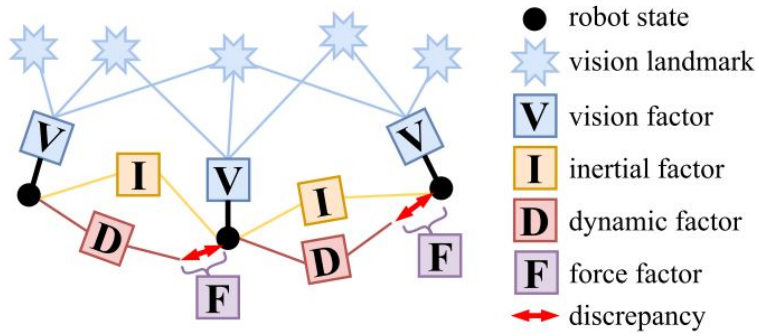
- · The residual makes current visual inertial odometry (VIO) framework jointly estimate this force in addition to the robot state and IMU bias
- · Built upon a state-of-the-art VIO system the VINS-mono system
- · Tightly-coupled, sliding-window visual inertial estimator
- · Input: visual-inertial measurements & the collective thrust
- · Estimation strategy: nonlinear optimization (BA)
- · (1st approach to) Exploit model dynamics by jointly estimating motion and external force
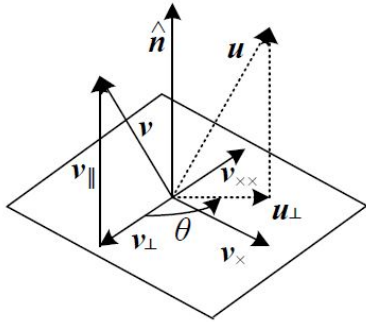
Miscellaneous

1. Basics

1.1 3D Rotation

1.1.1 Rotation Representations

a. $\mathbf{R}$ itself
- · 3×3 matrix

b. Axis/angle
- · Represent rotation by rotation axis $\hat{\mathbf{n}}$ and angle θ
- · Derivation of $\mathbf{R}(\hat{\mathbf{n}}, \theta)$:

- · A rotation from $\mathbf{v}$ to $\mathbf{u}$ based on rotation axis $\hat{\mathbf{n}}$ and rotation angle θ
- · $[\hat{\mathbf{n}}]_\times$: matrix form of cross product operator with vector $\hat{\mathbf{n}} = (\hat{n}_x, \hat{n}_y, \hat{n}_z)$
- · $[\hat{\mathbf{n}}]_\times = \begin{bmatrix} 0 & -\hat{n}_z & \hat{n}_y \\ \hat{n}_z & 0 & -\hat{n}_x \\ -\hat{n}_y & \hat{n}_x & 0 \end{bmatrix}$
- · $\mathbf{u} = \mathbf{u}_\perp + \mathbf{v}_\parallel = \mathbf{R}(\hat{\mathbf{n}}, \theta)\mathbf{v}$
- · $\mathbf{R}(\hat{\mathbf{n}}, \theta) = \mathbf{I} + \sin\theta[\hat{\mathbf{n}}]_\times + (1 - \cos\theta)[\hat{\mathbf{n}}]_\times^2$
  - · Known as *Rodrigues' formula*

c. Quaternions
- · 4-vector: $\mathbf{q} = (q_x, q_y, q_z, q_w)$ or $\mathbf{q} = (x, y, z, w)$
- · Unit quaternions: live on the sphere $\|\mathbf{q}\| = 1$
- · *Antipodal* (opposite sign) quaternions $\mathbf{q}$ and $-\mathbf{q}$ represent the same rotation
- · Representation is *continuous*: rotation matrices vary continuously
- · Quaternions can be derived from the axis/angle representation:
  - · $\mathbf{q} = (\mathbf{v}, w) = (\sin\dfrac{\theta}{2}\hat{\mathbf{n}}, \cos\dfrac{\theta}{2})$ where $\hat{\mathbf{n}}$ and θ are rotation axis and angle
- · Convert Rodrigues' formula using double-angle formulas:

$$\mathbf{R}(\hat{\mathbf{n}}, \theta) = \mathbf{I} + \sin\theta[\hat{\mathbf{n}}]_\times + (1 - \cos\theta)[\hat{\mathbf{n}}]_\times^2 = \mathbf{I} + 2w[\mathbf{v}]_\times + 2[\mathbf{v}]_\times^2$$

· $\mathbf{R}(\mathbf{q}) = \begin{bmatrix} 1 - 2(y^2 + z^2) & 2(xy - zw) & 2(xz + yw) \\ 2(xy + zw) & 1 - 2(x^2 + z^2) & 2(yz - xw) \\ 2(xz - yw) & 2(yz + xw) & 1 - 2(x^2 + y^2) \end{bmatrix}$

where $\mathbf{v} = (x, y, z)$ and $[\mathbf{v}]_\times = \begin{bmatrix} 0 & -z & y \\ z & 0 & -x \\ -y & x & 0 \end{bmatrix}$

· Quaternion multiply operator:

$$\mathbf{q}_2 = \mathbf{q}_0\mathbf{q}_1 = (\mathbf{v}_0 \times \mathbf{v}_1 + w_0\mathbf{v}_1 + w_1\mathbf{v}_0, w_0w_1 - \mathbf{v}_0 \cdot \mathbf{v}_1)$$

   · $\mathbf{R}(\mathbf{q}_2) = \mathbf{R}(\mathbf{q}_0)\mathbf{R}(\mathbf{q}_1)$

   · Multiply operator is *not commutative*

· Quaternion division: just flip the sign of $\mathbf{v}$ or w in the multiply operator

$$\mathbf{q}_2 = \mathbf{q}_0/\mathbf{q}_1 = \mathbf{q}_0\mathbf{q}_1^{-1} = (\mathbf{v}_0 \times \mathbf{v}_1 + w_0\mathbf{v}_1 - w_1\mathbf{v}_0, -w_0w_1 - \mathbf{v}_0 \cdot \mathbf{v}_1)$$

d. Euler angles

   · 3 angles to describe the orientation of a rigid body w.r.t. a fixed coordinate system

      · α, β, γ; or ϕ, θ, ψ

   · 2 groups of angles with 12 possible sequences of rotation axes

      (1) **Proper/Classic Euler angles**: *z-x-z, x-y-x, y-z-y, z-y-z, x-z-x, y-x-y*

      (2) **Tait-Bryan angles**: *x-y-z, y-z-x, z-x-y, x-z-y, z-y-x, y-x-z*

         · Also called: **Cardan angles**; **nautical angles**; **heading, elevation, and bank**; **yaw, pitch, and roll**

   · Proper Euler angles

      · Geometrical definition (one possibility):
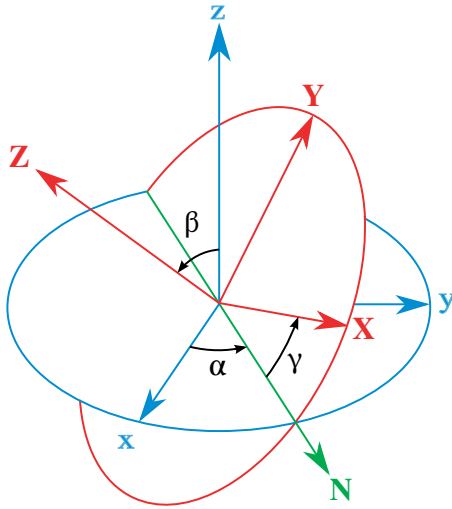
   · *x, y, z* : axes of the original frame

      · *X, Y, Z*: axes of the rotated frame

      · $N = z \times Z$ (i.e., the intersection of planes *xy* and *XY*)

      · α, β, γ: Euler angles

   · Definition by intrinsic rotations

      · Intrinsic rotations: *XYZ* system rotates after each elementary rotation; *xyz* system is fixed

      · Rotated frame *XYZ* from initial orientation to final orientation:

         · Initial: *x-y-z* or x_0-y_0-z_0

- · After 1st rotation: x'-y'-z' or x_1-y_1-z_1
- · After 2nd rotation: x''-y''-z'' or x_2-y_2-z_2
- · Final (after 3rd rotation): X-Y-Z or x_3-y_3-z_3
  - · Definition of Euler angles:
    - · α (or ϕ): a rotation around the z axis
    - · β (or θ): a rotation around the x' axis
    - · γ (or ψ): a rotation around the z'' axis
- · Definition by extrinsic rotations
  - · Extrinsic rotations: elemental rotations occur about the axes of fixed coordinate system xyz
  - · Target orientation can be achieved by rotating γ, β, and α around z, x, z axes respectively
- · Tait-Bryan angles
  - · Difference with proper Euler angles: Tait-Bryan angles represent rotations about 3 distinct axes (e.g. x-y-z or x-y'-z''), while proper Euler angles use same axis for both 1st & 3rd elemental rotations (e.g. z-x-z or z-x'-z'')
  - · Geometrical definition (one possibility):
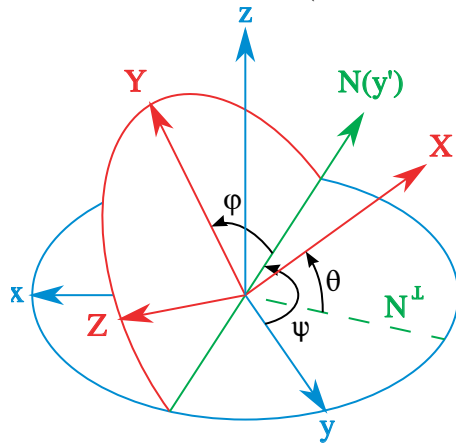
  - · Different line of nodes compared with proper Euler angles
    - · Intersection of planes xy and YZ for Tait-Bryan angles vs. xy and XY for proper Euler angles
  - · Rotation sequence: z-y'-x''
    - · Intrinsic rotations; N coincides with y'
    - · Known as **yaw, pitch, and roll**
  - · Angle rotation sequence: ψ, θ, ϕ

1.1.2 Conversions Between Different Rotation Representations

- · References:
  - · wiki page
- · Rotation matrix decomposition (wiki):
  - · Any orientation can be achieved by composing 3 elemental rotations
  - · Example: $\mathbf{R} = \mathbf{X}(\alpha)\mathbf{Y}(\beta)\mathbf{Z}(\gamma)$
    - · 2 possible representations
    - (1) A composition of **extrinsic** rotations about axes z-y-x

(2) A composition of **intrinsic** rotations about axes x-y'-z''

· Rotation matrix from all 12 kinds of Euler angles:

For the sake of simplicity, the following table uses the following nomenclature:

1. 1, 2, 3 represent the angles α, β and γ, i.e. the angles corresponding to the first, second and third elemental rotations respectively.
2. X, Y, Z are the matrices representing the elemental rotations about the axes x, y, z of the fixed frame (e.g., X_1 represents a rotation about x by an angle α).
3. s and c represent sine and cosine (e.g., s_1 represents the sine of α).
4. Each matrix is denoted by the formula used to calculate it. If $R = Z_1 X_2 Z_3$, we name it $Z_1 X_2 Z_3$.

Proper Euler angles	Tait–Bryan angles
$X_1 Z_2 X_3 = \begin{bmatrix} c_2 & -c_3 s_2 & s_2 s_3 \\ c_1 s_2 & c_1 c_2 c_3 - s_1 s_3 & -c_3 s_1 - c_1 c_2 s_3 \\ s_1 s_2 & c_1 s_3 + c_2 c_3 s_1 & c_1 c_3 - c_2 s_1 s_3 \end{bmatrix}$	$X_1 Z_2 Y_3 = \begin{bmatrix} c_2 c_3 & -s_2 & c_2 s_3 \\ s_1 s_3 + c_1 c_3 s_2 & c_1 c_2 & c_1 s_2 s_3 - c_3 s_1 \\ c_3 s_1 s_2 - c_1 s_3 & c_2 s_1 & c_1 c_3 + s_1 s_2 s_3 \end{bmatrix}$
$X_1 Y_2 X_3 = \begin{bmatrix} c_2 & s_2 s_3 & c_3 s_2 \\ s_1 s_2 & c_1 c_3 - c_2 s_1 s_3 & -c_1 s_3 - c_2 c_3 s_1 \\ -c_1 s_2 & c_3 s_1 + c_1 c_2 s_3 & c_1 c_2 c_3 - s_1 s_3 \end{bmatrix}$	$X_1 Y_2 Z_3 = \begin{bmatrix} c_2 c_3 & -c_2 s_3 & s_2 \\ c_1 s_3 + c_3 s_1 s_2 & c_1 c_3 - s_1 s_2 s_3 & -c_2 s_1 \\ s_1 s_3 - c_1 c_3 s_2 & c_3 s_1 + c_1 s_2 s_3 & c_1 c_2 \end{bmatrix}$
$Y_1 X_2 Y_3 = \begin{bmatrix} c_1 c_3 - c_2 s_1 s_3 & s_1 s_2 & c_1 s_3 + c_2 c_3 s_1 \\ s_2 s_3 & c_2 & -c_3 s_2 \\ -c_3 s_1 - c_1 c_2 s_3 & c_1 s_2 & c_1 c_2 c_3 - s_1 s_3 \end{bmatrix}$	$Y_1 X_2 Z_3 = \begin{bmatrix} c_1 c_3 + s_1 s_2 s_3 & c_3 s_1 s_2 - c_1 s_3 & c_2 s_1 \\ c_2 s_3 & c_2 c_3 & -s_2 \\ c_1 s_2 s_3 - c_3 s_1 & c_1 c_3 s_2 + s_1 s_3 & c_1 c_2 \end{bmatrix}$
$Y_1 Z_2 Y_3 = \begin{bmatrix} c_1 c_2 c_3 - s_1 s_3 & -c_1 s_2 & c_3 s_1 + c_1 c_2 s_3 \\ c_3 s_2 & c_2 & s_2 s_3 \\ -c_1 s_3 - c_2 c_3 s_1 & s_1 s_2 & c_1 c_3 - c_2 s_1 s_3 \end{bmatrix}$	$Y_1 Z_2 X_3 = \begin{bmatrix} c_1 c_2 & s_1 s_3 - c_1 c_3 s_2 & c_3 s_1 + c_1 s_2 s_3 \\ s_2 & c_2 c_3 & -c_2 s_3 \\ -c_2 s_1 & c_1 s_3 + c_3 s_1 s_2 & c_1 c_3 - s_1 s_2 s_3 \end{bmatrix}$
$Z_1 Y_2 Z_3 = \begin{bmatrix} c_1 c_2 c_3 - s_1 s_3 & -c_3 s_1 - c_1 c_2 s_3 & c_1 s_2 \\ c_1 s_3 + c_2 c_3 s_1 & c_1 c_3 - c_2 s_1 s_3 & s_1 s_2 \\ -c_3 s_2 & s_2 s_3 & c_2 \end{bmatrix}$	$Z_1 Y_2 X_3 = \begin{bmatrix} c_1 c_2 & c_1 s_2 s_3 - c_3 s_1 & s_1 s_3 + c_1 c_3 s_2 \\ c_2 s_1 & c_1 c_3 + s_1 s_2 s_3 & c_3 s_1 s_2 - c_1 s_3 \\ -s_2 & c_2 s_3 & c_2 c_3 \end{bmatrix}$
$Z_1 X_2 Z_3 = \begin{bmatrix} c_1 c_3 - c_2 s_1 s_3 & -c_1 s_3 - c_2 c_3 s_1 & s_1 s_2 \\ c_3 s_1 + c_1 c_2 s_3 & c_1 c_2 c_3 - s_1 s_3 & -c_1 s_2 \\ s_2 s_3 & c_3 s_2 & c_2 \end{bmatrix}$	$Z_1 X_2 Y_3 = \begin{bmatrix} c_1 c_3 - s_1 s_2 s_3 & -c_2 s_1 & c_1 s_3 + c_3 s_1 s_2 \\ c_3 s_1 + c_1 s_2 s_3 & c_1 c_2 & s_1 s_3 - c_1 c_3 s_2 \\ -c_2 s_3 & s_2 & c_2 c_3 \end{bmatrix}$

To change the formulas for passive rotations (or find reverse active rotation), transpose the matrices (then each matrix transforms the initial coordinates of a vector remaining fixed to the coordinates of the same vector measured in the rotated reference system; same rotation axis, same angles, but now the coordinate system rotates, rather than the vector).

· Rotation matrix to yaw, pitch, roll in Eigen (ref):

· `R.eulerAngles(2, 1, 0).reverse;`

References

Bailey, T., and H. Durrant-Whyte. 2006. "Simultaneous Localization and Mapping (SLAM): Part II." *IEEE Robotics Automation Magazine* 13 (3): 108–17. https://doi.org/10.1109/MRA.2006.1678144.

Baker, Simon, and Iain Matthews. 2004. "Lucas-Kanade 20 Years on: A Unifying Framework." *International Journal of Computer Vision* 56 (3): 221–55.

Barfoot, Timothy D. 2017. *State Estimation for Robotics.* 1st ed. New York, NY, USA: Cambridge University Press.

Benhimane, Selim, and Ezio Malis. 2004. "Real-Time Image-Based Tracking of Planes Using Efficient Second-Order Minimization." In *2004 Ieee/Rsj International Conference on Intelligent Robots and Systems (Iros)(IEEE Cat. No. 04CH37566)*, 1:943–48. IEEE.

Bustos, Á. P., T. Chin, A. Eriksson, and I. Reid. 2019. "Visual Slam: Why Bundle Adjust?" In *2019 International Conference on Robotics and Automation (Icra)*, 2385–91. https://doi.org/10.1109/ICRA.2019.8793749.

Cadena, Cesar, Luca Carlone, Henry Carrillo, Yasir Latif, Davide Scaramuzza, José Neira, Ian D. Reid, and John J. Leonard. 2016. "Simultaneous Localization and Mapping: Present, Future, and the Robust-Perception Age." *CoRR* abs/1606.05830. http://arxiv.org/abs/1606.05830.

Campos, C., J. M. M. Montiel, and J. D. Tardós. 2019. "Fast and Robust Initialization for Visual-Inertial Slam." In *2019 International Conference on Robotics and Automation (Icra)*, 1288–94. https://doi.org/10.1109/ICRA.2019.8793718.

Chakravarty, P., P. Narayanan, and T. Roussel. 2019. "GEN-Slam: Generative Modeling for Monocular Simultaneous Localization and Mapping." In *2019 International Conference on Robotics and Automation (Icra)*, 147–53. https://doi.org/10.1109/ICRA.2019.8793530.

Deng, Xinke, Arsalan Mousavian, Yu Xiang, Fei Xia, Timothy Bretl, and Dieter Fox. 2019. "PoseRBPF: A Rao-Blackwellized Particle Filter for 6D Object Pose Estimation." In *Proceedings of Robotics: Science and Systems.* Freiburg im Breisgau, Germany. https://doi.org/10.15607/RSS.2019.XV.049.

Dissanayake, G., S. Huang, Z. Wang, and R. Ranasinghe. 2011. "A Review of Recent Developments in Simultaneous Localization and Mapping." In *2011 6th International Conference on Industrial and Information Systems*, 477–82. https://doi.org/10.1109/ICIINFS.2011.6038117.

Durrant-Whyte, H., and T. Bailey. 2006. "Simultaneous Localization and Mapping: Part I." *IEEE Robotics Automation Magazine* 13 (2): 99–110. https://doi.org/10.1109/MRA.2006.1638022.

Eckenhoff, K., Y. Yang, P. Geneva, and G. Huang. 2019. "Tightly-Coupled Visual-Inertial Localization and 3-d Rigid-Body Target Tracking." *IEEE Robotics and Automation Letters* 4 (2): 1541–8. https://doi.org/10.1109/LRA.2019.2896472.

Engel, Jakob, Thomas Schöps, and Daniel Cremers. 2014. "LSD-Slam: Large-Scale Direct Monocular Slam." In *European Conference on Computer Vision*, 834–49. Springer.

Engel, Jakob, Jurgen Sturm, and Daniel Cremers. 2013. "Semi-Dense Visual Odometry for a Monocular Camera." In *Proceedings of the Ieee International Conference on Computer Vision*, 1449–56.

Forster, Christian, Matia Pizzoli, and Davide Scaramuzza. 2014. "SVO: Fast Semi-Direct Monocular Visual Odometry." In *2014 Ieee International Conference on Robotics and Automation (Icra)*, 15–22. IEEE.

Glover, A., W. Maddern, M. Warren, S. Reid, M. Milford, and G. Wyeth. 2011. "OpenFABMAP: An Open Source Toolbox for Appearance-Based Loop Closure Detection." In *The International Conference on Robotics and Automation.* St Paul, Minnesota: IEEE.

Hartley, R. I., and A. Zisserman. 2004. *Multiple View Geometry in Computer Vision.* Second. Cambridge University Press, ISBN: 0521540518.

Houseago, C., M. Bloesch, and S. Leutenegger. 2019. "KO-Fusion: Dense Visual Slam with Tightly-Coupled Kinematic and Odometric Tracking." In *2019 International Conference on Robotics and Automation (Icra)*, 4054–60. https://doi.org/10.1109/ICRA.2019.8793471.

Jadidi, Maani Ghaffari, William Clark, Anthony Bloch, Ryan Eustice, and Jessey W. Grizzle. 2019. "Continuous Direct Sparse Visual Odometry from Rgb-d Images." In *Proceedings of Robotics: Science and Systems.* Freiburg im Breisgau, Germany. https://doi.org/10.15607/RSS.2019.XV.044.

Kaiser, J., A. Martinelli, F. Fontana, and D. Scaramuzza. 2017. "Simultaneous State Initialization and Gyroscope Bias Calibration in Visual Inertial Aided Navigation." *IEEE Robotics and Automation Letters* 2 (1): 18–25. https://doi.org/10.1109/LRA.2016.2521413.

Laidlow, T., J. Czarnowski, and S. Leutenegger. 2019. "DeepFusion: Real-Time Dense 3D Reconstruction for Monocular Slam Using Single-View Depth and Gradient Predictions." In *2019 International Conference on Robotics and Automation (Icra)*, 4068–74. https://doi.org/10.1109/ICRA.2019.8793527.

Lajoie, P., S. Hu, G. Beltrame, and L. Carlone. 2019. "Modeling Perceptual Aliasing in Slam via Discrete–Continuous Graphical Models." *IEEE Robotics and Automation Letters* 4 (2): 1232–9. https://doi.org/10.1109/LRA.2019.2894852.

Lee, S. H., and J. Civera. 2019. "Loosely-Coupled Semi-Direct Monocular Slam." *IEEE Robotics and Automation Letters* 4 (2): 399–406. https://doi.org/10.1109/LRA.2018.2889156.

Marmol, A., A. Banach, and T. Peynot. 2019. "Dense-ArthroSLAM: Dense Intra-Articular 3-d Reconstruction with Robust Localization Prior for Arthroscopy." *IEEE Robotics and Automation Letters* 4 (2): 918–25. https://doi.org/10.1109/LRA.2019.2892199.

Martinelli, Agostino. 2014. "Closed-Form Solution of Visual-Inertial Structure from Motion." *International Journal of Computer Vision* 106 (2): 138–52.

Mourikis, A. I., and S. I. Roumeliotis. 2007. "A Multi-State Constraint Kalman Filter for Vision-Aided Inertial Navigation." In *Proceedings 2007 Ieee International Conference on Robotics and Automation*, 3565–72. https://doi.org/10.1109/ROBOT.2007.364024.

Nisar, Barza, Philipp Foehn, Davide Falanga, and Davide Scaramuzza. 2019. "VIMO: Simultaneous Visual Inertial Model-Based Odometry and Force Estimation." In *Proceedings of Robotics: Science and Systems.* Freiburg im Breisgau, Germany. https://doi.org/10.15607/RSS.2019.XV.082.

Scaramuzza, Christian Forster AND Luca Carlone AND Frank Dellaert AND Davide. 2015. "IMU Preintegration on Manifold for Efficient Visual-Inertial Maximum-a-Posteriori Estimation." In *Proceedings of Robotics: Science and Systems.* Rome, Italy. https://doi.org/10.15607/RSS.2015.XI.006.

Schenk, F., and F. Fraundorfer. 2019. "RESLAM: A Real-Time Robust Edge-Based Slam System." In *2019 International Conference on Robotics and Automation (Icra)*, 154–60. https://doi.org/10.1109/ICRA.2019.8794462.

Sola, Joan. 2017. "Quaternion Kinematics for the Error-State Kalman Filter." *arXiv Preprint arXiv:1711.02508.*

Tang, F., H. Li, and Y. Wu. 2019. "FMD Stereo Slam: Fusing Mvg and Direct Formulation Towards Accurate and Fast Stereo Slam." In *2019 International Conference on Robotics and Automation (Icra)*, 133–39. https://doi.org/10.1109/ICRA.2019.8793664.

Tang, J., J. Folkesson, and P. Jensfelt. 2019. "Sparse2Dense: From Direct Sparse Odometry to Dense 3-d Reconstruction." *IEEE Robotics and Automation Letters* 4 (2): 530–37. https://doi.org/10.1109/LRA.2019.2891433.

Vogiatzis, George, and Carlos Hernández. 2011. "Video-Based, Real-Time Multi-View Stereo." *Image and Vision Computing* 29 (7): 434–41.

Wu, X., and C. Pradalier. 2019. "Illumination Robust Monocular Direct Visual Odometry for Outdoor Environment Mapping." In *2019 International Conference on Robotics and Automation (Icra)*, 2392–8. https://doi.org/10.1109/ICRA.2019.8793607.