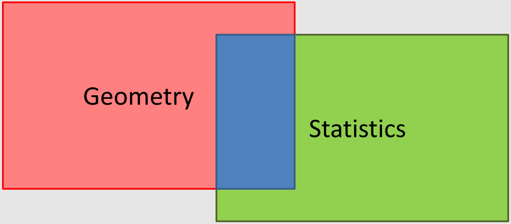
MOTION AVERAGING IN 3D VISION

A Framework for Efficient and Accurate Large-Scale Camera Estimation

Venu Madhav Govindu
Department of Electrical Engineering
Indian Institute of Science, Bengaluru
INDIA

21 July 2017

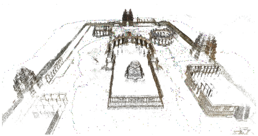
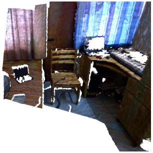
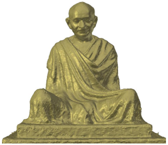
Introduction



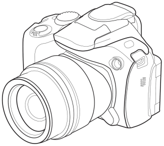
Geometric Estimation of Camera Motion

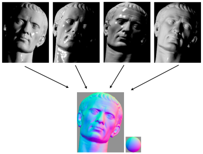
- **Geometric Representations** are fundamental in vision
- **Statistical Estimation** is of importance
- Our problems lie at the intersection of two disciplines
- 3D estimation of geometry
  - Recover both 3D *structure* and camera *motion*
  - Structure-from-motion using RGB images
  - 3D modeling using depth cameras

Introduction


- Geometry at the core of 3D reconstruction
- Consider two sources of data
  - camera images (RGB)
  - depth maps (Kinect etc.)
  - ~~Radiometry~~
- We are concerned with recovering camera motion

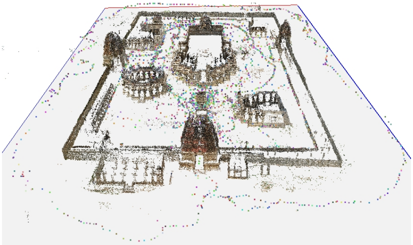


From http://www.clipartkid.com

https://commons.wikimedia.org/w/index.php?curid=44925507

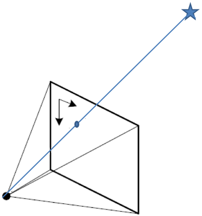
Introduction



Structure from Motion

- Classical problem of recovering 3D structure given multiple images
- Significant advances in past two decades
    - deeper theoretical understanding (projective geometry)
    - robust, efficient algorithms
    - can handle ever growing sizes of image datasets

Introduction

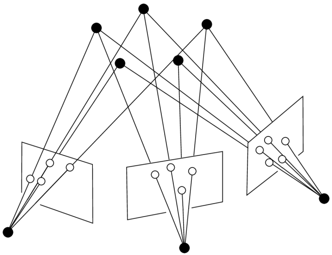


$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbf{K} \left[\begin{array}{c|c} \mathbf{R} & \mathbf{T} \end{array} \right] \begin{bmatrix} \boldsymbol{X} \\ \boldsymbol{Y} \\ \boldsymbol{Z} \\ 1 \end{bmatrix}$$

Camera Projection Model

- Pin-hole Projection Equation
- 3D Point Representation
- Rigid Motion of Camera $\{\boldsymbol{R}, \boldsymbol{T}\}$
- Camera Calibration $\boldsymbol{K}$ (assume $\boldsymbol{K} = \boldsymbol{I}$)
- Projection on image plane (u, v)
- Non-linear projection is key problem

Introduction



$$\min_{\boldsymbol{R},\boldsymbol{T},\boldsymbol{S}} \sum d^2(u_i^k, \hat{u}_i^k) + d^2(v_i^k, \hat{v}_i^k)$$

Bundle Adjustment

- i-th camera (motion), k-th 3D point (structure)
- Non-linear minimization over structure and motion variables
- High dimensional problem: local minima, initialization
- Huge number of advances on many fronts
- State-of-the-art approaches are incremental (Bundler, VSFM)
- Impressive quality but still suffer from limitations/problems
- Cannot recover from errors, no global view of SfM problem

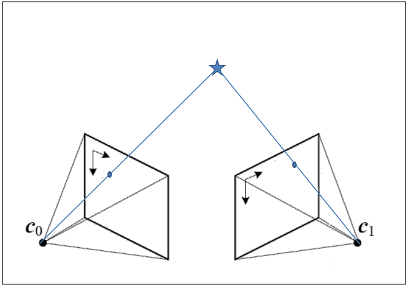
Figure from Foerstner *et al.* *Photogrammetric Computer Vision* 2016

Introduction

KEY PROBLEM IN SfM

- Non-linear interaction between structure and motion components
- Structure and motion "entangled" in observed image projections

Introduction



$$x'^T \mathbf{E} x = 0$$

$$\mathbf{E} = R[T]_\times$$

Can decompose $E \longrightarrow (R, t)$

Contrasting properties

- Motion estimation for two (few) views
  - is fast
  - Structure eliminated algebraically, but less accurate
  - Two-view or epipolar geometry well studied
- *Bundle Adjustment* : many images, slow, accurate
- *Algebraic Approaches* : few images, fast, low accuracy

Eliminate structure from SfM problem

Motion Averaging

- Simplify problem to camera motion estimation
- Achieved by “factoring” out structure from global problem
- Formulated in terms of graph representation

Typical Pipeline

- Feature matches across views (viewgraph)
- Pairwise geometry estimation (5-pt algorithm, two-view BA)
- Typically solve **rotations averaging** first, then **translation**
- Recover structure given motion (triangulation)
- Use solution as initialisation for batch BA

Eliminate structure from SfM problem

Motion Averaging

- Batch approach for camera motion estimation feasible
- Allows for a 'global' view of the problem
- Naturally accounts for loop closures
- Avoids pitfalls of incremental methods
- Desiderata: robust, fast, scalable, accurate
- Efficient solution can be refined by batch BA
- Similar approaches
  - Time synchronizations of wireless networks
  - Multi-dimensional Scaling

Introduction

Pin-hole Model

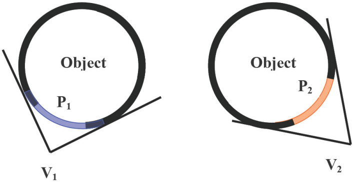
$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \left[\, \mathbf{R} \mid \mathbf{T} \,\right] \begin{bmatrix} \boldsymbol{X} \\ \boldsymbol{Y} \\ \boldsymbol{Z} \\ 1 \end{bmatrix}$$

$$= \overbrace{\left[\, \mathbf{I} \mid \mathbf{0} \,\right]}^{\text{Ideal camera}} \underbrace{\left[\begin{array}{c|c} \mathbf{R} & \mathbf{T} \\ \hline \mathbf{0} & 1 \end{array}\right]}_{\text{Camera Motion } \boldsymbol{M}} \begin{bmatrix} \boldsymbol{X} \\ \boldsymbol{Y} \\ \boldsymbol{Z} \\ 1 \end{bmatrix}$$

Properties

- Image formation is a projective mapping ($\mathbb{P}^3 \longrightarrow \mathbb{P}^2$)
- Camera motion is a Euclidean motion representation (rigid body)
- Has nice geometric properties we can exploit
- Same with 3D rotation $\boldsymbol{R}$
- Two-view geometry does not give translation scale

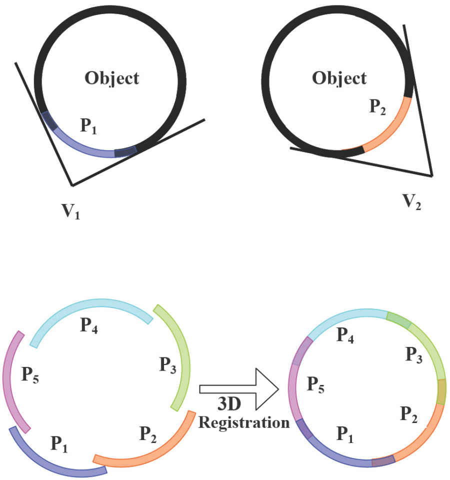
Introduction



Registration of 3D Scans

- Each scan is a *partial* model
- Has own *local* frame of reference
- Need to compute Euclidean motion to register/align scans

Introduction



- Need to **register** partial scans
- Need a **single** common frame of reference

Introduction

Pin-hole Model

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{I} \mid \mathbf{0} \end{bmatrix} \underbrace{\left[\begin{array}{c|c} \mathbf{R} & \mathbf{T} \\ \hline \mathbf{0} & 1 \end{array}\right]}_{\boldsymbol{M}} \begin{bmatrix} \boldsymbol{X} \\ \boldsymbol{Y} \\ \boldsymbol{Z} \\ 1 \end{bmatrix}$$

3D Registration

$$\begin{bmatrix} \boldsymbol{X}' \\ \boldsymbol{Y}' \\ \boldsymbol{Z}' \\ 1 \end{bmatrix} = \underbrace{\left[\begin{array}{c|c} \boldsymbol{R} & \boldsymbol{T} \\ \hline \mathbf{0} & 1 \end{array}\right]}_{\boldsymbol{M}} \begin{bmatrix} \boldsymbol{X} \\ \boldsymbol{Y} \\ \boldsymbol{Z} \\ 1 \end{bmatrix}$$

Types of Motion Representations

We will consider averaging problems for

- $\boldsymbol{M} \in \mathbb{SE}(3)$ (Rigid 3D Euclidean Motion)
- $\boldsymbol{R} \in \mathbb{SO}(3)$ (Rigid 3D Rotation)
- $\boldsymbol{t} \in \mathcal{S}^2$ (Translation Direction $\boldsymbol{t} = \frac{\boldsymbol{T}}{||\boldsymbol{T}||}$)

Introduction

First Rotate then Translate

$$
\begin{aligned}
\boldsymbol{P}^{'} &= \boldsymbol{R}\boldsymbol{P} + \boldsymbol{T} \\
&\Downarrow \\
\boldsymbol{P}_i &= \boldsymbol{R}_i\boldsymbol{P}_0 + \boldsymbol{T}_i \\
\boldsymbol{P}_j &= \boldsymbol{R}_j\boldsymbol{P}_0 + \boldsymbol{T}_j \\
&\Downarrow \\
\boldsymbol{R}_{ij} &= \boldsymbol{R}_j\boldsymbol{R}_i^{-1} \\
\boldsymbol{T}_{ij} &= \boldsymbol{T}_j - \boldsymbol{R}_j\boldsymbol{R}_i^{-1}\boldsymbol{T}_i
\end{aligned}
$$

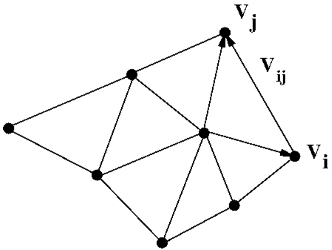
First Translate the Rotate

$$
\begin{aligned}
\boldsymbol{P}^{'} &= \boldsymbol{R}(\boldsymbol{P} + \boldsymbol{T}) \\
&\Downarrow \\
\boldsymbol{P}_i &= \boldsymbol{R}_i(\boldsymbol{P}_0 + \boldsymbol{T}_i) \\
\boldsymbol{P}_j &= \boldsymbol{R}_j(\boldsymbol{P}_0 + \boldsymbol{T}_j) \\
&\Downarrow \\
\boldsymbol{R}_{ij} &= \boldsymbol{R}_j\boldsymbol{R}_i^{-1} \\
\boldsymbol{T}_{ij} &= \boldsymbol{T}_j - \boldsymbol{T}_i
\end{aligned}
$$

Caveat on Conventions

- Two representations for Euclidean transformation
- Equivalent but result in differences in representation
- Also verify the direction of relation imputed $i \leftarrow j$ or $i \rightarrow j$
- Exercise care!

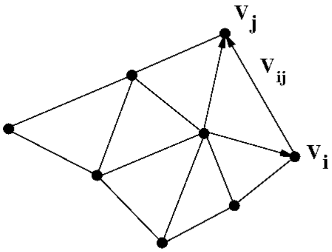
Theory and Formulation



Observation

- Viewgraph representation $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$
  - Vertices represent cameras
  - Edges represent camera-camera relative motions
- N image sequence described by $N - 1$ motions
- Typically one camera is the origin
- Sequence can provide *as many as* ${}^{N}C_2 = \frac{N(N-1)}{2}$ relative motions
- Relative motions form highly redundant system of equations

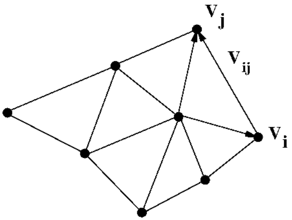
Theory and Formulation



Observation

- Assume we know values $\boldsymbol{v}_i$ (vertices)
- Difference $\boldsymbol{v}_{ij}$ (edges) easy to estimate given vertices
- $\boldsymbol{v}_{ij} = \boldsymbol{v}_j - \boldsymbol{v}_i$
- Motion Averaging is the converse problem
- Solve for vertices $\{\boldsymbol{v}_i \in \mathcal{V}\}$ given edges $\{\boldsymbol{v}_{ij} \in \mathcal{E}\}$

Theory and Formulation

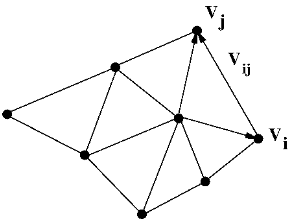


- Relative displacements observed
- Loop : $\boldsymbol{v}_j - \boldsymbol{v}_{ij} - \boldsymbol{v}_i = 0$
- $\boldsymbol{v}_j - \boldsymbol{v}_i = \boldsymbol{v}_{ij}$

Transformation Constraints

- "Loops" of transformations result in no change
- Effective transformation is equal to **zero**
- Constraint on composition of transformations
- Strong constraint on admissible solutions
- Solution is upto unknown shift (gauge freedom)
- Typically fix one vertex to origin
- How many edges do we need ?

Theory and Formulation

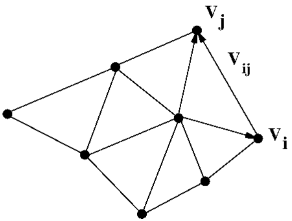


- Relative displacements observed
- Loop : $\boldsymbol{v}_j - \boldsymbol{v}_{ij} - \boldsymbol{v}_i = 0$
- $\boldsymbol{v}_j - \boldsymbol{v}_i = \boldsymbol{v}_{ij}$

Transformation Constraints

- What happens when edges are noisy ?
- $\boldsymbol{v}_j - \boldsymbol{v}_i \neq \boldsymbol{v}_{ij}$
- Observations are no longer ‘consistent’
- Different paths yield different answers
- Solution: Find estimate most consistent with edges

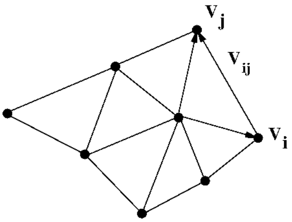
Theory and Formulation



- Relative displacements observed
- Loop : $\boldsymbol{v}_j - \boldsymbol{v}_{ij} - \boldsymbol{v}_i = 0$
- $\boldsymbol{v}_j - \boldsymbol{v}_i = \boldsymbol{v}_{ij}$

Averaging of Relative Motions

- Define degree of inconsistency $d(\boldsymbol{v}_{ij}, \boldsymbol{v}_j - \boldsymbol{v}_i) = ||\boldsymbol{v}_{ij} - (\boldsymbol{v}_j - \boldsymbol{v}_i)||$
- Minimise cost function : $\sum_{\mathcal{E}} d^2(\boldsymbol{v}_{ij}, \boldsymbol{v}_j - \boldsymbol{v}_i)$
- Solve minimisation of $\sum_{\mathcal{E}} ||\boldsymbol{v}_{ij} - (\boldsymbol{v}_j - \boldsymbol{v}_i)||^2$
- Linear estimation problem



$$\boldsymbol{v}_j - \boldsymbol{v}_i = \boldsymbol{v}_{ij}$$

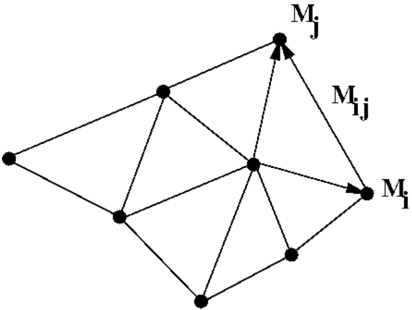
$$\overbrace{\begin{bmatrix} \cdots & -1 & \cdots & 1 & \cdots \end{bmatrix}}^{\boldsymbol{A}_{ij}} \underbrace{\begin{bmatrix} \vdots \\ \boldsymbol{v}_i \\ \vdots \\ \boldsymbol{v}_j \\ \vdots \end{bmatrix}}_{\boldsymbol{V}_{\mathcal{V}}} = \overbrace{\begin{bmatrix} \vdots \\ \boldsymbol{v}_{ij} \\ \vdots \end{bmatrix}}^{\boldsymbol{V}_{\mathcal{E}}}$$

Linear System of Equations : $\boldsymbol{A}\boldsymbol{V} = \boldsymbol{V}_{ij}$

$\boldsymbol{A}$ encodes the viewgraph

Averages the information on edges

Theory and Formulation



Averaging of Relative Motions

For our purposes, consider motion matrices $\boldsymbol{M}$

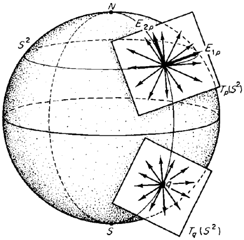
- $\boldsymbol{M}_{ij} = \boldsymbol{M}_j \boldsymbol{M}_i^{-1}, \forall i \neq j$
- Analogous to $\boldsymbol{v}_j - \boldsymbol{v}_i = \boldsymbol{v}_{ij}$
- LHS : Observations
- RHS : Global Motion to be fitted

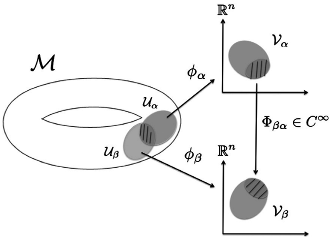
Theory and Formulation

$$\overbrace{\begin{bmatrix} \cdots & \boldsymbol{M}_{ij} & \cdots & -\boldsymbol{I} & \cdots \end{bmatrix}}^{\boldsymbol{A}_{ij}} \underbrace{\begin{bmatrix} \vdots \\ \boldsymbol{M}_i \\ \vdots \\ \boldsymbol{M}_j \\ \vdots \end{bmatrix}}_{\boldsymbol{M}_g} = \boldsymbol{0}$$

Averaging of Relative Motions

- $\boldsymbol{M}_{ij} = \boldsymbol{M}_j \boldsymbol{M}_i^{-1}, \forall i \neq j \Rightarrow \boldsymbol{M}_{ij}\boldsymbol{M}_i - \boldsymbol{M}_j = \boldsymbol{0}$
- Global Motion : $\boldsymbol{M}_g = \{\boldsymbol{M}_1, \cdots, \boldsymbol{M}_N\}$
- System of Equations $\boldsymbol{A}\boldsymbol{M}_g = \boldsymbol{0}$
- Linear Solution ?
- Not valid for **non-linear** motion groups
- Motion Groups are Lie Groups



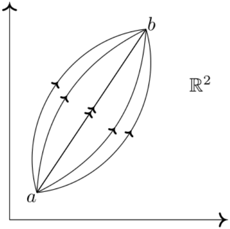


Riemannian Manifold

- Non-linear "surface" $\mathcal{M}$ embedded in $\mathbb{R}^n$
- Unlike vector spaces, no global basis to describe
- Local description in tangent space (vector space)
- $T_{\boldsymbol{p}}\mathcal{M}$ at point $\boldsymbol{p} \in \mathcal{M}$
- Vector (tangent) space equipped with inner product $g_{\boldsymbol{p}}$
- **Key idea:** Smooth, differentiable manifold

Figures from doCarmo and Y. M. Liu "Advances in matrix manifolds for computer vision"

Theory and Formulation



Riemannian Manifold

- **Key idea:** Smooth, differentiable manifold
- $T_{\boldsymbol{p}}\mathcal{M}$ at point $\boldsymbol{p} \in \mathcal{M}$
- Vector space $T_{\boldsymbol{p}}\mathcal{M}$ equipped with inner product $g_{\boldsymbol{p}}$
- Length of curve: $\int \sqrt{< dl, dl >_g} ds$
- Length of shortest path between points (geodesic/distance)
- Notion of length or distance important for us
- General geodesic computations are complicated (curvature)
- Significantly simpler for Lie Groups

Theory and Formulation

$$
\begin{aligned}
\boldsymbol{X} \circ \boldsymbol{Y} &\in \mathbb{G}\ (closure)\\
\boldsymbol{X} \circ (\boldsymbol{Y} \circ \boldsymbol{Z}) &= (\boldsymbol{X} \circ \boldsymbol{Y}) \circ \boldsymbol{Z}\ (associativity)\\
\exists\, \boldsymbol{E} \in \mathbb{G} \ni \boldsymbol{X} \circ \boldsymbol{E} &= \boldsymbol{E} \circ \boldsymbol{X} = \boldsymbol{X}\ (identity)\\
\exists\, \boldsymbol{X}^{-1} \in \mathbb{G} \ni \boldsymbol{X} \circ \boldsymbol{X}^{-1} &= \boldsymbol{X}^{-1} \circ \boldsymbol{X} = \boldsymbol{E}\ (inverse)
\end{aligned}
$$

What is a Group ?

- Group is set $\mathbb{G}$ equipped with operation $\circ$
- Satisfies specific properties
- Examples :
  - $\{\mathbb{R}, +\}$
  - $\{(0, 1, \cdots, n-1), mod(n)\}$
  - $\{\mathbf{M} \in \mathbb{R}^{n \times n} | \det(\boldsymbol{M}) > 0, \times\}$
- Groups can be open or closed
- Enormous literature on representation and classification

Riemannian Manifold + Group Structure = Lie Group

Key Idea

- Smooth, differentiable structure of Riemannian manifold
- Group properties $\implies$ lots of structure in representation
- Combination induces special properties
- Motion representations in 3D vision are Lie groups

Matrix Groups

- Finite dimensional Lie groups = Matrix Groups
- Consider $n \times n$ matrices $\boldsymbol{X}$
- Group representation $\mathbb{GL}(n, \mathbb{R})$ or $\mathbb{GL}(n)$

Theory and Formulation

$$\begin{aligned}
\boldsymbol{XY} &\in \mathbb{GL}(n)\ (\textit{closure}) \\
\boldsymbol{X}(\boldsymbol{YZ}) &= (\boldsymbol{XY})\boldsymbol{Z}\ (\textit{associativity}) \\
\exists\, \boldsymbol{I} \in \mathbb{GL}(n) \ni \boldsymbol{XI} &= \boldsymbol{IX} = \boldsymbol{X}\ (\textit{identity})
\end{aligned}$$

General Linear Group $\mathbb{GL}(n)$

- n dimensional matrices $\boldsymbol{X}$
- Group under matrix multiplication
- $\boldsymbol{X} \in \mathbb{R}^{n^2}$
- All points in $\mathbb{R}^{n^2}$ in $\mathbb{GL}(n)$?

Theory and Formulation

$$
\begin{aligned}
\boldsymbol{X}\boldsymbol{Y} &\in \mathbb{GL}(n)\ (\textit{closure}) \\
\boldsymbol{X}(\boldsymbol{Y}\boldsymbol{Z}) &= (\boldsymbol{X}\boldsymbol{Y})\boldsymbol{Z}\ (\textit{associativity}) \\
\exists\, \boldsymbol{I} \in \mathbb{GL}(n) \ni \boldsymbol{X}\boldsymbol{I} &= \boldsymbol{I}\boldsymbol{X} = \boldsymbol{X}\ (\textit{identity}) \\
\exists\, \boldsymbol{X}^{-1} \in \mathbb{GL}(n) \ni \boldsymbol{X}\boldsymbol{X}^{-1} &= \boldsymbol{X}^{-1}\boldsymbol{X} = \boldsymbol{I}\ (\textit{inverse})
\end{aligned}
$$

General Linear Group $\mathbb{GL}(n)$

- n dimensional matrices $\boldsymbol{X}$
- Group under matrix multiplication
- $\boldsymbol{X} \in \mathbb{R}^{n^2}$
- All points in $\mathbb{R}^{n^2}$ in $\mathbb{GL}(n)$?
- Require $|\boldsymbol{X}| \neq 0$ for inverse

Theory and Formulation

$$\begin{aligned}
\boldsymbol{X}\boldsymbol{Y} &\in \mathbb{GL}(n) \;(\textit{closure}) \\
\boldsymbol{X}(\boldsymbol{Y}\boldsymbol{Z}) &= (\boldsymbol{X}\boldsymbol{Y})\boldsymbol{Z} \;(\textit{associativity}) \\
\exists\, \boldsymbol{I} \in \mathbb{GL}(n) \ni \boldsymbol{X}\boldsymbol{I} &= \boldsymbol{I}\boldsymbol{X} = \boldsymbol{X} \;(\textit{identity}) \\
\exists\, \boldsymbol{X}^{-1} \in \mathbb{GL}(n) \ni \boldsymbol{X}\boldsymbol{X}^{-1} &= \boldsymbol{X}^{-1}\boldsymbol{X} = \boldsymbol{I} \;(\textit{inverse})
\end{aligned}$$

General Linear Group $\mathbb{GL}(n)$

- n dimensional matrices $\boldsymbol{X}$
- Group under matrix multiplication
- $\boldsymbol{X} \in \mathbb{R}^{n^2}$
- All points in $\mathbb{R}^{n^2}$ in $\mathbb{GL}(n)$?
- Require $|\boldsymbol{X}| \neq 0$ for inverse
- Singular $\boldsymbol{X}$ = affine algebraic variety
- $\mathbb{GL}(n)$: open set in $\mathbb{R}^{n^2}$ of dim n^2

Special Linear Group $\mathbb{SL}(n)$

- $\mathbb{GL}(n) = \{\boldsymbol{X} | \boldsymbol{X} \in \mathbb{R}^{n^2}, |\boldsymbol{X}| \neq 0\}$
- Specialise to matrices with determinant of 1
- Also forms a (sub)group since $|\boldsymbol{AB}| = |\boldsymbol{A}||\boldsymbol{B}| = 1$
- Special Linear Group $\mathbb{SL}(n)$
- $\mathbb{SL}(n)$: non-singular algebraic variety in $\mathbb{R}^{n^2}$ of dim $n^2 - 1$

Classical Groups

- Several classical matrix groups
- Symmetry groups for specific metric spaces
- Group acting on space leaves property invariant

Theory and Formulation

$$\boldsymbol{a}.\boldsymbol{b} \quad = \quad \boldsymbol{a}^T\boldsymbol{b}$$

Classical Groups

- Several classical matrix groups
- Symmetry groups for specific metric spaces
- Group acting on space leaves property invariant
- Orthogonal Group $\mathbb{O}(n)$
- Leaves dot product invariant (equivalently distance)

Theory and Formulation

$$\begin{aligned} \boldsymbol{a}.\boldsymbol{b} &= \boldsymbol{a}^T\boldsymbol{b} \\ \boldsymbol{a}^T\boldsymbol{M}^T\boldsymbol{M}\boldsymbol{b} &= \boldsymbol{a}^T\boldsymbol{b} \end{aligned}$$

Classical Groups

- Several classical matrix groups
- Symmetry groups for specific metric spaces
- Group acting on space leaves property invariant
- Orthogonal Group $\mathbb{O}(n)$
- Leaves dot product invariant (equivalently distance)
- Apply matrix transformation $\boldsymbol{M} \in \mathbb{O}(n)$

Theory and Formulation

$$
\begin{aligned}
\boldsymbol{a}.\boldsymbol{b} &= \boldsymbol{a}^T\boldsymbol{b} \\
\boldsymbol{a}^T\boldsymbol{M}^T\boldsymbol{M}\boldsymbol{b} &= \boldsymbol{a}^T\boldsymbol{b} \\
\text{Invariance } \Rightarrow \boldsymbol{M}^T\boldsymbol{M} &= \boldsymbol{I}_n
\end{aligned}
$$

Classical Groups

- Several classical matrix groups
- Symmetry groups for specific metric spaces
- Group acting on space leaves property invariant
- Orthogonal Group $\mathbb{O}(n)$
- Leaves dot product invariant (equivalently distance)
- Apply matrix transformation $\boldsymbol{M} \in \mathbb{O}(n)$

Orthogonal Group $\mathbb{O}(n)$

- $n \times n$ matrices s.t. $\boldsymbol{M}^T\boldsymbol{M} = \boldsymbol{I}_n$
- $\boldsymbol{M}^T$ is inverse by definition
- $\mathbb{O}(n) = \{\boldsymbol{M} \in \mathbb{GL}(n) : \boldsymbol{M}^T\boldsymbol{M} = \boldsymbol{I}_n\}$
- Since $|\boldsymbol{M}| = |\boldsymbol{M}^T|$
- $|\boldsymbol{M}^T||\boldsymbol{M}| = |\boldsymbol{M}^T\boldsymbol{M}| = |\boldsymbol{I}_n| = 1$
- Implies for $\mathbb{O}(n), |\boldsymbol{M}| = \pm 1$

Special Orthogonal Group $\mathbb{SO}(n)$

- $\mathbb{O}(n)^+ = \{\boldsymbol{M} \in \mathbb{O}(n) : |\boldsymbol{M}| = 1\}$
- $\mathbb{O}(n)^- = \{\boldsymbol{M} \in \mathbb{O}(n) : |\boldsymbol{M}| = -1\}$
- $\mathbb{O}(n)^+$ and $\mathbb{O}(n)^-$ are topologically disconnected
- Select $\mathbb{O}(n)^+$ as Special Orthogonal Group of dim n
- Denoted $\mathbb{SO}(n)$
- 3D Rotation Group $\mathbb{SO}(3)$ is of special interest for us
- Will not consider topological properties

Group Multiplication

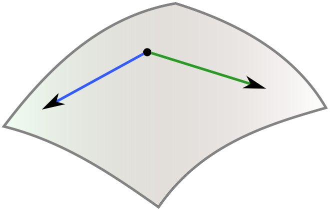
$$
\begin{aligned}
a : \mathbb{G} \times \mathbb{G} &\longrightarrow \mathbb{G} \\
\boldsymbol{R}_1 \in \mathbb{SO}(3),\ \boldsymbol{R}_2 &\in \mathbb{SO}(3) \\
\Rightarrow \boldsymbol{R}_1 \boldsymbol{R}_2 &\in \mathbb{SO}(3)
\end{aligned}
$$

Action on Vector Space

$$
\begin{aligned}
a : \mathbb{G} \times \mathcal{X} &\longrightarrow \mathcal{X} \\
\boldsymbol{R} \in \mathbb{SO}(3), \boldsymbol{v} &\in \mathbb{R}^3 \\
\Rightarrow \boldsymbol{R}\boldsymbol{v} &\in \mathbb{R}^3
\end{aligned}
$$

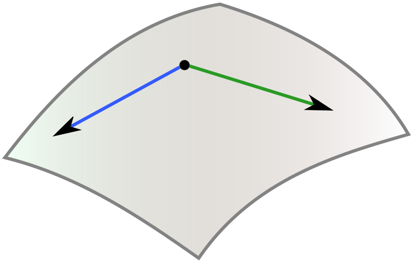
Group Action

- Two types of group action possible



Lie Algebra $\equiv$ Tangent Space

- Several ways of describing Lie Algebra
- Consider smooth mapping $\gamma : \mathbb{R} \longrightarrow \mathbb{G}$
- Require $\gamma(0) = \boldsymbol{I}$ (identity)
- Equivalence class : First derivatives at $\mathbf{0}$ are same
- Equivalence class is a **vector space**
- Denote Lie Algebra as $\mathfrak{g}$
- Can define basis in Lie Algebra



Tangent Space of $\mathbb{SO}(n)$

- Consider orthonormality constraint $\boldsymbol{M}^T\boldsymbol{M} = \boldsymbol{I}_n$
- Around $\boldsymbol{I}_n$, move locally $\boldsymbol{I} + \epsilon\boldsymbol{\Delta}$
- For $\epsilon \to 0$, $(\boldsymbol{I}_n + \epsilon\boldsymbol{\Delta})^T(\boldsymbol{I}_n + \epsilon\boldsymbol{\Delta}) = \boldsymbol{I}_n$
- Implies $\boldsymbol{\Delta} + \boldsymbol{\Delta}^T = \boldsymbol{0}$
- Tangent space is skew-symmetric in form
- $\boldsymbol{\Delta} \in \mathfrak{so}(n)$

Theory and Formulation

Conjugation and Adjoint

- For $\boldsymbol{G}, \boldsymbol{X} \in \mathbb{G}$, mapping $\boldsymbol{G}\boldsymbol{X}\boldsymbol{G}^{-1}$ is a conjugation
- Differentiation maps tangent space at $\boldsymbol{I}_n$ to itself
- Linearity gives $Ad(\boldsymbol{G})\boldsymbol{x} = \boldsymbol{G}\boldsymbol{x}\boldsymbol{G}^{-1}\ \forall \boldsymbol{G} \in \mathbb{G}$

Commutators in Lie Algebra

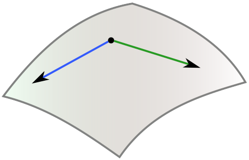
- Consider neighbourhood of identity
- Consider Lie Algebra elements $\boldsymbol{x}, \boldsymbol{y} \in \mathfrak{g}$
- $\boldsymbol{G} \approx \boldsymbol{I} + \epsilon\boldsymbol{x} + hot$
- Consider $\boldsymbol{G}(\boldsymbol{x})\boldsymbol{y}\boldsymbol{G}(\boldsymbol{x})^{-1}$
- $\boldsymbol{G}(\boldsymbol{x})\boldsymbol{y}\boldsymbol{G}(\boldsymbol{x})^{-1} = (\boldsymbol{I} + \epsilon\boldsymbol{x})\boldsymbol{y}(\boldsymbol{I} + \epsilon\boldsymbol{x})^{-1} = \boldsymbol{y} + \epsilon(\boldsymbol{x}\boldsymbol{y} - \boldsymbol{y}\boldsymbol{x}) + hot$
- **Lie Bracket**: $[\boldsymbol{x}, \boldsymbol{y}] = (\boldsymbol{x}\boldsymbol{y} - \boldsymbol{y}\boldsymbol{x}) \in \mathfrak{g}$
- Lie Algebra is vector space equipped with bracket
- Bracket measures degree of non-commutativity

Theory and Formulation

Properties of Lie Algebra

- Tangent Space at an element in $\mathbb{G}$
- Forms a Vector Space (denoted $\mathfrak{g}$)
- Anti-commutative Lie Bracket: $[\boldsymbol{x}, \boldsymbol{y}] = (\boldsymbol{x}\boldsymbol{y} - \boldsymbol{y}\boldsymbol{x}) \in \mathfrak{g}$
- Commutator is not associative
- Jacobi identity: $[\boldsymbol{x}, [\boldsymbol{y}, \boldsymbol{z}]] + [\boldsymbol{z}, [\boldsymbol{x}, \boldsymbol{y}]] + [\boldsymbol{y}, [\boldsymbol{z}, \boldsymbol{x}]] = 0$

Theory and Formulation



Lie Groups

- Special properties
  - $\boldsymbol{X} \times \boldsymbol{Y} \mapsto \boldsymbol{XY}$ is a smooth, differentiable mapping
  - $\boldsymbol{X} \mapsto \boldsymbol{X}^{-1}$ is a smooth, differentiable mapping
- Lie groups are locally topologically equivalent to vector space
- Local neighbourhood adequately described by tangent space
- Vector space forms a Lie algebra $\mathfrak{g}$
- Vector space equipped with Lie bracket $[.,.] : \mathfrak{g} \times \mathfrak{g} \to \mathfrak{g}$
- Properties of bracket

$$[\boldsymbol{x}, \boldsymbol{y}] = -[\boldsymbol{y}, \boldsymbol{x}]$$

$$[\boldsymbol{x}, [\boldsymbol{y}, \boldsymbol{z}]] + [\boldsymbol{y}, [\boldsymbol{z}, \boldsymbol{x}]] + [\boldsymbol{z}, [\boldsymbol{x}, \boldsymbol{y}]] = 0$$

Exponential Mapping

- To translate tangent vectors to element in $\mathbb{G}$
- Done using left-invariant mapping (linear operation)
- For $\boldsymbol{x}$ in Lie Algebra, $\boldsymbol{G}\boldsymbol{x}$ is tangent to $\boldsymbol{G} \in \mathbb{G}$

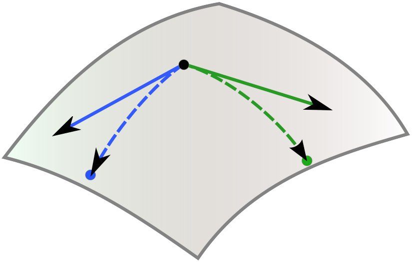
$$\frac{d\boldsymbol{G}}{ds} = \boldsymbol{G}\boldsymbol{x} \Rightarrow \boldsymbol{G}(s) = e^{s\boldsymbol{x}}\boldsymbol{G}(0)$$

Exponential Mapping

- To translate tangent vectors to element in $\mathbb{G}$
- Done using left-invariant mapping (linear operation)
- For $\boldsymbol{x}$ in Lie Algebra, $\boldsymbol{G}\boldsymbol{x}$ is tangent to $\boldsymbol{G} \in \mathbb{G}$
- Yields fundamental differential relationship
- Paths have corresponding $\boldsymbol{G}\boldsymbol{x}$ tangential at $\boldsymbol{G}$ (integral curves)
- **Exponential mapping** is of fundamental importance

Theory and Formulation

$$\frac{d\boldsymbol{G}}{ds} = \boldsymbol{G}\boldsymbol{x} \Rightarrow \boldsymbol{G}(s) = e^{s\boldsymbol{x}}\boldsymbol{G}(0)$$



Exponential Mapping

- To translate tangent vectors to element in $\mathbb{G}$
- Done using left-invariant mapping (linear operation)
- For $\boldsymbol{x}$ in Lie Algebra, $\boldsymbol{G}\boldsymbol{x}$ is tangent to $\boldsymbol{G} \in \mathbb{G}$
- Yields fundamental differential relationship
- Paths have corresponding $\boldsymbol{G}\boldsymbol{x}$ tangential at $\boldsymbol{G}$ (integral curves)
- **Exponential mapping** is of fundamental importance

Exponential Mapping

- **Exponential** is of fundamental importance for Lie Groups

$$e^{\boldsymbol{x}} = \boldsymbol{I} + \boldsymbol{x} + \frac{\boldsymbol{x}^2}{2} + \cdots + \frac{\boldsymbol{x}^n}{n!} + \cdots$$

Exponential Mapping

- **Exponential** is of fundamental importance for Lie Groups
- Power series for exponential
- Convergence of series ?

Logarithm Mapping

- In neighbourhood around $\boldsymbol{I}$, exp map is homeomorphism
- Around $\mathbf{0} \in \mathfrak{g}$
- In neighbourhood, inverse (logarithm) is defined
- **Logarithm mapping** : Lie Group to corresponding Algebra
- $exp()$ and $log()$ allow us to move between Lie Group and Lie Algebra

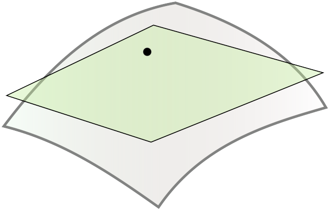
Theory and Formulation

$$\boldsymbol{x} = log(\boldsymbol{G}) = \sum_{k=1}^{\infty} \frac{(-1)^{k-1}(\boldsymbol{G}-\boldsymbol{I})^k}{k}$$
$$= (\boldsymbol{G}-\boldsymbol{I}) - \frac{1}{2}(\boldsymbol{G}-\boldsymbol{I})^2 + \frac{1}{3}(\boldsymbol{G}-\boldsymbol{I})^3 + \cdots$$

Logarithm Mapping

- In neighbourhood around $\boldsymbol{I}$, exp map is homeomorphism
- Around $\mathbf{0} \in \mathfrak{g}$
- In neighbourhood, inverse (logarithm) is defined
- **Logarithm mapping** : Lie Group to corresponding Algebra
- $exp()$ and $log()$ allow us to move between Lie Group and Lie Algebra

Theory and Formulation



Distances on Lie Groups

- Non-commutative : $e^{\boldsymbol{x}} e^{\boldsymbol{y}} \neq e^{(\boldsymbol{x}+\boldsymbol{y})}$
- However $e^{s_1 \boldsymbol{x}} e^{s_2 \boldsymbol{x}} = e^{(s_1+s_2)\boldsymbol{x}}$
- One-dimensional subgroup (commutative)
- Starting at $\boldsymbol{G} = \boldsymbol{I}$ consider path $\boldsymbol{G}(s) = e^{s\boldsymbol{x}}$
- Moving along direction $\boldsymbol{x}$ in Lie Algebra traces path on $\boldsymbol{G}$
- Trace out path for each direction $\boldsymbol{x}$
- Defines a "natural" distance metric :
  $d(\boldsymbol{X}(s), \boldsymbol{I}) = |s| = || \log(\boldsymbol{X}(s)) ||$

Distances on Lie Groups

- Measure distance $d(\boldsymbol{X}_1, \boldsymbol{X}_2)$?
- Left-translate both by $\boldsymbol{X}_1^{-1}$ (say)
- Now our elements are $\boldsymbol{I}$ and $\boldsymbol{X}_1^{-1}\boldsymbol{X}_2$ resp.
- Left-invariant distance measure
- $d(\boldsymbol{X}_1, \boldsymbol{X}_2) = d(\boldsymbol{I}, \boldsymbol{X}_1^{-1}\boldsymbol{X}_2)$
- Will use this extensively

Distance Metrics on Lie Groups

- Left-invariant: $d(\boldsymbol{X}_1, \boldsymbol{X}_2) = d(\boldsymbol{X}\boldsymbol{X}_1, \boldsymbol{X}\boldsymbol{X}_2)\ \forall \boldsymbol{X} \in \mathbb{G}$
- Right-invariant: $d(\boldsymbol{X}_1, \boldsymbol{X}_2) = d(\boldsymbol{X}_1\boldsymbol{X}, \boldsymbol{X}_2\boldsymbol{X})\ \forall \boldsymbol{X} \in \mathbb{G}$
- Bi-invariant: Both left- and right-invariant metric
- Intrinsic metric on $\mathbb{SO}(3)$ is bi-invariant
- No bi-invariant metric on $\mathbb{SE}(3)$

Exponential Representation

- Lie groups are non-commutative: $e^{\boldsymbol{x}}e^{\boldsymbol{y}} \neq e^{\boldsymbol{x}+\boldsymbol{y}}$
- Neighbourhood of $\boldsymbol{I}$: $\boldsymbol{X} = e^{\boldsymbol{x}} \approx \boldsymbol{I} + \boldsymbol{x}$
- Product commutes $\boldsymbol{X}\boldsymbol{Y} \approx \boldsymbol{I} + \boldsymbol{x} + \boldsymbol{y}$

Theory and Formulation

$$
\begin{aligned}
e^{\boldsymbol{x}} e^{\boldsymbol{y}} &\neq e^{(\boldsymbol{x}+\boldsymbol{y})} \\
e^{\boldsymbol{x}} e^{\boldsymbol{y}} &= e^{BCH(\boldsymbol{x},\boldsymbol{y})}
\end{aligned}
$$

Exponential Representation

- Lie groups are non-commutative: $e^{\boldsymbol{x}} e^{\boldsymbol{y}} \neq e^{\boldsymbol{x}+\boldsymbol{y}}$
- Neighbourhood of $\boldsymbol{I}$: $\boldsymbol{X} = e^{\boldsymbol{x}} \approx \boldsymbol{I} + \boldsymbol{x}$
- Product commutes $\boldsymbol{X}\boldsymbol{Y} \approx \boldsymbol{I} + \boldsymbol{x} + \boldsymbol{y}$
- Equivalent mapping is Baker-Campbell-Hausdorff (BCH) formula

Theory and Formulation

$$
\begin{aligned}
e^{\boldsymbol{x}} e^{\boldsymbol{y}} &\neq e^{(\boldsymbol{x}+\boldsymbol{y})} \\
e^{\boldsymbol{x}} e^{\boldsymbol{y}} &= e^{BCH(\boldsymbol{x},\boldsymbol{y})} \\
BCH(\boldsymbol{x},\boldsymbol{y}) &= \boldsymbol{x}+\boldsymbol{y}+\frac{1}{2}[\boldsymbol{x},\boldsymbol{y}]+\frac{1}{12}[\boldsymbol{x}-\boldsymbol{y},[\boldsymbol{x},\boldsymbol{y}]]+\cdots
\end{aligned}
$$

Exponential Representation

- Lie groups are non-commutative: $e^{\boldsymbol{x}} e^{\boldsymbol{y}} \neq e^{\boldsymbol{x}+\boldsymbol{y}}$
- Neighbourhood of $\boldsymbol{I}$: $\boldsymbol{X} = e^{\boldsymbol{x}} \approx \boldsymbol{I} + \boldsymbol{x}$
- Product commutes $\boldsymbol{X}\boldsymbol{Y} \approx \boldsymbol{I} + \boldsymbol{x} + \boldsymbol{y}$
- Equivalent mapping is Baker-Campbell-Hausdorff (BCH) formula
- Series form available

Theory and Formulation

$$
\begin{aligned}
e^{\boldsymbol{x}} e^{\boldsymbol{y}} &\neq e^{(\boldsymbol{x}+\boldsymbol{y})} \\
e^{\boldsymbol{x}} e^{\boldsymbol{y}} &= e^{BCH(\boldsymbol{x},\boldsymbol{y})} \\
BCH(\boldsymbol{x},\boldsymbol{y}) &= \boldsymbol{x}+\boldsymbol{y}+\frac{1}{2}[\boldsymbol{x},\boldsymbol{y}]+\frac{1}{12}[\boldsymbol{x}-\boldsymbol{y},[\boldsymbol{x},\boldsymbol{y}]]+\cdots
\end{aligned}
$$

Exponential Representation

- Lie groups are non-commutative: $e^{\boldsymbol{x}} e^{\boldsymbol{y}} \neq e^{\boldsymbol{x}+\boldsymbol{y}}$
- Neighbourhood of $\boldsymbol{I}$: $\boldsymbol{X} = e^{\boldsymbol{x}} \approx \boldsymbol{I} + \boldsymbol{x}$
- Product commutes $\boldsymbol{X}\boldsymbol{Y} \approx \boldsymbol{I} + \boldsymbol{x} + \boldsymbol{y}$
- Equivalent mapping is Baker-Campbell-Hausdorff (BCH) formula
- Series form available
- All higher terms given in terms of Lie bracket
- Closed forms exist for $\mathbb{SO}(3)$ and $\mathbb{SE}(3)$

Euclidean Motion :

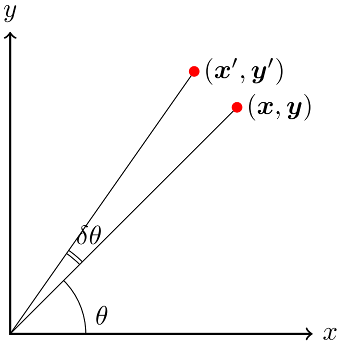
$$\boldsymbol{Q} = \boldsymbol{R}\boldsymbol{P} + \boldsymbol{T}$$

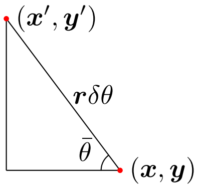
Homogeneous form :

$$\left[\begin{array}{c} \boldsymbol{Q} \\ \hline 1 \end{array}\right] = \left[\begin{array}{c|c} \boldsymbol{R} & \boldsymbol{T} \\ \hline \mathbf{0} & 1 \end{array}\right] \left[\begin{array}{c} \boldsymbol{P} \\ \hline 1 \end{array}\right]$$

Motion Groups

- We will consider two motion groups
- 3D Rotations : $\boldsymbol{R}\boldsymbol{R}^T = \boldsymbol{R}^T\boldsymbol{R} = \boldsymbol{I}$
- 3D Rotations : Special Orthogonal Group $\boldsymbol{R} \in \mathbb{SO}(3)$
- Euclidean Motions : $\boldsymbol{M} \in \mathbb{SE}(3)$
- $\boldsymbol{R}$ and $\boldsymbol{M}$ have 3 and 6 degrees of freedom respectively



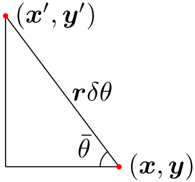


$$\begin{aligned} \boldsymbol{x}' &= \boldsymbol{x} - \boldsymbol{r}\delta\theta \cos(\frac{\pi}{2} - \theta) \\ \boldsymbol{y}' &= \boldsymbol{y} + \boldsymbol{r}\delta\theta \sin(\frac{\pi}{2} - \theta) \end{aligned}$$

Two-dimensional Rotations

- Consider a point at $\boldsymbol{p} = (\boldsymbol{x}, \boldsymbol{y})$
- Let $\boldsymbol{r} = \sqrt{\boldsymbol{x}^2 + \boldsymbol{y}^2}$ and $\theta = \tan^{-1}(\frac{y}{x})$
- Now lets rotate this point by infinitesimal $\delta\theta$ counterclockwise
- Let the new point location be $\boldsymbol{p}' = (\boldsymbol{x}', \boldsymbol{y}')$

Theory and Formulation



$$
\begin{aligned}
x' &= x - r\delta\theta\cos(\frac{\pi}{2} - \theta) = x - r\delta\theta\sin(\theta) \\
y' &= y + r\delta\theta\sin(\frac{\pi}{2} - \theta) = y + r\delta\theta\cos(\theta)
\end{aligned}
$$

Two-dimensional Rotations

$$
\begin{aligned}
\begin{bmatrix} x' \\ y' \end{bmatrix} &= \begin{bmatrix} x \\ y \end{bmatrix} + \delta\theta \begin{bmatrix} -y \\ x \end{bmatrix} \\
\Rightarrow \begin{bmatrix} x' \\ y' \end{bmatrix} &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \delta\theta \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \\
\Rightarrow \begin{bmatrix} x' \\ y' \end{bmatrix} &= (I + \delta\theta B) \begin{bmatrix} x \\ y \end{bmatrix}
\end{aligned}
$$

$$\boldsymbol{p}' = (\boldsymbol{I} + \delta\theta \boldsymbol{B})\,\boldsymbol{p}$$

Two-dimensional Rotations

- Note that $\boldsymbol{R} = \boldsymbol{I} + \delta\theta\boldsymbol{B} \in \mathbb{SO}(2)$
- Also note that $\boldsymbol{B}^2 = -\boldsymbol{I}$
- Now consider repeated application of infinitesimal $\boldsymbol{R}$

Theory and Formulation

$$\boldsymbol{p}' = (\boldsymbol{I} + \delta\theta\boldsymbol{B})\,\boldsymbol{p}$$

Two-dimensional Rotations

- Let us divide the total rotation θ into n rotations
- $\delta\theta = \frac{\theta}{n}$
- Now consider repeated application of infinitesimal $\delta\boldsymbol{R}$
- Total rotation is a binomial expansion:

$$\boldsymbol{R} = \prod_{i=1}^{n} \delta\boldsymbol{R} = \left(\boldsymbol{I} + \frac{\theta}{n}\boldsymbol{B}\right)^n$$

- Recall that $\boldsymbol{B}^2 = -\boldsymbol{I}$
- Hence $\boldsymbol{R}$ is sum of two series
- Even terms (multiples of $\boldsymbol{I}$), odd terms (multiples of $\boldsymbol{B}$)

Theory and Formulation

$$
\begin{aligned}
\boldsymbol{R} &= \lim_{n\to\infty}\left(\boldsymbol{I}+\frac{\theta}{n}\boldsymbol{B}\right)^n = \exp\left(\theta\boldsymbol{B}\right) \\
&= \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}
\end{aligned}
$$

Two-dimensional Rotations

- Let us divide the total rotation θ into n rotations $\delta\theta = \frac{\theta}{n}$
- Now consider repeated application of infinitesimal $\delta\boldsymbol{R}$
- Total rotation is a binomial expansion:

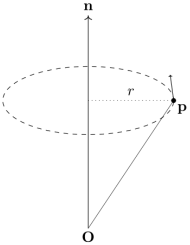
$$
\boldsymbol{R} = \prod_{i=1}^{n} \delta\boldsymbol{R} = \left(\boldsymbol{I}+\frac{\theta}{n}\boldsymbol{B}\right)^n
$$

- Recall that $\boldsymbol{B}^2 = -\boldsymbol{I}$
- Hence $\boldsymbol{R}$ is sum of two series
- Even terms (multiples of $\boldsymbol{I}$), odd terms (multiples of $\boldsymbol{B}$)

$$\begin{aligned}\boldsymbol{R} &= \lim_{n\to\infty}\left(\boldsymbol{I}+\frac{\theta}{n}\boldsymbol{B}\right)^n = \exp\left(\theta\boldsymbol{B}\right)\\ &= \begin{bmatrix}\cos\theta & -\sin\theta\\ \sin\theta & \cos\theta\end{bmatrix}\end{aligned}$$

Two-dimensional Rotations

- Lie Group : $\boldsymbol{R} \in \mathbb{SO}(2)$
- $\boldsymbol{B}$ is one-dimensional Lie Algebra $\mathfrak{so}(2)$
- Lie Group and Lie Algebra related by exp(.) and log(.) maps



$$\boldsymbol{p}^{'} = \boldsymbol{p} + r\delta\theta \frac{\boldsymbol{n} \times \boldsymbol{p}}{||\boldsymbol{n} \times \boldsymbol{p}||}$$

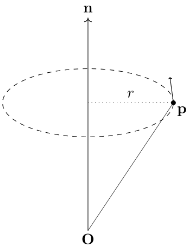
$$r = ||\boldsymbol{p} - (\boldsymbol{n}^T\boldsymbol{p})\boldsymbol{n}|| = ||\boldsymbol{n} \times \boldsymbol{p}||$$

$$\Rightarrow \boldsymbol{p}^{'} = (\boldsymbol{I} + [\delta\theta\boldsymbol{n}]_{\times})\boldsymbol{p}$$

Axis-Angle representation of 3D Rotation

- Consider rotation by infinitesimal angle $\delta\theta$
- Which direction will point $\boldsymbol{p}$ move in ?
- Direction orthogonal to $\boldsymbol{p}$ and $\boldsymbol{p} - (\boldsymbol{p}^T\boldsymbol{n})\boldsymbol{n}$, i.e. $\boldsymbol{n} \times \boldsymbol{p}$

Theory and Formulation



$$p^{'} = (I + [\delta\theta n]_{\times})p$$

$$p^{'} = \Pi_{i=1}^{k}(I + [\frac{\theta}{k} n]_{\times})^{k} p$$

$$\Rightarrow R = \lim_{k\to\infty} (I + [\frac{\theta}{k} n]_{\times})^{k} = \exp\left([\omega]_{\times}\right)$$

Axis-Angle representation of 3D Rotation

- As before we can take limit of product of infinitesimal rotations
- $R = \exp\left([\omega]_{\times}\right) \in \mathbb{SO}(3)$

$$\boldsymbol{B}_x = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{pmatrix} \boldsymbol{B}_y = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{pmatrix}$$

$$\boldsymbol{B}_z = \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

Three-dimensional Rotations

- $\boldsymbol{R} \in \mathbb{SO}(3)$
- Corresponding Lie Algebra parameters $\omega \in \mathfrak{so}(3)$
- Lie Algebra : $[\omega]_\times = \omega_x \boldsymbol{B}_x + \omega_y \boldsymbol{B}_y + \omega_z \boldsymbol{B}_z$
- $\boldsymbol{R} = exp([\omega]_\times)$

Theory and Formulation

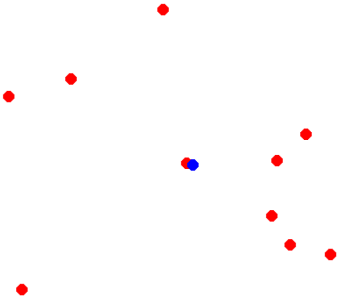
$\mathfrak{so}(3)$

$BCH(\omega_1, \omega_2) = \alpha\omega_1 + \beta\omega_2 + \gamma\omega_1 \times \omega_2$
α, β, γ : scalar functions of ω_1, ω_2

BCH forms for Motion Groups

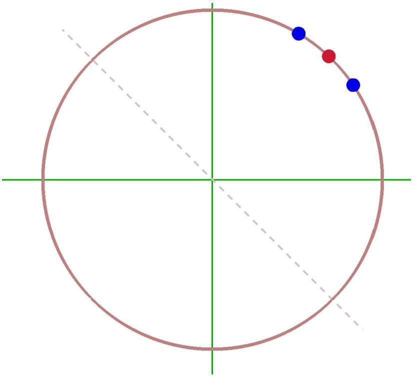
- Recall $e^{\boldsymbol{x}} e^{\boldsymbol{y}} \neq e^{(\boldsymbol{x}+\boldsymbol{y})}$
- Instead $e^{\boldsymbol{x}} e^{\boldsymbol{y}} = e^{BCH(\boldsymbol{x},\boldsymbol{y})}$
- General BCH series is unwieldy
- Dimensionality of $\mathfrak{so}(3)$ is 3
- $\mathfrak{so}(3)$ spanned by skew-sym forms of ω_1, ω_2 and $\omega_1 \times \omega_2$
- Closed forms available for BCH in $\mathfrak{so}(3)$ and $\mathfrak{se}(3)$
- BCH allows us to define distances between group elements

Theory and Formulation



Average of Linear Elements

- Given set of points $\{\boldsymbol{X}_i\}$
- Mean is $\mu = \frac{\sum_i \boldsymbol{X}_i}{\sum_i 1}$
- Does this work with nonlinear spaces ?



Averages of Nonlinear Elements

- No simple solution as in linear case
- Closed form might not exist
- Unique minimiser might not exist
- Harder to solve and prove properties

Theory and Formulation

Averaging on Lie Groups

- Motion Groups are **not** linear spaces
- $\frac{\boldsymbol{R}_1+\boldsymbol{R}_2}{2} \notin \mathbb{SO}(3)$
- Need an appropriate way of defining averages

Extrinsic Averages

- Ignore geometric constraints, e.g. $\boldsymbol{R}\boldsymbol{R}^T = \boldsymbol{I}$
- Carry out arithmetic average $\widehat{\boldsymbol{R}} = \frac{\boldsymbol{R}_1+\boldsymbol{R}_2}{2}$
- $\widehat{\boldsymbol{R}} \in \mathbb{R}^9$, i.e. $\widehat{\boldsymbol{R}} \notin \mathbb{SO}(3)$
- Project **extrinsic** estimate onto geometric manifold
- Works reasonably well for small noise
- $\mathcal{R} : \mathbb{R}^9 \mapsto \mathbb{SO}(3)$ here $\mathcal{R}$ is known (recall SVD method)
- Extrinsic estimation is easy to compute but non-optimal
- Difficult to assess quality of extrinsic estimation
- Eight point algorithm is an extrinsic estimator

Theory and Formulation

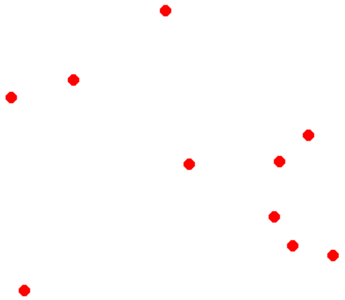
Averaging on Lie Groups

- Motion Groups are **not** linear spaces
- $\frac{\boldsymbol{R}_1+\boldsymbol{R}_2}{2} \notin \mathbb{SO}(3)$
- Need an appropriate way of defining averages

Intrinsic Estimators

- Averaging carried out while respecting geometric constraints
- In general, such intrinsic estimators are hard to define or solve
- Such averaging is easier on the Lie group
- $\mu\{\boldsymbol{R}_1, \cdots, \boldsymbol{R}_n\} \in \mathbb{SO}(3)$
- Recall Lie Group is both a group **and** a differentiable manifold
- Smooth properties of manifold can be used
- Averaging can be done via the Lie algebra

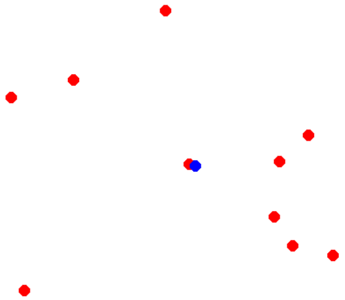
Theory and Formulation



Variational minimisation

- Given set of points $\{\boldsymbol{X}_i\}$
- Mean is $\mu = \frac{\sum_i \boldsymbol{X}_i}{\sum_i 1}$
- Alternate interpretation of μ as variational minimiser
- Minimises cost function $\boldsymbol{C} = \sum_i ||\boldsymbol{X}_i - \mu||^2$

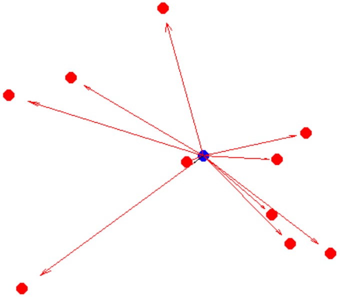
Theory and Formulation



Variational minimisation

- Given set of points $\{\boldsymbol{X}_i\}$
- Mean is $\mu = \frac{\sum_i \boldsymbol{X}_i}{\sum_i 1}$
- Alternate interpretation of μ as variational minimiser
- Minimises cost function $\boldsymbol{C} = \sum_i ||\boldsymbol{X}_i - \mu||^2$

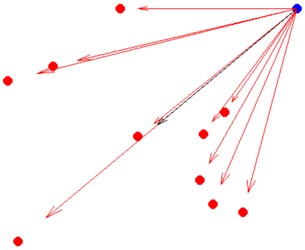
Theory and Formulation



Variational minimisation

- Given set of points $\{\boldsymbol{X}_i\}$
- Mean is $\mu = \frac{\sum_i \boldsymbol{X}_i}{\sum_i 1}$
- Alternate interpretation of μ as variational minimiser
- Minimises cost function $\boldsymbol{C} = \sum_i ||\boldsymbol{X}_i - \mu||^2$

Theory and Formulation



$$C = \sum_i (\boldsymbol{X}_i - \mu)^T (\boldsymbol{X}_i - \mu)$$

$$\nabla C = -2 \sum_i (\boldsymbol{X}_i - \mu)$$

$$\mu \leftarrow \mu - \lambda \nabla C$$

$$\mu \leftarrow \mu + \lambda \sum_i (\boldsymbol{X}_i - \mu)$$

- $\lambda = \frac{1}{N}$?
- Small λ ?
- ∇C is independent of choice of co-ordinates
- Not true for non-linear manifolds
- Need to 're-center' origin on Lie group

Averaging on Vector Spaces

- For vector space, sample mean of $\{\boldsymbol{X}_1, \cdots, \boldsymbol{X}_N\}$, $\bar{\boldsymbol{X}} = \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{X}_i$
- Mean is minimiser of "deviation" $\sum_{i=1}^{N} d^2(\boldsymbol{X}_i, \bar{\boldsymbol{X}}) = \sum_{i=1}^{N} \left(\boldsymbol{X}_i - \bar{\boldsymbol{X}}\right)^2$

Theory and Formulation

Averaging on Lie Groups

- **Extrinsic average:** $\bar{\boldsymbol{X}} = \mathcal{P}(\frac{1}{N}\sum_{i=1}^{N}\phi(\boldsymbol{X}_i))$
- **Intrinsic average:**

$$\arg\min_{\boldsymbol{X}\in\mathbb{G}}\sum_{i=1}^{N} d^2(\boldsymbol{X}_i, \boldsymbol{X})$$

- When $\boldsymbol{X} \in \mathcal{M}$ known as Karcher mean
- Generally hard to compute, but easier for Lie groups
- Exploit mapping from Lie group to algebra and vice-versa

Averaging on Lie Groups

- **Intrinsic average**:

$$\arg \min_{\boldsymbol{X} \in \mathbb{G}} \sum_{i=1}^{N} d^2(\boldsymbol{X}_i, \boldsymbol{X})$$

- $d(\boldsymbol{X}, \boldsymbol{Y}) = d(\boldsymbol{I}, \boldsymbol{X}^{-1}\boldsymbol{Y})$ (left-invariance)
- Geodesic distance on $\mathbb{SO}(3)$: $d(\boldsymbol{R}_1, \boldsymbol{R}_2) = \frac{1}{\sqrt{2}} || \log(\boldsymbol{R}_1 \boldsymbol{R}_2^{-1}) ||_F$
- Each term $d(.,.)$ can be expanded using BCH
- Approximate BCH (intrinsic distance) as $d(\boldsymbol{X}, \boldsymbol{Y}) = || \log(\boldsymbol{X}^{-1}\boldsymbol{Y}) || \approx || \log(\boldsymbol{Y}) - \log(\boldsymbol{X}) || = ||\boldsymbol{y} - \boldsymbol{x}||$
- Recall Lie algebra is vector space

Theory and Formulation

Algorithm for Intrinsic Average

Input : $\{\boldsymbol{X}_1, \cdots, \boldsymbol{X}_N\} \in \mathbb{G}$ (Matrix Group)
Output : $\boldsymbol{\mu} \in \mathbb{G}$ (Intrinsic Average)
Initialise : $\boldsymbol{\mu} = \boldsymbol{I}$ (Identity)

Do

$\Delta \boldsymbol{X}_i = \boldsymbol{\mu}^{-1} \boldsymbol{X}_i$

$\Delta \boldsymbol{x}_i = log(\Delta \boldsymbol{X}_i)$

$\Delta \boldsymbol{\mu} = exp(\frac{1}{N} \sum_{i=1}^{N} \Delta \boldsymbol{x}_i)$

$\boldsymbol{\mu} = \boldsymbol{\mu} \Delta \boldsymbol{\mu}$

Repeat till $||\Delta \boldsymbol{\mu}|| < \epsilon$

Algorithm for Intrinsic Average

Input : $\{\boldsymbol{X}_1, \cdots, \boldsymbol{X}_N\} \in \mathbb{G}$ (Matrix Group)
Output : $\boldsymbol{\mu} \in \mathbb{G}$ (Intrinsic Average)
Initialise : $\boldsymbol{\mu} = \boldsymbol{I}$ (Identity)

Do

$\Delta\boldsymbol{X}_i = \boldsymbol{\mu}^{-1}\boldsymbol{X}_i$

$\Delta\boldsymbol{x}_i = log(\Delta\boldsymbol{X}_i)$

$\Delta\boldsymbol{\mu} = exp(\frac{1}{N}\sum_{i=1}^{N}\Delta\boldsymbol{x}_i)$

$\boldsymbol{\mu} = \boldsymbol{\mu}\Delta\boldsymbol{\mu}$

Repeat till $||\Delta\boldsymbol{\mu}|| < \epsilon$

Algorithm for Intrinsic Average

Input : $\{\boldsymbol{X}_1, \cdots, \boldsymbol{X}_N\} \in \mathbb{G}$ (Matrix Group)
Output : $\boldsymbol{\mu} \in \mathbb{G}$ (Intrinsic Average)
Initialise : $\boldsymbol{\mu} = \boldsymbol{I}$ (Identity)
Do
$\Delta \boldsymbol{X}_i = \boldsymbol{\mu}^{-1} \boldsymbol{X}_i$ Subtract current mean from observations
$\Delta \boldsymbol{x}_i = log(\Delta \boldsymbol{X}_i)$
$\Delta \boldsymbol{\mu} = exp(\frac{1}{N} \sum_{i=1}^{N} \Delta \boldsymbol{x}_i)$
$\boldsymbol{\mu} = \boldsymbol{\mu} \Delta \boldsymbol{\mu}$
Repeat till $||\Delta \boldsymbol{\mu}|| < \epsilon$

Algorithm for Intrinsic Average

Input : $\{\boldsymbol{X}_1, \cdots, \boldsymbol{X}_N\} \in \mathbb{G}$ (Matrix Group)
Output : $\boldsymbol{\mu} \in \mathbb{G}$ (Intrinsic Average)
Initialise : $\boldsymbol{\mu} = \boldsymbol{I}$ (Identity)

Do

$\Delta\boldsymbol{X}_i = \boldsymbol{\mu}^{-1}\boldsymbol{X}_i$ Subtract current mean from observations

$\Delta\boldsymbol{x}_i = log(\Delta\boldsymbol{X}_i)$ Map to Lie Algebra

$\Delta\boldsymbol{\mu} = exp(\frac{1}{N}\sum_{i=1}^{N}\Delta\boldsymbol{x}_i)$

$\boldsymbol{\mu} = \boldsymbol{\mu}\Delta\boldsymbol{\mu}$

Repeat till $||\Delta\boldsymbol{\mu}|| < \epsilon$

Algorithm for Intrinsic Average

Input : $\{\boldsymbol{X}_1, \cdots, \boldsymbol{X}_N\} \in \mathbb{G}$ (Matrix Group)
Output : $\boldsymbol{\mu} \in \mathbb{G}$ (Intrinsic Average)
Initialise : $\boldsymbol{\mu} = \boldsymbol{I}$ (Identity)

Do

$\Delta \boldsymbol{X}_i = \boldsymbol{\mu}^{-1} \boldsymbol{X}_i$ — Subtract current mean from observations

$\Delta \boldsymbol{x}_i = log(\Delta \boldsymbol{X}_i)$ — Map to Lie Algebra

$\Delta \boldsymbol{\mu} = exp(\frac{1}{N} \sum_{i=1}^{N} \Delta \boldsymbol{x}_i)$ — Average and map back to Lie Group

$\boldsymbol{\mu} = \boldsymbol{\mu} \Delta \boldsymbol{\mu}$

Repeat till $||\Delta \boldsymbol{\mu}|| < \epsilon$

Theory and Formulation

Algorithm for Intrinsic Average

Input : $\{\boldsymbol{X}_1, \cdots, \boldsymbol{X}_N\} \in \mathbb{G}$ (Matrix Group)
Output : $\boldsymbol{\mu} \in \mathbb{G}$ (Intrinsic Average)
Initialise : $\boldsymbol{\mu} = \boldsymbol{I}$ (Identity)

Do

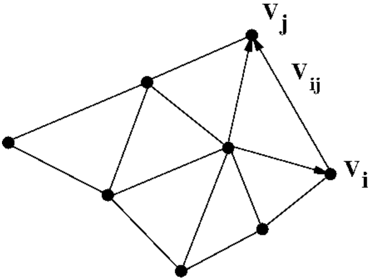
$\Delta\boldsymbol{X}_i = \boldsymbol{\mu}^{-1}\boldsymbol{X}_i$	Subtract current mean from observations
$\Delta\boldsymbol{x}_i = log(\Delta\boldsymbol{X}_i)$	Map to Lie Algebra
$\Delta\boldsymbol{\mu} = exp(\frac{1}{N}\sum_{i=1}^{N}\Delta\boldsymbol{x}_i)$	Average and map back to Lie Group
$\boldsymbol{\mu} = \boldsymbol{\mu}\Delta\boldsymbol{\mu}$	Add update to current mean

Repeat till $||\Delta\boldsymbol{\mu}|| < \epsilon$

Properties of Intrinsic Average

- Uses averaging in vector space of Lie algebra to move
- First-order approximation of BCH simplifies algorithm
- BCH approximation $\nRightarrow$ approximate solution
- Estimate is on manifold at all times
- **Convergence:**
  - Non-convex in general
  - Use weak convexity notion for closed groups
  - If $\boldsymbol{X}_i$ within closed ball of radius r
  - For $\mathbb{SO}(3)$, $r < \frac{\pi}{2}$ (Manton 2004, Hartley *et al.* 2013)
  - Riemannian gradient descent step

Theory and Formulation



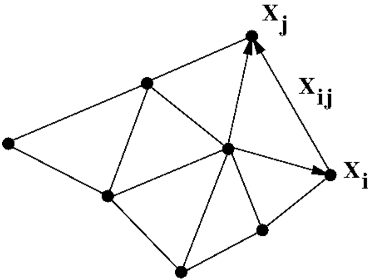
Observations: $\boldsymbol{v}_j - \boldsymbol{v}_i = \boldsymbol{v}_{ij}$

Cost function: $\sum_{\mathcal{E}} d^2(\boldsymbol{v}_{ij}, \boldsymbol{v}_j - \boldsymbol{v}_i)$

Motion Averaging in Vector Space

- Linear system of equations : $\boldsymbol{AV} = \boldsymbol{V}_{ij}$
- $\boldsymbol{A}$ encodes the viewgraph
- Averages the information on edges

Theory and Formulation



Observations: $\boldsymbol{M}_j \boldsymbol{M}_i^{-1} = \boldsymbol{M}_{ij}$

Cost function: $\sum_{\mathcal{E}} d^2(\boldsymbol{M}_{ij}, \boldsymbol{M}_j \boldsymbol{M}_i^{-1})$

Motion Averaging on Lie Groups

- No linear solution
- Can use averaging in Lie algebra (vector space)
- Iterative method as in case of averaging

Theory and Formulation

$$\text{Cost function: } \sum_{\mathcal{E}} d^2(\boldsymbol{M}_{ij}, \boldsymbol{M}_j \boldsymbol{M}_i^{-1})$$

Motion Averaging on Lie Groups

- No linear solution
- Can use averaging in Lie algebra (vector space)
- Iterative method as in case of averaging

Theory and Formulation

$$\text{Cost function: } \sum_{\mathcal{E}} d^2(\boldsymbol{M}_{ij}, \boldsymbol{M}_j \boldsymbol{M}_i^{-1})$$

$$d(\boldsymbol{M}_{ij}, \boldsymbol{M}_j \boldsymbol{M}_i^{-1}) = BCH(\boldsymbol{M}_{ij}, \boldsymbol{M}_i \boldsymbol{M}_j^{-1})$$

$$\Rightarrow d(\boldsymbol{M}_{ij}, \boldsymbol{M}_j \boldsymbol{M}_i^{-1}) = BCH(\boldsymbol{M}_{ij}, BCH(\boldsymbol{M}_i, \boldsymbol{M}_j^{-1}))$$

Motion Averaging on Lie Groups

- No linear solution
- Can use averaging in Lie algebra (vector space)
- Iterative method as in case of averaging

Theory and Formulation

$$\text{Cost function: } \sum_{\mathcal{E}} d^2(\boldsymbol{M}_{ij}, \boldsymbol{M}_j \boldsymbol{M}_i^{-1})$$

$$d(\boldsymbol{M}_{ij}, \boldsymbol{M}_j \boldsymbol{M}_i^{-1}) = BCH(\boldsymbol{M}_{ij}, \boldsymbol{M}_i \boldsymbol{M}_j^{-1})$$

$$\Rightarrow d(\boldsymbol{M}_{ij}, \boldsymbol{M}_j \boldsymbol{M}_i^{-1}) = BCH(\boldsymbol{M}_{ij}, BCH(\boldsymbol{M}_i, \boldsymbol{M}_j^{-1}))$$

$$\mathfrak{m}_j - \mathfrak{m}_i = \mathfrak{m}_{ij}$$

Motion Averaging on Lie Groups

- No linear solution
- Can use averaging in Lie algebra (vector space)
- Iterative method as in case of averaging
- First-order expansion of BCH form

Theory and Formulation

$$\text{Cost function: } \sum_{\mathcal{E}} d^2(\boldsymbol{M}_{ij}, \boldsymbol{M}_j \boldsymbol{M}_i^{-1})$$

$$d(\boldsymbol{M}_{ij}, \boldsymbol{M}_j \boldsymbol{M}_i^{-1}) = BCH(\boldsymbol{M}_{ij}, \boldsymbol{M}_i \boldsymbol{M}_j^{-1})$$

$$\Rightarrow d(\boldsymbol{M}_{ij}, \boldsymbol{M}_j \boldsymbol{M}_i^{-1}) = BCH(\boldsymbol{M}_{ij}, BCH(\boldsymbol{M}_i, \boldsymbol{M}_j^{-1}))$$

$$\mathfrak{m}_j - \mathfrak{m}_i = \mathfrak{m}_{ij}$$

Motion Averaging on Lie Groups

- First-order expansion of BCH form

Theory and Formulation

$$\text{Cost function: } \sum_{\mathcal{E}} d^2(\boldsymbol{M}_{ij}, \boldsymbol{M}_j \boldsymbol{M}_i^{-1})$$

$$d(\boldsymbol{M}_{ij}, \boldsymbol{M}_j \boldsymbol{M}_i^{-1}) = BCH(\boldsymbol{M}_{ij}, \boldsymbol{M}_i \boldsymbol{M}_j^{-1})$$

$$\Rightarrow d(\boldsymbol{M}_{ij}, \boldsymbol{M}_j \boldsymbol{M}_i^{-1}) = BCH(\boldsymbol{M}_{ij}, BCH(\boldsymbol{M}_i, \boldsymbol{M}_j^{-1}))$$

$$\mathfrak{m}_j - \mathfrak{m}_i = \mathfrak{m}_{ij}$$

$$\mathfrak{v}_j - \mathfrak{v}_i = \mathfrak{v}_{ij}$$

Motion Averaging on Lie Groups

- First-order expansion of BCH form
- Extract parameters $\mathfrak{v}$ of Lie algebra $\mathfrak{m}$

Theory and Formulation

$$\text{Cost function: } \sum_{\mathcal{E}} d^2(\boldsymbol{M}_{ij}, \boldsymbol{M}_j \boldsymbol{M}_i^{-1})$$

$$d(\boldsymbol{M}_{ij}, \boldsymbol{M}_j \boldsymbol{M}_i^{-1}) = BCH(\boldsymbol{M}_{ij}, \boldsymbol{M}_i \boldsymbol{M}_j^{-1})$$

$$\Rightarrow d(\boldsymbol{M}_{ij}, \boldsymbol{M}_j \boldsymbol{M}_i^{-1}) = BCH(\boldsymbol{M}_{ij}, BCH(\boldsymbol{M}_i, \boldsymbol{M}_j^{-1}))$$

$$\mathfrak{m}_j - \mathfrak{m}_i = \mathfrak{m}_{ij}$$

$$\mathfrak{v}_j - \mathfrak{v}_i = \mathfrak{v}_{ij}$$

$$\underbrace{\begin{bmatrix} \cdots - \boldsymbol{I} \cdots \boldsymbol{I} \cdots \end{bmatrix}}_{\boldsymbol{A}_{ij}} \mathfrak{V} = \mathfrak{v}_{ij}$$

Motion Averaging on Lie Groups

- First-order expansion of BCH form
- Extract parameters $\mathfrak{v}$ of Lie algebra $\mathfrak{m}$
- Notice that $\mathfrak{v}$ forms a vector space
- $\mathfrak{V}$ is stack of $\mathfrak{v}$ of all $N = |\mathcal{V}|$ vertices
- $\mathfrak{V} = [\mathfrak{v}_1; \mathfrak{v}_2; \cdots; \mathfrak{v}_N]$

Theory and Formulation

$$\text{Cost function: } \sum_{\mathcal{E}} d^2(\boldsymbol{M}_{ij}, \boldsymbol{M}_j \boldsymbol{M}_i^{-1})$$

$$\underbrace{\left[\ \cdots - \boldsymbol{I} \cdots \boldsymbol{I} \cdots\ \right]}_{\boldsymbol{A}_{ij}} \mathfrak{V} = \mathfrak{v}_{ij}$$

$$\boldsymbol{M}_j \boldsymbol{M}_i^{-1} = \boldsymbol{M}_{ij} \rightsquigarrow \boldsymbol{A}\mathfrak{V} = \mathbb{V}_{ij}$$

Motion Averaging on Lie Groups

- $\mathfrak{V}$ is stack of $\mathfrak{v}$ of all vertices
- $\mathfrak{V} = [\mathfrak{v}_1; \mathfrak{v}_2; \cdots; \mathfrak{v}_N]$
- Each edge in $\mathcal{E}$ contributes one such equation
- Collect relationships contributed by all edges in $\mathcal{E}$
- $\boldsymbol{A} = [\boldsymbol{A}_{ij1}; \boldsymbol{A}_{ij2}; \cdots]$
- Stack all observations $\mathbb{V}_{ij} = [\mathfrak{v}_{ij1}; \mathfrak{v}_{ij2}; \cdots]$

Theory and Formulation

$$\text{Cost function: } \sum_{\mathcal{E}} d^2(\boldsymbol{M}_{ij}, \boldsymbol{M}_j \boldsymbol{M}_i^{-1})$$

$$\underbrace{\left[\; \cdots - \boldsymbol{I} \cdots \boldsymbol{I} \cdots \; \right]}_{\boldsymbol{A}_{ij}} \mathfrak{V} = \mathfrak{v}_{ij}$$

$$\boldsymbol{M}_j \boldsymbol{M}_i^{-1} = \boldsymbol{M}_{ij} \rightsquigarrow \boldsymbol{A}\mathfrak{V} = \mathbb{V}_{ij}$$

$$\text{Solve system of equations: } \boldsymbol{A}\mathfrak{V} = \mathbb{V}_{ij}$$

Motion Averaging on Lie Groups

- $\mathfrak{V}$ is stack of $\mathfrak{v}$ of all vertices
- $\mathfrak{V} = [\mathfrak{v}_1; \mathfrak{v}_2; \cdots; \mathfrak{v}_N]$
- Each edge in $\mathcal{E}$ contributes one such equation
- Collect relationships contributed by all edges in $\mathcal{E}$
- $\boldsymbol{A} = [\boldsymbol{A}_{ij1}; \boldsymbol{A}_{ij2}; \cdots]$
- Stack all observations $\mathbb{V}_{ij} = [\mathfrak{v}_{ij1}; \mathfrak{v}_{ij2}; \cdots]$

Theory and Formulation

Algorithm for Motion Averaging on Lie Groups

Input : $\{\boldsymbol{M}_{ij1}, \boldsymbol{M}_{ij2} \cdots, \boldsymbol{M}_{ijn}\}$
Output : $\boldsymbol{M}_g : \{\boldsymbol{M}_2, \cdots, \boldsymbol{M}_N\}$
Set $\boldsymbol{M}_g$ to an initial guess (Linear solution)
Do
$\Delta\boldsymbol{M}_{ij} = \boldsymbol{M}_j{}^{-1}\boldsymbol{M}_{ij}\boldsymbol{M}_i$
$\Delta\mathfrak{m}_{ij} = log(\Delta\boldsymbol{M}_{ij})$
$\Delta\mathfrak{v}_{ij} = vec(\mathfrak{m}_{ij})$
$Solve\ \boldsymbol{A}\Delta\mathfrak{V} = \Delta\mathbb{V}_{ij}$
$\forall k \in [2, N], \boldsymbol{M}_k = \boldsymbol{M}_k exp(\Delta\mathfrak{v}_k)$
$Repeat\ till\ ||\Delta\mathfrak{V}|| < \epsilon$

Algorithm for Motion Averaging on Lie Groups

Input : $\{\boldsymbol{M}_{ij1}, \boldsymbol{M}_{ij2} \cdots, \boldsymbol{M}_{ijn}\}$
Output : $\boldsymbol{M}_g : \{\boldsymbol{M}_2, \cdots, \boldsymbol{M}_N\}$
Set $\boldsymbol{M}_g$ to an initial guess (Linear solution)
Do
$\Delta\boldsymbol{M}_{ij} = \boldsymbol{M}_j{}^{-1}\boldsymbol{M}_{ij}\boldsymbol{M}_i$
$\Delta\mathfrak{m}_{ij} = log(\Delta\boldsymbol{M}_{ij})$
$\Delta\mathfrak{v}_{ij} = vec(\mathfrak{m}_{ij})$
$Solve\ \boldsymbol{A}\Delta\mathfrak{V} = \Delta\mathbb{V}_{ij}$
$\forall k \in [2, N], \boldsymbol{M}_k = \boldsymbol{M}_k exp(\Delta\mathfrak{v}_k)$
$Repeat\ till\ ||\Delta\mathfrak{V}|| < \epsilon$

Theory and Formulation

Algorithm for Motion Averaging on Lie Groups

Input : $\{\boldsymbol{M}_{ij1}, \boldsymbol{M}_{ij2} \cdots, \boldsymbol{M}_{ijn}\}$
Output : $\boldsymbol{M}_g : \{\boldsymbol{M}_2, \cdots, \boldsymbol{M}_N\}$
Set $\boldsymbol{M}_g$ to an initial guess (Linear solution)
Do
$\Delta \boldsymbol{M}_{ij} = \boldsymbol{M}_j{}^{-1} \boldsymbol{M}_{ij} \boldsymbol{M}_i$ Diff of observations and estimate
$\Delta \mathfrak{m}_{ij} = log(\Delta \boldsymbol{M}_{ij})$
$\Delta \mathfrak{v}_{ij} = vec(\mathfrak{m}_{ij})$
Solve $\boldsymbol{A} \Delta \mathfrak{V} = \Delta \mathbb{V}_{ij}$
$\forall k \in [2, N], \boldsymbol{M}_k = \boldsymbol{M}_k exp(\Delta \mathfrak{v}_k)$
Repeat till $||\Delta \mathfrak{V}|| < \epsilon$

Algorithm for Motion Averaging on Lie Groups

Input : $\{\boldsymbol{M}_{ij1}, \boldsymbol{M}_{ij2} \cdots, \boldsymbol{M}_{ijn}\}$
Output : $\boldsymbol{M}_g : \{\boldsymbol{M}_2, \cdots, \boldsymbol{M}_N\}$
Set $\boldsymbol{M}_g$ to an initial guess (Linear solution)

Do

$\Delta \boldsymbol{M}_{ij} = \boldsymbol{M}_j{}^{-1} \boldsymbol{M}_{ij} \boldsymbol{M}_i$ — Diff of observations and estimate

$\Delta \mathfrak{m}_{ij} = log(\Delta \boldsymbol{M}_{ij})$ — Move into Lie algebra

$\Delta \mathfrak{v}_{ij} = vec(\mathfrak{m}_{ij})$

Solve $\boldsymbol{A} \Delta \mathfrak{V} = \Delta \mathbb{V}_{ij}$

$\forall k \in [2, N], \boldsymbol{M}_k = \boldsymbol{M}_k exp(\Delta \mathfrak{v}_k)$

Repeat till $||\Delta \mathfrak{V}|| < \epsilon$

Algorithm for Motion Averaging on Lie Groups

Input : $\{\boldsymbol{M}_{ij1}, \boldsymbol{M}_{ij2} \cdots, \boldsymbol{M}_{ijn}\}$

Output : $\boldsymbol{M}_g : \{\boldsymbol{M}_2, \cdots, \boldsymbol{M}_N\}$

Set $\boldsymbol{M}_g$ to an initial guess (Linear solution)

Do

$\Delta\boldsymbol{M}_{ij} = \boldsymbol{M}_j{}^{-1}\boldsymbol{M}_{ij}\boldsymbol{M}_i$ — Diff of observations and estimate

$\Delta\mathfrak{m}_{ij} = log(\Delta\boldsymbol{M}_{ij})$ — Move into Lie algebra

$\Delta\mathfrak{v}_{ij} = vec(\mathfrak{m}_{ij})$ — Extract parameters in Lie algebra

$Solve\ \boldsymbol{A}\Delta\mathfrak{V} = \Delta\mathbb{V}_{ij}$

$\forall k \in [2, N], \boldsymbol{M}_k = \boldsymbol{M}_k exp(\Delta\mathfrak{v}_k)$

$Repeat\ till\ ||\Delta\mathfrak{V}|| < \epsilon$

Algorithm for Motion Averaging on Lie Groups

Input : $\{\boldsymbol{M}_{ij1}, \boldsymbol{M}_{ij2} \cdots, \boldsymbol{M}_{ijn}\}$
Output : $\boldsymbol{M}_g : \{\boldsymbol{M}_2, \cdots, \boldsymbol{M}_N\}$
Set $\boldsymbol{M}_g$ to an initial guess (Linear solution)

Do	
$\Delta\boldsymbol{M}_{ij} = \boldsymbol{M}_j^{-1}\boldsymbol{M}_{ij}\boldsymbol{M}_i$	Diff of observations and estimate
$\Delta\mathfrak{m}_{ij} = log(\Delta\boldsymbol{M}_{ij})$	Move into Lie algebra
$\Delta\mathfrak{v}_{ij} = vec(\mathfrak{m}_{ij})$	Extract parameters in Lie algebra
Solve $\boldsymbol{A}\Delta\mathfrak{V} = \Delta\mathbb{V}_{ij}$	Solve in vector space
$\forall k \in [2, N], \boldsymbol{M}_k = \boldsymbol{M}_k exp(\Delta\mathfrak{v}_k)$	
Repeat till $\|\|\Delta\mathfrak{V}\|\| < \epsilon$	

Theory and Formulation

Algorithm for Motion Averaging on Lie Groups

Input : $\{\boldsymbol{M}_{ij1}, \boldsymbol{M}_{ij2} \cdots, \boldsymbol{M}_{ijn}\}$
Output : $\boldsymbol{M}_g : \{\boldsymbol{M}_2, \cdots, \boldsymbol{M}_N\}$
Set $\boldsymbol{M}_g$ to an initial guess (Linear solution)

Do	
$\Delta\boldsymbol{M}_{ij} = \boldsymbol{M}_j^{-1}\boldsymbol{M}_{ij}\boldsymbol{M}_i$	Diff of observations and estimate
$\Delta\mathfrak{m}_{ij} = log(\Delta\boldsymbol{M}_{ij})$	Move into Lie algebra
$\Delta\mathfrak{v}_{ij} = vec(\mathfrak{m}_{ij})$	Extract parameters in Lie algebra
$Solve\ \boldsymbol{A}\Delta\mathfrak{V} = \Delta\mathbb{V}_{ij}$	Solve in vector space
$\forall k \in [2, N], \boldsymbol{M}_k = \boldsymbol{M}_k exp(\Delta\mathfrak{v}_k)$	Update motions
$Repeat\ till\ \lvert\lvert\Delta\mathfrak{V}\rvert\rvert < \epsilon$	

Theory and Formulation

Input : $\{\boldsymbol{M}_{ij1}, \boldsymbol{M}_{ij2} \cdots, \boldsymbol{M}_{ijn}\}$
Output : $\boldsymbol{M}_g : \{\boldsymbol{M}_2, \cdots, \boldsymbol{M}_N\}$
Set $\boldsymbol{M}_g$ to an initial guess (Linear solution)

Do

$\Delta \boldsymbol{M}_{ij} = \boldsymbol{M}_j{}^{-1} \boldsymbol{M}_{ij} \boldsymbol{M}_i$

$\Delta \mathfrak{m}_{ij} = log(\Delta \boldsymbol{M}_{ij})$

$\Delta \mathfrak{v}_{ij} = vec(\mathfrak{m}_{ij})$

$Solve\ \boldsymbol{A} \Delta \mathfrak{V} = \Delta \mathbb{V}_{ij}$

$\forall k \in [2, N], \boldsymbol{M}_k = \boldsymbol{M}_k exp(\Delta \mathfrak{v}_k)$

$Repeat\ till\ ||\Delta \mathfrak{V}|| < \epsilon$

Properties of $\boldsymbol{A}$

- Row of $\boldsymbol{A}$ has sparse entries of ± 1 for $v_j - v_i = v_{ij}$
- $\boldsymbol{A}^T \boldsymbol{A}$: graph Laplacian
- For matrix problems, form is $\boldsymbol{A} \otimes \boldsymbol{I}$
- Note that $\boldsymbol{A}$ only depends on $\mathcal{E}$ (fixed)
- $\boldsymbol{A}$ is sparse with ± 1 entries
- Multiplications $\boldsymbol{A}\mathfrak{V}$ as additions

Theory and Formulation

Input : $\{\boldsymbol{M}_{ij1}, \boldsymbol{M}_{ij2} \cdots, \boldsymbol{M}_{ijn}\}$
Output : $\boldsymbol{M}_g : \{\boldsymbol{M}_2, \cdots, \boldsymbol{M}_N\}$
Set $\boldsymbol{M}_g$ to an initial guess (Linear solution)
Do
$\Delta \boldsymbol{M}_{ij} = \boldsymbol{M}_j^{-1} \boldsymbol{M}_{ij} \boldsymbol{M}_i$
$\Delta \mathfrak{m}_{ij} = log(\Delta \boldsymbol{M}_{ij})$
$\Delta \mathfrak{v}_{ij} = vec(\mathfrak{m}_{ij})$
$Solve\ \boldsymbol{A} \Delta \mathfrak{V} = \Delta \mathbb{V}_{ij}$
$\forall k \in [2, N], \boldsymbol{M}_k = \boldsymbol{M}_k exp(\Delta \mathfrak{v}_k)$
$Repeat\ till\ ||\Delta \mathfrak{V}|| < \epsilon$

Solutions

- Key issue is solution for $\boldsymbol{A} \Delta \mathfrak{V} = \Delta \mathbb{V}_{ij}$
- Intrinsic methods differ only in this step
    - Least-squares (Govindu 2004)
    - Distributed approach (Weiszfeld; Hartley *et al.* 2011)
    - Robust Least-squares (Chatterjee & Govindu 2013, 2017)

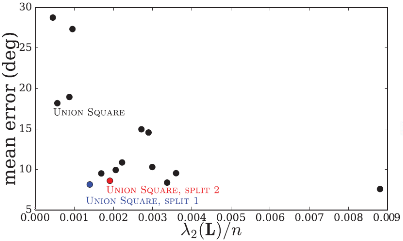
"Convexity" of Rotation Averaging

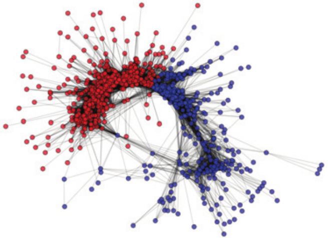
- Rotation Averaging is a non-convex problem
- Main problem: Two geodesic segments joining two points on $\mathbb{SO}(3)$
- Ambiguity when geodesic distance is π
- Weak convexity: $U \subset \mathbb{SO}(3)$
- All $\boldsymbol{R}_1, \boldsymbol{R}_2 \in U$ have only one geodesic in U
- Hessian of $f : \mathbb{SO}(3) \to \mathbb{R}$
- If Hessian psd at $\boldsymbol{R}$, f is locally convex at $\boldsymbol{R}$
- $\mathbb{SO}(3)$ is locally convex almost everywhere
- Hessian $\boldsymbol{H}_{ij}$ of $d^2(\boldsymbol{R}_{ij}, \boldsymbol{R}_j \boldsymbol{R}_i^{-1})$ at $(\boldsymbol{R}_i, \boldsymbol{R}_j)$
- $\boldsymbol{H}_{ij}$ is positive semidefinite at $d(.,.) = 0$, indefinite elsewhere
- Important issue is gauge ambiguity

Hartley *et al.* "Rotation Averaging" 2013

Wilson *et al.* "When is Rotations Averaging Hard ?" 2016

Theory and Formulation

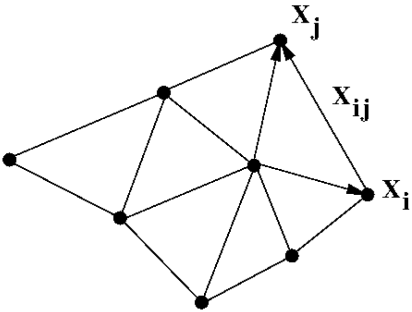




Analysis of Averaging (Wilson *et al.* 2016)

- Rotation Averaging is a non-convex problem
- "Hardness" is not the same for all datasets (problems)
- Interaction between viewgraph structure (Laplacian) and noise
- PSD Hessian $\implies$ local convexity of problem
- Fixing gauge ambiguity plays important role
- High degree of connectivity *improves* convexity
- Need second eigen value of Laplacian to be large enough
- CRLB of rotation averaging available (Boumal *et al.*)

Theory and Formulation



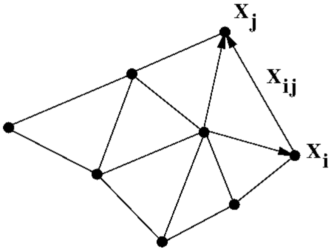
$$\min_{\boldsymbol{y}} \sum_i \|\boldsymbol{x}_i - \boldsymbol{y}\|^2$$

$$\nabla = -2 \sum_i (\boldsymbol{x}_i - \boldsymbol{y})$$

$$\Rightarrow \boldsymbol{y}_i \leftarrow (1-\lambda)\boldsymbol{y}_i + \lambda \frac{1}{|\mathcal{N}_i|} \sum_i \boldsymbol{x}_i$$

Distributed Averaging

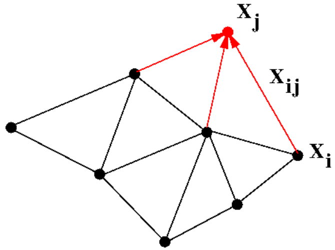
- Distributed Consensus methods (mean computation)
- No central observer
- Will converge under appropriate assumptions
- Rate of convergence depends on spectral radius of graph
- Can be very slow
- Weiszfeld algorithm is a robust variant



Distributed Averaging on $\mathbb{SO}(3)$

- $\boldsymbol{X}_j \boldsymbol{X}_i^{-1} = \boldsymbol{X}_{ij}$
- $\boldsymbol{X}_j = \boldsymbol{X}_{ij} \boldsymbol{X}_i$
- Each vertex in $\mathcal{N}(j)$ suggests $\boldsymbol{X}_j$
- Averaging takes consensus of these estimates
- Each $\boldsymbol{R}_j$ updated $\forall j \in \mathcal{V}$
- Repeat till convergence
- Weiszfeld algorithm is a robust variant

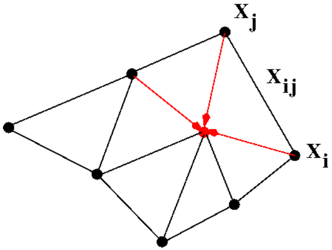
Theory and Formulation



Distributed Averaging on $\mathbb{SO}(3)$

- $\boldsymbol{X}_j \boldsymbol{X}_i^{-1} = \boldsymbol{X}_{ij}$
- $\boldsymbol{X}_j = \boldsymbol{X}_{ij} \boldsymbol{X}_i$
- Each vertex in $\mathcal{N}(j)$ suggests $\boldsymbol{X}_j$
- Averaging takes consensus of these estimates
- Each $\boldsymbol{R}_j$ updated $\forall j \in \mathcal{V}$
- Repeat till convergence
- Weiszfeld algorithm is a robust variant

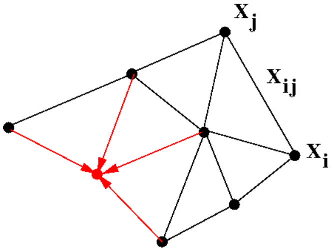
Theory and Formulation



Distributed Averaging on $\mathbb{SO}(3)$

- $\boldsymbol{X}_j \boldsymbol{X}_i^{-1} = \boldsymbol{X}_{ij}$
- $\boldsymbol{X}_j = \boldsymbol{X}_{ij} \boldsymbol{X}_i$
- Each vertex in $\mathcal{N}(j)$ suggests $\boldsymbol{X}_j$
- Averaging takes consensus of these estimates
- Each $\boldsymbol{R}_j$ updated $\forall j \in \mathcal{V}$
- Repeat till convergence
- Weiszfeld algorithm is a robust variant

Theory and Formulation



Distributed Averaging on $\mathbb{SO}(3)$

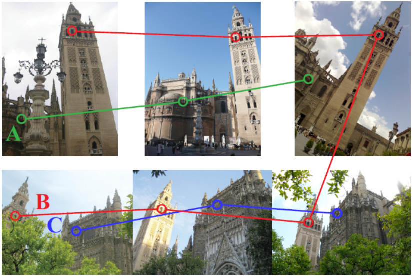
- $\boldsymbol{X}_j \boldsymbol{X}_i^{-1} = \boldsymbol{X}_{ij}$
- $\boldsymbol{X}_j = \boldsymbol{X}_{ij} \boldsymbol{X}_i$
- Each vertex in $\mathcal{N}(j)$ suggests $\boldsymbol{X}_j$
- Averaging takes consensus of these estimates
- Each $\boldsymbol{R}_j$ updated $\forall j \in \mathcal{V}$
- Repeat till convergence
- Weiszfeld algorithm is a robust variant

Robustness

Outliers in data

- Outliers are ubiquitous in real data
- Estimation needs to be robust
- Large body of literature in statistics and computer vision
- Gross errors in camera motion estimates
- Multiple sources of outliers that cause errors

Robustness

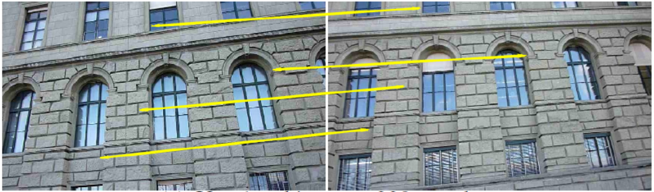


Outliers due to symmetric structures

- Feature matches for symmetric structures
- Results in grossly wrong feature tracks

Figure from Wilson & Snavely "Network Principles for SfM"

Robustness



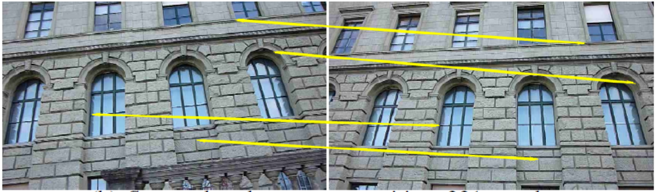
(a) Unrelated images, 228 matches

Outliers in data

- Outliers are ubiquitous in real data
- Estimation needs to be robust
- Large body of literature in statistics and computer vision
- Multiple sources of outliers
  - Repeated structures are common in man-made objects
  - Translational symmetry
  - Results in wrong local feature matches across views

Figure from Zach *et al.* “Disambiguating Visual Relations Using Loop Constraints”

Robustness



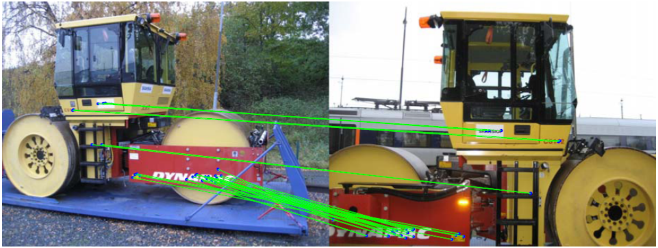
(b) Snapped to the wrong repetition, 331 matches

Outliers in data

- Outliers are ubiquitous in real data
- Estimation needs to be robust
- Large body of literature in statistics and computer vision
- Multiple sources of outliers
  - Repeated structures are common in man-made objects
  - Translational symmetry
  - Results in wrong local feature matches across views

Figure from Zach *et al.* "Disambiguating Visual Relations Using Loop Constraints"

Robustness

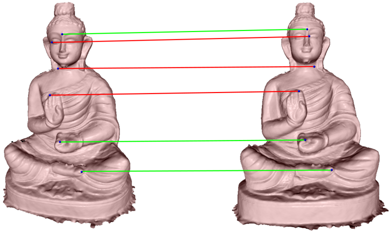


Outliers in data

- Wrong matches based on local features
- Outliers are geometrically consistent
- Impossible to disambiguate using epipolar relationships
- Need global inference of consistency of motion estimates

From "Non-sequential Structure from Motion"

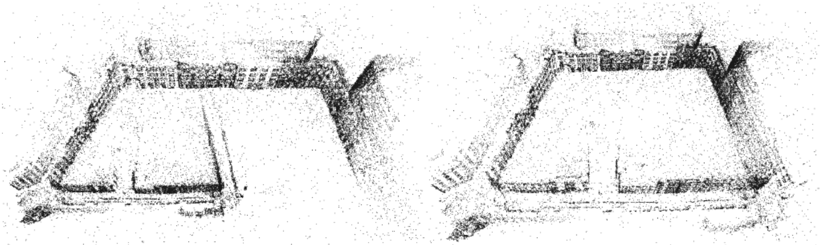
Robustness



Outliers in data

- Outliers are ubiquitous in real data
- Estimation needs to be robust
- Large body of literature in statistics and computer vision
- Multiple sources of outliers
  - ICP makes greedy decisions for point correspondences across scans
  - Heuristics for filtering can be inadequate
  - Can result in catastrophic failure to register 3D scans

Robustness



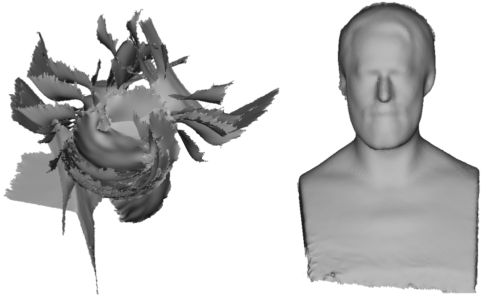
(a) 3D model using all visual relations (b) 3D model using only consistent relations

Errors in viewgraph relations

- Multiple sources of errors in low-level feature matching
- Impossible to disambiguate outlier correspondences in image pairs
- Results in erroneous relative motion relationships
- Sparse errors are easy to identify

Figure from Zach *et al.* "Disambiguating Visual Relations Using Loop Constraints"

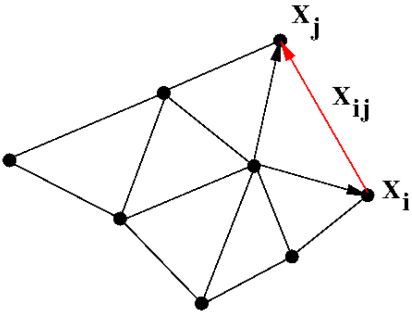
Robustness



Errors in viewgraph relations

- Multiple sources of errors in low-level feature matching
- Impossible to disambiguate outlier correspondences in image pairs
- Results in erroneous relative motion relationships
- Sparse errors are easy to identify

Robustness

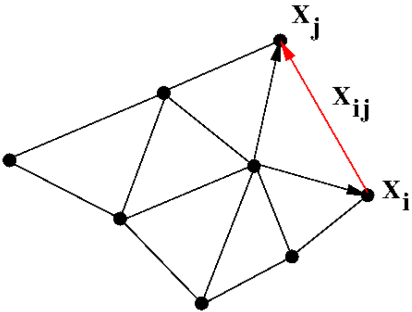


$$\boldsymbol{M}_j \boldsymbol{M}_i^{-1} \neq \boldsymbol{M}_{ij}$$

Errors in relative relationships

- Individual observations can be highly erroneous

Robustness



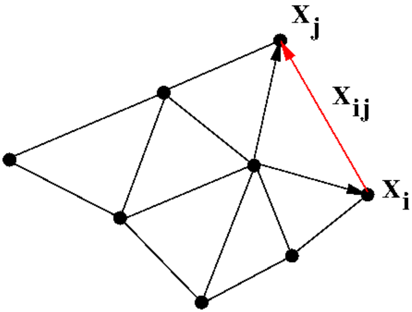
$$\boldsymbol{M}_j \boldsymbol{M}_i^{-1} \neq \boldsymbol{M}_{ij}$$

Recall $\mathfrak{v}_j - \mathfrak{v}_i = \mathfrak{v}_{ij}$

Errors in relative relationships

- Individual observations can be highly erroneous
- Observations inconsistent with other edges

Robustness



$$\boldsymbol{M}_j \boldsymbol{M}_i^{-1} \neq \boldsymbol{M}_{ij}$$

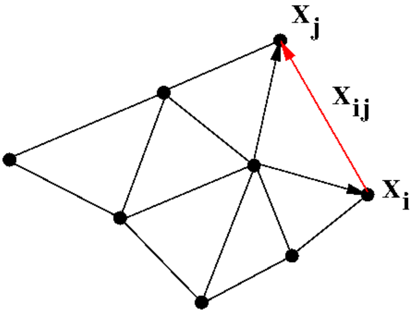
Recall $\mathfrak{v}_j - \mathfrak{v}_i = \mathfrak{v}_{ij}$

$$\boldsymbol{A}\mathfrak{V} = \mathbb{V}_{ij}$$

Errors in relative relationships

- Individual observations can be highly erroneous
- Observations inconsistent with other edges
- Will contribute outlier equation in Lie algebra representation

Robustness



$$M_j M_i^{-1} \neq M_{ij}$$

$$\text{Recall } \mathfrak{v}_j - \mathfrak{v}_i = \mathfrak{v}_{ij}$$

$$A\mathfrak{V} = \mathbb{V}_{ij}$$

Errors in relative relationships

- Individual observations can be highly erroneous
- Observations inconsistent with other edges
- Will contribute outlier equation in Lie algebra representation
- Least squares solution is non-robust

Robustness

Robustness in Computer Vision

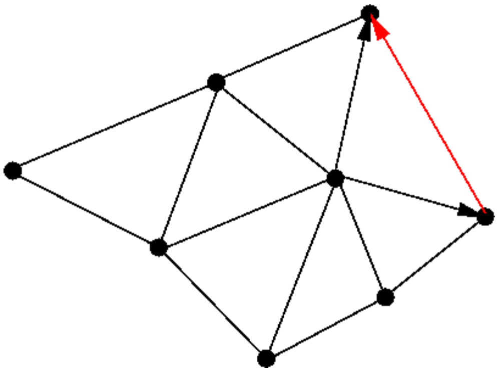
- Outliers is a common issue in vision problems
- Two broad approaches for robustness
  - Identify outliers and remove them (RANSAC etc.)
  - Estimate in the presence of outliers (M-estimators)
- We will consider robust solutions of both types

Robustness

Outliers in Motion Averaging

- Consider viewgraph on $\mathbb{SO}(3)$
- Two approaches to identifying outliers

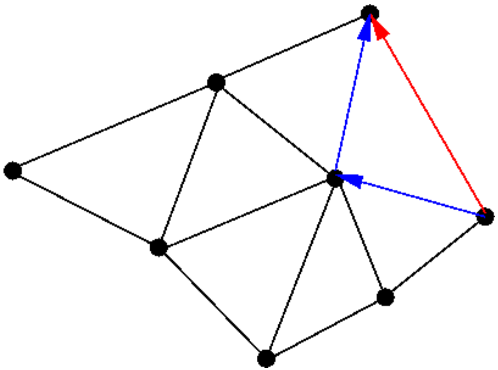
Robustness



Outliers in Motion Averaging

- Consider viewgraph on $\mathbb{SO}(3)$
- Two approaches to identifying outliers
  - Outliers inconsistent with alternative paths

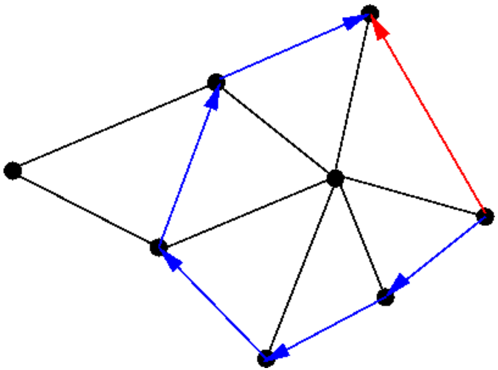
Robustness



Outliers in Motion Averaging

- Consider viewgraph on $\mathbb{SO}(3)$
- Two approaches to identifying outliers
  - Outliers inconsistent with alternative paths

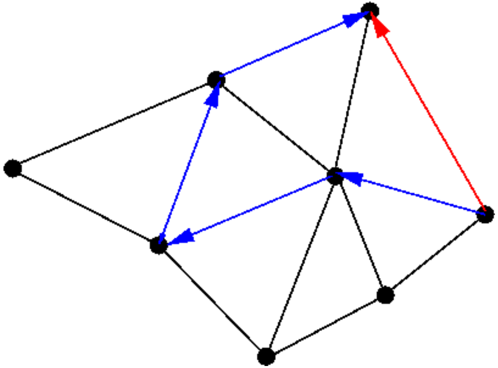
Robustness



Outliers in Motion Averaging

- Consider viewgraph on $\mathbb{SO}(3)$
- Two approaches to identifying outliers
  - Outliers inconsistent with alternative paths

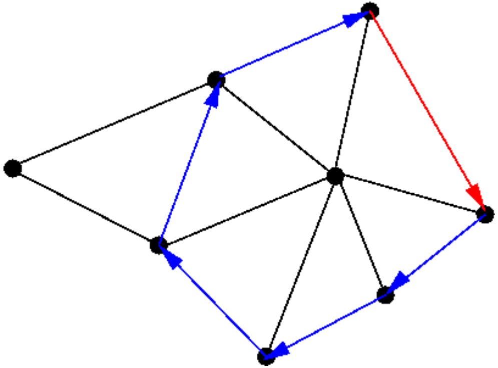
Robustness



Outliers in Motion Averaging

- Consider viewgraph on $\mathbb{SO}(3)$
- Two approaches to identifying outliers
    - Outliers inconsistent with alternative paths

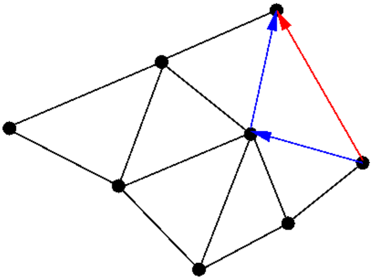
Robustness



Outliers in Motion Averaging

- Consider viewgraph on $\mathbb{SO}(3)$
- Two approaches to identifying outliers
  - Outliers inconsistent with alternative paths
  - Loops of transformation should yield identity
  - Composition far from identity $\implies$ outliers present

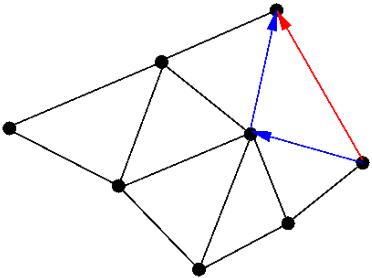
Robustness



Outliers in Motion Averaging

- Identify outliers by comparing with alternate path
- Need to generate alternate path (compositions)
- Need to do this for all edges

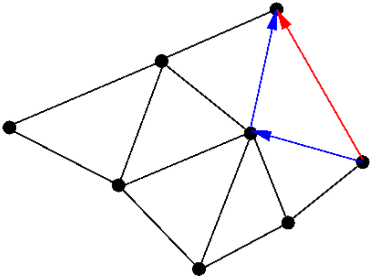
Robustness



Outliers in Motion Averaging

- Identify outliers by comparing with alternate path
- Need to generate alternate path (compositions)
- Need to do this for all edges
- Solved using RANSAC
    - RANSAC is very popular in computer vision
    - Empirical robust fitting

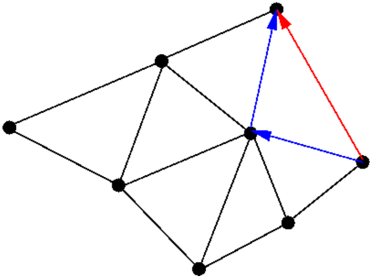
Robustness



RANSAC for Motion Averaging

- Identify outliers using RANSAC
- Minimal sample for a viewgraph ?

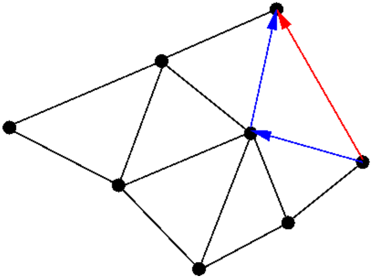
Robustness



RANSAC for Motion Averaging

- Identify outliers using RANSAC
- Minimal sample for a viewgraph ? Spanning Tree

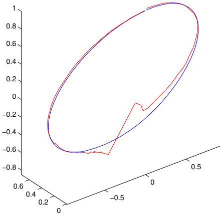
Robustness

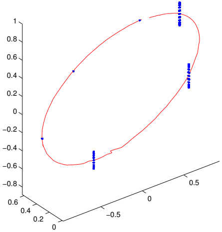


RANSAC for Motion Averaging

- Identify outliers using RANSAC
- Minimal sample for a viewgraph ? Spanning Tree
- Sample from set of spanning trees
- Score each spanning tree
- Select winner with max inliers
- Classify inliers/outliers

Robustness

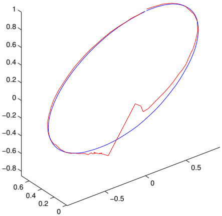


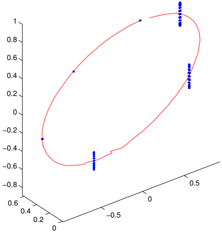


MOVI Image Sequence

- 118 images of a sequence
- 2209 relative geometries computed
- $\boldsymbol{T} = 10000$ samples used
- Sliding window of 10 images
- Shift of 5 images
- 1130 inliers survive the tests

Robustness

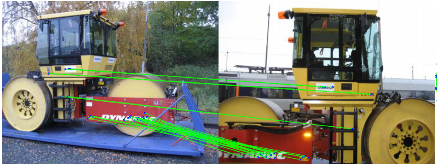


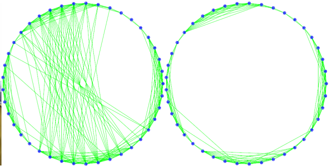


Limitations of RANSAC approach

- Very large number of putative spanning trees
- "Clean" spanning trees very unlikely
- Solution:
  - Can use fitting error to bias sampling technique
  - Careful tuning of sampling leads to significant improvement
  - Tame complexity by heuristics and sampling

Robustness





Paper: "Non-sequential SfM"

- Uses consistency of rotation estimates
- Tests greedy spanning tree + edges for loop consistency
- Grow set of inlier edge using consistency test
- Variety of simplifying heuristics leads to robust results
- Similar approach
    - Bourmaud *et al.* "Global Motion Estimation from Relative Measurements ..." 2014
    - ST + Kalman filtering to grow edge set

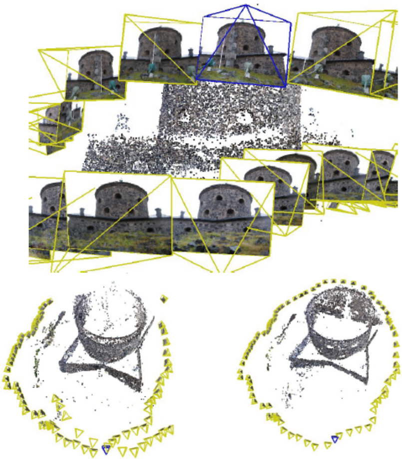
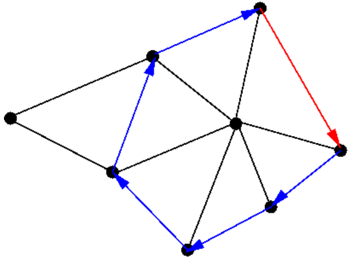
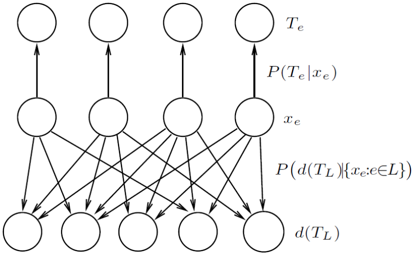


Figure 5. CASTLE. The top and left figures shows the 3D result from BUNDLER and on the right, our result is given. By careful inspection, one can see that the top and bottom image rows of the top figure display *different* facades of the castle. (A window is blocked by stairs in the top row.) This confusion of facades yields an incomplete and false reconstruction. Using rotational consistency, a complete trajectory is obtained.

Robustness



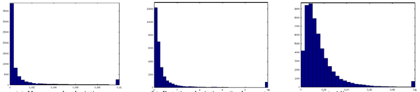


Outlier detection using loop statistics

- Argument: RANSAC inappropriate for large hypothesis space
- Proposed remedy
    - Loop statistics are informative
    - Amenable to tractable Bayesian inference

Zach *et al.* "Disambiguating Visual Relationships using Loop Constraints"

Robustness



(a) Homography deviations (b) Rotation deviations (in degrees) (c) Alignment errors

Figure 4. Histograms of empirical deviations from the respective identity map. The inlier portion of the histogram roughly follows an exponential distribution.

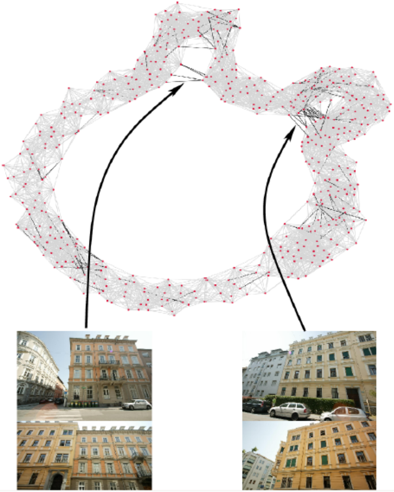
Outlier detection using loop statistics

- Latent binary variable $x_e \in \{0, 1\}$
- Composition of Transformations on Loop Cycle : T_L
- $P(d(T_L)|x_L = 0)$: No outliers $\implies$ noise model
- $P(d(T_L)|x_L = 1)$: At least one outlier $\implies$ uniform prior
- Solve maximization of joint probability of $P(x_e|d(T_L))$
- Loopy belief propagation
- Convex relaxation gives alternative of BnB
- Reduce search space using cycles of max length of 6 (min 3)

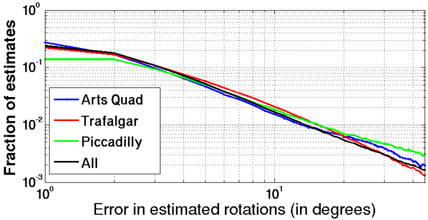
Zach *et al.* “Disambiguating Visual Relationships using Loop Constraints”

(a) W/o edge filtering (143 views registered) (b) With edge filtering (all 189 views registered)

Figure 6. Model generated by Bundler for a facade with highly repetitive elements (a) without using epipolar graph filtering, and (b) with epipolar filtering solely using relative rotations.



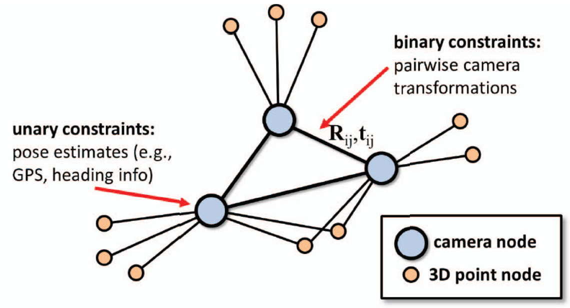
Robustness



Estimation with Outliers

- Outlier classification assumes separability
- Real-world data does not show clear separation of inlier/outliers
- No good choice of classification threshold
- Preferable to estimate in the presence of outliers
- Need for robust estimation directly on Lie groups ($\mathbb{SO}(3)$)
- Will consider three solutions
  - DISCO/MRF solution by Crandall *et al.* 2011
  - Weiszfeld method by Hartley *et al.* 2011
  - IRLS approach by Chatterjee & Govindu 2013, 2017

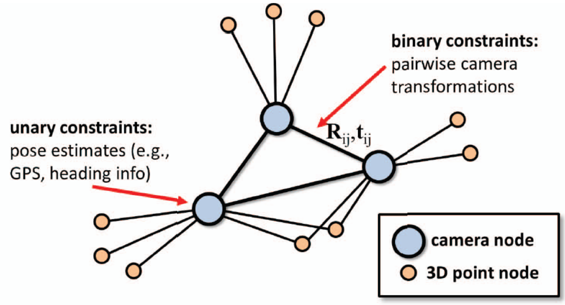
Robustness



Paper: Discrete-Continuous .../SfM with MRF

- Addresses robustness in averaging of rotations and translations
- Also adds constraint penalties for absolute measurements
- Treats robust averaging as a large cost minimization problem

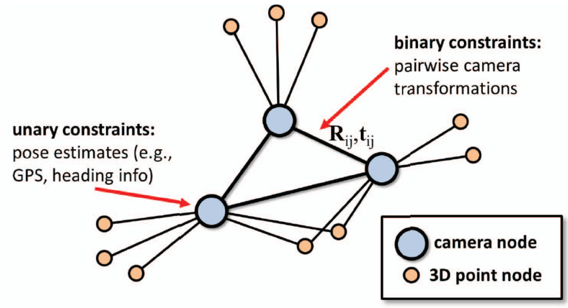
Robustness



Robust rotation averaging cost function

Original cost function: $\sum_{\mathcal{E}} d^2(\boldsymbol{R}_{ij}, \boldsymbol{R}_j \boldsymbol{R}_i^{-1})$

Robustness

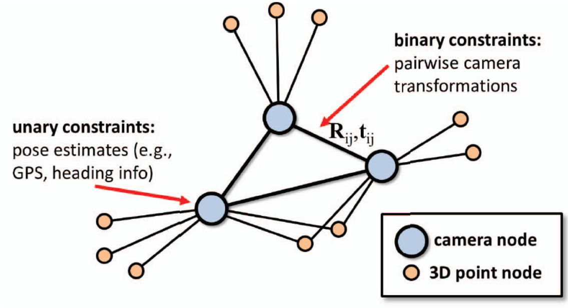


Robust rotation averaging cost function

$$\text{Original cost function: } \sum_{\mathcal{E}} d^2(\boldsymbol{R}_{ij}, \boldsymbol{R}_j \boldsymbol{R}_i^{-1})$$

$$\text{Robust version: } d_R(\boldsymbol{R}_a, \boldsymbol{R}_b) = \rho_R(||\boldsymbol{R}_a - \boldsymbol{R}_b||)$$

Robustness



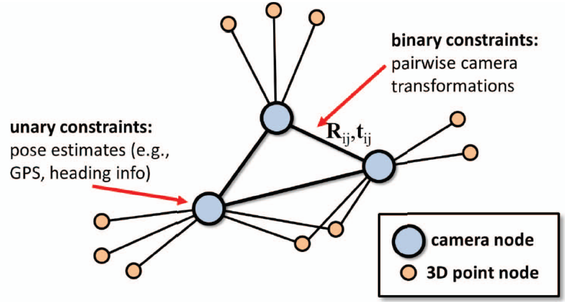
Robust rotation averaging cost function

$$\text{Original cost function: } \sum_{\mathcal{E}} d^2(\boldsymbol{R}_{ij}, \boldsymbol{R}_j \boldsymbol{R}_i^{-1})$$

$$\text{Robust version: } d_R(\boldsymbol{R}_a, \boldsymbol{R}_b) = \rho_R(||\boldsymbol{R}_a - \boldsymbol{R}_b||)$$

$$\text{Add absolute constraints: } d(\boldsymbol{R}_a)$$

Robustness



Robust rotation averaging cost function

$$\text{Original cost function: } \sum_{\mathcal{E}} d^2(\boldsymbol{R}_{ij}, \boldsymbol{R}_j \boldsymbol{R}_i^{-1})$$

$$\text{Robust version: } d_R(\boldsymbol{R}_a, \boldsymbol{R}_b) = \rho_R(||\boldsymbol{R}_a - \boldsymbol{R}_b||)$$

$$\text{Add absolute constraints: } d(\boldsymbol{R}_a)$$

$$\text{Solve: } \min_{\boldsymbol{R}_1, \cdots, \boldsymbol{R}_N} \sum_{\mathcal{E}} d_R^2(\boldsymbol{R}_{ij}, \boldsymbol{R}_j \boldsymbol{R}_i^{-1}) + \sum_{i \in \mathcal{I}} d(\boldsymbol{R}_i)$$

Robustness

Solve:

$$\min_{\boldsymbol{R}_1,\cdots,\boldsymbol{R}_N} \sum_{\mathcal{E}} d_R^2(\boldsymbol{R}_{ij}, \boldsymbol{R}_j\boldsymbol{R}_i^{-1}) + \sum_{i\in\mathcal{I}} d(\boldsymbol{R}_i)$$

Paper: Discrete-Continuous .../SfM with MRF

- Robust cost function to be optimized
- Note the form of $d_R(\boldsymbol{R}_a, \boldsymbol{R}_b) = \rho_R(||\boldsymbol{R}_a - \boldsymbol{R}_b||)$ (extrinsic)
- Discretizes the representation space of $\mathbb{SO}(3)$ and $\mathbb{R}^3$
- Individual labels assigned for quantized “cells” in $\mathbb{SO}(3)$
- Rotation estimation is a label assignment problem
- Solved using large-scale Bayesian inference (discrete BP on MRF)
- Solution refined using continuous optimization (hence DISCO)
- Major approximations involved
  - Assume cameras have no twist (in-plane rotation)
  - Coarse quantization of $\mathbb{SO}(3)$ into 2D labels
- Same principle used for estimation of 3D translation

Dataset	Images matched	Largest component size ($\|V\|$)	Camera-camera edges ($\|E_C\|$)	Camera-point edges ($\|E_P\|$)	% images geotagged	Scene size (km^2)	Reconstructed images
Acropolis	2,961	463	22,842	42,255	100.0%	0.1×0.1	454
Quad	6,514	5,520	444,064	551,670	77.2%	0.4×0.3	5,233
Dubrovnik	12,092	6,854	1,000,178	835,310	56.7%	1.0×0.5	6,532
CentralRome	74,394	15,242	864,758	1,393,658	100.0%	1.5×0.8	14,754
SanFrancisco	17,357	7,866	203,024	515,100	100.0%	1.0×0.4	5,197

Fig. 5. *Sample reconstructions* for (clockwise from top left) Acropolis, Dubrovnik, Quad, and CentralRome.

Quad

Reconstructed images: 5,233

Edges in MRF: 995,734

Robustness

TABLE 2

Comparison with Incremental BA in terms of median differences for point positions and camera poses.

Dataset	Rotational difference: Our approach BP	Our approach NLLS	Our approach Final BA	Linear approach [14] Linear	Linear approach [14] NLLS	Translational difference: Geotags	Our approach BP	Our approach NLLS	Our approach Final BA	Point difference: Our approach Final BA
Acropolis	14.1°	1.5°	0.2°	1.6°	1.6°	12.9m	8.1m	2.4m	0.1m	0.2m
Quad	4.7°	4.6°	0.2°	41°	41°	15.5m	16.6m	14.2m	0.6m	0.5m
Dubrovnik	9.1°	4.9°	0.1°	11°	6°	127.6m	25.7m	15.1m	1.0m	0.9m
CentralRome	6.2°	3.3°	1.3°	27°	25°	413.0m	27.3m	27.7m	25.0m	24.5m

TABLE 4

Running times of our approach compared to incremental bundle adjustment.

Dataset	Our approach: Rot BP	Rot NLLS	Trans BP	Trans NLLS	Bund Adj	Total	Incremental BA
Acropolis	50s	16s	7m 24s	49s	5m 36s	0.2 hours	0.5 hours
Quad	40m 57s	8m 46s	53m 51s	40m 22s	5h 18m 00s	7.7 hours	62 hours
Dubrovnik	28m 19s	8m 28s	29m 27s	7m 22s	4h 15m 57s	5.5 hours	28 hours
CentralRome	1h 8m 24s	40m 0s	2h 56m 36s	1h 7m 51s	7h 20m 00s	13.2 hours	82 hours

Paper: Discrete-Continuous .../SfM with MRF

- Overall pipeline results in significant speedup
- Impressive quality of results
- Discretization leads to large-scale inference problem
- Discrete inference is computationally very expensive
- Completely ignores group structure of $\mathbb{SO}(3)$
- Will compare results of robust rotation averaging alone later

Robustness

$$\text{Original cost: } \sum_{\mathcal{E}} d^2(\boldsymbol{R}_{ij}, \boldsymbol{R}_j \boldsymbol{R}_i^{-1})$$

$$\text{Robust cost: } \sum_{\mathcal{E}} d(\boldsymbol{R}_{ij}, \boldsymbol{R}_j \boldsymbol{R}_i^{-1})$$

Robust intrinsic methods for averaging

- ℓ_2 methods are non-robust (unbounded influence function)
- Discretization and extrinsic methods ignore geometry
- Alternate robust norm is L_1 mean ($\neq \ell_1$)
- Geometric median has no closed form solution
- Iterative solution is the Weiszfeld algorithm
- Approach taken by Hartley *et al.*
- Leads to a distributed rotation averaging approach
- First, we consider the Weiszfeld algorithm

Given set of observations $\boldsymbol{x}_k \in \mathbb{R}^n$
Geometric distance minimiser is

$$\begin{aligned}\mu &= \arg\min_{\boldsymbol{y}} \sum_k d(\boldsymbol{x}_k, \boldsymbol{y}) \\ &= \arg\min_{\boldsymbol{y}} \sum_k ||\boldsymbol{x}_k - \boldsymbol{y}||\end{aligned}$$

Gradient of cost function is

$$\nabla = -\sum_k \frac{\boldsymbol{x}_k - \boldsymbol{y}}{||\boldsymbol{x}_k - \boldsymbol{y}||}$$

Gradient is sum of unit vectors towards $\boldsymbol{y}$

Robustness

Gradient of cost function is

$$\nabla = -\sum_k \frac{\boldsymbol{x}_k - \boldsymbol{y}}{||\boldsymbol{x}_k - \boldsymbol{y}||}$$

The gradient descent step becomes

$$\boldsymbol{y} \leftarrow \boldsymbol{y} + \lambda \sum_k \frac{\boldsymbol{x}_k - \boldsymbol{y}}{||\boldsymbol{x}_k - \boldsymbol{y}||}$$

Weiszfeld uses a specific step size

$$\lambda = \sum_k \frac{1}{||\boldsymbol{x}_k - \boldsymbol{y}||}$$

Robust Methods

For $\lambda = \sum_k \frac{1}{||\boldsymbol{x}_k - \boldsymbol{y}||}$ the iteration now becomes

$$\begin{aligned} \boldsymbol{y} &\leftarrow \boldsymbol{y} + \lambda \sum_k \frac{\boldsymbol{x}_k - \boldsymbol{y}}{||\boldsymbol{x}_k - \boldsymbol{y}||} \\ &\leftarrow \boldsymbol{y} + \sum_k \frac{1}{||\boldsymbol{x}_k - \boldsymbol{y}||} . \sum_k \frac{\boldsymbol{x}_k - \boldsymbol{y}}{||\boldsymbol{x}_k - \boldsymbol{y}||} \end{aligned}$$

This yields the update

$$\boldsymbol{y} \leftarrow \boldsymbol{y} + \frac{\sum_k (\boldsymbol{x}_k - \boldsymbol{y}) / ||\boldsymbol{x}_k - \boldsymbol{y}||}{\sum_k ||\boldsymbol{x}_k - \boldsymbol{y}||}$$

Weiszfeld method

- Geometric median in n-dimensions
- Guaranteed to converge as long as $\boldsymbol{y}$ avoids $\boldsymbol{x}_k$
- Method is rather slow in convergence

Theory and Formulation

Algorithm for Intrinsic Average on $\mathbb{SO}(3)$

Input : $\{\boldsymbol{R}_1, \cdots, \boldsymbol{R}_N\} \in \mathbb{SO}(3)$ (Matrix Group)
Output : $\boldsymbol{\mu} \in \mathbb{SO}(3)$ (Intrinsic Average)
Initialise : $\boldsymbol{\mu} = \boldsymbol{I}$ (Identity)

Do

$\Delta\boldsymbol{R}_i = \boldsymbol{\mu}^{-1}\boldsymbol{R}_i$

$[\Delta\omega_i]_\times = log(\Delta\boldsymbol{R}_i)$

$\Delta\boldsymbol{\mu} = exp(\frac{1}{N}\sum_{i=1}^{N}[\Delta\omega_i]_\times)$

$\boldsymbol{\mu} = \boldsymbol{\mu}\Delta\boldsymbol{\mu}$

Repeat till $||\Delta\boldsymbol{\mu}|| < \epsilon$

Using Geometric Median in $\mathfrak{so}(3)$

- Recall intrinsic averaging involves arithmetic mean in $\mathfrak{so}(3)$
- This is a non-robust estimator
- Replace this step with the Geometric Median in $\mathfrak{so}(3)$
- Results in a robust average estimate of $\{\boldsymbol{R}_1, \cdots, \boldsymbol{R}_N\}$

Robustness

$$\text{Robust cost: } \sum_{\mathcal{E}} d(\boldsymbol{R}_{ij}, \boldsymbol{R}_j \boldsymbol{R}_i^{-1})$$

Weiszeld method for Rotation Averaging

- Weiszfeld method = robust average of $\{\boldsymbol{R}_1, \cdots, \boldsymbol{R}_N\}$
- Need joint solution for all cameras using relative $\boldsymbol{R}_{ij}$'s
- Proposed method uses two nested iterations:
    - Initialize all camera (vertices) rotations in $\mathbb{SO}(3)$
    - Select camera vertex j
    - Update $\boldsymbol{R}_j \leftarrow$ Weiszfeld $\mathbb{SO}(3)$ median of $\{\boldsymbol{R}_{ij}\boldsymbol{R}_i | \forall i \in \mathcal{N}(j)\}$
    - This is a distributed robust average
    - Cycle through all vertices
    - Repeat updates till convergence

Robustness

Robust L_1 cost: $\sum_{\mathcal{E}} d(\boldsymbol{R}_{ij}, \boldsymbol{R}_j \boldsymbol{R}_i^{-1})$

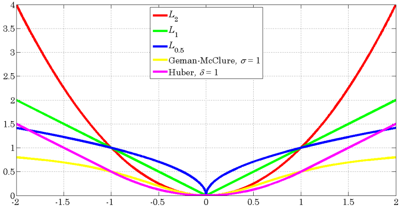
Properties of Weiszeld method for Rotation Averaging

- Intrinsic estimator on $\mathbb{SO}(3)$
- Convergence can be **very** slow due to distributed averaging
- Convergence rate related to spectral properties of viewgraph
- Can get stuck in poor minima due to initialization + distributed estimation

Robustness

General robust cost:

$$\sum_{\mathcal{E}} \rho(d(\boldsymbol{R}_{ij}, \boldsymbol{R}_j \boldsymbol{R}_i^{-1}))$$



Robust Rotation Averaging

- Need to generalize to any loss function
- Better convergence than Weiszfeld
- Can be achieved by a simple modification of intrinsic approach
- Achieves state-of-the-art

Robustness

Algorithm for Motion Averaging on Lie Groups

Input : $\{\boldsymbol{M}_{ij1}, \boldsymbol{M}_{ij2} \cdots, \boldsymbol{M}_{ijn}\}$
Output : $\boldsymbol{M}_g : \{\boldsymbol{M}_2, \cdots, \boldsymbol{M}_N\}$
Set $\boldsymbol{M}_g$ to an initial guess (Linear solution)

Do

$\Delta\boldsymbol{M}_{ij} = {\boldsymbol{M}_j}^{-1}\boldsymbol{M}_{ij}\boldsymbol{M}_i$

$\Delta\mathfrak{m}_{ij} = log(\Delta\boldsymbol{M}_{ij})$

$\Delta\mathfrak{v}_{ij} = vec(\mathfrak{m}_{ij})$

Solve $\boldsymbol{A}\Delta\mathfrak{V} = \Delta\mathbb{V}_{ij}$

$\forall k \in [2, N], \boldsymbol{M}_k = \boldsymbol{M}_k exp(\Delta\mathfrak{v}_k)$

Repeat till $||\Delta\mathfrak{V}|| < \epsilon$

Robust estimation in $\mathfrak{so}(3)$

- We now need a robust solution for problem $\boldsymbol{A}\Delta\mathfrak{V} = \Delta\mathbb{V}_{ij}$
- Our solution: Replace least squares with robust estimator
- Rest of the algorithm remains the same

Robustness

M-estimators

- Introduced in statistics by Huber in 1964
- Robust equivalent to the least squares solutions
- Generalises maximum likelihood estimator with tolerance for data contamination
- Vast body of literature on this topic

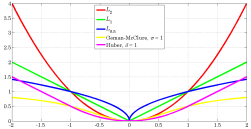
Robust Methods

M-estimators

$$\theta = \arg\min_{\theta} \frac{1}{n} \sum_{i=1}^{n} \rho\left(r_i(\boldsymbol{x}_i, \theta); \sigma\right)$$

- σ is a tuning parameter
- σ is a function of noise-level
- Loss function $\rho(u)$ satisfies
    - Non-negative with $\rho(0) = 0$
    - Even symmetric $\rho(u) = \rho(-u)$
    - Non-decreasing with $|u|$
    - Usual least-squares with $\rho(u) = u^2$
- Minimisation problem exists

Robustness



$$\theta = \arg\min_{\theta} \frac{1}{n} \sum_{i=1}^{n} \rho\left(r_i(\boldsymbol{x}_i, \theta); \sigma\right); \rho(x) = \frac{x^2}{x^2 + \sigma^2}$$

Behaviour

- Close to 0, behaves like least-squares
- Further away from 0, cost tapers off
- Large deviations have a fixed cost
- Reduce the influence of large deviants
- Robustness dependent on *breakdown* point
- One good solution is IRLS

Robustness

Iteratively Reweighted Least Squares (IRLS)

- Define $\boldsymbol{e} = \boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}$
- Least squares solution minimizes $\boldsymbol{e}^T\boldsymbol{e}$
- Robust modification : use robust loss function ρ
- Example : $\rho(x) = \frac{x^2}{x^2+\sigma^2}$

Minimize $E = \sum_i \rho(||\boldsymbol{e}_i||)$

Robustness

Iteratively Reweighted Least Squares (IRLS)

Minimize $E = \sum_i \rho(||\boldsymbol{e}_i||)$

$$\begin{aligned}
\min_{\boldsymbol{x}} E = \min_{\boldsymbol{x}} \sum_i \rho(||\boldsymbol{e}_i||) &= \min_{\boldsymbol{x}} \sum_i \frac{\boldsymbol{e}_i^2}{\boldsymbol{e}_i^2 + \sigma^2} \\
\Rightarrow \frac{\partial E}{\partial \boldsymbol{x}} &= \frac{\partial E}{\partial \boldsymbol{e}} \frac{\partial \boldsymbol{e}}{\partial \boldsymbol{x}} = 0 \\
\Rightarrow \boldsymbol{A}^T \boldsymbol{\Phi}(\boldsymbol{e}) \boldsymbol{A} \boldsymbol{x} &= \boldsymbol{A}^T \boldsymbol{\Phi}(\boldsymbol{e}) \boldsymbol{b}
\end{aligned}$$

where $\boldsymbol{\Phi}(\boldsymbol{e})$ is a diagonal matrix with $\boldsymbol{\Phi}(i,i) = \frac{\sigma^2}{(\boldsymbol{e}_i^2 + \sigma^2)^2}$

Solution is $(\boldsymbol{A}^T \boldsymbol{\Phi} \boldsymbol{A})^{-1} \boldsymbol{A}^T \boldsymbol{\Phi} \boldsymbol{b}$

Leads to a greedy iterative solution

- Set $\boldsymbol{x}$ to initial guess
- While $||\boldsymbol{x} - \boldsymbol{x}_{prev}|| > \epsilon$
  - $\boldsymbol{x}_{prev} \leftarrow \boldsymbol{x}$
  - $\boldsymbol{e} \leftarrow \boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}$
  - $\boldsymbol{\Phi} \leftarrow \boldsymbol{\Phi}(\boldsymbol{e})$
  - $\boldsymbol{x} \leftarrow \left(\boldsymbol{A}^T \boldsymbol{\Phi} \boldsymbol{A}\right)^{-1} \boldsymbol{A}^T \boldsymbol{\Phi} \boldsymbol{b}$
- EndWhile

- Greedy method
- Converges to fixed point
- Does well if $\boldsymbol{x}_{init}$ is reasonable
- Initial weights $\Phi(\boldsymbol{x}_{init})$ determines convergence

Robustness

$$\text{General robust cost: } \sum_{\mathcal{E}} \rho(d(\boldsymbol{R}_{ij}, \boldsymbol{R}_j \boldsymbol{R}_i^{-1}))$$

$$\text{Equivalent problem: } \sum_{\mathcal{E}} \rho\left(\left\|\omega(\Delta\boldsymbol{R}_j^{-1} \Delta\boldsymbol{R}_{ij} \Delta\boldsymbol{R}_i)\right\|\right)$$

$$\text{Leads to: } \min_{\Delta\Omega_{\mathcal{V}}} \sum_{\mathcal{E}} \phi_{ij} \|\boldsymbol{r}_{ij}(\Delta\Omega_{\mathcal{V}})\|^2$$

Iterative Reweighted NLSQ Problem

- Define $\Delta\boldsymbol{R}_{ij} = \boldsymbol{R}_j^{-1} \boldsymbol{R}_{ij} \boldsymbol{R}_i$
- Define $\boldsymbol{r}_{ij}(\Delta\Omega_{\mathcal{V}}) = \omega(\boldsymbol{R}(-\Delta\omega_j)\boldsymbol{R}(\Delta\omega_{ij})\boldsymbol{R}(\Delta\omega_i))$
- Weight function $\phi_{ij} = \phi_{ij}(||\boldsymbol{r}_{ij}(\boldsymbol{0})||)$
- Constitutes an iterative reweighted non-linear least squares problem

Robustness

$$\min_{\Delta\Omega_{\mathcal{V}}} \sum_{\mathcal{E}} \phi_{ij} \left\| \boldsymbol{r}_{ij}(\Delta\Omega_{\mathcal{V}}) \right\|^2$$

First-order expansion: $$\min_{\Delta\Omega_{\mathcal{V}}} \sum_{\mathcal{E}} \phi_{ij} \left\| \boldsymbol{r}_{ij}(\mathbf{0}) + \mathbb{J}\boldsymbol{r}_{ij}(\mathbf{0})^T \Delta\Omega_{\mathcal{V}} \right\|^2$$

Yields: $$\sqrt{\phi_{ij}}\mathbb{J}\boldsymbol{r}_{ij}(\mathbf{0})^T \Delta\Omega_{\mathcal{V}} = -\sqrt{\phi_{ij}}\boldsymbol{r}_{ij}(\mathbf{0}) \;\; \forall (i,j) \in \mathcal{E}$$

Iterative Reweighted NLSQ Problem

- Define $\Delta\boldsymbol{R}_{ij} = \boldsymbol{R}_j^{-1}\boldsymbol{R}_{ij}\boldsymbol{R}_i$
- Define $\boldsymbol{r}_{ij}(\Delta\Omega_{\mathcal{V}}) = \omega(\boldsymbol{R}(-\Delta\omega_j)\boldsymbol{R}(\Delta\omega_{ij})\boldsymbol{R}(\Delta\omega_i))$
- Weight function $\phi_{ij} = \phi_{ij}(||\boldsymbol{r}_{ij}(\mathbf{0})||)$
- Gives us a Gauss-Newton form to solve

Robustness

$$\mathbb{J}\mathbf{r}_{ij}\left(\mathbf{0}\right) = \alpha_{ij} \underbrace{\left[\ \cdots + \boldsymbol{I} \cdots - \boldsymbol{I} \cdots\ \right]}_{\boldsymbol{A}_{ij}}$$

$$+(1-\alpha_{ij}) \underbrace{\left[\ \cdots + \frac{\Delta\boldsymbol{\omega}_{ij}\Delta\boldsymbol{\omega}_{ij}^{T}}{\theta_{ij}^{2}} \cdots - \frac{\Delta\boldsymbol{\omega}_{ij}\Delta\boldsymbol{\omega}_{ij}^{T}}{\theta_{ij}^{2}} \cdots\ \right]}_{\boldsymbol{B}_{ij}}$$

$$+\frac{1}{2} \underbrace{\left[\ \cdots + [\Delta\boldsymbol{\omega}_{ij}]_{\times} \cdots + [\Delta\boldsymbol{\omega}_{ij}]_{\times} \cdots\ \right]}_{\boldsymbol{C}_{ij}}$$

IRLS Form

- $\theta_{ij} = ||\Delta\omega_{ij}||$, $\alpha_{ij} = \frac{\theta_{ij}}{2} \cot\left(\frac{\theta_{ij}}{2}\right)$
- $\mathbf{A}_{ij}, \mathbf{B}_{ij}, \mathbf{C}_{ij}$ have 3×3 blocks in i and j locations

Robustness

Gauss-Newton form:

$$\sqrt{\phi_{ij}}\left(\alpha_{ij}\boldsymbol{A}_{ij} + (1-\alpha_{ij})\boldsymbol{B}_{ij} + \frac{1}{2}\boldsymbol{C}_{ij}\right)\Delta\Omega_{\mathcal{V}} = \sqrt{\phi_{ij}}\Delta\omega_{ij}$$

Simplify to: $\sqrt{\phi_{ij}}\boldsymbol{A}_{ij}\Delta\Omega_{\mathcal{V}} = \sqrt{\phi_{ij}}\Delta\omega_{ij}$

IRLS Form

- First-order approximation yields a Gauss-Newton form
- In quasi-Newton, approx Hessian with psd form
- Positive semidefiniteness ensures descent direction
- Drop dependent terms $\boldsymbol{B}_{ij}$ and $\boldsymbol{C}_{ij}$
- $\boldsymbol{A}_{ij}$ independent of iteration
- Consists only of terms ± 1
- Depends solely on viewgraph (compute only once)
- Yields significant speedup

Robustness

General robust cost: $\sum_{\mathcal{E}} \rho(d(\boldsymbol{R}_{ij}, \boldsymbol{R}_j \boldsymbol{R}_i^{-1}))$

Equiv. IRLS step: $\Delta\boldsymbol{\Omega}_{\mathcal{V}} = -\left(\boldsymbol{A}^T \Phi \boldsymbol{A}\right)^{-1} \boldsymbol{A}^T \Phi \Delta\boldsymbol{\Omega}_{\mathcal{E}}$

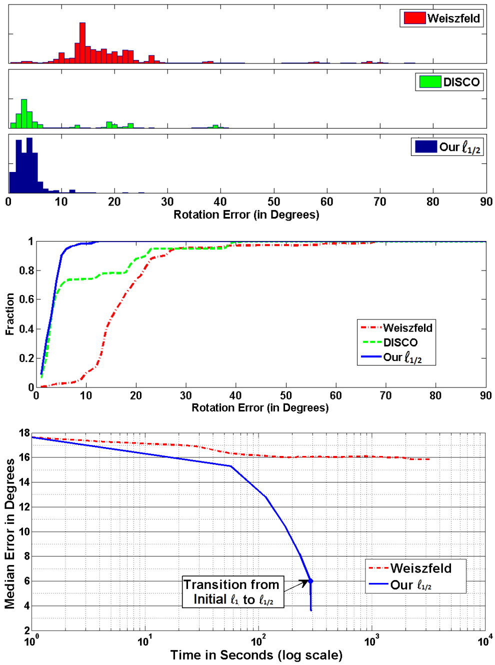
Robust Averaging on $\mathbb{SO}(3)$

- Can use **any** loss function $\rho(.)$
- Equivalent weight function $\phi(.)$
- $\boldsymbol{A}^T \Phi \boldsymbol{A}$ represents weighted Laplacian matrix of $\mathcal{G}$
- Overall minimization approach has a quasi-Newton form (reaches stationary point)
- Initialize IRLS with solution from $\rho = \ell_1$
- Recommended loss function for rotation averaging $\rho = \ell_{\frac{1}{2}}$

Chatterjee & Govindu, 'Robust Relative Rotation Averaging', 2017

Dataset	# Camera	# Edge	# Ground truth	Ground truth fitting error						
				Mean	Median	RMS	% > 10°	% > 30°	% > 60°	% > 90°
Ellis Island (ELS)	247	20297	227	12.50	2.89	27.43	27.5	12.3	4.6	2.0
Piazza Del Popolo (PDP)	354	24710	338	8.38	1.81	21.50	15.3	8.0	3.8	1.6
NYC Library (NYC)	376	20680	332	14.14	4.22	28.57	31.8	13.6	6.0	2.7
Madrid Metropolis (MDR)	394	23784	341	29.30	9.34	51.45	48.8	29.0	16.6	10.4
Yorkminster (YKM)	458	27729	437	11.16	2.68	27.42	19.2	9.6	5.2	3.0
Montreal Notre Dame (MND)	474	52424	450	7.51	1.67	21.26	13.0	6.3	3.0	1.6
Tower of London (TOL)	508	23863	472	11.58	2.60	28.28	19.9	10.3	5.5	3.2
Notre Dame (ND1)	553	103932	553	14.15	2.70	33.48	22.7	12.9	7.6	4.4
Alamo (ALM)	627	97206	577	9.09	2.78	21.73	17.9	7.3	3.3	1.7
Notre Dame (ND2)	715	64678	715	3.58	1.48	8.20	7.8	1.3	0.3	0.1
Vienna Cathedral (VNC)	918	103550	836	11.26	2.59	27.54	20.7	9.4	5.2	2.9
Union Square (USQ)	930	25561	789	9.02	3.61	19.23	22.8	5.8	2.2	1.1
Roman Forum (ROF)	1134	70187	1084	13.83	2.97	31.85	23.7	12.6	7.1	4.1
Piccadilly (PIC)	2508	319257	2152	19.09	4.93	37.40	35.9	19.3	9.9	5.2
Trafalgar (TFG)	5433	680012	5058	8.62	3.01	18.75	21.3	6.5	2.2	0.9
Arts Quad (ARQ)	5530	222044	4978	9.23	2.49	19.76	22.5	8.7	2.7	0.8
San Francisco (SNF)	7866	101512	7866	1.80	0.99	3.93	1.6	0.3	0.1	0.0

Data set	Median Error (degree)					Iterations				Computation Time (second)			
	DISCO			Our				Our				Our	
	BP	BP+ℓ_2	Weiszfeld	Initial ℓ_1	$\ell_{\frac{1}{2}}$	DISCO	Weiszfeld	Initial ℓ_1	$\ell_{\frac{1}{2}}$	DISCO	Weiszfeld	Initial ℓ_1	$\ell_{\frac{1}{2}}$
ELS	5.54	1.82	1.66	1.86	**1.15**	20	154	12	5+19	470	21	1	3
PDP	12.11	5.25	3.35	5.12	**2.62**	20	90	12	5+19	583	17	1	4
NYC	9.12	2.59	2.43	3.03	**1.40**	20	331	10	5+15	446	63	1	3
MDR	12.12	6.64	4.37	5.95	**3.08**	20	149	19	5+19	560	30	2	4
YKM	26.17	2.34	2.73	2.53	**1.62**	20	66	11	5+12	641	16	1	3
MND	6.81	1.03	0.92	1.40	**0.71**	20	37	9	5+11	1608	10	2	10
TOL	10.38	2.89	2.73	3.14	**2.45**	20	136	12	5+17	479	34	1	3
ND1	7.48	1.31	1.04	1.53	**0.98**	20	63	14	5+11	4070	24	5	32
ALM	7.86	4.21	3.57	2.72	**2.14**	20	137	11	5+13	3917	55	4	33
ND2	—	—	0.50	0.76	**0.49**	—	63	7	5+10	—	27	2	13
VNC	22.35	14.57	5.14	5.45	**4.64**	20	405	16	5+17	4085	222	8	41
USQ	26.27	7.50	13.54	5.92	**4.97**	20	498	18	5+53	466	221	4	8
ROF	35.36	13.69	2.11	3.62	**1.70**	20	205	16	5+17	1559	121	8	17
PIC	36.00	14.66	7.65	10.46	**3.12**	20	1055	39	5+28	15604	1635	156	620
TFG	91.02	91.62	13.20	3.03	**2.03**	20	1492	23	5+16	43616	5128	541	1184
ARQ	87.12	88.58	6.95	4.19	**2.54**	20	1271	20	5+20	5227	5707	437	208
SNF	54.38	3.61	15.85	4.35	**3.56**	20	881	11	5+13	1413	3186	632	294



Comparison of our method with DISCO and Weiszfeld method on the San Francisco data set

Extrinsic Methods

$$\text{Averaging cost function: } \sum_{\mathcal{E}} d^2(\boldsymbol{R}_{ij}, \boldsymbol{R}_j \boldsymbol{R}_i^{-1})$$

$$\text{Intrinsic methods: } \{\boldsymbol{R}_1, \cdots, \boldsymbol{R}_N\} \in \mathbb{SO}(3)$$

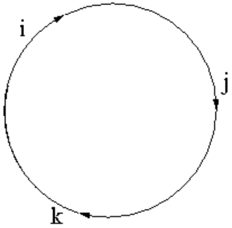
$$\text{Extrinsic methods: } \{\boldsymbol{R}_1, \cdots, \boldsymbol{R}_N\} \in \mathbb{R}^k, \text{ then project onto } \mathbb{SO}(3)$$

Properties of Extrinsic Methods

- We will focus on $\mathbb{SO}(3)$ for concreteness
- **Intrinsic** methods ensure all steps lie in $\mathbb{SO}(3)$
- **Intrinsic** methods explicitly exploit geometry of Lie group
- **Extrinsic** methods solve in $\mathbb{R}^k$ and project on $\mathbb{SO}(3)$
- Some methods use quaternion representation
- Various **relaxations** are available for extrinsic estimation
- **Extrinsic** methods use various tricks for robustness

Extrinsic Methods

×	**1**	**i**	**j**	**k**
1	1	i	j	k
i	i	-1	k	-j
j	j	-k	-1	i
k	k	j	-i	-1



Quaternions

- Introduced by Hamilton, extends complex numbers ($\mathbb{H}$)
- $\boldsymbol{q} = q_0 + q_1\boldsymbol{i} + q_2\boldsymbol{j} + q_3\boldsymbol{k}$
- Constraint : $q_0^2 + q_1^2 + q_2^2 + q_3^2 = 1$
- Quaternions lie on a unit sphere
- Half sphere : Identify $\boldsymbol{q}$ with $-\boldsymbol{q}$
- Rules of multiplication
- Inverse : $\boldsymbol{q}^* = q_0 - q_1\boldsymbol{i} - q_2\boldsymbol{j} - q_3\boldsymbol{k}$
- Identity : $\boldsymbol{q}\boldsymbol{q}^* = \boldsymbol{q}^*\boldsymbol{q} = \{1, 0, 0, 0\}$

Extrinsic Methods

Quaternions

- We consider only unit quaternions $||\boldsymbol{q}|| = 1$
- $\boldsymbol{R}_3 = \boldsymbol{R}_2\boldsymbol{R}_1 \Leftrightarrow \boldsymbol{q}_3 = \boldsymbol{q}_2\boldsymbol{q}_1$
- Quaternion multiplication is not commutative
- Quaternions have Lie group structure, but also $\mathcal{S}^{3+}$
- How do we operate on a vector ?
- Let $\boldsymbol{v} \in \mathbb{H}$
- For vector in $\mathbb{R}^3$, real part is 0 ($\boldsymbol{v}' = \boldsymbol{q}\boldsymbol{v}\boldsymbol{q}^*$)
- Interpolation in graphics (Shoemake)
- Dual quaternions can also handle 3D translations
- **Extrinsic projection** is simple normalization
- $\boldsymbol{q} \leftarrow \frac{\boldsymbol{q}}{||\boldsymbol{q}||}$

Extrinsic Methods

$$\boldsymbol{R}_{ij} = \boldsymbol{R}_j \boldsymbol{R}_i^{-1} \implies \boldsymbol{R}_{ij} \boldsymbol{R}_i = \boldsymbol{R}_j$$

Quaternion form: $\boldsymbol{q}_{ij}\boldsymbol{q}_i = \boldsymbol{q}_j \rightsquigarrow \boldsymbol{Q}_{ij}\boldsymbol{q}_i = \boldsymbol{q}_j$

where $\boldsymbol{Q}(\boldsymbol{q}) = \begin{pmatrix} q_0 & -q_1 & -q_2 & -q_3 \\ q_1 & q_0 & -q_3 & q_2 \\ q_2 & q_3 & q_0 & -q_1 \\ q_3 & -q_2 & q_1 & q_0 \end{pmatrix}$

Extrinsic Averaging using Quaternions

- Move problem to quaternion representation
- Leads to a linear system of equations (with scale constraints)

Extrinsic Methods

$$
\begin{aligned}
\boldsymbol{Q}_{ij}\boldsymbol{q}_i &= \boldsymbol{q}_j \\
\implies [\cdots \boldsymbol{Q}_{ij} \cdots - \boldsymbol{I} \cdots]\boldsymbol{q} &= \boldsymbol{0} \\
\implies \boldsymbol{M}\boldsymbol{q} &= \boldsymbol{0}
\end{aligned}
$$

Extrinsic Averaging using Quaternions (Govindu 2001)

- Stack all rotations into single vector $\boldsymbol{q} = [\boldsymbol{q}_1^T, \cdots, \boldsymbol{q}_N^T]^T$
- Sparse linear estimation problem
- Does not enforce individual norm constraints, i.e. $\boldsymbol{q}_i^T \boldsymbol{q}_i = 1$
- Results in an extrinsic estimation that can be easily projected
- **Caveat:** Need to address equivalence of $\boldsymbol{q}$ and $-\boldsymbol{q}$
- Use approximate solution to disambiguate
- Efficient but less accurate

$$\boldsymbol{Q}_{ij}\boldsymbol{q}_i = \boldsymbol{q}_j \rightsquigarrow \min \sum_{\mathcal{E}} ||\boldsymbol{Q}_{ij}\boldsymbol{q}_i - \boldsymbol{q}_j||^2$$

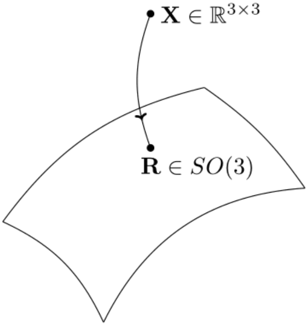
$$\max \sum_{\mathcal{E}} \boldsymbol{q}_j^T \boldsymbol{Q}_{ij} \boldsymbol{q}_i \text{ subject to } \boldsymbol{q}_i^T \boldsymbol{q}_i = 1 \ \forall i \in \{1 \cdots N\}$$

$$\text{Lagrangian form: } ||A\boldsymbol{q} - b||^2 + \sum_{i \in \mathcal{V}} \lambda_i (1 - \boldsymbol{q}_i^T \boldsymbol{q}_i)$$

Extrinsic Averaging using Lagrangian Duality

- Linear version is sparse eigen-value problem
- Linear problem does not enforce quadratic quaternion constraints
- Can be solved as an unconstrained Lagrangian cost function
- Dual form leads to a semi-definite program (SDP)
- Duality gap can be tested for optimality of solution
- IRLS weights can be incorporated for robustness
- Results for small datasets (scalability of SDP is unclear)
- Frederiksson & Olson, 2012

Extrinsic Methods



Optimal Projection

$$\text{Given } \mathbf{X} \in \mathbb{R}^{3\times 3}$$
$$\min_{\boldsymbol{R} \in \mathbb{SO}(3)} ||\boldsymbol{X} - \boldsymbol{R}||_F$$

SVD based solution

$$\boldsymbol{X} = \boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^T$$
$$\implies \boldsymbol{R} = \boldsymbol{U}\mathrm{diag}(1, 1, |\boldsymbol{U}\boldsymbol{V}^T|)\boldsymbol{V}^T$$

Extrinsic Estimation on $\mathbb{SO}(3)$

- Need a means of projection onto the $\mathbb{SO}(3)$ group
- $\mathbb{P} : \mathbb{R}^{3\times 3} \longrightarrow \mathbb{SO}(3)$
- Common distance metric used is the Frobenius norm
- By now, a classical solution provided by Umeyama

Extrinsic Methods

$$\sum_{\mathcal{E}} d^2(\boldsymbol{R}_{ij}, \boldsymbol{R}_j \boldsymbol{R}_i^{-1})$$

$$\text{Relaxed to } \sum_{\mathcal{E}} \left|\left|\boldsymbol{R}_{ij} - \boldsymbol{R}_j \boldsymbol{R}_i^{-1}\right|\right|^2$$

Extrinsic Estimation on $\mathbb{SO}(3)$

- Drop geometric constraints and solve extrinsically
- Work with Frobenius norm (chordal distance) as distance metric
- Solve an estimation problem for $\boldsymbol{X}$
- Project solution $\boldsymbol{X}$ onto $\mathbb{SO}(3)$
- Need to incorporate robustness into estimation of $\boldsymbol{X}$
- A variety of **relaxations** available in the literature
  - Spectral Relaxation
  - Semi-definite Relaxation
  - Rank Relaxation

Following Arrigoni *et al.* "Robust Rotation Synchronization ..."

Extrinsic Methods

$$\boldsymbol{R} = \begin{bmatrix} \boldsymbol{R}_1 \\ \boldsymbol{R}_2 \\ \dots \\ \boldsymbol{R}_N \end{bmatrix}, \boldsymbol{G} = \begin{pmatrix} \boldsymbol{I} & \boldsymbol{R}_{21} & \cdots & \boldsymbol{R}_{N1} \\ \boldsymbol{R}_{12} & \boldsymbol{I} & \cdots & \boldsymbol{R}_{N2} \\ \dots & & & \dots \\ \boldsymbol{R}_{1N} & \boldsymbol{R}_{2N} & \cdots & \boldsymbol{I} \end{pmatrix}$$

Properties of $\mathbb{SO}(3)$ Grammian matrix

- Consider rhs of averaging relationship, i.e. $\boldsymbol{R}_j \boldsymbol{R}_i^{-1}$
- Using orthonormality we get $\boldsymbol{R}_j \boldsymbol{R}_i^T$
- Observation matrix $\boldsymbol{G} = \boldsymbol{R}\boldsymbol{R}^T$ with properties
    - $\text{rank}(\boldsymbol{G}) = 3$
    - $\boldsymbol{G}$ is symmetric and psd
- Can use this to implement various relaxations

Extrinsic Methods

$$\boldsymbol{R} = \begin{bmatrix} \boldsymbol{R}_1 \\ \boldsymbol{R}_2 \\ \cdots \\ \boldsymbol{R}_N \end{bmatrix}, \boldsymbol{G} = \begin{pmatrix} \boldsymbol{I} & \boldsymbol{R}_{21} & \cdots & \boldsymbol{R}_{N1} \\ \boldsymbol{R}_{12} & \boldsymbol{I} & \cdots & \boldsymbol{R}_{N2} \\ \cdots & & & \cdots \\ \boldsymbol{R}_{1N} & \boldsymbol{R}_{2N} & \cdots & \boldsymbol{I} \end{pmatrix}$$

Spectral Relaxation

- In general $\boldsymbol{G} = \boldsymbol{R}\boldsymbol{R}^T$ is rank 3
- Noise in individual $\boldsymbol{R}_{ij}$ entries of $\boldsymbol{G}$
- Estimate $\boldsymbol{R}$ as the three leading eigen vectors of $\boldsymbol{G}$
- Extract individual matrices $\boldsymbol{R}_i$ from $\boldsymbol{R}$
- Enforce orthonormality constraints on estimated $\boldsymbol{R}_i$
- Modify appropriately for $\boldsymbol{R}_{ij}$ entries missing in $\boldsymbol{G}$

Extrinsic Methods

$$\boldsymbol{R} = \begin{bmatrix} \boldsymbol{R}_1 \\ \boldsymbol{R}_2 \\ \cdots \\ \boldsymbol{R}_N \end{bmatrix}, \boldsymbol{G} = \begin{pmatrix} \boldsymbol{I} & \boldsymbol{R}_{21} & \cdots & \boldsymbol{R}_{N1} \\ \boldsymbol{R}_{12} & \boldsymbol{I} & \cdots & \boldsymbol{R}_{N2} \\ \cdots & & & \cdots \\ \boldsymbol{R}_{1N} & \boldsymbol{R}_{2N} & \cdots & \boldsymbol{I} \end{pmatrix}$$

$$\text{Objective function: } \max_{\boldsymbol{X}} \quad trace(\boldsymbol{G}^T \boldsymbol{X})$$

Semi-definite Relaxation

- Recall $\boldsymbol{G} = \boldsymbol{R}\boldsymbol{R}^T$
- Let estimate be $\widehat{\boldsymbol{R}}$ and $\boldsymbol{X} = \widehat{\boldsymbol{R}}\widehat{\boldsymbol{R}}^T$
- $\boldsymbol{X}$ should be decomposable into form $\boldsymbol{R}\boldsymbol{R}^T$
- Implies $\boldsymbol{X}$ should be positive semidefinite (with $\boldsymbol{I}$ on diagonal)

Extrinsic Methods

$$\text{Objective function: } \max_{\boldsymbol{X}} \quad trace(\boldsymbol{G}^T\boldsymbol{X})$$
$$s.t. \qquad \boldsymbol{X} \succeq 0$$
$$\boldsymbol{X}_{ii} = \boldsymbol{I} \qquad 1 \leq i \leq N$$

Semi-definite Relaxation

- $\boldsymbol{X}$ should be decomposable into form $\boldsymbol{R}\boldsymbol{R}^T$
- Implies $\boldsymbol{X}$ should be positive semidefinite (with $\boldsymbol{I}$ on diagonal)
- Gives a semi-definite relaxation of the problem
- Project estimated decomposition onto $\mathbb{SO}(3)$ as before
- Tighter relaxation compared to spectral relaxation
- Alternate relaxation for Least Unsquared Deviation $\sum ||\boldsymbol{R}_{ij} - \boldsymbol{R}_j\boldsymbol{R}_i^{-1}||$

Extrinsic Methods

$$\min_{L,S_1,S_2} ||\mathcal{P}_\Omega(\widehat{X}(\boldsymbol{R})) - L - S_1 - S_2||^2 + \lambda||S_1||_{2,1}$$
$$\text{s. t. } rank(L) \leq 3$$
$$supp(S_1) \subseteq \Omega$$
$$supp(S_2) = \Omega^C$$

Rank Relaxation

- Enforce $\boldsymbol{X}$ to have rank of at most 3
- Becomes a matrix completion problem for incomplete observations in $\boldsymbol{G}$
- Need to also account for outliers (sparse) and noise (small terms)
- This gives us the standard model used $X = L + S + N$
- Enforce 3×3 block structure of sparse outliers S_1
- Solved using various optimization methods
- Efficient algorithms available

Rotation Averaging Methods

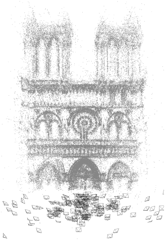
Solve:

$$\min_{\boldsymbol{R}_1,\cdots,\boldsymbol{R}_N \in \mathbb{SO}(3)} \sum_{\mathcal{E}} \rho(d(\boldsymbol{R}_{ij}, \boldsymbol{R}_j \boldsymbol{R}_i^{-1}))$$

Conclusions on Rotation Averaging Methods

- Fundamental distinction between intrinsic and extrinsic
- Extrinsic methods are
  - easier to analyse (only a proxy for original problem)
  - properties and convergence analysis inherited from the nature of relaxation
  - are rather slow and can be memory intensive
  - limited application for smaller datasets
  - limited set of loss functions
- **Intrinsic methods**
  - difficult to establish convergence properties
  - make full use of smoothness of Lie group structure
  - can handle very general forms of loss functions
  - can be solved efficiently
  - Our recommendation is ℓ_1 initialization + IRLS of $\ell_{\frac{1}{2}}$
  - **State-of-the-art : efficient, scalable and robust**

Rotation Averaging



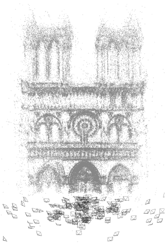
Recall image projection equation

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{I} \mid \mathbf{0} \end{bmatrix} \left[\begin{array}{c|c} \boldsymbol{R} & \boldsymbol{T} \\ \hline \mathbf{0} & 1 \end{array}\right] \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Rotation Averaging in large-scale SfM

- Known $\boldsymbol{R}_i$ linearizes the image projection equation
- Problem is now linear in translation and 3D structure
- Can be solved as a bundle-adjustment problem (SOCP)
- Can also solve for only translation using epipolar relationships
- Results in the **translation averaging** problem

Rotation Averaging



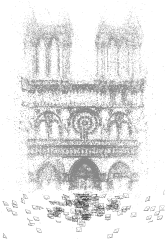
Recall image projection equation

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{I} \mid \mathbf{0} \end{bmatrix} \left[\begin{array}{c|c} \color{red}{\boldsymbol{R}} & \boldsymbol{T} \\ \hline \mathbf{0} & 1 \end{array}\right] \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Rotation Averaging in large-scale SfM

- Known $\boldsymbol{R}_i$ linearizes the image projection equation
- Problem is now linear in translation and 3D structure
- Can be solved as a bundle-adjustment problem (SOCP)
- Can also solve for only translation using epipolar relationships
- Results in the **translation averaging** problem

Rotation Averaging



Recall image projection equation

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{I} \mid \mathbf{0} \end{bmatrix} \left[\begin{array}{c|c} \color{red}{\boldsymbol{R}} & \boldsymbol{T} \\ \hline \mathbf{0} & 1 \end{array}\right] \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

$$\Longrightarrow \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{I} \mid \mathbf{0} \end{bmatrix} \left[\begin{array}{c|c} \boldsymbol{R} & \boldsymbol{I} \\ \hline \mathbf{0} & 1 \end{array}\right] \begin{bmatrix} X \\ Y \\ Z \\ \boldsymbol{T} \end{bmatrix}$$

Rotation Averaging in large-scale SfM

- Known $\boldsymbol{R}_i$ linearizes the image projection equation
- Problem is now linear in translation and 3D structure
- Can be solved as a bundle-adjustment problem (SOCP)
- Can also solve for only translation using epipolar relationships
- Results in the **translation averaging** problem

Rotation Averaging

$$
\begin{aligned}
\rho(\boldsymbol{T}_i, \boldsymbol{S}_j) &= \left\| (u_j^i - \frac{\boldsymbol{R}_i^1 \boldsymbol{S}_j + \boldsymbol{T}_i^1}{\boldsymbol{R}_i^3 \boldsymbol{S}_j + \boldsymbol{T}_i^3}, v_j^i - \frac{\boldsymbol{R}_i^2 \boldsymbol{S}_j + \boldsymbol{T}_i^2}{\boldsymbol{R}_i^3 \boldsymbol{S}_j + \boldsymbol{T}_i^3}) \right\| \\
\implies \min_{\boldsymbol{T}, \boldsymbol{S}, \gamma} \gamma & \\
\text{s. t. } \rho(\boldsymbol{T}_i, \boldsymbol{S}_j) &\leq \gamma, \forall i, j \\
\boldsymbol{R}_i^3 \boldsymbol{S}_j + \boldsymbol{T}_i^3 &\geq 1, \forall i, j \\
\boldsymbol{T}_1 &= (0, 0, 0)
\end{aligned}
$$

Rotation Averaging in large-scale SfM

- Known $\boldsymbol{R}_i$ linearizes the image projection equation
- Problem is now linear in translation and 3D structure
- Denote structure as $\boldsymbol{S} = [X, Y, Z]^T$
- Approach used in Moulon *et al.* "Global Fusion of ..." 2013
- Uses a linear program version for image triplets
- Translation direction from **trifocal** relationships are better

Results from Moulon *et al.*

Scene	Accuracy (mm)					Running times (s)						
	Ours	Bundler [31]	VSfM [35]	Olsson [25]	Arie [3]	Ours	OursP	Bundler [31]	VSfM [35]	Olsson [25]	Ratio [25]/Ours	Ratio [25]/OursP
FountainP11	2.5	7.0	7.6	**2.2**	4.8	12	5	36	3	133	11.1	26
EntryP10	**5.9**	55.1	63.0	6.9	N.A.	16	5	16	3	88	5.5	17
HerzJesusP8	**3.5**	16.4	19.3	3.9	N.A.	6	2	10	2	34	5.6	17
HerzJesusP25	**5.3**	21.5	22.4	5.7	7.8	47	10	100	12	221	4.7	22
CastleP19	**25.6**	344	258	76.2	N.A.	20	6	78	9	99	4.9	16
CastleP30	**21.9**	300	522	66.8	N.A.	55	14	300	18	317	5.7	22

Table 3. Left: Average position error, in millimeters, w.r.t. ground truth for different incremental [31, 35] and global [25, 3] SfM pipelines, given internal calibration. Right: running times in seconds and speed ratio. OursP means our parallel version.

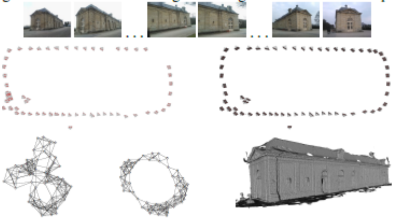


Figure 5. Top: excerpt of the Orangerie dataset. Center: Bundler camera positions (cycle failure), and ours. Bottom: input epipolar graph, our cleaned graph, and mesh obtained from our calibration.

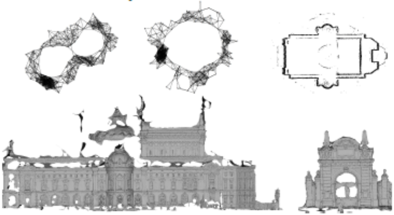


Figure 6. Opera dataset (160 images). Top: input epipolar graph (corrupted by façade symmetries), our cleaned graph, and calibration point cloud. Bottom: orthographic façade and close-up.

Translation Averaging

Recall epipolar relationship

$$
\begin{aligned}
& \boldsymbol{E}_{ij} && = \boldsymbol{R}_{ij}[\boldsymbol{T}_{ij}]_\times \\
\text{where} \quad & \boldsymbol{T}_{ij} && = \boldsymbol{T}_j - \boldsymbol{R}_{ij}\boldsymbol{T}_i \text{ (projective relationship)} \\
\implies \quad & \boldsymbol{t}_{ij} && = \boldsymbol{T}_j - \boldsymbol{R}_{ij}\boldsymbol{T}_i, \quad \boldsymbol{t}_{ij} \in \mathcal{S}^2
\end{aligned}
$$

Translation from Pairwise Relationships

- Above formulation is for specific parametrisation
- Rotate then Translate
- Translate then Rotate gives different (equivalent) relationship
- **Key problem:**
  - Relative translations are scaled
  - Simple geometry of $\mathcal{S}^2$
  - Difficult estimation problem

Translation Averaging

$$
\begin{aligned}
\boldsymbol{t}_{ij} \quad & = \boldsymbol{T}_j - \boldsymbol{R}_{ij}\boldsymbol{T}_i, \quad \boldsymbol{t}_{ij} \in \mathcal{S}^2 \\
\implies \quad & [\boldsymbol{t}_{ij}]_\times [\boldsymbol{T}_j - \boldsymbol{R}_{ij}\boldsymbol{T}_i] = 0 \\
\implies \quad & \Big[\cdots [\boldsymbol{t}_{ij}]_\times \cdots - [\boldsymbol{t}_{ij}]_\times \boldsymbol{R}_{ij} \cdots \Big] \mathbb{T} = \mathbf{0}
\end{aligned}
$$

Linear Averaging of Heading Directions

- Collect all absolute translations into $\mathbb{T} = [\boldsymbol{T}_1; \boldsymbol{T}_2; \cdots; \boldsymbol{T}_N]$
- Results in a sparse linear system of equations to be solved
- Translation recovered upto a single unknown scale factor
- Introduced in Govindu 2001

Translation Averaging

$$\begin{aligned} \boldsymbol{t}_{ij} &= \boldsymbol{T}_j - \boldsymbol{R}_{ij}\boldsymbol{T}_i, \quad \boldsymbol{t}_{ij} \in \mathcal{S}^2 \\ \implies & \left[\boldsymbol{t}_{ij}\right]_\times \left[\boldsymbol{T}_j - \boldsymbol{R}_{ij}\boldsymbol{T}_i\right] = 0 \\ \implies & \left[\cdots \left[\boldsymbol{t}_{ij}\right]_\times \cdots - \left[\boldsymbol{t}_{ij}\right]_\times \boldsymbol{R}_{ij} \cdots\right] \mathbb{T} = \mathbf{0} \end{aligned}$$

Ideally, should use

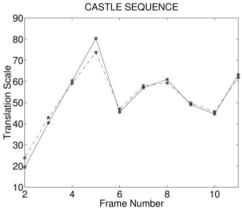
$$\boldsymbol{t}_{ij} = \frac{\boldsymbol{T}_j - \boldsymbol{R}_{ij}\boldsymbol{T}_i}{\|\boldsymbol{T}_j - \boldsymbol{R}_{ij}\boldsymbol{T}_i\|}$$

Problems with Linear Method

- Cross-product results in unequal scaling of equations
- Results in a biased estimate for $\mathbb{T}$
- Scale factors on rhs are unknown
- **Remedy:** Iterative reweighting of system of equations

Translation Averaging

- Set $\lambda_{ij} = 1, \ \forall i, j \in \mathcal{E}$
- Solve
  $[\lambda_{ij} \boldsymbol{t}_{ij}]_\times \, [\boldsymbol{T}_j - \boldsymbol{R}_{ij} \boldsymbol{T}_i] = \boldsymbol{0}$
- Update $\lambda_{ij} \longleftarrow \frac{1}{\|\boldsymbol{T}_j - \boldsymbol{R}_{ij} \boldsymbol{T}_i\|}$
- Repeat till convergence



Reweighted Least Squares Averaging

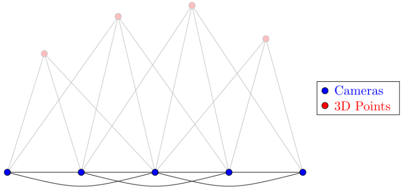
- Iterative reweighting can correct for bias in estimates
- Each edge contributes same amount of information
- Can be shown to converge

Translation Averaging

Limitations of Linear Translation Averaging

- Elimination using cross-product is less informative
- Lack of enforcement of cheirality
- Linear camera motion results in degeneracies
- Need to incorporate robustness into solution
    - Remove outliers, then estimate
    - Estimate in presence of outliers

Translation Averaging

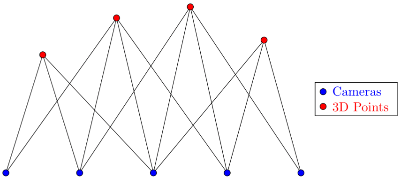


$$
\begin{aligned}
& \boldsymbol{T}_j - \boldsymbol{R}_{ij}\boldsymbol{T}_i \parallel \boldsymbol{t}_{ij} \\
\Rightarrow \quad & \boldsymbol{T}_j - \boldsymbol{R}_j\boldsymbol{R}_i^T\boldsymbol{T}_i \parallel \boldsymbol{t}_{ij} \\
\Rightarrow \quad & \boldsymbol{R}_j^T\boldsymbol{T}_j - \boldsymbol{R}_i^T\boldsymbol{T}_i \parallel \boldsymbol{R}_j^T\boldsymbol{t}_{ij} \\
\Rightarrow \quad & \boldsymbol{C}_i - \boldsymbol{C}_j \parallel \boldsymbol{R}_j^T\boldsymbol{t}_{ij}
\end{aligned}
$$

Camera-camera constraints

- Camera-camera relationship induced by common 3D points
- Viewgraph we have considered till now
- Linear translation averaging has degeneracies
- Fails for collinear translations in 3D

Translation Averaging



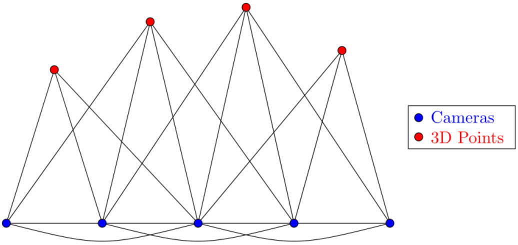
k-th 3D point $\boldsymbol{P}^k$ imaged at $\boldsymbol{p}_i^k$ in i-th camera

$$\begin{aligned} & \boldsymbol{p}_i^k \parallel \boldsymbol{R}_i \boldsymbol{P}^k + \boldsymbol{T}_i \\ \Rightarrow \quad & \boldsymbol{P}^k + \boldsymbol{R}_i^T \boldsymbol{T}_i \parallel \boldsymbol{R}_i^T \boldsymbol{p}_i^k \\ \Rightarrow \quad & \boldsymbol{P}^k - \boldsymbol{C}_i \parallel \boldsymbol{R}_i^T \boldsymbol{p}_i^k \end{aligned}$$

Camera-point relationships

- Assume calibrated camera ($\boldsymbol{K} = \boldsymbol{I}$) for simplicity
- The camera locations and the structure points play similar roles
- Resolves degeneracies due to collinear camera motions

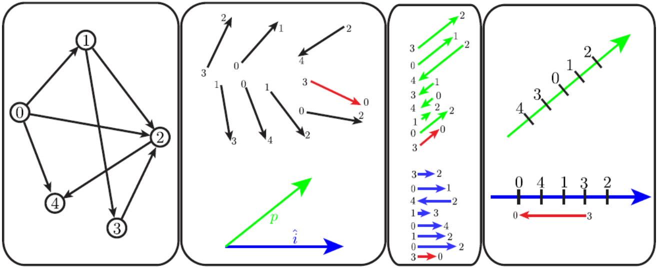
Translation Averaging



Combined camera-camera and camera-point relationships

- Linear method has degeneracies of collinear camera motions
- Can “stabilise” solution by also solving for some 3D points
- Points and camera centres are interchangeable
- Consider both types of relationships
  - camera-camera (epipolar)
  - camera-point (projection)
- Need a scheme to select a few 3D points to solve for efficiency
- Criteria is coverage over the entire camera-camera viewgraph

Translation Averaging



Wilson *et al.* "Robust Global Translation with 1DSfM"

- Identify outliers and remove
- Edge consistency as an embedding problem
- Embedding in 1D $\implies$ ordering constraints
- Find ordering most consistent pairwise constraints
- Solved as Minimum Feedback Arc Set problem

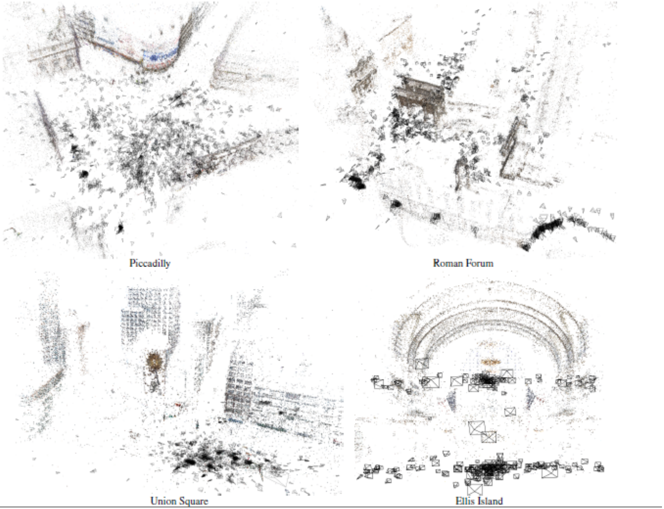
Translation Averaging

$$\min_{\mathbb{T}} \sum_{\mathcal{E}} \left\| \boldsymbol{t}_{ij} - \frac{\boldsymbol{T}_j - \boldsymbol{T}_i}{||\boldsymbol{T}_j - \boldsymbol{T}_i||} \right\|_2^2$$

Wilson *et al.* "Robust Global Translation with 1DSfM"

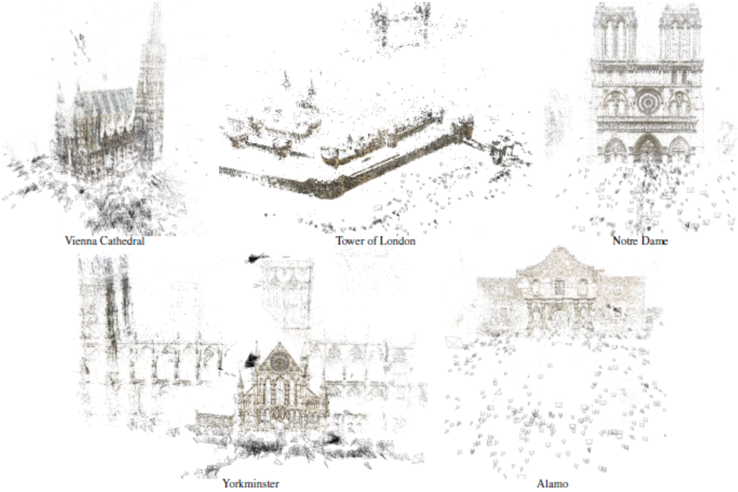
- Use chordal distance as metric to be minimized
- Solve as a non-linear minimization problem (Ceres)
- Huber loss function helps
- Properties
  - In noise-free case, cost function non-decreasing away from $\mathbb{T}_0$
  - Contribution of each edge is the same (unlike other costs)
  - Does not suffer from bias as cross-product does
- 1DSfM+BA improves average error for lesser time

Translation Averaging



Results from Wilson *et al.* "Robust Global Translation with 1DSfM"

Translation Averaging



Results from Wilson *et al.* “Robust Global Translation with 1DSfM”

Translation Averaging

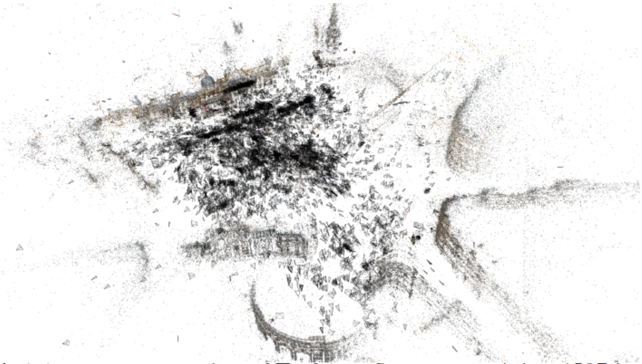


Fig.4. A large reconstruction of Trafalgar Square containing 4597 images.

Result from Wilson *et al.* "Robust Global Translation with 1DSfM"

Translation Averaging

Camera-camera

$$
\begin{aligned}
& \boldsymbol{T}_j - \boldsymbol{R}_{ij}\boldsymbol{T}_i \parallel \boldsymbol{t}_{ij} \\
\Rightarrow\quad & \boldsymbol{T}_j - \boldsymbol{R}_j\boldsymbol{R}_i^T\boldsymbol{T}_i \parallel \boldsymbol{t}_{ij} \\
\Rightarrow\quad & \boldsymbol{R}_j^T\boldsymbol{T}_j - \boldsymbol{R}_i^T\boldsymbol{T}_i \parallel \boldsymbol{R}_j^T\boldsymbol{t}_{ij} \\
\Rightarrow\quad & \boldsymbol{C}_i - \boldsymbol{C}_j \parallel \boldsymbol{R}_j^T\boldsymbol{t}_{ij}
\end{aligned}
$$

Camera-point

k-th 3D point $\boldsymbol{P}^k$
imaged at $\boldsymbol{p}_i^k$ in i-th camera

$$
\begin{aligned}
& \boldsymbol{p}_i^k \parallel \boldsymbol{R}_i\boldsymbol{P}^k + \boldsymbol{T}_i \\
\Rightarrow\quad & \boldsymbol{P}^k + \boldsymbol{R}_i^T\boldsymbol{T}_i \parallel \boldsymbol{R}_i^T\boldsymbol{p}_i^k \\
\Rightarrow\quad & \boldsymbol{P}^k - \boldsymbol{C}_i \parallel \boldsymbol{R}_i^T\boldsymbol{p}_i^k
\end{aligned}
$$

Existence of solution

- Role of 3D points and camera locations are interchangeable
- Let $\boldsymbol{T}_i$ denote camera location or 3D point
- $\boldsymbol{t}_{ij}$ is a known direction vector
- Constraints have general form $\boldsymbol{T}_j - \boldsymbol{T}_i \parallel \boldsymbol{t}_{ij}$
- When does a unique solution exist for translation averaging ?

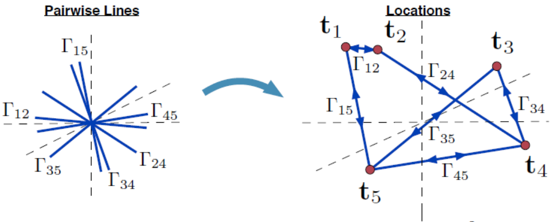


FIG. 1.1. *A (noiseless) instance of the line estimation problem in* $\mathbb{R}^3$, *with* $n = 5$ *locations and* $m = 6$ *pairwise lines.*

Parallel Rigidity of Graph

- Rotation case is simple (spanning tree)
- Translation case is more complicated
- Solution exists when graph satisfies **parallel rigidity**
- Remember solution is upto scale and shift of origin

Figure from Ozyesil *et al.* 2015

Theorem: For a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, let $(d-1)\mathcal{E}$ denote the set consisting of $(d-1)$ copies of each edge in $\mathcal{E}$. Then, $\mathcal{G}$ is generically parallel rigid in $\mathbb{R}^d$ iff there exists a nonempty set $D \subseteq (d-1)\mathcal{E}$, with $|D| = d|V| - (d+1)$, such that for all subsets $D^{'}$ of D, $|D^{'}| \leq d|\mathcal{V}(D^{'})| - (d+1)$, where $\mathcal{V}(D^{'})$ denotes the vertex set of the edges in D.

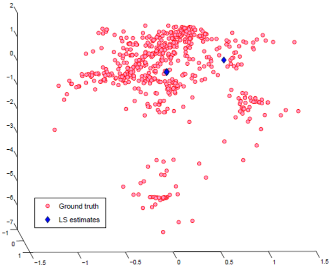
Parallel Rigidity

- Efficient algorithms to verify parallel rigidity
- Efficient method to extract maximal parallel rigid subgraph

From Ozyesil *et al.* 2015

$$\min_{\mathbb{T}} \sum_{\mathcal{E}} (\boldsymbol{T}_j - \boldsymbol{T}_i)^T Q_{ij} (\boldsymbol{T}_j - \boldsymbol{T}_i)$$

$$\text{s. t. } \sum_i \boldsymbol{T}_i = \boldsymbol{0}, \sum_i ||\boldsymbol{T}_i||_2^2 = 1$$



Linear Solutions

- Consider projection matrix $\boldsymbol{Q}_{ij} = \boldsymbol{I} - \boldsymbol{t}_{ij}\boldsymbol{t}_{ij}^T$
- Cost $\sum ||\boldsymbol{Q}_{ij}(\boldsymbol{T}_i - \boldsymbol{T}_j)||_2^2$ since $Q_{ij} = Q_{ij}^T Q_{ij}$
- Constraints exclude trivial solution $\boldsymbol{T}_i = \boldsymbol{0}$
- Solution tends to collapse to have similar $\boldsymbol{T}_i$ for strongly connected nodes

Follows Ozyesil *et al.* "Stable Camera Motion Estimation"

Translation Averaging

$$\min_{\mathbb{T}} \sum_{\mathcal{E}} (\boldsymbol{T}_j - \boldsymbol{T}_i)^T Q_{ij} (\boldsymbol{T}_j - \boldsymbol{T}_i)$$

$$\text{s. t. } \sum_i \boldsymbol{T}_i = \mathbf{0}, \sum_i ||\boldsymbol{T}_i||_2^2 = 1$$

$$\min_{\mathbb{T}} \sum_{\mathcal{E}} Tr\left(\boldsymbol{Q}_{ij} (\boldsymbol{T}_i - \boldsymbol{T}_j)(\boldsymbol{T}_i - \boldsymbol{T}_j)^T\right)$$

$$\text{s. t. } \sum_i \boldsymbol{T}_i = \mathbf{0}, \sum_i ||\boldsymbol{T}_i - \boldsymbol{T}_j||_2^2 > 1$$

SDP Relaxation of Linear Problem

- “Repulsion” constraints prevent collapse of solution
- Results in non-convex constraints
- Convert problem into an SDP form
- Alternating direction augmented Lagrangian (ADM) method
- Also includes stitching overlapping local patches
- Only considers smaller datasets
- Also LUD version (convex relaxation, IRLS) (CVPR 2015)

Ozyesil *et al.* “Stable Camera Motion Estimation”

Results from Ozyesil & Singer 2015 (no BA)

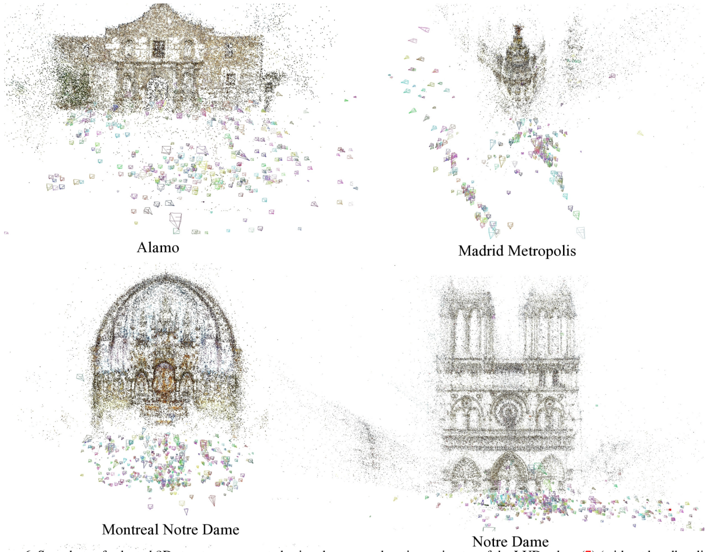


Figure 6. Snapshots of selected 3D structures computed using the camera location estimates of the LUD solver (7) (without bundle adjustment). Each 3D point is visible through at least three cameras.

Translation Averaging

$$\min_{\mathbb{T}} \sum_{\mathcal{E}} \|\mathbb{P}(\boldsymbol{T}_j - \boldsymbol{T}_i)\|_2$$

$$\text{s. t. } \sum_i \boldsymbol{T}_i = \mathbf{0}, \sum_{\mathcal{E}} < \boldsymbol{T}_i - \boldsymbol{T}_j, \boldsymbol{t}_{ij} >= 1$$

ShapeFit & ShapeKick

- Projection $\mathbb{P}$ onto orthogonal complement of $\boldsymbol{T}_{ij}$
- Convex, second order cone problem
- Robust due to unsquared distance term
- Solved using ADMM (Kick in accelaration of steps)
- Faster than methods of similar class of approaches

Translation Averaging

General Observations on Translation Averaging

- Geometry of $\boldsymbol{T}_{ij} \in \mathcal{S}^2$ is simple
- Difficulty lies in relating direction measurements with absolute displacements
- Variety of design issues arise out of this representation discrepancy (ill-posedness)
- No such problem in $\boldsymbol{R}_{ij} = \boldsymbol{R}_j \boldsymbol{R}_i^{-1} \in \mathbb{SO}(3)$
- Need to enforce "repulsion" constraints to avoid collapse
- Direction measurements ($\boldsymbol{t}_{ij}$) from epipolar geometry can be highly biased
- Structure points may be needed for stabilization of solution (camera-point)
- Scalability of efficient and robust solutions not fully resolved

Euclidean Motion Averaging

Euclidean Motion

$$Q = RP + T$$

$$P, Q \in \mathbb{R}^3$$

Homogeneous form

$$\begin{bmatrix} Q \\ \hline 1 \end{bmatrix} = \overbrace{\left[\begin{array}{c|c} R & T \\ \hline \mathbf{0} & 1 \end{array}\right]}^{M} \begin{bmatrix} P \\ \hline 1 \end{bmatrix}$$

Rigid Motion Transformations

- Consider 3D point $P \in \mathbb{R}^3$ and transformed Q
- Rigid (Euclidean) Transformations preserve
  - Length
  - Angles
- 3D Rotation R has 3 dof
- 3D Translation T has 3 dof
- 3D Euclidean transformation M has 6 dof

Euclidean Motion Averaging

Euclidean Motion

$$\boldsymbol{Q} = \boldsymbol{R}\boldsymbol{P} + \boldsymbol{T}$$

$$\boldsymbol{P}, \boldsymbol{Q} \in \mathbb{R}^3$$

Homogeneous form

$$\left[\begin{array}{c} \boldsymbol{Q} \\ \hline 1 \end{array}\right] = \overbrace{\left[\begin{array}{c|c} \boldsymbol{R} & \boldsymbol{T} \\ \hline \boldsymbol{0} & 1 \end{array}\right]}^{\boldsymbol{M}} \left[\begin{array}{c} \boldsymbol{P} \\ \hline 1 \end{array}\right]$$

- At the heart of geometric computer vision
- Essential for representing camera and object geometry
- Rigid transformations $\Leftrightarrow$ Euclidean Motions
- Elegant geometric structure
- Significant developments in estimation
- Fundamental Differences between 2D and 3D Rotations
- Euclidean Motions play central role
  - 2D-2D geometry (image registration, mosaics)
  - 3D-2D geometry (structure from motion)
  - 3D-3D geometry (scan alignment, ICP)

Euclidean Motion Averaging

Consider two Euclidean transformations

$$
\begin{aligned}
\boldsymbol{Q} &= \boldsymbol{R}_2(\boldsymbol{R}_1\boldsymbol{P} + \boldsymbol{T}_1) + \boldsymbol{T}_2 \\
\Rightarrow \boldsymbol{M}_2\boldsymbol{M}_1 &= \left[\begin{array}{c|c} \boldsymbol{R}_2 & \boldsymbol{T}_2 \\ \hline \boldsymbol{0} & 1 \end{array}\right]\left[\begin{array}{c|c} \boldsymbol{R}_1 & \boldsymbol{T}_1 \\ \hline \boldsymbol{0} & 1 \end{array}\right] \\
&= \left[\begin{array}{c|c} \boldsymbol{R}_2\boldsymbol{R}_1 & \boldsymbol{R}_2\boldsymbol{T}_1 + \boldsymbol{T}_2 \\ \hline \boldsymbol{0} & 1 \end{array}\right]
\end{aligned}
$$

Now consider

$$
\begin{aligned}
\boldsymbol{R}_2\boldsymbol{R}_1 &= \boldsymbol{I} \Rightarrow \boldsymbol{R}_2 = \boldsymbol{R}_1^{-1} \\
\boldsymbol{R}_2\boldsymbol{T}_1 + \boldsymbol{T}_2 &= \boldsymbol{0} \Rightarrow \boldsymbol{T}_2 = -\boldsymbol{R}_1^{-1}\boldsymbol{T}_1
\end{aligned}
$$

- Implies that $\boldsymbol{M}_2\boldsymbol{M}_1$ and $\boldsymbol{M}^{-1}$ are elements of $\mathbb{SE}(3)$
- $\mathbb{SO}(3)$ and $\mathbb{SE}(3)$ are Lie groups (matrix groups)
- Note that these groups are not Abelian $\boldsymbol{M}_2\boldsymbol{M}_1 \neq \boldsymbol{M}_1\boldsymbol{M}_2$

Euclidean Motion Averaging

$\mathfrak{se}(3)$

$$\boldsymbol{M} = exp(\mathfrak{m}) = \sum_{k=0}^{\infty} \frac{\mathfrak{m}^k}{k!} = \left[\begin{array}{c|c} \boldsymbol{R} & \boldsymbol{t} \\ \hline 0 & 1 \end{array}\right] \text{ where } \mathfrak{m} = \left[\begin{array}{c|c} \boldsymbol{\Omega} & \boldsymbol{u} \\ \hline 0 & 0 \end{array}\right]$$

$$\text{i.e. } \boldsymbol{R} = exp(\boldsymbol{\Omega}) \text{ and } \boldsymbol{t} = \boldsymbol{P}\boldsymbol{u}$$

$$\text{where } \boldsymbol{P} = \boldsymbol{I} + \frac{(1-\cos\theta)}{\theta^2}\boldsymbol{\Omega} + \frac{(\theta - \sin\theta)}{\theta^3}\boldsymbol{\Omega}^2 \text{ and } \theta = \sqrt{\frac{1}{2} tr(\boldsymbol{\Omega}^T\boldsymbol{\Omega})}$$

BCH forms for Motion Groups

- General BCH series is unwieldy
- Closed forms available for BCH in $\mathfrak{so}(3)$ and $\mathfrak{se}(3)$

Distance Metrics on Lie Groups

- Left-invariant: $d(\boldsymbol{X}_1, \boldsymbol{X}_2) = d(\boldsymbol{X}\boldsymbol{X}_1, \boldsymbol{X}\boldsymbol{X}_2)\ \forall \boldsymbol{X} \in \mathbb{G}$
- Right-invariant: $d(\boldsymbol{X}_1, \boldsymbol{X}_2) = d(\boldsymbol{X}_1\boldsymbol{X}, \boldsymbol{X}_2\boldsymbol{X})\ \forall \boldsymbol{X} \in \mathbb{G}$
- Bi-invariant: Both left- and right-invariant metric
- Intrinsic metric on $\mathbb{SO}(3)$ is bi-invariant
- No bi-invariant metric on $\mathbb{SE}(3)$

Euclidean Motion Averaging

Homogeneous form

$$\left[\begin{array}{c} \boldsymbol{Q} \\ \hline 1 \end{array}\right] = \overbrace{\left[\begin{array}{c|c} \boldsymbol{R} & \boldsymbol{T} \\ \hline \mathbf{0} & 1 \end{array}\right]}^{\boldsymbol{M}} \left[\begin{array}{c} \boldsymbol{P} \\ \hline 1 \end{array}\right]$$

Left invariance

$$d(\boldsymbol{M}\boldsymbol{M}_1, \boldsymbol{M}\boldsymbol{M}_2) = d(\boldsymbol{M}_1, \boldsymbol{M}_2)$$
$$\boldsymbol{Q}_1^{'} = \boldsymbol{M}\boldsymbol{M}_1\boldsymbol{P} = \boldsymbol{M}(\boldsymbol{Q}_1)$$

Right invariance

$$d(\boldsymbol{M}_1\boldsymbol{M}, \boldsymbol{M}_2\boldsymbol{M}) = d(\boldsymbol{M}_1, \boldsymbol{M}_2)$$
$$\boldsymbol{Q}_1^{'} = \boldsymbol{M}_1(\boldsymbol{M}\boldsymbol{P})$$

Transformations in $\mathbb{SE}(3)$

- Left-invariant Riemannian metric (camera centric)
- Right-invariant Riemannian metric (object centric)
- Both cannot be satisfied on $\mathbb{SE}(3)$

Euclidean Motion Averaging

Consider metric in $\mathbb{R}^n$, $\psi : \mathbb{R}^n \times \mathbb{R}^n \longrightarrow \mathbb{R}$

- $\psi(\boldsymbol{v}_1, \boldsymbol{v}_2) = \boldsymbol{v}_1^T \psi \boldsymbol{v}_2$
- Symmetric $\psi(\boldsymbol{v}_1, \boldsymbol{v}_2) = \psi(\boldsymbol{v}_2, \boldsymbol{v}_1)\ \forall \boldsymbol{v}_1, \boldsymbol{v}_2 \in \mathbb{R}^n$
- Positive semi-definite: $\psi(\boldsymbol{v}, \boldsymbol{v}) \geq 0, \psi(\boldsymbol{v}, \boldsymbol{v}) = 0$ iff $\boldsymbol{v} = \mathbf{0}$
- For $\mathbb{SE}(3)$ $\psi_x : T_x M \times T_x M \longrightarrow \mathbb{R}$

No bi-invariant metric in $\mathbb{SE}(3)$

- Sketch of argument
- Define quadratic form on Lie algebra
- Require preservation of quadratic form under transformation
- Resulting constraint lacks positive semi-definiteness
- Implies that $\mathbb{SE}(3)$ does not have bi-invariant metric
- Loncaric 1985
- Murray *et al.* *A Mathematical Introduction to Robotic Manipulation* 1994
- Park “Distance Metrics on the Rigid-Body Motions ...” 1995

Euclidean Motion Averaging

Consider the average of $\boldsymbol{M}_1, \boldsymbol{M}_2 \in \mathbb{SE}(3)$

$$\boldsymbol{M}_1 = \left[\begin{array}{c|c} \boldsymbol{R}_1 & \boldsymbol{T}_1 \\ \hline \mathbf{0} & 1 \end{array}\right] \qquad \boldsymbol{M}_2 = \left[\begin{array}{c|c} \boldsymbol{R}_2 & \boldsymbol{T}_2 \\ \hline \mathbf{0} & 1 \end{array}\right]$$

versus

$$\boldsymbol{M}_1 = \left[\begin{array}{c|c} \boldsymbol{R}_1 & 100\,\boldsymbol{T}_1 \\ \hline \mathbf{0} & 1 \end{array}\right] \qquad \boldsymbol{M}_2 = \left[\begin{array}{c|c} \boldsymbol{R}_2 & 100\,\boldsymbol{T}_2 \\ \hline \mathbf{0} & 1 \end{array}\right]$$

Properties of $\mathbb{SE}(3)$

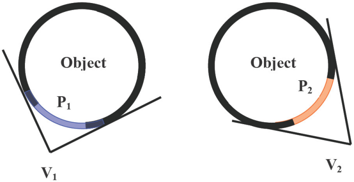
- Lacks bi-invariant metric
- $\mathbb{SO}(3)$ is compact, $\mathbb{SE}(3)$ is not
- Translation scale (units for length) is arbitrary
- Scaling determines weightage in metric
- Epipolar geometry does not give elements in $\mathbb{SE}(3)$
- Can use 3D structure
- Applicable in 3D registration

Euclidean Motion Averaging

Scans from multiple orientations

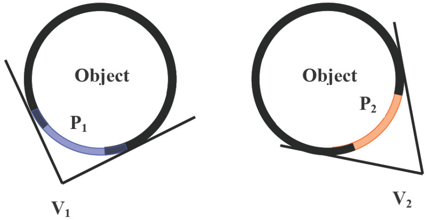
Figure from Stanford Repository

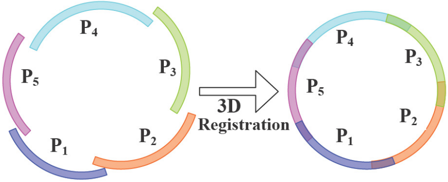
Euclidean Motion Averaging



- Each scan is a *partial* model
- Has own *local* frame of reference

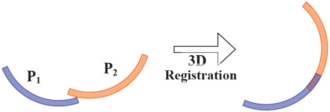
Euclidean Motion Averaging

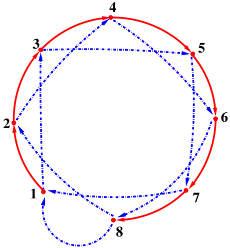




- Need to **register** partial scans
- Need a **single** common frame of reference

Euclidean Motion Averaging

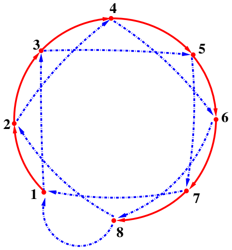




Iterative Closest Point Algorithm

- Standard solution for 3D registration
- ICP limited to 2 views at a time
- Does not carry out *simultaneous* registration of scans
- Incremental drift in the solution
- Fails to exploit available constraints like 'loop closure'

Euclidean Motion Averaging



Consider turntable moving 45° between scans

- **ICP**: Uses spanning tree (Red edges)
- $\boldsymbol{M}_8 = \boldsymbol{M}_{78}\boldsymbol{M}_{67}\cdots\boldsymbol{M}_{12}\boldsymbol{I}$
- **Motion Averaging** : Spanning tree + Blue dashed edges
- $\boldsymbol{M}_{81}$ acts as an 'anchor'
- Reduces drift significantly

Euclidean Motion Averaging

Our Contribution

- Combine ICP with Motion Averaging
- Motion Averaged ICP (**MAICP**)
- Utilises all two-view relationships available
- Significantly reduces drift
- Distributes errors uniformly
- Batch method works well in contexts where KinectFusion fails

Euclidean Motion Averaging

Motion Averaged ICP

Given

- scans $\{\boldsymbol{P}_1, \boldsymbol{P}_2, \ldots, \boldsymbol{P}_N\}$
- initial motions

Iterate till convergence

- **Correspondence Step**
  - Straightforward extension of the 'correspondence step' of ICP
  - Correspondences for all scan pairs possible
- **Motion Step**
  - Using correspondence pairs, compute relative motions
  - Average these $\boldsymbol{M}_{ij}$'s to get global motion $\boldsymbol{M}$
  - Use averaged solution

$$\forall (i,j) \in \mathcal{E} \ \boldsymbol{M}_{ij} \leftarrow \boldsymbol{M}_j \boldsymbol{M}_i^{-1}$$

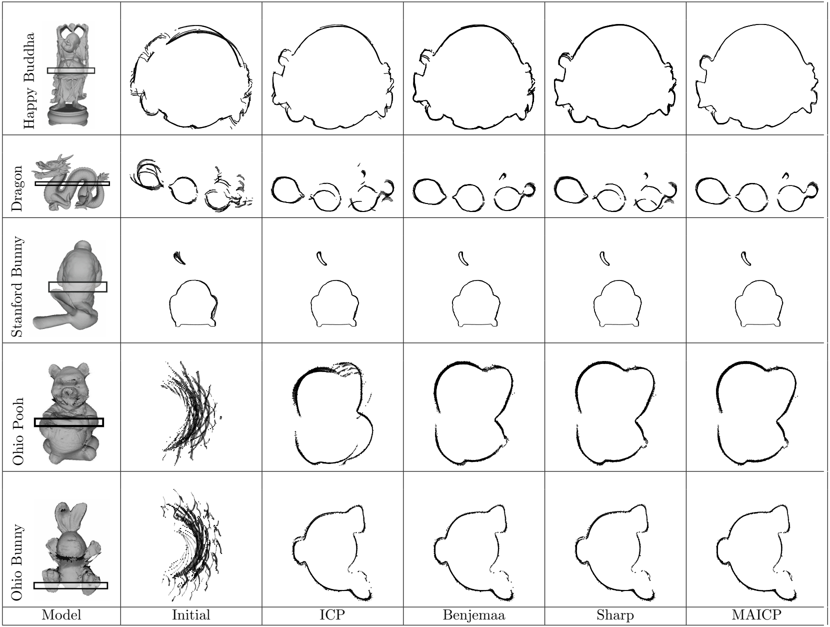
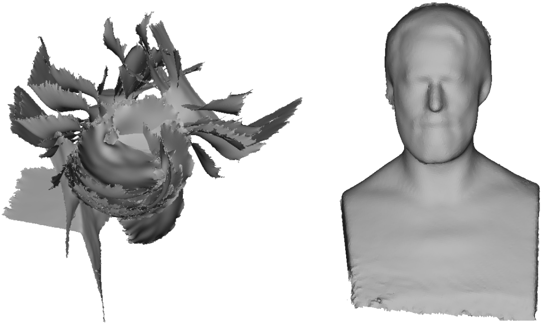


Figure : Cross-sections of registered scans for different methods

Improvements on MAICP

- Our MAICP used standard formulation of ICP
- Can use improvements on ICP in literature
- Replace ICP with trimmed ICP (Li *et al.* 3DV 2014)
- Makes motion steps more robust
- Replace ℓ_2 averaging in $\mathbb{SE}(3)$ with robust form

Euclidean Motion Averaging



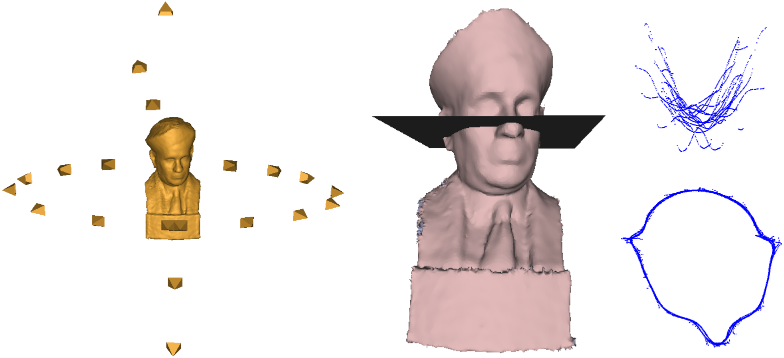
Alignment using ℓ_2 and robust motion averaging

Motion Averaged ICP

- Robust global motion estimation using ℓ_1 optimization

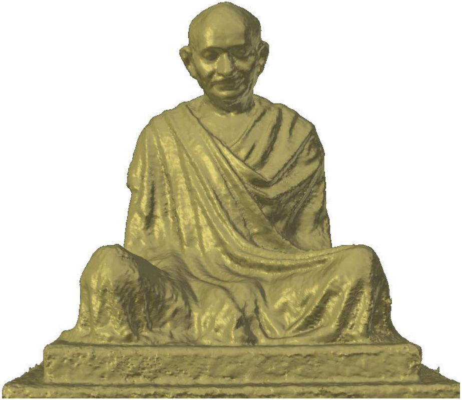
$$\sum_{(i,j)\in\mathcal{E}} d(\boldsymbol{M}_{ij}, \boldsymbol{M}_j \boldsymbol{M}_i{}^{-1})$$

Euclidean Motion Averaging



Left: Final model with estimated scanner locations
Right: Cross-sections before and after alignment

Euclidean Motion Averaging



Statue of Mahatma Gandhi at Sabarmati Ashram, Ahmedabad (90 cm height)

Euclidean Motion Averaging

$\mathbb{SE}(3)$ Averaging

$$\boldsymbol{M}_1 \boldsymbol{M}_2 \in \mathbb{SE}(3)$$
$$\boldsymbol{M}_{ij} = \boldsymbol{M}_j \boldsymbol{M}_i^{-1}$$

Transformation of $\mathbb{R}^3$

$$\boldsymbol{P}_k = \boldsymbol{M}_k \boldsymbol{P}_0$$
$$\implies \min \sum \left\| \boldsymbol{M}_i^{-1} \boldsymbol{P}_i - \boldsymbol{M}_j^{-1} \boldsymbol{P}_j \right\|^2$$

Two Types of Group Action of $\mathbb{SE}(3)$

- First one works on $\mathbb{SE}(3)$ group
- Uses intrinsic metric as distance
- Second one measures distortion of $\mathbb{R}^3$ space of 3D points
- Compare observations transformed into common reference frame
- Older methods: Bergevin *et al.* 1996, Benjemaa *et al.* 1997
- Can also represent $\mathbb{SE}(3)$ as dual quaternions
- Torsello *et al.* "Multiview Registration ..." 2011

Euclidean Motion Averaging

$$\boldsymbol{X} = \begin{pmatrix} \boldsymbol{I}_4 & \boldsymbol{M}_{12} & \cdots & \boldsymbol{M}_{1N} \\ \boldsymbol{M}_{21} & \boldsymbol{I}_4 & \cdots & \boldsymbol{M}_{2N} \\ \cdots & \cdots & \cdots & \cdots \\ \boldsymbol{M}_{N1} & \boldsymbol{M}_{N2} & \cdots & \boldsymbol{I}_4 \end{pmatrix}$$

$$\boldsymbol{M} = \begin{bmatrix} \boldsymbol{M}_1 \\ \boldsymbol{M}_2 \\ \cdots \\ \boldsymbol{M}_N \end{bmatrix}, \boldsymbol{M}^{-\flat} = \begin{bmatrix} \boldsymbol{M}_1^{-1} & \boldsymbol{M}_2^{-1} & \cdots & \boldsymbol{M}_N^{-1} \end{bmatrix}$$

Rank relaxation for $\mathbb{SE}(3)$

- Here $\boldsymbol{X} = \boldsymbol{M}\boldsymbol{M}^{-\flat}$
- $rank(\boldsymbol{X}) = 4$
- Can proceed as in case of $\mathbb{SO}(3)$

Extrinsic averaging becomes

$$\min_{\boldsymbol{X},\boldsymbol{S}} \left\| \mathcal{P}_\Omega(\widehat{\boldsymbol{X}} - \boldsymbol{X}) - \boldsymbol{S} \right\|^2$$
$$\text{s. t. } \boldsymbol{X} = \boldsymbol{M}\boldsymbol{M}^{-\flat},\ \boldsymbol{M} \in \mathbb{SE}(3)^n, \boldsymbol{S} \text{ sparse in } \Omega$$

Using rank relaxation

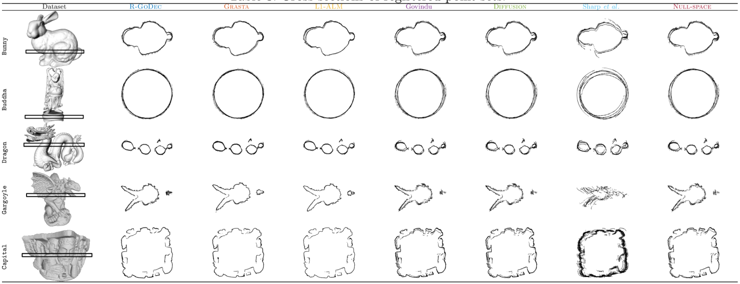
$$\min_{\boldsymbol{L},\boldsymbol{S}} \left\| \mathcal{P}_\Omega(\widehat{\boldsymbol{X}} - \boldsymbol{L}) - \boldsymbol{S} \right\|^2$$
$$\text{s. t. } rank(\boldsymbol{L}) \leq 4, \boldsymbol{S} \text{ sparse in } \Omega$$

Rank relaxation for $\mathbb{SE}(3)$

- $\boldsymbol{S}$ is sparse set of outliers
- Estimated $\boldsymbol{L}$ only enforces rank constraint
- Need to enforce last row of $\boldsymbol{M}_i = [0\ 0\ 0\ 1]$
- Project onto $\mathbb{SE}(3)$ group
- Arrigoni *et al.* “Global Registration of 3D Point Sets ...” 2016

Euclidean Motion Averaging

Table 3: Cross-sections of registered point-sets.



Rank relaxation for $\mathbb{SE}(3)$

- Arrigoni *et al.* “Global Registration of 3D Point Sets ...” 2016
- Compares multiple approaches to $\mathbb{SE}(3)$ registration
- For ℓ_2 averaging, many methods have similar performance
- Fixing translation scale before averaging improves performance
- Speed improvement achieved by rank relaxation methods over others

Conclusions on $\mathbb{SE}(3)$ Averaging

- Fundamental differences between $\mathbb{SE}(3)$ and $\mathbb{SO}(3)$
- Lack of bi-invariant metric and scale dependence needs careful attention
- Significantly improves performance of multiview 3D registration methods
- Provides a natural framework to account for motion redundancy and loop closures
- Similar problems arise in the context of SLAM
- Differences between causal vs. batch approaches to averaging
- Viewgraphs for 3D scans (and SLAM) are different in nature from that of SfM

Conclusions

Advantages of Motion Averaging

- New paradigm for solving camera motion estimation problems
- Provides a unified view of camera relationships
- Lie Group structure can be effectively utilized
- Yields efficient, accurate and robust solutions
- Significant understanding of properties of intrinsic and extrinsic methods

Conclusions

Future Directions

- Scaling issues for ever growing problem sizes
- Deeper understanding of global properties of intrinsic methods
- Conditions for global guarantees
- New methods for more complex motion models
- Better models for uncertainty of observations
- Convergence with large body of work on SLAM (g2o etc.)

Acknowledgments

For figures and animations

- Uttaran Bhattacharya
- Mohammadul Haque
- Avishek Chatterjee