

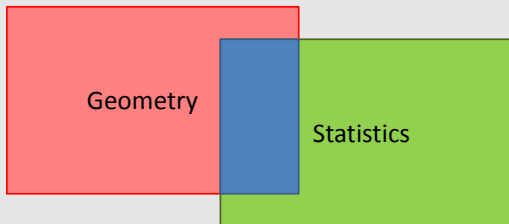
MOTION AVERAGING IN 3D VISION

A Framework for Efficient and Accurate Large-Scale Camera
Estimation

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

18 June 2018

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

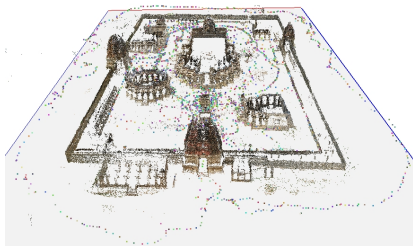
- Geometry at the core of 3D reconstruction
- Consider two sources of data
 - camera images (RGB)
 - depth maps (Kinect etc.)
 - ~~Radiometry~~
- Recovering camera motions for
 - RGB cameras (structure-from-motion)
 - depth cameras (3D scanning)



From <http://www.clipartkid.com>

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

Introduction

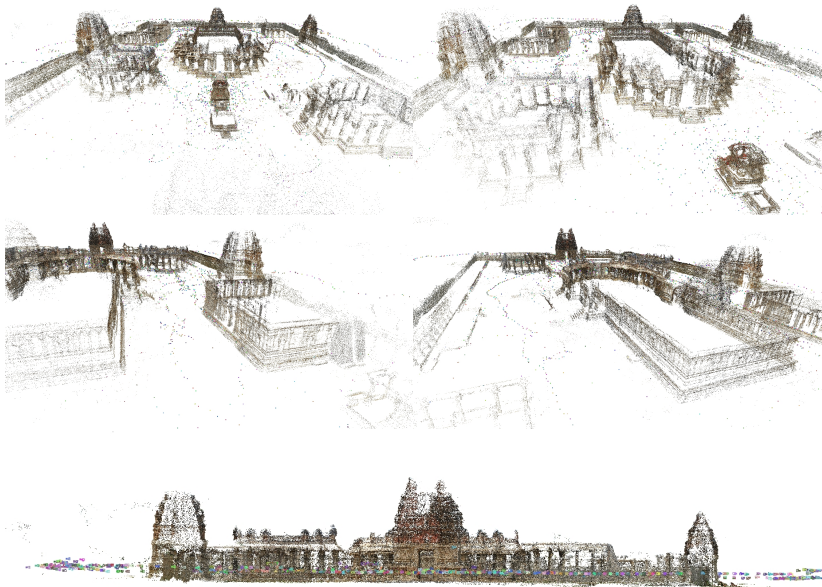


Structure from Motion

- 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

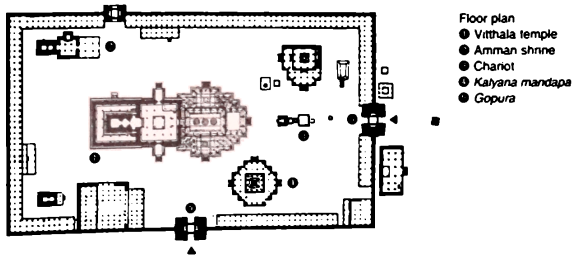
Vitthala Temple, Hampi

3D Reconstruction

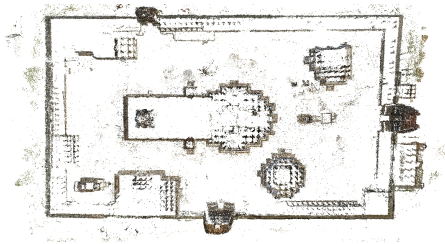


Vitthala Temple, Hampi

Comparison of Reconstruction

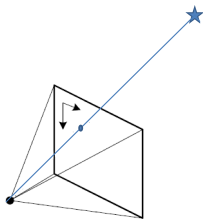


(a) Architectural Drawing



(b) Reconstruction

Introduction

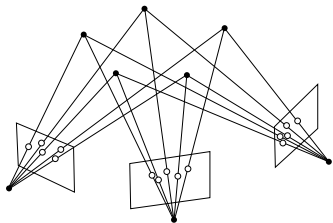


$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbf{K} \left[\mathbf{R} \mid \mathbf{T} \right] \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Camera Projection Model

- Pin-hole Projection Equation ($\mathbb{P}^3 \rightarrow \mathbb{P}^2$)
- 3D Point Representation
- Rigid Motion of Camera $\{\mathbf{R}, \mathbf{T}\}$
- Camera Calibration \mathbf{K} (assume $\mathbf{K} = \mathbf{I}$)
- Projection on image plane (u, v)
- Non-linear projection is key problem

Introduction

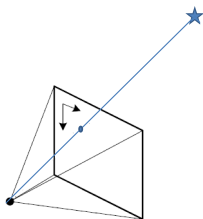


$$\min_{\mathbf{R}, \mathbf{T}, \mathbf{S}} \sum d^2(u_i^k, \hat{u}_i^k) + d^2(v_i^k, \hat{v}_i^k)$$

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
- Standard approaches are incremental (Bundler, VSFM)
- Impressive quality but still suffer from limitations/problems
- Cannot recover from errors, no global view of SfM problem

Introduction

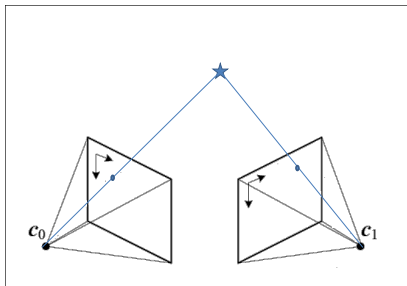


$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbf{K} \left[\mathbf{R} \mid \mathbf{T} \right] \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

KEY PROBLEM IN SfM

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

Introduction



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

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

Can decompose $\mathbf{E} \longrightarrow (\mathbf{R}, \mathbf{t})$

Contrasting properties

- Motion estimation for two 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

Introduction

Original Motivation 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)
- Many variations exist
- 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
- Even distribution of errors across solution
- Naturally accounts for loop closures
- Avoids pitfalls of incremental methods
- Desiderata: robust, fast, scalable, accurate
- Efficient solution can be refined by batch BA
- Also applicable for Euclidean motion averaging
- Similar approaches
 - Time synchronizations of wireless networks
 - Multi-dimensional Scaling

Introduction

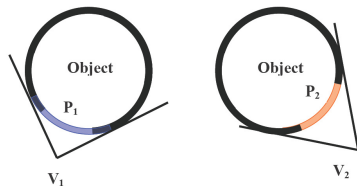
Pin-hole Model

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R} & | & \mathbf{T} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$
$$= \overbrace{\begin{bmatrix} \mathbf{I} & | & \mathbf{0} \end{bmatrix}}^{\text{Ideal camera}} \underbrace{\begin{bmatrix} \mathbf{R} & | & \mathbf{T} \\ \hline \mathbf{0} & | & 1 \end{bmatrix}}_{\text{Camera Motion } \mathbf{M}} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Properties

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

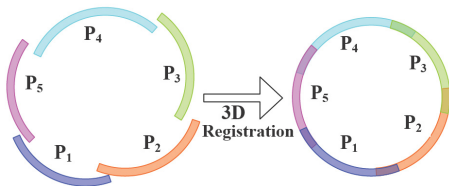
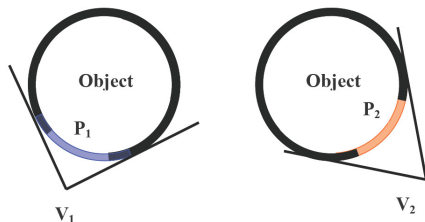
Introduction



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 Camera Model

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{I} & \mathbf{0} \end{bmatrix} \underbrace{\begin{bmatrix} \mathbf{R} & \mathbf{T} \\ \mathbf{0} & 1 \end{bmatrix}}_{\mathbf{M}} \begin{bmatrix} \mathbf{X} \\ \mathbf{Y} \\ \mathbf{Z} \\ 1 \end{bmatrix}$$

3D Registration

$$\begin{bmatrix} \mathbf{X}' \\ \mathbf{Y}' \\ \mathbf{Z}' \\ 1 \end{bmatrix} = \underbrace{\begin{bmatrix} \mathbf{R} & \mathbf{T} \\ \mathbf{0} & 1 \end{bmatrix}}_{\mathbf{M}} \begin{bmatrix} \mathbf{X} \\ \mathbf{Y} \\ \mathbf{Z} \\ 1 \end{bmatrix}$$

Types of Motion Representations

We will consider averaging problems for

- $\mathbf{M} \in \mathbb{SE}(3)$ (Rigid 3D Euclidean Motion)
- $\mathbf{R} \in \mathbb{SO}(3)$ (Rigid 3D Rotation)
- $\mathbf{T} \in \mathbb{R}^3$ (Rigid 3D Translation)

$\mathbf{t} \in \mathcal{S}^2$ (Translation Direction $\mathbf{t} = \frac{\mathbf{T}}{\|\mathbf{T}\|}$)

Introduction

First Rotate then Translate

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

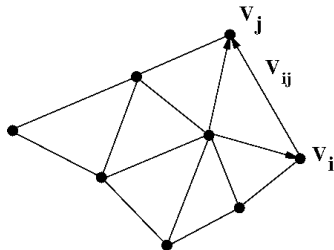
First Translate then Rotate

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

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!

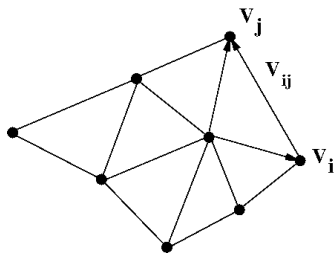
Theory and Formulation



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

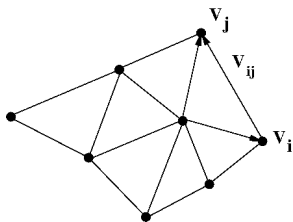
Theory and Formulation



Observation

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

Theory and Formulation

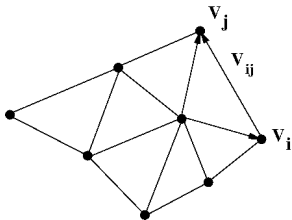


- Relative displacements observed
- $v_j - v_i = v_{ij}$
- Loop : $v_j - v_{ij} - v_i = 0$

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

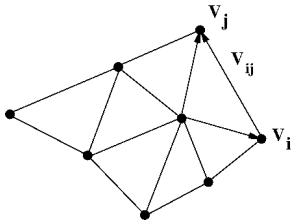


- Relative displacements observed
- $\mathbf{v}_j - \mathbf{v}_i = \mathbf{v}_{ij}$
- Loop : $\mathbf{v}_j - \mathbf{v}_{ij} - \mathbf{v}_i = 0$

Transformation Constraints

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

Theory and Formulation

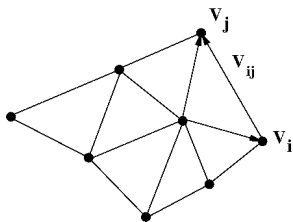


- Relative displacements observed
- $v_j - v_i = v_{ij}$
- Loop : $v_j - v_{ij} - v_i = 0$

Averaging of Relative Motions

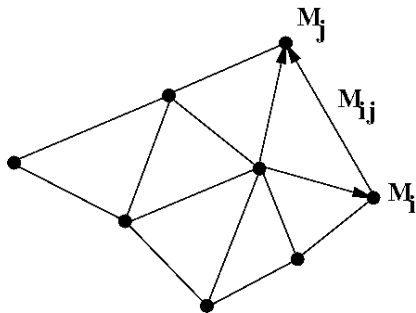
- Define degree of inconsistency $d(v_{ij}, v_j - v_i) = \|v_{ij} - (v_j - v_i)\|$
- Minimise cost function : $\sum_{\mathcal{E}} d^2(v_{ij}, v_j - v_i)$
- Solve minimisation of $\sum_{\mathcal{E}} \|v_{ij} - (v_j - v_i)\|^2$
- Linear estimation problem
- Averages the information of all edges

Theory and Formulation



$$\overbrace{\begin{bmatrix} \dots & -1 & \dots & 1 & \dots \end{bmatrix}}^{\mathbf{A}_{ij}} \underbrace{\begin{bmatrix} \vdots \\ \mathbf{v}_i \\ \vdots \\ \mathbf{v}_j \\ \vdots \end{bmatrix}}_{\mathbf{V}_V} = \overbrace{\begin{bmatrix} \vdots \\ \mathbf{v}_{ij} \\ \vdots \end{bmatrix}}^{\mathbf{V}_E}$$

- Linear System of Equations : $\mathbf{A}\mathbf{V} = \mathbf{V}_{ij}$
- \mathbf{A} encodes the viewgraph
- \mathbf{A} is the graph incidence matrix
- Related to graph Laplacian $\mathbf{L} = \mathbf{A}^T \mathbf{A}$
- Graph Laplacian plays role in characterising problem difficulty
- Gauge freedom: $\mathbf{A}\mathbf{V} = \mathbf{A}(\mathbf{V} + s\mathbf{1})$



Averaging of Relative Motions

For our purposes, consider motion matrices M

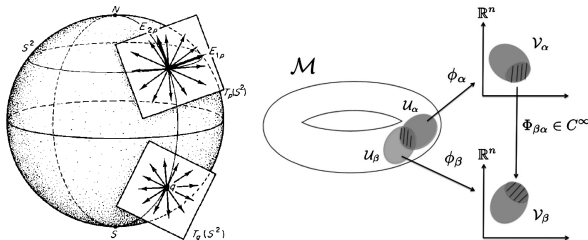
- $M_j M_i^{-1} = M_{ij}, \forall i \neq j$
- Analogous to $\mathbf{v}_j - \mathbf{v}_i = \mathbf{v}_{ij}$
- LHS : Global Motion to be fitted
- RHS : Observations

$$\overbrace{\begin{bmatrix} \cdots & M_{ij} & \cdots & -I & \cdots \end{bmatrix}}^{B_{ij}} \underbrace{\begin{bmatrix} \vdots \\ M_i \\ \vdots \\ M_j \\ \vdots \end{bmatrix}}_{M_g} = 0$$

Averaging of Relative Motions

- $M_{ij} = M_j M_i^{-1}, \forall i \neq j \Rightarrow M_{ij} M_i - M_j = 0$
- Global Motion : $M_g = \{M_1, \dots, M_N\}$
- System of Equations $BM_g = 0$
- Linear Solution ?
- Not valid for **non-linear** motion groups
- Motion Groups are Lie Groups

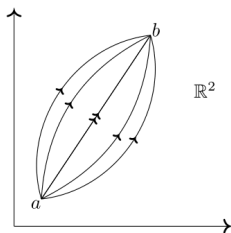
Theory and Formulation



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)
- $T_p\mathcal{M}$ at point $p \in \mathcal{M}$
- Vector (tangent) space equipped with inner product g_p
- **Key idea:** Smooth, differentiable manifold

Theory and Formulation



Riemannian Manifold

- **Key idea:** Smooth, differentiable manifold
- $T_p\mathcal{M}$ at point $p \in \mathcal{M}$
- Vector space $T_p\mathcal{M}$ equipped with inner product g_p
- Length of curve: $\int \sqrt{\langle dl, dl \rangle_g} ds$
- 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

$$\mathbf{X} \circ \mathbf{Y} \in \mathbb{G} \text{ (closure)}$$

$$\mathbf{X} \circ (\mathbf{Y} \circ \mathbf{Z}) = (\mathbf{X} \circ \mathbf{Y}) \circ \mathbf{Z} \text{ (associativity)}$$

$$\exists \mathbf{E} \in \mathbb{G} \ni \mathbf{X} \circ \mathbf{E} = \mathbf{E} \circ \mathbf{X} = \mathbf{X} \text{ (identity)}$$

$$\exists \mathbf{X}^{-1} \in \mathbb{G} \ni \mathbf{X} \circ \mathbf{X}^{-1} = \mathbf{X}^{-1} \circ \mathbf{X} = \mathbf{E} \text{ (inverse)}$$

What is a Group ?

- Group is set \mathbb{G} equipped with operation \circ
- Satisfies specific properties
- Examples :
 - $\{\mathbb{R}, +\}$
 - $\{(0, 1, \dots, n-1), \text{mod}(n)\}$
 - $\{\mathbf{M} \in \mathbb{R}^{n \times n} \mid \det(\mathbf{M}) > 0, \times\}$
- 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 \mathbf{X}
- Group representation $\mathrm{GL}(n, \mathbb{R})$ or $\mathrm{GL}(n)$

Theory and Formulation

$$\mathbf{XY} \in \mathbb{GL}(n) \text{ (closure)}$$

$$\mathbf{X}(\mathbf{YZ}) = (\mathbf{XY})\mathbf{Z} \text{ (associativity)}$$

$$\exists \mathbf{I} \in \mathbb{GL}(n) \ni \mathbf{XI} = \mathbf{IX} = \mathbf{X} \text{ (identity)}$$

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

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

Theory and Formulation

$$\mathbf{XY} \in \mathbb{GL}(n) \text{ (closure)}$$

$$\mathbf{X}(\mathbf{YZ}) = (\mathbf{XY})\mathbf{Z} \text{ (associativity)}$$

$$\exists \mathbf{I} \in \mathbb{GL}(n) \ni \mathbf{XI} = \mathbf{IX} = \mathbf{X} \text{ (identity)}$$

$$\exists \mathbf{X}^{-1} \in \mathbb{GL}(n) \ni \mathbf{XX}^{-1} = \mathbf{X}^{-1}\mathbf{X} = \mathbf{I} \text{ (inverse)}$$

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

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

Theory and Formulation

$$\mathbf{XY} \in \mathbb{GL}(n) \text{ (closure)}$$

$$\mathbf{X}(\mathbf{YZ}) = (\mathbf{XY})\mathbf{Z} \text{ (associativity)}$$

$$\exists \mathbf{I} \in \mathbb{GL}(n) \ni \mathbf{XI} = \mathbf{IX} = \mathbf{X} \text{ (identity)}$$

$$\exists \mathbf{X}^{-1} \in \mathbb{GL}(n) \ni \mathbf{XX}^{-1} = \mathbf{X}^{-1}\mathbf{X} = \mathbf{I} \text{ (inverse)}$$

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

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

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

- $\mathbb{GL}(n) = \{\mathbf{X} | \mathbf{X} \in \mathbb{R}^{n^2}, |\mathbf{X}| \neq 0\}$
- Specialise to matrices with determinant of 1
- Also forms a (sub)group since $|\mathbf{AB}| = |\mathbf{A}||\mathbf{B}| = 1$
- 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

$$\mathbf{a} \cdot \mathbf{b} = \mathbf{a}^T \mathbf{b}$$

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)

$$\begin{aligned} \mathbf{a} \cdot \mathbf{b} &= \mathbf{a}^T \mathbf{b} \\ \mathbf{a}^T \mathbf{M}^T \mathbf{M} \mathbf{b} &= \mathbf{a}^T \mathbf{b} \end{aligned}$$

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 $\mathbf{M} \in \mathbb{O}(n)$

$$\mathbf{a} \cdot \mathbf{b} = \mathbf{a}^T \mathbf{b}$$

$$\mathbf{a}^T \mathbf{M}^T \mathbf{M} \mathbf{b} = \mathbf{a}^T \mathbf{b}$$

$$\text{Invariance} \Rightarrow \mathbf{M}^T \mathbf{M} = \mathbf{I}_n$$

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 $\mathbf{M} \in \mathbb{O}(n)$

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

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

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

- $\mathbb{O}(n)^+ = \{\mathbf{M} \in \mathbb{O}(n) : |\mathbf{M}| = 1\}$
- $\mathbb{O}(n)^- = \{\mathbf{M} \in \mathbb{O}(n) : |\mathbf{M}| = -1\}$
- $\mathbb{O}(n)^+$ 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

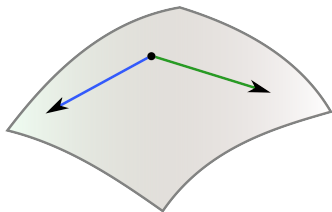
$$\begin{aligned} a : \mathbb{G} \times \mathbb{G} &\longrightarrow \mathbb{G} \\ \mathbf{R}_1 \in \mathrm{SO}(3), \mathbf{R}_2 \in \mathrm{SO}(3) \\ &\Rightarrow \mathbf{R}_1 \mathbf{R}_2 \in \mathrm{SO}(3) \end{aligned}$$

Action on Vector Space

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

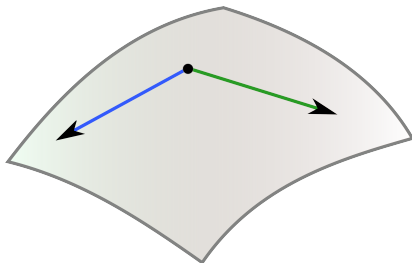
Group Action

- Will consider two types of group action



Lie Algebra \equiv Tangent Space

- Several ways of describing Lie Algebra
- Consider smooth mapping $\gamma : \mathbb{R} \rightarrow \mathbb{G}$
- Require $\gamma(0) = \mathbf{I}$ (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 $\mathbf{M}^T \mathbf{M} = \mathbf{I}_n$
- Around \mathbf{M} , move locally $\mathbf{M} + \epsilon \mathbf{\Delta}$
- For $\epsilon \rightarrow 0$, $(\mathbf{M} + \epsilon \mathbf{\Delta})^T (\mathbf{M} + \epsilon \mathbf{\Delta}) = \mathbf{I}_n$
- Implies $\mathbf{\Delta} + \mathbf{\Delta}^T = \mathbf{0}$
- Tangent space is skew-symmetric in form
- $\mathbf{\Delta} \in \mathfrak{so}(n)$

Theory and Formulation

Conjugation and Adjoint

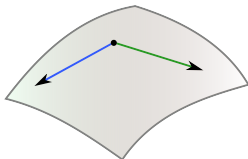
- For $\mathbf{G}, \mathbf{X} \in \mathbb{G}$, mapping $\mathbf{G}\mathbf{X}\mathbf{G}^{-1}$ is a conjugation
- Differentiation maps tangent space at \mathbf{I}_n to itself
- Linearity gives $Ad(\mathbf{G})\mathbf{x} = \mathbf{G}\mathbf{x}\mathbf{G}^{-1} \forall \mathbf{G} \in \mathbb{G}$

Commutators in Lie Algebra

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

Properties of Lie Algebra

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



Lie Groups

- Special properties
 - $\mathbf{X} \times \mathbf{Y} \mapsto \mathbf{XY}$ is a smooth, differentiable mapping
 - $\mathbf{X} \mapsto \mathbf{X}^{-1}$ 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 $[\cdot, \cdot] : \mathfrak{g} \times \mathfrak{g} \rightarrow \mathfrak{g}$
- Properties of bracket

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

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

Exponential Mapping

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

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

Exponential Mapping

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

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

Exponential Mapping

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

Exponential Mapping

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

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

Exponential Mapping

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

Logarithm Mapping

- In neighbourhood around 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

$$\begin{aligned}x = \log(\mathbf{G}) &= \sum_{k=1}^{\infty} \frac{(-1)^{k-1}(\mathbf{G} - \mathbf{I})^k}{k} \\&= (\mathbf{G} - \mathbf{I}) - \frac{1}{2}(\mathbf{G} - \mathbf{I})^2 + \frac{1}{3}(\mathbf{G} - \mathbf{I})^3 + \dots\end{aligned}$$

Logarithm Mapping

- In neighbourhood around \mathbf{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

Distances on Lie Groups

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

Distances on Lie Groups

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

Distance Metrics on Lie Groups

- Left-invariant: $d(\mathbf{X}_1, \mathbf{X}_2) = d(\mathbf{X}\mathbf{X}_1, \mathbf{X}\mathbf{X}_2) \quad \forall \mathbf{X} \in \mathbb{G}$
- Right-invariant: $d(\mathbf{X}_1, \mathbf{X}_2) = d(\mathbf{X}_1\mathbf{X}, \mathbf{X}_2\mathbf{X}) \quad \forall \mathbf{X} \in \mathbb{G}$
- 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: $e^{\mathbf{x}}e^{\mathbf{y}} \neq e^{\mathbf{x}+\mathbf{y}}$
- Neighbourhood of \mathbf{I} : $\mathbf{X} = e^{\mathbf{x}} \approx \mathbf{I} + \mathbf{x}$
- Product commutes $\mathbf{XY} \approx \mathbf{I} + \mathbf{x} + \mathbf{y}$

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

Exponential Representation

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

$$e^{\mathbf{x}}e^{\mathbf{y}} \neq e^{(\mathbf{x}+\mathbf{y})}$$

$$e^{\mathbf{x}}e^{\mathbf{y}} = e^{BCH(\mathbf{x},\mathbf{y})}$$

$$BCH(\mathbf{x},\mathbf{y}) = \mathbf{x} + \mathbf{y} + \frac{1}{2}[\mathbf{x},\mathbf{y}] + \frac{1}{12}[\mathbf{x} - \mathbf{y}, [\mathbf{x},\mathbf{y}]] + \dots$$

Exponential Representation

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

$$e^{\mathbf{x}}e^{\mathbf{y}} \neq e^{(\mathbf{x}+\mathbf{y})}$$

$$e^{\mathbf{x}}e^{\mathbf{y}} = e^{BCH(\mathbf{x},\mathbf{y})}$$

$$BCH(\mathbf{x},\mathbf{y}) = \mathbf{x} + \mathbf{y} + \frac{1}{2}[\mathbf{x},\mathbf{y}] + \frac{1}{12}[\mathbf{x} - \mathbf{y}, [\mathbf{x},\mathbf{y}]] + \dots$$

Exponential Representation

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

Euclidean Motion :

Homogeneous form :

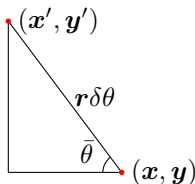
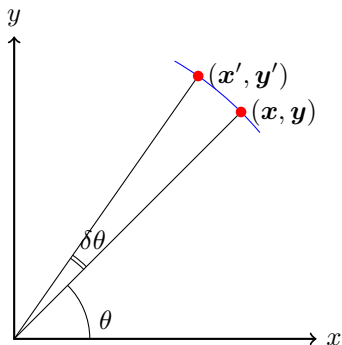
$$\mathbf{Q} = \mathbf{R}\mathbf{P} + \mathbf{T}$$

$$\begin{bmatrix} \mathbf{Q} \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R} & \mathbf{T} \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} \mathbf{P} \\ 1 \end{bmatrix}$$

Motion Groups

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

Theory and Formulation



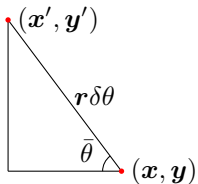
$$x' = x - r\delta\theta \cos\left(\frac{\pi}{2} - \theta\right)$$

$$y' = y + r\delta\theta \sin\left(\frac{\pi}{2} - \theta\right)$$

Two-dimensional Rotations

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

Theory and Formulation



$$x' = x - r\delta\theta \cos\left(\frac{\pi}{2} - \theta\right) = x - r\delta\theta \sin(\theta)$$

$$y' = y + r\delta\theta \sin\left(\frac{\pi}{2} - \theta\right) = y + r\delta\theta \cos(\theta)$$

Two-dimensional Rotations

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

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

where $\mathbf{B} = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$

Two-dimensional Rotations

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

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

Two-dimensional Rotations

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

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

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

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

Two-dimensional Rotations

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

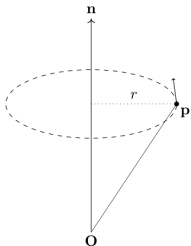
$$\mathbf{R} = \prod_{i=1}^n \delta\mathbf{R} = \left(\mathbf{I} + \frac{\theta}{n} \mathbf{B} \right)^n$$

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

$$\begin{aligned}\mathbf{R} &= \lim_{n \rightarrow \infty} \left(\mathbf{I} + \frac{\theta}{n} \mathbf{B} \right)^n = \exp(\theta \mathbf{B}) \\ &= \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} \\ \text{where } \mathbf{B} &= \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}\end{aligned}$$

Two-dimensional Rotations

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



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

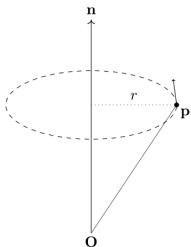
$$r = \|\mathbf{p} - (\mathbf{n}^T \mathbf{p})\mathbf{n}\| = \|\mathbf{n} \times \mathbf{p}\|$$

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

Axis-Angle representation of 3D Rotation

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

Theory and Formulation



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

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

$$\Rightarrow \mathbf{R} = \lim_{k \rightarrow \infty} \prod_{i=1}^k \left(\mathbf{I} + \left[\frac{\theta}{k} \mathbf{n} \right]_{\times} \right)^k = \exp([\omega]_{\times})$$

Axis-Angle representation of 3D Rotation

- As before we can take limit of product of infinitesimal rotations
- $\mathbf{R} = \exp([\omega]_{\times}) \in \mathbb{SO}(3)$; $\omega = \theta \mathbf{n}$
- Rodrigues formula:

$$\mathbf{R} = \exp([\omega]_{\times}) = \mathbf{I} + \sin \theta [\mathbf{n}]_{\times} + (1 - \cos \theta) [\mathbf{n}]_{\times}^2$$

$$\mathbf{B}_x = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{pmatrix} \quad \mathbf{B}_y = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{pmatrix} \quad \mathbf{B}_z = \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

Three-dimensional Rotations

- $\mathbf{R} \in \mathbb{SO}(3)$
- Corresponding Lie Algebra parameters $\omega \in \mathfrak{so}(3)$
- Lie Algebra : $[\omega]_{\times} = \omega_x \mathbf{B}_x + \omega_y \mathbf{B}_y + \omega_z \mathbf{B}_z$
- $\mathbf{R} = \exp([\omega]_{\times})$
- $\mathbf{B}_x, \mathbf{B}_y, \mathbf{B}_z$ related to canonical basis vectors $\mathbf{e}_x, \mathbf{e}_y, \mathbf{e}_y$

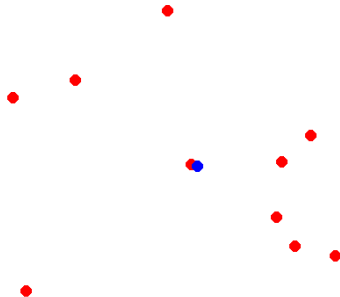
$\mathfrak{so}(3)$

$$BCH(\omega_1, \omega_2) = \alpha\omega_1 + \beta\omega_2 + \gamma\omega_1 \times \omega_2$$

α, β, γ : scalar functions of ω_1, ω_2

BCH forms for Motion Groups

- Recall $e^{\mathbf{x}}e^{\mathbf{y}} \neq e^{(\mathbf{x}+\mathbf{y})}$
- Instead $e^{\mathbf{x}}e^{\mathbf{y}} = e^{BCH(\mathbf{x}, \mathbf{y})}$
- 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



Average of Linear Elements

- Given set of points $\{\mathbf{X}_i\}$
- Mean is $\mu = \frac{\sum_i \mathbf{X}_i}{\sum_i 1}$
- 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
- $\frac{\mathbf{R}_1 + \mathbf{R}_2}{2} \notin \mathbb{SO}(3)$
- Need an appropriate way of defining averages

Extrinsic Averages

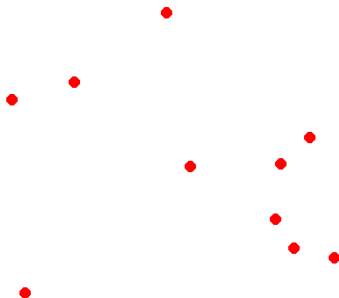
- Ignore geometric constraints, e.g. $\mathbf{R}\mathbf{R}^T = \mathbf{I}$
- Carry out arithmetic average $\hat{\mathbf{R}} = \frac{\mathbf{R}_1 + \mathbf{R}_2}{2}$
- $\hat{\mathbf{R}} \in \mathbb{R}^9$, i.e. $\hat{\mathbf{R}} \notin \mathbb{SO}(3)$
- Project **extrinsic** estimate onto geometric manifold
- 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

Averaging on Lie Groups

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

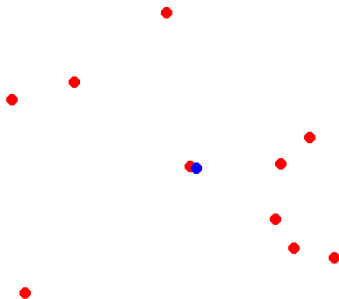
Intrinsic Estimators

- 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
- $\mu\{\mathbf{R}_1, \dots, \mathbf{R}_n\} \in \mathbb{SO}(3)$
- 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



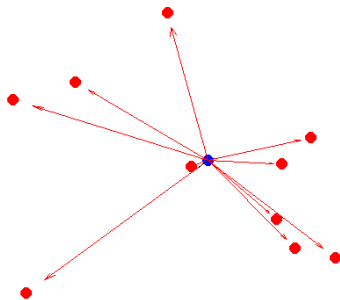
Variational minimisation

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



Variational minimisation

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



Variational minimisation

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

Theory and Formulation

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

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

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

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

- $\lambda = \frac{1}{N}$?
- 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 $\{\mathbf{X}_1, \dots, \mathbf{X}_N\}$,
$$\bar{\mathbf{X}} = \frac{1}{N} \sum_{i=1}^N \mathbf{X}_i$$
- Mean is minimiser of “deviation”
$$\sum_{i=1}^N d^2(\mathbf{X}_i, \bar{\mathbf{X}}) = \sum_{i=1}^N \|\mathbf{X}_i - \bar{\mathbf{X}}\|^2$$

Averaging on Lie Groups

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

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

- When $\mathbf{X} \in \mathcal{M}$ 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:**

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

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

Algorithm for Intrinsic Average

Input : $\{\mathbf{X}_1, \dots, \mathbf{X}_N\} \in \mathbb{G}$ (Matrix Group)

Output : $\boldsymbol{\mu} \in \mathbb{G}$ (Intrinsic Average)

Initialise : $\boldsymbol{\mu} = \mathbf{I}$ (Identity)

Do

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

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

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

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

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

Algorithm for Intrinsic Average

Input : $\{\mathbf{X}_1, \dots, \mathbf{X}_N\} \in \mathbb{G}$ (Matrix Group)

Output : $\boldsymbol{\mu} \in \mathbb{G}$ (Intrinsic Average)

Initialise : $\boldsymbol{\mu} = \mathbf{I}$ (Identity)

Do

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

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

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

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

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

Algorithm for Intrinsic Average

Input : $\{\mathbf{X}_1, \dots, \mathbf{X}_N\} \in \mathbb{G}$ (Matrix Group)

Output : $\boldsymbol{\mu} \in \mathbb{G}$ (Intrinsic Average)

Initialise : $\boldsymbol{\mu} = \mathbf{I}$ (Identity)

Do

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

Subtract current mean from observations

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

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

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

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

Algorithm for Intrinsic Average

Input : $\{\mathbf{X}_1, \dots, \mathbf{X}_N\} \in \mathbb{G}$ (Matrix Group)

Output : $\boldsymbol{\mu} \in \mathbb{G}$ (Intrinsic Average)

Initialise : $\boldsymbol{\mu} = \mathbf{I}$ (Identity)

Do

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

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

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

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

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

Subtract current mean from observations
Map to Lie Algebra

Algorithm for Intrinsic Average

Input : $\{\mathbf{X}_1, \dots, \mathbf{X}_N\} \in \mathbb{G}$ (Matrix Group)

Output : $\boldsymbol{\mu} \in \mathbb{G}$ (Intrinsic Average)

Initialise : $\boldsymbol{\mu} = \mathbf{I}$ (Identity)

Do

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

Subtract current mean from observations

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

Map to Lie Algebra

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

Average and map back to Lie Group

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

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

Algorithm for Intrinsic Average

Input : $\{\mathbf{X}_1, \dots, \mathbf{X}_N\} \in \mathbb{G}$ (Matrix Group)

Output : $\boldsymbol{\mu} \in \mathbb{G}$ (Intrinsic Average)

Initialise : $\boldsymbol{\mu} = \mathbf{I}$ (Identity)

Do

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

Subtract current mean from observations

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

Map to Lie Algebra

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

Average and map back to Lie Group

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

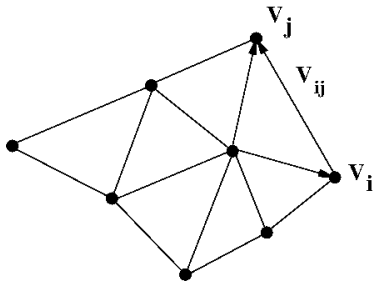
Add update to current mean

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 \mathbf{X}_i within closed ball of radius r
 - For $\text{SO}(3)$, $r < \frac{\pi}{2}$ (Manton 2004, Hartley *et al.* 2013)
 - Riemannian gradient descent step

Theory and Formulation



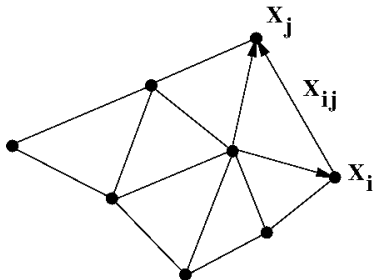
Observations: $\mathbf{v}_j - \mathbf{v}_i = \mathbf{v}_{ij}$

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

Motion Averaging in Vector Space

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

Theory and Formulation



Observations: $M_j M_i^{-1} = M_{ij}$

Cost function: $\sum_{\mathcal{E}} d^2(M_{ij}, M_j M_i^{-1})$

Motion Averaging on Lie Groups

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

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

Motion Averaging on Lie Groups

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

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

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

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

Motion Averaging on Lie Groups

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

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

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

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

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

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

Cost function: $\sum_{\mathcal{E}} d^2(M_{ij}, M_j M_i^{-1})$

$$d(M_{ij}, M_j M_i^{-1}) = BCH(M_{ij}, M_i M_j^{-1})$$

$$\Rightarrow d(M_{ij}, M_j M_i^{-1}) = BCH(M_{ij}, BCH(M_i, M_j^{-1}))$$

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

Motion Averaging on Lie Groups

- First-order expansion of BCH form

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

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

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

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

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

Motion Averaging on Lie Groups

- First-order expansion of BCH form
- Extract parameters \mathbf{v} of Lie algebra \mathfrak{m} : $\mathbf{v} = \mathbf{m}^\vee$

Theory and Formulation

$$\text{Cost function: } \sum_{\mathcal{E}} d^2(M_{ij}, M_j M_i^{-1})$$

$$d(M_{ij}, M_j M_i^{-1}) = BCH(M_{ij}, M_i M_j^{-1})$$

$$\Rightarrow d(M_{ij}, M_j M_i^{-1}) = BCH(M_{ij}, BCH(M_i, M_j^{-1}))$$

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

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

$$\underbrace{\left[\dots - I \dots I \dots \right]}_{A_{ij}} \mathfrak{V} = \mathfrak{v}_{ij}$$

Motion Averaging on Lie Groups

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

$$\begin{aligned} \text{Cost function: } \sum_{\mathcal{E}} d^2(M_{ij}, M_j M_i^{-1}) \\ \underbrace{\begin{bmatrix} \dots & -I & \dots & I & \dots \end{bmatrix}}_{\mathbf{A}_{ij}} \mathfrak{V} = \mathfrak{v}_{ij} \\ M_j M_i^{-1} = M_{ij} \rightsquigarrow \mathbf{A} \mathfrak{V} = \mathbb{V}_{ij} \end{aligned}$$

Motion Averaging on Lie Groups

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

Cost function: $\sum_{\mathcal{E}} d^2(M_{ij}, M_j M_i^{-1})$

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

$$M_j M_i^{-1} = M_{ij} \rightsquigarrow \mathbf{A} \mathfrak{V} = \mathbb{V}_{ij}$$

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

Motion Averaging on Lie Groups

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

Algorithm for Motion Averaging on Lie Groups

Input : $\{M_{ij1}, M_{ij2} \cdots, M_{ijn}\}$

Output : $M_g : \{M_2, \cdots, M_N\}$

Set M_g to an initial guess (Linear solution)

Do

$$\Delta M_{ij} = M_j^{-1} M_{ij} M_i$$

$$\Delta \mathbf{m}_{ij} = \log(\Delta M_{ij})$$

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

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

$$\forall k \in [2, N], M_k = M_k \exp(\Delta \mathbf{v}_k)$$

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

Algorithm for Motion Averaging on Lie Groups

Input : $\{M_{ij1}, M_{ij2} \cdots, M_{ijn}\}$

Output : $M_g : \{M_2, \cdots, M_N\}$

Set M_g to an initial guess (Linear solution)

Do

$$\Delta M_{ij} = M_j^{-1} M_{ij} M_i$$

$$\Delta \mathbf{m}_{ij} = \log(\Delta M_{ij})$$

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

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

$$\forall k \in [2, N], M_k = M_k \exp(\Delta \mathbf{v}_k)$$

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

Algorithm for Motion Averaging on Lie Groups

Input : $\{M_{ij1}, M_{ij2} \cdots, M_{ijn}\}$

Output : $M_g : \{M_2, \cdots, M_N\}$

Set M_g to an initial guess (Linear solution)

Do

$$\Delta M_{ij} = M_j^{-1} M_{ij} M_i$$

Diff of observations and estimate

$$\Delta \mathbf{m}_{ij} = \log(\Delta M_{ij})$$

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

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

$$\forall k \in [2, N], M_k = M_k \exp(\Delta \mathbf{v}_k)$$

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

Algorithm for Motion Averaging on Lie Groups

Input : $\{M_{ij1}, M_{ij2} \cdots, M_{ijn}\}$

Output : $M_g : \{M_2, \cdots, M_N\}$

Set M_g to an initial guess (Linear solution)

Do

$$\Delta M_{ij} = M_j^{-1} M_{ij} M_i$$

Diff of observations and estimate

$$\Delta \mathbf{m}_{ij} = \log(\Delta M_{ij})$$

Move into Lie algebra

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

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

$$\forall k \in [2, N], M_k = M_k \exp(\Delta \mathbf{v}_k)$$

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

Algorithm for Motion Averaging on Lie Groups

Input : $\{M_{ij1}, M_{ij2} \cdots, M_{ijn}\}$

Output : $M_g : \{M_2, \cdots, M_N\}$

Set M_g to an initial guess (Linear solution)

Do

$$\Delta M_{ij} = M_j^{-1} M_{ij} M_i$$

$$\Delta \mathbf{m}_{ij} = \log(\Delta M_{ij})$$

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

$$\text{Solve } \mathbf{A} \Delta \mathfrak{V} = \Delta \mathfrak{V}_{ij}$$

$$\forall k \in [2, N], M_k = M_k \exp(\Delta \mathbf{v}_k)$$

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

Diff of observations and estimate

Move into Lie algebra

Extract parameters in Lie algebra

Algorithm for Motion Averaging on Lie Groups

Input : $\{M_{ij1}, M_{ij2} \cdots, M_{ijn}\}$

Output : $M_g : \{M_2, \cdots, M_N\}$

Set M_g to an initial guess (Linear solution)

Do

$$\Delta M_{ij} = M_j^{-1} M_{ij} M_i$$

$$\Delta \mathbf{m}_{ij} = \log(\Delta M_{ij})$$

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

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

$$\forall k \in [2, N], M_k = M_k \exp(\Delta \mathbf{v}_k)$$

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

Diff of observations and estimate

Move into Lie algebra

Extract parameters in Lie algebra

Solve in vector space

Algorithm for Motion Averaging on Lie Groups

Input : $\{M_{ij1}, M_{ij2} \cdots, M_{ijn}\}$

Output : $M_g : \{M_2, \cdots, M_N\}$

Set M_g to an initial guess (Linear solution)

Do

$$\Delta M_{ij} = M_j^{-1} M_{ij} M_i$$

$$\Delta \mathbf{m}_{ij} = \log(\Delta M_{ij})$$

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

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

$$\forall k \in [2, N], M_k = M_k \exp(\Delta \mathbf{v}_k)$$

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

Diff of observations and estimate

Move into Lie algebra

Extract parameters in Lie algebra

Solve in vector space

Update motions

Theory and Formulation

Input : $\{M_{ij1}, M_{ij2} \cdots, M_{ijn}\}$

Output : $M_g : \{M_2, \cdots, M_N\}$

Set M_g to an initial guess (Linear solution)

Do

$$\Delta M_{ij} = M_j^{-1} M_{ij} M_i$$

$$\Delta \mathbf{m}_{ij} = \log(\Delta M_{ij})$$

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

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

$$\forall k \in [2, N], M_k = M_k \exp(\Delta \mathbf{v}_k)$$

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

Properties of \mathbf{A}

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

Theory and Formulation

Input : $\{M_{ij1}, M_{ij2} \cdots, M_{ijn}\}$

Output : $M_g : \{M_2, \cdots, M_N\}$

Set M_g to an initial guess (Linear solution)

Do

$$\Delta M_{ij} = M_j^{-1} M_{ij} M_i$$

$$\Delta \mathbf{m}_{ij} = \log(\Delta M_{ij})$$

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

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

$$\forall k \in [2, N], M_k = M_k \exp(\Delta \mathbf{v}_k)$$

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

Solutions

- Key issue is solution for $\mathbf{A} \Delta \mathfrak{V} = \Delta \mathbb{V}_{ij}$
- 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)

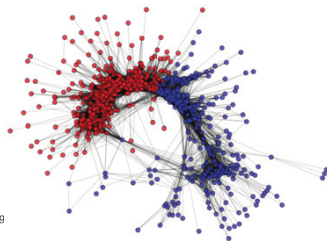
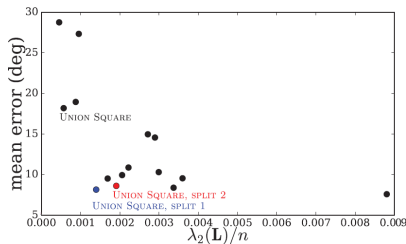
“Convexity” of Rotation Averaging

- 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 $\mathbf{R}_1, \mathbf{R}_2 \in U$ have only one geodesic in U
- Hessian of $f : \mathbb{SO}(3) \rightarrow \mathbb{R}$
- If Hessian psd at \mathbf{R} , f is locally convex at \mathbf{R}
- $\mathbb{SO}(3)$ is locally convex almost everywhere
- Hessian \mathbf{H}_{ij} of $d^2(\mathbf{R}_{ij}, \mathbf{R}_j \mathbf{R}_i^{-1})$ at $(\mathbf{R}_i, \mathbf{R}_j)$
- \mathbf{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

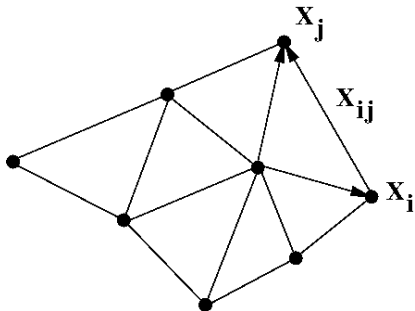
Theory and Formulation



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



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

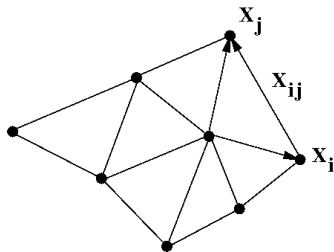
$$\nabla = -2 \sum_i (\mathbf{x}_i - \mathbf{y})$$

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

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

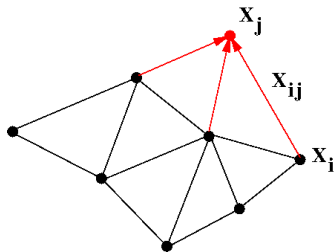
Theory and Formulation



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

- $X_j X_i^{-1} = X_{ij}$
- $X_j = X_{ij} X_i$
- Each vertex in $\mathcal{N}(j)$ suggests X_j
- Averaging takes consensus of these estimates
- Each R_j updated $\forall j \in \mathcal{V}$
- Repeat till convergence
- Weiszfeld algorithm is a robust variant

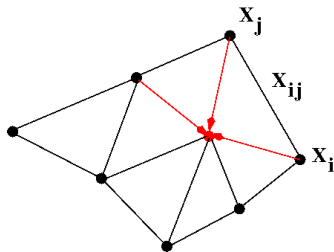
Theory and Formulation



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

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

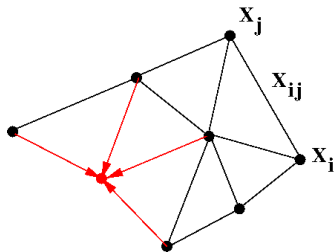
Theory and Formulation



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

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

Theory and Formulation

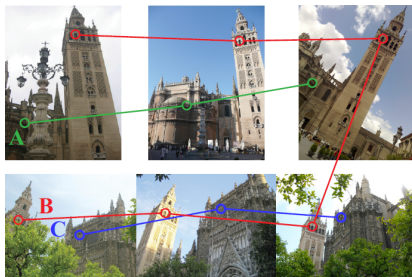


Distributed Averaging on $\mathbb{SO}(3)$

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

Outliers in data

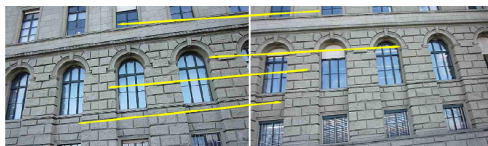
- Outliers are ubiquitous in real data
- Estimation needs to be robust
- Large body of literature in statistics and computer vision
- Gross errors in camera motion estimates
- Multiple sources of outliers that cause errors



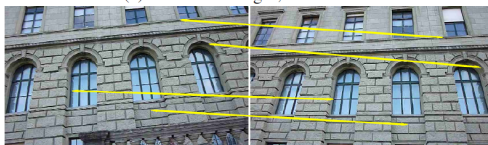
Outliers due to symmetric structures

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

Figure from Wilson & Snavely "Network Principles for SfM"



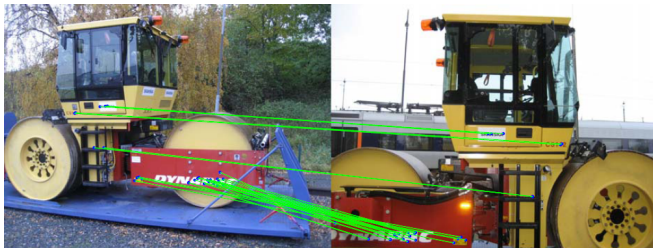
(a) Unrelated images, 228 matches



(b) Snapped to the wrong repetition, 331 matches

Outliers in data

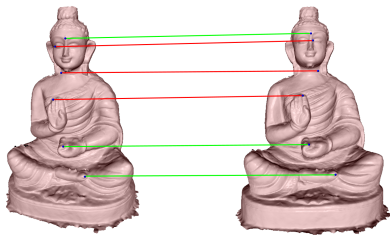
- Repeated structures are common in man-made objects
- Translational symmetry
- Results in wrong local feature matches across views



Outliers in data

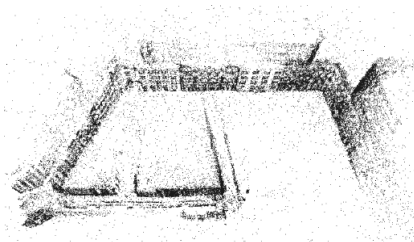
- Wrong matches based on local features
- Outliers are geometrically consistent
- Impossible to disambiguate using epipolar relationships
- Need global inference of consistency of motion estimates

From “Non-sequential Structure from Motion”

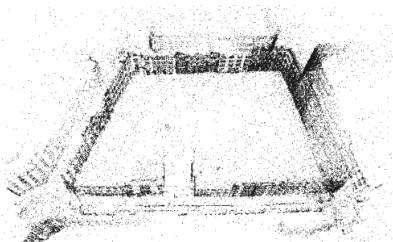


Outliers in data

- 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



(a) 3D model using all visual relations

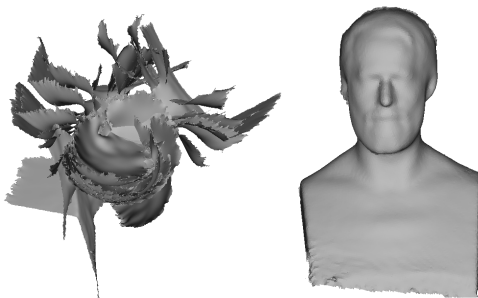


(b) 3D model using only consistent relations

Errors in viewgraph relations

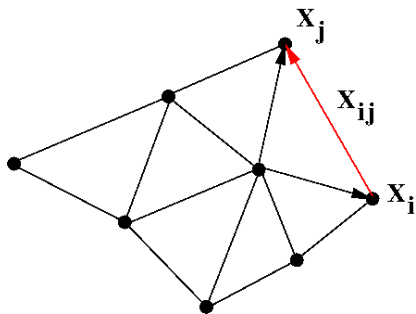
- Multiple sources of errors in low-level feature matching
- Impossible to disambiguate outlier correspondences in image pairs
- Results in erroneous relative motion relationships
- Sparse errors are easy to identify

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



Errors in viewgraph relations

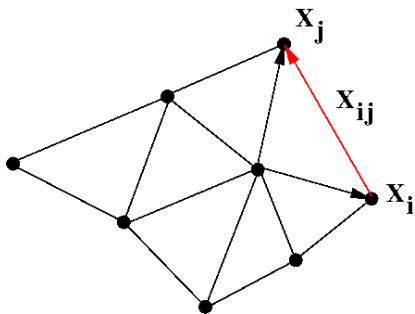
- Multiple sources of errors in low-level feature matching
- Impossible to disambiguate outlier correspondences in image pairs
- Results in erroneous relative motion relationships
- Sparse errors are easy to identify



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

Errors in relative relationships

- Individual observations can be highly erroneous

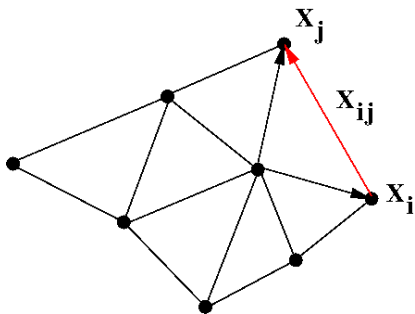


$$M_j M_i^{-1} \neq M_{ij}$$

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

Errors in relative relationships

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



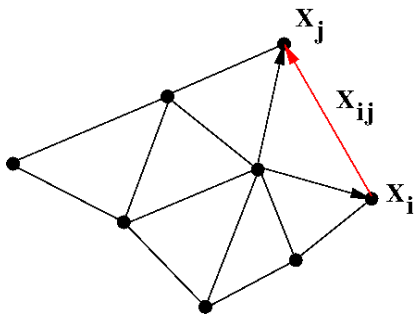
$$M_j M_i^{-1} \neq M_{ij}$$

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

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

Errors in relative relationships

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



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

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

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

Errors in relative relationships

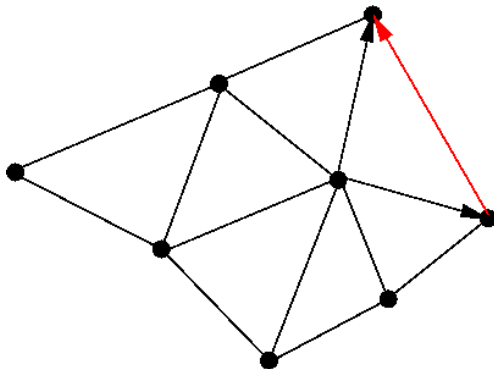
- Individual observations can be highly erroneous
- Observations inconsistent with other edges
- Will contribute outlier equation in Lie algebra representation
- Least squares solution is non-robust

Robustness 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

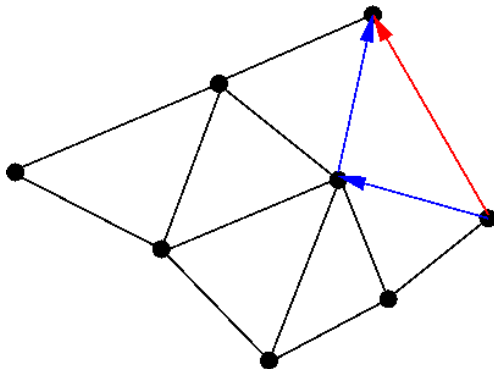
Outliers in Motion Averaging

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



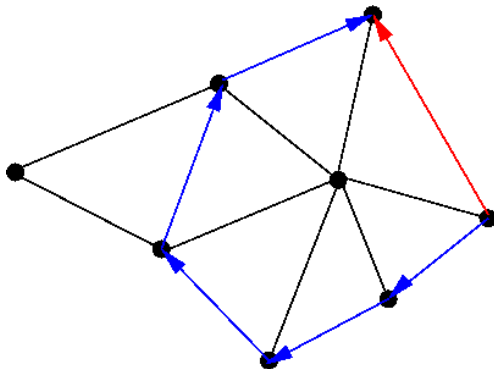
Outliers in Motion Averaging

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



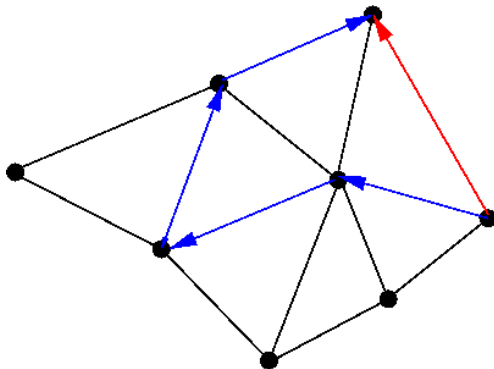
Outliers in Motion Averaging

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



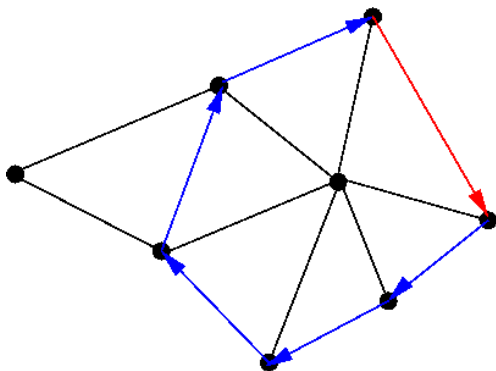
Outliers in Motion Averaging

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



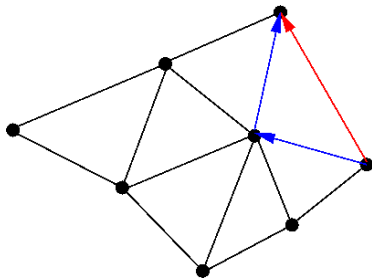
Outliers in Motion Averaging

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



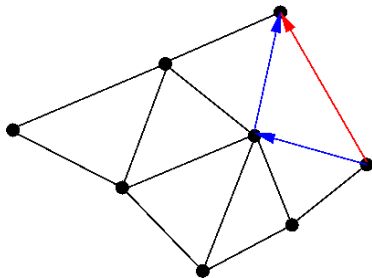
Outliers in Motion Averaging

- 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



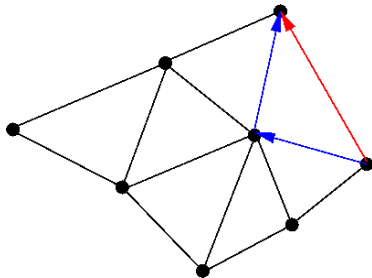
Outliers in Motion Averaging

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



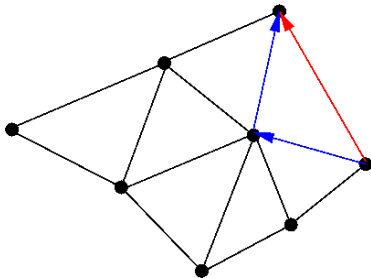
Outliers in Motion Averaging

- 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



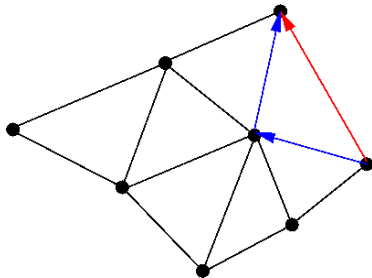
RANSAC for Motion Averaging

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



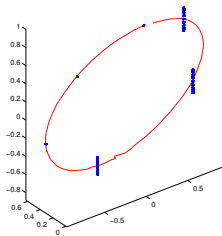
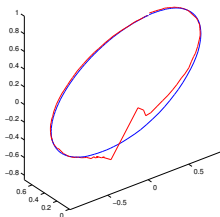
RANSAC for Motion Averaging

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



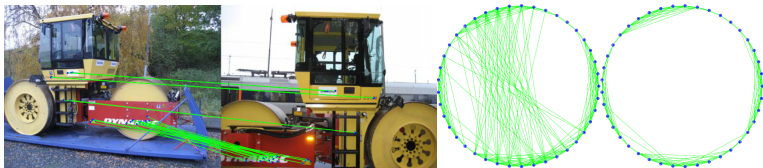
RANSAC for Motion Averaging

- 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



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



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

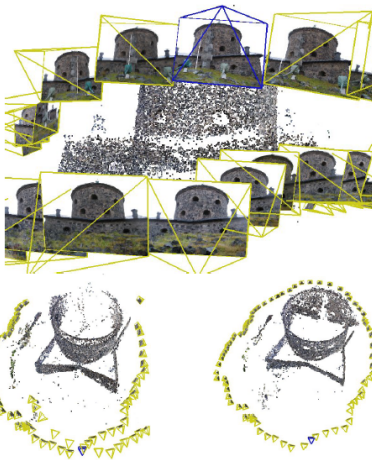
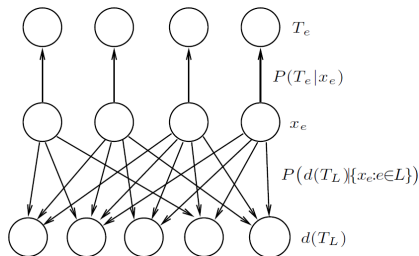
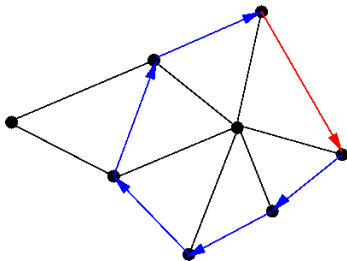


Figure 5. CASTLE. The top and left figures shows the 3D result from BUNDLER and on the right, our result is given. By careful inspection, one can see that the top and bottom image rows of the top figure display *different* facades of the castle. (A window is blocked by stairs in the top row.) This confusion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



Outlier detection using loop statistics

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

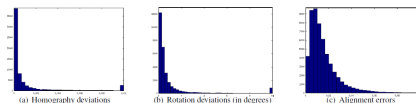


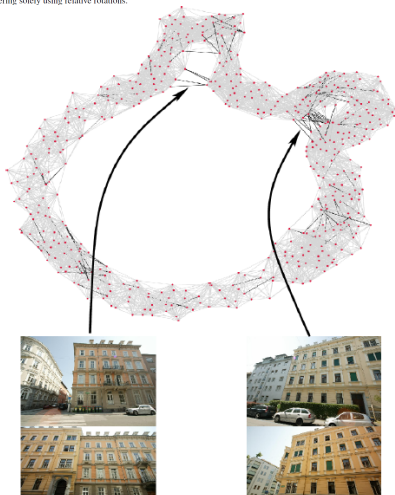
Figure 4. Histograms of empirical deviations from the respective identity map. The inlier portion of the histogram roughly follows an exponential distribution.

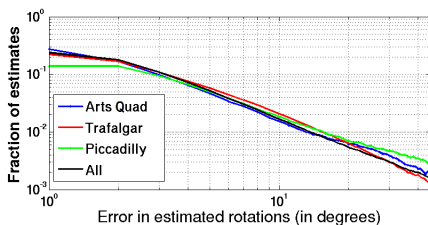
Outlier detection using loop statistics

- 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)



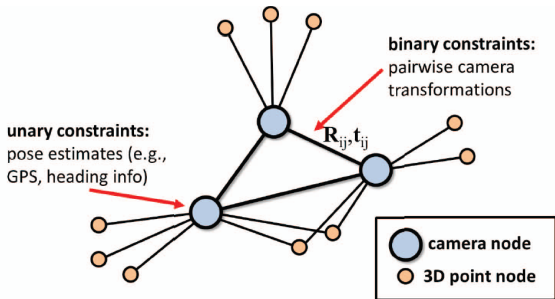
(a) W/o edge filtering (143 views registered) (b) With edge filtering (all 189 views registered)
 Figure 6. Model generated by Bundler for a facade with highly repetitive elements (a) without using epipolar graph filtering, and (b) with epipolar filtering solely using relative rotations.





Estimation with Outliers

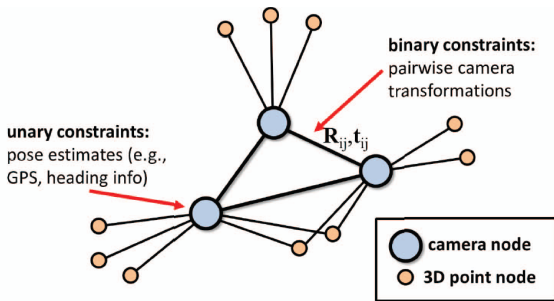
- Outlier classification assumes separability
- Real-world data does not show clear separation of inlier/outliers
- No good choice of classification threshold
- Preferable to estimate in the presence of outliers
- Need for robust estimation directly on Lie groups ($\mathbb{SO}(3)$)
- Will consider three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



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

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

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

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

Solve:

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

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

- Quantization of $\mathbb{SO}(3)$ and \mathbb{R}^3 into “cells” (labels)
- Rotation estimation = label assignment problem (coarse)
- Solved using large-scale Bayesian inference (discrete BP on MRF)
- 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



TABLE 2
Comparison with Incremental BA in terms of median differences for point positions and camera poses.

Dataset	Rotational difference			Translational difference			Point difference	
	Our approach			Our approach			Our approach	Final BA
	BP	NLLS	Final BA	Geotags	BP	NLLS	Final BA	Final BA
Acropolis	14.1°	1.5°	0.2°	12.9m	8.1m	2.4m	0.1m	0.2m
Quad	4.7°	4.6°	0.2°	15.5m	16.6m	14.2m	0.6m	0.5m
Dubrovnik	9.1°	4.9°	0.1°	127.6m	25.7m	15.1m	1.0m	0.9m
CentralRome	6.2°	3.3°	1.3°	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						Incremental BA
	Rot BP	Rot NLLS	Trans BP	Trans NLLS	Bund Adj	Total	
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

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

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

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 $\mathbf{x}_k \in \mathbb{R}^n$

Geometric distance minimiser is

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

Gradient of cost function is

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

Gradient is sum of unit vectors towards \mathbf{y}

Gradient of cost function is

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

The gradient descent step becomes

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

Weiszfeld uses a specific step size

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

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

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

This yields the update

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

Weiszfeld method

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

Theory and Formulation

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

Input : $\{\mathbf{R}_1, \dots, \mathbf{R}_N\} \in \mathbb{SO}(3)$ (Matrix Group)

Output : $\boldsymbol{\mu} \in \mathbb{SO}(3)$ (Intrinsic Average)

Initialise : $\boldsymbol{\mu} = \mathbf{I}$ (Identity)

Do

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

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

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

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

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 $\{\mathbf{R}_1, \dots, \mathbf{R}_N\}$

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

Weiszfeld method for Rotation Averaging

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

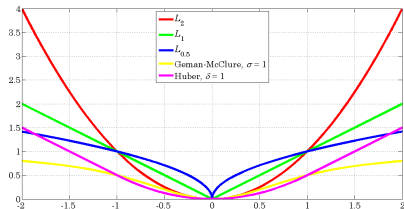
$$\text{Robust } L_1 \text{ cost: } \sum_{\mathcal{E}} d(\mathbf{R}_{ij}, \mathbf{R}_j \mathbf{R}_i^{-1})$$

Properties of Weiszfeld 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

General robust cost:

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



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

Algorithm for Motion Averaging on Lie Groups

Input : $\{M_{ij1}, M_{ij2} \dots, M_{ijn}\}$

Output : $M_g : \{M_2, \dots, M_N\}$

Set M_g to an initial guess (Linear solution)

Do

$$\Delta M_{ij} = M_j^{-1} M_{ij} M_i$$

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

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

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

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

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

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

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

M-estimators

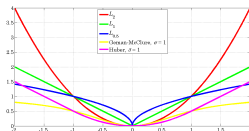
- Introduced in statistics by Huber in 1964
- Robust equivalent to the least squares solutions
- Generalises maximum likelihood estimator with tolerance for data contamination
- Vast body of literature on this topic

M-estimators

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

- 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



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

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

Iteratively Reweighted Least Squares (IRLS)

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

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

Iteratively Reweighted Least Squares (IRLS)

$$\text{Minimize } E = \sum_i \rho(\|e_i\|)$$

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

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

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

Leads to a greedy iterative solution

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

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

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

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

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

Iterative Reweighted NLSQ Problem

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

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

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

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

Iterative Reweighted NLSQ Problem

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

$$\begin{aligned}
 \mathbb{J}\mathbf{r}_{ij}(\mathbf{0}) = & \alpha_{ij} \underbrace{\left[\cdots + \mathbf{I} \cdots - \mathbf{I} \cdots \right]}_{\mathbf{A}_{ij}} \\
 & + (1 - \alpha_{ij}) \underbrace{\left[\cdots + \frac{\Delta\boldsymbol{\omega}_{ij}\Delta\boldsymbol{\omega}_{ij}^T}{\theta_{ij}^2} \cdots - \frac{\Delta\boldsymbol{\omega}_{ij}\Delta\boldsymbol{\omega}_{ij}^T}{\theta_{ij}^2} \cdots \right]}_{\mathbf{B}_{ij}} \\
 & + \frac{1}{2} \underbrace{\left[\cdots + [\Delta\boldsymbol{\omega}_{ij}]_{\times} \cdots + [\Delta\boldsymbol{\omega}_{ij}]_{\times} \cdots \right]}_{\mathbf{C}_{ij}}
 \end{aligned}$$

IRLS Form

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

Gauss-Newton form:

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

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

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 \mathbf{B}_{ij} and \mathbf{C}_{ij}
- \mathbf{A}_{ij} independent of iteration
- Consists only of terms ± 1
- Depends solely on viewgraph (compute only once)
- Yields significant speedup

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

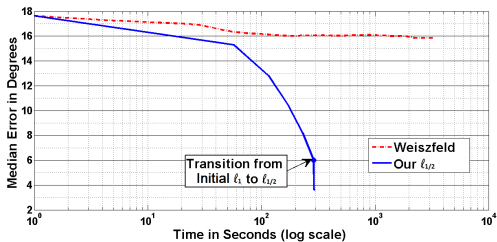
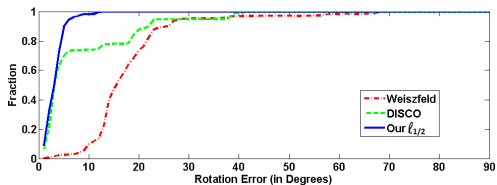
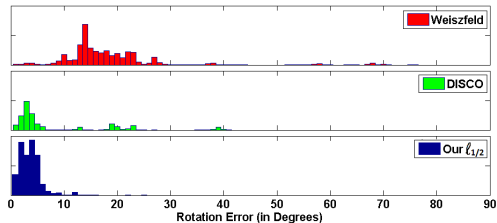
Equiv. IRLS step: $\Delta \mathbf{\Omega}_{\mathcal{V}} = - \left(\mathbf{A}^T \Phi \mathbf{A} \right)^{-1} \mathbf{A}^T \Phi \Delta \mathbf{\Omega}_{\mathcal{E}}$

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

- Can use **any** loss function $\rho(\cdot)$
- Equivalent weight function $\phi(\cdot)$
- $\mathbf{A}^T \Phi \mathbf{A}$ 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 $\rho = \ell_{\frac{1}{2}}$

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		Weiszfeld	Our		DISCO	Weiszfeld	Our		DISCO	Weiszfeld	Our	
	BP	BP+ ℓ_2		Initial ℓ_1	$\ell_{\frac{1}{2}}$			Initial ℓ_1	$\ell_{\frac{1}{2}}$			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



Extrinsic Methods

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

Intrinsic methods: $\{\mathbf{R}_1, \dots, \mathbf{R}_N\} \in \mathbb{SO}(3)$

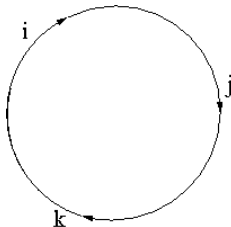
Extrinsic methods: $\{\mathbf{R}_1, \dots, \mathbf{R}_N\} \in \mathbb{R}^k$, then project onto $\mathbb{SO}(3)$

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

\times	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})
- $\mathbf{q} = q_0 + q_1\mathbf{i} + q_2\mathbf{j} + q_3\mathbf{k}$
- Constraint : $q_0^2 + q_1^2 + q_2^2 + q_3^2 = 1$
- Quaternions lie on a unit sphere
- Half sphere : Identify \mathbf{q} with $-\mathbf{q}$
- Rules of multiplication
- Inverse : $\mathbf{q}^* = q_0 - q_1\mathbf{i} - q_2\mathbf{j} - q_3\mathbf{k}$
- Identity : $\mathbf{q}\mathbf{q}^* = \mathbf{q}^*\mathbf{q} = \{1, 0, 0, 0\}$

Quaternions

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

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

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

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

Extrinsic Averaging using Quaternions

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

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

Extrinsic Averaging using Quaternions (Govindu 2001)

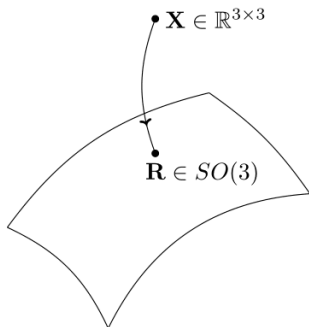
- Stack all rotations into single vector $\mathbf{q} = [\mathbf{q}_1^T, \cdots, \mathbf{q}_N^T]^T$
- Sparse linear estimation problem
- Does not enforce individual norm constraints, i.e. $\mathbf{q}_i^T \mathbf{q}_i = 1$
- Results in an extrinsic estimation that can be easily projected
- **Caveat:**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

$$\begin{aligned} Q_{ij} \mathbf{q}_i = \mathbf{q}_j &\rightsquigarrow \min \sum_{\mathcal{E}} \|Q_{ij} \mathbf{q}_i - \mathbf{q}_j\|^2 \\ \max \sum_{\mathcal{E}} \mathbf{q}_j^T Q_{ij} \mathbf{q}_i &\text{ subject to } \mathbf{q}_i^T \mathbf{q}_i = 1 \quad \forall i \in \{1 \dots N\} \\ \text{Lagrangian form: } &\|A\mathbf{q} - b\|^2 + \sum_{i \in \mathcal{V}} \lambda_i (1 - \mathbf{q}_i^T \mathbf{q}_i) \end{aligned}$$

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)

Optimal Projection



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

SVD based solution

$$\begin{aligned} \mathbf{X} &= \mathbf{U} \mathbf{S} \mathbf{V}^T \\ \Rightarrow \mathbf{R} &= \mathbf{U} \text{diag}(1, 1, |\mathbf{U} \mathbf{V}^T|) \mathbf{V}^T \end{aligned}$$

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

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

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

Relaxed to $\sum_{\mathcal{E}} \|\mathbf{R}_{ij} - \mathbf{R}_j \mathbf{R}_i^{-1}\|^2$

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 \mathbf{X}
- Project solution \mathbf{X} onto $\mathbb{SO}(3)$
- Need to incorporate robustness into estimation of \mathbf{X}
- A variety of **relaxations** available in the literature
 - Spectral Relaxation
 - Semi-definite Relaxation
 - Rank Relaxation

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

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

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

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

Spectral Relaxation

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

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

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

Semi-definite Relaxation

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

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

Semi-definite Relaxation

- \mathbf{X} should be decomposable into form $\mathbf{R}\mathbf{R}^T$
- Implies \mathbf{X} should be positive semidefinite (with \mathbf{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

$$\min_{L, S_1, S_2} \|\mathcal{P}_\Omega(\hat{X}(\mathbf{R})) - L - S_1 - S_2\|^2 + \lambda \|S_1\|_{2,1}$$

s. t. $\text{rank}(L) \leq 3$, $\text{supp}(S_1) \subseteq \Omega$, $\text{supp}(S_2) = \Omega^C$

Rank Relaxation

- Enforce \mathbf{X} to have rank of at most 3
- Matrix completion incomplete observations in \mathbf{G}
- Need to also account for
 - outliers (sparse)
 - 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:

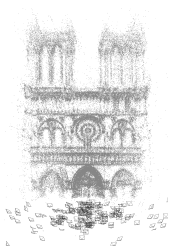
$$\min_{\mathbf{R}_1, \dots, \mathbf{R}_N \in \mathbb{SO}(3)} \sum_{\mathcal{E}} \rho(d(\mathbf{R}_{ij}, \mathbf{R}_j \mathbf{R}_i^{-1}))$$

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 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



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

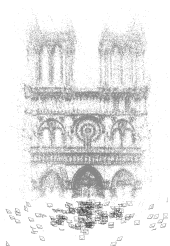
Known Rotation in large-scale SfM

- Known \mathbf{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

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

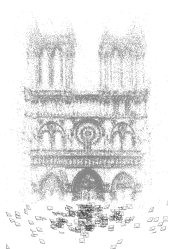


Known Rotation in large-scale SfM

- Known \mathbf{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



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

Known Rotation in large-scale SfM

- Known \mathbf{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

$$\rho(\mathbf{T}_i, \mathbf{S}_j) = \left\| \left(u_j^i - \frac{\mathbf{R}_i^1 \mathbf{S}_j + \mathbf{T}_i^1}{\mathbf{R}_i^3 \mathbf{S}_j + \mathbf{T}_i^3}, v_j^i - \frac{\mathbf{R}_i^2 \mathbf{S}_j + \mathbf{T}_i^2}{\mathbf{R}_i^3 \mathbf{S}_j + \mathbf{T}_i^3} \right) \right\|$$

$$\implies \min_{\mathbf{T}, \mathbf{S}, \gamma} \gamma$$

$$\begin{aligned} \text{s. t. } \rho(\mathbf{T}_i, \mathbf{S}_j) &\leq \gamma, \forall i, j \\ \mathbf{R}_i^3 \mathbf{S}_j + \mathbf{T}_i^3 &\geq 1, \forall i, j \\ \mathbf{T}_1 &= (0, 0, 0) \end{aligned}$$

Known Rotation in large-scale SfM

- Known \mathbf{R}_i linearizes the image projection equation
- Problem is now linear in translation and 3D structure
- Denote structure as $\mathbf{S} = [X, Y, Z]^T$
- 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.

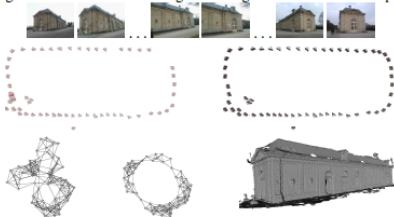


Figure 5. Top: excerpt of the Orangerie dataset. Center: Bundler camera positions (cycle failure), and ours. Bottom: input epipolar graph, our cleaned graph, and mesh obtained from our calibration.

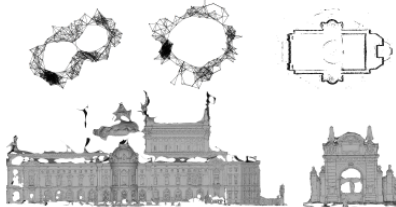


Figure 6. Opera dataset (160 images). Top: input epipolar graph (corrupted by façade symmetries), our cleaned graph, and calibration point cloud. Bottom: orthographic façade and close-up.

Translation Averaging

Recall epipolar relationship

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

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

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

Linear Averaging of Heading Directions

- Collect all absolute translations into $\mathbb{T} = [\mathbf{T}_1; \mathbf{T}_2; \cdots; \mathbf{T}_N]$
- 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

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

Ideally, should use

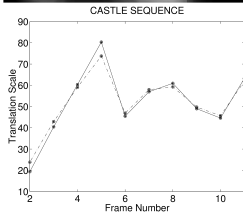
$$\mathbf{t}_{ij} = \frac{\mathbf{T}_j - \mathbf{R}_{ij}\mathbf{T}_i}{\|\mathbf{T}_j - \mathbf{R}_{ij}\mathbf{T}_i\|}$$

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
$$[\lambda_{ij} \mathbf{t}_{ij}]_{\times} [\mathbf{T}_j - \mathbf{R}_{ij} \mathbf{T}_i] = \mathbf{0}$$
- Update $\lambda_{ij} \leftarrow \frac{1}{\|\mathbf{T}_j - \mathbf{R}_{ij} \mathbf{T}_i\|}$
- Repeat till convergence



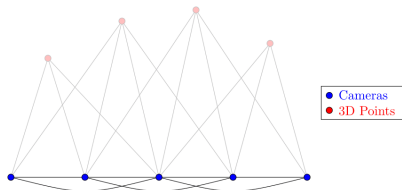
Reweighted Least Squares Averaging

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

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



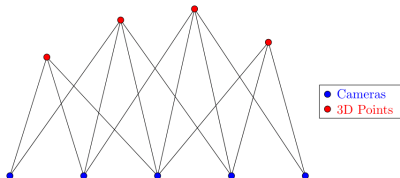
$$\begin{aligned} T_j - R_{ij}T_i &\parallel t_{ij} \\ \Rightarrow T_j - R_j R_i^T T_i &\parallel t_{ij} \\ \Rightarrow R_j^T T_j - R_i^T T_i &\parallel R_j^T t_{ij} \\ \Rightarrow C_i - C_j &\parallel R_j^T t_{ij} \end{aligned}$$

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 P^k imaged at p_i^k in i -th camera

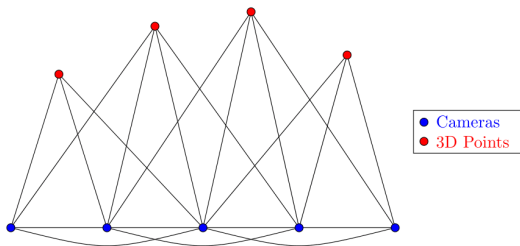


$$\begin{aligned} p_i^k &\parallel R_i P^k + T_i \\ \Rightarrow P^k + R_i^T T_i &\parallel R_i^T p_i^k \\ \Rightarrow P^k - C_i &\parallel R_i^T p_i^k \end{aligned}$$

Camera-point relationships

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

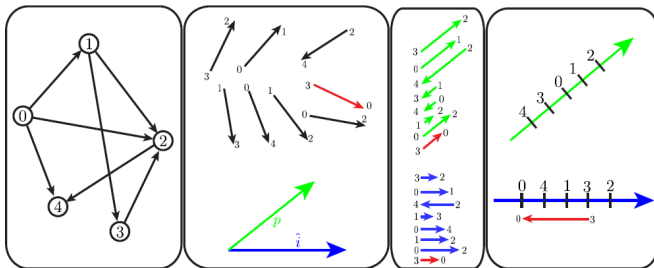
Translation Averaging



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”

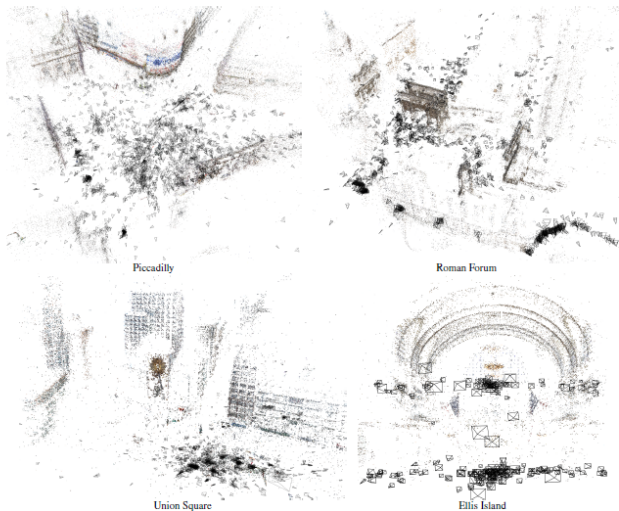
- Identify outliers and remove
- Edge consistency as an embedding problem
- Embedding in 1D \implies ordering constraints
- Find ordering most consistent pairwise constraints
- Solved as Minimum Feedback Arc Set problem

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

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

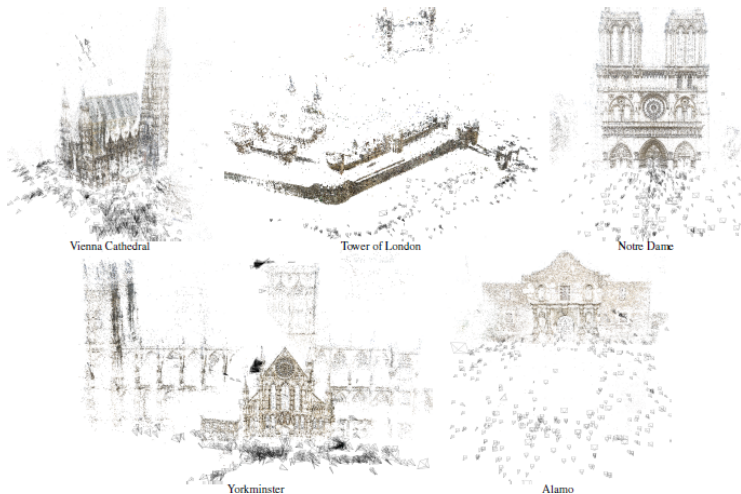
- Use chordal distance as metric to be minimized
- Solve as a non-linear minimization problem (Ceres)
- Huber loss function helps
- Properties
 - In noise-free case, cost function non-decreasing away from \mathbb{T}_0
 - Contribution of each edge is the same (unlike other costs)
 - Does not suffer from 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”

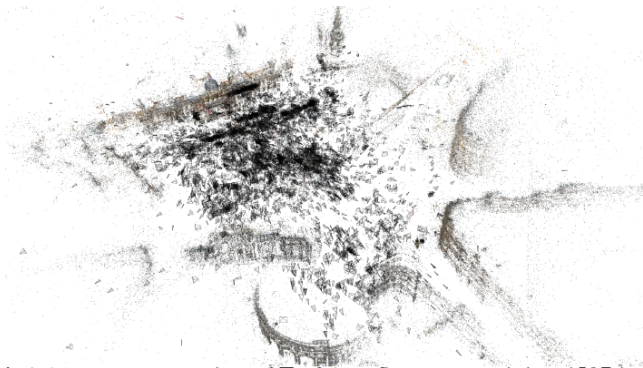


Fig.4. A large reconstruction of Trafalgar Square containing 4597 images.

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

Translation Averaging

Camera-camera

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

Camera-point

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

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

Existence of solution

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

Translation Averaging

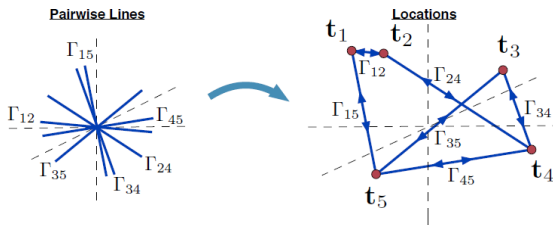


FIG. 1.1. A (noiseless) instance of the line estimation problem in \mathbb{R}^3 , with $n=5$ locations and $m=6$ pairwise lines.

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

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 .

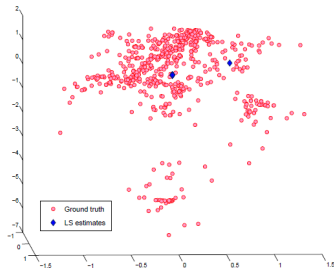
Parallel Rigidity

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

From Ozyesil *et al.* 2015

Translation Averaging

$$\begin{aligned} \min_{\mathbb{T}} \quad & \sum_{\mathcal{E}} (\mathbf{T}_j - \mathbf{T}_i)^T \mathbf{Q}_{ij} (\mathbf{T}_j - \mathbf{T}_i) \\ \text{s. t.} \quad & \sum_i \mathbf{T}_i = \mathbf{0}, \sum_i \|\mathbf{T}_i\|_2^2 = 1 \end{aligned}$$



Linear Solutions

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

Translation Averaging

$$\begin{aligned} \min_{\mathbb{T}} \sum_{\mathcal{E}} (\mathbf{T}_j - \mathbf{T}_i)^T \mathbf{Q}_{ij} (\mathbf{T}_j - \mathbf{T}_i) \\ \text{s. t. } \sum_i \mathbf{T}_i = \mathbf{0}, \sum_i \|\mathbf{T}_i\|_2^2 = 1 \end{aligned}$$

$$\begin{aligned} \min_{\mathbb{T}} \sum_{\mathcal{E}} \text{Tr} \left(\mathbf{Q}_{ij} (\mathbf{T}_i - \mathbf{T}_j) (\mathbf{T}_i - \mathbf{T}_j)^T \right) \\ \text{s. t. } \sum_i \mathbf{T}_i = \mathbf{0}, \sum_i \|\mathbf{T}_i - \mathbf{T}_j\|_2^2 > 1 \end{aligned}$$

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)

Results from Ozyesil & Singer 2015 (no BA)

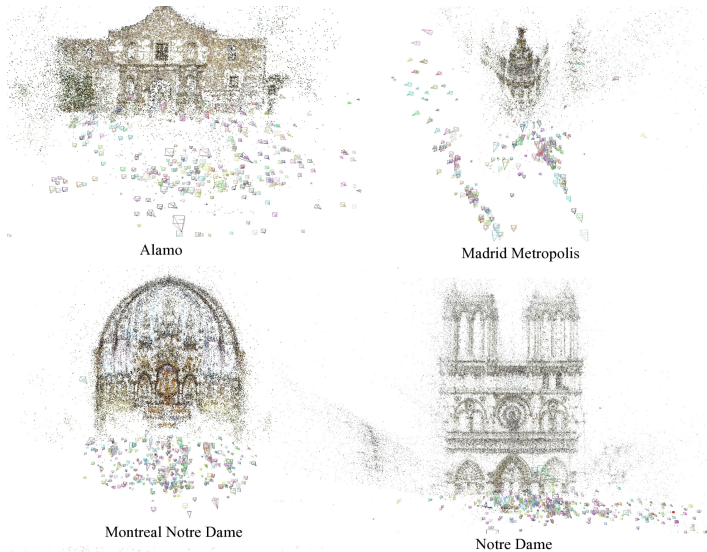


Figure 6. Snapshots of selected 3D structures computed using the camera location estimates of the LUD solver (7) (without bundle adjustment). Each 3D point is visible through at least three cameras.

$$\begin{aligned} & \min_{\mathbb{T}} \sum_{\mathcal{E}} \|Q_{ij}(\mathbf{T}_j - \mathbf{T}_i)\|_2 \\ \text{s. t. } & \sum_i \mathbf{T}_i = \mathbf{0}, \sum_{\mathcal{E}} \langle \mathbf{T}_i - \mathbf{T}_j, \mathbf{t}_{ij} \rangle = 1 \end{aligned}$$

ShapeFit & ShapeKick

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

Translation Averaging

Wilson 1DSfM

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

Compares heading directions

Ozyesil LUD

$$\begin{aligned} \min_{\mathbb{T}, \lambda} \quad & \sum_{\mathcal{E}} \|\mathbf{T}_i - \mathbf{T}_j - \lambda_{ij} \mathbf{t}_{ij}\|_2^1 \\ \text{s. t.} \quad & \sum_i \mathbf{T}_i = \mathbf{0}; \lambda_{ij} \geq c \end{aligned}$$

Compares displacement distances

ShapeFit

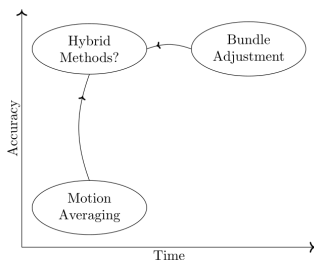
$$\begin{aligned} \min_{\mathbb{T}} \quad & \sum_{\mathcal{E}} \|Q_{ij}(\mathbf{T}_j - \mathbf{T}_i)\|_2 \\ \text{s. t.} \quad & \sum_i \mathbf{T}_i = \mathbf{0}, \sum_{\mathcal{E}} \langle \mathbf{T}_i - \mathbf{T}_j, \mathbf{t}_{ij} \rangle = 1 \end{aligned}$$

Minimize projection in complementary direction

Translation Averaging

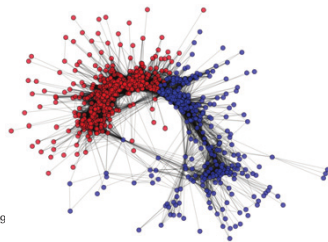
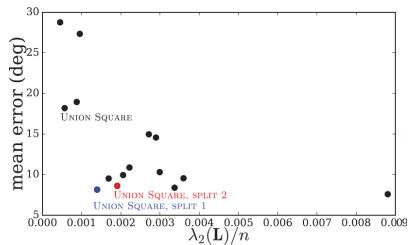
General Observations on Translation Averaging

- Geometry of $\mathbf{T}_{ij} \in \mathcal{S}^2$ is simple
- Ill-posedness of relating direction measurements with absolute displacements
- Variety of design issues due to representation discrepancy
- What should we compare ? Direction or distance ?
- Scaling constraint plays key role
- \mathbf{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
- Robustness using convex ℓ_1 or least unsquared forms
- “Baseline Desensitizing In Translation Averaging” (CVPR 2018)



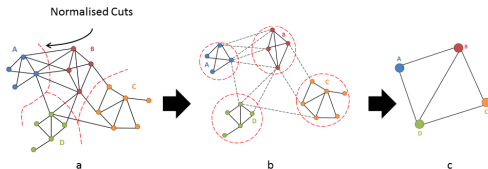
Motion Averaging in SfM

- After averaging, 3D structure estimated by triangulation
- Inputs for averaging come from epipolar geometry
- Lower accuracy due to structure being factored out (algebraic)
- Optimal estimation is in terms of reprojection error (BA)
- Motion averaging used as good initial guess for batch BA
- Can we do better with hybrid methods ?
- Interleaved synthesis of motion averaging and bundle adjustment



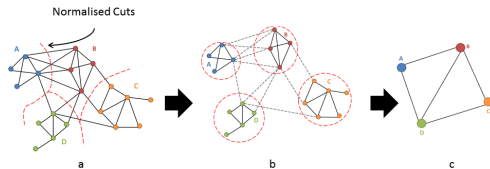
Divide and Conquer

- Recall the “hardness” measure in Wilson *et al.* 2016
- Strong connectivity within subsets makes problem easier
- Suggests that subsets of image collections are easier to solve
 - high connectivity within subset
 - smaller BA problems are easier to solve



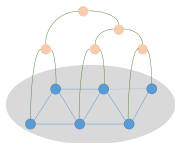
Divide and Conquer via Graph Partitioning

- Edge weights depend on feature matches between image pairs
- Cuts \implies within-class images strongly connected
- Can be easily solved by BA
- Each BA solution in independent frame of reference
- Need to scale and align all in a common frame of reference
- Utilize epipolar relationships across clusters
- Aggregate across-cluster epipolar relations
- Results in a motion averaging problem



Issues with clustering solution

- How many clusters ?
- Camera-camera edges across clusters of poor quality
- Errors within clusters need rectification



Agglomerative method of Bhowmick *et al.* CVIU 2017

- Tree structure instead of clusters on graph
- Bottom-up agglomerative clustering
- Solve BA problem for leaf node vertices (base clusters)
- Align two child vertices and merge
- Refine merged reconstruction if needed
- Move up tree to root vertex
- **Big Bonus:** Matching costs reduced due to within cluster search
- Distributed camera model along similar lines Sweeney *et al.* 2016
- Alignment can also be done using 3D Procrustes method

Chen *et al.* "Efficient tree-structure SfM ..." 2017

Hierarchical SfM



(a) Reconstruction failure by VSFM

(b) Successful reconstruction using epipolar registration.

Figure 5: (a) VSFM fails to reconstruct due to insufficient 3D-2D match during re-sectioning. (b) Our approach correctly reconstruct using epipolar registration.

Table 5: Camera rotation (in degrees) and translation distance (in ratio of graph diameter) between our method and VSFM [1].

Dataset	Rotation		Translation	
	<i>Mean</i>	<i>Median</i>	<i>Mean</i>	<i>Median</i>
Central Rome	0.001	0.0005	0.0031	0.0017
Vitthala Temple	1.25	1.12	0.0100	0.0070
St Peter's Basilica	1.3	0.54	0.0069	0.0020
Art's Quad	1.2	0.7	0.03	0.01
Colosseum	0.69	0.53	0.002	0.001
Piazza Bra [NEW]	2.08	1.58	0.021	0.013
SanGiacomo	0.53	0.25	0.001	0.0050

Table 7: Computational time of our method on different datasets compared with VSFM on a single processor.

Dataset	Match graph creation using vocabulary tree, geometric refinement and clustering (mins)	Total reconstruction and registration time for all clusters (mins)	BBA refinement using PBA and re-section (mins)	Total time by our method (mins)	Pairwise matching by VSFM (mins)	Reconstruction by VSFM (mins)	Total time by VSFM (mins)
Central Rome	1358	312	35	1705	N/A	N/A	N/A
Vitthala Temple	592	80	5	677	9522	59	9581
St. Peter's Basilica	108	28	1	137	1385	10	1395
Art's Quad	874	156	11	1041	N/A	N/A	N/A
Colosseum	102	21	1	124	1231	29	1260
Piazza Bra [NEW]	30	2	1	33	110	5	115
San Giacomo	24	2	1	27	101	5	106



(a) Architectural plan view of the Vitthala temple, Hampi.



(b) Reconstruction of Vitthala temple, Hampi



(c) Reconstruction overlaid on Google map.

Figure 9: Reconstruction of Vitthala temple, Hampi.



Hierarchical SfM



Euclidean Motion Averaging

Euclidean Motion

$$Q = RP + T$$

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

Homogeneous form

$$\begin{bmatrix} Q \\ 1 \end{bmatrix} = \overbrace{\begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix}}^M \begin{bmatrix} P \\ 1 \end{bmatrix}$$

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

$$Q = RP + T$$

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

Homogeneous form

$$\begin{bmatrix} Q \\ 1 \end{bmatrix} = \overbrace{\begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix}}^M \begin{bmatrix} P \\ 1 \end{bmatrix}$$

- 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

$$\begin{aligned} Q &= R_2(R_1P + T_1) + T_2 \\ \Rightarrow M_2M_1 &= \left[\begin{array}{c|c} R_2 & T_2 \\ \hline \mathbf{0} & 1 \end{array} \right] \left[\begin{array}{c|c} R_1 & T_1 \\ \hline \mathbf{0} & 1 \end{array} \right] \\ &= \left[\begin{array}{c|c} R_2R_1 & R_2T_1 + T_2 \\ \hline \mathbf{0} & 1 \end{array} \right] \end{aligned}$$

Now consider

$$\begin{aligned} R_2R_1 &= I \Rightarrow R_2 = R_1^{-1} \\ R_2T_1 + T_2 &= \mathbf{0} \Rightarrow T_2 = -R_1^{-1}T_1 \end{aligned}$$

- Implies that M_2M_1 and 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 $M_2M_1 \neq M_1M_2$

Euclidean Motion Averaging

$\mathfrak{se}(3)$

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

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

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

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: $d(\mathbf{X}_1, \mathbf{X}_2) = d(\mathbf{X}\mathbf{X}_1, \mathbf{X}\mathbf{X}_2) \quad \forall \mathbf{X} \in \mathbb{G}$
- Right-invariant: $d(\mathbf{X}_1, \mathbf{X}_2) = d(\mathbf{X}_1\mathbf{X}, \mathbf{X}_2\mathbf{X}) \quad \forall \mathbf{X} \in \mathbb{G}$
- 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

$$\begin{bmatrix} Q \\ 1 \end{bmatrix} = \overbrace{\begin{bmatrix} R & T \\ \mathbf{0} & 1 \end{bmatrix}}^M \begin{bmatrix} P \\ 1 \end{bmatrix}$$

Left invariance

$$d(MM_1, MM_2) = d(M_1, M_2)$$

$$Q'_1 = MM_1P = M(Q_1)$$

Right invariance

$$d(M_1M, M_2M) = d(M_1, M_2)$$

$$Q'_1 = M_1(MP)$$

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

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

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

- Sketch of argument
- Define quadratic form on Lie algebra
- Require preservation of quadratic form under transformation
- Resulting constraint lacks positive semi-definiteness
- Implies that $\text{SE}(3)$ 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 $M_1, M_2 \in \mathbb{SE}(3)$

$$\begin{array}{ccc} M_1 = \left[\begin{array}{c|c} R_1 & T_1 \\ \hline \mathbf{0} & 1 \end{array} \right] & M_2 = \left[\begin{array}{c|c} R_2 & T_2 \\ \hline \mathbf{0} & 1 \end{array} \right] & \\ & \text{versus} & \\ M_1 = \left[\begin{array}{c|c} R_1 & 100 T_1 \\ \hline \mathbf{0} & 1 \end{array} \right] & M_2 = \left[\begin{array}{c|c} R_2 & 100 T_2 \\ \hline \mathbf{0} & 1 \end{array} \right] & \end{array}$$

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

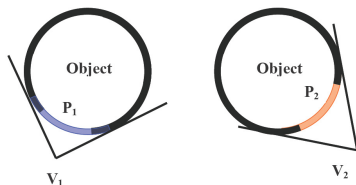
Euclidean Motion Averaging



Scans from multiple orientations

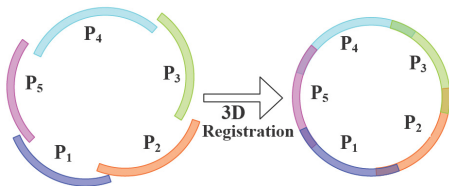
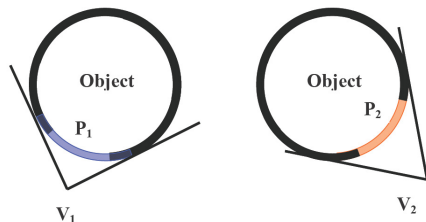
Figure from Stanford Repository

Euclidean Motion Averaging



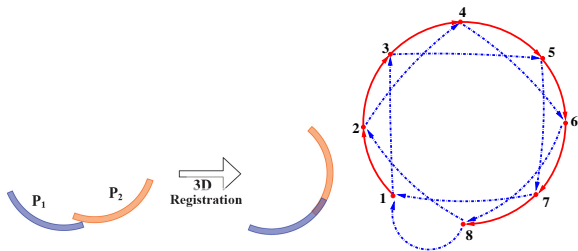
- Each scan is a *partial* model
- Has own *local* frame of reference

Euclidean Motion Averaging



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

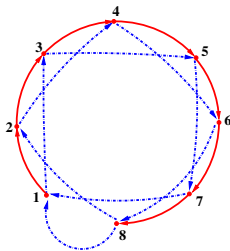
Euclidean Motion Averaging



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)
- $M_8 = M_{78}M_{67} \cdots M_{12}I$
- **Motion Averaging** : Spanning tree + Blue dashed edges
- M_{81} acts as an ‘anchor’
- Reduces drift significantly

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

Motion Averaged ICP

Given

- scans $\{P_1, P_2, \dots, P_N\}$
- 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 M_{ij} ’s to get global motion M
 - Use averaged solution

$$\forall (i, j) \in \mathcal{E} \quad M_{ij} \leftarrow M_j M_i^{-1}$$











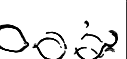



















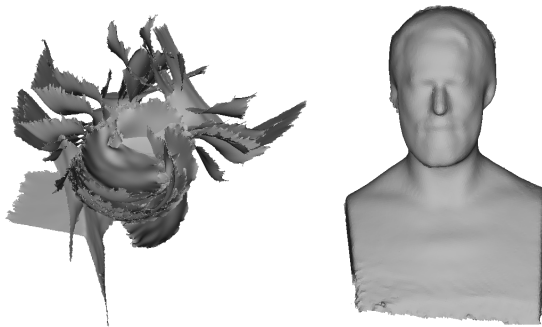
Happy Buddha 					
Dragon 					
Stanford Bunny 					
Ohio Pooh 					
Ohio Bunny 					
Model	Initial	ICP	Benjemaa	Sharp	MAICP

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



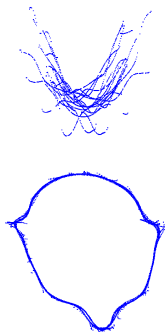
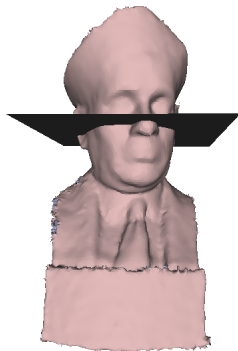
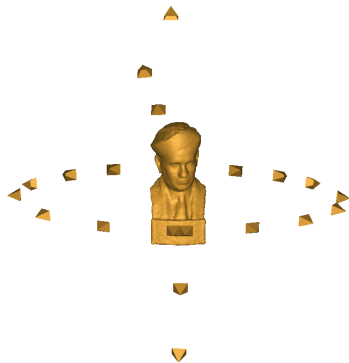
Alignment using ℓ_2 and robust motion averaging

Motion Averaged ICP

- Robust global motion estimation using ℓ_1 optimization

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

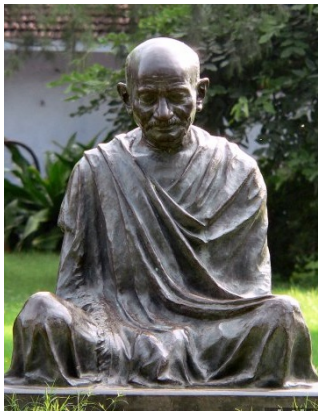
Euclidean Motion Averaging



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

$$M_1 M_2 \in \mathbb{SE}(3)$$
$$M_{ij} = M_j M_i^{-1}$$

Transformation of \mathbb{R}^3

$$P_k = M_k P_0$$
$$\Rightarrow \min \sum \|M_i^{-1} P_i - M_j^{-1} P_j\|^2$$

or $\min \sum \rho(\|M_i^{-1} P_i - M_j^{-1} P_j\|)$

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

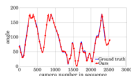
Euclidean Motion Averaging

TABLE: Mean distance from ground truth surface for full reconstruction in METERS. Running time measured on Intel Core i7-5960X 3 GHz processor with 32 GB RAM in SECONDS

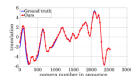
Dataset	Mean distance			Running time		
	ICP+MA	Zhou <i>et al.</i>	Our method	ICP+MA	Zhou <i>et al.</i>	Our method
livingroom1_noisy	0.10	0.05	0.05	14,260	7,460	1,380
livingroom2_noisy	0.07	0.06	0.06	11,680	6,110	970
office1_noisy	0.08	0.03	0.03	13,370	6,990	1,270
office2_noisy	0.08	0.05	0.05	12,620	6,600	1,050
harvard_c8/hv_c8_3	0.07	0.05	0.05	4,900	2,620	950
mit_32_d507/d507_2	0.07	0.04	0.04	26,590	14,240	5,500



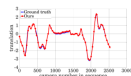
FIGURE: Registration alignment achieved by the different methods on a sample pair of scans from mit_32_d507/d507_2



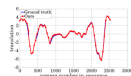
Comparison of recovered rotation angle with ground truth



Comparison of recovered translation along x-coordinate with ground truth



Comparison of recovered translation along y-coordinate with ground truth



Comparison of recovered translation along z-coordinate with ground truth

FIGURE: Comparison of full camera trajectory recovered by our method with the ground truth for office2_noisy

Averaging on SE(3) Group

- Can solve relative motions accurately using planar structures
- Combine both types of group actions for averaging
- Works well in distributing errors
- Fast and accurate

Euclidean Motion Averaging

$$\mathbf{X} = \begin{pmatrix} \mathbf{I}_4 & \mathbf{M}_{12} & \cdots & \mathbf{M}_{1N} \\ \mathbf{M}_{21} & \mathbf{I}_4 & \cdots & \mathbf{M}_{2N} \\ \cdots & \cdots & \cdots & \cdots \\ \mathbf{M}_{N1} & \mathbf{M}_{N2} & \cdots & \mathbf{I}_4 \end{pmatrix}$$
$$\mathbf{M} = \begin{bmatrix} \mathbf{M}_1 \\ \mathbf{M}_2 \\ \cdots \\ \mathbf{M}_N \end{bmatrix}, \mathbf{M}^{-b} = \begin{bmatrix} \mathbf{M}_1^{-1} & \mathbf{M}_2^{-1} & \cdots & \mathbf{M}_N^{-1} \end{bmatrix}$$

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

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

Extrinsic averaging becomes

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

Using rank relaxation


































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

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

- \mathbf{S} is sparse set of outliers
- Estimated \mathbf{L} only enforces rank constraint
- Need to enforce last row of $\mathbf{M}_i = [0 \ 0 \ 0 \ 1]$
- 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.

Dataset	R-GoPro	CoSIFT	L-LALM	GoPro	DIFFUSION	Shang et al.	NULL-SPACE
Bunny							
Buddha							
Dragon							
Gargoyle							
Capital							

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 $\text{SE}(3)$ Averaging

- Fundamental differences between $\text{SE}(3)$ and $\text{SO}(3)$
- 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

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

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

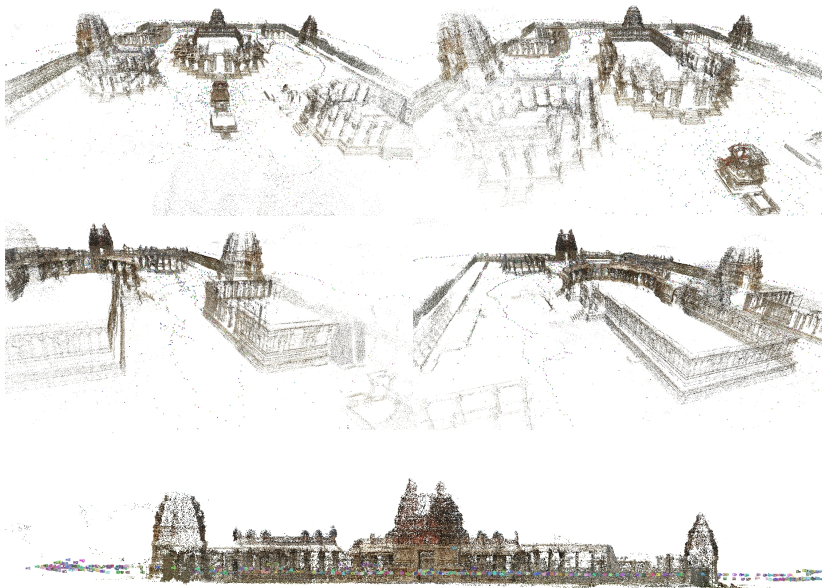
Uttaran Bhattacharya

Mohammadul Haque

Avishek Chatterjee

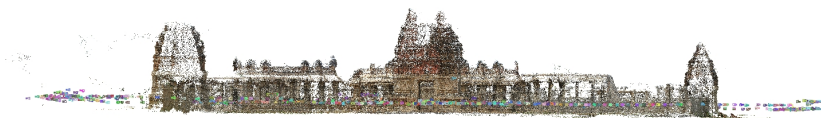
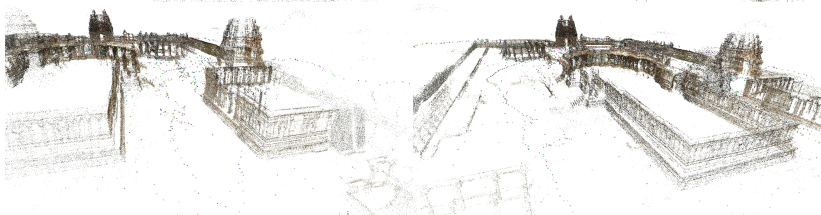
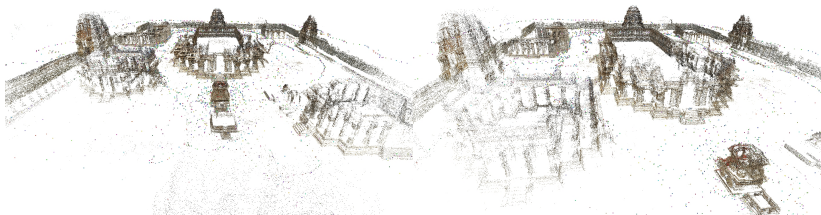
Vitthala Temple, Hampi

Rendering of our 3D model



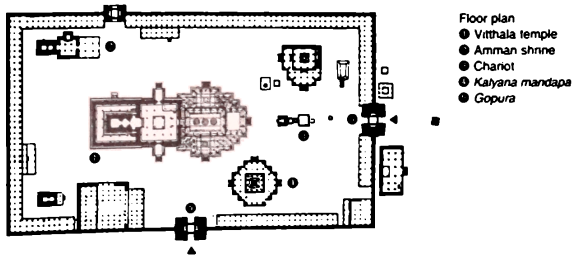
Vitthala Temple, Hampi

Rendering of our 3D model

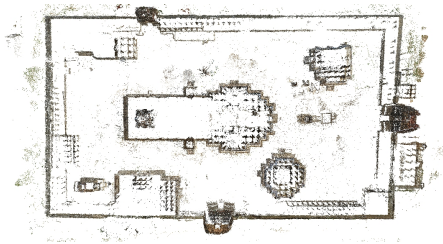


Vitthala Temple, Hampi

Comparison of Reconstruction



(a) Architectural Drawing



(b) Reconstruction

Introduction

Structure from Motion (SfM)

- Well studied problem
- Two broad approaches
 - *Algebraic Approaches* : few images, fast, low accuracy
 - *Bundle Adjustment* : many images, slow, accurate

Contrasting issues

- Motion estimation for few views
 - is fast
 - Two-view or epipolar geometry well studied
 - Structure eliminated algebraically, but not accurate
- Bundle-adjustment for large datasets
 - treats problem as non-linear optimisation
 - does not fully utilise geometric understanding
 - can be accurate
 - is rather slow (despite recent advances)
 - needs a good initial guess (major issue)

Introduction

Pin-hole Model

$$\begin{bmatrix} u_i^k \\ v_i^k \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R}_i & | & \mathbf{T}_i \end{bmatrix} \begin{bmatrix} \mathbf{X}^k \\ \mathbf{Y}^k \\ \mathbf{Z}^k \\ 1 \end{bmatrix}$$

$\underbrace{\hspace{10em}}_{\mathbf{S}}$

Reprojection Error

$$\hat{u}_i^k = \frac{\mathbf{r}_1 \cdot \mathbf{S} + \mathbf{t}_1}{\mathbf{r}_3 \cdot \mathbf{S} + \mathbf{t}_3}$$
$$\hat{v}_i^k = \frac{\mathbf{r}_2 \cdot \mathbf{S} + \mathbf{t}_2}{\mathbf{r}_3 \cdot \mathbf{S} + \mathbf{t}_3}$$

Structure from Motion

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

- i -th camera (motion), k -th 3D point (structure)
- Conventionally solved using Bundle Adjustment

Motion Representations

- 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
- Multiple Representations for 3D Rotations
 - Euler Angles
 - Quaternions
 - Rotation about an axis
 - **Lie Group**
- Euclidean Motions play central role
 - 2D-2D geometry (Image Registration, Mosaics)
 - 3D-2D geometry (Structure from Motion)
 - 3D-3D geometry (scan alignment, ICP)

Theory and Formulation

Group Multiplication

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

Action on Vector Space

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

Group Action

- $\mathbf{G} \in \mathbb{G}, \mathbf{h} \in \mathbb{H}$, direct product $(\mathbf{G}_1, \mathbf{h}_1)(\mathbf{G}_2, \mathbf{h}_2) = (\mathbf{G}_1 \mathbf{G}_2, \mathbf{h}_1 \mathbf{h}_2)$
- Group \mathbb{G} , manifold \mathcal{X} , group action $a : \mathbb{G} \times \mathcal{X} \longrightarrow \mathcal{X}$
- Satisfies
 - $a(\mathbf{E}, \mathbf{x}) = \mathbf{x} \ \forall \mathbf{x} \in \mathcal{X}$
 - $a(\mathbf{G}_1, a(\mathbf{G}_2, \mathbf{x})) = a(\mathbf{G}_1 \mathbf{G}_2, \mathbf{x}) \ \forall \mathbf{x} \in \mathcal{X} \text{ and } \mathbf{G}_1, \mathbf{G}_2 \in \mathbb{G}$
- Group action of $GL(n)$ on vector space is linear

Semi-Direct Product

- Consider group \mathbb{G} and commutative group \mathbb{H}
- Consider group action $\mathbf{G}(\mathbf{h}) : \mathbb{G} \times \mathbb{H} \longrightarrow \mathbb{H}$
- **Semi-direct product** : $\mathbb{G} \ltimes \mathbb{H}$
- $(\mathbf{G}_1, \mathbf{h}_1)(\mathbf{G}_2, \mathbf{h}_2) = (\mathbf{G}_1\mathbf{G}_2, \mathbf{h}_1 + \mathbf{G}_1(\mathbf{h}_2))$
- Special Euclidean Group $\mathbb{SE}(n)$
- $\mathbb{SE}(n) = \mathbb{SO}(n) \ltimes \mathbb{R}^n$

Injective homomorphism

$$\begin{aligned}\mathbb{SE}(n) &\longrightarrow GL(n+1) \\ (\mathbf{R}, \mathbf{t}) &\mapsto \begin{pmatrix} \mathbf{R} & \mathbf{t} \\ 0 & 1 \end{pmatrix}\end{aligned}$$

Semi-Direct Product

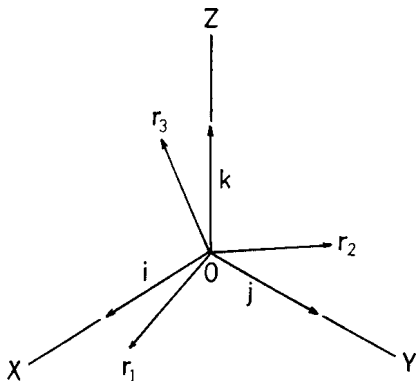
- Consider group \mathbb{G} and commutative group \mathbb{H}
- Consider group action $\mathbf{G}(\mathbf{h}) : \mathbb{G} \times \mathbb{H} \longrightarrow \mathbb{H}$
- **Semi-direct product** : $\mathbb{G} \ltimes \mathbb{H}$
- $(\mathbf{G}_1, \mathbf{h}_1)(\mathbf{G}_2, \mathbf{h}_2) = (\mathbf{G}_1\mathbf{G}_2, \mathbf{h}_1 + \mathbf{G}_1(\mathbf{h}_2))$
- Special Euclidean Group $\mathbb{SE}(n)$
- $\mathbb{SE}(n) = \mathbb{SO}(n) \ltimes \mathbb{R}^n$
- Direct $(n+1)$ dim representation available for $\mathbb{SE}(n)$

$$\begin{aligned}\mathbf{R} &= [\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3] \\ &= \mathbf{R} \begin{bmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \end{bmatrix}\end{aligned}$$

3D Rotations

- Consider linear transformation
- Canonical basis : $\mathbf{i} = (1, 0, 0)^T, \mathbf{j} = (0, 1, 0)^T, \mathbf{k} = (0, 0, 1)^T$
- Apply linear transformation
- New vectors : $\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3$
- Linear mapping defined by matrix of concatenated vectors

Theory and Formulation



$$\langle \mathbf{r}_i, \mathbf{r}_i \rangle = 1$$

$$\langle \mathbf{r}_i, \mathbf{r}_j \rangle = \delta_{ij}$$

$$|\mathbf{r}_1 \mathbf{r}_2 \mathbf{r}_3| = \langle \mathbf{r}_2 \times \mathbf{r}_3, \mathbf{r}_1 \rangle = 1$$

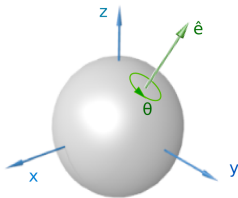
Properties of rigid 3D rotation

- Preserve vector lengths and angles between them
- Preserve parity, i.e. no reflection
- Maintain handedness of basis
- Results in condition $\mathbf{R}\mathbf{R}^T = \mathbf{R}^T\mathbf{R} = \mathbf{I}$

Properties of rigid 3D rotation

- Constraint : $\mathbf{R}\mathbf{R}^T = \mathbf{I}$
- Consider N -dimensional \mathbf{R} (N^2 entries)
- N constraints for vector lengths
- ${}^N C_2 = \frac{N(N-1)}{2}$ orthogonality constraints
- Degrees of freedom = $N^2 - N - \frac{N(N-1)}{2} = \frac{N(N-1)}{2}$
- For rotation in 2D plane, dof = 1
- For 3 dimensional vector space, dof = 3

Theory and Formulation



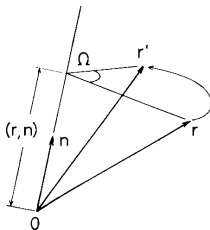
Euler's Rotation Theorem

Any rigid displacement with a fixed point is equivalent to a single rotation about an axis through the point

Figure from Wikipedia

Euler's Rotation Theorem

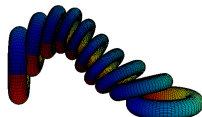
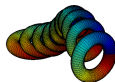
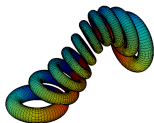
- Eigen decomposition $\mathbf{R}\mathbf{n} = \sigma\mathbf{n}$
- $\mathbf{R}\mathbf{n}^* = \sigma^*\mathbf{n}^*$
- $\langle \mathbf{R}\mathbf{n}, \mathbf{R}\mathbf{n}^* \rangle = \langle \mathbf{R}^T \mathbf{R}\mathbf{n}, \mathbf{n}^* \rangle = \langle \mathbf{n}, \mathbf{n}^* \rangle$
- $\sigma\sigma^* \langle \mathbf{n}, \mathbf{n}^* \rangle = |\sigma|^2 \langle \mathbf{n}, \mathbf{n}^* \rangle$
- Therefore $|\sigma| = 1$
- $|\sigma\mathbf{I} - \mathbf{R}| = 0$ is a polynomial of degree 3 with real coefficients
- Either all $\sigma_1, \sigma_2, \sigma_3$ real or one complex conjugate pair
- σ either $(1, 1, 1)$ or $(1, -1, -1)$
- At least one σ is equal to 1
- $\mathbf{R}\mathbf{n} = \mathbf{n}$
- Points along direction \mathbf{n} do not move
- \mathbf{n} specifies the axis of rotation



Axis-Angle representation of 3D Rotation

- \mathbf{n} specifies the axis of rotation
- Every point rotated by angle Ω about the axis \mathbf{n}
- Consider point \mathbf{r}
- $\|\mathbf{R}\mathbf{r} - \langle \mathbf{R}\mathbf{r}, \mathbf{n} \rangle \mathbf{n}\| = \|\mathbf{R}(\mathbf{r} - \langle \mathbf{R}\mathbf{n}, \mathbf{R}\mathbf{n} \rangle \mathbf{n})\| = \|\mathbf{r} - \langle \mathbf{r}, \mathbf{n} \rangle \mathbf{n}\|$
- \mathbf{n} is a unit vector (2 dof)
- Together with Ω specifies 3 dof for 3D rotation

Theory and Formulation



- Same averaging formulation applies to $\mathbb{SE}(3)$
- Euclidean metric in Lie algebra induces distance metric on group
- Given $d(\mathbf{X}_i, \mathbf{X}_j)$ we can carry out many operations
- Path interpolation, control etc. are common

From Allen-Blanchette *et al.* "Motion Interpolation in $\text{SIM}(3)$ "

Theory and Formulation

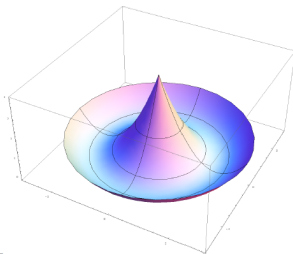
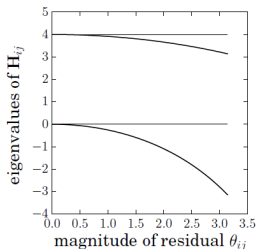
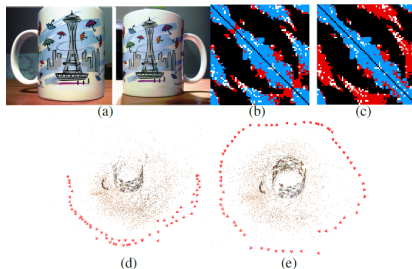


Fig. 2: The eigenvalues of the Hessian \mathbf{H}_{ij} . Fig. 3: A function which is not convex. Two are constant: $\lambda_6 = 4$ and $\lambda_3 = 0$, and because of a rotational gauge ambiguity, the other four appear in pairs.

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

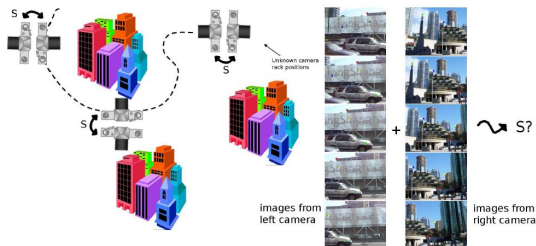
- Hessian \mathbf{H}_{ij} of $d^2(\mathbf{R}_{ij}, \mathbf{R}_j \mathbf{R}_i^{-1})$ at $(\mathbf{R}_i, \mathbf{R}_j)$
- \mathbf{H}_{ij} is positive semidefinite at $d(.,.) = 0$, indefinite elsewhere
- Important issue is gauge ambiguity
- Locally convex $\{(r, \theta) | 1 \leq r < 2\}$
- Not locally convex $\{(r, \theta) | 0 < r < 1\}$
- Actually problem $\min (r - 1)^2, 0 < r < 2$



Paper: “SfM for scenes with large duplicate structures”

- Model relative motion observation error as Gaussian distribution
- Latent variable y_i for inlier/outlier classification
- Maximize likelihood of poses instead of searching over latent space
- Can use EM, but independence assumption inadequate
- Relies on RANSAC-like sampling of weighted spanning trees

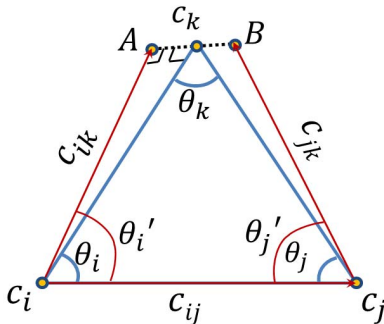
Rotation Averaging



Conjugate Rotations

- Non-overlapping camera sequences related rigidly
- Solve for single transformation between non-overlapping cameras
- Also known as the hand-eye calibration problem
- Can derive relative rotations within sequences (L_{ij}, R_{ij})
- Relative rotations related by calibration rotation S
- $R_{ij} = SL_{ij}S^{-1}$
- Can use averaging principles here

Translation Averaging



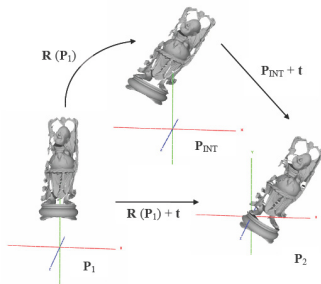
Fixing Linear Translation Averaging

- Jiang *et al.* “A Global Linear Method for Camera Pose Registration” 2013
- Geometric errors are better models than algebraic ones
- Coplanarity constraint
- Stitch triplets together

Euclidean Motion Averaging

3D Registration

- Motion model is 3D rotation \mathbf{R} and translation \mathbf{t}
- $\mathbf{R} \in SO(3)$ and $\mathbf{t} \in \mathbb{R}^3$, full motion $\in SE(3)$
- Consider cost : $\sum_{i=1}^N \|\mathbf{R}\mathbf{P}_1(i) + \mathbf{t} - \mathbf{P}_2(i)\|^2$
- Makes sense with Gaussian noise model



Euclidean Motion Averaging

$$\text{Cost} : \sum_{i=1}^N \| \mathbf{R} \mathbf{P}_1(i) + \mathbf{t} - \mathbf{P}_2(\pi(i)) \|^2$$

3D Registration : Known Correspondences

- Assume **known correspondence** : $\pi(\cdot)$ is known
- Solving for motion is **easy**
- Closed form solution exists (Umeyama etc.)

3D Registration : Known Motion

- Need $\pi(\cdot) : \mathbf{P}_1(i) \leftrightarrow \mathbf{P}_2(\pi(i))$
- apply motion to $\mathbf{P}_1(i)$
- find its closest point in \mathbf{P}_2
- justified for Gaussian noise model

Iterative Closest Point Algorithm

- 3D Registration problem is
 - **Easy** for known **motion** or **correspondence**
 - **Hard** if *both* are unknown
- Search space grows enormously if neither known
- ICP provides a *greedy* solution
- Iterative method
- Converges to local minima
- Needs a good initial guess

Iterative Closest Point Algorithm

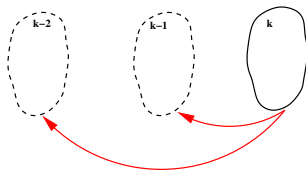
- Start at initial guess of motion
- Correspondence for **given** motion (correspondence step)
- Update motion using this correspondence (motion step)
- Iterate till convergence
- Many practical issues and refinements

ICP is the workhorse for 3D registration

Other Comparable Multiview Methods

- Bergevin *et al.* and Benjemaa *et al.* :
 - Place all scans into the common frame of reference
 - Repeatedly select a scan and register it against all scans
- Sharp *et al.* :
 - Uses graph of relative motions like us
 - Assumes every edge present in at least one cycle
 - Inconsistency accumulated by traversing the cycle
 - Error uniformly distributed over all the edges of the cycle
 - Highly limiting assumption, gives suboptimal solutions

Euclidean Motion Averaging



Causal Variant

Often need to register scans in a causal manner

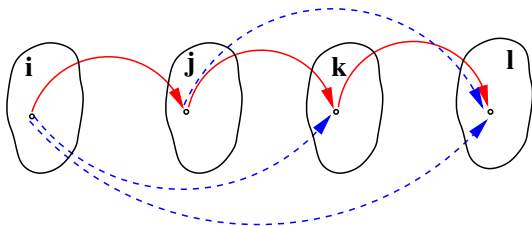
- **ICP** : $M_k = M_{k-1 \ k} M_{k-1}$
- **Causal ICP** : Two estimates for M_k

$$M_{k-2 \ k} = M_k M_{k-2}^{-1} \Rightarrow M_k = M_{k-2 \ k} M_{k-2}$$

$$M_{k-1 \ k} = M_k M_{k-1}^{-1} \Rightarrow M_k = M_{k-1 \ k} M_{k-1}$$

- M_k obtained by averaging on **SE(3)**
- Reduces drift while maintaining causality

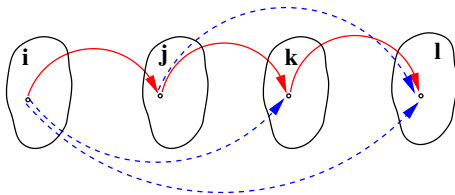
Euclidean Motion Averaging



Transitive MAICP

- π_i^j : Correspondence map from scans i to j
- **ICP**: Needs only π_i^j , π_j^k and π_k^l (red = 3)
- **MAICP**: Also requires π_i^k , π_j^l and π_i^l (red + blue = 6)
- Large computational overhead due to additional correspondences

Euclidean Motion Averaging



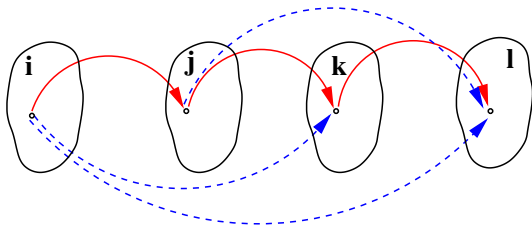
Transitive MAICP

Using transitivity

$$\begin{aligned} P_i(n) &\leftrightarrow P_k(\pi_i^k(n)) \\ P_i(n) &\leftrightarrow P_k(\pi_j^k(\pi_i^j(n))) \\ &\Rightarrow \pi_i^k = \pi_j^k \circ \pi_i^j \end{aligned}$$

- No significant additional computational load to compute π_i^k
- Extra M_{ij} 's for “free”, can be averaged

Euclidean Motion Averaging



π_i^l computed by **direct method**:

Introduces substantial error as i and l are far apart

π_i^l computed by **transitive method**:

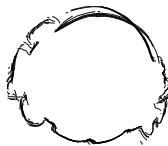
Single steps b/w adjacent scans considered from i to l $i \rightarrow j$, $j \rightarrow k$ and $k \rightarrow l$

$$\pi_i^l \approx \pi_k^l \circ \pi_j^k \circ \pi_i^j$$

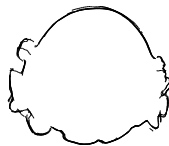
Euclidean Motion Averaging



(c) Model



(d) Initial Alignment



(e) ICP



(f) Benjemaa et al.

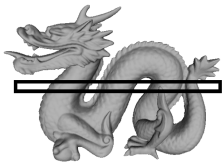


(g) Sharp et al.



(h) MAICP

Euclidean Motion Averaging



(i) Model



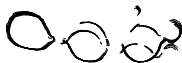
(j) Initial Alignment



(k) ICP



(l) Benjemaa et al.



(m) Sharp et al.



(n) MAICP

Euclidean Motion Averaging



(o) Model



(p) Initial Alignment



(q) ICP



(r) Benjemaa et al.



(s) Sharp et al.



(t) MAICP

Theory and Formulation

	ICP	Other Multiview Methods		Multiview Motion Averaged ICP			
		Benjemaa <i>et al.</i>	Sharp <i>et al.</i>	Closure	Causal	MAICP	Transitive
Happy Buddha	5.78	1.96	1.53	2.62	3.67	0.85	0.91
Stanford Bunny	0.92	0.61	0.61	0.78	0.72	0.59	0.55
Stanford Dragon	7.69	3.75	5.54	5.51	5.72	2.74	3.39
Ohio Bunny	1.55	0.78	0.69	0.27	1.18	0.28	0.47
Ohio Pooh	13.33	1.55	1.00	0.68	4.52	0.5	0.36

Table : Average Registration Errors for real datasets (in degrees). Our motion-averaged registration methods significantly outperform ICP and the other multiview registration methods listed.

	ICP	Other Multiview Methods		Multiview Motion Averaged ICP			
		Benjemaa <i>et al.</i>	Sharp <i>et al.</i>	Closure	Causal	MAICP	Transitive
Happy Buddha	9.73	20.47	33.86	17.69	8.64	23.09	19.41
Stanford Bunny	4.8	3.32	12.7	1.39	2.56	2.61	2.5
Stanford Dragon	13.22	24.00	41.84	12.7	7.6	25.7	22.92
Ohio Bunny	30.17	4.56	68.28	5.19	3.54	3.96	3.12
Ohio Pooh	58.3	44.8	129.18	35.66	24.65	28.54	23.28

Table : Computation time for real datasets (in seconds). Our motion-averaged registration method is faster than the other multiview registration methods listed.