3D Computer Vision

Radim Šára Martin Matoušek

Center for Machine Perception
Department of Cybernetics
Faculty of Electrical Engineering
Czech Technical University in Prague

https://cw.fel.cvut.cz/wiki/courses/tdv/start

http://cmp.felk.cvut.cz
mailto:sara@cmp.felk.cvut.cz
phone ext. 7203

rev. November 28, 2023

Open Informatics Master's Course

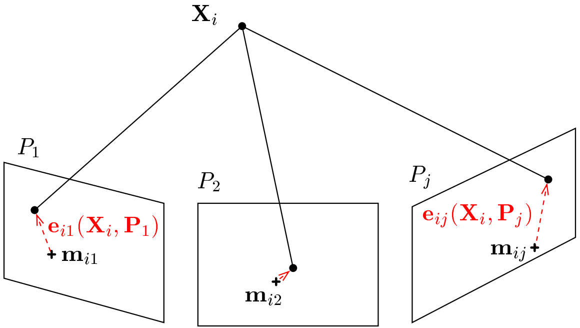
▶Bundle Adjustment

Goal: Use a good (and expensive) error model and <u>improve</u> the initial estimates of all parameters

Given:
1. set of 3D points $\{\mathbf{X}_i\}_{i=1}^{p}$
2. set of cameras $\{\mathbf{P}_j\}_{j=1}^{c}$
3. correspondence & fixed tentative projections $\mathbf{m}_{ij}$

Required:
1. corrected 3D points $\{\mathbf{X}'_i\}_{i=1}^{p}$
2. corrected cameras $\{\mathbf{P}'_j\}_{j=1}^{c}$

Latent:
1. visibility decision $v_{ij} \in \{0, 1\}$ per $\mathbf{m}_{ij}$



- for simplicity, $\mathbf{X}$, $\mathbf{m}$ are considered Cartesian (not homogeneous)
- we have projection error $\mathbf{e}_{ij}(\mathbf{X}_i, \mathbf{P}_j) = \mathbf{x}_i - \mathbf{m}_i$ per image feature, where $\underline{\mathbf{x}}_i = \mathbf{P}_j \underline{\mathbf{X}}_i$
- for simplicity, we will work with scalar error $e_{ij} = \|\mathbf{e}_{ij}\|$

Robust Objective Function for Bundle Adjustment

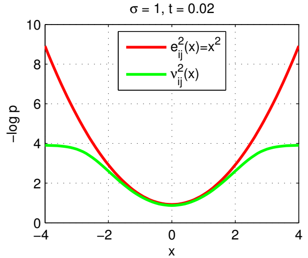
The data model is constructed by marginalization over v_{ij}, as in the Robust Matching Model →120

$$p(\{\mathbf{e}\} \mid \{\mathbf{P},\mathbf{X}\}) = \prod_{\text{pts}:i=1}^{p} \prod_{\text{cams}:j=1}^{c} \Big((1-P_0)p_1(e_{ij} \mid \mathbf{X}_i,\mathbf{P}_j) + P_0\, p_0(e_{ij} \mid \mathbf{X}_i,\mathbf{P}_j) \Big)$$

the marginalized negative log-density is (→121)

$$-\log p(\{\mathbf{e}\} \mid \{\mathbf{P},\mathbf{X}\}) = \sum_i \sum_j \underbrace{-\log\Big(e^{-\frac{e_{ij}^2(\mathbf{X}_i,\mathbf{P}_j)}{2\sigma_1^2}} + t\Big)}_{\rho(e_{ij}^2(\mathbf{X}_i,\mathbf{P}_j)) = \nu_{ij}^2(\mathbf{X}_i,\mathbf{P}_j)} \stackrel{\text{def}}{=} \sum_i \sum_j \nu_{ij}^2(\mathbf{X}_i,\mathbf{P}_j)$$

- $\boldsymbol{\theta} = \{\mathbf{P},\mathbf{X}\}$
- we can use LM, e_{ij} is the exact projection error function (not Sampson error)
- ν_{ij} is a 'robust' error fcn.; it is non-robust ($\nu_{ij} = e_{ij}$) when $t = 0$
- $\rho(\cdot)$ is a 'robustification function' often found in M-estimation
- the $\mathbf{L}_{ij}$ in Levenberg-Marquardt changes to vector



$$(\mathbf{L}_{ij})_l = \frac{\partial \nu_{ij}}{\partial \theta_l} = \underbrace{\frac{1}{1 + t\, e^{e_{ij}^2(\boldsymbol{\theta})/(2\sigma_1^2)}}}_{\text{small for } e_{ij} \gg \sigma_1} \cdot \frac{1}{\nu_{ij}(\theta)} \cdot \frac{1}{4\sigma_1^2} \cdot \frac{\partial e_{ij}^2(\theta)}{\partial \theta_l} \qquad (35)$$

but the LM method stays the same as before →110–111

- outliers (wrong v_{ij}): almost no impact on $\mathbf{d}_s$ in normal equations because the red term in (35) scales contributions to both sums down for the particular ij

$$-\sum_{i,j} \mathbf{L}_{ij}^\top \nu_{ij}(\theta^s) = \Big(\sum_{i,j}^{k} \mathbf{L}_{ij}^\top \mathbf{L}_{ij}\Big)\mathbf{d}_s$$

▸Sparsity in Bundle Adjustment

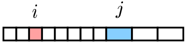
We have $q = 3p + 11k$ parameters: $\boldsymbol{\theta} = (\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_p;\ \mathbf{P}_1, \mathbf{P}_2, \ldots, \mathbf{P}_k)$ — points, cameras

We will use a multi-index $r = 1, \ldots, z$, $z = p \cdot k$. Then — r correspond to point-cam pairs (i, j)

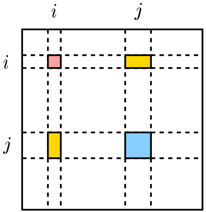
$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \sum_{r=1}^{z} \nu_r^2(\boldsymbol{\theta}), \qquad \boldsymbol{\theta}^{s+1} := \boldsymbol{\theta}^s + \mathbf{d}_s, \qquad -\sum_{r=1}^{z} \mathbf{L}_r^\top \nu_r(\boldsymbol{\theta}^s) = \left(\sum_{r=1}^{z} \mathbf{L}_r^\top \mathbf{L}_r + \lambda\ \mathrm{diag}(\mathbf{L}_r^\top \mathbf{L}_r)\right) \mathbf{d}_s$$

The block-form of $\mathbf{L}_r$ in Levenberg-Marquardt (→110) is zero except in columns i and j:

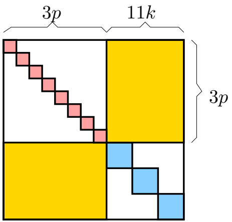
r-th error term is $\nu_r^2 = \rho(e_{ij}^2(\mathbf{X}_i, \mathbf{P}_j))$

$\mathbf{L}_r =$ 

$r = (i, j)$ blocks:
- (pink): $\mathbf{X}_i$, 1×3
- (blue): $\mathbf{P}_j$, 1×11

$\mathbf{L}_r^\top \mathbf{L}_r =$ 

blocks:
- (pink): $\mathbf{X}_i - \mathbf{X}_i$, 3×3
- (yellow): $\mathbf{X}_i - \mathbf{P}_j$, 3×11
- (blue): $\mathbf{P}_j - \mathbf{P}_j$, 11×11

$\sum_{r=1}^{z} \mathbf{L}_r^\top \mathbf{L}_r =$ 

- "points-first-then-cameras" parameterization scheme

▶Choleski Decomposition for B. A.

The most expensive computation in B. A. is solving the normal eqs:

$$\text{find } \mathbf{x} \text{ such that} \quad \mathbf{b} \stackrel{\text{def}}{=} -\sum_{r=1}^{z} \mathbf{L}_r^\top \nu_r(\theta^s) = \Big(\sum_{r=1}^{z} \mathbf{L}_r^\top \mathbf{L}_r + \lambda \operatorname{diag}\big(\mathbf{L}_r^\top \mathbf{L}_r\big)\Big)\mathbf{x} \stackrel{\text{def}}{=} \mathbf{A}\mathbf{x}$$

- $\mathbf{A}$ is very large — approx. $3 \cdot 10^4 \times 3 \cdot 10^4$ for a small problem of 10000 points and 5 cameras
- $\mathbf{A}$ is sparse, symmetric, positive definite, $\mathbf{A}^{-1}$ is dense — direct matrix inversion is prohibitive

> Choleski: A symmetric positive definite matrix $\mathbf{A}$ can be decomposed to $\mathbf{A} = \mathbf{L}\mathbf{L}^\top$, where $\mathbf{L}$ is lower triangular. If $\mathbf{A}$ is sparse then $\mathbf{L}$ is sparse, too.

1. decompose $\mathbf{A} = \mathbf{L}\mathbf{L}^\top$ — `L = chol(A);` transforms the problem to $\mathbf{L}\underbrace{\mathbf{L}^\top\mathbf{x}}_{\mathbf{c}} = \mathbf{b}$
2. solve for $\mathbf{x}$ in two passes:

$$\mathbf{L}\,\mathbf{c} = \mathbf{b} \qquad \mathbf{c}_i := \mathbf{L}_{ii}^{-1}\Big(\mathbf{b}_i - \sum_{j<i} \mathbf{L}_{ij}\mathbf{c}_j\Big)$$

forward substitution, $i = 1, \ldots, q$ (params)

$$\mathbf{L}^\top\mathbf{x} = \mathbf{c} \qquad \mathbf{x}_i := \mathbf{L}_{ii}^{-1}\Big(\mathbf{c}_i - \sum_{j>i} \mathbf{L}_{ji}\mathbf{x}_j\Big)$$

back-substitution

- Choleski decomposition is fast (does not touch zero blocks)
  non-zero elements are $9p + 121k + 66pk \approx 3.4 \cdot 10^6$; ca. $250\times$ fewer than all elements
- it can be computed on single elements or on entire blocks
- use profile Choleski for sparse $\mathbf{A}$ and diagonal pivoting for semi-definite $\mathbf{A}$ — see above; [Triggs et al. 1999]
- λ controls the definiteness

Profile Choleski Decomposition is Simple

```
function L = pchol(A)
%
% PCHOL profile Choleski factorization,
%    L = PCHOL(A) returns lower-triangular sparse L such that A = L*L'
%    for sparse square symmetric positive definite matrix A,
%    especially efficient for arrowhead sparse matrices.

% (c) 2010 Radim Sara (sara@cmp.felk.cvut.cz)

 [p,q] = size(A);
 if p ~= q, error 'Matrix A is not square'; end

 L = sparse(q,q);
 F = ones(q,1);
 for i=1:q
  F(i) = find(A(i,:),1); % 1st non-zero on row i; we are building F gradually
  for j = F(i):i-1
   k = max(F(i),F(j));
   a = A(i,j) - L(i,k:(j-1))*L(j,k:(j-1))';
   L(i,j) = a/L(j,j);
  end
  a = A(i,i) - sum(full(L(i,F(i):(i-1))).^2);
  if a < 0, error 'Matrix A is not positive definite'; end
  L(i,i) = sqrt(a);
 end
end
```

▶Gauge Freedom (kalibrační invariance)

1. The external frame is not fixed: See the Projective Reconstruction Theorem →135

$$\underline{\mathbf{m}}_{ij} \simeq \mathbf{P}_j \underline{\mathbf{X}}_i = \mathbf{P}_j \mathbf{H}^{-1} \mathbf{H} \underline{\mathbf{X}}_i = \mathbf{P}'_j \underline{\mathbf{X}}'_i$$

2. Some representations are not minimal, e.g.

- $\mathbf{P}$ is 12 numbers for 11 parameters
- we may represent $\mathbf{P}$ in decomposed form $\mathbf{K}$, $\mathbf{R}$, $\mathbf{t}$ $\quad 5+3+3=11$
- but $\mathbf{R}$ is 9 numbers representing the 3 parameters of rotation

If ignored, then

- there is no unique solution
- matrix $\sum_r \mathbf{L}_r^\top \mathbf{L}_r$ is singular

Solutions

1. fixing the external frame (e.g. a selected camera frame) explicitly or by constraints
2. fixing the scale (e.g. $s_{1,2} = 1$)

3a. either imposing constraints on projective entities

- cameras, e.g. $\mathbf{P}_{3,4} = 1$ — this excludes affine cameras
- points, e.g. $(\underline{\mathbf{X}}_i)_4 = 1$ or $\|\underline{\mathbf{X}}_i\|^2 = 1$ — the 2nd: can represent points at infinity

3b. or using minimal representations

- points in their Cartesian representation $\mathbf{X}_i$ — but finite points may be an unrealistic model
- rotation matrices can be represented by (the exponential of) skew-symmetric matrices →152

What for?

1. fixing external frame as in $\theta_i = \mathbf{t}_i$, $s_{kl} = 1$ for some i, k, l — 'trivial gauge'
2. representing additional knowledge as in $\theta_i = \theta_j$ — e.g. cameras share calibration matrix $\mathbf{K}$

Introduce reduced parameters $\hat{\theta}$ and replication matrix $\mathbf{T}$:

$$\theta = \mathbf{T}\,\hat{\theta} + \mathbf{t}, \quad \mathbf{T} \in \mathbb{R}^{p,\hat{p}}, \quad \hat{p} \le p$$

then $\mathbf{L}_r$ in LM changes to $\mathbf{L}_r\mathbf{T}$ and everything else stays the same →110

$\mathbf{T} =$

	$\hat{\theta}_1$	$\hat{\theta}_2$	$\hat{\theta}_3$	$\hat{\theta}_4$
θ_1	1			
θ_2		1		
θ_3				
θ_4				1
θ_5				1

$\mathbf{t} =$

1

these $\mathbf{T}$, $\mathbf{t}$ represent	
$\theta_1 = \hat{\theta}_1$	no change
$\theta_2 = \hat{\theta}_2$	no change
$\theta_3 = t_3$	constancy
$\theta_4 = \theta_5 = \hat{\theta}_4$	equality

- $\mathbf{T}$ deletes columns of $\mathbf{L}_r$ that correspond to fixed parameters — it reduces the problem size
- consistent initialisation: $\theta^0 = \mathbf{T}\,\hat{\theta}^0 + \mathbf{t}$ — or filter the init by pseudoinverse $\theta^0 \mapsto \mathbf{T}^\dagger \theta^0$
- no need for computing derivatives for θ_j corresponding to all-zero rows of $\mathbf{T}$ — fixed θ

- constraining projective entities →152–154
- more complex constraints tend to make normal equations dense
- implementing constraints is safer than reparameterization, it gives a flexibility to experiment
- other methods are much more involved, see [Triggs et al. 1999]
- **BA resource:** http://www.ics.forth.gr/~lourakis/sba/ [Lourakis 2009]

Matrix Exponential: A Path to Minimal Parameterization and Motion Representation

- for any square matrix we define

$$\operatorname{expm}(\mathbf{A}) = \sum_{k=0}^{\infty} \frac{1}{k!}\,\mathbf{A}^k \qquad \text{note: } \mathbf{A}^0 = \mathbf{I}$$

- some properties:

$$\operatorname{expm}(x) = e^x, \quad x \in \mathbb{R}, \quad \operatorname{expm}\mathbf{0} = \mathbf{I}, \quad \operatorname{expm}(-\mathbf{A}) = \big(\operatorname{expm}\mathbf{A}\big)^{-1},$$

$$\operatorname{expm}(a\mathbf{A} + b\mathbf{A}) = \operatorname{expm}(a\,\mathbf{A})\operatorname{expm}(b\,\mathbf{A}), \quad \operatorname{expm}(\mathbf{A} + \mathbf{B}) \neq \operatorname{expm}(\mathbf{A})\operatorname{expm}(\mathbf{B})$$

$\operatorname{expm}(\mathbf{A}^\top) = (\operatorname{expm}\mathbf{A})^\top$ hence if $\mathbf{A}$ is skew symmetric then $\operatorname{expm}\mathbf{A}$ is orthogonal:

$$\big(\operatorname{expm}(\mathbf{A})\big)^\top = \operatorname{expm}(\mathbf{A}^\top) = \operatorname{expm}(-\mathbf{A}) = \big(\operatorname{expm}(\mathbf{A})\big)^{-1}$$

$$\det\big(\operatorname{expm}\mathbf{A}\big) = e^{\operatorname{tr}\mathbf{A}}$$

Some consequences

- traceless matrices ($\operatorname{tr}\mathbf{A} = 0$) map to unit-determinant matrices $\Rightarrow$ we can represent homogeneous matrices
- skew-symmetric matrices map to orthogonal matrices $\Rightarrow$ we can represent rotations
- matrix exponential provides the exponential map from the powerful (matrix) Lie group theory

Lie Groups Useful in 3D Vision

group		matrix	represent
special linear	$\mathrm{SL}(3,\mathbb{R})$	real 3×3, unit determinant $\mathbf{H}$	2D homography
special linear	$\mathrm{SL}(4,\mathbb{R})$	real 4×4, unit determinant $\mathbf{H}$	3D homography
special orthogonal	$\mathrm{SO}(3)$	real 3×3 orthogonal $\mathbf{R}$	3D rotation
special Euclidean	$\mathrm{SE}(3)$	4×4 $\begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix}$, $\mathbf{R} \in \mathrm{SO}(3)$, $\mathbf{t} \in \mathbb{R}^3$	3D rigid motion
similarity	$\mathrm{Sim}(3)$	4×4 $\begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & s^{-1} \end{bmatrix}$, $s \in \mathbb{R} \setminus 0$	rigid motion + scale

- Lie group G = topological group that is also a smooth manifold with nice properties
- Lie algebra $\mathfrak{g}$ = vector space associated with a Lie group (tangent space of the manifold)
- group: this is where we need to work
- algebra: this is how to represent group elements with a minimal number of parameters
- Exponential map = map between algebra and its group $\exp\colon \mathfrak{g} \to G$
- for matrices $\exp = \mathrm{expm}$
- in most of the above groups we a have a closed-form formula for the exponential and for its principal inverse
- Jacobians are also readily available for $\mathrm{SO}(3)$, $\mathrm{SE}(3)$ [Solà 2020]

Homography

$$\mathbf{H} = \operatorname{expm}(\mathbf{Z})$$

- $\mathrm{SL}(3, \mathbb{R})$ group element

$$\mathbf{H} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \quad \text{s.t.} \quad \det(\mathbf{H}) = 1$$

- $\mathfrak{sl}(3, \mathbb{R})$ algebra element

8 parameters

$$\mathbf{Z} = \begin{bmatrix} z_{11} & z_{12} & z_{13} \\ z_{21} & z_{22} & z_{23} \\ z_{31} & z_{32} & -(z_{11} + z_{22}) \end{bmatrix}$$

- note that $\operatorname{tr} \mathbf{Z} = 0$

►Rotation in 3D

$$\mathbf{R} = \operatorname{expm}\,[\boldsymbol{\phi}]_\times, \quad \boldsymbol{\phi} = (\phi_1,\, \phi_2,\, \phi_3) = \varphi\, \mathbf{e}_\varphi \in \mathbb{R}^3, \quad 0 \le \varphi < \pi, \quad \|\mathbf{e}_\varphi\| = 1$$

- $\mathrm{SO}(3)$ group element

$$\mathbf{R} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \quad \text{s.t.} \quad \mathbf{R}^{-1} = \mathbf{R}^\top$$

- $\mathfrak{so}(3)$ algebra element — 3 parameters

$$[\boldsymbol{\phi}]_\times = \begin{bmatrix} 0 & -\phi_3 & \phi_2 \\ \phi_3 & 0 & -\phi_1 \\ -\phi_2 & \phi_1 & 0 \end{bmatrix}$$

- exponential map in closed form — Rodrigues' formula

$$\mathbf{R} = \operatorname{expm}\,[\boldsymbol{\phi}]_\times = \sum_{n=0}^{\infty} \frac{[\boldsymbol{\phi}]_\times^n}{n!} = \overset{\circledast\,1}{\cdots} = \mathbf{I} + \frac{\sin\varphi}{\varphi}\,[\boldsymbol{\phi}]_\times + \frac{1-\cos\varphi}{\varphi^2}\,[\boldsymbol{\phi}]_\times^2$$

- (principal) logarithm — log is a periodic function

$$0 \le \varphi < \pi, \quad \cos\varphi = \frac{1}{2}\left(\operatorname{tr}(\mathbf{R}) - 1\right), \quad [\boldsymbol{\phi}]_\times = \frac{\varphi}{2\,\sin\varphi}(\mathbf{R} - \mathbf{R}^\top),$$

- $\boldsymbol{\phi}$ is rotation axis vector $\mathbf{e}_\varphi$ scaled by rotation angle φ in radians
- finite limits for $\varphi \to 0$ exist: $\sin(\varphi)/\varphi \to 1$, $(1-\cos\varphi)/\varphi^2 \to 1/2$

3D Rigid Motion

$$\mathbf{M} = \operatorname{expm}\,[\boldsymbol{\nu}]_\wedge\,, \quad \boldsymbol{\nu} \in \mathbb{R}^6$$

- $\mathrm{SE}(3)$ group element — 4×4 matrix

$$\mathbf{M} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} \quad \text{s.t.} \quad \mathbf{R} \in \mathrm{SO}(3),\ \mathbf{t} \in \mathbb{R}^3$$

- $\mathfrak{se}(3)$ algebra element — 4×4 matrix; $\wedge = \times$ in $\mathrm{SO}(3)$

$$[\boldsymbol{\nu}]_\wedge = \begin{bmatrix} [\boldsymbol{\phi}]_\times & \boldsymbol{\rho} \\ \mathbf{0} & 0 \end{bmatrix} \quad \text{s.t.} \quad \boldsymbol{\phi} \in \mathbb{R}^3,\ \varphi = \|\boldsymbol{\phi}\| < \pi,\ \boldsymbol{\rho} \in \mathbb{R}^3$$

- exponential map in closed form

$$\mathbf{R} = \operatorname{expm}\,[\boldsymbol{\phi}]_\times\,, \quad \mathbf{t} = \operatorname{dexpm}([\boldsymbol{\phi}]_\times)\,\boldsymbol{\rho}$$

$$\operatorname{dexpm}([\boldsymbol{\phi}]_\times) = \sum_{n=0}^{\infty} \frac{[\boldsymbol{\phi}]_\times^n}{(n+1)!} = \mathbf{I} + \frac{1-\cos\varphi}{\varphi^2}[\boldsymbol{\phi}]_\times + \frac{\varphi - \sin\varphi}{\varphi^3}[\boldsymbol{\phi}]_\times^2$$

$$\operatorname{dexpm}^{-1}([\boldsymbol{\phi}]_\times) = \mathbf{I} - \frac{1}{2}[\boldsymbol{\phi}]_\times + \frac{1}{\varphi^2}\left(1 - \frac{\varphi}{2}\cot\frac{\varphi}{2}\right)[\boldsymbol{\phi}]_\times^2$$

- dexpm: differential of the exponential in $\mathrm{SO}(3)$
- (principal) logarithm via a similar trick as in $\mathrm{SO}(3)$
- finite limits exist: $(\varphi - \sin\varphi)/\varphi^3 \to 1/6$
- this form is preferred to $\mathrm{SO}(3) \times \mathbb{R}^3$

▶Minimal Representations for Other Entities

- fundamental matrix via $\mathrm{SO}(3) \times \mathrm{SO}(3) \times \mathbb{R}^+$

$$\mathbf{F} = \mathbf{U}\mathbf{D}\mathbf{V}^\top, \quad \mathbf{D} = \mathrm{diag}(1, d^2, 0), \quad \mathbf{U}, \mathbf{V} \in \mathrm{SO}(3), \quad 3 + 1 + 3 = 7 \text{ DOF}$$

- essential matrix via $\mathrm{SO}(3) \times \mathbb{R}^3$

$$\mathbf{E} = [-\mathbf{t}]_\times \mathbf{R}, \quad \mathbf{R} \in \mathrm{SO}(3), \quad \mathbf{t} \in \mathbb{R}^3, \ \|\mathbf{t}\| = 1, \qquad 3 + 2 = 5 \text{ DOF}$$

- camera pose via $\mathrm{SO}(3) \times \mathbb{R}^3$ or $\mathrm{SE}(3)$

$$\mathbf{P} = \mathbf{K}\begin{bmatrix}\mathbf{R} & \mathbf{t}\end{bmatrix} = \begin{bmatrix}\mathbf{K} & \mathbf{0}\end{bmatrix}\mathbf{M}, \qquad 5 + 3 + 3 = 11 \text{ DOF} \qquad \mathbf{M} \in \mathrm{SE}(3)$$

- $\mathrm{Sim}(3)$ useful for SfM without scale
  - closed-form formulae still exist but they are a bit too messy [Eade(2017)]
- a (bit too brief) intro to Lie groups in 3D vision/robotics and SW:

J. Solà, J. Deray, and D. Atchuthan. A micro Lie theory for state estimation in robotics. arXiv:1812.01537v7 [cs.RO], August 2020.

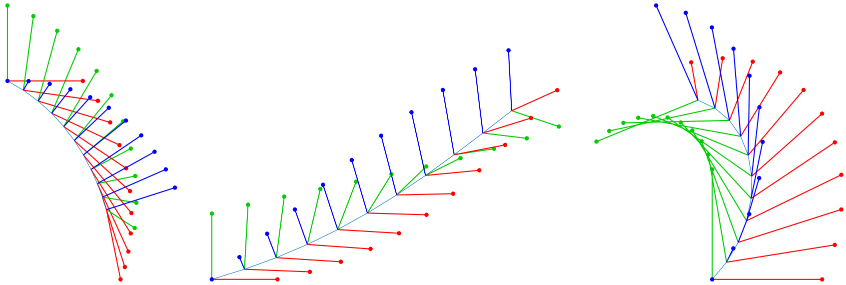
E. Eade. Lie groups for 2D and 3D transformations. On-line at http://www.ethaneade.org/, May 2017.

- let G be a Lie group — like $\mathrm{SO}(3)$, $\mathrm{SE}(3)$
- let $\mathbf{M} \in G$ be motion from time $t = 0$ to time $t = 1$
- then the motion from $t = 0$ to t is interpolated as

$$\mathbf{M}(t) = \exp\big(t \, \log(\mathbf{M})\big), \qquad t \in [0, 1]$$

- the trajectory is constant-speed,
- and the speed is $\log(\mathbf{M})$

Examples in $\mathrm{SE}(3)$:



Distance between Lie Group Elements

- Integration formula — the motion is along the geodesic (shortest-distance curve)

$$\lim_{n\to\infty} \prod_{i=1}^{n} \exp\left(\frac{1}{n}\log(\mathbf{M})\right) = \mathbf{M}$$

- hat and vee functions:
  - $[\mathbf{a}]_\wedge$ maps vector $\mathbf{a} \in \mathbb{R}^d$ to algebra $\mathfrak{g}$ element (matrix)
  - $(\mathbf{B})_\vee$ maps algebra element $\mathbf{B} \in \mathfrak{g}$ to vector element, $\left([\mathbf{a}]_\wedge\right)_\vee = \mathbf{a}$

- the Log function is a composition of log and vee, $\mathrm{Log} : G \to \mathbb{R}^d$, $\mathrm{Log}(\mathbf{M}) = (\log(\mathbf{M}))_\vee$ — $G \to \mathfrak{g} \to \mathbb{R}^d$
- then: left/right difference — $\mathbf{Y} \overset{\leftarrow}{\ominus} \mathbf{X} \in \mathbb{R}^d$

$$\mathbf{Y} \overset{\leftarrow}{\ominus} \mathbf{X} = \mathrm{Log}(\mathbf{Y}\mathbf{X}^{-1}), \quad \mathbf{Y} \overset{\rightarrow}{\ominus} \mathbf{X} = \mathrm{Log}(\mathbf{X}^{-1}\mathbf{Y})$$

- skew-symmetry

$$\mathbf{Y} \overset{\leftarrow}{\ominus} \mathbf{X} = -(\mathbf{X} \overset{\leftarrow}{\ominus} \mathbf{Y}), \quad \mathbf{Y} \overset{\rightarrow}{\ominus} \mathbf{X} = -(\mathbf{X} \overset{\rightarrow}{\ominus} \mathbf{Y})$$

- left/right distance

$$\overset{\leftarrow}{d}(\mathbf{X}, \mathbf{Y}) = \|\mathbf{Y} \overset{\leftarrow}{\ominus} \mathbf{X}\|, \quad \vec{d}(\mathbf{X}, \mathbf{Y}) = \|\mathbf{Y} \overset{\rightarrow}{\ominus} \mathbf{X}\|$$

- not equal but both are non-negative, symmetric — + additional properties, e.g. left/right invariance,...

Thank You