Zusammenfassung - SS2019

Computer Vision II: Multiple view geometry (Technische Universität München)

Computer Vision 2 - Multiple View Geometry Class (IN2228): Summary

Marcel Brucker
Technical University of Munich

Summer Semester 2019

This document summarizes the main topics and key aspects of the lecture "Computer Vision 2 - Multiple View Geometry". The structure follows the script by Prof. Daniel Cremers. However, for better understanding some content from linear algebra is added.

1. Mathematical Background: Linear Algebra

1.1. Set

A Set S of different elements e.g. Natural numbers $\mathbb{N} = \{1, 2, 3, \dots\}$.

1.2. Group

A set S combined with a defined operation $\circ$ e.g. $(\mathbb{N}, \cdot)$ with properties:

1. $a \circ b \in S$ (closed)
2. $(a \circ b) \circ c = a \circ (b \circ c)$ (associative)
3. $a \circ n = a$ (n is a neutral element in S)
4. $a \circ a^{-1} = n$ (every element has an inverse element)

(Abelian groups are commutative in addition: $a \circ b = b \circ a$.)

1.3. Ring

A set S combined with two defined operations and properties from above e.g. $(\mathbb{N}, +, \cdot)$.

1.4. Field

A ring which is distributive in addition: $a \cdot (b + c) = a \cdot b + a \cdot c$ e.g. $(\mathbb{R}, +, \cdot)$.

1.5. Vector Space

A set V is called a vector space over the field $\mathbb{R}$ if it is closed under vector summation

$$+ : V \times V \to V$$

and under scalar multiplication

$$\cdot : \mathbb{R} \times V \to V$$

1.6. Linear Independence

A set of vectors $S = \{v_1, \ldots, v_k\} \subset V$ is the subspace formed by all linear combinations of these vectors:

$$span(S) = \left\{ v \in V \,\middle|\, v = \sum_{i=1}^{k} \alpha_i v_i \right\}$$

Linear independence holds if none of the vectors can be expressed as a linear combination of the remaining vectors.

1.7. Basis

A set of vectors $B = v_1, \ldots, v_n$ is called a basis of V if it is linearly independent and if it spans the vector space V. A basis is a maximal set of linearly independent vectors.

1.8. Dot Product

$$\langle \cdot, \cdot \rangle : V \times V \to \mathbb{R}$$

Two vectors v and w are orthogonal iff $\langle v, w \rangle = 0$

1.9. Kronecker Product

Given two matrices $A \in \mathbb{R}^{m \times n}$ $B \in \mathbb{R}^{k \times l}$, one can define their Kronecker product $A \otimes B$ by:

$$A \otimes B \equiv \begin{pmatrix} a_{11}B & \cdots & a_{1n}B \\ \vdots & \ddots & \vdots \\ a_{m1}B & \cdots & a_{mn}B \end{pmatrix}$$

1.10. Popular Groups

- $GL(n)$: $n \times n$ Matrices w.r.t. multiplication - $det(A) \neq 0 \Leftrightarrow rank(A) = n \Leftrightarrow kernel(A) = \{0\} \Rightarrow$ unique solution
- $SL(n)$: $det(A) = 1$
- $O(n)$: Preserves dot product for vector multiplication - $R^T R = R R^T = I$
- $SO(n)$: 3D rotation matrices, Lie Group - $det(R) = 1$ (see appendix B)
- $A(n)$: $L(x) = Ax + b \qquad A \in GL(n)$ for Rotation, $b \in \mathbb{R}^n$ for Translation
- $E(n)$: $L(x) = Rx + T \qquad R \in O(n)$ for Rotation, $T \in \mathbb{R}^n$ for Translation
- $SE(n)$: Rigid-body motion - $L(x) = Rx + T \qquad R \in SO(n)$ for Rotation, $T \in \mathbb{R}^n$ for Translation

1.11. Range/Span

The range of a matrix A is given by the span of its column vectors

1.12. Kernel/Null Space

The null space or kernel of a matrix A is given by the subset of vectors $x \in \mathbb{R}^n$ which are mapped to zero

$$null(A) \equiv ker(A) = \{x \in \mathbb{R} \mid Ax = 0\}$$

1.13. Rank

The rank of a matrix is the dimension of its range

$$rank(A) = dim(range(A))$$

Properties of the rank of a matrix $A \in \mathbb{R}^{m \times n}$:

1. $rank(A) = n - dim(ker(A))$
2. $0 \leq rank(A) \leq min\{m, n\}$
3. $rank(A)$ is equal to the maximum number of linearly independent row or column vectors of A.

1.14. Eigenvalues and Eigenvectors

Let $A \in \mathbb{R}^{n \times n}$ be a square matrix

$$Av = \lambda v \text{ (v is a right eigenvector of A)}$$
$$v^T A = \lambda v^T \text{ (v is a left eigenvector of A)}$$
$$A = V \Sigma V^{-1}$$

Properties:

1. The eigenvectors of a matrix A associated with different eigenvalues are linearly independent - $\lambda_a \neq \lambda_b \Rightarrow v_a$ and v_b are linearly independent
2. All eigenvalues $\sigma(A)$ are the roots of the characteristic polynomial $det(\lambda I - A) = 0$

1.15. Singular Value Decomposition (SVD)

Let $A \in \mathbb{R}^{m \times n}$ be a matrix

$$A = U \Sigma V^T$$

1.16. Pseudo Inverse

Let $A \in \mathbb{R}^{m \times n}$ be a matrix

$$A^+ = V \Sigma^+ U^T, \text{ where } \Sigma^+ = \begin{pmatrix} \Sigma_1^{-1} & 0 \\ 0 & 0 \end{pmatrix}$$

2. Representing a Moving Scene

2.1. 3D Space & Rigid Body Motion

The three-dimensional Euclidean space $\mathbb{E}^3$ consists of all points $p \in \mathbb{E}^3$ characterized by coordinates

$$X \equiv (X_1, X_2, X_3)^T \in \mathbb{R}^3$$

Given two points X and Y, one can define a bound vector v as

$$v = X - Y \in \mathbb{R}^3$$

On $\mathbb{R}^3$ one can define a cross product

$$\times : \mathbb{R}^3 \times \mathbb{R}^3 \to \mathbb{R}^3 : u \times v = \begin{pmatrix} u_2 v_3 - u_3 v_2 \\ u_3 v_1 - u_1 v_3 \\ u_1 v_2 - u_2 v_1 \end{pmatrix} \in \mathbb{R}^3$$

which is a vector orthogonal to u and v ($u \times v = -v \times u$). The skew-symmetric matrix $\hat{u} \in \mathbb{R}^{3 \times 3}$ replaces the cross product by a matrix multiplication

$$\hat{u} \cdot v = \begin{pmatrix} 0 & -u_3 & u_2 \\ u_3 & 0 & -u_1 \\ -u_2 & u_1 & 0 \end{pmatrix} \cdot v = u \times v$$

The cross product of a vector with itself equals zero ($u \times u = 0$)
A rigid body motion is a map $g_t : \mathbb{R}^3 \to \mathbb{R}^3$ which preserves the norm or inner product (length), cross product (orientation) and the triple product (volume) of any two vectors. The rigid body motion can be written as $g_t(x) = Rx + T$.

2.2. The Lie Group SO(3)

The rotation group $SO(3)$ is called a Lie group. The space $so(3)$ is called its Lie algebra.
The matrix exponential defines a map from the Lie algebra to the Lie group

$$exp : so(3) \to SO(3) : \hat{w} \mapsto e^{\hat{w}} = R \in SO(3)$$

(w as angular velocity, for the hat operator see subsection 2.1). The inverse of the matrix exponential sets

$$log : SO(3) \to so(3) : \hat{w} = ln(R)$$

The matrix exponential $e^{\hat{w}} = R$ is calculated through the Rodrigues' formula

$$e^{\hat{w}} = I + \frac{\hat{w}}{|w|} sin(|w|) + \frac{\hat{w}^2}{|w|^2}(1 - cos(|w|))$$

and its inverse for obtaining w through

$$|w| = cos^{-1}\left(\frac{trace(R) - 1}{2}\right),$$
$$\frac{w}{|w|} = \frac{1}{2sin(|w|)} \begin{pmatrix} r_{32} - r_{23} \\ r_{13} - r_{31} \\ r_{21} - r_{12} \end{pmatrix}$$

Any orthogonal transformation $R \in SO(3)$ (see appendix B) can be realized by rotating by an angle $|w|$ around an axis $\frac{w}{|w|}$.

2.3. The Lie Group SE(3)

The space of rigid body motions given by the group of special euclidean transformations

$$SE(3) \equiv \left\{ g = (R, T) \,\middle|\, R \in SO(3), T \in \mathbb{R}^3 \right\}$$

In homogeneous coordinates, we have:

$$SE(3) \equiv \left\{ g = \begin{pmatrix} R & T \\ 0 & 1 \end{pmatrix} \,\middle|\, R \in SO(3), \; T \in \mathbb{R}^3 \right\} \subset \mathbb{R}^{4\times 4}$$

As in the case of $so(3)$, the set of all twists forms a tangent space which is the Lie algebra

$$se(3) \equiv \left\{ \hat{\xi} = \begin{pmatrix} \hat{w} & v \\ 0 & 0 \end{pmatrix} \,\middle|\, \hat{w} \in so(3), v \in \mathbb{R}^3 \right\}$$
$$\subset \mathbb{R}^{4\times 4} \quad \text{with} \quad v(t) = \dot{T}(t) - \hat{w}(t)T(t)$$

to the Lie Group SE(3) (v as linear velocity). Operators $\wedge$ and $\vee$ convert between a twist $\hat{\xi} \in se(3)$ and its twist coordinates $\xi \in \mathbb{R}^6$:

$$\hat{\xi} \equiv \begin{pmatrix} v \\ w \end{pmatrix}^{\wedge} \equiv \begin{pmatrix} \hat{w} & v \\ 0 & 0 \end{pmatrix} \in \mathbb{R}^{4\times 4},$$
$$\begin{pmatrix} \hat{w} & v \\ 0 & 0 \end{pmatrix}^{\vee} = \begin{pmatrix} v \\ w \end{pmatrix} \in \mathbb{R}^6$$

For $w = 0$, we have $e^{\hat{\xi}} = \begin{pmatrix} I & v \\ 0 & 1 \end{pmatrix}$, while for $w \neq 0$ one can show:

$$e^{\hat{\xi}} = \begin{pmatrix} e^{\hat{w}} & \frac{(I - e^{\hat{w}})\hat{w}v + ww^T v}{|w|^2} \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} R & T \\ 0 & 1 \end{pmatrix} = g$$

Exponential map defines a transformation from the Lie Algebra $se(3)$ to the Lie Group $SE(3)$:

$$exp : se(3) \to SE(3); \hat{\xi} \mapsto e^{\hat{\xi}}$$

For the inverse function and further information see http://ethaneade.com/lie.pdf.

2.4. Representing the Camera Motion

Equation

$$g(t) = \begin{pmatrix} R(t) & T(t) \\ 0 & 1 \end{pmatrix} \in SE(3)$$

represents the motion from a fixed world frame to the camera frame at time t. In particular we assume that at time $t = 0$ the camera frame coincides with the world frame, i.e. $g(0) = I$. For any point X_0 in world coordinates, its coordinates in the camera frame at time t are:

$$X(t) = R(t)X_0 + T(t)$$

or in homogeneous representation

$$X(t) = g(t)X_0 \quad \text{and further}$$
$$X(t_2) = g(t_2, t_1)X(t_1) \quad \text{as well as}$$
$$g(t_1, t_2)g(t_2, t_1) = I \Leftrightarrow g^{-1}(t_2, t_1) = g(t_1, t_2)$$

For velocity transformation it applies

$$\dot{X}(t) = \hat{w}(t)X(t) + v(t) \quad \text{or}$$
$$\dot{X}(t) = \hat{V}(t)X(t) \quad \text{with}$$
$$\hat{V}(t) = \begin{pmatrix} \hat{w}(t) & v(t) \\ 0 & 0 \end{pmatrix} \in se(3)$$

Summary

	Rotation $SO(3)$	Rigid body SE(3)
Matrix representation	$R \in GL(3)$: $R^T R = I$, $det(R) = 1$	$g = \begin{pmatrix} R & T \\ 0 & 1 \end{pmatrix}$
3-D coordinates	$X = RX_0$	$X = RX_0 + T$
Inverse	$R^{-1} = R^T$	$g^{-1} = \begin{pmatrix} R^T & -R^T T \\ 0 & 1 \end{pmatrix}$
Exp. representation	$R = exp(\hat{w})$	$g = exp(\hat{\xi})$
Velocity	$\dot{X} = \hat{w}X$	$\dot{X} = \hat{w}X + v$

3. Perspective Projection

3.1. Mathematical Representation

The perspective transformation π is given by

$$\pi : \mathbb{R}^3 \to \mathbb{R}^2; \quad X \mapsto x = \pi(X) = \begin{pmatrix} f\frac{X}{Z} \\ f\frac{Y}{Z} \end{pmatrix}$$

The transformation from world coordinates to image coordinates ($Z = \lambda$: camera distance) is given by

$$\lambda x' = \lambda \begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = K\Pi_0 X = \Pi X_0$$

with the general projection matrix $\Pi \equiv K\Pi_0 g = K_s K_f \Pi_0 g$

$$= \underbrace{\begin{pmatrix} s_x & s_\theta & o_x \\ 0 & s_y & o_y \\ 0 & 0 & 1 \end{pmatrix}}_{K_s} \underbrace{\begin{pmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{pmatrix}}_{K_f} \cdot \underbrace{\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}}_{\Pi_0} \underbrace{\begin{pmatrix} R & T \\ 0 & 1 \end{pmatrix}}_{g}$$

In other words for homogeneous coordinates: 4D world coordinates $\xrightarrow{g \in SE(3)}$ 4D camera coordinates $\xrightarrow{K_f \Pi_0}$ 3D image coordinates $\xrightarrow{K_s}$ 3D pixel coordinates

3.2. Intrinisc Parameters

The entries of the intrinsic parameter matrix

$$K = \begin{pmatrix} fs_x & fs_\theta & o_x \\ 0 & fs_y & o_y \\ 0 & 0 & 1 \end{pmatrix}$$

can be interpreted as follows:

- f: focal length
- fs_x: size of unit length in horizontal pixels
- fs_y: size of unit length in vertical pixels
- fs_θ: skew of the pixel
- o_x: x-coordinate of principal point in pixels

- o_y: y-coordinate of principal point in pixels

3.3. Spherical Projection

The spherical projection π_s of a 3D point X is given by:

$$\pi_s : \mathbb{R}^3 \to \mathbb{S}^2; \quad X \mapsto x = \frac{X}{|X|}$$

3.4. Radial Distortion

A simple effective model for distortions along the radial axis is:

$$x = x_d(1 + a_1 r^2 + a_2 r^4),$$
$$y = y_d(1 + a_1 r^2 + a_2 r^4)$$
$$r = x_d^2 + y_d^2$$

where $x_d \equiv (x_d, y_d)$ is the distorted point.

3.5. Preimage and Coimage

A line L in 3D is characterized by a base point $X_0 = (X_0, Y_0, Z_0, 1)^T \in \mathbb{R}^4$ and a vector $V = (V_1, V_2, V_3, 0)^T \in \mathbb{R}^4$:

$$X = X_0 + \mu V \subset \mathbb{R}^4, \quad \mu \in \mathbb{R}$$

The image of the line L is given by

$$x \sim \Pi_0 X = \Pi_0(X_0 + \mu V) = \Pi_0 X_0 + \mu \Pi_0 V$$

All points x treated as vectors from the origin o span a 2D subspace P. The intersection of this plane P with the image plane gives the image of the line. P is called the preimage of the line.
The coimage of a point or a line is the subspace in $\mathbb{R}^3$ that is the unique orthogonal complement, i.e. the normal vector, of its preimage. Image, preimage and coimage are equivalent.

	Image	Preimage	Coimage
Point	$span(x) \cap im.plane$	$span(x) \subset \mathbb{R}^3$	$span(\hat{x}) \subset \mathbb{R}^3$
Line	$span(\hat{\ell}) \cap im.plane$	$span(\hat{\ell}) \subset \mathbb{R}^3$	$span(\ell) \subset \mathbb{R}^3$

$\ell \in \mathbb{R}^3$ is the normal vector to the preimage (2D subspace) of a Line L.

4. Estimating Point Correspondence

4.1. From Photometry to Geometry

In practice, instead of points or lines we observe characteristic image features like brightness or color values in pixels. By selecting a small number of feature points from each image, we throw away a large amount of potentially useful information.
Two cases are distinguished in point matching: small deformation and wide baseline stereo (large displacement)

4.2. Small Deformation & Optical Flow

The optic flow refers to the apparent 2D motion field observable between consecutive images of a video (not motion of objects in the scene).

4.3. The Lucas-Kanade Method

Let $x(t)$ denote a moving point at time t, and $I(x, t)$ a video sequence.
Optical flow constraint: the brightness of point x(t) is constant.
The desired local flow vector (velocity) is

given by

$$v = \frac{dx}{dt} = -M^{-1}q$$
$$\text{with } M = \int_{W(x)} \nabla I \nabla I^T \, \mathrm{d}x'$$
$$\text{and } q = \int_{W(x)} I_t \nabla I \, \mathrm{d}x'$$
$$\text{where } \nabla I = \begin{pmatrix} I_x \\ I_y \end{pmatrix} = \begin{pmatrix} \frac{\partial I}{\partial x} \\ \frac{\partial I}{\partial y} \end{pmatrix}; \quad I_t = \frac{\mathrm{d}I}{\mathrm{d}t}$$

Simple feature tracking algorithm:

1. For a given time instance t, compute at each point $x \in \Omega$ the structure tensor
$$M(x) = \int_{W(x)} \begin{pmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{pmatrix} \mathrm{d}x'$$
2. Mark all points $x \in \Omega$ for which the determinant of M is larger than a threshold $\theta > 0$:
$$det(M(x)) \geq \theta$$
3. For all these points the local velocity is given by:
$$b(x,t) = -M(x)^{-1} \begin{pmatrix} \int I_x I_t \, \mathrm{d}x' \\ \int I_y I_t \, \mathrm{d}x' \end{pmatrix}$$
4. Repeat the above steps for the points $x + b$ at time $t + 1$

4.4. Feature Point Extraction

One corner detecting algorithm is based on the structure tensor

$$M(x) \equiv G_\sigma * \nabla I \nabla I^T$$
$$= \int G_\sigma(x - x') \begin{pmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{pmatrix} (x') \, \mathrm{d}x'$$
$$\text{and } q = G_\sigma * I_t \nabla I$$

where rather than simple summing over the window $W(x)$ we perform a summation weighted by a Gaussian G of width σ. Additionally, the scoring function:

$$C(x) = det(M) - \kappa \, trace^2(M)$$

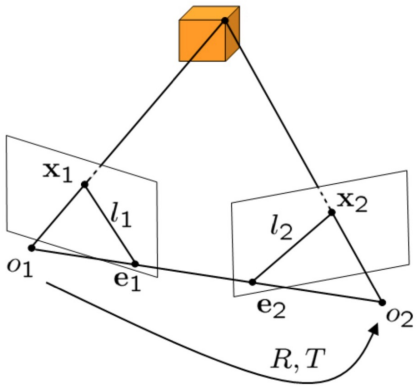
and select points for which $C(x) > \theta$ with a treshold $\theta > 0$

4.5. Wide Baseline Matching

In the case of wide baseline matching, large parts of the image plane will not match at all because they are not visible in the other image.
The normalized cross correlation is robust to illumination changes. It can also be used to determine the optimal affine transformation between two given patches.

5. Reconstruction from Two Views: Linear Algorithms

5.1. The Reconstruction Problem



- x_1 and x_2: projections of a point X onto the two images
- o_1 and o_2: optical centers of each camera
- e_1 and e_2: epipoles are the intersections of the line $(o1, o2)$ with each image plane
- l_1 and l_2: intersections between the epipolar plane $(o1, o2, X)$ and the image planes

5.2. The Epipolar Constraint

Assuming known camera parameters and no rotation or translation of the first camera, we merely have a projection with unknown depth λ_1. From the first to the second frame we additionally have a camera rotation R (see appendix B) and translation T, hence

$$\lambda_1 x_1 = X \quad \text{and} \quad \lambda_2 x_2 = RX + T$$

from which one can derive the epipolar constraint:

$$x_2^T \hat{T} R x_1 = 0 \quad \text{for } \hat{T} \text{ see 2.1}$$

It provides a relation between the 2D point coordinates of a 3D point in each of the two images and the camera transformation parameters. The matrix

$$E = \hat{T}R \in \mathbb{R}^{3\times 3}$$

is called the essential matrix if and only if E has a SVD $E = U\Sigma V^T$ with $\Sigma = diag\{\sigma, \sigma, 0\}$ where $\sigma > 0$ ($\sigma < 0$ as well according to tutor) and $U, V \in SO(3)$.
For $E = U\Sigma V^T$ we have:

$$(\hat{T_1}, R_1) = \left(UR_Z(+\frac{\pi}{2})\Sigma U^T,\ UR_Z^T(+\frac{\pi}{2})V^T\right)$$
$$(\hat{T_2}, R_2) = \left(UR_Z(-\frac{\pi}{2})\Sigma U^T,\ UR_Z^T(-\frac{\pi}{2})V^T\right)$$

5.3. Eight-Point Algorithm

One basic reconstruction algorithm proceeds as follows:

1. Recover the essential matrix E from the epipolar constraints associated with a set of point pairs
2. Extract the relative translation and rotation from the essential matrix E

Let

$$E^s = (e_{11}, e_{21}, e_{31}, e_{12}, e_{22}, e_{32}, e_{13}, e_{23}, e_{33})^T \in \mathbb{R}^9$$

be the stack of E and

$$a \equiv x_1 \otimes x_2 = (x_1x_2, x_1y_2, x_1z_2, y_1x_2, y_1y_2, y_1z_2, z_1x_2, z_1y_2, z_1z_2)^T \in \mathbb{R}^9$$

the Kronecker product of the vectors $x_i = (x_i, y_i, z_i)$.
Then the epipolar constraint can be written as:

$$x_2^T E x_1 = a^T E^s = 0$$

For n point pairs, we can combine this into the linear system:

$$\chi E^s = 0, \text{ with } \chi = (a^1, a^2, \ldots, a^n)^T$$

The numerically estimated coefficients E^s will in general not correspond to an essential matrix. One can resolve this problem by projecting it back to the essential space. Let $F \in \mathbb{R}^{3\times 3}$ be an arbitrary matrix with SVD $F = U\, diag\{\lambda_1, \lambda_2, \lambda_3\}\, V^T$, $\lambda_1 \geq \lambda_2 \geq \lambda_3$. Then the essential matrix E is given by

$$E = U\, diag\{\sigma, \sigma, 0\}\, V^T, \text{ with } \sigma = \frac{\lambda_1 + \lambda_2}{2}$$

Complete Eight Point Algorithm (Longuet-Higgins 1981)
Given a set of $n = 8$ or more point pairs x_1^i, x_2^i:

1. Compute an approximation of the essential matrix
   Construct the matrix $\chi = (a^1, a^2, \ldots, a^n)^T$, where $a^i = x_1^i \otimes x_2^i$. Find the vector $E^s \in \mathbb{R}^9$ which minimizes $||\chi E^s||$ as the ninth column of V_χ in the SVD $\chi = U_\chi \Sigma_\chi V_\chi^T$. Unstack E^s into 3×3-matrix E.

2. Project onto essential space
   Compute the SVD $E = U\, diag\{\sigma_1, \sigma_2, \sigma_3\}\, V^T$. Since in the reconstruction, E is only defined up to a scalar, we project E onto the normalized essential space by replacing the singular values σ_1, σ_2, σ_3 with 1, 1, 0.

3. Recover the displacement from the essential matrix.
   The four possible solutions for rotation and translation are:

$$R = UR_Z^T(\pm\frac{\pi}{2})V^T,$$
$$\hat{T} = UR_Z(\pm\frac{\pi}{2})\Sigma U^T$$

   with a rotation by $\pm\frac{\pi}{2}$ around z:

$$R_Z^T(\pm\frac{\pi}{2}) = \begin{pmatrix} 0 & \pm 1 & 0 \\ \mp 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

5.4. Structure Reconstruction

In the linear eight-point algorithm, the essential matrix E and hence the translation T are only defined up to an arbitrary scale $\gamma \in \mathbb{R}^+$. Combining all unknown scale parameters $\vec{\lambda} = \left(\lambda_1^1, \lambda_1^2, \ldots, \lambda_1^n, \gamma\right)^T \in \mathbb{R}^{n+1}$, we get the linear equation system

$$M\vec{\lambda} = 0 \quad \text{with}$$
$$M \equiv \begin{pmatrix} \hat{x_2^1}Rx_1^1 & 0 & 0 & 0 \\ 0 & \hat{x_2^2}Rx_1^2 & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & \hat{x_2^1}T \\ 0 & 0 & 0 & \hat{x_2^2}T \\ 0 & 0 & 0 & \vdots \\ \hat{x_2^{n-1}}Rx_1^{n-1} & 0 & 0 & \hat{x_2^{n-1}}T \\ 0 & \hat{x_2^n}Rx_1^n & 0 & \hat{x_2^n}T \end{pmatrix}$$

The linear least squares estimate for $\vec{\lambda}$ is given by the eigenvector corresponding to the smallest eigenvalue of $M^T M$.
The eight-point algorithm only provides unique solutions (up to a scalar factor) if all 3D points are in a "general position". This is no longer the case for certain degenerate configurations, for which all points lie on certain 2D surface such as walls or floors.

5.5. Four Point Algorithm

If $X_1 \in \mathbb{R}^3$ denotes point coordinates in the first frame, and these lie on a plane with normal $N \in \mathbb{S}^2$, then we have:

$$X_2 = HX_1 \text{ with } H = R + \frac{1}{d}TN^T \in \mathbb{R}^{3\times 3}$$

where H is called a homography matrix. Certain conversions lead to the planar epipolar constraint with a pair of corresponding 2D points:

$$\hat{x_2}Hx_1 = 0$$

Again, we can cast this equation into the form $a^T H^s = 0$ where we have stacked the elements of H into a vector $H^s = (H_{11}, H_{21}, \ldots, H_{33})^T \in \mathbb{R}^9$ and introduced

the matrix $a \equiv x_1 \otimes \hat{x_2} \in \mathbb{R}^{9\times 3}$. Let us now assume we have $n \geq 4$ pairs of corresponding 2D points $\{x_1^j, x_2^j\}, j = 1, \ldots, n$ in the two images. Each point pair induces a matrix a^j, we integrate these into a larger matrix $\chi \equiv (a^1, \ldots, a^n)^T \in \mathbb{R}^{3n\times 9}$ and obtain the system

$$\chi H^s = 0$$

Four Point Algorithm

1. For the point pairs, compute the matrix χ
2. Compute a solution H^s for the above equation by SVD of χ
3. Extract the motion parameters from the homography matrix $H = R + \frac{1}{d} T N^T$

5.6. The Uncalibrated Case

The reconstruction algorithms introduced above all assume that the camera is calibrated ($K = 1$). If the parameters of K are known then one can simply transform the pixel coordinates x' to normalized coordinates $x = K^{-1}x'$ (for K see subsection 3.2) to obtain the representation used in the previous sections.
Reconstruction from uncalibrated views works with

$$x_2^{'T} \, F \, x_1' = 0$$

with the fundamental matrix defined as

$$F \equiv K^{-T} \hat{T} R K^{-1} = K^{-T} E K^{-1}$$

F has an SVD $F = U\Sigma V^T$ with $\Sigma = diag(\sigma_1, \sigma_2, 0)$.

6. Reconstruction from Multiple Views

6.1. From Two Views to Multiple Views

In this section, we deal with the problem of 3D reconstruction given multiple views of a static scene, either obtained simultaneously, or sequentially from a moving camera.

6.2. Preimage & Coimage from Multiple Views

The preimage of multiple images of points and lines can be defined by the intersection:

$$\begin{aligned} &preimage(x_1, \ldots, x_m) \\ &= preimage(x_1) \cap \cdots \cap preimage(x_m) \\ &preimage(\ell_1, \ldots, \ell_m) \\ &= preimage(\ell_1) \cap \cdots \cap preimage(\ell_m) \end{aligned}$$

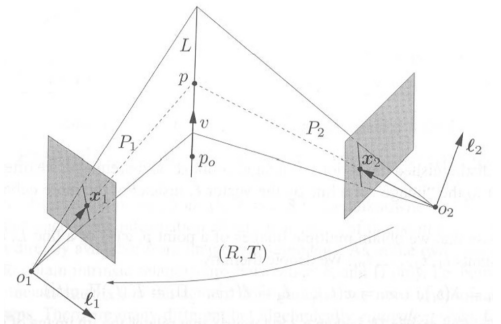


Figure 1: Images of a point p on a line L

- Preimages $P1$ and $P2$ of the image lines should intersect in the line L
- Preimages of the two image points x_1 and x_2 should intersect in the point p.
- Normals l_1 and l_2 define the coimages of the line L.

6.3. From Preimages to Rank Constraints

Point Features

The multiple-view projection matrix $\Pi \in \mathbb{R}^{3m\times 4}$ (see subsection 3.1) associated with the image matrix $\mathcal{I} \in \mathbb{R}^{3m\times m}$ gives

$$N_p \equiv (\Pi, \mathcal{I}) = \begin{pmatrix} \Pi_1 & x_1 & 0 & \dots & 0 \\ \Pi_2 & 0 & x_2 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \Pi_m & 0 & 0 & \dots & x_m \end{pmatrix} \in \mathbb{R}^{3m\times(m+4)}$$

with its implied rank constraint

$$rank(N_p) \leq m + 3$$

Some conversions also lead to

$$W_p \equiv \mathcal{I}^{\perp}\Pi = \begin{pmatrix} \hat{x_1}\Pi_1 \\ \hat{x_2}\Pi_2 \\ \vdots \\ \hat{x_m}\Pi_m \end{pmatrix} \in \mathbb{R}^{3m\times 4}$$

plus

$$rank(W_p) = rank(N_p) - m \leq 3$$

Line Features

Similarly,

$$W_l \equiv \begin{pmatrix} \ell_1^T\Pi_1 \\ \ell_2^T\Pi_2 \\ \vdots \\ \ell_m^T\Pi_m \end{pmatrix} \in \mathbb{R}^{m\times 4}$$

with the rank constraint

$$rank(W_l) \leq 2$$

6.4. Geometric Interpretation

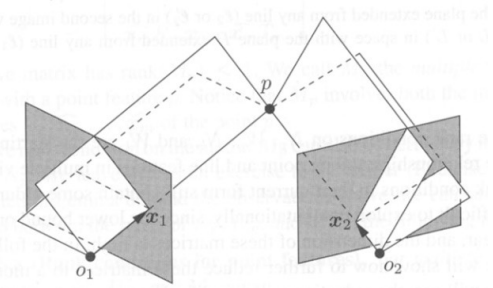


Figure 2: Preimage of two image points

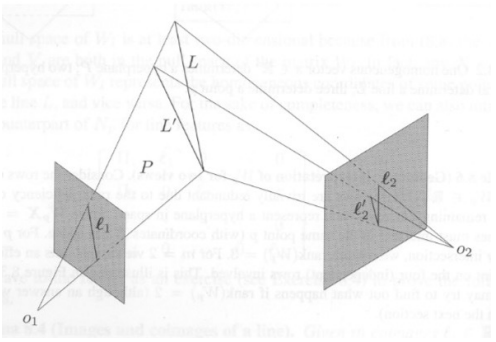


Figure 3: Preimage of two image lines

6.5. The Multiple-view Matrix

The matrix

$$M_p \equiv \begin{pmatrix} \hat{x_2}R_2x_1 & \hat{x_2}T_2 \\ \hat{x_3}R_3x_1 & \hat{x_3}T_3 \\ \vdots & \vdots \\ \hat{x_m}R_mx_1 & \hat{x_m}T_m \end{pmatrix} \in \mathbb{R}^{3(m-1)\times 2}$$

is called the multiple-view matrix associated with a point p.

In summary: For multiple images of a point p the matrices N_p, W_p and M_p satisfy:

$$rank(M_p) = rank(W_p) - 2 = rank(N_p) - (m + 2) \leq 1$$

6.6. Relation to Epipolar Constraints

M_p contains geometric information which is the epipolar constraint by

$$x_i^T \hat{T}_i R_i x_1 = 0$$

Furthermore, the trilinear constraint:

$$\hat{x_i}(T_i x_1^T R_j^T - R_i x_1 T_j^T)\hat{x_j} = 0$$

Let $x_1, x_2, x_3 \in \mathbb{R}^3$ be the 2D point coordinates in three camera frames with distinct optical centers. If the trilinear constraints

$$\hat{x_j}\left(T_{ji} x_i^T R_{ki}^T - R_{ji} x_i T_{ki}^T\right)\hat{x_k} = 0, \quad i, j, k = 1, 2, 3$$

(combination of the two trilinear constraints from above) hold then a unique preimage is determined unless the three lines associated to image points $x1$, $x2$, $x3$ are colinear.

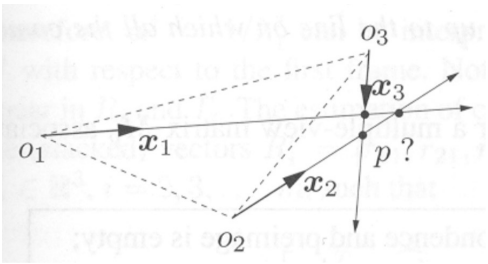


Figure 4: All pairs of lines intersect, yet it does not imply a unique 3D point p (a unique preimage)

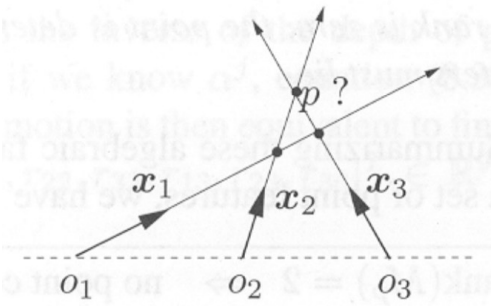


Figure 5: Again, all pairs of lines may intersect without there being a unique preimage p - fortunately, the trilinear constraint assures a unique preimage (unless p is also on the same line with the optical centers)

In summary we get:

$rank(M_p) = 2 \Rightarrow$ no point correspondence & empty preimage

$rank(M_p) = 1 \Rightarrow$ point correspondence & unique preimage

$rank(M_p) = 0 \Rightarrow$ point correspondence & preimage not unique

6.7. Multiple-View Reconstruction Algorithms

-

6.8. Multiple-View Reconstruction of Lines

The matrix

$$M_l \equiv \begin{pmatrix} \ell_2^T R_2 \hat{\ell}_1 & \ell_2^T T_2 \\ \vdots & \vdots \\ \ell_m^T R_m \hat{\ell}_1 & \ell_m^T T_m \end{pmatrix} \in \mathbb{R}^{(m-1)\times 4}$$

has the rank constraint

$$rank(M_l) \leq 1$$

From the trilinear constraint:

$$\ell_j^T T_j \ell_i^T R_i \hat{\ell}_1 - \ell_i^T T_i \ell_j^T R_j \hat{\ell}_1 = 0$$

one can conclude that any multiview constraint on lines can be reduced to constraints which only involve three lines at a time.

Lemma

Given three camera frames with distinct optical centers and any three vectors $\ell_1, \ell_2, \ell_3 \in \mathbb{R}^3$ that represent three image lines. If the three image lines satisfy the trilinear constraints

$$\ell_j^T T_{ji} \ell_k^T R_{ki} \hat{\ell}_i - \ell_k^T T_{ki} \ell_j^T R_{ji} \hat{\ell}_i = 0, \quad i, j, k \in \{1, 2, 3\}$$

(combination of the two trilinear constraints from above) then their preimage L is uniquely determined except for the case in which the preimage of every ℓ_i is the same plane in space. This is the only degenerate case, and in this case, the matrix M_l becomes zero.

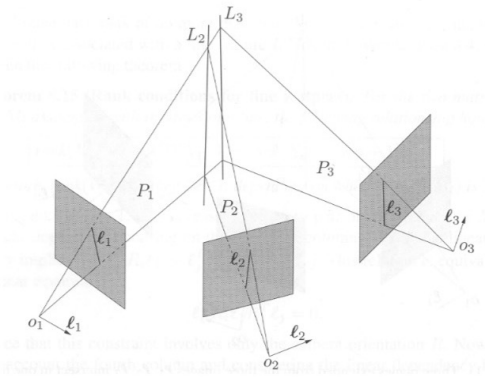


Figure 6: No preimage: The lines L_2 and L_3 do not coincide

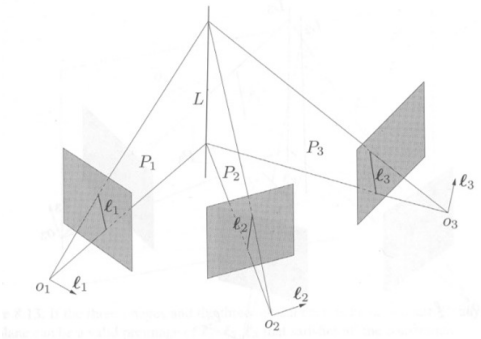


Figure 7: Uniqueness of the preimage: The lines L_2 and L_3 coincide

Overall we have the following cases:

$$rank(M_l) = 2 \Rightarrow \text{no line correspondence}$$
$$rank(M_l) = 1 \Rightarrow \text{line correspondence \& unique preimage}$$
$$rank(M_l) = 0 \Rightarrow 8\text{line correspondence \& preimage not unique}$$

Summary

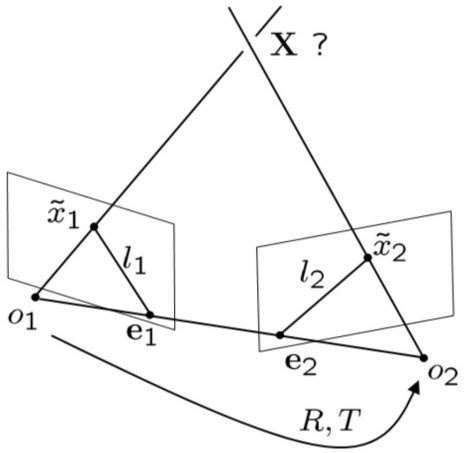
	(Pre) image	Coimage	Jointly
Point	$rank(N_p) \leq m+3$	$rank(W_p) \leq 3$	$rank(M_p) \leq 1$
Line	$rank(N_l) \leq m+3$	$rank(W_l) \leq 3$	$rank(M_l) \leq 1$

These rank conditions capture the relationships among corresponding geometric primitives in multiple images. They impose the existence of unique preimages (up to degenerate cases). Moreover, they give rise to natural factorization-based algorithms for multiview recovery of 3D structure and motion (i.e. generalizations of the eight-point algorithm).

7. Bundle Adjustment & Nonlinear Optimization

7.1. Optimality in Noisy Real World Conditions

For noisy data $\tilde{x_1}$, $\tilde{x_2}$ we have neither a guarantee that R and T are as close as possible to the true solution nor that we will get a consistent reconstruction.



Previous linear approaches could be elegantly computed in closed form. For noisy point locations, they often provide suboptimal performance or even fail.
A Bayesian formulation determines the most likely camera transformation R, T and true 2D coordinates x by performing a maximum aposteriori estimate. This approach however involves modeling probability densities $\mathcal{P}$ on a fairly complicated space.

7.2. Bundle Adjustment

Under the assumption that the observed 2D point coordinates $\tilde{x}$ are corrupted by zero-mean Gaussian noise, maximum likelihood estimation leads to bundle adjustment:

$$E(R,T,X_1,\ldots,X_N) = \sum_{j=1}^{N} |\tilde{x_1}^j - \pi(X_j)|^2 + |\tilde{x_2}^j - \pi(R,T,X_j)|^2$$

It aims at minimizing the reprojection error between the observed 2D coordinates $\tilde{x_i}^j$ and the projected 3D coordinate X_j with respect to camera 1.
For the general case of m images, we get:

$$E(\{R_i,T_i\}_{i=1\ldots m},\{X_j\}_{j=1\ldots N}) = \sum_{i=1}^{m}\sum_{j=1}^{N} \Theta_{ij}|\tilde{x_i}^j - \pi(R_i,T_i,X_j)|^2$$

with $T_1 = 0$ and $R_1 = 1$. $\Theta_{ij} = 1$ if point j is visible in image i, $\Theta_{ij} = 0$ else. The problems are non-convex.

The same optimization problem can be parameterized differently. We introduce x_i^j to denote the true 2D coordinate associated with the measured coordinate $\tilde{x_i}^j$:

$$E(\{x_1^j,\lambda_1^j\}_{j=1\ldots N},R,T) = \sum_{j=1}^{N} ||x_1^j - \tilde{x_1}^j||^2 + ||\tilde{x_2}^j - \pi(R\lambda_1^j x_1^j + T)||^2$$

Alternatively, a constrained optimization by minimizing the cost function:

$$E(\{x_i^j\}_{j=1\ldots N},R,T) = \sum_{j=1}^{N}\sum_{i=1}^{2} ||x_i^j - \tilde{x_i}^j||^2$$

with constraints

$$\begin{aligned} x_2^{jT}\hat{T}Rx_1^j &= 0 \text{ (epipolar constraint)} \\ x_1^{jT}e_3 &= 1 \ , \\ x_2^{jT}e_3 &= 1 \text{ (lying on image plane)} \\ j &= 1,\dots,N \end{aligned}$$

Bundle adjustment aims at jointly estimating 3D coordinates of points and camera parameters – typically the rigid body motion, but sometimes also intrinsic calibration parameters or radial distortion.
The name is derived from the attempt of adjusting the bundles of light rays emitted from the 3D points.
The highly non-convex cost function requires good initialization. Hence, it is used in the last step in a reconstruction pipeline.

7.3. Nonlinear Optimization

Nonlinear programming denotes the process of iteratively solving a nonlinear optimization problem, i.e. a problem involving the maximization or minimization of an objective function over a set of real variables under a set of equality or inequality constraints.
In the following we will discuss:

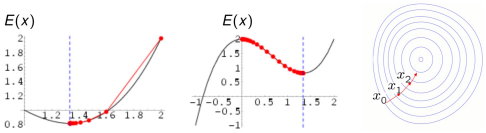
- the gradient descent,
- Newton methods,
- the Gauss-Newton algorithm,
- the Levenberg-Marquardt algorithm.

7.4. Gradient Descent

Gradient descent is a first-order optimization method. It aims at computing a local minimum of a (generally) non-convex cost function by iteratively stepping in the direction in which the energy decreases most. This is given by the negative energy gradient.

$$x_{k+1} = x_k - \epsilon \frac{dE}{dx}(x_k) \quad k = 0, 1, 2, \dots$$

For the case of convex E, the found local minimum will also be the global minimum. The step size ϵ can be chosen differently in each iteration.
Gradient descent is very popular, however not the fastest method in most cases.



7.5. Least Squares Estimation

-

7.6. Newton Methods

Newton methods are second-order methods. Let x_t be the estimated solution after t iterations. Then the Taylor approximation of the cost function $E(x)$ in the vicinity of this estimate is (1. Fitting the parabola):

$$\begin{aligned} E(x) \approx E(x_t) + g^T(x - x_t) \\ + \frac{1}{2}(x - x_t)^T H(x - x_t) \end{aligned}$$

The first and second derivative are denoted by the Jacobian $g = \frac{dE}{dx(x_t)}$ and the Hessian matrix $\frac{d^2E}{dx^2(x_t)}$. The optimality condition is (2. Going to minimum of parabola in each iteration):

$$\frac{dE}{dx} = g + H(x - x_t) = 0$$

Setting the next iterate to the minimizer x leads to:

$$x_{t+1} = x_t - \gamma H^{-1} g \quad \gamma \in (0, 1)$$

When applicable, second-order methods are often faster than first-order methods,

at least when measured in number of iterations.
For large optimization problems, computing and inverting the Hessian may be challenging.

7.7. The Gauss-Newton Algorithm

The Gauss-Newton algorithm is a method to solve non-linear least-squares problems of the form:

$$\min_x \sum_i r_i(x)^2$$

The approximation of the Newton method (dropping second order derivative)

$$H \approx 2J^T J \quad \text{with the Jacobian } J = \frac{dr}{dx}$$

together with $g = 2J^T$ leads to the Gauss-Newton algorithm

$$\begin{aligned} x_{t+1} &= x_t + \Delta \\ \Delta &= -(J^T J)^{-1} J^T r \end{aligned}$$

In contrast to the Newton algorithm, the Gauss-Newton algorithm does not require the computation of second derivatives.

7.8. The Levenberg-Marquardt Algorithm

The Newton algorithm can be modified by adding a bit of unit matrix I

$$x_{t+1} = x_t - (H + \lambda I_n)^{-1} g$$

to create a hybrid between the Newton method ($\lambda = 0$) and a gradient descent with step size $\frac{1}{\lambda}$ (for $\lambda \to \inf$)
In the same manner Levenberg suggested to damp the Gauss-Newton algorithm for nonlinear least squares:

$$\begin{aligned} x_{t+1} &= x_t + \Delta \\ \Delta &= -(J^T J + \lambda I_n)^{-1} J^T r \end{aligned}$$

Marquardt suggested a more adaptive component-wise damping of the form

$$\Delta = -(J^T J + \lambda diag(J^T J))^{-1} J^T r$$

which avoids slow convergence in directions of small gradient.

7.9. Summary

Bundle adjustment was pioneered in the 1950s as a technique for structure and motion estimation in noisy real-world conditions. It aims at estimating the locations of N 3D points X_j and camera motions (R_i,T_i), given noisy 2D projections $\tilde{x_i}^j$ in m images.
The solution to the weighted nonlinear least squares problem can be computed by various iterative algorithms, most importantly the Gauss-Newton algorithm or its damped version, the Levenberg-Marquardt algorithm.
Bundle adjustment is typically initialized by an algorithm such as the eight-point or five-point algorithm.

8. Direct Approaches to Visual SLAM

In the past chapters we have studied classical approaches to multiple view reconstruction. These methods tackle the problem of structure and motion estimation (visual SLAM) in several steps:

1. A set of feature points is extracted from the images – ideally points such as corners which can be reliably iden-

tified in subsequent images as well.

2. One determines a correspondence of these points across the various images. This can be done either through local tracking (using optical flow approaches) or by random sampling of possible partners based on a feature descriptor (SIFT, SURF, etc.) associated with each point.

3. The camera motion is estimated based on a set of corresponding points. In many approaches this is done by a series of algorithms such as the eight-point algorithm or the five-point algorithm followed by bundle adjustment.

4. For a given camera motion one can then compute a dense reconstruction using stereo reconstruction methods.

Such classical approaches are indirect in the sense that they do not compute structure and motion directly from the images but rather from a sparse set of precomputed feature points. Despite a number of successes, they have several drawbacks:

- From the point of view of statistical inference, they are suboptimal: In the selection of feature points much potentially valuable information contained in the colors of each image is discarded.

- They invariably lack robustness: Errors in the point correspondence may have devastating effects on the estimated camera motion. Since one often selects very few point pairs only (8 points for the eight-point algorithm, 5 points for the five-point algorithm), any incorrect correspondence will lead to an incorrect motion estimate.

- They do not address the highly coupled problems of motion estimation and dense structure estimation. They merely do so for a sparse set of points. As a consequence, improvements in the estimated dense geometry will not be used to improve the camera motion estimates.

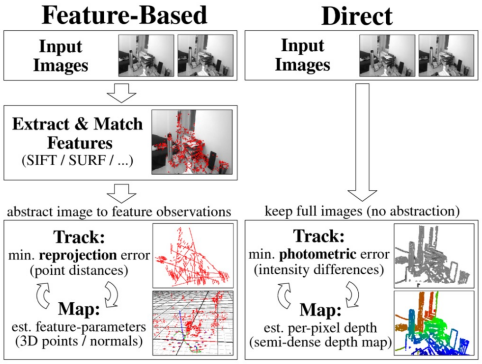
8.1. Direct Methods

Rather than extracting a sparse set of feature points to determine the camera motion, direct methods aim at estimating camera motion and dense or semi-dense scene geometry directly from the input images.
This has several advantages:

- Direct methods tend to be more robust to noise and other nuisances because they exploit all available input information.

- Direct methods provide a semi-dense geometric reconstruction of the scene which goes well beyond the sparse point cloud generated by the eight-point algorithm or bundle adjustment. Depending on the application, a separate dense reconstruction step may no longer be necessary.

- Direct methods are typically faster because the feature-point extraction and correspondence finding is omitted: They can provide fairly accurate camera motion and scene structure in real-time on a CPU.

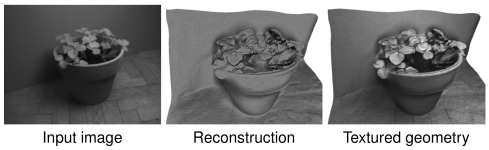


8.2. Realtime Dense Geometry

Let $g_i \in SE(3)$ be the rigid body motion from the first camera to the i-th camera, and let $I_i : \Omega \to \mathbb{R}$ be the i-th image. A dense depth map $h : \Omega \to \mathbb{R}$ can be computed by solving the optimization problem:

$$\min_h \sum_{i=2}^{n} \int_\Omega |I_1(x) - I_i(\pi g_i(hx))| \, \mathrm{d}x + \lambda \int_\Omega |\nabla h| \, \mathrm{d}x$$

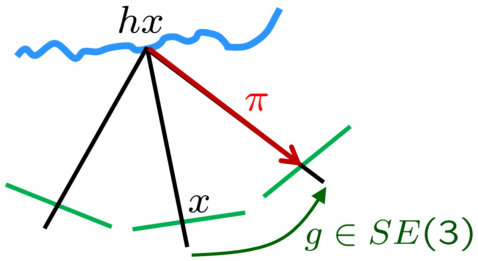
where a pixel $x \in \Omega$ is represented in homogeneous coordinates and hx is the corresponding 3D point.
The transformation into the other images I_i should give rise to the same color as in the reference image I_1.



The approach (Stuehmer, Gumhold, Cremers, DAGM 2010) relies on a sparse feature-point based camera tracker (PTAM) and computes dense geometry directly on the images.

8.3. Dense RGB-D Tracking

Steinbruecker, Sturm, Cremers (2011) propose a complementary approach to directly compute the camera motion from RGB-D images. The idea is to compute the rigid body motion g_ξ which optimally aligns two subsequent color images I_1 and I_2:



$$\min_{\xi \in se(3)} \int_\Omega |I_1(x) - I_2(\pi g_\xi(hx))|^2 \, \mathrm{d}x$$

The above non-convex problem can be approximated as a convex problem by linearizing the residuum around an initial guess ξ_0:

$$E(\xi) \approx \int_\Omega \left| I_1(x) - I_2(\pi g_{\xi 0}(hx)) - \nabla I_2^T \left(\frac{d\pi}{dg_\xi} \right) \left(\frac{dg_\xi}{d\xi} \right) \xi \right|^2 \mathrm{d}x$$

This is a convex quadratic cost function which gives rise to a linear optimality condition:

$$\frac{dE(\xi)}{d\xi} = A\xi + b = 0$$

For small camera motions, this image aligning approach provides more accurate camera motion than the commonly used generalized Iterated Closest Points (GICP) approach.

Kerl, Sturm, Cremers, IROS 2013 propose an extension of the RGB-D camera tracker which combines color consistency and geometric consistency of subsequent RGB-D images. Assuming that the vector $r_i =$

$(r_{ci}, r_{zi}) \in \mathbb{R}^2$ containing the color and geometric discrepancy for pixel i follows a bivariate t-distribution, the maximum likelihood pose estimate can be computed as:

$$\min_{\xi \in \mathbb{R}^6} \sum_i w_i r_i^T \Sigma^{-1} r_i$$

with weights w_i based on the student t-distribution:

$$w_i = \frac{v+1}{v + r_i^T \Sigma^{-1} r_i}$$

Solvable by alternating a Gauss-Newton style optimization with a re-estimation of the weights w_i and the matrix Σ.

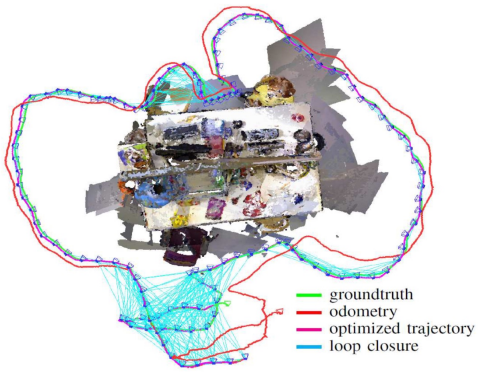
8.4. Loop Closure and Global Consistency

When tracking a camera over a longer period of time, errors tend to accumulate. While a single room may still be mapped more or less accurately, mapping a larger environment will lead to increasing distortions.

A remedy is to introduce pose graph optimization and loop closuring, a technique popularized in laser-based SLAM systems. The key idea is to estimate the relative camera motion $\hat{\xi_{ij}}$ for any camera pair i and j in a certain neighborhood. Subsequently, one can determine a globally consistent camera trajectory $\xi = \xi_{i i=1\ldots T}$ by solving the nonlinear least squares problem

$$\min_{\xi} \sum_{i \sim j} \left(\hat{\xi_i} j - \xi_i \circ \xi_j^{-1}\right)^T \Sigma_{ij}^{-1} \left(\hat{\xi_{ij}} - \xi_i \circ \xi_j^{-1}\right)$$

where Σ_{ij}^{-1} denotes the uncertainty of measurement $\hat{\Sigma_{ij}}$. The problem can be solved by Levenberg-Marquardt algorithm for instance.



8.5. Dense Tracking and Mapping

Newcombe, Lovegrove & Davison (ICCV 2011) propose an algorithm which computes both the geometry of the scene and the camera motion from a direct and dense algorithm.

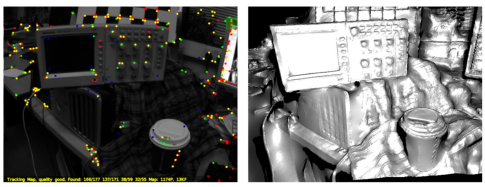
They compute the inverse depth $u = \frac{1}{h}$ by minimizing a cost function of the form

$$\min_u \sum_{i=2}^{n} \int_\Omega \left| I_1(x) - I_i\left(\pi g_i \left(\frac{x}{u}\right)\right)\right| \, \mathrm{d}x + \lambda \int_\Omega \phi(x) |\nabla u| \, \mathrm{d}x$$

for fixed camera motions g_i . The function ϕ introduces an edge-dependent weighting assigning small weights in locations where the input images exhibit strong gradients:

$$\phi(x) = exp(-|\nabla I_\sigma(x)|^\alpha)$$

The camera tracking is then performed with respect to the textured reconstruction in a manner similar to Steinbrücker et al. (2011). The method is initialized using feature point based stereo.



8.6. Large Scale Direct Monocular SLAM

A method for real-time direct monocular SLAM is proposed in Engel, Sturm, Cremers, ICCV 2013 and Engel, Schöps, Cremers, ECCV 2014. It combines several contributions which make it well-suited for robust large-scale monocular SLAM:

- Rather than tracking and putting into correspondence a sparse set of feature points, the method estimates a semi-dense depth map which associates an inverse depth with each pixel that exhibits sufficient gray value variation.
- To account for noise and uncertainty each inverse depth value is associated with an uncertainty which is propagated and updated over time like in a Kalman filter.
- Since monocular SLAM is invariably defined up to scale only, we explicitly facilitate scaling of the reconstruction by modeling the camera motion using the Lie group of 3D similarity transformations Sim(3).
- Global consistency is assured by loop closuring on Sim(3).

Since reconstructions from a monocular camera are only defined up to scale, Engel, Schöps, Cremers, ECCV 2014 account for rescaling of the environment by representing the camera motion as an element in the Lie group of 3D similarity transformations Sim(3) which is defined as:

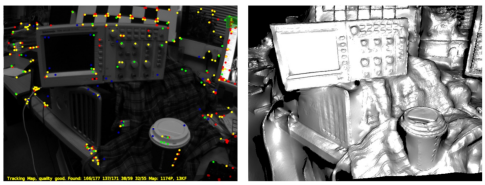
$$Sim(3) = \left\{ \begin{pmatrix} sR & T \\ 0 & 1 \end{pmatrix} \right.$$
$$\left. \text{with } R \in SO(3), T \in \mathbb{R}^3, s \in \mathbb{R}_+ \right\}$$

One can minimize a nonlinear least squares problem

$$min_{\xi \in sim(3)} \sum_i w_i r_i^2(\xi)$$

where r_i denotes the color residuum across different images and w_i a weighting as suggested in Kerl et al. IROS 2013. The above cost function can then be optimized by a weighted Gauss-Newton algorithm on the Lie group $Sim(3)$:

$$\xi^{t+1} = \Delta_\xi \circ \xi^{(t)}$$
$$\text{with } \Delta_\xi = (J^T W J)^{-1} J^T W r, \quad J = \frac{\partial r}{\partial \xi}$$



Despite its popularity, LSD SLAM has several shortcomings:

- While the pose graph optimization allows to impose global consistency, it merely performs a joint optimization of the extrinsic parameters associated with all keyframes. In contrast to a full bundle adjustment, it does not optimize the geometry. This is hard to do in realtime, in particular for longer sequences.
- LSD SLAM actually optimizes two different cost functions for estimating geometry and camera motion.
- LSD SLAM introduces spatial regularity by a spatial filtering of the inverse depth values. This creates correlations among the geometry parameters which in turn makes Gauss-Newton optimization difficult.

- LSD SLAM is based on the assumption of brightness constancy. In real-world videos, brightness is often not preserved. Due to varying exposure time, vignette and gamma correction, the brightness can vary substantially. While feature descriptors are often invariant to these changes, the local brightness itself is not.

8.7. Direct Sparse Odometry

Engel, Koltun, Cremers, PAMI 2018 suggest that brightness variations due to vignette, gamma correction and exposure time can be eliminated by a complete photometric calibration:

$$I(x) = G\left(tV(x)B(x)\right)$$

where the measured brightness I depends on the irradiance B, the vignette V , the exposure time t and the camera response function G (gamma function). G and V can be calibrated beforehand, t can be read out from the camera.
A complete bundle adjustment over longer sequences is difficult to carry out in real-time because the number of 3D point coordinates may grow very fast over time. Furthermore new observations are likely to predominantly affect parameters associated with neighboring structures and cameras. For a given data set, one can study the connectivity graph, i.e. a graph where each node represents an image and two nodes are connected if they look at the same 3D structure.
Direct Sparse Odometry therefore reverts to a windowed joint optimization, the idea being that from all 3D coordinates and camera frames only those in a recent time window are included. The remaining ones are marginalized out.
If one avoid spatial filtering and selects only a sparser subset of points, then the points can be assumed to be fairly independent. As a result the Hessian matrix becomes sparser and the Schur complement can be employed to make the Gauss-Newton updates more efficient.

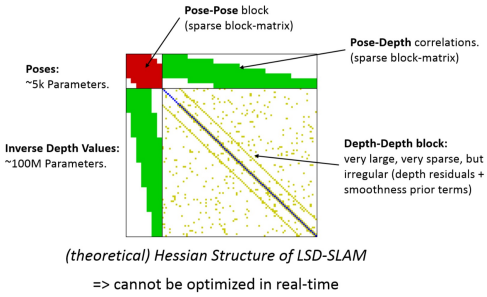


Figure 8: Effect of Spatial Correlation on the Hessian Matrix

Solving the Newton update step (called normal equation)

$$Hx = \begin{pmatrix} H_{\alpha\alpha} & H_{\alpha\beta} \\ H_{\alpha\beta} & H_{\beta\beta} \end{pmatrix} \begin{pmatrix} x_\alpha \\ x_\beta \end{pmatrix} = \begin{pmatrix} g_\alpha \\ g_\beta \end{pmatrix}$$

for the unknowns x_α and x_β is usually done by QR decomposition for large problems. Here, left-multiplication with the matrix

$$\begin{pmatrix} I & -H\alpha\beta H_{\beta\beta}^{-1} \\ 0 & I \end{pmatrix}$$

leads to

$$\begin{pmatrix} S & 0 \\ H_{\alpha\beta}^T & H_{\beta\beta} \end{pmatrix} \begin{pmatrix} x_\alpha & x_\beta \end{pmatrix} = \begin{pmatrix} g_\alpha - H_{\alpha\beta}H_{\beta\beta}^{-1}g_\beta \\ g_\beta \end{pmatrix}$$

where $S = H_{\alpha\alpha} - H_{\alpha\beta}H_{\beta\beta}^{-1}H_{\alpha\beta}^T$ is the Schur complement of $H_{\beta\beta}$ in H.

9. Variational Methods: A Short Intro

9.1. Variational Methods

Variational methods are a class of optimization methods. They are popular because they allow to solve many problems in a mathematically transparent manner. Instead of implementing a heuristic sequence of processing steps (as was commonly done in the 1980's), one clarifies beforehand what properties an optimal solution should have.
That does not require extensive training data like for better-performing neural networks.
Applications are e.g. spatially dense multiple view reconstruction or motion estimation and optical flow.
Variational methods have many advantages over heuristic multi-step approaches (such as the Canny edge detector):

- A mathematical analysis of the considered cost function allows to make statements on the existence and uniqueness of solutions.
- Approaches with multiple processing steps are difficult to modify. All steps rely on the input from a previous step. Exchanging one module by another typically requires to re-engineer the entire processing pipeline.
- Variational methods make all modeling assumptions transparent, there are no hidden assumptions.
- Variational methods typically have fewer tuning parameters. In addition, the effect of respective parameters is clear.
- Variational methods are easily fused – one simply adds respective energies / cost functions.

9.2. Variational Image Smoothing

9.3. Euler-Lagrange Equation

9.4. Gradient Descent

9.5. Adaptive Smoothing

9.6. Euler and Lagrange

A. Special Matrices and Properties

symmetric matrices $A^T = A$
$\Rightarrow \langle Ax, y\rangle = \langle x, Ay\rangle$ and eigenvectors $\langle v_i, v_j\rangle = 0 \quad i \neq j$

skew-symmetric/antisymmetric matrices $A^T = -A$
$\Rightarrow \begin{pmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{pmatrix} \cdot b = a \times b \quad a, b \in \mathbb{R}^3$

orthogonal matrices $Q^T = Q^{-1}$
$\Rightarrow$ eigenvectors form a basis
$\Rightarrow QQ^T = Q^TQ = I$

singular matrices $det(A) = 0$
$\Rightarrow$ A not invertible

diagonal matrices all entries but the main diagonal are zero

positive-definite $\lambda_1, \ldots, \lambda_n > 0$
$\Rightarrow x^TAx > 0 \quad x \neq \vec{0}$

positive-semidefinite $\lambda_1, \ldots, \lambda_n \geq 0$

negative-definite $\lambda_1, \ldots, \lambda_n < 0$

indefinite $-\infty < \lambda_1, \ldots, \lambda_n < \infty$

B. Rotation Matrices in SO(3)

Rotation around x-axis:

$$R_x(\alpha) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & cos\alpha & -sin\alpha \\ 0 & sin\alpha & cos\alpha \end{pmatrix}$$

Rotation around y-axis:

$$R_y(\alpha) = \begin{pmatrix} cos\alpha & 0 & sin\alpha \\ 0 & 1 & 0 \\ -sin\alpha & 0 & cos\alpha \end{pmatrix}$$

Rotation around z-axis:

$$R_z(\alpha) = \begin{pmatrix} cos\alpha & -sin\alpha & 0 \\ sin\alpha & cos\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix}$$