3D Computer Vision

Structure from motion II

Prof. Didier Stricker

Dr. Gabriele Bleser

Kaiserlautern University

http://ags.cs.uni-kl.de/

DFKI – Deutsches Forschungszentrum für Künstliche Intelligenz

http://av.dfki.de
Introduction

- Previous lecture: structure from motion I
  - Robust fundamental matrix estimation
  - Factorization
  - Intro: structure and motion loop

- Today: structure from motion II
  - Structure and motion loop
  - Triangulation
  - Drift reduction methods
  - State-of-the-art approaches

- Next lectures: dense reconstruction
Recall: structure and motion (SAM)

Reconstruct

- Sparse scene geometry
- Camera motion

Unknown camera viewpoints
Euclidean form:

\[ m_n = \frac{1}{z_c} m_c \quad m_c = R_{cw}(m_w - c_w) \]

Homogeneous (matrix) form:

\[ m_n \propto (R_{cw} - R_{cw}c_w) \begin{pmatrix} m_w \\ 1 \end{pmatrix} \]

Homogenization = division through \( z_c \)
Problem statement: structure and motion

- **Assumption:** camera is calibrated
- **Given:** $l$ matching (normalized) image points $m_{n, t}^{(j)}$ over $k$ views
- **Find:** the cameras $s_t = \{R_{cw}, c_w\}_t$ and the 3D points $m_w^{(j)}$ such that
  \[
  m_{n, t}^{(j)} \propto (R_{cw, t} \quad w_{c, t}) \begin{pmatrix} m_w^{(j)} \\ 1 \end{pmatrix} \quad \forall \ 1 \leq t \leq k, 1 \leq j \leq l
  \]
- **Number of parameters**
  - for each camera there are 6 parameters
  - for each 3D point there are 3 parameters
  - Total of $6k + 3l$ parameters must be estimated
  - e.g. 50 frames, 1000 points: 3300 unknowns
**Offline vs. online structure and motion**

**Offline:**
- E.g. as basis for dense 3D model reconstruction
- No real-time requirements, all images are available at once

**Online:**
- E.g. for mobile Augmented Reality in unknown environments
- Real-time requirements, images become available one by one, output required at each time-step
Offline vs. online structure and motion

Offline:

- Use MANY features and computationally intense image processing methods
- Forward and backward feature matching
- Batch processing, e.g. bundle adjustment

Online:

- Reduced number of features and lightweight image processing methods
- Sequential/recursive processing scheme
- Possibly local bundle adjustment (keyframe/window based)

→ harder problem
Online structure and motion (basic pipeline)

- **Image processing**
- **Pose estimation**
- **Scene model**
- **Structure estimation**

Additional sensor information?
Online structure and motion (calibrated case)

- Alternating estimation of camera poses and 3D feature locations (triangulation) from a (continuous) image sequence.

How do we initialize the first camera poses?

\[
\begin{align*}
    \text{2D feature location} & \quad \text{(from image processing)} \\
    \text{Camera pose} & \\
    t = 1 & \quad \text{Object points} \\
    t = 2 & \quad \text{Camera poses}
\end{align*}
\]

\[
s_t = \{R_{cw}, c_w\}_t
\]
• Compute the fundamental matrix, $F$, from point correspondences.

• Compute the camera poses from the fundamental matrix by factorization:
  Recall: $F = K'^{-T}[t]_x R K^{-1}$
  Obtain: $P = K[I|0], P' = K'[R|t]$

• Note:
  – $K$ and $K'$ are the intrinsic camera parameters
  – $P$ and $P'$ are the projection matrices of both cameras
  – $R$ and $t$ are the relative rotation and translation between both cameras
Epipolar constraint: calibrated case

The vectors $x$, $t$, and $Rx'$ are coplanar

$$x \cdot [t \times (Rx')] = 0 \quad \rightarrow \quad x^T Ex' = 0 \quad \text{with} \quad E = [t \times]R$$

**Essential Matrix**
(Longuet-Higgins, 1981)
Problem statement

- **Given:** \( n \) corresponding points \( \{x_i \leftrightarrow x'_i, i = 1, \ldots, n\} \)
- compute the fundamental matrix \( F \)
- such that
  \[
  x'_i \top F x_i = 0 \quad 1 \leq i \leq n
  \]

Solution

Each point correspondence \( x_i \leftrightarrow x'_i \) generates one constraint on \( F \)

\[
(x'_i \ y'_i \ 1) \begin{bmatrix} f_1 & f_2 & f_3 \\ f_4 & f_5 & f_6 \\ f_7 & f_8 & f_9 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} = 0
\]

which may be written

\[
x' x f_1 + x' y f_2 + x' f_3 + y' x f_4 + y' y f_5 + y' f_6 + x f_7 + y f_8 + f_9 = 0
\]
The “8-point” algorithm – Least squares solution

Given \( n \) corresponding points (\( n \) is typically hundreds) with noise on their measured positions

For \( n > 8 \) point correspondences, \( \mathbf{A} \) is a \( n \times 9 \) matrix,

\[
\begin{bmatrix}
    x'_1 x_1 & x'_1 y_1 & x'_1 & y'_1 x_1 & y'_1 y_1 & y'_1 & x_1 & y_1 & 1 \\
    \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
    x'_n x_n & x'_n y_n & x'_n & y'_n x_n & y'_n y_n & y'_n & x_n & y_n & 1
\end{bmatrix}
\quad \mathbf{f} = \mathbf{0}
\]

and in general there will not be an exact solution to \( \mathbf{A} \mathbf{f} = \mathbf{0} \).

\( \mathbf{A} \) (linear) solution which minimises \( \| \mathbf{A} \mathbf{f} \| \), subject to \( \| \mathbf{f} \| = 1 \) is obtained from the eigenvector with least eigenvalue of \( \mathbf{A}^\top \mathbf{A} \).

See lecture on parameter estimation!
Decomposing the fundamental matrix

\[ F = K'^{-T} [t] \times RK^{-1} \]

Form the Essential matrix \( E = [t] \times R = K'^{T} FK \)

1. Compute \( t \) as left null-vector of \( E \), i.e. \( E^T t = 0 \)
   This determines \( t \) up to scale.

2. Compute \( R \) from \( E \)
   There are two solutions \( R_1 \) and \( R_2 \).

3. Set \( P = K[I \mid 0] \) for the first camera

   The four solutions for the second camera are
   \[ P' = K'[R_1 \mid \mu t] \quad P' = K'[R_1 \mid -\mu t] \quad \mu > 0 \]
   \[ P' = K'[R_2 \mid \mu t] \quad P' = K'[R_2 \mid -\mu t] \]
The four camera solutions

The 3D point is only in front of both cameras in one case
Online structure and motion (calibrated case)

- Alternating estimation of camera poses and 3D feature locations (triangulation) from a (continuous) image sequence.

Now we have the two first camera poses. How do we go on?

\[ t = 1 \]

2D feature location (from image processing)

Camera pose

\[ t = 2 \]

\[ m_{n,t}^{(j)} \]

\[ s_t = \{ R_{cw}, c_w \}_t \]
Online structure and motion (calibrated case)

- Alternating estimation of camera poses and 3D feature locations (triangulation) from a (continuous) image sequence.

\[
3D \text{ feature location} \quad m^{(j)}_w
\]

\[
2D \text{ feature location (from image processing)} \quad m^{(j)}_{n,t}
\]

\[
\text{Camera pose} \quad s_t = \{R_{cw}, c_w\}_t
\]
Online structure and motion (calibrated case)

- Alternating estimation of camera poses and 3D feature locations (triangulation) from a (continuous) image sequence.

$$s_t = \{R_{cw}, c_w\}_t$$

Estimate next camera pose (now from 2D/3D correspondences)
Online structure and motion (calibrated case)

- Alternating estimation of camera poses and 3D feature locations (triangulation) from a (continuous) image sequence.

\[ m_w^{(j)} \]

3D feature location

2D feature location (from image processing)

Camera pose

\[ s_t = \{ R_{cw}, c_w \}_t \]

Triangulate additional 3D points
Online structure and motion (calibrated case)

- Alternating estimation of camera poses and 3D feature locations (triangulation) from a (continuous) image sequence.

\[ s_t = \{R_{cw}, c_w\}_t \]

Refine known 3D points with new camera poses
Online structure and motion (calibrated case)

- Alternating estimation of camera poses and 3D feature locations (triangulation) from a (continuous) image sequence.

Refine known cameras with new 3D points

E.g. for some selected keyframes or more extensively in offline SAM
Structure and motion: the big problem

- Drift: accumulating error
Outline

Until now:
- Overview: (online) structure and motion pipeline
- Reminder: F-matrix estimation and decomposition
- Reminder: P-matrix estimation

Now:
- Triangulation
- Drift reduction methods (online/offline)
- State-of-the-art structure and motion systems
Triangulation
Estimate the 3D point, $X$, given at least 2 known cameras, $C$ and $C'$, and 2 corresponding feature points, $x$ and $x'$, (i.e. 2 camera views).

Parallel cameras (simple case)

\[
K = K' = \begin{bmatrix}
    f & 0 & 0 \\
    0 & f & 0 \\
    0 & 0 & 1
\end{bmatrix} \quad R = I \quad t = \begin{pmatrix}
    t_x \\
    0 \\
    0
\end{pmatrix}
\]

\[
Z = \frac{ft_x}{d}, \quad d = x' - x
\]

- Then, \( y = y' \) and depth \( Z \) can be computed from disparity:

- Derivation via equal triangles

\[
\frac{x}{f} = \frac{X}{Z} \quad \frac{x'}{f} = \frac{X + t_x}{Z}
\]

\[
\frac{x'}{f} = \frac{x}{f} + \frac{t_x}{Z}
\]

- Note: feature displacement (disparity) is inversely proportional to depth

as \( d \to 0, Z \to \infty \)
General configuration

- Arbitrary camera poses \( \Rightarrow \) 3D triangulation required
Triangulation

- Alternating estimation of camera poses and 3D feature locations (triangulation) from a (continuous) image sequence.

Refine known 3D points with new camera poses
Triangulation: vector solution

- Compute the mid-point of the shortest line between the two rays

How could this be solved otherwise?
Triangulation: algebraic solution

- Equation for one camera view (camera pose, \( \mathbf{c} \), and feature point, \( \mathbf{p} \))

\[ x \propto P X \iff \lambda x = PX \iff x \times \begin{bmatrix} p_1^T \\ p_2^T \\ p_3^T \end{bmatrix} X = 0 \]

4 entries, 3 DOF

\( i^{th} \) row of \( P \)

(3x4) matrix

\[ x(p^3^T X) - (p^1^T X) = 0 \]
\[ y(p^3^T X) - (p^2^T X) = 0 \]
\[ x(p^2^T X) - y(p^1^T X) = 0 \]

(2x4) for one equation

3 equations, only 2 linearly independent
\( \Rightarrow \) drop third row
Triangulation: algebraic solution

- Stack equations for all camera views:

\[
\begin{bmatrix}
A_1 \\
A_2 \\
\vdots \\
A_n
\end{bmatrix} X = 0
\]

- \( X \) has 3 DOF (4 parameters due to homogeneous representation)
- One camera view gives two linearly independent equations
- Two camera views are sufficient for a minimal solution
- Solution for \( X \) is eigenvector of \( A^T A \) corresponding to smallest eigenvalue
- Homogenize solution for \( X \) to obtain Euclidean vector

Minimizes algebraic error \( \rightarrow \) no geometrical interpretation
Triangulation: minimize geometric error

- Estimate 3D point $\hat{X}$ which exactly satisfies the supplied camera geometry and $P, P'$, so it projects as 
  
  $\hat{x} \propto PX \quad \hat{x} \propto P'X$

  where $\hat{x}$ and $\hat{x}'$ are closest to the actual image measurements.

\[
\begin{align*}
\min_{\hat{X}} & \quad d(x, \hat{x})^2 + d(x', \hat{x}')^2 \\
\text{subject to} & \quad \hat{x} = P\hat{X} \quad \text{and} \quad \hat{x}' = P'\hat{X}
\end{align*}
\]

Assumes perfect camera poses!

Nonlinear problem: can be solved with e.g. Levenberg-Marquard → parameter estimation lecture
Bundle Adjustment

A valid solution $R_1|t_1$, $R_2|t_2$, $R_3|t_3$ and $X^1, X^2, X^3, \ldots$ must let the Re-projection close to the Observation, i.e. to minimize the reprojection error

$$\min_{\hat{X}, R_i, t_i} \left( \begin{array}{c|c|c|c} \hat{x}_i^j & K & R_i|t_i & X_j \end{array} \right)^2$$
- Global bundle adjustment: jointly optimize over all camera poses and 3D points (previous lecture)

\[ \hat{x} = \arg \min \sum_{t=1}^{k} \sum_{l=1}^{n} r^{(i)} \]

- Estimate
- Minimize over parameter vector containing all camera poses and 3D points
- Over all frames and 3D points
- Residual/reprojection error

6 parameters for each camera + 3 for each 3D point
\[ 6k + 3l \] parameters must be estimated
\[ \Rightarrow \] matrices are sparse!

Nonlinear estimation problem: use e.g. Levenberg-Marquard
Open source libraries available, e.g. Sparse Bundle Adjustment (SBA)
Euler angle representation

Euler Angles for Rotation $R=R_z R_y R_x$:

- **Sequence not standardized!**

rotation by $\psi$ about the z axis: $R_z = \begin{bmatrix} \cos \psi & -\sin \psi & 0 \\ \sin \psi & \cos \psi & 0 \\ 0 & 0 & 1 \end{bmatrix}$

rotation by $\theta$ about the y axis: $R_y = \begin{bmatrix} \cos \theta & 0 & \sin \theta \\ 0 & 1 & 0 \\ -\sin \theta & 0 & \cos \theta \end{bmatrix}$

rotation by $\phi$ about the x axis: $R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \phi & -\sin \phi \\ 0 & \sin \phi & \cos \phi \end{bmatrix}$

+ Minimal parametrization, nice geometric interpretation
- Periodicity, difficult to interpolate
Axis-angle representation

- A rotation can be represented as a 3D vector
  - direction ⇒ rotation axis \( \mathbf{r} = [a, b, c]^T \)
  - length ⇒ rotation angle \( \theta = ||\mathbf{r}|| \)
- Conversion to rotation matrix by Rodrigues’ formula:

\[
R = \text{rot}(\mathbf{r}) = \mathbf{I}_3 + k(\theta) \mathbf{S}(\mathbf{r}) + g(\theta) \mathbf{S}(\mathbf{r})^2
\]

\[
\mathbf{S}(\mathbf{v}) = \begin{bmatrix} 0 & -v_z & v_y \\ v_z & 0 & -v_x \\ -v_y & v_x & 0 \end{bmatrix} \quad k(\theta) = \frac{\sin(\theta)}{\theta} \quad g(\theta) = \frac{1-\cos(\theta)}{\theta}
\]

+ Minimal parametrization
- Singularity at \( \theta=0 \)
QuatERNion repREsentation

- Compromise between rotation matrix and minimal representation
- A quaternion is a 4D vector:

\[ q = [q_w, q_x, q_y, q_z]^T \quad q_v = [q_x, q_y, q_z]^T \]

- Conversion to rotation matrix:

\[
R = \text{rot}(q) = \begin{bmatrix}
q_w^2 + q_x^2 - q_y^2 - q_z^2 & 2(q_xq_y - q_zq_w) & 2(q_zq_x + q_wq_y) \\
2(q_xq_y + q_wq_z) & q_w^2 - q_x^2 + q_y^2 - q_z^2 & 2(q_yq_z - q_wq_x) \\
2(q_zq_x - q_wq_y) & 2(q_yq_z + q_wq_x) & q_w^2 - q_x^2 - q_y^2 + q_z^2
\end{bmatrix}
\]

+ Easier conversion to rotation matrix
+ Singularity-free, can be interpolated
- Not minimal, 4 parameters, unit magnitude constraint
Triangulation: properties

- The smaller the angle (small baseline, big distance), the bigger the reconstruction uncertainty!
Drift reduction methods

Online and offline methods
Drift reduction (feature level)

- Reduce drift in feature tracking/matching
- Extend feature tracks
- Reacquire lost features

- See topics: image processing, recursive filtering, advanced visual detection and tracking algorithms
Drift reduction (geometry level)

- Careful (precise, robust) 3D point initialization
- How?
  - Triangulate over a whole set of camera views (>> 2)
  - Enforce a minimal angle $\theta$ between view rays
  - Use RANSAC to eliminate outliers
  - Use only well reconstructed points for further estimation

- Incorporate uncertainties, e.g. simple stochastic model and WLS estimation $\Rightarrow$ lecture on parameter estimation:
  All entities modelled as Gaussian random variables
Initial triangulation using RANSAC

Algorithm:
- If: store the new camera view
- If size of history > n
  - Until no more options or valid result found
    - Triangulate pairs of camera views
    - Validate the results
- If a valid result is found:
  - Compute a least squares (LS) estimate from all inliers
  - Compute a weighted least squares (WLS) estimate from all inliers

See lecture on parameter estimation!
3D point refinement

- Incorporate new camera view, each time the feature is observed in an image
- Methods:
  - Repeated triangulation
  - Recursive filtering
    (e.g. extended Kalman filter)
Results: online SAM (360° rotation)

With drift reduction

Without drift reduction
State-of-the-art approaches

Published at International Symposium on Mixed and Augmented Reality (ISMAR)
Title: Online camera pose estimation in partially known and dynamic scenes

Real-time structure and motion for Augmented Reality applications

Topics: feature tracking with optical flow, weighted least squares estimation, recursive filtering for structure estimation, feature quality tracking, map management
Bleser et al., ISMAR 2006

2D feature tracking
(Lost, Valid)

3D augmentation

- Line model
- Projected 3D covariances
- Feature quality (colour gradient)

External view on 3D uncertainty ellipsoids
Klein and Murray, ISMAR 2007

- **Title:** *Parallel tracking and mapping for small AR workspaces*
- Known as PTAM system
- MANY features, (simple) correlation based tracking
- Parallel pose tracking and 3D reconstruction threads
- Local bundle adjustment (based on keyframes)
- Code, videos, papers, slides available [here](#)
Ventura and Höllerer, ISMAR 2012

- **Title:** *Wide-Area Scene Mapping for Mobile Visual Tracking*

- Offline structure and motion → tracking model (sparse point cloud)
- Online tracking and model extension using mobile devices
- Gyroscopes as additional sensors for capturing quick rotations
- Server-side localization, jitter reduction

- [Paper](#)
- [Video](#)
- [Code](#)
Tan et al., ISMAR 2013

- Title: *Robust monocular SLAM in Dynamic Environments*
- Based on PTAM
- Uses invariant features (Scale Invariant Feature Transform)
- Explicitly handles occlusions and moving scene parts ⇒ keyframe updating
- Variation of RANSAC called PARSAC (prior-based sampling + enforcing even distribution of inliers, not just high inlier ratio)
Tan et al., ISMAR 2013

Robust Monocular SLAM in Dynamic Environments

Submission # 293
ISMAR 2013
Thank you!

Next lecture: dense reconstruction