AIMS Computer Vision

Lecture 4.1: Reconstruction
MT 2015
Andrea Vedaldi

For slides and up-to-date information:
http://www.robots.ox.ac.uk/~vedaldi/teach.html

Outline

Introduction

Computing $H$ or $F$ from point matches

Feature detection and matching

RANSAC

Determining the ego-motion from $F$

Structure and motion from more than two views

AIMS Computer Vision

1. Matching, indexing, and search
2. Object category detection
3. Visual geometry 1/2: Camera models and triangulation
4. Visual geometry 2/2: Reconstruction from multiple views
5. Segmentation, tracking, and depth sensors
Introduction

In the previous lectures we have seen stereo reconstruction from two views:

1. Obtain (somehow) the camera parameters $P = K[I0]$ and $P' = K'R[t]$.
2. Compute the fundamental matrix $F = K'^{-T}[t]_x R K^{-1}$
3. Match points $x$ in an image to corresponding points $x'$ in the second along the epipolar lines $l' = Fx$.
4. Triangulation: compute the 3D points $X$ from $x, x', P, P'$.

**Next – What happens if:**

1. you do not know how the camera parameters?
2. you have more than two images?

---

The Structure from Motion (SFM) problem

Given two or more images of a scene:

[Camera C]     [Camera C']

compute (i) the camera motion and (ii) the scene structure.

Assumptions:

- **Known** intrinsic calibration $K, K'$.
- **Unknown** extrinsic calibration $R, t$ (egomotion).

---

You get Structure from Motion

[Carl Olsson]

---

Prototypical SFM pipeline

1. Match corner points to find point correspondence. This is harder than before as the epipolar geometry is unavailable.
2. Compute the egomotion $R, t$:
   - For planar scenes:
     - Compute the homography matrix $H$ (e.g. four points algorithm seen in B14);
     - Extract the egomotion from $H$.
   - For general 3D scenes:
     - Compute the fundamental matrix $F$ (e.g. eight points algorithm);
     - Extract the egomotion from $F$.
3. Triangulate as before to obtain the 3D points.
### Outline

Introduction

Computing $H$ or $F$ from point matches

Feature detection and matching

RANSAC

Determining the ego-motion from $F$

Structure and motion from more than two views

---

### Corner Points computed for each frame

Extract some **corner points**, for example using the Harris detector:

- Given point correspondences $x_i \leftrightarrow x'_i$ for $i = 1 \ldots n$, we want to determine $R$ and $t$.
- Intuition: Keep $C$ still, and move $C'$ until all rays intersect.
- Obviously three correspondences are not enough to fix $C'$. How many do we need?

### Egomotion from corner points

Egomotion $= \text{transformation between the cameras.}$
Outline of egomotion computation

1. Compute the fundamental matrix $F$ from the correspondences $x_i \leftrightarrow x'_i$.
2. Decompose $F = K'^{-T} [t] \times RK^{-1}$ to find $R, t$ (given the known $K$ and $K'$).
3. Compute the projection matrices $P$ and $P'$ if needed.

How many correspondences are required?

- Because of the depth/speed scaling ambiguity the rotation (3 DoF) can be determined completely but only the translation direction (2 DoF) is recoverable.
- This allows us to evaluate the number of correspondence needed:
  1. For $n$ scene points there are $3n$ unknowns
  2. Between 2 views there are $5 = (3 \text{ rot} + 2 \text{ trans})$ unknowns
  3. Each correspondence yields 4 measurements
  4. Hence $4n \geq 3n + 5$ and
     \[ n \geq 5 \text{ correspondences are needed} \]
- For $n < 7$ the solutions are non-linear, so we'll see solutions for $n = 7$ and $n = 8$.

Actually, $t$ found only up to scale

- $F$ is a homogeneous matrix, so
  \[ F \propto K'^{-1} [t] \times RK^{-1} \\]
- Therefore translation and all lengths are recovered only up to scale:
  \[ t \equiv \lambda t, \quad X_i \equiv \lambda X_i. \]

Depth/scale ambiguity

We cannot distinguish:

- a large translation when viewing a large distant scene; from
- a small translation when viewing a small near-to scene.

Question: How might you resolve the depth/scale scaling ambiguity?

Computing the fundamental matrix for $n \geq 8$

- Task: Given $n$ correspondences $x_i \leftrightarrow x'_i$ compute $F$ such that
  \[ \forall i : x'_i ^T F x_i = 0. \]
- Solution: Each correspondence generates one constraint
  \[
  \begin{bmatrix}
  x'_1 & y'_1 & 1 \\
  x'_2 & y'_2 & 1 \\
  x'_3 & y'_3 & 1 \\
  \vdots & \vdots & \vdots \\
  x'_n & y'_n & 1
  \end{bmatrix}
  \begin{bmatrix}
  f_1 \\
  f_2 \\
  f_3 \\
  f_4 \\
  f_5 \\
  f_6 \\
  f_7 \\
  f_8 \\
  f_9
  \end{bmatrix}
  \begin{bmatrix}
  x_1 \\
  y_1 \\
  \vdots \\
  x_n \\
  y_n
  \end{bmatrix}
  = 0. \]

which can be written as

\[
 x'_1 x_1 f_1 + x'_1 y_1 f_2 + x'_2 f_3 + y'_1 x_1 f_4 + y'_1 y_1 f_5 + y'_2 f_6 + x_1 f_7 + y_1 f_8 + f_9 = 0
\]

or

\[
 \begin{bmatrix}
  x'_1 & x'_2 & x'_3 & \cdots & x'_n & y'_1 & y'_2 & y'_3 & \cdots & y'_n & x_1 & y_1 & \cdots & x_n & y_n
  \end{bmatrix}
  \begin{bmatrix}
  f_1 \\
  \vdots \\
  f_9
  \end{bmatrix}
  = 0. \]
Computing the fundamental matrix /ctd

- For \( n \) correspondences build up the \( n \times 9 \) system
  
  \[
  A_{n \times 9} \mathbf{f} = \begin{bmatrix}
  x_1' x_1 & x_1' y_1 & \cdots & x_1' y_1 & y_1' x_1 & y_1' y_1 & \cdots & y_1' y_1 & 1 \\
  \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
  x_n' x_n & x_n' y_n & \cdots & x_n' y_n & y_n' x_n & y_n' y_n & \cdots & y_n' y_n & 1
  \end{bmatrix} \begin{bmatrix}
  f_1 \\
  \vdots \\
  f_9
  \end{bmatrix}.
  
- For \( n = 8 \) points \( \mathbf{f} \) can be found as the null-space of \( A \), and so \( \mathbf{f} \) and \( \mathbf{F} \) are determined up to scale (as expected).
- Since the points are noisy, in general one wants to use \( n > 8 \). This can be done using least square.

Proof of the eigendecomposition solution

- The squared sum of the residuals \( r = Af \) is
  
  \[
  \|r\|^2 = r^\top r = f^\top A^\top A f = f^\top M f
  \]

- \( M = A^\top A \) is a \( n \times n \) symmetric real matrix; hence it can be decomposed as
  
  \[
  M = \Sigma V^\top \Sigma = \begin{bmatrix}
  \lambda_1 & & \\
  & \lambda_2 & \\
  & & \ddots \\
  & & & \lambda_n
  \end{bmatrix} V^\top = \sum_{i=1}^{n} \lambda_i [\hat{e}_i \hat{e}_i^\top]
  \]

  where

  - \( V = [\hat{e}_1 \ldots \hat{e}_n] \) is the orthonormal matrix of eigenvectors
  - eigenvalues are non-decreasing: \( 0 \leq \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n \).
  - The eigenvalues are non-negative because:
    
    \[
    \hat{M} \hat{e}_i = \hat{e}_i \lambda_i \Rightarrow \hat{e}_i^\top \hat{M} \hat{e}_i = \hat{e}_i^\top \hat{e}_i \lambda_i = \lambda_i \geq 0.
    
    Then
    
    \[
    f^\top M f = \lambda_1 (f^\top \hat{e}_1)^2 + \lambda_2 (f^\top \hat{e}_2)^2 + \ldots + \lambda_n (f^\top \hat{e}_n)^2.
    
  - This expression is minimised when \( f = \hat{e}_1 \).

Proof of the SVD solution

- Any \( m \times n \) matrix \( \mathbf{A} \) where \( m \geq n \) can be decomposed as
  
  \[
  \mathbf{A}_{m \times n} = \mathbf{U}_{m \times n} \begin{bmatrix}
  \sigma_1 & & \\
  & \sigma_2 & \\
  & & \ddots \\
  & & & \sigma_n
  \end{bmatrix} \mathbf{V}_{n \times n}^\top
  \]

  where \( \mathbf{U} \) is column-orthogonal, \( \mathbf{V} \) is fully orthogonal, and \( \mathbf{\Sigma} \) contains the singular values ordered so \( 0 \leq \sigma_1 \leq \sigma_2 \leq \ldots \leq \sigma_n \).

  - The singular vectors \( \mathbf{V} \) of \( \mathbf{A} \) are the same as the eigenvectors of \( \mathbf{M} = \mathbf{A}^\top \mathbf{A} \):
    
    \[
    \mathbf{M} = \mathbf{A}^\top \mathbf{A} = \mathbf{V} \Sigma^\top \mathbf{U} \Sigma \mathbf{V}^\top = \mathbf{V} \Sigma^2 \mathbf{V}^\top
    
    \]

  - In particular \( f = \hat{e}_1 \) is the first column of \( \mathbf{V} \).
  - The SVD is usually preferred to the eigenvalue decomposition because it is numerically more stable.
Computing $F$ from 7 points

- For the $7 \times 9$ set of equations $Af = 0$ we know that $f$ is in the null space of $A$.
- This null space is 2-dimensional and hence spanned by two vectors $f_1$ and $f_2$. Since $f$ is determined up to scale, all solutions are given by:
  $$f = \alpha f_1 + (1 - \alpha)f_2$$
- Reshaping the vectors, results in a family of candidate fundamental matrices
  $$F = \alpha F_1 + (1 - \alpha)F_2$$
- To find which one is a “proper” fundamental matrix, use the non-linear constraint $\det F = 0$. This gives a cubic equation in $\alpha$. Can you see why?
- The cubic has either one or three real solutions for $\alpha$.

A Visual Compass

If the motion of the camera is known to be a pure rotation, then the images are related by an homography

$$x' = H_\infty x$$

where

$$H_\infty = K'RK^{-1}.$$  

Algorithm:

- Find correspondences $x_i \leftrightarrow x'_i$
- Compute $H_\infty$ from the correspondences (see B14)
- Extract $R$ to find relative rotations

But of course we cannot recover any scene structure!

Outline

Introduction

Computing $H$ or $F$ from point matches

Feature detection and matching

RANSAC

Determining the ego-motion from $F$

Structure and motion from more than two views
Feature detection, matching and the $F$ matrix

So far, we have not discussed matching. The reason is that computation of the fundamental matrix can be incorporated into the matching.

Outline:
- Extract image points as corners. Why corners?
- Obtain an initial corner matches using local descriptors.
- Remove outlier and estimate the fundamental matrix $F$ using RANSAC.
- Obtain further corner matches using $F$.

Corner Points computed for each frame

Recall that points with distinctively high autocorrelation provide the best chance of deriving a distinctive cross-correlation signal.

Why not use lines, or take a dense pixel-based approach?
- The key reason is that the search for matches is no longer 1D when the camera motion is unknown. A 2D region has to be searched.
- A dense approach is then likely to be too expensive, and matching sections of a line suffers the aperture problem.
- Corners are:
  - relatively sparse;
  - reasonably cheap to compute;
  - well-localized;
  - appear quite robustly from frame to frame.
- Hence corners are good for matching.

Initial matching

- Extract corners in both images (feature detection).
- For each corner $x$ in $C$, make a list of potential matches $x'$ in a region in $C'$ around $x$ (heuristic).
- Rank the matches by comparing the regions around the corners using cross-correlation.
- Sift them to reconcile forward-backward inconsistencies.
- The idea here is to not to do too much work — just enough to get some good matches.
Matches — some good matches, some mismatches. Can still compute $F$ with around 50% mismatches. How?

**Outline**

- Introduction
- Computing $H$ or $F$ from point matches
- Feature detection and matching
- RANSAC
- Determining the ego-motion from $F$
- Structure and motion from more than two views

**RANSAC – RANdom SAmple Concensus**

- Suppose you tried to fit a straight line to data containing outliers — points which are not properly described by the assumed probability distribution.
- The usual methods of least squares are hopelessly corrupted.
- Need to detect outliers and exclude them.

- Use estimation based on robust statistics. RANSAC was the first, devised by vision researchers, Fischler & Bolles (1981).
RANSAC algorithm for lines

1. For many repeated trials:
   1.1 Select a random sample of two points
   1.2 Fit a line through them
   1.3 Count how many other points are within a threshold distance of the line \textit{(inliers)}
2. Select the line with the largest number of inliers
3. Refine the line by fitting it to all the inliers (using least squares)

Remarks:
- Sample a minimal set of points for your problem (2 for lines).
- Repeat such that there is a high chance that at least one minimal set contains only inliers (see tutorial sheet).

---

RANSAC algorithm for F

1. For many repeated trials:
   1.1 Select a random sample of seven correspondences
   1.2 Compute $F$ using the cubic method
   1.3 Count how many other correspondences are within threshold distance of the epipolar lines \textit{(inliers)}
2. Select the $F$ with the largest number of inliers
3. Refine $F$ by fitting it to all the inliers (using the SVD method)
RANSAC algorithm for $H$

1. For many repeated trials:
   1.1 Select a random sample of four correspondences
   1.2 Compute $H$ (as in B14)
   1.3 Count how many other correspondences are within threshold distance of the predicted locations (inliers)
2. Select the $H$ with the largest number of inliers
3. Refine $H$ by fitting it to all the inliers, optimizing the reprojection error

$$\min_H \sum_{(x,x') \in \text{Inliers}} d^2(x', H x) + d^2(H^{-1}x', x)$$

Epipolar geometry

Correspondences consistent with epipolar geometry

Outline

Introduction

Computing $H$ or $F$ from point matches

Feature detection and matching

RANSAC

Determining the ego-motion from $F$

Structure and motion from more than two views
Computing R and t from F

Recall that \( F = K'^{-1} [t] \times RK^{-1} \). We now show how to recover \( R \) and \( t \) from \( F \) (given \( K \) and \( K' \)).

1. Compute the essential matrix \( E = [t] \times R = K' TFK \).
2. Compute \( t \) as the null vector of \( E^T \) (i.e. \( E^T t = 0 \)).
   - \( t \) is determined up to a scaling factor \( \mu \)
   - there are two solutions \( \pm \mu t \)
3. Compute \( R \) from \( E \)
   - the algorithm for this step is given later.
   - it returns two solutions \( R_1 \) and \( R_2 \).
4. Overall, there are four solutions for the projection matrix:
   - \( P' = K'[R_1 | \mu t] \)
   - \( P' = K'[R_1 | - \mu t] \)
   - \( P' = K'[R_2 | \mu t] \)
   - \( P' = K'[R_2 | - \mu t] \)
5. Exclude 3 of these using a visibility test

Computing \( R_{1,2} \) from the essential matrix \( E \)

Recall that \( E = [t] \times R \); we now recover \( R \) from \( E \). Algorithm:

1. Compute the Singular Value Decomposition (SVD) of \( E \).
   \[
   \begin{bmatrix}
   1 & 0 & 0 \\
   0 & 1 & 0 \\
   0 & 0 & 0
   \end{bmatrix}
   \begin{bmatrix}
   U \\
   V^T \leftarrow M
   \end{bmatrix}
   \]
2. Set
   \[
   W = \begin{bmatrix}
   0 & -1 & 0 \\
   1 & 0 & 0 \\
   0 & 0 & 1
   \end{bmatrix}
   \]
3. The two solutions are:
   \[
   R_1 = UWV^T, \quad R_2 = UW^TV^T
   \]

The four solutions

The 3D point is in front of both cameras in only one case.

Note these are “computer vision” cameras, so to be visible a ray must pass through the image on its way to the optic centre!

Outline

Introduction

Computing H or F from point matches

Feature detection and matching

RANSAC

Determining the ego-motion from F

Structure and motion from more than two views
Why bother?
1. Matching becomes more verifiable, as 3D point estimates are available to re-project.
2. 3D point estimates improve as further views over a range of angles is obtained.
3. There is no increase in the degree of ambiguity, though the overall scale ambiguity persists.

Notation for three+ views

For three views let the cameras be $C$, $C'$, $C''$ with projection matrices $P$, $P'$ and $P''$, and with image points $x$, $x'$ and $x''$.

For $m$ views, a point $x_j$ is imaged in the $i$-th camera $C_i$ at $x_{ij} = P_i X_j$.

Point correspondence over 3 views

Given the projection matrices and $x \leftrightarrow x'$ how is the point $x''$ found?

Algorithm:
1. Compute the 3D point from $x$ and $x'$
2. Then re-project using $P''$.
3. The search in the third image is zero-D, and the size of the search region depends only on uncertainty.

Problem statement: structure and motion

Given: $n$ matching image points $x_j$ over $m$ views

Find: the cameras $P_i$ and the 3D points $X_j$ such that $x_j \approx P_i X_j$ by finding:

$$
\min_{P_i, X_j} \sum_{j=1}^{n} \sum_{i=1}^{m} d^2(x_j, P_i X_j)
$$

This is a serious minimization:
1. For each camera, 6 parameters
2. For each 3D point, 3 parameters
3. Total of $6m + 3n - 1$ ($-1$ for scale) parameters overall

For 50 frames, 1000 points, we have $3.3 \times 10^3$ unknowns!
Building block is computing correspondences \( x_j^i \leftrightarrow x_j^{i+1} \), finding \( F_i^{i+1} \) and then matrices \( P_i, P_i^{i+1} \).

**Algorithm**

1. Compute interest points in each image
2. Compute matches between consecutive image pairs \( i, i+1 \)
3. Compute \( F_i^{i+1} \). Recover \( P_i, P_i^{i+1} \)
4. Compute scene points
5. Extend correspondences over image triples
6. Extend correspondences over all images
7. Optimize over all \( P_i, X_j \)

**2d3’s Boujou system**
Zisserman, Fitzgibbon, Torr, Beardsley

**Batch SFM**

- Up to now **batch**, offline processing of video sequences
- Post-production, 3D model reconstruction, etc.
Real-time, sequential SFM

- **Real-time, sequential**, fixed time budget (10s of milliseconds)
- Build and maintain a map, and localise w.r.t. the map

- Real-time robotics applications, but in simplified 2D environments, specialised sensors, etc
- Reliable, repeated measurement is crucial – mitigates against drift giving repeatable accuracy.

Sequential structure from motion: visual SLAM

- Represent joint distribution over camera and feature positions using a single **multi-variate Gaussian**.

\[
x = \begin{bmatrix} x_1 \\ y_1 \\ y_2 \\ \vdots \\ \vdots \\ \vdots \\
\end{bmatrix}, \quad 
P = \begin{bmatrix} P_{xx} & P_{xy} & P_{x1y1} & P_{xy2} & \cdots \\
P_{yx1} & P_{yy} & P_{y1y1} & P_{y1y2} & \cdots \\
P_{y2x} & P_{y2y1} & P_{y2y2} & \cdots & \cdots \\
\vdots & \vdots & \vdots & \ddots & \ddots \\
\vdots & \vdots & \vdots & \ddots & \ddots \\
\vdots & \vdots & \vdots & \ddots & \ddots \\
\end{bmatrix}
\]

- Use Kalman Filter (see C4B Mobile Robotics)
  
  **predict → measure → update**

  framework to propagate uncertainty, and fuse measurement data

Example: real-time, sequential structure from motion

Davison, Reid, Smith, Williams, Klein, *et al.*