Computer Vision I - Algorithms and Applications: *Multi-View 3D reconstruction*

Carsten Rother

09/12/2013
Roadmap this lecture

• Two-view reconstruction
• From projective to metric space (e.g. self-calibration)
• Multi-view reconstruction
  • Iterative projective cameras
  • Closed form: affine cameras
  • Closed form: reference plane

Next lecture:
• dense labeling problems in computer vision:
  • stereo matching
  • ICP, KinectFusion
3D reconstruction - Definitions

• Sparse Structure from Motion (SfM)

• SLAM in robotics: Simultaneous Localization and Mapping:
  “Place a robot in an unknown location and in an unknown environment and have the robot incrementally build a map of this environment while simultaneously using this map to compute its location”

• Dense Multi-view reconstruction
Roadmap this lecture

• Two-view reconstruction
• From projective to metric space (e.g. self-calibration)
• Multi-view reconstruction
  • Iterative projective cameras
  • Closed form: affine cameras
  • Closed form: reference plane

Next lecture:
• dense labeling problems in computer vision:
  • stereo matching
  • ICP, KinectFusion
First: Two-View reconstruction
Epipolar Geometry (Reminder)

Fundamental matrix $F$: $p_2^T F p_1 = 0$
2-view reconstruction

• We have $x'^T F x = 0$

• Can we get $P, P'$ such that:

  $$x = PX ; x' = P'X$$

  and $x'^T F x = 0$ for all 3D points $X$

• Derivation (blackboard) see HZ page 256

  $$P = [I_{3 \times 3} \mid 0 ]; \quad P' = [ [e'] \times F \mid e' ]$$

  $$3 \times 4 \quad 3 \times 4$$
We need \( P, P' \) such that:
\[
x = PX, \quad x' = P'X \quad \text{for all } X
\]
\[
\exists x \quad x^T P^T + P^T x = 0 \quad (A)
\]

Choose \( P = \begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix}, \quad P' = \begin{bmatrix} S & I \\ 0 & 0 \end{bmatrix} \)

where \( S = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \)

we show that (A) holds for any \( X \)

\[
P^T + P = \begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} T^T S T & 0 \\ 0 & 0 \end{bmatrix}
\]

\[
= \begin{bmatrix} T^T S T & 0 \\ 0 & 0 \end{bmatrix}
\]

\( \exists \) we have to show:

\[
\begin{bmatrix} x, y, z, 1 \end{bmatrix}^T \begin{bmatrix} T^T S T & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x, y, z \end{bmatrix} = 0 \quad \text{and } (x, y, z) T^T S T (z) = 0
\]

This is true if \( T^T S T = [m] \),

\[
\begin{bmatrix} a & b & c \\ c & 0 & -a \\ -b & a & 0 \end{bmatrix}
\]
Derivation

\[
\begin{bmatrix}
a & b & c \\
d & e & f \\
g & h & i
\end{bmatrix}
\begin{bmatrix}
c' \ b + b' \ c & -c' \ e + b' \ f & -c' \ h + b' \ i \\
-\ b' \ a + a' \ b & -b' \ d + a' \ e & -b' \ g + a' \ h \\
-\ b' \ a + a' \ b & -b' \ d + a' \ e & -b' \ g + a' \ h
\end{bmatrix}
= \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \end{bmatrix}
\]

\[a' \ c' + a' \ b' \ c + b' \ c' \ a - b' \ a' \ c - c' \ b' \ a + c' \ a' \ b = 0
\]

\[a' \ c' + a' \ b' \ c + b' \ c' \ a - b' \ a' \ c - c' \ b' \ a + c' \ a' \ b = 0
\]

\[
\begin{align*}
1. & -ac' + ab' + bc' + ca' = 0 \\
2. & -ac' + ab' + bc' + ca' = m_1 \\
3. & -ac' + ab' + bc' + ca' = m_2 \\
4. & -ac' + ab' + bc' + ca' = m_3 \\
5. & -ac' + ab' + bc' + ca' = m_4 \\
6. & -ac' + ab' + bc' + ca' = m_5 \\
7. & -ac' + ab' + bc' + ca' = m_6 \\
8. & -ac' + ab' + bc' + ca' = m_7 \\
9. & -ac' + ab' + bc' + ca' = m_8 \\
10. & -ac' + ab' + bc' + ca' = m_9
\end{align*}
\]
Triangulation - algebraic

- Input: $x, x', P, P'$
- Output: $X$’s
- Triangulation is also called intersection
- Simple algebraic solution:

\[ \lambda x = P X \quad \text{and} \quad \lambda' x' = P' X \]

2) Eliminate $\lambda, \lambda'$ by taking ratios. This gives 4 linear independent equations for 3 unknowns: $X = (X_1, X_2, X_3, X_4)$ where $\|X\| = 1$.

An example ratio is:

\[ \frac{x_1}{x_2} = \frac{p_1 X_1 + p_2 X_2 + p_3 X_3 + p_4 X_4}{p_5 X_1 + p_6 X_2 + p_7 X_3 + p_8 X_4} \]

3) This gives (as usual) a least square optimization problem:

\[ A X = 0 \quad \text{with} \quad \|X\| = 1 \quad \text{where} \quad A \text{ is of size } 4 \times 4. \]

This can be solved in closed-form using SVD.
Minimize re-projection error with Fixed fundamental matrix.

\[ \{ \hat{x}, \hat{x}' \} = \underset{\hat{x}, \hat{x}'}{\text{argmin}} \ d(x, \hat{x})^2 + d(x', \hat{x}')^2 \quad \text{subject to} \quad \hat{x}' F \hat{x} = 0 \]
Minimize re-projection error with fixed fundamental matrix.

\[
\{\hat{x}, \hat{x}'\} = \underset{\{\hat{x}, \hat{x}'\}}{\text{argmin}} \ d(x, \hat{x})^2 + d(x', \hat{x}')^2 \text{ subject to } \hat{x}' F \hat{x} = 0
\]

- Solution can be expressed as a 6-degree polynomial in \( t \)
- This has up to 6 solutions and can computed (roots of polynomial)
Triangulation - uncertainty

- Large baseline
  - Smaller area

- Smaller baseline
  - Larger area

- Very small baseline
  - Very larger area
Roadmap this lecture

• Two-view reconstruction
• From projective to metric space (e.g. self-calibration)
• Multi-view reconstruction
  • Iterative projective cameras
  • Closed form: affine cameras
  • Closed form: reference plane

Next lecture:
• dense labeling problems in computer vision:
  • stereo matching
  • ICP, KinectFusion
3D reconstruction

• Given: m cameras and n static 3D points
• Formally: \( x_{ij} = P_j X_i \) for \( j = 1 \ldots m; i = 1 \ldots n \)
• Important in practice we do not have all points visible in all views, i.e. Number \( x_{ij} \leq mn \)
• Goal: find all \( P_j \)’s and \( X_i \)’s

Example: “Visibility” matrix
Iterative multi-view reconstruction – Method 1

Three views of an un-calibrated or calibrated camera
Iterative multi-view reconstruction – Method 1

Three views of an un-calibrated or calibrated camera

Similiar problem as intersection (and same as camera calibration):
\[ \lambda x = P X \] and \[ \lambda' x' = P X' \]
with 6 points you get 12 constraints and we have 11 unknowns.
Iterative multi-view reconstruction – Method 1

Three views of an un-calibrated or calibrated camera

You can iterate between: intersection and re-sectioning to get all points and cameras reconstructed (in projective or metric space)

Similiar problem as intersection (and same as camera calibration): $\lambda x = P X$ and $\lambda' x' = P X'$

with 6 points you get 12 constraints and we have 11 unknowns.
Iterative multi-view reconstruction – Method 2

Three views of an un-calibrated or calibrated camera

Reconstruct Points and Camera 1 and 2
Iterative multi-view reconstruction – Method 2

Three views of an un-calibrated or calibrated camera

Reconstruct Points and Camera 2 and 3

Zipp the 2 reconstructions together.
1. They share a camera and 7+ points (needed for $F$-matrix).

2. Get $H$ ($4 \times 4$) from
   \[ X_{1-7} = HX'_{1-7} \]
   (here points $X'_{1-7}$ are in second reconstruction and $X_{1-7}$ in first reconstruction)

3. Zipp them together:
   \[ P_3 = P'_3 H^{-1}; X_i = HX'_i \]
   (Here $P'_3$ is camera in second reconstruction)

Note:
Objective

Given a sequence of frames in a video, compute correspondences and a reconstruction of the scene structure and the camera for each frame.

Algorithm

(i) Interest points: Compute interest points in each image.
(ii) 2 view correspondences: Compute interest point correspondences and $F$ between consecutive frames using algorithm 11.4(p291) (frames may be omitted if the baseline motion is too small).
(iii) 3 view correspondences: Compute interest point correspondences and $T$ between all consecutive image triplets using algorithm 16.4(p401).
(iv) Initial reconstruction: See text.
(v) Bundle adjust the cameras and 3D structure for the complete sequence.
(vi) Auto-calibration: see chapter 19 (optional).

Algorithm 18.3. Overview of reconstruction from a sequence of images.

Comment: Between 3 and 4 views there exists Trifocal and Quadrifocal tensor (as Fundamental matrix for 2 views) – not discussed in this lecture

[See page 453 HZ]
Bundle adjustment

• Global refinement of jointly structure (points) and cameras

• Minimize geometric error:

\[
\arg\min_{\{P'_s, X'_s\}} \sum_j \sum_i \alpha_{ij} d(P_j X_i, x_{ij})
\]

here \(\alpha_{ij}\) is 1 if \(X_j\) visible in view \(P_j\) (otherwise 0)

• Non-linear optimization with e.g. Levenberg-Marquard
Example
Main Problem of iterative methods is Drift

Solution: 1) look for “Loop closure” if possible
2) global methods (next)
Roadmap this lecture

- Two-view reconstruction
- From projective to metric space (e.g. self-calibration)
- Multi-view reconstruction
  - Iterative projective cameras
  - Closed form: affine cameras
  - Closed form: reference plane

Next lecture:
- dense labeling problems in computer vision:
  - stereo matching
  - ICP, KinectFusion
Reminder: affine cameras (from lecture 4)

• Affine camera has 8 DoF:

\[
\begin{bmatrix}
  x \\
  y \\
  1
\end{bmatrix} =
\begin{bmatrix}
  a & b & c & d \\
  e & f & g & h \\
  0 & 0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
  X \\
  Y \\
  Z \\
  1
\end{bmatrix}
\]

• In short: \( \tilde{x} = M \tilde{X} + t \)

\[
\begin{array}{ccc}
2 \times 1 & 2 \times 3 & 2 \times 1
\end{array}
\]
Reminder: affine cameras (from lecture 4)

Parallel 3D lines map to parallel 2D lines (since points stay at infinity)

“Close to parallel projection”
Multi-View Reconstruction for affine cameras

(derivation on blackboard)

$$\hat{x}_{ij} = M_j \hat{x}_i + t_j$$

$$\min_{M_j, t_j} \sum_{i,j} \| \hat{x}_{ij} - M_j \hat{x}_i - t_j \|_2^2$$

assumes all points visible in all views.

Get all $t_j$'s in closed form:

Consider 1D case and one camera $M, t$.

$$\frac{\partial}{\partial t} \sum_{i} (\hat{x}_i - M \hat{x}_i - t) (x_i - M x_i - t) = 0$$

$$\Rightarrow \frac{\partial}{\partial t} \sum_{i} (\text{const} - 2x_i t + 2M x_i t + t^2) = 0$$

$$\Rightarrow \sum_{i} (-2x_i + 2M x_i + 2t) = 0$$

$$\Rightarrow -2 \sum_{i} x_i + 2M \sum_{i} x_i + 2nt = 0$$

$$\Rightarrow t = \frac{1}{n} \sum_{i} x_i - \frac{1}{n} M \sum_{i} x_i$$

we choose centroid $\sum_{i} x_i = 0$

$$\Rightarrow t = \frac{1}{n} \sum_{i} x_i$$
Multi-View Reconstruction for affine cameras

(derivation on blackboard)

\[ \min_{M, \hat{X}} \sum_{i,j} \| \hat{x}_{ij} - M \hat{x}_i \|^2 \]

\[ = \min_{M, \hat{X}} \| (X_{\hat{X}}) - (M_{\hat{X}}) (X_{\hat{X}}) \|^2 \]

Optimum from SVD of W

\[ W = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{pmatrix} \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{pmatrix} \begin{pmatrix} v_1^T \\ v_2^T \\ \vdots \\ v_m^T \end{pmatrix} \begin{pmatrix} x \end{pmatrix} \]

\[ W = MX \]

\[ W - MX = 0 \]

Also optimal under Frobenius norm if measurements are noisy.
Comments / Extensions

• Main restriction is that all points have to be visible in all views. (can be used for a subset of views and then zipping subviews together)

• Extensions to missing data have been done (see HZ chapter 18)

• Extensions to projective cameras have been done (ch. 18.4)

• Extensions to non-rigidly moving scenes (ch. 18.3)
Roadmap this lecture

• Two-view reconstruction
• From projective to metric space (e.g. self-calibration)

• Multi-view reconstruction
  • Iterative projective cameras
  • Closed form: affine cameras
  • Closed form: reference plane

Next lecture:
• dense labeling problems in computer vision:
  • stereo matching
  • ICP, KinectFusion
• $H_\infty = KR$ is called infinity homography since it is the mapping from plane at infinity to image:
\[
x = KR \begin{pmatrix} I \vert - \tilde{C} \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 0 \end{pmatrix} = KR \begin{pmatrix} x \\ y \\ z \end{pmatrix} = H_\infty \begin{pmatrix} x \\ y \\ z \end{pmatrix}
\]

• Idea: simply define an arbitrary plane as $\pi_\infty = (0,0,0,1)^T$
(this can be done in projective space)

[Rother PhD Thesis 2003]
Direct reference plane approach (DRP)

Derivation on blackboard

1) Compute $H_0$

4 image points $x_1$-$x_4$ \[ \rightarrow \] (1), (2), (3), (4)
8 eqn. and 8 unknowns

2) It is for arbitrary point $\tilde{x}$ and camera $\tilde{C}$:

\[ \tilde{x} = H_0 (I - \tilde{C})(\tilde{x}) \]
\[ \Rightarrow \tilde{H}_0 \tilde{x} = \tilde{x} - \tilde{C} \]

\[ \text{Take ratios:} \]

\[ \frac{\tilde{x}_1'}{\tilde{x}_2'} = \frac{x_1' - \tilde{C}_1}{x_2' - \tilde{C}_1} \Rightarrow x_1'x_2' - x_1'\tilde{C}_1 - x_2'\tilde{C}_1 + \tilde{C}_1 = 0 \]

\[ \frac{\tilde{x}_3'}{\tilde{x}_4'} = \frac{x_3' - \tilde{C}_3}{x_4' - \tilde{C}_3} \Rightarrow x_3'x_4' - x_3'\tilde{C}_3 - x_4'\tilde{C}_3 + \tilde{C}_3 = 0 \]

3) For every 3D point $x_3'$ visible in camera $P_3$ we get a constraint. Big linear system:

\[ \begin{pmatrix} x_1' & x_2' & x_3' & x_4' \\ x_1' & x_2' & x_3' & x_4' \\ \vdots & \vdots & \vdots & \vdots \end{pmatrix} \begin{pmatrix} \tilde{C}_1 \\ \tilde{C}_2 \\ \tilde{C}_3 \\ \tilde{C}_4 \end{pmatrix} = 0 \]

4) Solved with SVD gives 4D Null-space

[Rother PhD Thesis 2003]
Result
How to get infinite Homographies

• Real Plane in the scene:

• Fixed / known $K$ and $R$, e.g. translating camera with fixed camera intrinsics

• Orthogonal scene directions and a square pixel camera see above we get out: $K, R$ (up to a small ambiguity)
Result: University Stockholm

(Show video)
3 Minutes break
Roadmap this lecture

• Two-view reconstruction
• From projective to metric space (e.g. self-calibration)
• Multi-view reconstruction
  • Iterative projective cameras
  • Closed form: affine cameras
  • Closed form: reference plane

Next lecture:
• dense labeling problems in computer vision:
  • stereo matching
  • ICP, KinectFusion
Scale ambiguity

Is the pumpkin 5 meters or 30 cm high?
Structure from Motion Ambiguity

• We can always write:

\[ x = P \frac{k}{k} X = \left( \frac{1}{k} P \right) (kX) \]

• It is impossible to recover the absolute scale of the scene!
Projective ambiguity

We can write (most general): \[ x_{ij} = P_j X_i = P_j Q^{-1} Q X_i = P'_j X'_i \]

\[ Q = \begin{bmatrix} A & t \\ \vdots & \vdots \\ v^\top & v \end{bmatrix} \]

- Q has 15 DoF (projective ambiguity)
- If we do not have any additional information about the cameras or points then we cannot recover Q.
- Possible information (we will see details later)
  - known calibration matrix
  - calibration matrix is same for all cameras
  - external constraints: given 3D points
Projective ambiguity

This is a “protectively” correct reconstruction
... but not a nice looking one
Affine ambiguity

We can write (most general): \[ x_{ij} = P_jX_i = P_jQ^{-1}QX_i = P_j'X_i' \]

- \( Q \) has now 12 DoF (affine ambiguity)
- With affine cameras we get only affine ambiguity
- \( Q \) leaves the plane at infinity \( \pi_\infty = (0,0,0,1)^T \) in place: any point on \( \pi_\infty \) moves like: \( Q(a, b, c, 0)^T = (a', b', c', 0) \)
  Therefore parallel 3D lines stay parallel for any \( Q \)
From Projective to Affine

Step:
Take points $X_{1-3} = (x_{1-3}, y_{1-3}, z_{1-3}, 1)^T$ and move to a point at infinity:
$(x'_{1-3}, y'_{1-3}, z'_{1-3}, 0)^T$.
Same as having plane at infinity in its canonical position $\pi_\infty = (0,0,0,1)^T$.

The red directions point to places where the projection of a vanishing point lies.
Affine ambiguity

3D Points at infinity stay at infinity
We can write (most general):

\[ x_{ij} = P_jX_i = P_jQ^{-1}QX_i = P'_jX'_i \]

- \( Q \) has now 7 DoF (similarity ambiguity)
- \( Q \) preserves angles, ratios of lengths, etc.
- For visualization purpose this ambiguity is sufficient. We don’t need to know what 1m, 1cm, 1mm, etc. means
- Note, if we don’t care about the choice of \( Q \) we can set for instance the camera center of first camera to 0.
Essentially, we need the true camera calibration matrices $K$ (see details later)

**Steps:** (see HZ page 277)
One option: with known $K$ operate with essential matrix and get out $R, C$ of all cameras
Similarity Ambiguity

Computer Vision I: Multi-View 3D reconstruction
How to “upgrade” a reconstruction

Illustrating some ways to upgrade from Projective to Affine and Metric (see details in HZ page 270ff and chapter 19)

- Camera is calibrated
- Calibration from external constraints
- Calibration from a mix of in- and external constraints
- Calibration from internal constraints, called self/auto calibration

\[
x_{ij} = P_jX_i = P_jQ^{-1}QX_i = P'_jX'_i
\]

- Find plane at infinity and move in canonical position:
  - One of the cameras is affine see HZ page 271
  - 3 non-collinear 3D vanishing points
- Translational motion (HZ page 268)
Projective to Affine – Three 3D vanishing points

**Goal:** Find plane at infinity and move to \( \pi_\infty = (0,0,0,1)^T \)

Possible method
1. Identify three pairs of lines in the 3D reconstruction that have to be parallel, respectively.
2. Compute the three 3D intersection points \( X_{1-3} \)
3. Move \( X_{1-3} \) somewhere on the plane at infinity \( \pi_\infty = (0,0,0,1)^T \). Define the equation:
   \[
   X'_{1-3} = Q X_{1-3}
   \]
   where
   \[
   X'_{1-3} = (1,0,0,0)^T, (0,1,0,0)^T, (0,0,1,0)^T
   \]
4. Compute an \( Q \) using SVD
5. Update all points and cameras
   \[
   P_j = Q^{-1} P_j; \quad X_i = Q X_i
   \]
Given: five known 3D points

Compute $Q$:
1) $QX_i = X_i'$ (each 3D point gives 3 linear independent equations)
2) 5 points give 15 equations, enough to compute $Q$ using SVD

Upgrade cameras and points:
$P_j' = P_j Q^{-1}$ and $X_i' = QX_i$
But without external knowledge?

- For a camera \( P = K [I \mid 0] \) the ray outwards is:
  \[ x = P X \text{ hence } X = K^{-1}x \]

- The angle \( \Theta \) is computed as the normalized rays \( d_1, d_2 \):
  \[
  \cos \Theta = \frac{d_1^T d_2}{\sqrt{d_1^T d_1 \sqrt{d_2^T d_2}}} = \frac{(K^{-1}x_1)^T (K^{-1}x_2)}{\sqrt{(K^{-1}x_1)^T (K^{-1}x_1) \sqrt{(K^{-1}x_2)^T (K^{-1}x_2)}}}
  \]
  \[
  = \frac{x_1^T \omega \ x_2}{\sqrt{x_1^T \omega \ x_1 \sqrt{x_2^T \omega \ x_2}}}
  \]

- We define the matrix: \( \omega = K^{-T}K^{-1} \)
- Comment: \( (K^{-1})^T = (K^T)^{-1} =: K^{-T} \)
But without external knowledge?

\[
\cos \Theta = \frac{x_1^T \omega x_2}{\sqrt{x_1^T \omega x_1} \sqrt{x_2^T \omega x_2}}
\]

- We have: \( \cos \Theta = \frac{x_1^T \omega x_2}{\sqrt{x_1^T \omega x_1} \sqrt{x_2^T \omega x_2}} \)

- If we were to know \( \omega \) then we can compute angle \( \Theta \)

- \( K \) can be derived from \( \omega = K^{-T} K^{-1} \) using Cholesky decomposition (see HZ page 582)

- Comment, if \( \Theta = 90^\circ \) then we have \( x_1^T \omega x_2 = 0 \)

- How do we get \( \omega \)?
$\omega$ is the Image of the absolute Conic (IAC)

- $\omega$ is called the image of the absolute conic $\Omega_\infty = I_{3 \times 3}$
on the plane at infinity $\pi_\infty = (0, 0, 0, 1)^T$

- Remember a conic $C$ is defined as: $x^T C x = 0$.
Here it is: $(x, y, 1)\Omega_\infty (x, y, 1)^T = 0$
That is: $x^2 + y^2 = -1$
an imaginary circle with radius $i$
the points $(i, 0)$ and $(0, i)$ lie on the conic.
\( \omega \) is called the image of the absolute conic \( \Omega_{\infty} = I_{3 \times 3} \) on the plane at infinity \( \pi_{\infty} = (0,0,0,1)^T \)

**Proof.**

1. The homography \( H_{\infty} = KR \) is the mapping from the pane at infinity to the image plane.
   Since: 
   \[
   x = KR \begin{bmatrix} I & -\tilde{C} \end{bmatrix} (x, y, z, 0)^T \quad \text{is} \quad x = KR (x, y, z)^T
   \]

2. The conic \( C = \Omega_{\infty} = I_{3 \times 3} \) maps from the plane at infinity to \( \pi_{\infty} \) to the image as:
   \[
   \omega = H_{\infty}^{-T} C H_{\infty}^{-1} = (KR)^{-T} I (KR)^{-1} = K^{-T} R^{-T} R^{-1} K^{-1} = K^{-T} K^{-1}
   \]

**Note,** \( \omega \) depends on \( K \) only and not on \( R, \tilde{C} \).
Hence it plays a central role in auto-calibration.
Degrees of Freedom of $\omega$

- We have:

$$K = \begin{bmatrix} f & s & p_x \\ 0 & mf & p_y \\ 0 & 0 & 1 \end{bmatrix} \text{ then } K^{-1} = \begin{bmatrix} a & b & c \\ 0 & d & e \\ 0 & 0 & 1 \end{bmatrix}$$

where $a, b, c, d, e$ are some values that depend on: $f, m, s, p_x, p_y$

- Then it is:

$$\omega = (K^{-1})^T K^{-1} = \begin{bmatrix} a & 0 & 0 \\ b & d & 0 \\ c & e & 1 \end{bmatrix} \begin{bmatrix} a & b & c \\ 0 & d & e \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} a^2 & ab & ac \\ ab & b^2 + d^2 & bc + de \\ ac & bc + de & c^2 + e^2 + 1 \end{bmatrix}$$

$$= \begin{bmatrix} \omega_1 & \omega_2 & \omega_3 \\ \omega_2 & \omega_4 & \omega_5 \\ \omega_3 & \omega_5 & \omega_6 \end{bmatrix}$$

- This means that $\omega$ has 5 DoF (scale is not unique)
How to compute the IAC?

• External constraints:
  orthogonal directions, etc.

• Internal constraints:
  • Square pixels (i.e. $m = 1, s = 0$ in $K$)
  • Camera matrix is the same in two or more views
    (e.g. video sequence with no zooming)

• If only internal constraints are used then it is called auto/self-calibration.

$$K = \begin{bmatrix}
    f & s & p_x \\
    0 & mf & p_y \\
    0 & 0 & 1
\end{bmatrix}$$
camera matrix
How to compute the IAC?

<table>
<thead>
<tr>
<th>Condition</th>
<th>constraint</th>
<th>type</th>
<th># constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>vanishing points $v_1, v_2$ corresponding to orthogonal lines</td>
<td>$v_1^T \omega v_2 = 0$</td>
<td>linear</td>
<td>1</td>
</tr>
<tr>
<td>vanishing point $v$ and vanishing line $l$ corresponding to orthogonal line and plane</td>
<td>$[l]_x \omega v = 0$</td>
<td>linear</td>
<td>2</td>
</tr>
<tr>
<td>metric plane imaged with known homography $H = [h_1, h_2, h_3]$</td>
<td>$h_1^T \omega h_2 = 0$</td>
<td>linear</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>$h_1^T \omega h_1 = h_2^T \omega h_2$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>zero skew</td>
<td>$\omega_{12} = \omega_{21} = 0$</td>
<td>linear</td>
<td>1</td>
</tr>
<tr>
<td>square pixels</td>
<td>$\omega_{12} = \omega_{21} = 0$</td>
<td>linear</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>$\omega_{11} = \omega_{22}$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 8.1. Scene and internal constraints on $\omega$.

We need 5 constraints on $\omega$ to determine it uniquely.
Example: internal + external constraints

- Square pixel cameras (i.e. \( m = 1, s = 0 \) in \( K \)) gives two constraints: \( \omega_1 = \omega_4 \) and \( \omega_2 = 0 \)

- Then \( \omega = \begin{bmatrix} \omega_1 & 0 & \omega_3 \\ 0 & \omega_1 & \omega_5 \\ \omega_3 & \omega_5 & \omega_6 \end{bmatrix} \) with only 3 DoF

- Given 3 image points \( v_1 \sim 3 \) that point to orthogonal directions, respectively:
  \[
  v_1^T \omega v_2 = 0; \ v_1^T \omega v_3 = 0; \ v_2^T \omega v_3 = 0
  \]

- This gives an equation system \( A\omega = 0 \) with \( A \) of size \( 3 \times 4 \). Hence \( \omega \) can be obtained with SVD

- \( K \) can be derived from \( \omega \) using Cholesky decomposition
Example: internal constraints (practically important)

- Assume two cameras with: $s = 0$, and $m, p_x, p_y$ known.
- The only unknowns are the two focal lengths: $f_0, f_1$.
- Then you get the so-called Kruppa equations:

$$\frac{u_1^T \omega_0^{-1} u_1}{\sigma_0^2 v_0^T \omega_1^{-1} v_0} = \frac{u_0^T \omega_0^{-1} u_1}{\sigma_0 \sigma_1 v_0^T \omega_1^{-1} v_1} = \frac{u_0^T \omega_0^{-1} u_0}{\sigma_1^2 v_1^T \omega_1^{-1} v_1}$$

where SVD of $F = [u_0 \ u_1 \ e_1] \begin{bmatrix} \sigma_0 & 0 & 0 \\ 0 & \sigma_1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} v_0^T \\ v_1^T \\ e_0^T \end{bmatrix}$

and $\omega_i^{-1} = (K_i^{-T} K_i^{-1})^{-1} = K_i K_i^T = \text{diag}(f_i^2, f_i^2, 1)$

- This can be solved in closed form (see next slide).
Sketch of the solution for \( f_0, f_1 \)

\[
\frac{a + b f_0^2}{c + d f_1^2} = \frac{a' + b' f_0^2}{c' + d' f_1^2} = \frac{a'' + b'' f_0^2}{c'' + b'' f_1^2}
\]

(Sliding const)

\[
d \cdot a + b f_0^2 + c f_0^2 + d f_0 f_1^2 = 0 \\
\quad a' + b' f_1^2 + c' f_0^2 + d' f_0 f_1^2 = 0
\]

s.t. \( x = f_0^2 \quad y = f_1^2 \)

\( \begin{align*}
1 & \quad a + b y + c x^2 + d x y = 0 & \rightarrow & & x = \frac{-a - b y}{c + d y} \\
2 & \quad a' + b' y + c' x + d' x y = 0 \\
\text{and} & \quad 1 \text{ in } 2 \\
& \quad a' + b' y + c' x + d' x y = 0 \quad \left| \begin{array}{c}
\left( \frac{a - b y}{c + d y} \right) \\
\end{array} \right. \quad (c + d y)
\end{align*} \)

\( \rightarrow \quad a + b y + c y^2 = 0 \)

\( \rightarrow \quad y = \pm \sqrt{\frac{a}{c}} \)

\( \rightarrow \quad f_1 = \pm \sqrt{\frac{a}{c}} = d^{\frac{1}{4}} \quad \text{since } f_1 \text{ positive} \)

from 0: \( a + b f_0^2 = 0 \rightarrow f_0 = \sqrt[4]{\frac{a}{b}} \)
Example: internal constraints (sketch)

• Assume $K$ is constant over 3+ frames then $K$ can be computed

• We know (lecture 6) we can get $K, R, \tilde{C}$ from $P = KR (I_{3\times3} | - \tilde{C})$

• We have already $P_1, P_2, P_3$ and it is
  \[
  x_{i1} = P_1 X_i = P_1 Q^{-1} QX_i = P'_1X'_i
  \]
  \[
  x_{i2} = P_2 X_i = P_2 Q^{-1} QX_i = P'_2X'_i
  \]
  \[
  x_{i3} = P_3 X_i = P_3 Q^{-1} QX_i = P'_3X'_i
  \]

• Try to find a $Q$ such that all $P_1, P_2, P_3$ have the same $K$ but different $R_{1-3}$ and $\tilde{C}_{1-3}$

• See details in chapter 19 HZ
How to “upgrade” a reconstruction

Illustrating some ways to upgrade from Projective to Affine and Metric (see details in HZ page 270ff and chapter 19)

- Camera is calibrated
- Calibration from external constraints (example: 5 known 3D points)
- Calibration from a mix of in- and external constraints (example: single camera: 3 orthogonal vanishing points and a square pixel camera)
- Calibration from internal constraints, called self/auto calibration (example: 3+ views with same \( K \); or two cameras with unknown focal length)
  - Find plane at infinity and move in canonical position:
    - One of the cameras is affine. See HZ page 271)
    - 3 non-collinear 3D vanishing points
  - Translational motion (HZ page 268)
Roadmap this lecture

- Two-view reconstruction
- From projective to metric space (e.g. self-calibration)
- Multi-view reconstruction
  - Iterative projective cameras
  - Closed form: affine cameras
  - Closed form: reference plane

Next lecture:
- dense labeling problems in computer vision:
  - stereo matching
  - ICP, KinectFusion