TSBB09 Image Sensors

Lecture X
Panorama Stitching

Equivalent cameras

- Two cameras are equivalent if they share a common camera centre **n**
- For equivalent camera matrices **C** and **C'**:

$$\boxed{\begin{aligned}\mathbf{C}\,\mathbf{n} &= \mathbf{0}\\ \mathbf{C}'\,\mathbf{n} &= \mathbf{0}\end{aligned}}$$

Equivalent cameras

- A 3D point **x** is mapped by the two cameras as

$$\boxed{\begin{aligned}\mathbf{y} &\sim \mathbf{C}\,\mathbf{x}\\ \mathbf{y}' &\sim \mathbf{C}'\,\mathbf{x}\end{aligned}}$$

and we have already shown that, in this case,

$$\mathbf{y}' \sim \mathbf{C}'\mathbf{C}^{+}\,\mathbf{y} = \mathbf{H}\,\mathbf{y}$$

A homography

Rotating camera

- A (rather common) special case of equivalent cameras appears when we have one single camera, with fixed internal parameters, and just rotate it about the camera centre
- Let the internal camera parameters be represented by 3×3 matrix **K**
  - Same **K** for both cameras
- Let 3D coordinates be defined relative to coordinate system with origin at **n**
  - This means that **n** = [0 0 0 1]

Rotational homography

This gives

$$\begin{aligned}\mathbf{C} &= \mathbf{K}\,[\mathbf{R}_1\ \mathbf{0}]\\ \mathbf{C}' &= \mathbf{K}\,[\mathbf{R}_2\ \mathbf{0}]\end{aligned}$$

$\mathbf{R}_1$ and $\mathbf{R}_2$ represent the absolute rotation of each camera relative to the 3D coordinate system

- We can set $\mathbf{R}_1 = \mathbf{I}$ and $\mathbf{R}_2 = \mathbf{R}$ = the rotation from the coordinate system of camera 1 to the coordinate system of camera 2

$$\begin{aligned}\mathbf{C} &= \mathbf{K}\,[\mathbf{I}\ \ \mathbf{0}]\\ \mathbf{C}' &= \mathbf{K}\,[\mathbf{R}\ \mathbf{0}]\end{aligned}$$

Rotational homography

Furthermore:

$$\mathbf{C}\,\mathbf{C}^T = \mathbf{K}\,[\mathbf{I}\ \mathbf{0}]\ [\mathbf{I}\ \mathbf{0}]^T\,\mathbf{K}^T = \mathbf{K}\,\mathbf{K}^T$$

$$\mathbf{C}'\mathbf{C}^T = \mathbf{K}\,[\mathbf{R}\ \mathbf{0}]\ [\mathbf{I}\ \mathbf{0}]^T\,\mathbf{K}^T = \mathbf{K}\,\mathbf{R}\,\mathbf{K}^T$$

Finally: $\mathbf{H} = \underbrace{\mathbf{K}\,\mathbf{R}\,\mathbf{K}^T(\mathbf{K}\,\mathbf{K}^T)^{-1}}_{\mathbf{C}'\,\underbrace{\mathbf{C}^T(\mathbf{C}\,\mathbf{C}^T)^{-1}}_{\mathbf{C}^{+}}} = \mathbf{K}\,\mathbf{R}\,\mathbf{K}^{-1}$

Summary

- We take two images with a camera that just rotates about its centre. Corresponding image coordinates in the two images are then related as

$$\mathbf{y} = \mathbf{H}\,\mathbf{y}'$$

where **H** a homography given by

$$\mathbf{H} = \mathbf{K}\,\mathbf{R}\,\mathbf{K}^{-1} \Rightarrow \mathbf{R} = \mathbf{K}^{-1}\,\mathbf{H}\,\mathbf{K}$$

If **H** and **K** can be determined, we can also determine **R**

where **R** is the relative rotations between the cameras and **K** is the internal camera calibration

Panorama stitching

- In panorama stitching, we have a set of images that are all taken **from the same view-point** but in different directions
- Given that the objects in the images are far away this implies that the **camera matrices are approximately equivalent**
  - The camera centres do not have to be exactly at the same point

Panorama stitching

- Each image can be transformed into any other by a homography
  - At least in the overlapping region in the two images
- In this region the homography can be estimated (how?)
- By applying the homography to an entire image from camera **C**', it can the *stitched* onto the image from camera **C**
- Given a set of images, they can be stitched into some *reference image* in the set by determining the corresponding homography for each other image in the set relative to the reference image

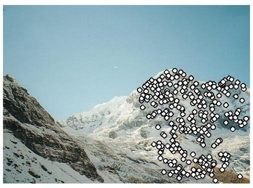
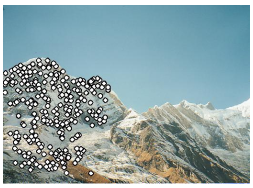
Example

- Two images


Images from: *Automatic Panoramic Image Stitching using Invariant Features*, IJCV 2007, Matthew Brown

Example

- From a set of corresponding points

- Estimate **H** that relates the 2 images

Example

- Right image stitched onto the left image

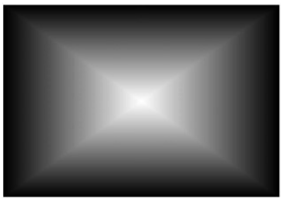
Practical issues

- The pixel values in overlapping regions may differ even if the geometric transformation is correct
  - Vignetting effects
  - Interpolation effects
  - Exposure time or illumination may be different in two images
  - Moving objects in the scene
- At each pixel:
  - Take the value from only one of the two images
  - Alternatively: blend

Blending weight

- For example, use a weight that is smaller at the edges of the image and larger at the center

This is the weight image before the homography transformation



Blending

Without blending

With blending

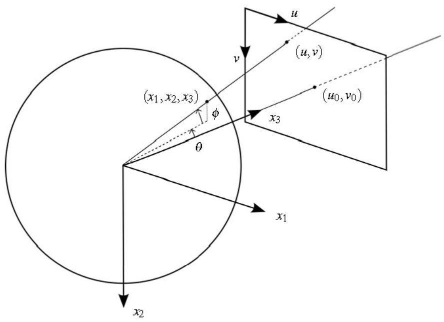
Practical issues

- To assure a homography between any pair of images, we must have a pin-hole camera
  - No significant amount of lens distortion
  - Alternatively: lens distortion can be estimated and compensated for before the stitching

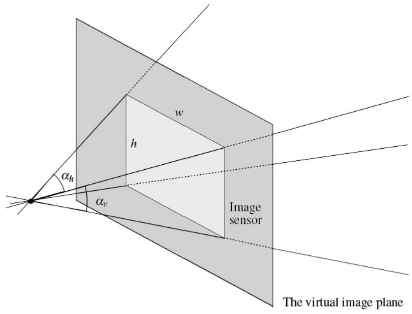
Practical issues

- If the view directions between two images are very different, the corresponding homography will introduce a large amount of geometric distortion in the stitched image
- Proper points in one image -> points at ∞ in another
  - Map the images onto a sphere and stitch them there instead of in the image plane of one of the cameras
  - Image points represented by unit vectors in $\mathbb{R}^3$
- In this case the vectors are related by the rotation **R** between the two images!

Mapping to spherical coordinates



Field of view (FOV)



Field of view (FOV)

- The two field of view angles can be computed from the internal camera parameters

$$\tan\frac{\alpha_h}{2} = \frac{w}{2k_{11}}$$

$$\tan\frac{\alpha_v}{2} = \frac{h}{2k_{22}}$$

- Assumes that the *principal point* (=intersection of the optical axis) has pixel coordinates $(k_{13},k_{23})/2$

Images of a plane

Alternative application of the **homography mapping**:

Images of a 3D plane
- Facades of building
- Aerial images of the Earth (satellite, aircraft…)
  - Google Earth

Interpolation of images in “Google street view”

Break

Eigenvalues and eigenvectors

Definition:

- Given an $N \times N$ matrix **A**, an N-dimensional vector
  e $\neq$ **0**, and a scalar λ that satisfy

  $$\mathbf{A}\,\mathbf{e} = \lambda\,\mathbf{e}$$

  e is an *eigenvector* of **A**, with *eigenvalue* λ

The spectral theorem

If **A** is real and symmetric ($\mathbf{A}^T = \mathbf{A}$) then

- All eigenvalues are real
- We can determine an *orthonormal basis* of $\mathbb{R}^N$ consisting of eigenvectors of **A**

The spectral theorem

An algebraic consequence of this theorem:

- When **A** is symmetric, we can find an orthogonal matrix **E** such that

  A = E D E^T

- **E** holds an ON-basis of eigenvectors in its columns
- **D** is a diagonal matrix of the corresponding eigenvalues

Eigenvalue decomposition

- The spectral theorem implies that any symmetric matrix **A** can be decomposed into a matrix product of orthogonal and diagonal matrices
- This is the *eigenvalue decomposition* (EVD) of symmetric matrices

Eigenvalue decomposition

- Obvious limitations of the eigenvalue decomposition
  - Cannot cope with non-square matrices
  - Even some square matrices cannot be decomposed
- These shortcomings can be overcome using the singular value decomposition

Singular value decomposition

Theorem:

- For any $N \times M$ real valued matrix **A**,
  - we can find an $N \times N$ orthogonal matrix **U**
  - we can find an $M \times M$ orthogonal matrix **V**
  - we can find an $N \times M$ real diagonal matrix **S**
  - such that

A = U S V^T

This is the *singular value decomposition* (SVD) of **A**

Singular value decomposition

Example: **A** is 3×4

$$\mathbf{A} = \begin{pmatrix} | & | & | \\ \mathbf{u}_1 & \mathbf{u}_2 & \mathbf{u}_3 \\ | & | & | \end{pmatrix} \begin{pmatrix} \sigma_1 & 0 & 0 & 0 \\ 0 & \sigma_2 & 0 & 0 \\ 0 & 0 & \sigma_3 & 0 \end{pmatrix} \begin{pmatrix} - & \mathbf{v}_1^T & - \\ - & \mathbf{v}_2^T & - \\ - & \mathbf{v}_3^T & - \\ - & \mathbf{v}_4^T & - \end{pmatrix}$$

Matrix **U**
3×3
U $\in$ O(3)

Matrix **S**
3×4

Matrix **V**^{T}
4×4
V $\in$ O(4)

Singular value decomposition

- **S** is $N \times M$ diagonal (non-zero values only in the diagonal)
- The diagonal elements of **S** , σ_1, ..., σ_P, are real and non-negative (with P = min(N, M))
- They are the *singular values* of **A**
- The singular values are usually ordered such that $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_P$

Singular value decomposition

- For singular value σ_k, the corresponding columns $\mathbf{u}_k$ and $\mathbf{v}_k$ of **U** and **V** are the *left and right singular vectors* of **A**, respectively
- Notice that

$$\mathbf{A}\,\mathbf{v}_k = \sigma_k\,\mathbf{u}_k$$
$$\mathbf{A}^T\mathbf{u}_k = \sigma_k\,\mathbf{v}_k$$

Singular value decomposition

- In the case of a non-square matrix **A** there will be some left (or right) singular vectors that neither have a corresponding singular value σ_k nor a right (or a left) singular vector
- In this case they are simply said to have singular value 0 since, for example

$\mathbf{A}\,\mathbf{v}_k = \mathbf{0}$ $\quad$ ($k > P$, and $N < M$)

SVD and EVD

- Let **A** be $N \times M$,
- Let σ_k be a singular value of **A**, with left and right singular vectors $\mathbf{u}_k$ and $\mathbf{v}_k$
- Then

$$\mathbf{A}^T\mathbf{A}\,\mathbf{v}_k = \sigma_k^2\,\mathbf{v}_k$$

$\mathbf{v}_k$ is an eigenvector of $\mathbf{A}^T\mathbf{A}$, with eigenvalue σ_k^2

$$\mathbf{A}\,\mathbf{A}^T\,\mathbf{u}_k = \sigma_k^2\,\mathbf{u}_k$$

$\mathbf{u}_k$ is an eigenvector of $\mathbf{A}\mathbf{A}^T$, with eigenvalue σ_k^2

Computation of SVD

- This result suggests that the SVD of **A** can be computed by an EVD of $\mathbf{A}^T\mathbf{A}$ and $\mathbf{A}\,\mathbf{A}^T$
- This is in principle correct, but numerically not a good approach in general
- These matrix products, in particular when *N* or *M* are large, or **A** is close to singular, can introduce numerical errors that produce larges errors in the resulting **U**, **S**, and **V**
- Instead the SVD can be computed using special numerical algorithms, not discussed here

SVD and Matlab

- Matlab can compute the singular value decomposition of matrix `A`:

```
[U S V]=svd(A)
```

produces orthogonal matrices `U` and `V` of singular vectors and a diagonal matrix `S` of singular values

Why SVD?

- SVD can be applied to **any matrix**,
  - this is not the case for EVD
- Some common applications:
  - The null space of **A** is spanned by the right singular vectors of singular value 0
  - The range of **A** is spanned by the left singular vectors of singular values > 0
  - The rank of **A** is the number of non-zero singular values
  - (why?)

The orthogonal Procrustes problem

- The problem of determining **R** $\in$ SO(3) that minimises

$$\epsilon_{OPP} = \sum_{k=1}^{N} \|\mathbf{r}'_k - \mathbf{R}\,\mathbf{r}_k\|^2$$

with known {$\mathbf{r}'_k$, $\mathbf{r}_k$, k = 1, ..., N} is sometimes called the *orthogonal Procrustes problem* (OPP)
- Correspondences between {$\mathbf{r}'_k$, $\mathbf{r}_k$} is known!

The orthogonal Procrustes problem

- Let **A** be a matrix of all vectors $\mathbf{r}'_k$ in its columns
- Let **B** be a matrix of all vectors $\mathbf{r}_k$ in its columns
- The OPP then implies to minimise

$$\epsilon_{\mathrm{OPP}} = \|\, \mathbf{A} - \mathbf{R}\,\mathbf{B} \,\|^2$$

This is the Frobenius norm for matrices

with **R** $\in$ O(3)

The set of all 3 $\times$ 3 matrices **R** such that $\mathbf{R}^T\mathbf{R} = \mathbf{I}$

Solving OPP

- We want to determine **R** $\in$ O(3) that minimises

$$\boxed{\sum_{k=1}^{N} \|\mathbf{r}'_k - \mathbf{R}\,\mathbf{r}_k\|^2}$$

- Solution
  1. Form matrices **A** and **B** from $\mathbf{r}_k$ and $\mathbf{r}'_k$
  2. Compute SVD: $\mathbf{B}\,\mathbf{A}^T = \mathbf{U}\,\mathbf{S}\,\mathbf{V}^T$
  3. $\mathbf{R} = \mathbf{V}\,\mathbf{U}^T$

This algorithm can be modified to assure that det R = 1