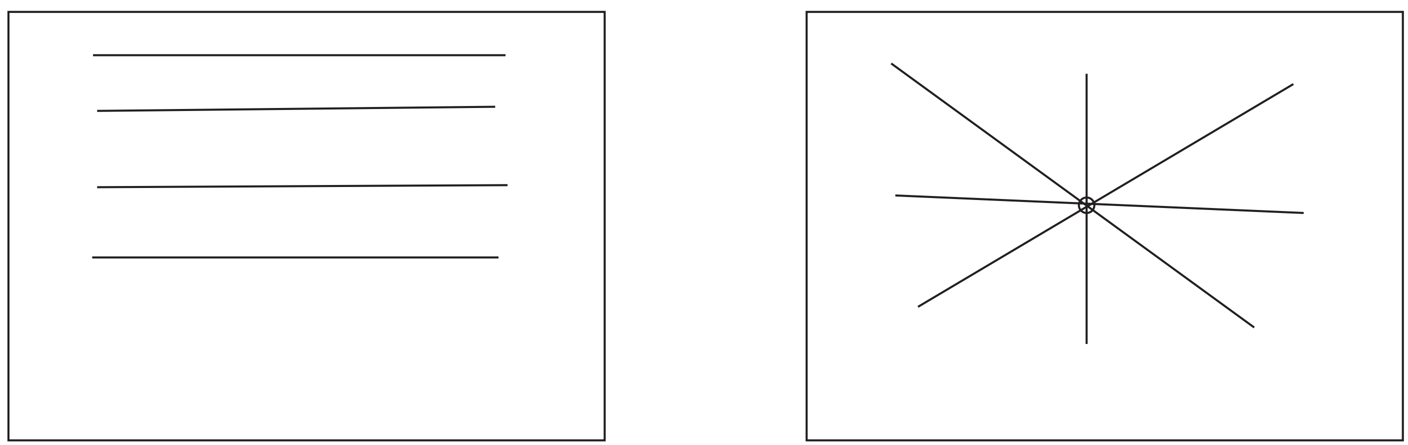
Visual odometry

D.A. Forsyth UIUC

Epipoles (resp. epipolar lines)

- Informative



Epipole and epipolar lines in camera 1 - where is camera 2?

Odometry from two camera geometry

- Idea:
  - use calibrated camera
  - move; track some points
    - in reading slides
  - compute essential matrix (calibrated fundamental matrix) to get
    - rotation
    - translation up to scale
- Options:
  - fix scale later
  - use (say) wheel measurements + Kalman filter to fix
  - use stereo

RECALL: The Fundamental Matrix

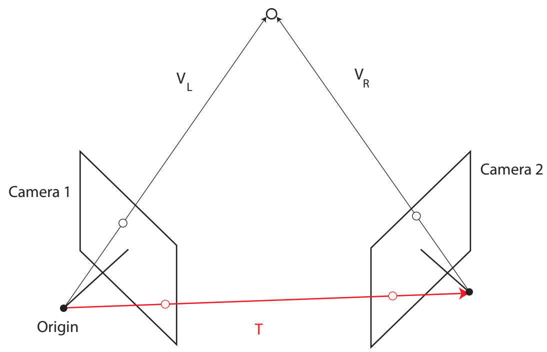
$$\mathbf{p_1}^T \mathcal{F} \mathbf{p_2} = 0$$

- Can be fit a pair of images using feature correspondences
  - 8 point algorithm
  - robustness is an important issue
  - we'll do this

$$\mathcal{F} = k\mathcal{C}_L^{-T} \mathcal{R}\mathcal{S}\mathcal{C}_R^{-1}$$

If we know these

we can recover info about R, T from F



Camera translation

$$\mathbf{V}_R = \mathcal{R}(\mathbf{V}_L - \mathbf{T})$$

Camera rotation

$$\mathcal{S} = \begin{pmatrix} 0 & -T_z & T_y \\ T_z & 0 & -T_x \\ -T_y & T_x & 0 \end{pmatrix}$$

$$\mathbf{T}^T \mathcal{S} = \mathbf{0}$$

The Essential matrix

- Assume camera calibration is known
  - Cameras are normalized so that C=Id

$$\mathbf{p_1}^T \mathcal{F} \mathbf{p_2} = 0$$

becomes

$$\mathbf{p}_1^T \mathcal{E} \mathbf{p}_2 = 0$$

$$\mathcal{F} = k \mathcal{C}_L^{-T} \mathcal{R} \mathcal{S} \mathcal{C}_R^{-1}$$

becomes

$$\mathcal{E} = k \mathcal{R} \mathcal{S}$$

The essential matrix

From fundamental matrix

Antisymmetric

$$\mathcal{E} = k\mathcal{R}\mathcal{S}$$

Orthonormal

Unknown constant

Getting R, S from E

- Recall SVD:

Diagonal

$$\mathcal{M} = \mathcal{U}\Sigma\mathcal{V}^T$$

Orthonormal

- Notice that, for R a rotation,
  - M and RM have the same singular values
- So singularvalues(E)=singularvalues(S)
  - check:

$$\Sigma(\mathcal{S}) = \begin{pmatrix} s & 0 & 0 \\ 0 & s & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

Recovering R, S - I

- Write

$$\mathcal{W} = \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

U Sigma V^T = SVD(E)

$$\mathcal{R} = \mathcal{U}\mathcal{W}^{-1}\mathcal{V}^T$$
$$\mathcal{S} = \mathcal{V}\mathcal{W}\Sigma\mathcal{V}^T$$

- Check that
  - RS=E
  - R is orthonormal
  - S is antisymmetric

BUT

- There are ambiguities
  - check that for any Q of the form
    - square root of identity

$$\mathcal{Q} = \mathrm{diag}(\pm 1, \pm 1, \pm 1)$$

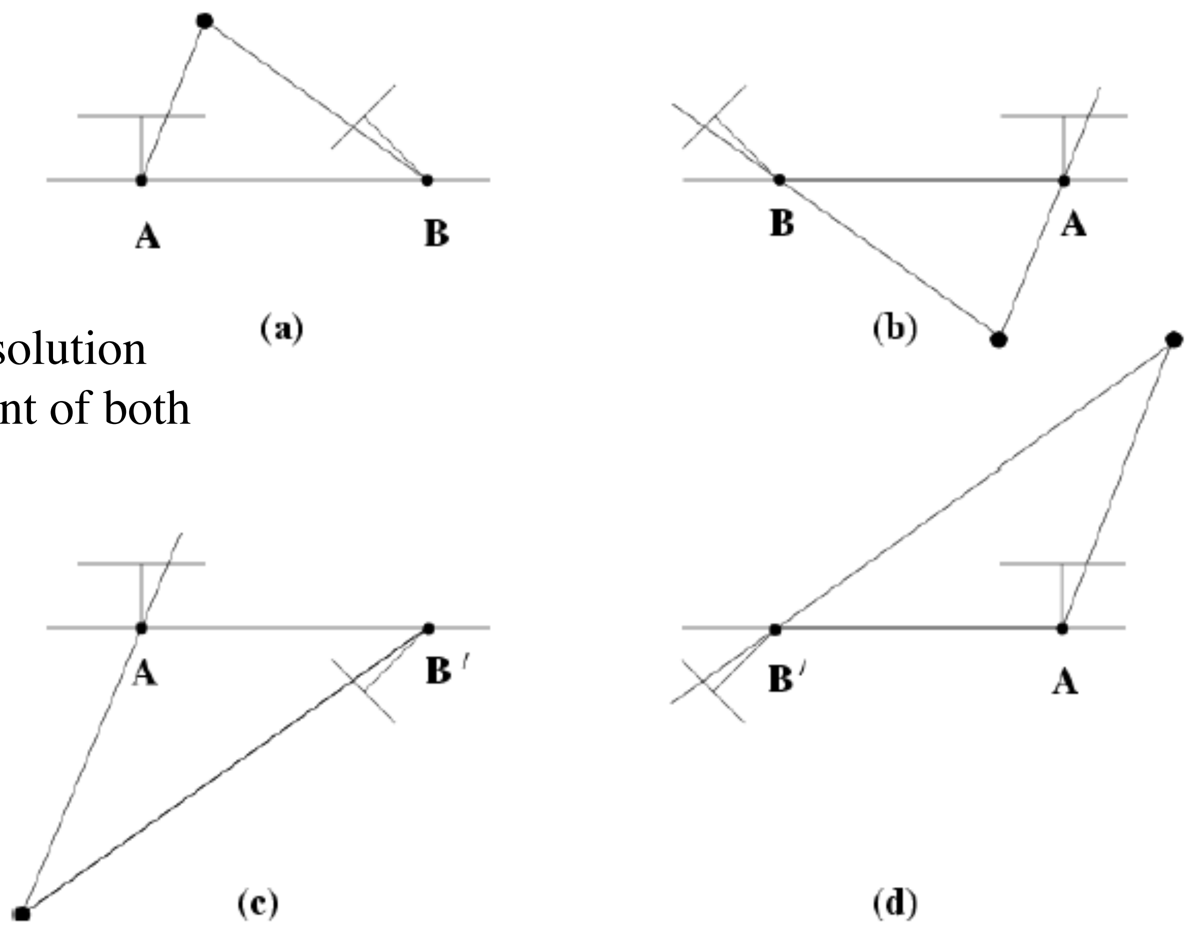
  - R’, S’ as given also work
    - R’ is orthonormal
    - S’ is antisymmetric

$$\mathcal{S}' = \mathcal{Q}\mathcal{S}$$

$$\mathcal{R}' = \mathcal{R}\mathcal{Q}$$

- Four of these don’t matter
  - cause det(R’)=-1
  - implies camera was reflected as well as rotated
    - and that doesn’t happen

The other four…..



Only one gives a solution where point is in front of both cameras.

S. Weiss' notes on visual odometry from CVPR 14 tutorial

But the unknown constant is unknown…

Antisymmetric

$$\mathcal{E} = k\mathcal{R}\mathcal{S}$$

Orthonormal

Unknown constant

Different values of k will lead to different scales of S - equivalently, different scales of translation between cameras - you need extra information to sort this out.

What we have

- Can determine
  - the rotation between two cameras
  - the translation *up to scale*
- From this, we can recover 3D points
  - up to scale

What we have

- 3D points: $\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}$ And $\begin{pmatrix} x_1^t \\ x_2^t \\ x_3^t \end{pmatrix} = \mathcal{R}\left[\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} - \mathbf{t}\right]$

Original point in camera two's coordinate system

- normalized image points:

$$\begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \begin{pmatrix} x_1/x_3 \\ x_2/x_3 \end{pmatrix} \qquad \begin{pmatrix} y_1^t \\ y_2^t \end{pmatrix} = \begin{pmatrix} x_1^t/x_3^t \\ x_2^t/x_3^t \end{pmatrix}$$

Recovering the point in 3D

- Write

$$\mathcal{R} = \begin{pmatrix} \mathbf{r}_1^T \\ \mathbf{r}_2^T \\ \mathbf{r}_3^T \end{pmatrix}$$

$$\mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ 1 \end{pmatrix}$$

- Then

$$x_3 = \frac{(\mathbf{r}_1 - y_1^t \mathbf{r}_3)^T \mathbf{t}}{(\mathbf{r}_1 - y_2^t \mathbf{r}_3)^T \mathbf{y}}$$

And we have everything in 3D!

The effect of scale

$$x_3 = \frac{(\mathbf{r}_1 - y_1^t \mathbf{r}_3)^T \mathbf{t}}{(\mathbf{r}_1 - y_2^t \mathbf{r}_3)^T \mathbf{y}}$$

- Notice that if k changes, t gets bigger or smaller
  - point coordinates scale
    - x_1=y_1 x_3, x_2=y_2 x_3
- So if we can match points across more than two cameras
  - there is only one scale ambiguity in the whole sequence
  - This could be quite easy to sort out
    - eg you know the size of high bay
    - eg you know some reference scale
    - etc

Alternatives

- Filter the scale using estimates from wheels
  - etc
- Stereo odometry
  - If I have two cameras then there is no issue with scale

Pragmatics

- Need
  - good fast feature computation and tracking
    - fast features and good robust methods seem to beat good features

  - reliable camera calibration
    - and robust FM/EM estimation
    - Ransac remains reliable

- More to follow once we've seen filtering