Computer vision: models, learning and inference

Chapter 16

Multiple Cameras

Photo Tourism

Exploring photo collections in 3D

Noah Snavely Steven M. Seitz Richard Szeliski

University of Washington *Microsoft Research*

SIGGRAPH 2006

Structure from motion

Given

- an object that can be characterized by I 3D points $\{\mathbf{w}_i\}_{i=1}^{I}$
- projections into J images $\{\mathbf{x}_{ij}\}_{i=1,j=1}^{I,J}$

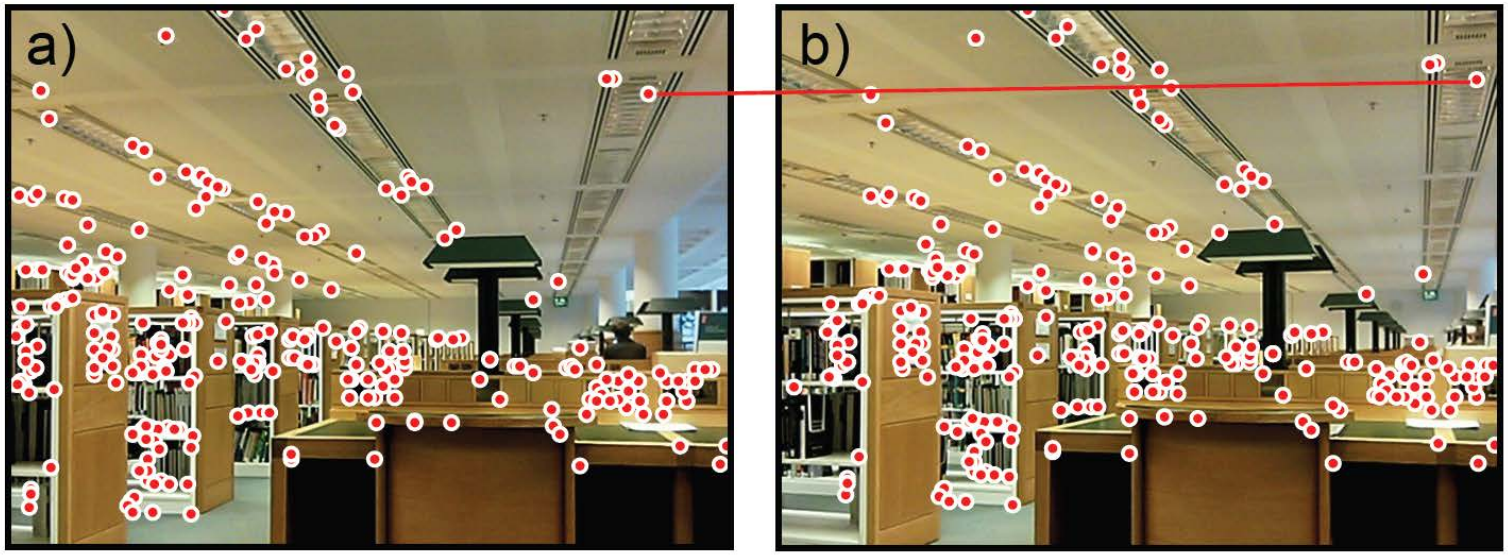
Find

- Intrinsic matrix $\mathbf{\Lambda}$
- Extrinsic matrix for each of J images $\{\boldsymbol{\Omega}_j, \boldsymbol{\tau}_j\}_{j=1}^{J}$
- 3D points $\{\mathbf{w}_i\}_{i=1}^{I}$

$$= \underset{\mathbf{w},\boldsymbol{\Omega},\boldsymbol{\tau},\mathbf{\Lambda}}{\operatorname{argmax}} \left[\sum_{i=1}^{I} \sum_{j=1}^{J} \log[Pr(\mathbf{x}_{ij}|\mathbf{w}_i, \mathbf{\Lambda}, \boldsymbol{\Omega}_j, \boldsymbol{\tau}_j)] \right]$$

$$= \underset{\mathbf{w},\boldsymbol{\Omega},\boldsymbol{\tau},\mathbf{\Lambda}}{\operatorname{argmax}} \left[\sum_{i=1}^{I} \sum_{j=1}^{J} \log \left[\text{Norm}_{\mathbf{x}_{ij}} \left[\mathbf{pinhole}[\mathbf{w}_i, \mathbf{\Lambda}, \boldsymbol{\Omega}_j, \boldsymbol{\tau}_j], \sigma^2 \mathbf{I} \right] \right] \right]$$

Structure from motion



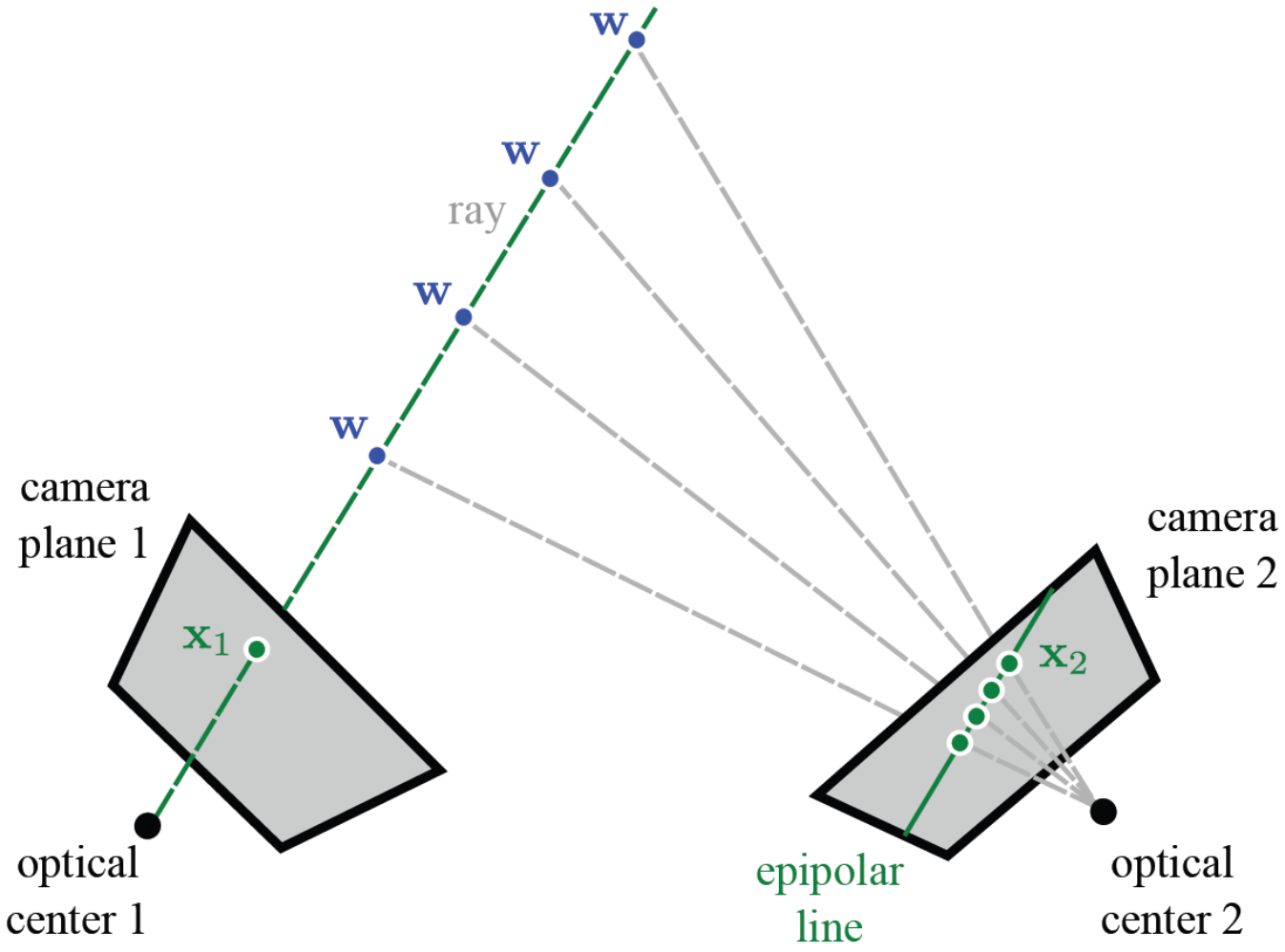
For simplicity, we'll start with simpler problem

- Just J=2 images
- Known intrinsic matrix $\mathbf{\Lambda}$

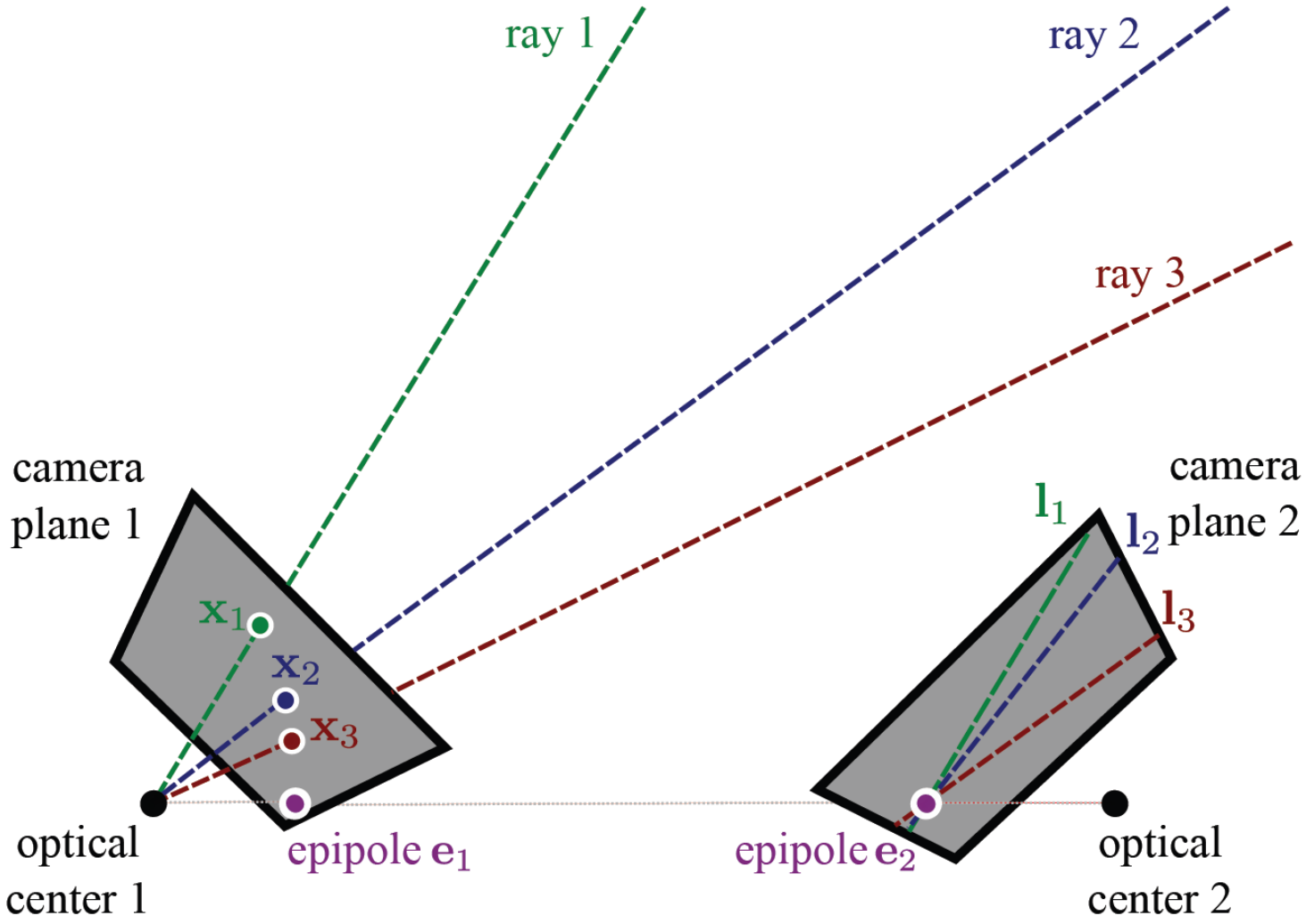
Structure

- Two view geometry
- The essential and fundamental matrices
- Reconstruction pipeline
- Rectification
- Multi-view reconstruction
- Applications

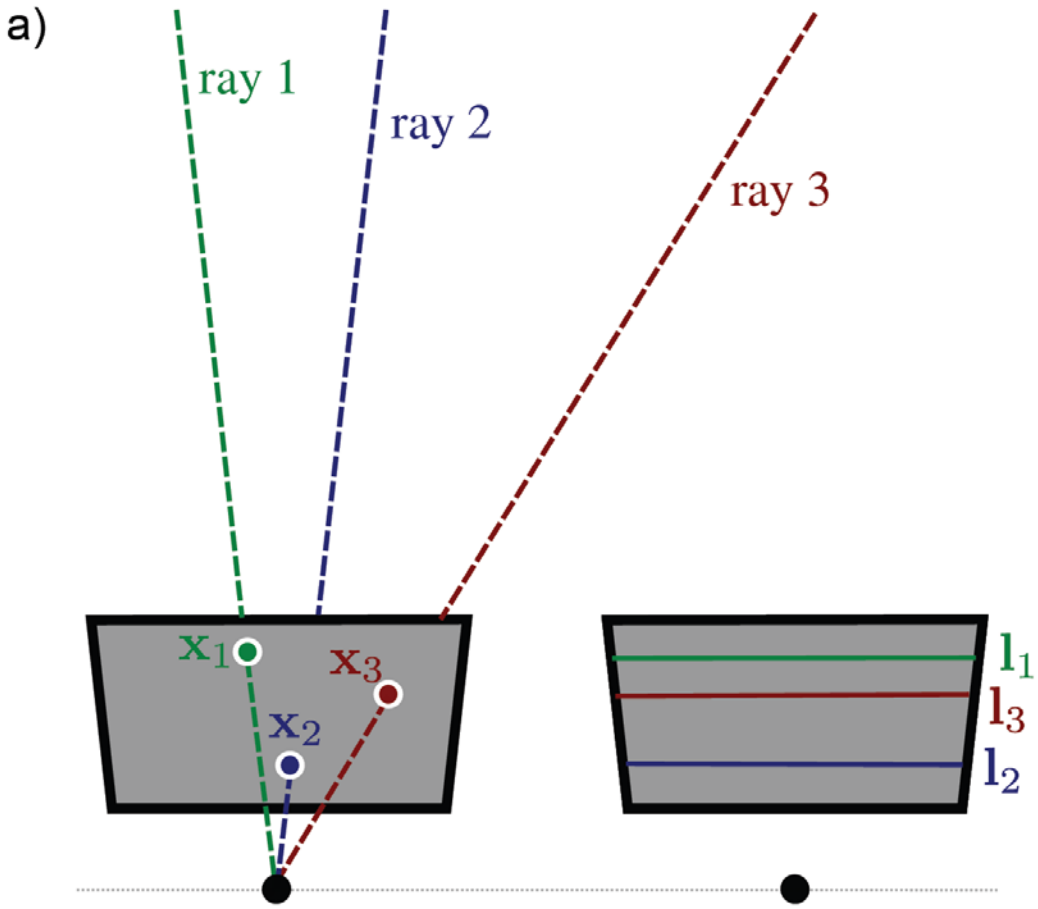
Epipolar lines

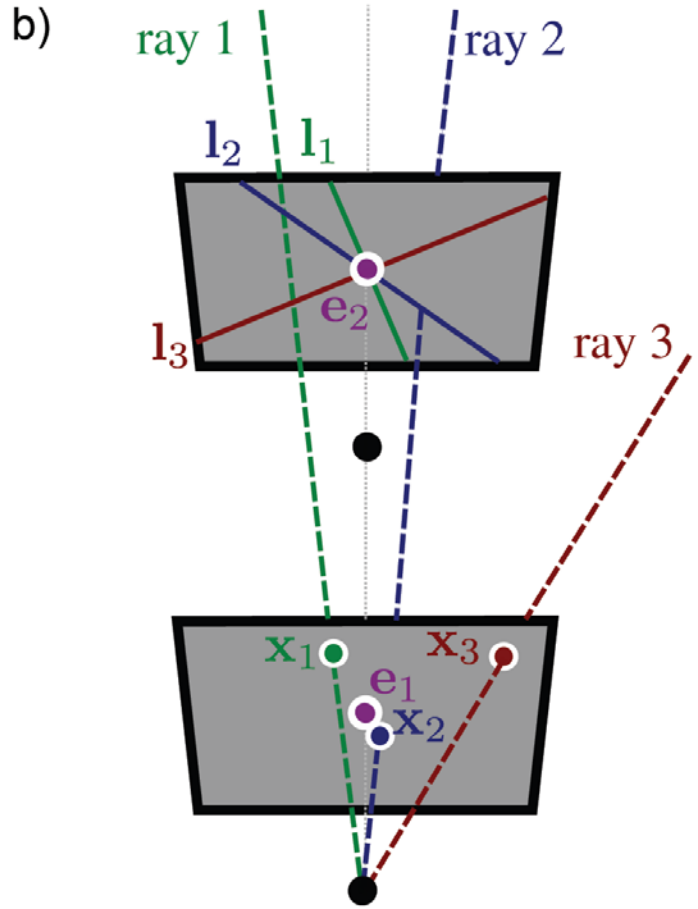


Epipole


ray 1
ray 2
ray 3
camera
plane 1
camera
plane 2
x_1
x_2
x_3
l_1
l_2
l_3
optical
center 1
epipole e_1
epipole e_2
optical
center 2

Special configurations





Structure

- Two view geometry
- The essential and fundamental matrices
- Reconstruction pipeline
- Rectification
- Multi-view reconstruction
- Applications

The essential matrix

The geometric relationship between the two cameras is captured by the essential matrix.

Assume normalized cameras, first camera at origin.

$$\begin{aligned}\lambda_1\tilde{\mathbf{x}}_1 &= [\mathbf{I}, \mathbf{0}]\tilde{\mathbf{w}} \\ \lambda_2\tilde{\mathbf{x}}_2 &= [\boldsymbol{\Omega}, \boldsymbol{\tau}]\tilde{\mathbf{w}}\end{aligned}$$

First camera:

$$\lambda_1 \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} u \\ v \\ w \\ 1 \end{bmatrix} = \begin{bmatrix} u \\ v \\ w \end{bmatrix}$$

$$\lambda_1\tilde{\mathbf{x}}_1 = \mathbf{w}$$

Second camera: $$\lambda_2\tilde{\mathbf{x}}_2 = \boldsymbol{\Omega}\mathbf{w} + \boldsymbol{\tau}$$

The essential matrix

First camera:

$$\lambda_1 \tilde{\mathbf{x}}_1 = \mathbf{w}$$

Second camera:

$$\lambda_2 \tilde{\mathbf{x}}_2 = \boldsymbol{\Omega}\mathbf{w} + \boldsymbol{\tau}$$

Substituting:

$$\lambda_2 \tilde{\mathbf{x}}_2 = \lambda_1 \boldsymbol{\Omega} \tilde{\mathbf{x}}_1 + \boldsymbol{\tau}$$

This is a mathematical relationship between the points in the two images, but it's not in the most convenient form.

The essential matrix

$$\lambda_2\tilde{\mathbf{x}}_2 = \lambda_1\boldsymbol{\Omega}\tilde{\mathbf{x}}_1 + \boldsymbol{\tau}$$

Take cross product with τ (last term disappears)

$$\lambda_2\boldsymbol{\tau}\times\tilde{\mathbf{x}}_2 = \lambda_1\boldsymbol{\tau}\times\boldsymbol{\Omega}\tilde{\mathbf{x}}_1$$

Take inner product of both sides with $\mathbf{x}_2$.

$$\tilde{\mathbf{x}}_2^T\boldsymbol{\tau}\times\boldsymbol{\Omega}\tilde{\mathbf{x}}_1 = 0$$

The essential matrix

$$\tilde{\mathbf{x}}_2^T \boldsymbol{\tau} \times \boldsymbol{\Omega} \tilde{\mathbf{x}}_1 = 0$$

The cross product term can be expressed as a matrix

$$\boldsymbol{\tau}_\times = \begin{bmatrix} 0 & -\tau_z & \tau_y \\ \tau_z & 0 & -\tau_x \\ -\tau_y & \tau_x & 0 \end{bmatrix}$$

Defining:

$$\mathbf{E} = \boldsymbol{\tau}_\times \boldsymbol{\Omega}$$

We now have the essential matrix relation

$$\tilde{\mathbf{x}}_2^T \mathbf{E} \tilde{\mathbf{x}}_1 = 0$$

Properties of the essential matrix

$$\tilde{\mathbf{x}}_2^T \mathbf{E} \tilde{\mathbf{x}}_1 = 0$$

- Rank 2: $\det[\mathbf{E}] = 0$
- 5 degrees of freedom
- Non-linear constraints between elements

$$2\mathbf{E}\mathbf{E}^T\mathbf{E} - \text{trace}[\mathbf{E}\mathbf{E}^T]\mathbf{E} = \mathbf{0}$$

The 3×3 essential matrix captures the geometric relationship between the two cameras and has rank 2 so that $\det[\mathbf{E}] = 0$. The first two singular values of the essential matrix are always identical and the third is zero. It depends only on the rotation and translation between the cameras, each of which has 3 parameters, and so one might think it would have 6 degrees of freedom. However, it operates on homogeneous variables $\tilde{\mathbf{x}}_1$ and $\tilde{\mathbf{x}}_2$ and is hence ambiguous up to scale: multiplying all of the entries of the essential matrix by any constant does not change its properties. For this reason, it is usually considered as having 5 degrees of freedom.

Since there are fewer degrees of freedom than there are unknowns, the nine entries of the matrix must obey a set of algebraic constraints. These can be expressed compactly as

$$2\mathbf{E}\mathbf{E}^T\mathbf{E} - \text{trace}[\mathbf{E}\mathbf{E}^T]\mathbf{E} = \mathbf{0}. \tag{16.12}$$

These constraints are sometimes exploited in the computation of the essential matrix, although in this volume we use a simpler method (section 16.4).

Recovering epipolar lines

Equation of a line:

$$ax + by + c = 0$$

or

$$\begin{bmatrix} a & b & c \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = 0$$

or

$$\mathbf{l}\tilde{\mathbf{x}} = 0$$

Recovering epipolar lines

Equation of a line: $\mathbf{l}\tilde{\mathbf{x}} = 0$

Now consider $\tilde{\mathbf{x}}_2^T \mathbf{E} \tilde{\mathbf{x}}_1 = 0$

This has the form $\mathbf{l}_1 \tilde{\mathbf{x}}_1 = 0$ where $\mathbf{l}_1 = \tilde{\mathbf{x}}_2^T \mathbf{E}$

So the epipolar lines are

$$\begin{aligned} \mathbf{l}_1 &= \tilde{\mathbf{x}}_2^T \mathbf{E} \\ \mathbf{l}_2 &= \tilde{\mathbf{x}}_1^T \mathbf{E}^T \end{aligned}$$

Recovering epipoles

Every epipolar line in image 1 passes through the epipole $\mathbf{e}_1$.

In other words $\tilde{\mathbf{x}}_2^T\mathbf{E}\tilde{\mathbf{e}}_1 = 0$ for ALL $\tilde{\mathbf{x}}_2^T$

This can only be true if $\mathbf{e}_1$ is in the nullspace of $\mathbf{E}$.

$$\tilde{\mathbf{e}}_1 = \mathbf{null}[\mathbf{E}]$$

Similarly:

$$\tilde{\mathbf{e}}_2 = \mathbf{null}[\mathbf{E}^T]$$

We find the null spaces by computing $\mathbf{E} = \mathbf{U}\mathbf{L}\mathbf{V}^T$, and taking the last column of $\mathbf{V}$ and the last row of $\mathbf{U}$.

Decomposition of E

Essential matrix:

$$\mathbf{E} = \boldsymbol{\tau}_{\times}\boldsymbol{\Omega}$$

To recover translation and rotation use the matrix:

$$\mathbf{W} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

We take the SVD $\mathbf{E} = \mathbf{U}\mathbf{L}\mathbf{V}^T$ and then we set

$$\begin{aligned} \boldsymbol{\tau}_{\times} &= \mathbf{U}\mathbf{L}\mathbf{W}\mathbf{U}^T \\ \boldsymbol{\Omega} &= \mathbf{U}\mathbf{W}^{-1}\mathbf{V}^T \end{aligned}$$

We will defer the question of how to compute the essential matrix from a set of corresponding points until section 16.3. For now, we will concentrate on how to decompose a given essential matrix $\mathbf{E}$ to recover this rotation $\boldsymbol{\Omega}$ and translation $\boldsymbol{\tau}$. This is known as the *relative orientation* problem.

In due course, we shall see that we can compute the rotation exactly, whereas it is only possible to compute the translation up to an unknown scaling factor. This remaining uncertainty reflects the geometric ambiguity of the system; from the images alone, we cannot tell if these cameras are far apart and looking at a large distant object or close together and looking at a small nearby object.

To decompose $\mathbf{E}$, we define the matrix

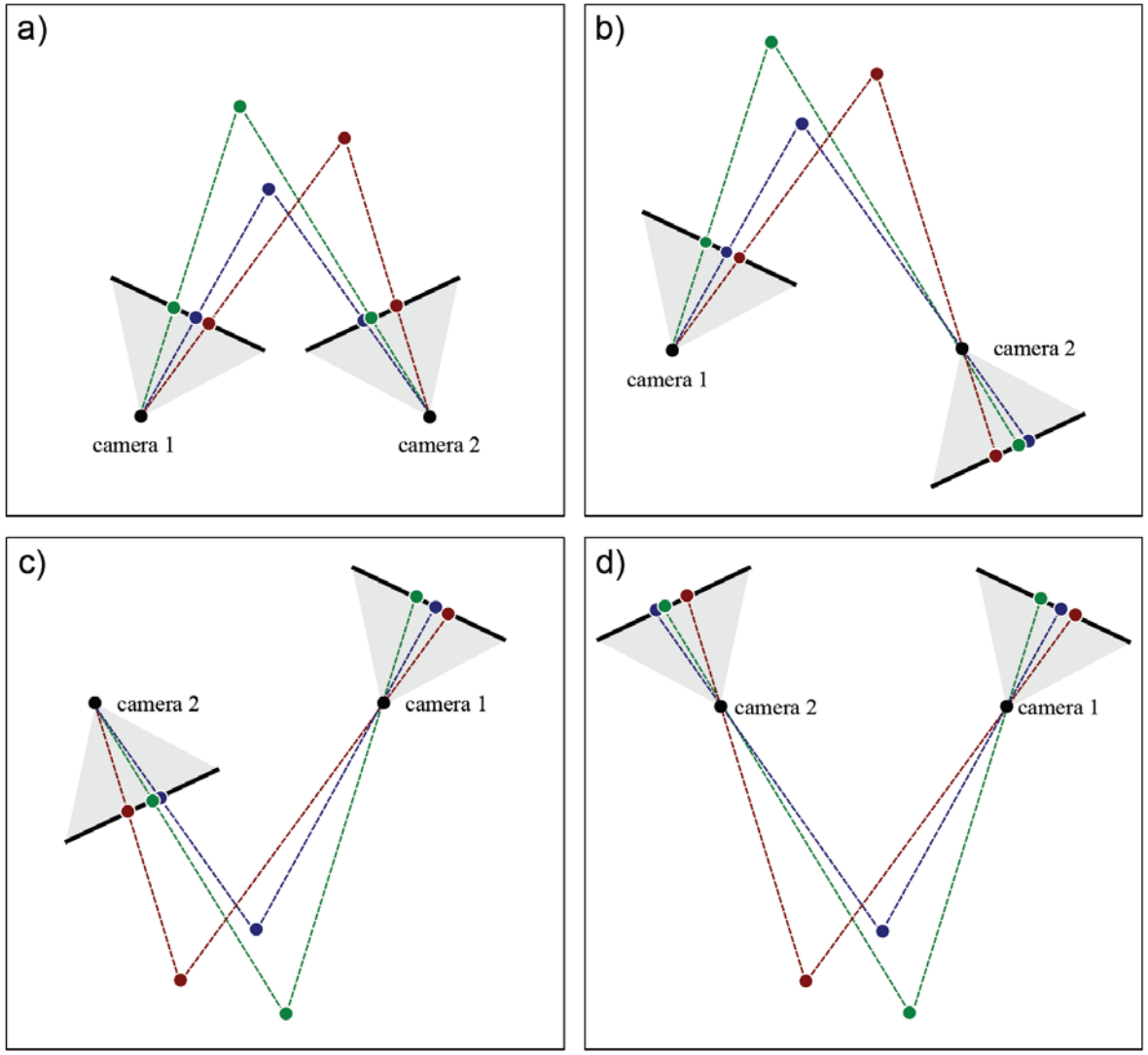
$$\mathbf{W} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{16.18}$$

and then take the singular value decomposition $\mathbf{E} = \mathbf{U}\mathbf{L}\mathbf{V}^T$. We now choose

$$\begin{aligned} \boldsymbol{\tau}_\times &= \mathbf{U}\mathbf{L}\mathbf{W}\mathbf{U}^T \\ \boldsymbol{\Omega} &= \mathbf{U}\mathbf{W}^{-1}\mathbf{V}^T. \end{aligned} \tag{16.19}$$

It is convention to set the magnitude of the translation vector $\boldsymbol{\tau}$ that is recovered from the matrix $\boldsymbol{\tau}_\times$ to unity. The above decomposition is not obvious, but it is easily checked that multplying the derived expressions for $\boldsymbol{\tau}_\times$ and $\boldsymbol{\Omega}$ yields $\mathbf{E} = \mathbf{U}\mathbf{L}\mathbf{V}^T$. This method assumes that we started with a valid essential matrix where the first two singular values are identical and the third is zero. If this is not the case (due to noise) we can substitute $\mathbf{L}' = \mathbf{diag}[1, 1, 0]$ for $\mathbf{L}$ in the solution for $\boldsymbol{\tau}_\times$. For a detailed proof of this decomposition, consult Hartley & Zisserman (2004).

Four interpretations



To get the different solutions, we mutliply τ by -1 and substitute $\mathbf{W}$ for $\mathbf{W}^{-1}$

The fundamental matrix

Now consider two cameras that are not normalised

$$\lambda_1\tilde{\mathbf{x}}_1 = \mathbf{\Lambda}_1[\mathbf{I},\mathbf{0}]\tilde{\mathbf{w}}$$

$$\lambda_2\tilde{\mathbf{x}}_2 = \mathbf{\Lambda}_2[\mathbf{\Omega},\boldsymbol{\tau}]\tilde{\mathbf{w}}$$

By a similar procedure to before, we get the relation

$$\tilde{\mathbf{x}}_2^T\mathbf{\Lambda}_2^{-T}\mathbf{E}\mathbf{\Lambda}_1^{-1}\tilde{\mathbf{x}}_1 = 0$$

or

$$\tilde{\mathbf{x}}_2^T\mathbf{F}\tilde{\mathbf{x}}_1 = 0$$

where

$$\mathbf{F} = \mathbf{\Lambda}_2^{-T}\mathbf{E}\mathbf{\Lambda}_1^{-1} = \mathbf{\Lambda}_2^{-T}\boldsymbol{\tau}_\times\mathbf{\Omega}\mathbf{\Lambda}_1^{-1}$$

Relation between essential and fundamental

$$\mathbf{E} = \mathbf{\Lambda}_2^T\mathbf{F}\mathbf{\Lambda}_1$$

Fundamental matrix criterion

Figure 15.5 Cost function for estimating fundamental matrix. The point $\mathbf{x}_{i1}$ in image 1 induces the epipolar line $\mathbf{l}_{i2}$ in image 2. When the fundamental matrix is correct, the matching point $\mathbf{x}_{i2}$ will be on this line. Similarly the point $\mathbf{x}_{i2}$ in image 2 induces the epipolar line $\mathbf{l}_{i1}$ in image 1. When the fundamental matrix is correct, the point $\mathbf{x}_{i1}$ will be on this line. The cost function is the sum of the squares of the distances between these epipolar lines and points (green arrows). This is termed *symmetric epipolar distance.*

Estimation of fundamental matrix

When the fundamental matrix is correct, the epipolar line induced by a point in the first image should pass through the matching point in the second image and vice-versa.

This suggests the criterion

$$\hat{\mathbf{F}} = \underset{\mathbf{F}}{\operatorname{argmin}} \left[\sum_{i=1}^{I} \left((\text{dist}[\mathbf{x}_{i1}, \mathbf{l}_{i1}])^2 + (\text{dist}[\mathbf{x}_{i2}, \mathbf{l}_{i2}])^2 \right) \right]$$

If $\mathbf{l} = [a, b, c]$ and $\mathbf{x} = [x, y]^T$ then $\text{dist}[\mathbf{x}, \mathbf{l}] = \dfrac{ax + by + c}{\sqrt{a^2 + b^2}}$

Unfortunately, there is no closed form solution for this quantity.

The 8 point algorithm

Approach:

- solve for fundamental matrix using homogeneous coordinates
- closed form solution (but to wrong problem!)
- Known as the 8 point algorithm

Start with fundamental matrix relation $\tilde{\mathbf{x}}_2^T \mathbf{F} \tilde{\mathbf{x}}_1 = 0$

Writing out in full: $\begin{bmatrix} x_{i2} & y_{i2} & 1 \end{bmatrix} \begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix} \begin{bmatrix} x_{i1} \\ y_{i1} \\ 1 \end{bmatrix}$

or $x_{i2}x_{i1}f_{11}+x_{i2}y_{i1}f_{12}+x_{i2}f_{13}+y_{i2}x_{i1}f_{21}+y_{i2}y_{i1}f_{22}+y_{i2}f_{23}+x_{i1}f_{31}+y_{i1}f_{32}+f_{33}=0.$

The 8 point algorithm

$$x_{i2}x_{i1}f_{11}+x_{i2}y_{i1}f_{12}+x_{i2}f_{13}+y_{i2}x_{i1}f_{21}+y_{i2}y_{i1}f_{22}+y_{i2}f_{23}+x_{i1}f_{31}+y_{i1}f_{32}+f_{33}=0.$$

Can be written as:

$$[x_{i2}x_{i1}, x_{i2}y_{i1}, x_{i2}, y_{i2}x_{i1}, y_{i2}y_{i1}, y_{i2}, x_{i1}, y_{i1}, 1]\mathbf{f} = 0$$

where $\mathbf{f} = [f_{11}, f_{12}, f_{13}, f_{21}, f_{22}, f_{23}, f_{31}, f_{32}, f_{33}]^T$

Stacking together constraints from at least 8 pairs of points, we get the system of equations

$$\mathbf{Af} = \begin{bmatrix} x_{12}x_{11} & x_{12}y_{11} & x_{12} & y_{12}x_{11} & y_{12}y_{11} & y_{12} & x_{11} & y_{11} & 1 \\ x_{22}x_{21} & x_{22}y_{21} & x_{22} & y_{22}x_{21} & y_{22}y_{21} & y_{22} & x_{21} & y_{21} & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{I2}x_{I1} & x_{I2}y_{I1} & x_{I2} & y_{I2}x_{I1} & y_{I2}y_{I1} & y_{I2} & x_{I1} & y_{I1} & 1 \end{bmatrix} \mathbf{f} = \mathbf{0}.$$

The 8 point algorithm

$$\mathbf{Af} = \begin{bmatrix} x_{12}x_{11} & x_{12}y_{11} & x_{12} & y_{12}x_{11} & y_{12}y_{11} & y_{12} & x_{11} & y_{11} & 1 \\ x_{22}x_{21} & x_{22}y_{21} & x_{22} & y_{22}x_{21} & y_{22}y_{21} & y_{22} & x_{21} & y_{21} & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{I2}x_{I1} & x_{I2}y_{I1} & x_{I2} & y_{I2}x_{I1} & y_{I2}y_{I1} & y_{I2} & x_{I1} & y_{I1} & 1 \end{bmatrix} \mathbf{f} = \mathbf{0}.$$

Minimum direction problem of the form $\mathbf{Ab} = \mathbf{0}$,
Find minimum of $|\mathbf{Ab}|^2$ subject to $|\mathbf{b}| = 1$.

To solve, compute the SVD $\mathbf{A} = \mathbf{ULV}^T$ and then set $\hat{\mathbf{b}}$ to the last column of $\mathbf{V}$.

Fitting concerns

- This procedure does not ensure that solution is rank 2. Solution: set last singular value to zero.
- Can be unreliable because of numerical problems to do with the data scaling – better to re-scale the data first
- Needs 8 points in general positions (cannot all be planar).
- Fails if there is not sufficient translation between the views
- Use this solution to start non-linear optimisation of true criterion (must ensure non-linear constraints obeyed).
- There is also a 7 point algorithm (useful if fitting repeatedly in RANSAC)

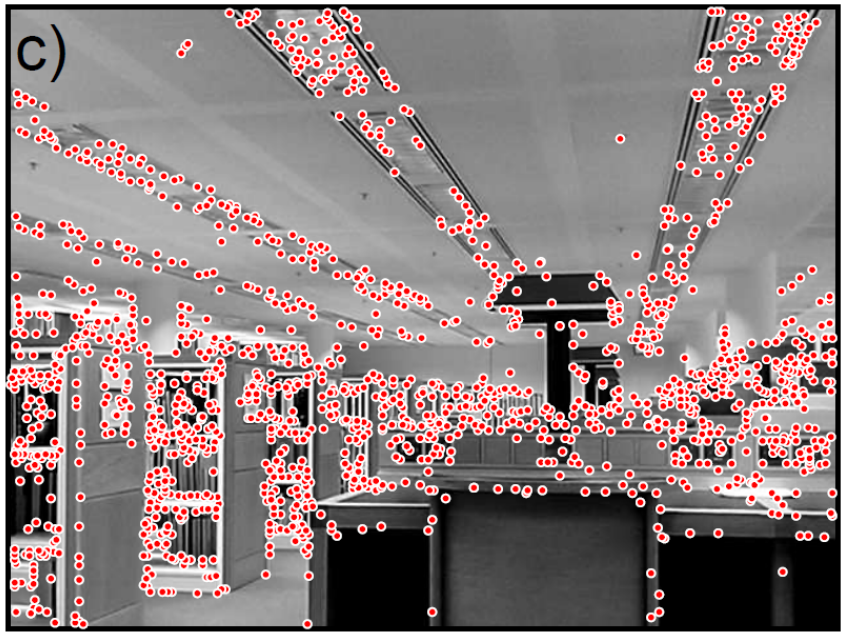
Structure

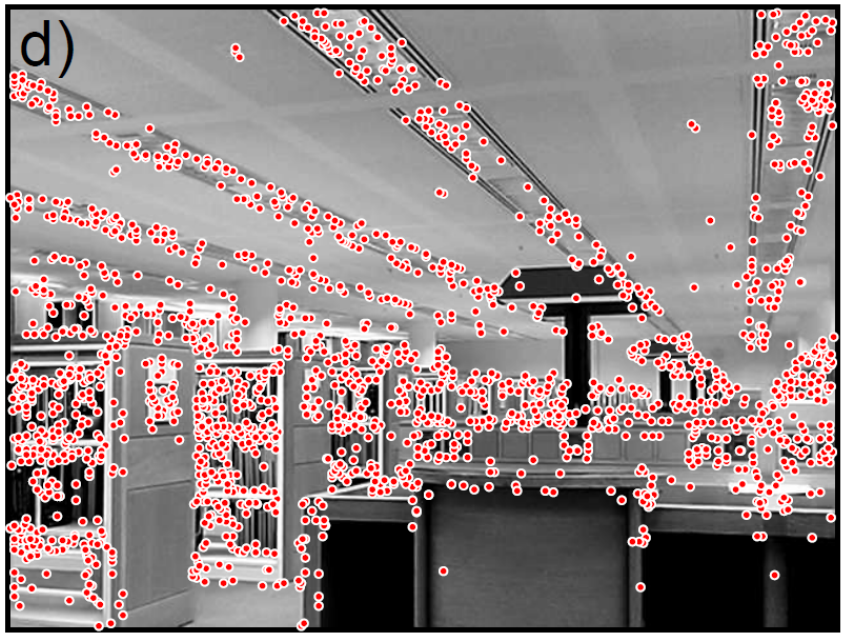
- Two view geometry
- The essential and fundamental matrices
- Reconstruction pipeline
- Rectification
- Multi-view reconstruction
- Applications

Two view reconstruction pipeline

Start with pair of images taken from slightly different viewpoints

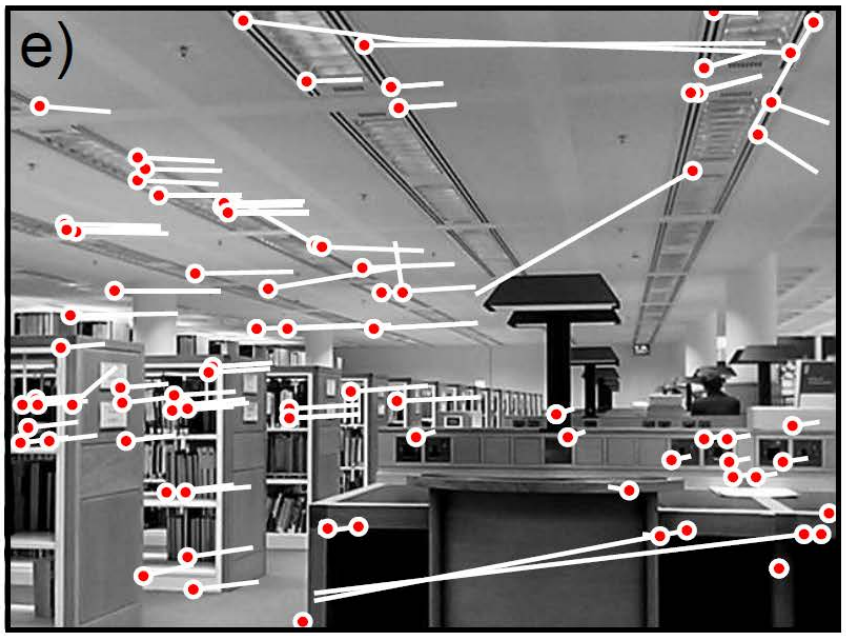
Two view reconstruction pipeline

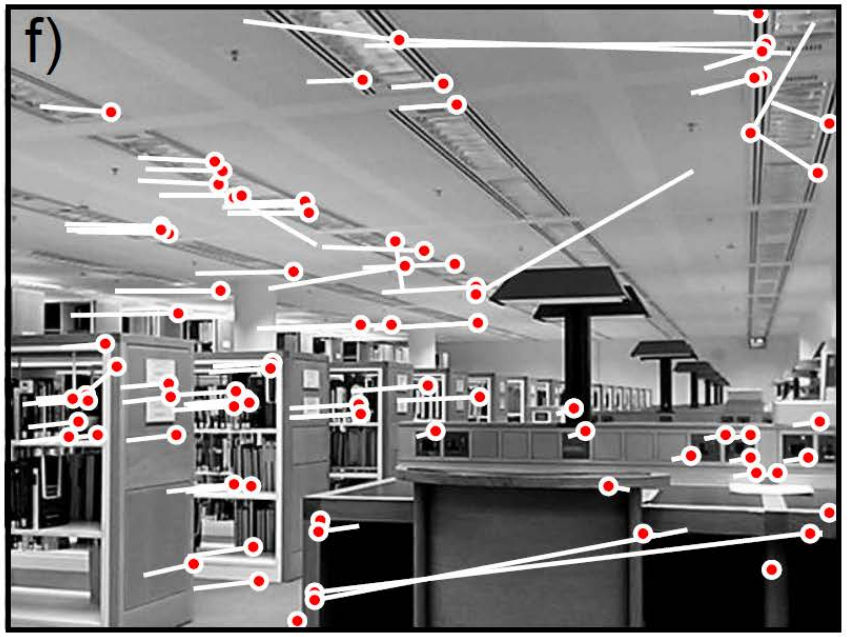




Find features using a corner detection algorithm

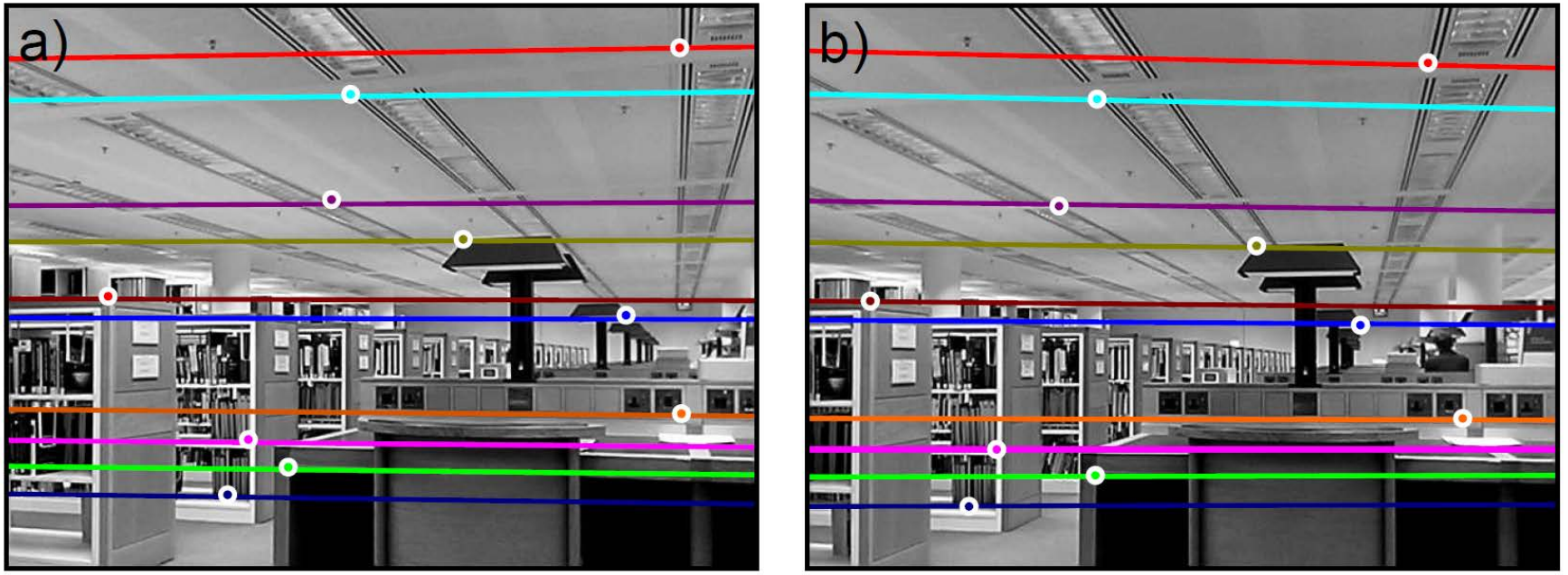
Two view reconstruction pipeline





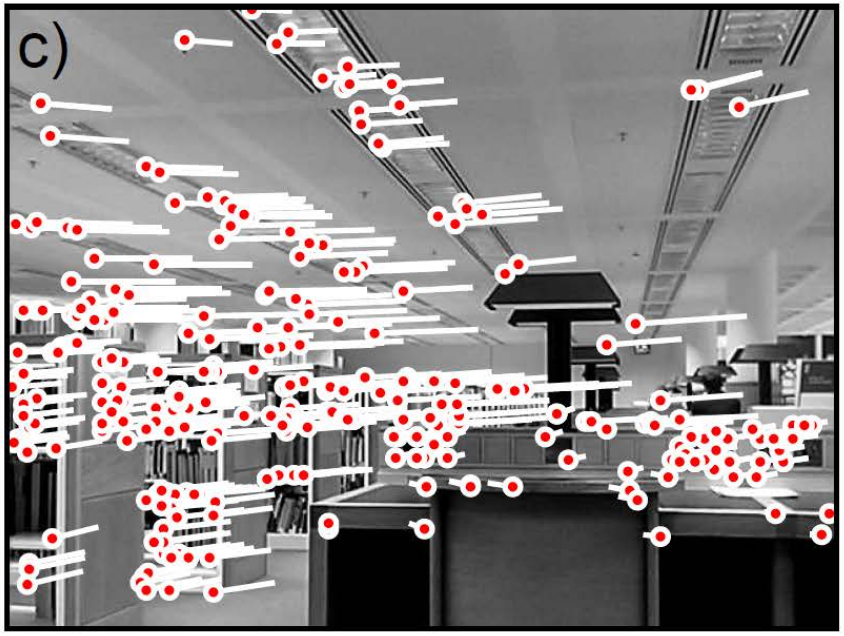
Match features using a greedy algorithm

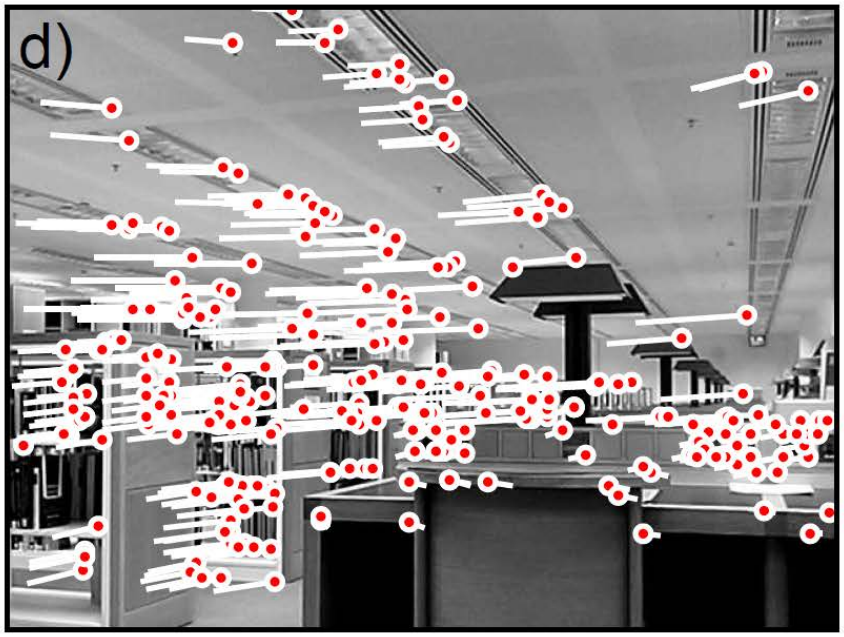
Two view reconstruction pipeline



Fit fundamental matrix using robust algorithm such as RANSAC

Two view reconstruction pipeline





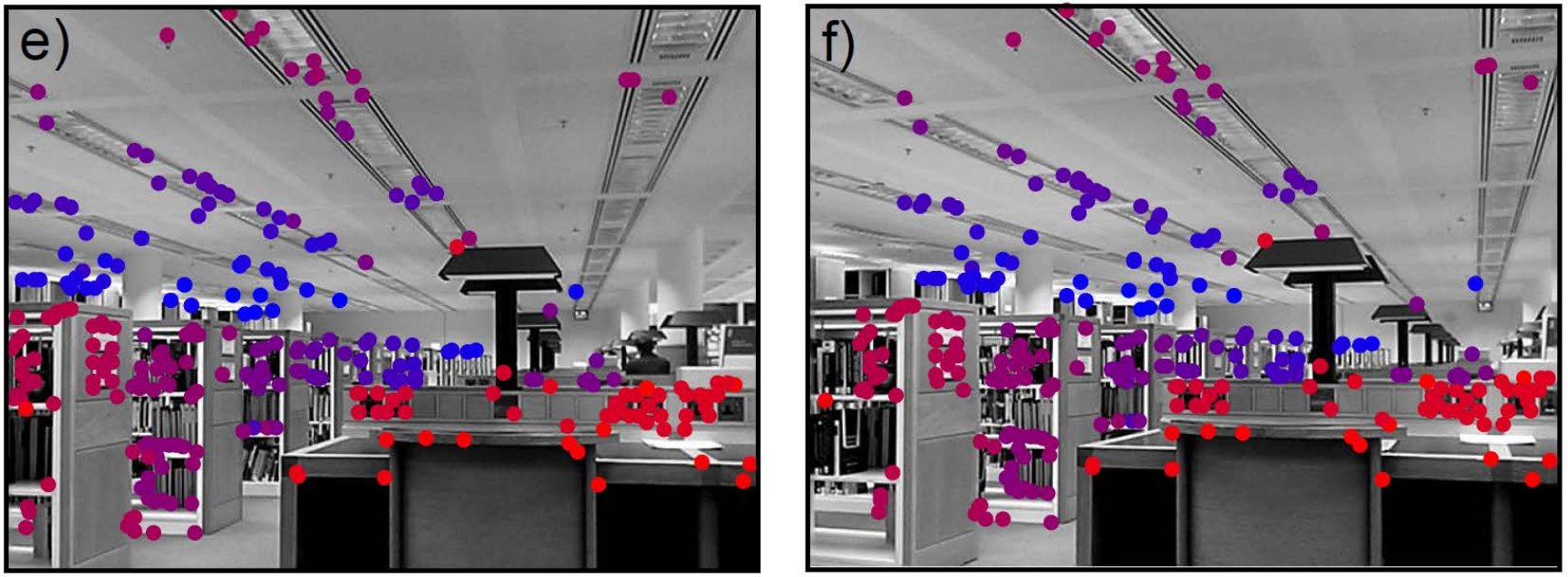
Find matching points that agree with the fundamental matrix

Two view reconstruction pipeline

- Extract essential matrix from fundamental matrix
- Extract rotation and translation from essential matrix
- Reconstruct the 3D positions **w** of points
- Then perform non-linear optimisation over points and rotation and translation between cameras

$$\begin{aligned}
\hat{\mathbf{w}}_{1\ldots I}, \hat{\boldsymbol{\Omega}}, \hat{\boldsymbol{\tau}} &= \underset{\mathbf{w},\boldsymbol{\Omega},\boldsymbol{\tau}}{\operatorname{argmax}} \left[\sum_{i=1}^{I} \sum_{j=1}^{2} \log[Pr(\mathbf{x}_{ij}|\mathbf{w}_i, \boldsymbol{\Lambda}_j, \boldsymbol{\Omega}, \boldsymbol{\tau})] \right] \\
&= \underset{\mathbf{w},\boldsymbol{\Omega},\boldsymbol{\tau}}{\operatorname{argmax}} \left[\sum_{i=1}^{I} \log \left[\mathrm{Norm}_{\mathbf{x}_{i1}}[\mathbf{pinhole}[\mathbf{w}_i, \boldsymbol{\Lambda}_1, \mathbf{I}, \mathbf{0}], \sigma^2\mathbf{I}] \right] \right. \\
&\qquad \left. + \sum_{i=1}^{I} \log \left[\mathrm{Norm}_{\mathbf{x}_{i2}}[\mathbf{pinhole}[\mathbf{w}_i, \boldsymbol{\Lambda}_2, \boldsymbol{\Omega}, \boldsymbol{\tau}], \sigma^2\mathbf{I}] \right] \right]
\end{aligned}$$

Two view reconstruction pipeline



Reconstructed depth indicated by color

Dense Reconstruction

- We'd like to compute a dense depth map (an estimate of the disparity at every pixel)
- Approaches to this include dynamic programming and graph cuts
- However, they all assume that the correct match for each point is on the same horizontal line.
- To ensure this is the case, we warp the images
- This process is known as rectification

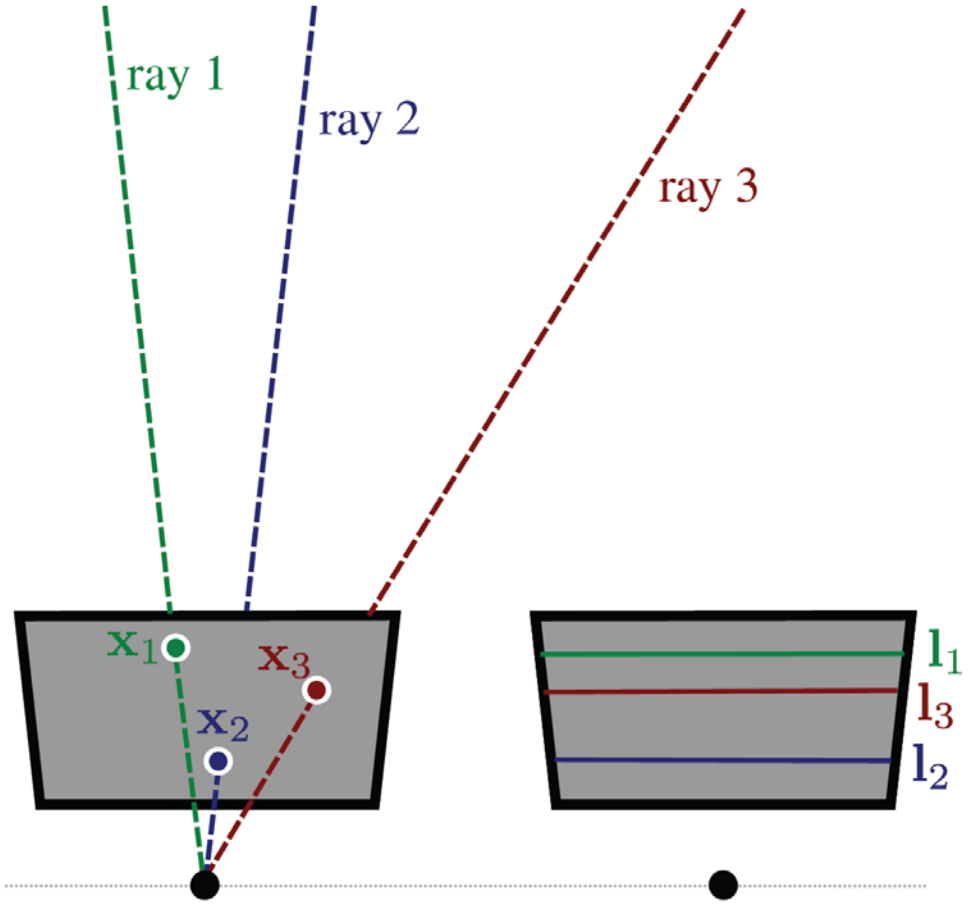
Structure

- Two view geometry
- The essential and fundamental matrices
- Reconstruction pipeline
- Rectification
- Multi-view reconstruction
- Applications

Rectification

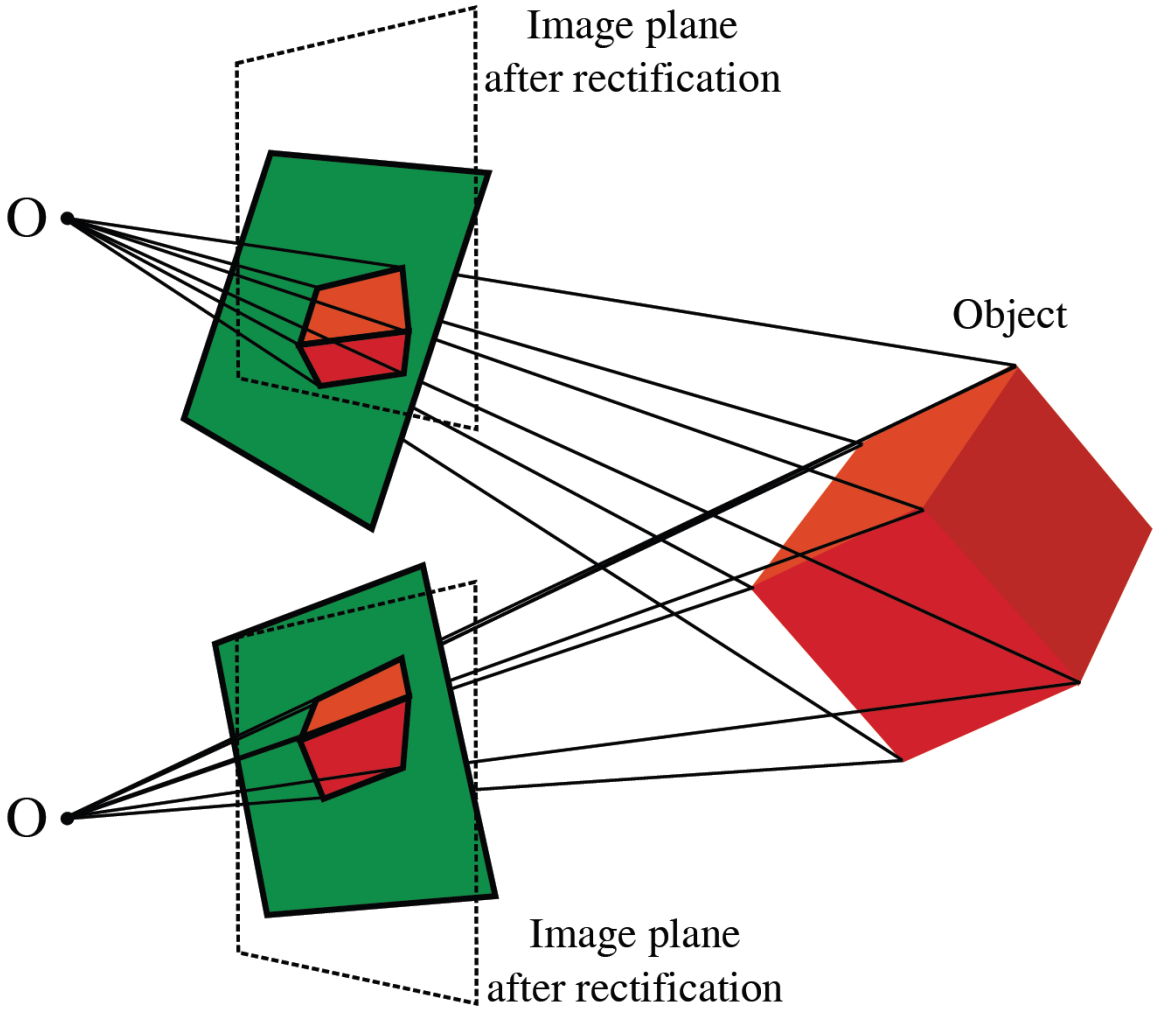
We have already seen one situation where the epipolar lines are horizontal and on the same line:

when the camera movement is pure translation in the u direction.



Planar rectification

Apply homographies $\boldsymbol{\Phi}_1$ and $\boldsymbol{\Phi}_2$ to image 1 and 2



Planar rectification

- Start with $\mathbf{\Phi}_2$ which breaks down as $\mathbf{\Phi}_2 = \mathbf{T}_3\mathbf{T}_2\mathbf{T}_1$
  - Move origin to center of image

$$\mathbf{T}_1 = \begin{bmatrix} 1 & 0 & -\delta_x \\ 0 & 1 & -\delta_y \\ 0 & 0 & 1 \end{bmatrix}$$

  - Rotate epipole to horizontal direction

$$\mathbf{T}_2 = \begin{bmatrix} \cos[-\theta] & -\sin[-\theta] & 0 \\ \sin[-\theta] & \cos[-\theta] & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

  - Move epipole to infinity

$$\mathbf{T}_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -1/e_x & 0 & 1 \end{bmatrix}$$

Planar rectification

- There is a family of possible homographies that can be applied to image 1 to achieve the desired effect

- These can be parameterized as

where

$$\mathbf{F} = \mathbf{SM}$$

$$\boldsymbol{\Phi}_1[\boldsymbol{\alpha}] = (\mathbf{I} + \mathbf{e}_2\boldsymbol{\alpha}^T)\boldsymbol{\Phi}_2\mathbf{M}$$

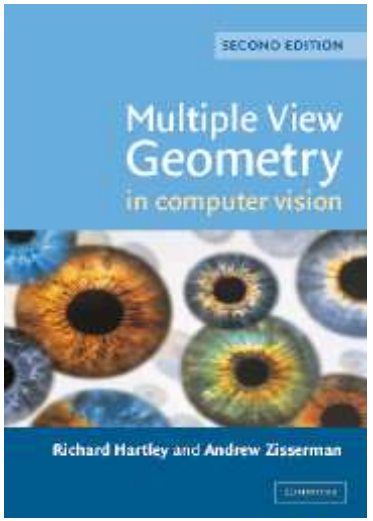
- One way to choose this, is to pick the parameter that makes the mapped points in each transformed image closest in a least squares sense:

$$\underset{\boldsymbol{\alpha}}{\text{argmax}}\left[\sum_{i=1}^{I}(\mathbf{hom}[\mathbf{x}_{i1},\boldsymbol{\Phi}_1[\boldsymbol{\alpha}]]-\mathbf{hom}[\mathbf{x}_{i2},\boldsymbol{\Phi}_2])^T(\mathbf{hom}[\mathbf{x}_{i1},\boldsymbol{\Phi}_1[\boldsymbol{\alpha}]]-\mathbf{hom}[\mathbf{x}_{i2},\boldsymbol{\Phi}_2])\right].$$

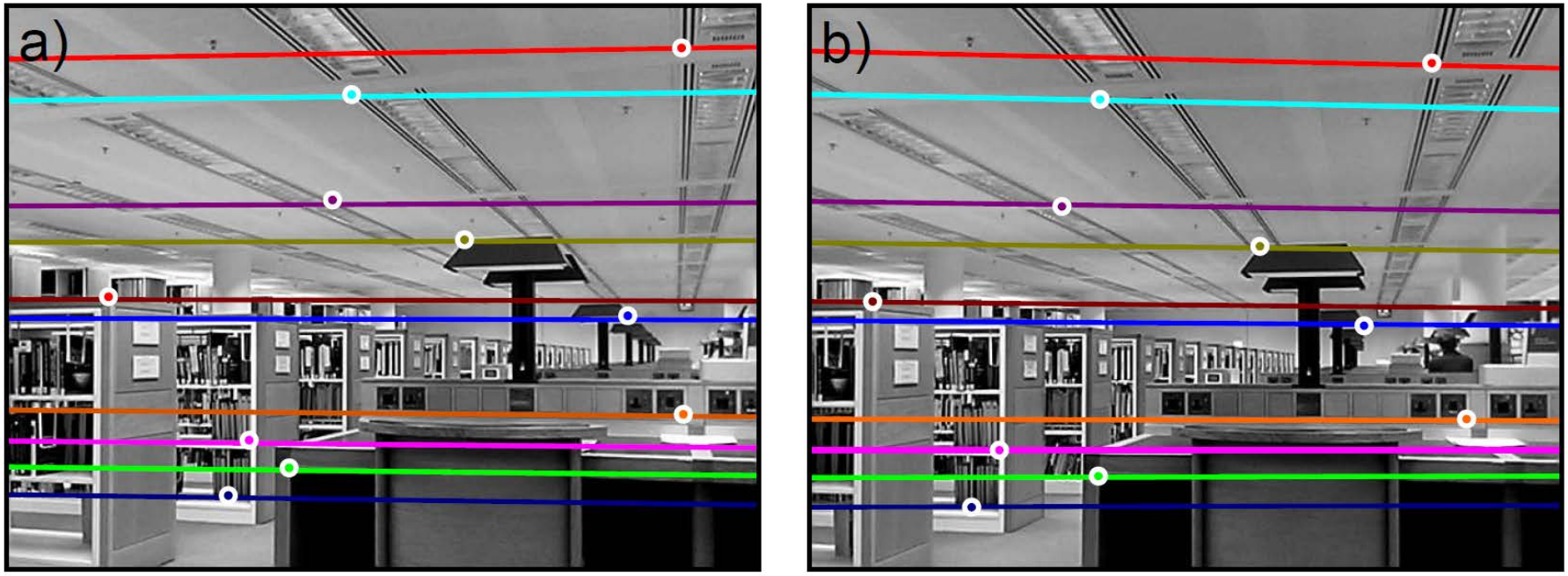
Now we consider the first image. We cannot simply apply the same procedure as this will not guarantee that the epipolar lines in the first image will be aligned with those in the second. It transpires however, that there is a family of possible transformations that do make the epipolar lines of this image horizontal and aligned with those in the second image. This family can (not obviously) be parameterized as

$$\mathbf{\Phi}_1[\boldsymbol{\alpha}] = (\mathbf{I} + \mathbf{e}_2\boldsymbol{\alpha}^T)\mathbf{\Phi}_2\mathbf{M}, \tag{16.34}$$

where $\mathbf{e}_2 = [1, 0, 0]^T$ is the transformed epipole in the second image, and $\boldsymbol{\alpha} = [\alpha_1, \alpha_2, \alpha_3]^T$ is a 3D vector that selects the particular transformation from the family. The matrix $\mathbf{M}$ comes from the decomposition of the fundamental matrix into $\mathbf{F} = \mathbf{SM}$, where $\mathbf{S}$ is skew symmetric (see below). A proof of the relation in equation 16.34 can be found in Hartley & Zisserman (2004).

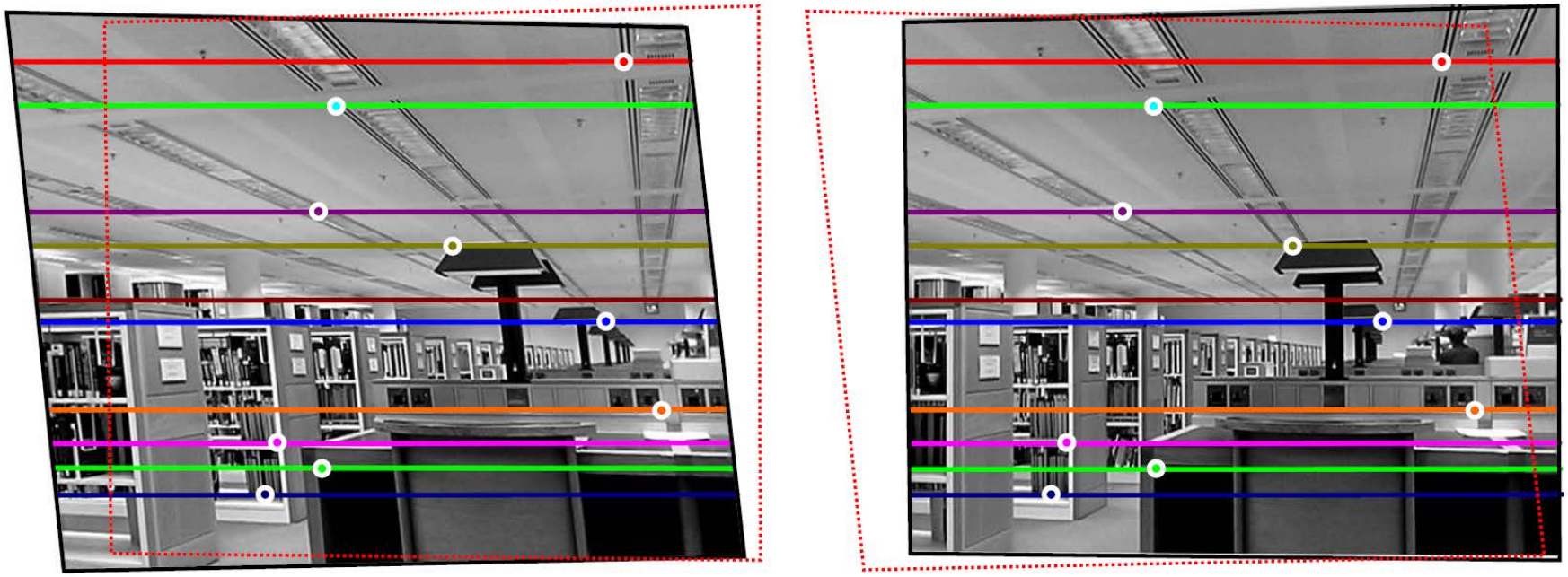


Before rectification



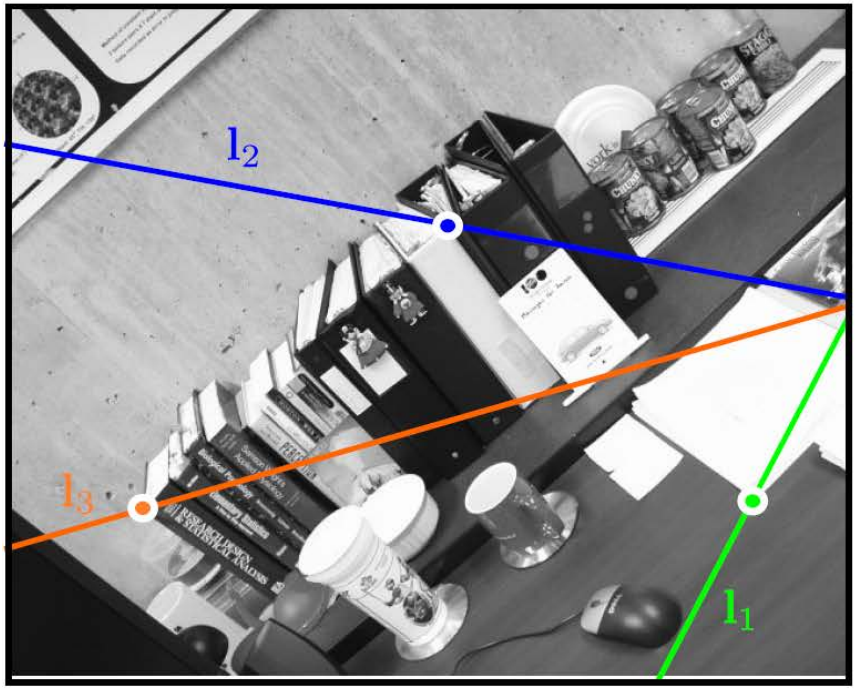
Before rectification, the epipolar lines converge

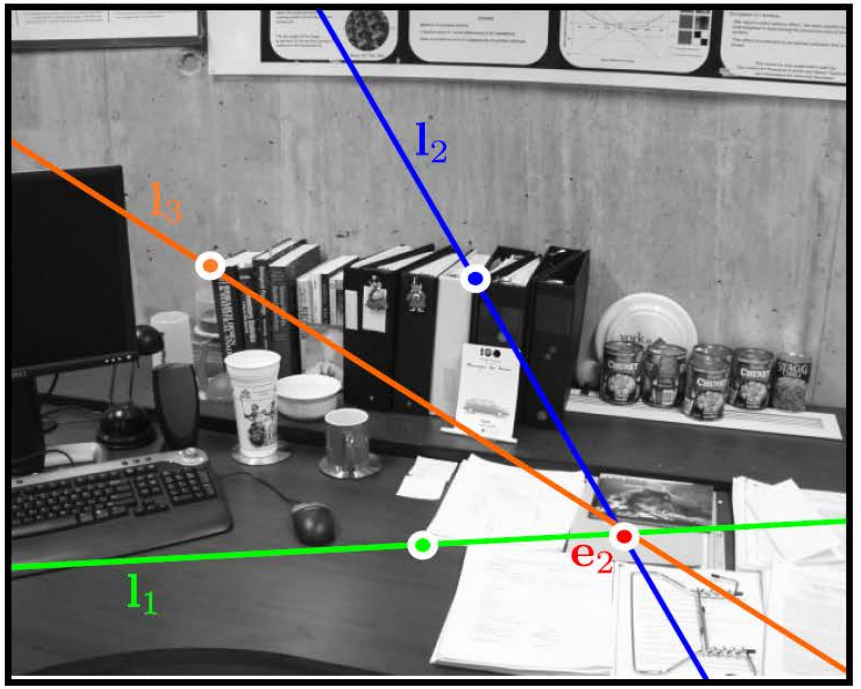
After rectification



After rectification, the epipolar lines are horizontal and aligned with one another

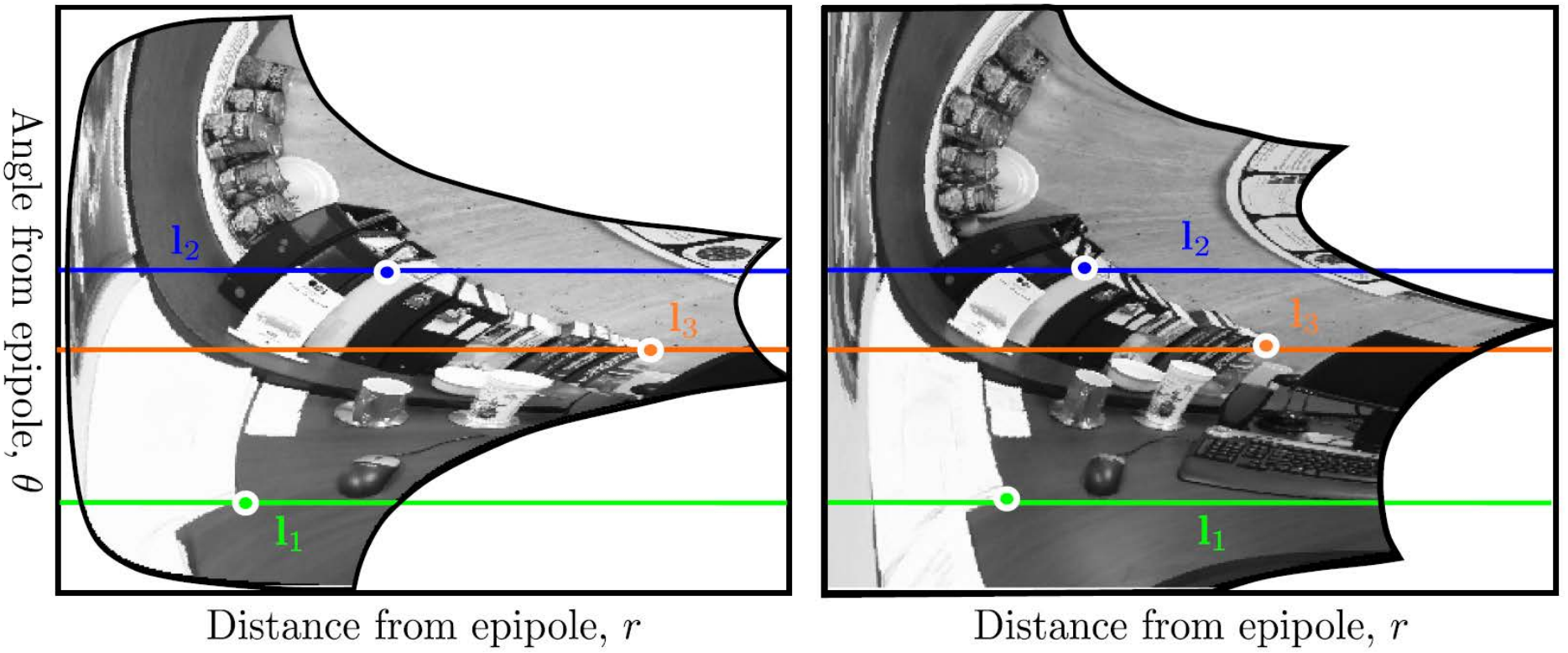
Polar rectification





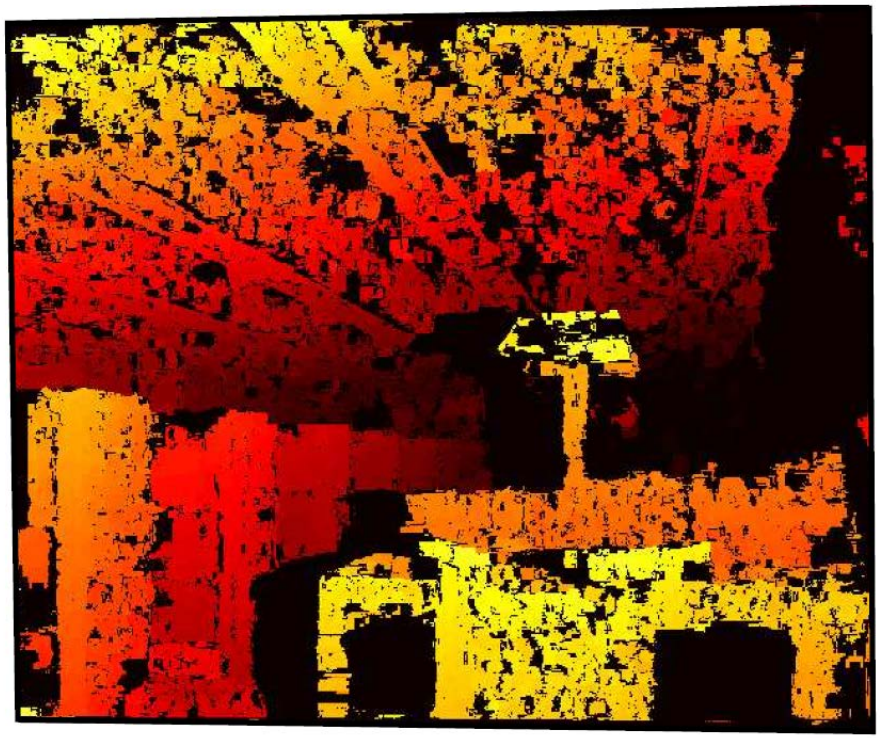
Planar rectification does not work if epipole lies within the image.

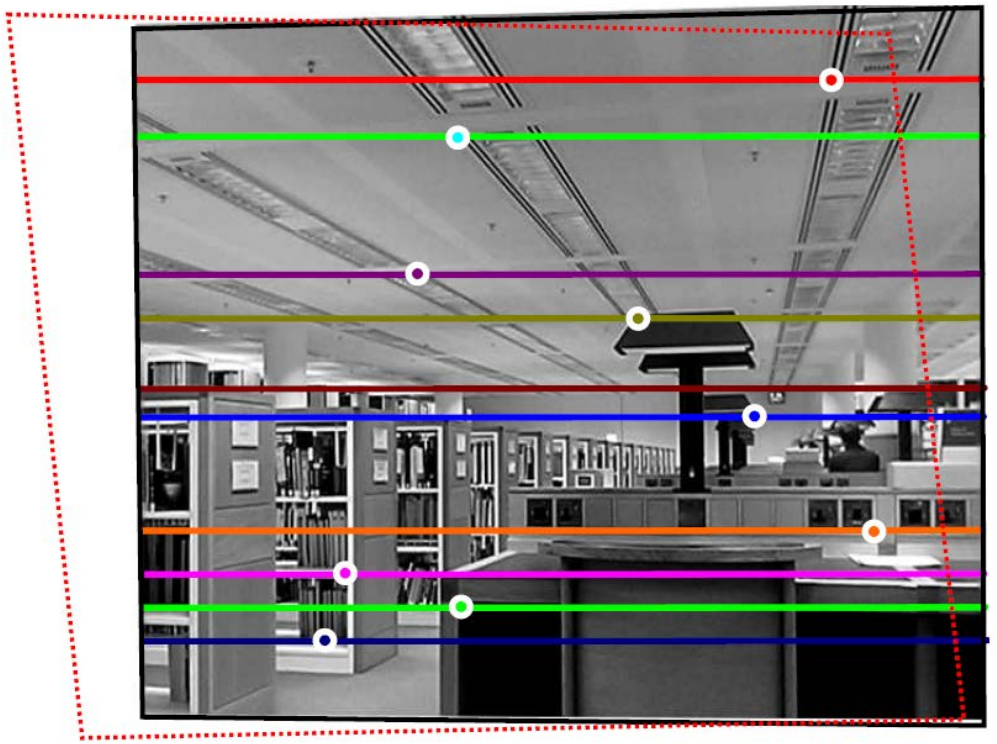
Polar rectification



Polar rectification works in this situation, but distorts the image more

Dense Stereo

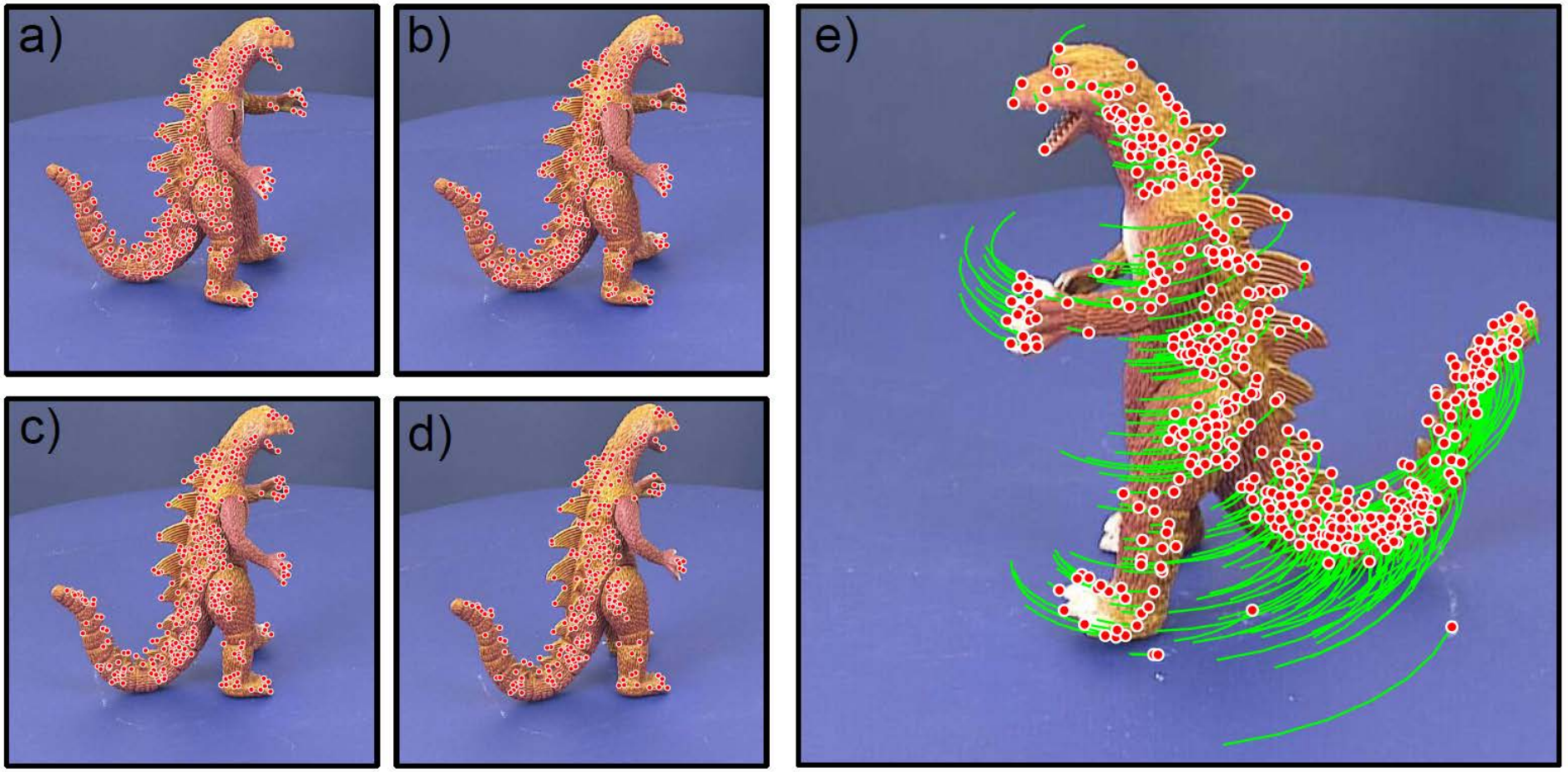




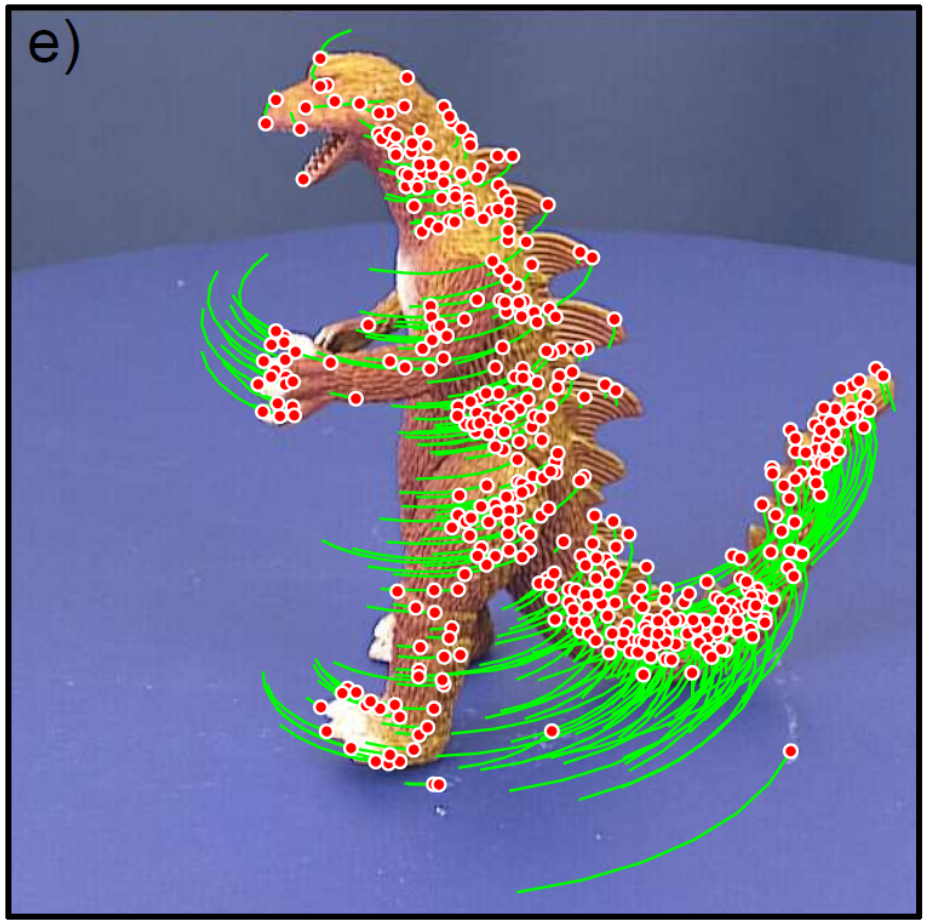
Structure

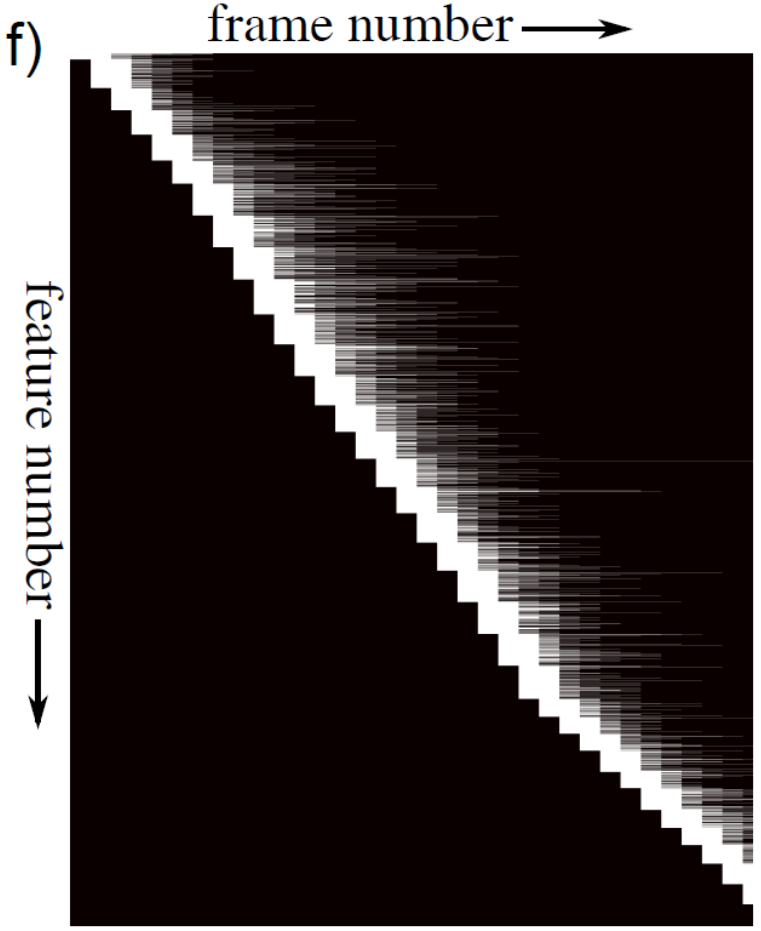
- Two view geometry
- The essential and fundamental matrices
- Reconstruction pipeline
- Rectification
- Multi-view reconstruction
- Applications

Multi-view reconstruction



Multi-view reconstruction





Reconstruction from video

1. Images taken from same camera; can also optimise for intrinsic parameters (auto-calibration)

2. Matching points is easier as can track them through the video

3. Not every point is within every image

4. Additional constraints on matching: three-view equivalent of fundamental matrix is tri-focal tensor

5. New ways of initialising all of the camera parameters simultaneously (factorisation algorithm)

Bundle Adjustment

Bundle adjustment refers to process of refining initial estimates of structure and motion using non-linear optimisation.

$$\begin{aligned}\hat{\boldsymbol{\theta}} &= \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \left[\sum_{i=1}^{I}\sum_{j=1}^{J} \log[Pr(\mathbf{x}_{ij}|\mathbf{w}_i, \boldsymbol{\Lambda}, \boldsymbol{\Omega}_j, \boldsymbol{\tau}_j)]\right] \\ &= \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \left[\sum_{i=1}^{I}\sum_{j=1}^{J} \log\left[\mathrm{Norm}_{\mathbf{x}_{ij}}[\mathbf{pinhole}[\mathbf{w}_i, \boldsymbol{\Lambda}, \boldsymbol{\Omega}_j, \boldsymbol{\tau}_j], \sigma^2\mathbf{I}]\right]\right]\end{aligned}$$

This problem has the least squares form:

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmin}}\, \mathbf{z}^T\mathbf{z}$$

where:

$$\mathbf{z} = \begin{bmatrix} \mathbf{x}_{11} - \mathbf{pinhole}[\mathbf{w}_1, \boldsymbol{\Lambda}, \boldsymbol{\Omega}_1, \boldsymbol{\tau}_1] \\ \mathbf{x}_{12} - \mathbf{pinhole}[\mathbf{w}_1, \boldsymbol{\Lambda}, \boldsymbol{\Omega}_2, \boldsymbol{\tau}_2] \\ \vdots \\ \mathbf{x}_{IJ} - \mathbf{pinhole}[\mathbf{w}_I, \boldsymbol{\Lambda}, \boldsymbol{\Omega}_J, \boldsymbol{\tau}_J] \end{bmatrix}$$

Bundle Adjustment

This type of least squares problem is suited to optimisation techniques such as the Gauss-Newton method:

$$\boldsymbol{\theta}^{[t]} = \boldsymbol{\theta}^{[t-1]} + \lambda(\mathbf{J}^T\mathbf{J})^{-1}\frac{\partial f}{\partial \boldsymbol{\theta}}$$

Where

$$J_{mn} = \frac{\partial z_m}{\partial \theta_n}$$

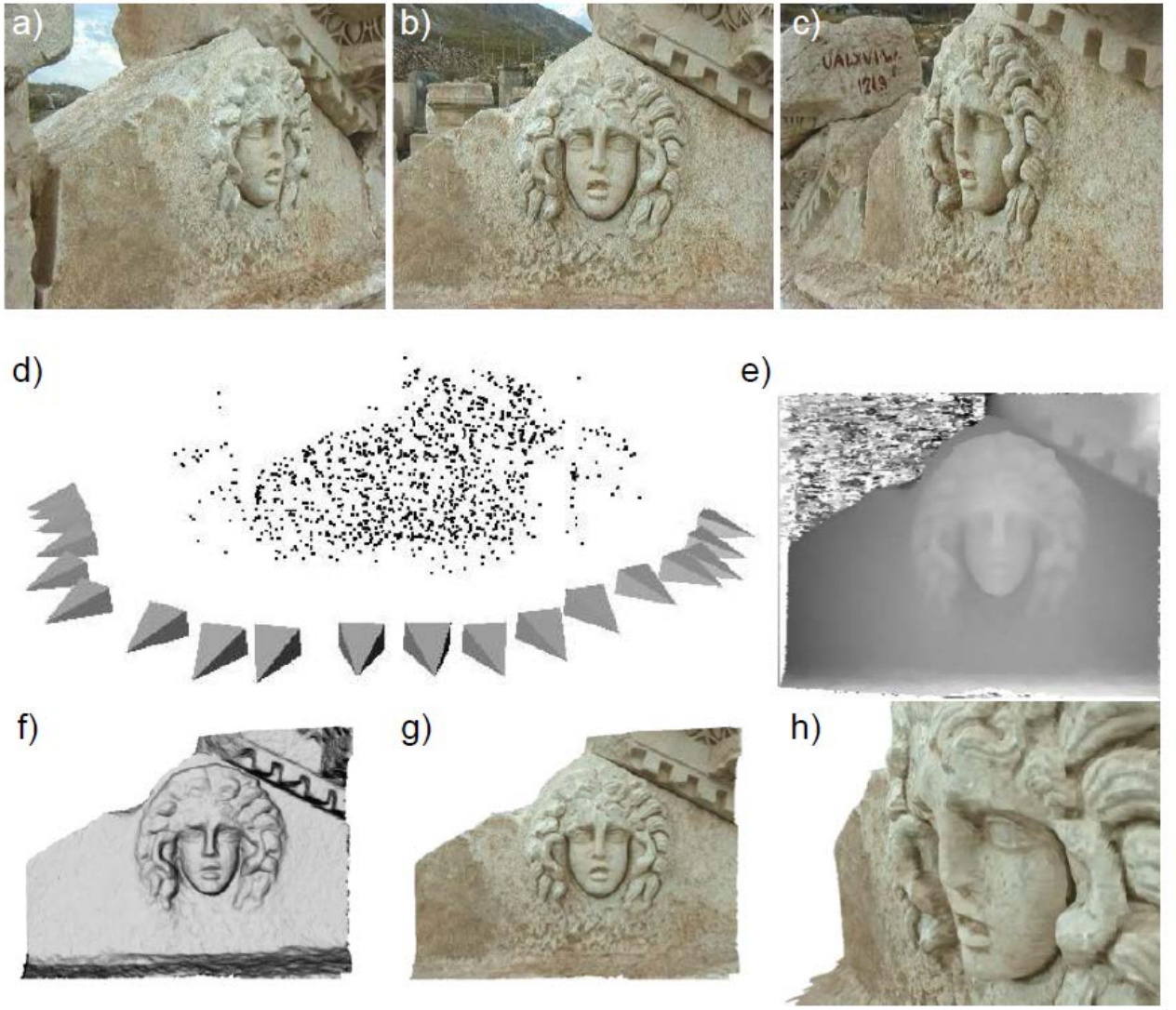
The bulk of the work is inverting $\mathbf{J}^{\mathrm{T}}\mathbf{J}$. To do this efficiently, we must exploit the structure within the matrix.

$$\mathbf{J}^T\mathbf{J} = \begin{bmatrix} \mathbf{J}_{\mathbf{w}}^T\mathbf{J}_{\mathbf{w}} & \mathbf{J}_{\mathbf{w}}^T\mathbf{J}_{\boldsymbol{\Omega}} \\ \mathbf{J}_{\boldsymbol{\Omega}}^T\mathbf{J}_{\mathbf{w}} & \mathbf{J}_{\boldsymbol{\Omega}}^T\mathbf{J}_{\boldsymbol{\Omega}} \end{bmatrix}$$

Structure

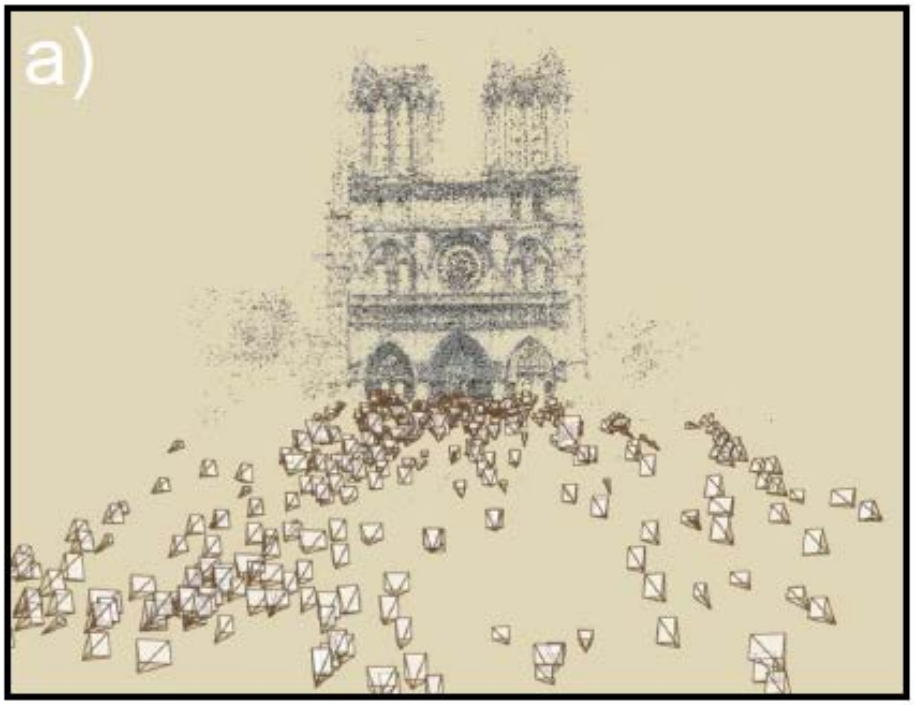
- Two view geometry
- The essential and fundamental matrices
- Reconstruction pipeline
- Rectification
- Multi-view reconstruction
- Applications

3D reconstruction pipeline



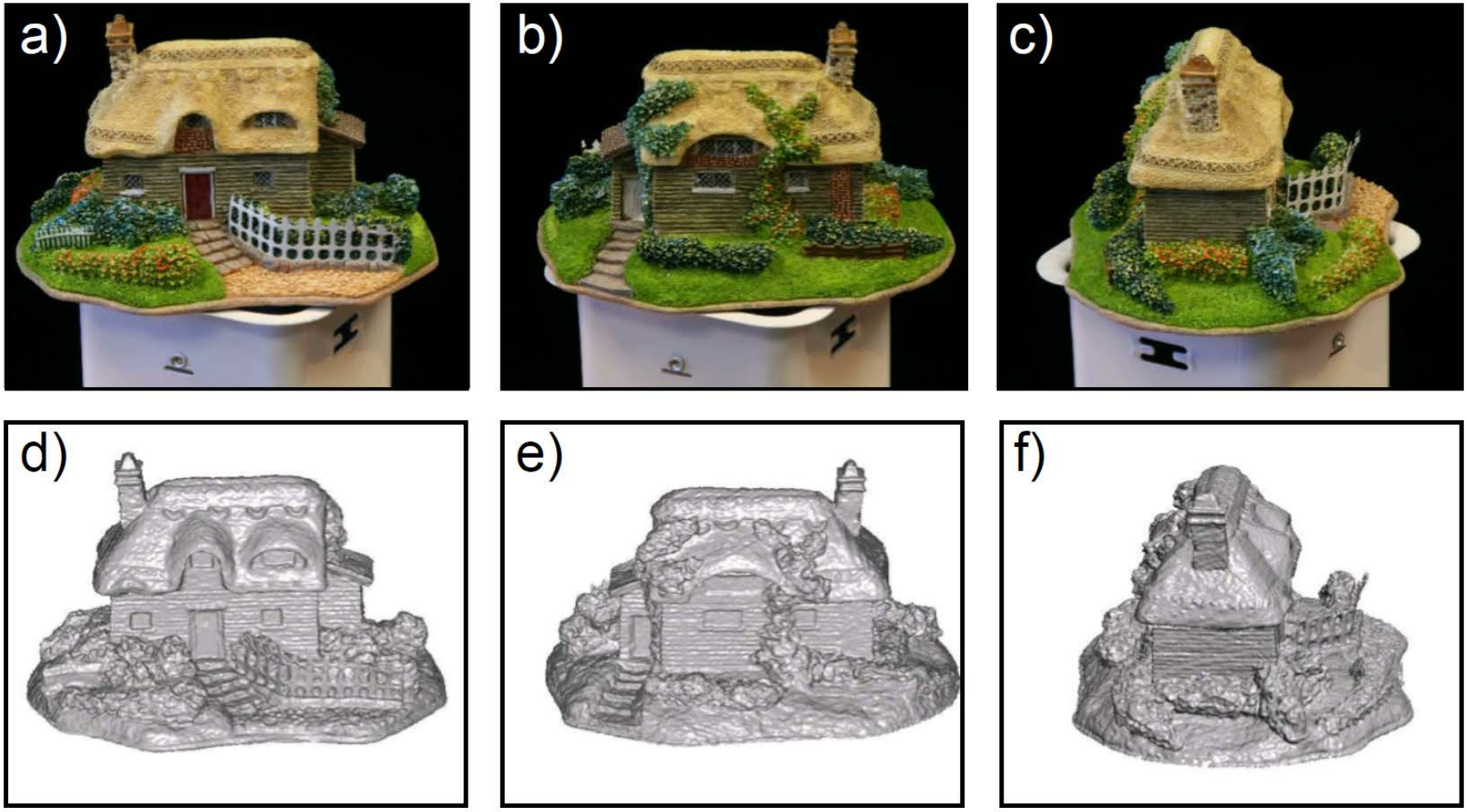
Pollefeys & Van Gool (2002)

Photo-Tourism



Snavely *et al.* (2006)

Volumetric graph cuts



Adapted from Vogiatzis *et al.* (2007)

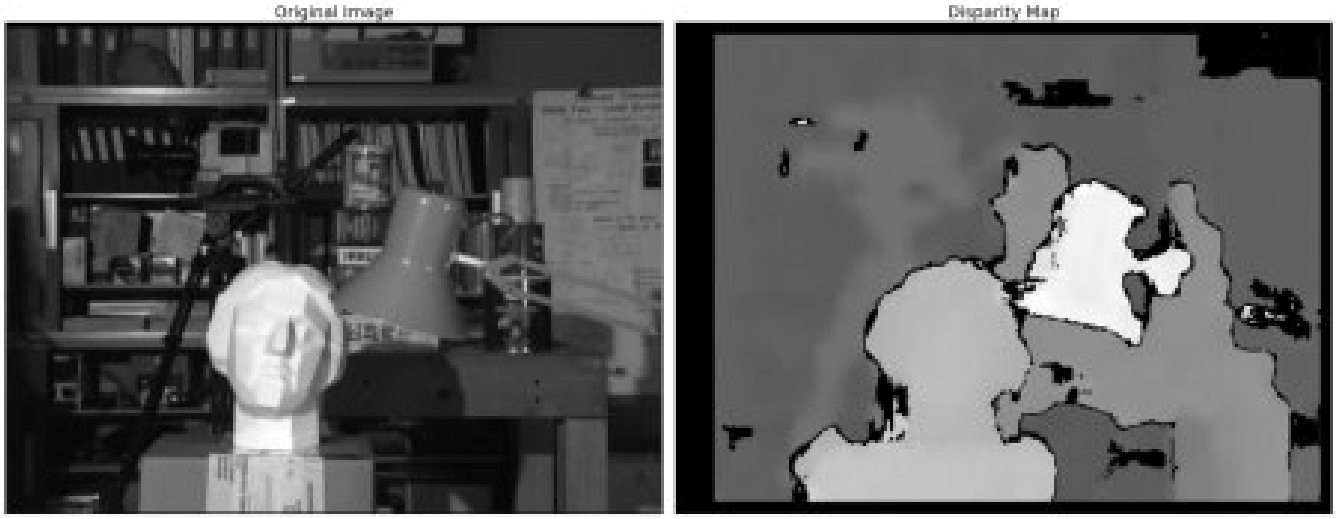
Conclusions

- Given a set of a photos of the same rigid object, it is possible to build an accurate 3D model of the object and reconstruct the camera positions
- Ultimately relies on a large-scale non-linear optimisation procedure.
- Works if optical properties of the object are simple (no specular reflectance etc.)

https://docs.opencv.org/3.4.2/dd/d53/tutorial_py_depthmap.html

Stereo matching – simple example

Download and execute tsukuba.py & data/tsukuba_{l|r}.png




```
import numpy as np
import cv2 as cv
from matplotlib import pyplot as plt

imgL = cv.imread('tsukuba_l.png',0)
imgR = cv.imread('tsukuba_r.png',0)

stereo = cv.StereoBM_create(numDisparities=16, blockSize=15)
disparity = stereo.compute(imgL,imgR)
plt.imshow(disparity,'gray')
plt.show()
```

Reporting Assessment 1 – 50pts

Calibrate stereo cameras with chessboard images, and estimate a disparity map of rabbit images. The instructor selects your target images. Summarize your used functions and their explanations, and resulting images in your report. Also submit all the code for your implementation.

(1) Show the undistorted rabbit images. (15pts)

(2) Show the rectified rabbit images. (15pts)

(3) Estimate disparity by local and global matching techniques. (20pts)

Rectification

https://docs.opencv.org/3.4.2/d9/db7/tutorial_py_table_of_contents_calib3d.html

1. Estimate intrinsic matrices & distorted parameters + extrinsic parameters of two cameras:

https://docs.opencv.org/3.4.2/dc/dbb/tutorial_py_calibration.html

```
retval, cameraMatrix1, distCoeffs1, cameraMatrix2,
distCoeffs2, R, T, E, F =
cv2.stereoCalibrate(obj_points, img_points1,
img_points2, cameraMatrix1, distCoeffs1,
cameraMatrix2, distCoeffs2, imageSize)
```

R,T: Extrinsic parameters (1->2)

cameraMatrixN: Intrinsic paremeters

distCoeffsN: distortion parameters

Rectification

https://docs.opencv.org/3.4.2/d9/db7/tutorial_py_table_of_contents_calib3d.html

2. Estimate Rectification matrix for two cameras:

```
R1, R2, P1, P2, Q, validPixROI1, validPixROI2 =
cv2.stereoRectify(cameraMatrix1, distCoeffs1,
cameraMatrix2, distCoeffs2, imageSize, R, T,
flags, alpha, newimageSize)
```

https://docs.opencv.org/3.4.2/d9/d0c/group__calib3d.html#ga617b1685d4059c6040827800e72ad2b6

flags=0, alpha=1, newimageSize=(h,w) for example

3. Calculate rectification maps:

```
map1_l, map2_l =
cv2.initUndistortRectifyMap(cameraMatrix1,
distCoeffs1, R1, P1, newimageSize, m1type)
```

4. Show remapped images:

```
Re_TgtImg_l = cv2.remap(TgtImg_l, map1_l, map2_l,
interpolation)
```

Stereo matching

https://docs.opencv.org/3.4.2/d9/db7/tutorial_py_table_of_contents_calib3d.html

```
stereo = cv2.StereoSGBM_create(
    minDisparity = 32,
    numDisparities = 112 - min_disp,
    blockSize = 5,
    P1 = 8*3*window_size**2,
    P2 = 32*3*window_size**2,
    disp12MaxDiff = 1,
    uniquenessRatio = 10,
    speckleWindowSize = 100,
    speckleRange = 32,
    mode = cv2.STEREO_SGBM_MODE_SGBM_3WAY
)
disp = stereo.compute(Re_TgtImg_l,
Re_TgtImg_r).astype(numpy.float32) / 16.0
```

For instance:

StereoBM(block matching): local

StereoSGBM(semi-global matching): (semi-)global