EE6221 Robotics and Intelligent Sensors (Part 3)

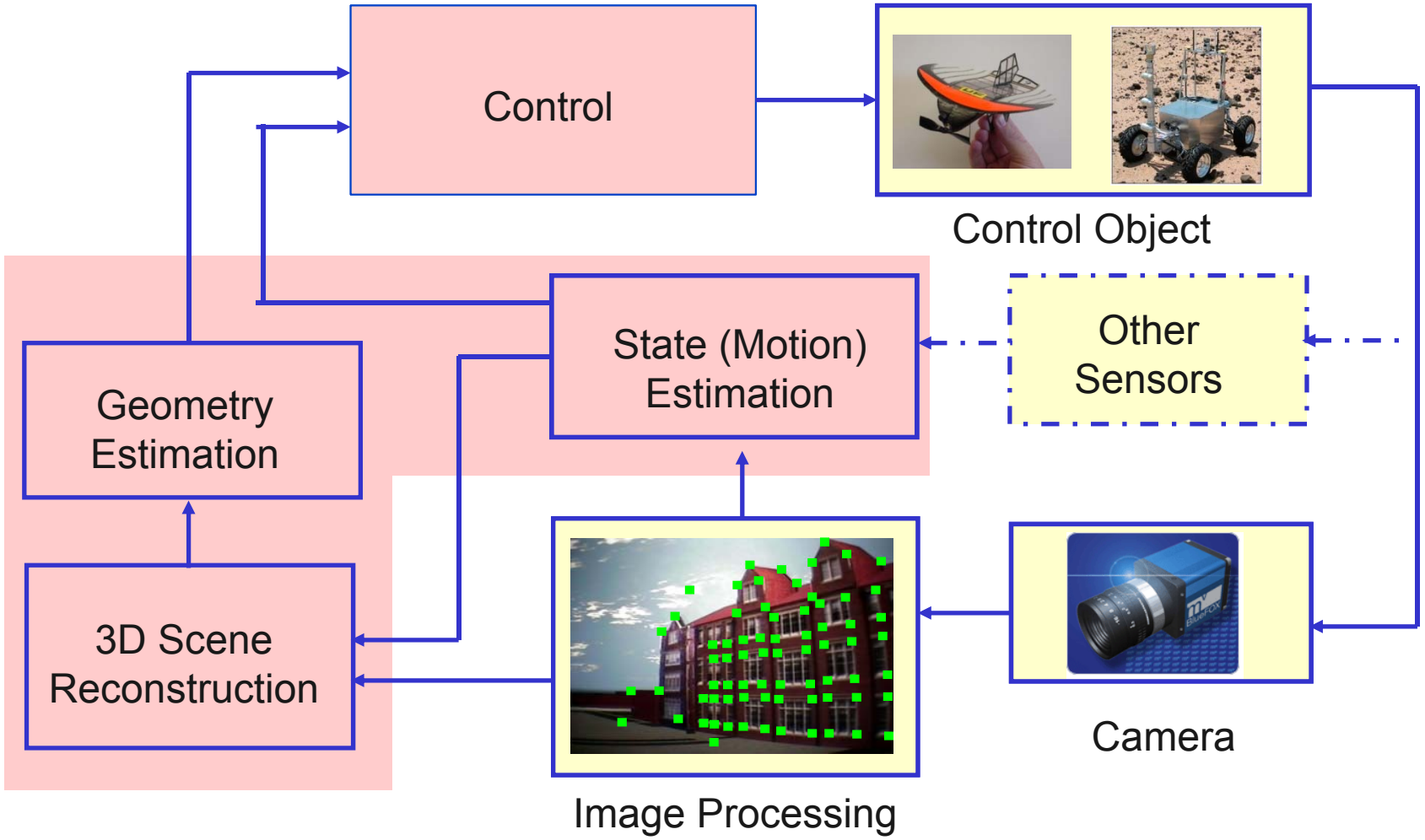
Lecture 3: Structure and Pose Estimation

Guoqiang Hu
School of EEE, NTU

Outline

- Structure from Motion
- Pose reconstruction
- Essential matrix and 8-point algorithm
- Homography matrix and 4-point algorithm
- Stereo vision
- Optical flow

Vision-Based Control & Estimation



Structure From Motion - Introduction

- Given two images of matched points, it is possible to determine:
  - The rotation and translation that separates these two camera views
  - The relative structure of the feature points being viewed
- We assume feature points have already been identified and matched between two views through feature point detection and tracking methods
- Commonly referred to as “Structure From Motion (SFM)”
- Different situations/scenes will require different methods of reconstruction
- The methods presented here are linear, in that they only require methods of linear algebra

Reference Books

Reference books for structure from motion:

- Richard Hartley and Andrew Zisserman (2003). *Multiple View Geometry in Computer Vision*. Cambridge University Press.
- Olivier Faugeras and Quang-Tuan Luong and Theodore Papadopoulo (2001). *The Geometry of Multiple Images*. MIT Press.
- Yi Ma, S. Shankar Sastry, Jana Kosecka, Stefano Soatto, Jana Kosecka (November 2003). *An Invitation to 3-D Vision: From Images to Geometric Models*. Interdisciplinary Applied Mathematics Series, #26. Springer-Verlag New York, LLC.

Free Software Resources

Free software resources for structure from motion:

- Intel Open Source Computer Vision Library for C++
  - http://www.intel.com/technology/computing/opencv/
- GNU Scientific Library (Linear Algebra in C++)
  - http://www.gnu.org/software/gsl/ (Unix/Linux)
  - http://gnuwin32.sourceforge.net/packages/gsl.htm (Windows)
- Machine Vision Toolbox for Matlab
  - http://www.petercorke.com/Machine_Vision_Toolbox.html
- Robotics Toolbox for Matlab
  - http://www.petercorke.com/Machine%20Robotics%20Toolbox.html

SFM-History

- Euclid developed notions of perspective and projection in 3rd century B.C., Greek artwork reflects some understanding
- Perspective projection “rediscovered” during the Renaissance.
- In 1913 Kruppa proved that two camera views of five Euclidean points could be used to estimate the translation and rotation separating the two camera views.
- Longuet-Higgins introduced the Essential matrix and eight-point algorithm in 1981 to solve for the translation and rotation given eight noncoplanar points.
- O. Faugeras and F. Lustman introduced the Homography matrix and four-point algorithm in 1988 to solve for the translation and rotation given four coplanar points
- Philip in 1996 and Nister in 2004 introduced the 5-point algorithms, which involves solving polynomials and has 10 possible solutions

Linear Algebra Background

- **Singular Value Decomposition (SVD)** – Any matrix can be decomposed into three matrices such that

$$\underset{m \times n}{M} = \underset{m \times m}{U} \; \underset{m \times n}{\Sigma} \; \underset{n \times n}{V} \qquad \Sigma = diag\{\sigma_1,\ldots,\sigma_m\}$$

  - $\sigma_1,\ldots,\sigma_m$ are "singular values" and are similar to eigenvalues. If M is not full rank, some singular values will be 0.
  - For a non-full-rank M, the column of V corresponding to a 0 singular value is in the null space of M.
- **Skew-symmetric matrix** can be defined for any 3 element vector

$$a = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} \Rightarrow [a]_\times = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix}.$$

  - $[a]_x$ has the property that $[a]_x b = a \times b$

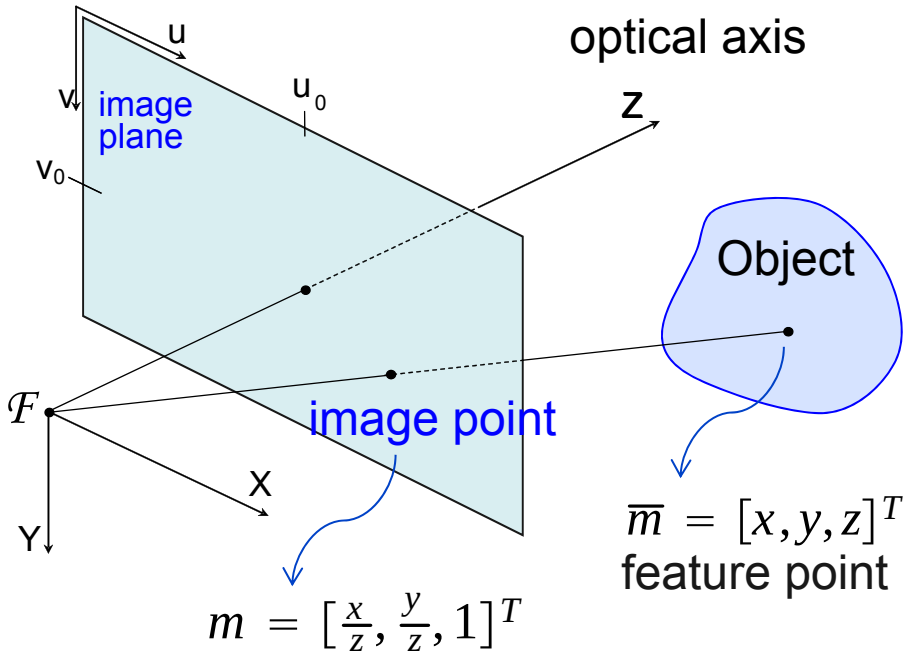
Pinhole Camera Model

- 3D feature point with Euclidean coordinates in camera frame $\mathcal{F}$

$$\overline{m} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

- Projected to image plane with normalized coordinates in $\mathcal{F}$

$$m = \begin{bmatrix} m_x \\ m_y \\ 1 \end{bmatrix} = \frac{\overline{m}}{z} = \begin{bmatrix} x/z \\ y/z \\ 1 \end{bmatrix}$$



- Mapped to pixel coordinates by camera calibration matrix A

$$p = \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = Am = \begin{bmatrix} f\sigma_x & -f\sigma_x \tan\alpha & u_0 \\ 0 & f\sigma_y \sec\alpha & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} m_x \\ m_y \\ 1 \end{bmatrix}$$

A: camera calibration matrix

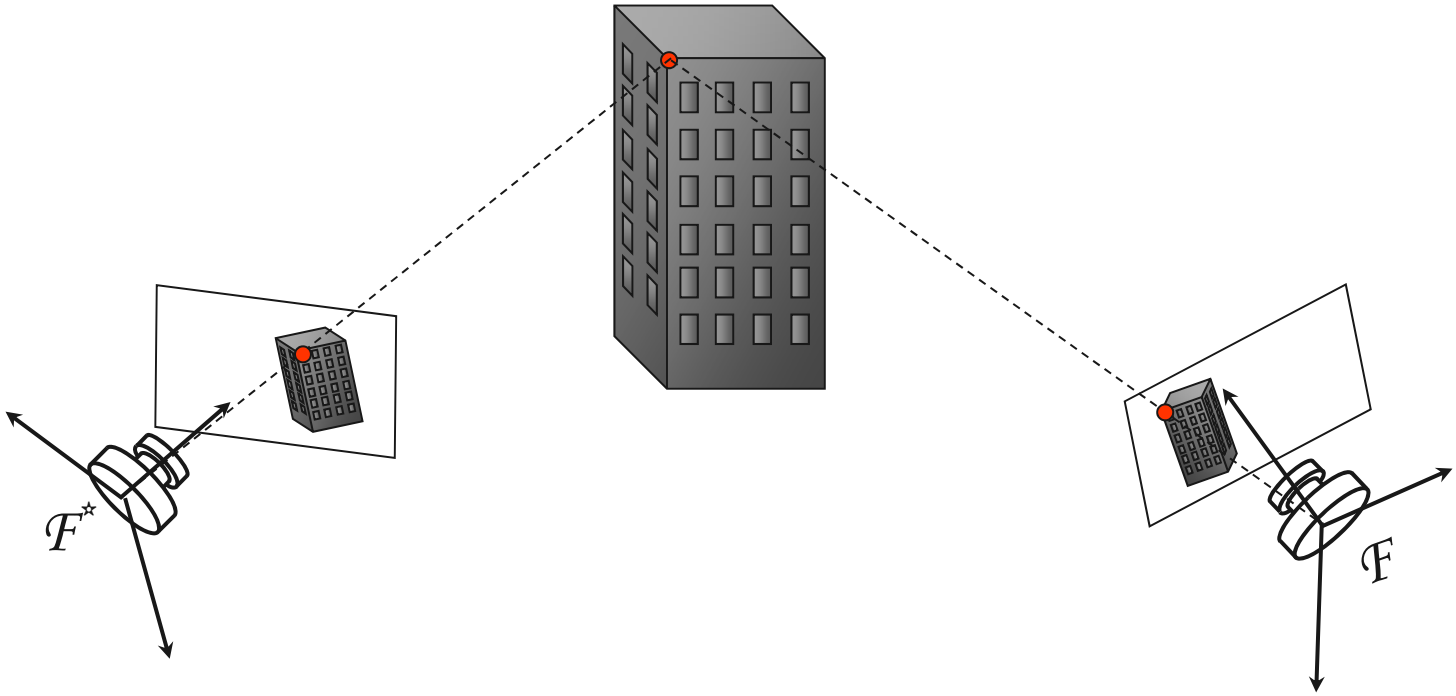
Pinhole Camera Model (Remarks)

- Many estimation and control schemes require normalized coordinates for points
- Given pixel coordinates p of a point in the image and knowledge of calibration matrix A, recover normalized coordinates

$$m = A^{-1} p$$

- There is no way to recover the Euclidean coordinates without additional information. Depth ambiguity is a consequence of imaging.

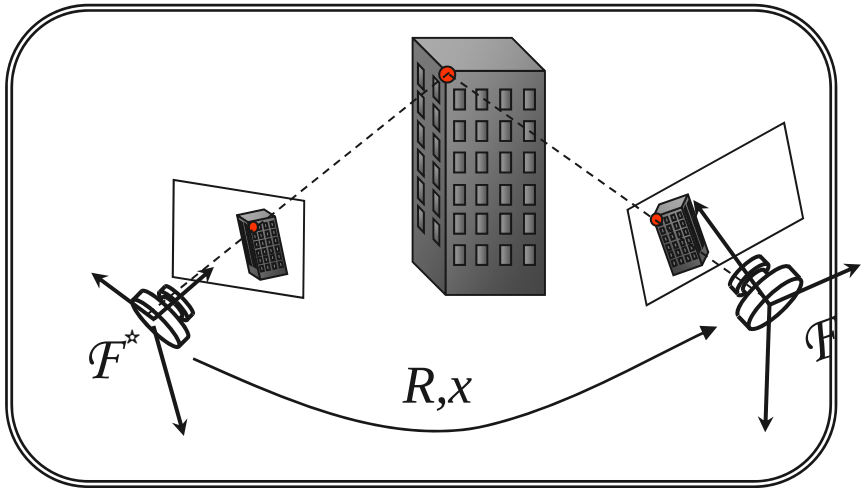
Pose Reconstruction



- Given two images of a scene, and matched corresponding feature points, reconstruct the relative position and orientation between the camera poses, i.e., “pose reconstruction”.
- Can also solve for relative feature point coordinates, up to a scale factor, hence “structure from motion”.

Pose Reconstruction

- A rotation $R \in SO(3)$ and translation $x \in \mathbb{R}^3$ separate the camera poses $\mathcal{F}$ and $\mathcal{F}^*$
- N feature points are extracted and matched in both images



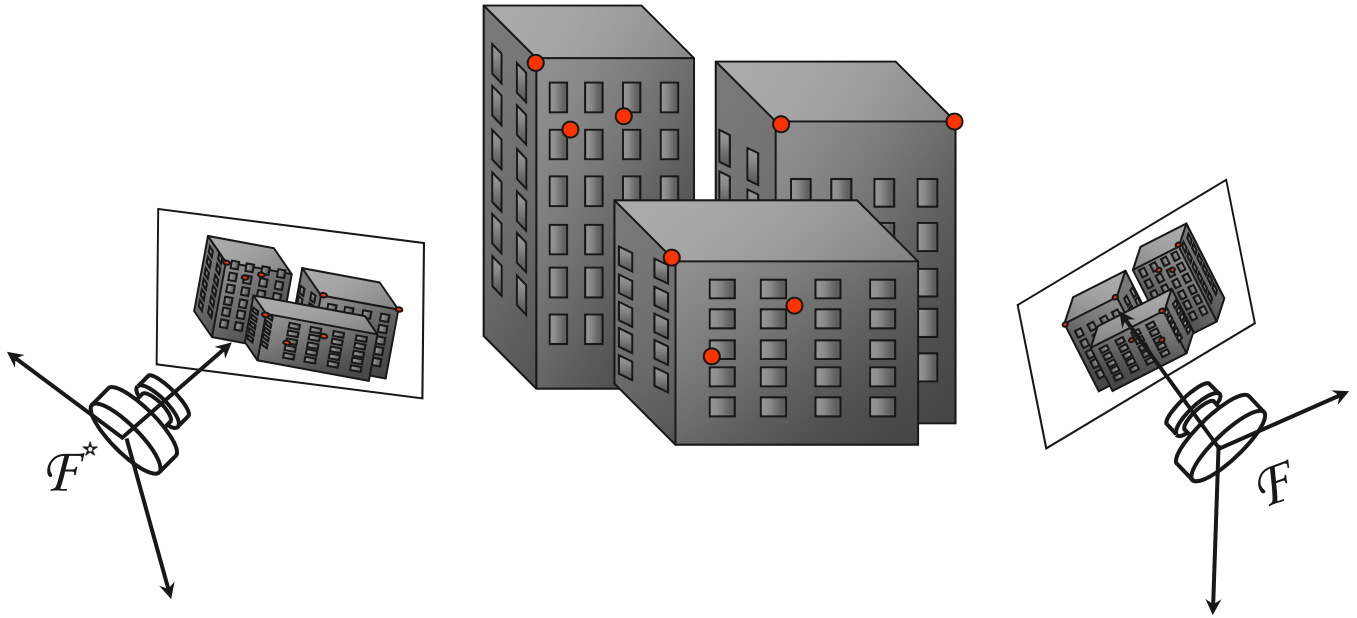
- Feature points have 3D coordinates in the two frames given by

$$\bar{m}_j^* = \begin{bmatrix} x_j^* & y_j^* & z_j^* \end{bmatrix}^T, \quad \bar{m}_j = \begin{bmatrix} x_j & y_j & z_j \end{bmatrix}^T, \quad \forall j \in \{1 \dots N\}$$

- The coordinates of each point in the camera frames are related by

$$\bar{m}_j = R\bar{m}_j^* + x, \ \forall j \in \{1 \dots N\}$$

Essential Matrix & 8 Point Algorithm



- We begin with the Essential Matrix and 8 Point Algorithm.
- Assume at least 8 points are matched in two images, no four 3D points are coplanar.
  - H. Longuet-Higgins, “A computer algorithm for reconstructing a scene from two projections,” *Nature*, pp. 133 – 135, Sept. 1981.
  - T. Huang and O. Faugeras, “Some properties of the E matrix in two-view motion estimation,” *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 11, no. 12, pp. 1310–1312, 1989.

Essential Matrix & 8 Point Algorithm

- Image feature points have 2D coordinates given by

$$m_j^* = \begin{bmatrix} \frac{x_j^*}{z_j^*} & \frac{y_j^*}{z_j^*} & 1 \end{bmatrix}^T, \quad m_j(t) = \begin{bmatrix} \frac{x_j(t)}{z_j(t)} & \frac{y_j(t)}{z_j(t)} & 1 \end{bmatrix}^T, \quad \forall j \in \{1 \dots N\}$$

- The coordinates of each point in the images are related by

$$z_j m_j = z_j^* R m_j^* + x, \ \forall j \in \{1 \dots N\}$$

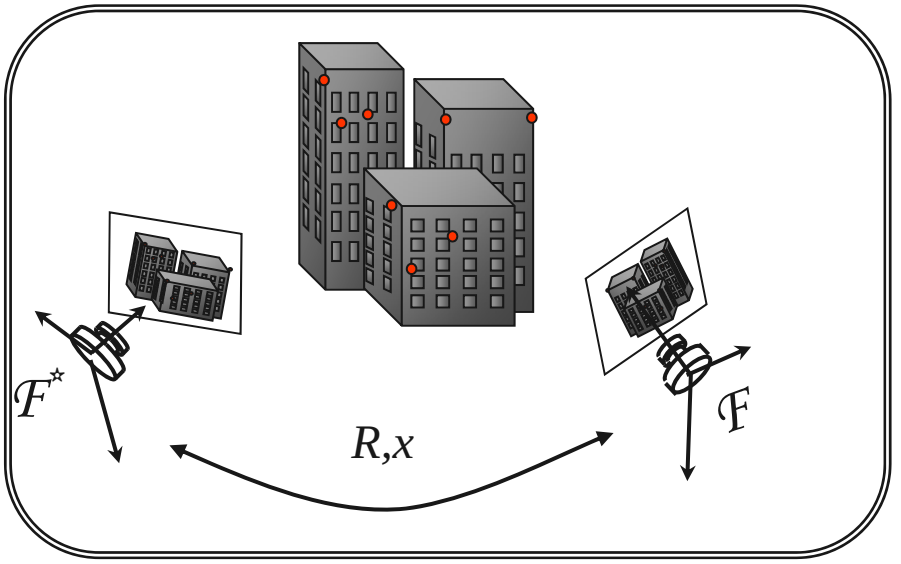
- Define $[x]_x$ as a skew-symmetric matrix such that $[x]_x m = x \times m$

$$z_j [x]_\times m_j = z_j^* [x]_\times R m_j^*$$

- Note that

$$z_j m_j^T [x]_\times m_j = 0$$

$$z_j m_j \cdot (x \times m_j) = 0$$



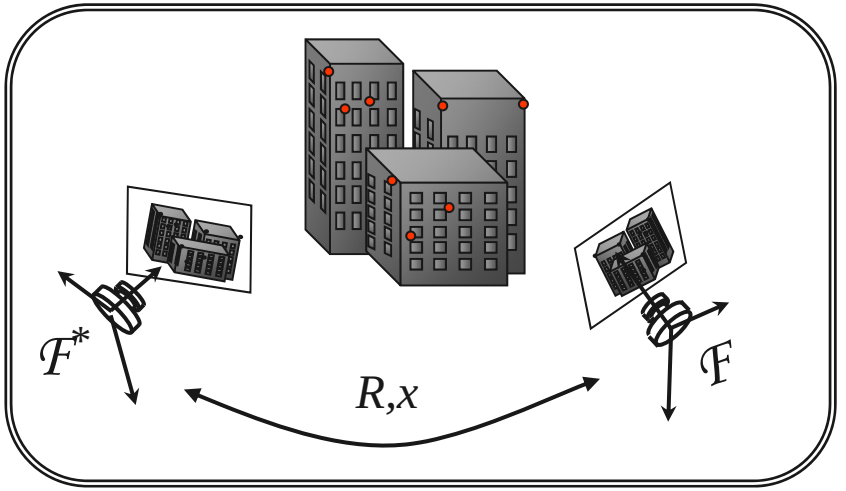
Essential Matrix & 8 Point Algorithm

- The equation

$$z_j^* m_j^T [x]_\times R m_j^* = 0$$

Loss of scale information

$$m_j^T E m_j^* = 0$$



holds for all points in the images, and is known as the Essential Constraint or Epipolar Constraint.

- $E = [x]_\times R$ is the Essential Matrix and contains relative pose information

Essential Matrix & 8 Point Algorithm

- If eight or more points are available, E can be estimated using linear algebra methods. This is known as the Eight Point Algorithm.
- "Unwrap" $E \in \mathbb{R}^{3x3}$ to a vector $e \in \mathbb{R}^{9}$

$$E = \begin{bmatrix} e_1 & e_2 & e_3 \\ e_4 & e_5 & e_6 \\ e_7 & e_8 & e_9 \end{bmatrix}^T \Rightarrow e = \begin{bmatrix} e_1, & e_2, & e_3, & e_4, & e_5, & e_6, & e_7, & e_8, & e_9 \end{bmatrix}^T$$

- Define a row vector M_j for the jth feature point

$$M_j = [m_x^* m_x, m_x^* m_y, m_x^*, m_y^* m_x, m_y^* m_y, m_y^*, m_x, m_y, 1]$$

- The Epipolar constraint can be rewritten as

$$M_j e = 0$$

- Concatenate all N vectors M_j to form matrix $M \in \mathbb{R}^{Nx9}$

$$Me = 0$$

Essential Matrix & 8 Point Algorithm

- The null space of M will give a vector e that satisfies the equation

$$Me = 0$$

- "Wrap" e back into a 3x3 matrix to get estimate of E

  Note that E will only be estimated up to a scale factor

- Noise and calibration errors tend to make the estimate of E incorrect, and won't necessarily satisfy $E = [x]_{\times} R$
- Any matrix that satisfies the above will have two identical singular values and one zero singular value
- We project our estimate of E to the closest true Essential Matrix
- Take SVD of E such that $E=U\Sigma V^T$, with $\Sigma=diag\{\sigma_1,\sigma_2,\sigma_3\}$
- Replace Σ with $\Sigma'=diag\{1,1,0\}$

Essential Matrix & 8 Point Algorithm

- With E now estimated, how do we recover x and R?
- Define a rotation matrix

$$R_z(\pm\tfrac{\pi}{2}) = \begin{bmatrix} 0 & \mp 1 & 0 \\ \pm 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- Having taken the SVD of E such that $E=U\Sigma V^T$, with $\Sigma=diag\{1,1,0\}$, it can be shown that

$$[x']_\times = UR_z(\pm\tfrac{\pi}{2})\Sigma U^T \qquad R = UR_z(\pm\tfrac{\pi}{2})V^T$$

Two Solutions!!

where $[x']_x = \lambda\,[x]_x$ for some unknown scale factor λ

Essential Matrix & 8 Point Algorithm

- It gets worse, note that the essential constraint $m^T E m^* = 0$ is valid for ± E
- We need to repeat the previous analysis with $\Sigma = diag\{-1,-1,0\}$
- This gives <u>four</u> possible solutions for the pose

- Luckily, three solutions will be physically impossible as they would require at least some feature points to be behind the camera
- For all points, test the equation $\lambda m_j = \lambda^* R m_j^* + x'$

it should be that for 3 solutions, λ<0, λ^*<0, or both to be satisfied

We have thus recovered rotation R and scaled translation $x' = \lambda' x$

Essential Matrix & 8 Point Algorithm

- With R and x' known, we can recover the relative depths of the feature points up to a scale factor λ'
- Manipulate the rigid body equation

$$\lambda m_j = \lambda^* R m_j^* + x'$$

$$\lambda^* [m_j]_\times R m_j^* + [m_j]_\times x' = 0$$

$$\begin{bmatrix} [m_j]_\times R m_j^* & [m_j]_\times x' \end{bmatrix} \begin{bmatrix} \lambda_j^* \\ 1 \end{bmatrix} = 0$$

- Combined Matrix for all feature points

$$\begin{bmatrix} [m_1]_\times R m_1^* & 0 & \cdots & 0 & [m_1]_\times x' \\ 0 & [m_2]_\times R m_2^* & \cdots & 0 & [m_2]_\times x' \\ \vdots & \vdots & \ddots & 0 & [m_{N-1}]_\times x' \\ 0 & 0 & 0 & [m_N]_\times R m_N^* & [m_N]_\times x' \end{bmatrix} \begin{bmatrix} \lambda^* \\ 1 \end{bmatrix} = 0$$

$3N \times N+1$ $\qquad$ $N+1$

Null Space gives all depths up to scale factor

Example of 8 Point Algorithm

Example using the computer vision toolbox

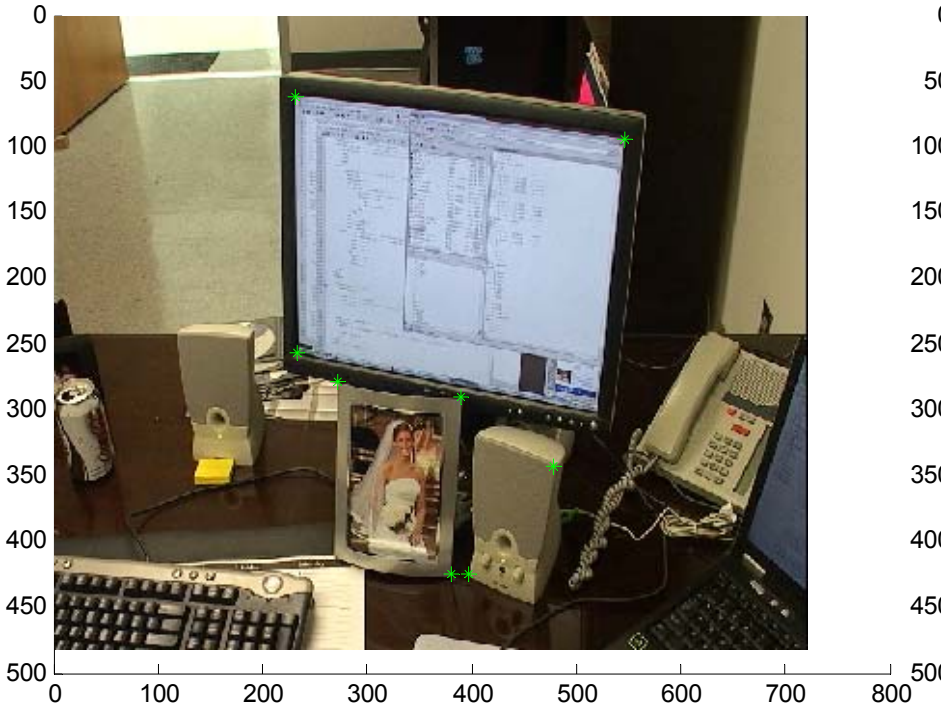


Image at $\mathcal{F}^*$

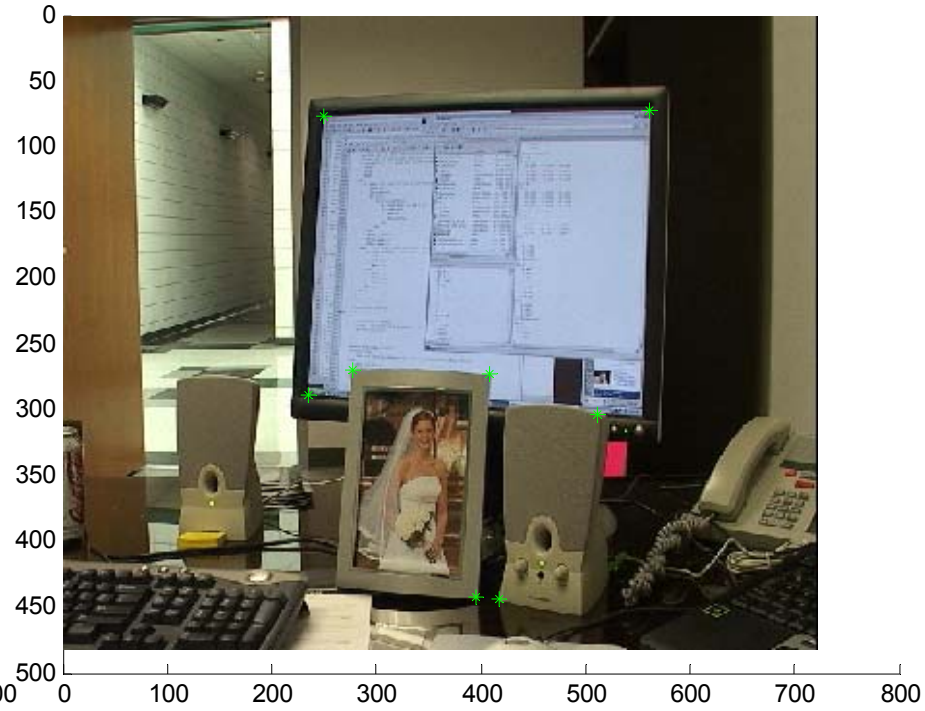


Image at $\mathcal{F}$

Example of 8 Point Algorithm

Essential Matrix Solution is

$$\begin{bmatrix} -0.0872 & -0.1641 & 0.9757 \\ 0.3792 & -0.0098 & 0.1414 \\ -0.9205 & -0.0261 & -0.0403 \end{bmatrix}$$

Four Solutions

$$x1 = \begin{bmatrix} -0.1165 \\ 0.9144 \\ 0.3877 \end{bmatrix} \quad R1 = \begin{bmatrix} -0.9847 & -0.1349 & -0.1107 \\ -0.1731 & 0.8350 & 0.5224 \\ 0.0219 & 0.5335 & -0.8455 \end{bmatrix}$$

$$x2 = \begin{bmatrix} -0.1165 \\ 0.9144 \\ 0.3877 \end{bmatrix} \quad R2 = \begin{bmatrix} -0.9928 & 0.0948 & -0.0727 \\ -0.1090 & -0.9683 & 0.2248 \\ 0.0491 & -0.2311 & -0.9717 \end{bmatrix}$$

$$x3 = \begin{bmatrix} 0.1165 \\ -0.9144 \\ -0.3877 \end{bmatrix} \quad R3 = \begin{bmatrix} 0.9847 & 0.1349 & 0.1107 \\ 0.1731 & -0.8350 & -0.5224 \\ -0.0219 & -0.5335 & 0.8455 \end{bmatrix}$$

$$x4 = \begin{bmatrix} -0.1165 \\ 0.9144 \\ 0.3877 \end{bmatrix} \quad R4 = \begin{bmatrix} 0.9928 & -0.0948 & 0.0727 \\ 0.1090 & 0.9683 & -0.2248 \\ -0.0491 & 0.2311 & 0.9717 \end{bmatrix}$$

Example of 8 Point Algorithm

Check positive depth constraint

$$\lambda m_j = \lambda^* R m_j^* + x' \Rightarrow \frac{\lambda}{\lambda^*} \left\| [x']_\times m_j \right\|^2 = \left([x']_\times m_j \right)^T [x']_\times R m_j^*$$

Negative value means signs in ratio don't agree, i.e. one is negative

Doing this with each solution for all points m_j and m_j^* gives

Solution 1	0.3478	0.3453	0.3114	0.3535	0.4036	0.4107	0.3125	0.3440
Solution 2	-0.3478	-0.3453	-0.3114	-0.3535	-0.4036	-0.4107	-0.3125	-0.3440
Solution 3	-0.3478	-0.3453	-0.3114	-0.3535	-0.4036	-0.4107	-0.3125	-0.3440
Solution 4	0.3478	0.3453	0.3114	0.3535	0.4036	0.4107	0.3125	0.3440

Solutions 2 and 3 are eliminated

Example of 8 Point Algorithm

Solve for relative depths using remaining solutions

$$\begin{bmatrix} [m_1]_\times R m_1^* & 0 & \cdots & 0 & [m_1]_\times x' \\ 0 & [m_2]_\times R m_2^* & \cdots & 0 & [m_2]_\times x' \\ \vdots & \vdots & \ddots & 0 & [m_{N-1}]_\times x' \\ 0 & 0 & 0 & [m_N]_\times R m_N^* & [m_N]_\times x' \end{bmatrix} \begin{bmatrix} \lambda^* \\ 1 \end{bmatrix} = 0$$

Solution 1

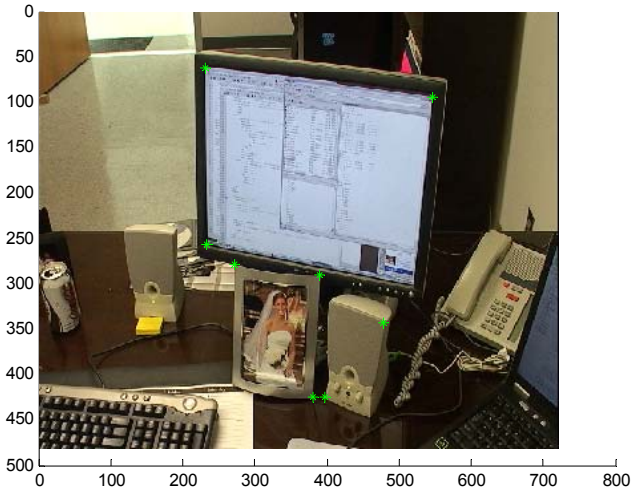
```
-1.6519   -1.6825   -1.0102   -1.6197   -8.3984   -9.7505   -1.0132   -1.48871.0000
```

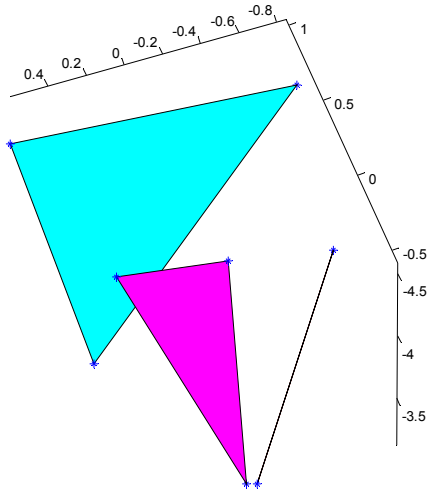
Solution 4

```
3.9339    4.5560    3.3930    3.1714    3.6137    4.8309    3.4084    5.6772    1.0000
```

Solution 4 has all positive depths, we keep it

Example of 8 Point Algorithm





$$x_4 = \begin{bmatrix} -0.1165 \\ 0.9144 \\ 0.3877 \end{bmatrix} \qquad R_4 = \begin{bmatrix} 0.9928 & -0.0948 & 0.0727 \\ 0.1090 & 0.9683 & -0.2248 \\ -0.0491 & 0.2311 & 0.9717 \end{bmatrix}$$

Translation and rotation seem like good match

Reconstructed points are reasonable

Exercise

Task:

- Take a sequence of images: take one image, rotate and translation the camera, then take another one.
- Determine 8 feature points with no more than 4 coplanar feature points
- Calculate the Essential matrix
- Decompose the essential matrix to find the rotation and scaled translation for the camera's poses for the image sequence

Note:

- Can use either Matlab or C/C++ for programming

Exercise (Cont'd)

Reference:

http://petercorke.com/Machine_Vision_Toolbox.html

Other public domain tools for Matlab based machine vision or image processing on the Web

- MATLAB and Octave Functions for Computer Vision and Image Processing (Kovesi)
- Epipolar Geometry Toolbox
- Visual Servoing Toolbox
- Camera calibration toolbox for Matlab (Bouguet)
- Camera calibration toolbox for Matlab (Heikkila)
- FAQ about color and gamma
- Stereo matching for Matlab (Jasmine Banks)

Olivier Faugeras and Quang-Tuan Luong and Theodore Papadopoulo (2001). *The Geometry of Multiple Images*. MIT Press.

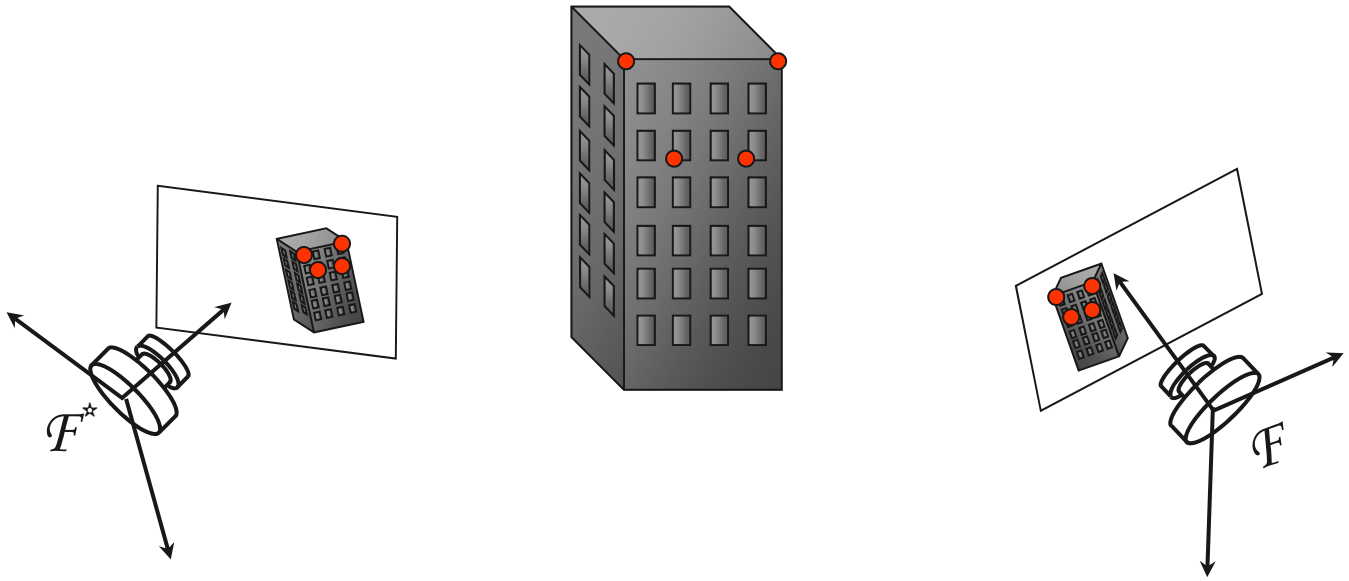
H. Longuet-Higgins, "A computer algorithm for reconstructing a scene from two projections," *Nature*, pp. 133 – 135, Sept. 1981.

T. Huang and O. Faugeras, "Some properties of the E matrix in two-view motion estimation," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 11, no. 12, pp. 1310–1312, 1989.

Homography Matrix

- **Homography**: The geometric concept of homography is a one-to-one and on-to transformation or mapping between two sets of points. In computer vision, homography refers to the mapping between points in two Euclidean-planes (Euclidean homography), or to the mapping between points in two images (projective homography).

Multi-View Geometry



- We now address the issue of planar scenes using the Euclidean Homography Matrix and the 4 point algorithm
- Assume at least 4 coplanar, noncollinear points are matched in two images

O. Faugeras and F. Lustman, “Motion and structure from motion in a piecewise planar environment,” *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 2, no. 3, pp. 485–508, 1988.

Euclidean Geometry

- Image features points have 2D coordinates given by

$$m_j^* = \begin{bmatrix} \frac{x_j^*}{z_j^*} & \frac{y_j^*}{z_j^*} & 1 \end{bmatrix}^T, \quad m_j(t) = \begin{bmatrix} \frac{x_j(t)}{z_j(t)} & \frac{y_j(t)}{z_j(t)} & 1 \end{bmatrix}^T, \quad \forall j \in \{1 \ldots N\}$$

- The coordinates of each point in the images are related by

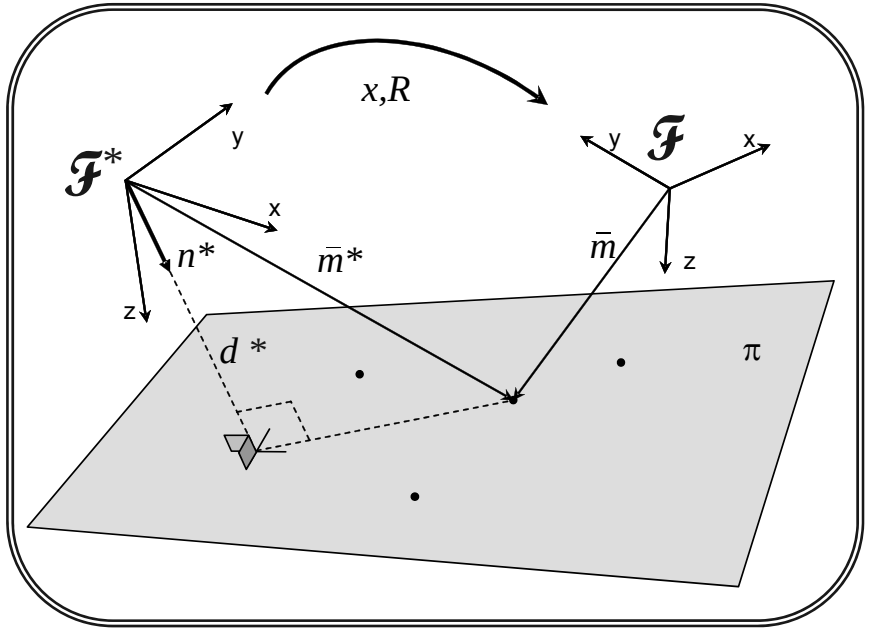
$$\overline{m}_j(t) = R(t)\overline{m}_j^* + x(t)$$

$$z_j m_j = z_j^* R m_j^* + x, \ \forall j \in \{1 \ldots N\}$$

- The points all lie in a plane π, which has normal vector n^* in $\mathcal{F}^*$, and lies a distance d^* from the origin

- The following relationship is true for all $\overline{m}_j^*$

$$d^* = n^{*T}\overline{m}_j^*$$



Euclidean Geometry & Homography Matrix

- Combine equations

$$\overline{m}_j(t) = R(t)\overline{m}_j^* + x(t)$$

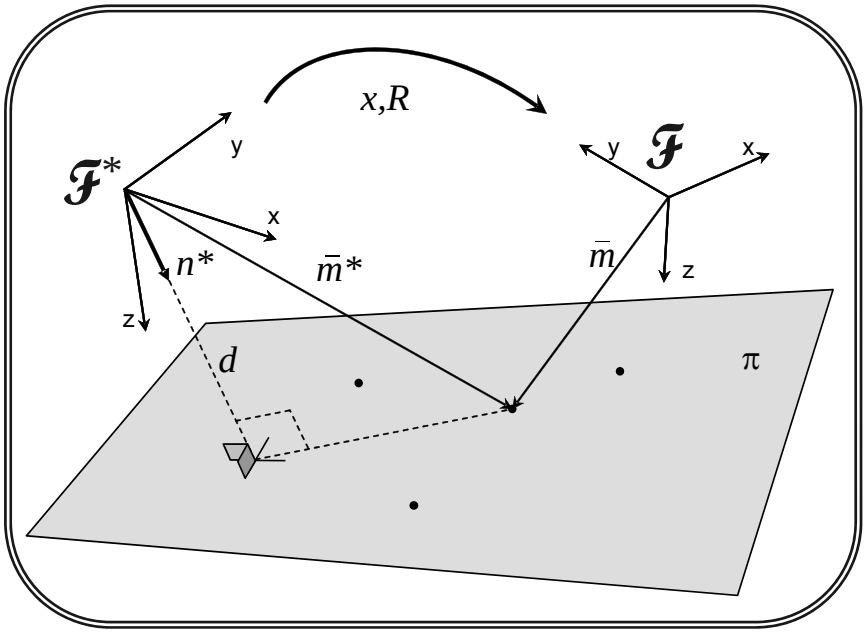
$$d^* = n^{*T}\overline{m}_j^*$$

to get

Loss of scale information

$$m_j = \frac{z_j^*}{z_j}\left(R + \frac{x}{d^*}n^{*T}\right)m_j^*$$

$$= \alpha_j H m_j^*$$



- $H = R + \frac{x}{d^*}n^{*T}$ is the **Euclidean Homography Matrix** that contains relative pose information and some structure information

- $\alpha_j = \frac{z_j^*}{z_j}$ is a ratio of depths (called depth ratio)

Homography Decomposition

- If four or more points are available, H can be estimated up to a scale factor using linear algebra methods (we recover scale later).
- Scale H by its 3,3 element to get H_n
- "Unwrap" $H_n \in \mathbb{R}^{3x3}$ to a vector $h \in \mathbb{R}^8$

$$H_n = \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & 1 \end{bmatrix} \Longrightarrow h = [h_1, h_2, h_3, h_4, h_5, h_6, h_7, h_8]^T$$

- Define a 2x8 matrix M_j for the j^{th} feature point

$$M_j = \begin{bmatrix} m_x^*, m_y^*, 1, 0, 0, 0, -m_x^* m_x, -m_y^* m_x \\ 0, 0, 0, m_x^*, m_y^*, 1, -m_x^* m_y, -m_y^* m_y \end{bmatrix} \text{ such that } M_j h = m_j$$

- Concatenate all N vectors m_j and matrices M_j to form matrix $M \in \mathbb{R}^{2N x 8}$

$$Mh = m$$

Homography Decomposition (Cont'd)

- Use standard linear algebra to find vector h that satisfies

$$Mh = m$$

- Wrap h back into a 3x3 matrix to get estimate of H_n
- Take the SVD of H_n such that $H_n = U\Sigma V^T$, this maps to an orthogonal space such that

$$\Sigma = R' + x' n'^T$$

- If we find R', x', n', we can recover R, x/d^*, n^* as

$$R = UR'V^T \qquad \frac{x}{d^*} = Ux' \qquad n^* = Vn'$$

- Recovery of these terms gets pretty complicated

Homography Decomposition (Cont'd)

- First recover the lost scale of H, i.e. find h_{33}
- $\Sigma = diag\{\sigma_1, \sigma_2, \sigma_3\}$,
- $h_{33} = \pm\sigma_2$ such that $det(U)det(V)\sigma_2 > 0$
- There are different solutions depending on number of unique singular values, and the sign of $det(U)det(V)$

- If all singular values are the same, there was no translation, so

$$R = H,\ x = 0,\ n^* \text{ is unknown}$$

$$H = R + \frac{x}{d^*} n^{*T}$$

Homography Decomposition (Cont'd)

- If two singular value the same, translation was along the normal

case: $det(U)det(V) > 0$

$$R' = I$$
$$n' = [0, 0, \pm 1]^T$$
$$x' = (\sigma_3 - \sigma_1)n'$$

case: $det(U)det(V) < 0$

$$R' = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$
$$n' = [0, 0, \pm 1]^T$$
$$x' = (\sigma_3 + \sigma_1)n'$$

- Note the sign ambiguity on n'
- There are two mathematically valid solutions, but we eliminate one later due to positive depth constraint

Homography Decomposition (Cont'd)

- If all singular values are unique, the motion was general

case: $det(U)det(V) > 0$

$$R' = \begin{bmatrix} c\theta & 0 & -s\theta \\ 0 & 1 & 0 \\ s\theta & 0 & c\theta \end{bmatrix}$$

$$n' = \left[\pm\sqrt{\frac{\sigma_1^2 - \sigma_2^2}{\sigma_1^2 - \sigma_3^2}}, 0, \pm\sqrt{\frac{\sigma_2^2 - \sigma_2^2}{\sigma_1^2 - \sigma_3^2}} \right]^T$$

$$x' = (\sigma_1 - \sigma_3)n'$$

$$s\theta = \pm\sqrt{\frac{(\sigma_1^2 - \sigma_2^2)(\sigma_2^2 - \sigma_3^2)}{(\sigma_1 + \sigma_3)\sigma_2}}$$

$$c\theta = \frac{\sigma_2^2 + \sigma_1\sigma_3}{(\sigma_1 + \sigma_3)\sigma_2}$$

case: $det(U)det(V) < 0$

$$R' = \begin{bmatrix} c\theta & 0 & s\theta \\ 0 & -1 & 0 \\ s\theta & 0 & -c\theta \end{bmatrix}$$

$$n' = \left[\pm\sqrt{\frac{\sigma_1^2 - \sigma_2^2}{\sigma_1^2 - \sigma_3^2}}, 0, \mp\sqrt{\frac{\sigma_2^2 - \sigma_2^2}{\sigma_1^2 - \sigma_3^2}} \right]^T$$

$$x' = (\sigma_1 + \sigma_3)n'$$

$$s\theta = \pm\sqrt{\frac{(\sigma_1^2 - \sigma_2^2)(\sigma_2^2 - \sigma_3^2)}{(\sigma_1 - \sigma_3)\sigma_2}}$$

$$c\theta = \frac{\sigma_1\sigma_3 - \sigma_2^2}{(\sigma_1 - \sigma_3)\sigma_2}$$

Four mathematically valid solutions, but we can eliminate two

Homography Decomposition (Cont'd)

- Recover $H = H_n h_{33}$
- Recover $\alpha_j = 1/(H_3^T m_j)$ where H_3^T is the third row of H
- Test each solution for positive depth constraint

$$n^{*T}\overline{m}_j^* = d^* > 0 \quad \Longrightarrow \quad n^{*T} m_j > 0 \;\; \forall j$$

- In the general case, two solutions will pass the depth constraint, how to choose the correct one?
  - A priori knowledge of the scene can be used to test n^* vs expected normal
  - Coherence of multiple views can be used. Solve for solutions to a third view with respect to $\mathcal{F}^*$, pick the solutions that have same (or closest) n^*

Homography Decomposition (Cont'd)

- We have thus recovered rotation R and scaled translation $x'=x/d^*$
- Can recover relative feature point depths as of the feature points up to a scale factor by noting

$$\bar{m}_j^T n^* = d^*$$

$$z_j m_j^T n^* = d^*$$

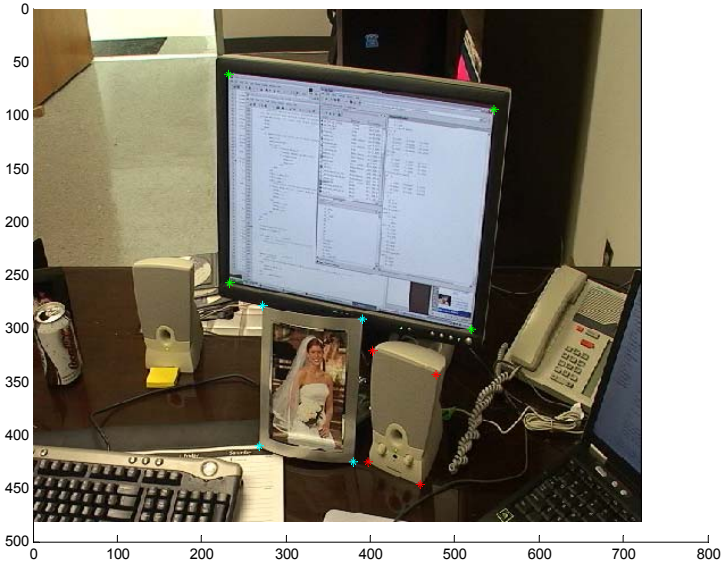
$$\lambda_j m_j^T n^* = 1$$

$$\lambda_j = \frac{1}{m_j^T n^*}$$

- If d^* is known or can be measured or estimated, we can recover translation and Euclidean coordinates of all points

Example of 4 Point Algorithm

Example using the computer vision toolbox



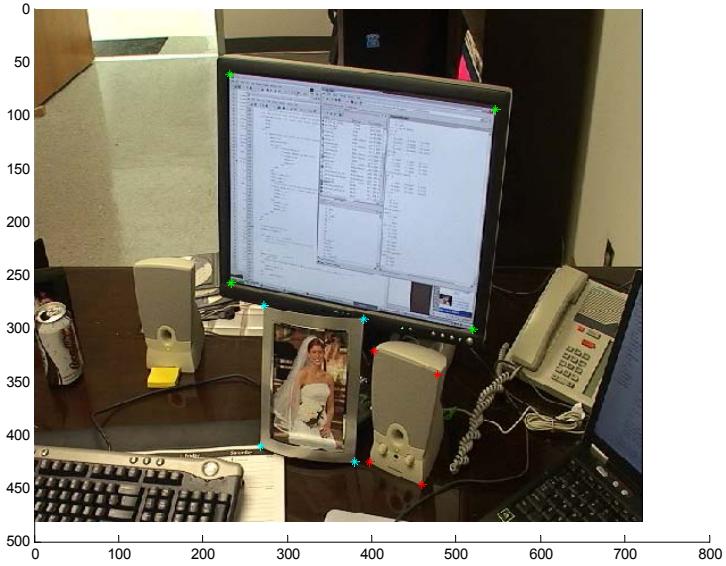


Image at $\mathcal{F}^*$

Image at $\mathcal{F}$

A third image was taken for coherence

Example of 4 Point Algorithm

Focus on monitor points

Homography Matrix Solution is

$$\begin{bmatrix} 1.0456 & 0.0099 & -0.0163 \\ 0.0869 & 1.0322 & 0.0210 \\ -0.0192 & 0.5301 & 1.1621 \end{bmatrix}$$

Four Solutions

$$x1 = \begin{bmatrix} -0.1361 \\ 0.1060 \\ -0.5024 \end{bmatrix} \quad R1 = \begin{bmatrix} 0.9924 & -0.1174 & -0.0378 \\ 0.1173 & 0.9931 & -0.0031 \\ 0.0379 & -0.0014 & 0.9993 \end{bmatrix} \quad n^*1 = \begin{bmatrix} 0.1580 \\ 0.1789 \\ 0.9711 \end{bmatrix}$$

$$x2 = \begin{bmatrix} -0.0607 \\ 0.4837 \\ 0.2109 \end{bmatrix} \quad R2 = \begin{bmatrix} 0.9992 & 0.0076 & 0.0385 \\ 0.0108 & 0.8905 & -0.4548 \\ -0.0377 & 0.4549 & 0.8898 \end{bmatrix} \quad n^*2 = \begin{bmatrix} 0.0896 \\ -0.9788 \\ -0.1839 \end{bmatrix}$$

$$x3 = \begin{bmatrix} 0.0607 \\ -0.4837 \\ -0.2109 \end{bmatrix} \quad R3 = \begin{bmatrix} 0.9992 & 0.0076 & 0.0385 \\ 0.0108 & 0.8905 & -0.4548 \\ -0.0377 & 0.4549 & 0.8898 \end{bmatrix} \quad n^*3 = \begin{bmatrix} -0.0896 \\ 0.9788 \\ 0.1839 \end{bmatrix}$$

$$x4 = \begin{bmatrix} 0.1361 \\ -0.1060 \\ 0.5024 \end{bmatrix} \quad R4 = \begin{bmatrix} 0.9924 & -0.1174 & -0.0378 \\ 0.1173 & 0.9931 & -0.0031 \\ 0.0379 & -0.0014 & 0.9993 \end{bmatrix} \quad n^*4 = \begin{bmatrix} -0.1580 \\ -0.1789 \\ -0.9711 \end{bmatrix}$$

Example of 4 Point Algorithm

Check positive depth constraint

$$n^{*T} m_j > 0 \quad \forall j$$

Doing this with each solution for all points m_j and m_j^* gives

Solution 1	0.9565	0.9798	1.0139	0.9885
Solution 2	-0.1935	-0.1845	-0.3832	-0.3858
Solution 3	0.1935	0.1845	0.3832	0.3858
Solution 4	-0.9565	-0.9798	-1.0139	-0.9885

Solutions 2 and 4 are eliminated

Example of 4 Point Algorithm

A third image yielded valid estimates for n*

n*1b	0.2134 0.1771 0.9608	n*2b	0.8945 0.2072 0.3962

Compare to remaining solutions from previous image

n*1	0.1580 0.1789 0.9711	n*3	-0.0896 0.9788 0.1839

One pair matched closely, so keep that solution

Essential vs. Homography Matrix

- The two methods have different requirements and benefits that may determine which is better for a specific problem
- Essential Matrix requires at least 8 points, Homography requires at least 4
- Essential Matrix cannot have all point coplanar, Homography requires all points to be coplanar
- Essential Matrix requires nonzero translation, and results degenerate as $x \rightarrow 0$, since $E = [x]_{\times} R$
- Homography requires an extra view or knowledge to select from multiple solutions
- In my experience, Homography is more accurate

Essential vs. Homography Matrix

- It is possible to recover E from H by solving for x and R
- It is possible to recover H from E through additional work
  - Plane is defined by three of the eight points used to find E
  - See “An Invitation to 3D Vision” for details
- Can recover H from eight noncoplanar points through “Virtual Parallax”
  - Plane is defined by subset of three points
  - Reference: B. Boufama and R. Mohr, “Epipole and fundamental matrix estimation using virtual parallax”. Proc. Int. Conf. on Computer Vision, pp. 1030–1036, 1995.

Exercise (not mandatory)

Task:

- Take a sequence of images: take one image, rotate and translation the camera, then take another one.
- Determine 4 coplanar feature points that are not collinear
- Calculate the Euclidean Homography matrix or Projective Homography matrix
- Decompose the Homography matrix to find the rotation and scaled translation for the camera's poses for the image sequence

Note:

- Can use either Matlab or C/C++ for programming

Exercise (Cont'd)

Reference:

http://petercorke.com/Machine_Vision_Toolbox.html

Other public domain tools for Matlab based machine vision or image processing on the Web

- MATLAB and Octave Functions for Computer Vision and Image Processing (Kovesi)
- Epipolar Geometry Toolbox
- Visual Servoing Toolbox
- Camera calibration toolbox for Matlab (Bouguet)
- Camera calibration toolbox for Matlab (Heikkila)
- FAQ about color and gamma
- Stereo matching for Matlab (Jasmine Banks)

O. Faugeras, Three-Dimensional Computer Vision: A Geometric Viewpoint, MIT Press, Cambridge, Massachusetts, 1993.