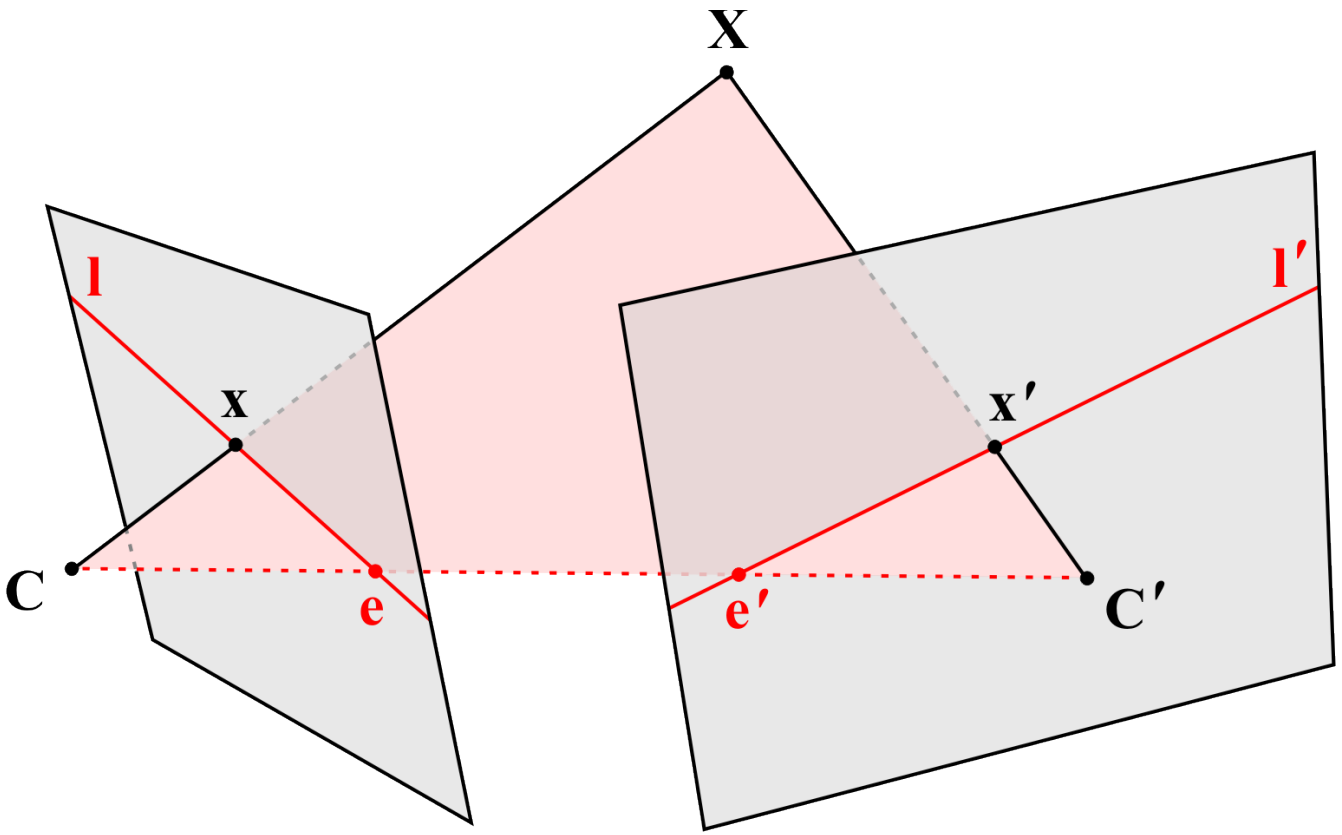
Photogrammetric Computer Vision

Berlin University of Technology (TUB),
Computer Vision and Remote Sensing Group
Berlin, Germany

5. Exercise

Projective and direct Euclidean reconstruction

With knowledge of the relative orientation spatial object coordinates can be triangulated from corresponding image points.



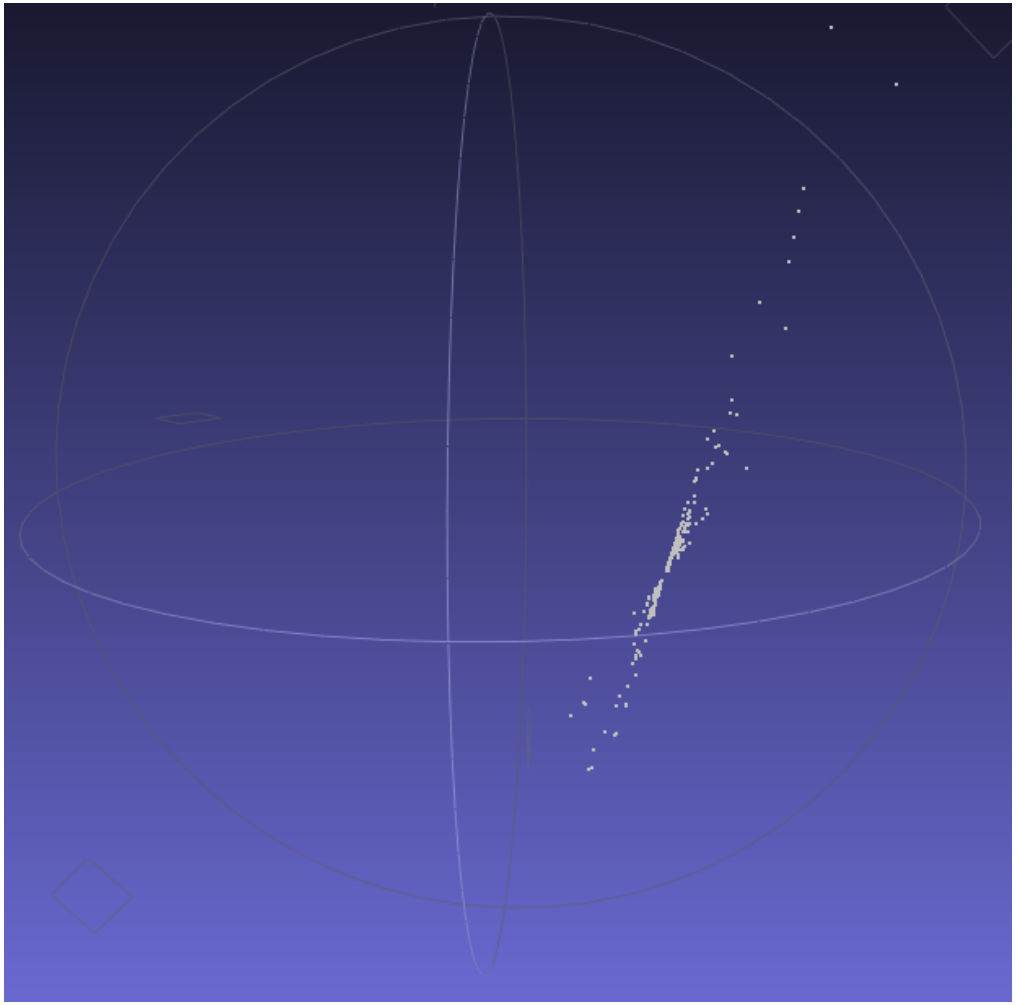
$$l' = F \cdot x$$

$$l = F^T \cdot x'$$

5. Exercise

Projective and direct Euclidean reconstruction

With knowledge of the relative orientation spatial object coordinates can be triangulated from corresponding image points.



If the parameters of interior orientation are unknown, then only a projective reconstruction is possible.

5. Exercise

Projective and direct Euclidean reconstruction

With knowledge of the relative orientation spatial object coordinates can be triangulated from corresponding image points.

If the parameters of interior orientation are unknown, then only a projective reconstruction is possible.

Using five control points this intermediate result can be transformed quite simply into a Euclidean reconstruction.

5. Exercise

1. Homologous points:

- Homologous image points for the image pair showing
  the bust of Beethoven given in a text file

- Read the homologous image coordinates x1 ↔ x2
  in the format (x1, y1; x2, y2)

2. Computation of the fundamental matrix:

- Determine the relative orientation of the images with the fundamental matrix F

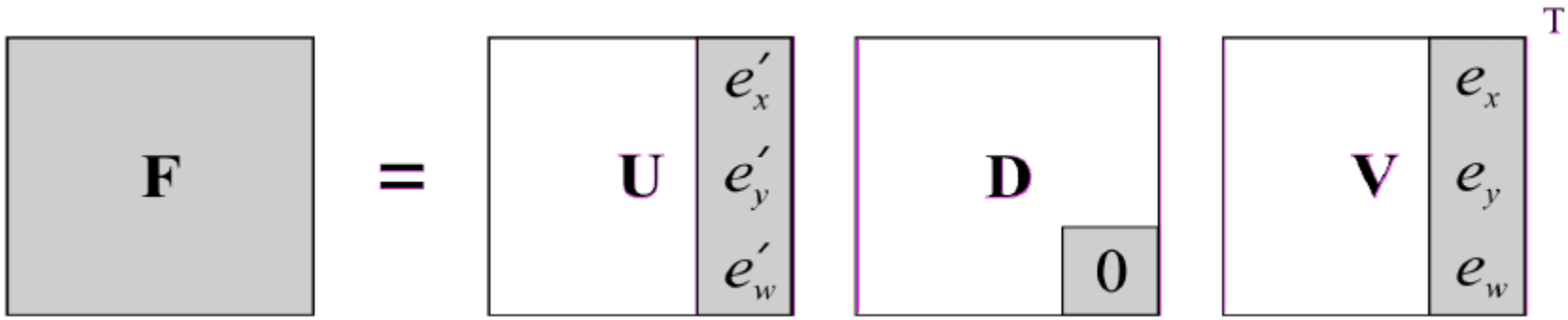
  → Exercise 4 (just with far more points)
  - Point conditioning
  - Creation of corresponding design matrix
  - Solving the equation system by SVD
  - Forcing rank 2
  - De-conditioning

5. Exercise

3. Camera definition:

Implement a new function, which defines two corresponding projection-matrices P_N and P' by means of F.

$$\boldsymbol{P}_N = [\boldsymbol{I} \,|\, \boldsymbol{0}] \qquad \boldsymbol{P}' = [[\boldsymbol{e}'_x]\boldsymbol{F} \,|\, \boldsymbol{e}'] \qquad [\boldsymbol{a}_x] = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix}$$



5. Exercise

3. Camera definition:

Implement a new function, which defines two corresponding projection-matrices P_N and P' by means of F.

$$\boldsymbol{P}_N = [\boldsymbol{I} \,|\, \boldsymbol{0}] \qquad \boldsymbol{P}' = [[\boldsymbol{e}'_x]\boldsymbol{F} \,|\, \boldsymbol{e}'] \qquad [\boldsymbol{a}_x] = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix}$$

4. Linear triangulation:

Realize a function for the linear triangulation of projective object points X_P and try to visualize the computed spatial object coordinates.

4.1 Design matrix:

$$A = \begin{bmatrix} x\,\boldsymbol{p}^3 - \boldsymbol{p}^1 \\ y\,\boldsymbol{p}^3 - \boldsymbol{p}^2 \\ x'\,\boldsymbol{p}'^3 - \boldsymbol{p}'^1 \\ y'\,\boldsymbol{p}'^3 - \boldsymbol{p}'^2 \end{bmatrix}$$

4.2 Solve with SVD to obtain $X_O = (X,Y,Z,W)^T$

4.3 Visualization: e.g. meshlab (http://www.meshlab.org/)

5. Exercise

5. Control Points

- Control points (triple of 2 image and 1 corresponding object point) for the image pair showing the bust of Beethoven given in a text file

- Read the control coordinates in x1 ↔ x2 ↔ X
  the format (x1, y1; x2, y2; X1, X2, X3)

6. Linear triangulation

- Triangulation of control points with same P_N and P' as for image points before
  → Creation of design matrix
  → Application of SVD to obtain $X_P = (X,Y,Z,W)^T$

5. Exercise

7. 3D spatial homography

- Extend algorithm from 2nd exercise to three dimensions

- Determine spatial transformation H from
  projective points X_P to Euclidian points X_E

- Apply transformation H to object points X_O of projective reconstruction

8. Visualization

- e.g. using meshlab (http://www.meshlab.org/)

5. Exercise - Given

```
int readMatchingPoints(string fname, Mat& x1, Mat& x2)
int readControlPoints(string fname,Mat& x1,Mat& x2,Mat& Xm)
```

`fname`: path of file containing point list
`x1`: 3xN matrix containing points of first image
`x2`: 3xN matrix containing points of second image
`Xm`: 4xN matrix containing object points
`ret`: number of points

- Read homologeous points (x_1,y_1;x_2,y_2) or control points (x_1,y_1;x_2,y_2;X_o,Y_o,Z_o) from file

```
void savePointList(string fname, Mat& points)
```

`fname`: file name
`points`: 4xN (3xN) matrix containing reconstructed N 3D object points

- Save reconstructed points
- File might be used by visualization software

5. Exercise - To Do

```
Mat getFundamentalMatrix(Mat& p1, Mat& p2)
```

`p1`: N points as 3xN matrix from first “image”
`p2`: N points as 3xN matrix from second “image”
`ret`: Output, 3x3 fundamental matrix

- Estimates fundamental matrix from given set of point pairs
- Calls:
  - `getCondition2D(...)`
  - `transform(...)`
  - `getDesignMatrix_fundamental(...)`
  - `solve_dlt(...)`
  - `forceSingularity(...)`
  - `decondition_fundamental(...)`

5. Exercise - To Do

```
void defineCameras(Mat& F, Mat& P1, Mat& P2)
```

F: the fundamental matrix
P1: first projection matrix (standard P matrix)
P2: second projection matrix based on F

- Generates two projection matrices by using fundamental matrix
- Calls:
  - `getEpipols(...)`
  - `makeSkewMatrix(...)`

$$P_N = [I \mid 0] \qquad P' = [[e'_x]F \mid e'] \qquad [a_x] = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix}$$

5. Exercise - To Do

```
void getEpipols(Mat& F, Mat& e1, Mat& e2)
```

F: the fundamental matrix
e1: first epipol
e2: second epipol

- Calculates epipols from fundamental matrix by SVD

```
Mat makeSkewMatrix(Mat& e)
```

e: given 3D – vector
ret: skew matrix

$$[\boldsymbol{a}_x] = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix}$$

- Generates skew matrix from vector

5. Exercise - To Do

```
Mat linearTriangulation(Mat& P1, Mat& P2, Mat& x1,Mat& x2)
```

`P1:`	first projection matrix (standard P matrix)
`P2:`	second projection matrix based on F
`x1:`	image point set of first image
`x2:`	image point set of second image
`ret:`	triangulated object points

- Triangulates given set of image points based on projection matrices

$$A = \begin{bmatrix} x\,\boldsymbol{p}^{3} - \boldsymbol{p}^{1} \\ y\,\boldsymbol{p}^{3} - \boldsymbol{p}^{2} \\ x'\,\boldsymbol{p}'^{3} - \boldsymbol{p}'^{1} \\ y'\,\boldsymbol{p}'^{3} - \boldsymbol{p}'^{2} \end{bmatrix}$$

5. Exercise - To Do

```
Mat homography3D(Mat& Xp, Mat& Xm)
```

X1: first set of points
X2: second set of points
ret: computed homography

- Computes 3D homography
- Calls
  - getCondition3D(...)
  - transform(...)
  - getDesignMatrix_homography3D(...)
  - solve_dlt(...)
  - decondition_homography3D(...)

Deadline: 03.02.2017