# VAN course
# Lesson 4

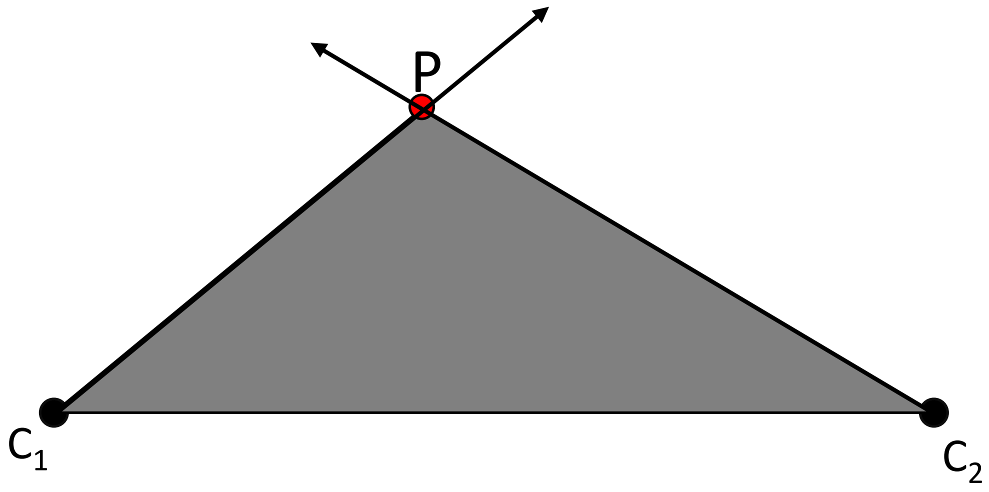
Dr. Refael Vivanti
refael.vivanti@mail.huji.ac.il

# Today’s topics

- Epipolar geometry
- Epipole
- Fundamental Matrix Calculation
- Rectification
- RANSAC

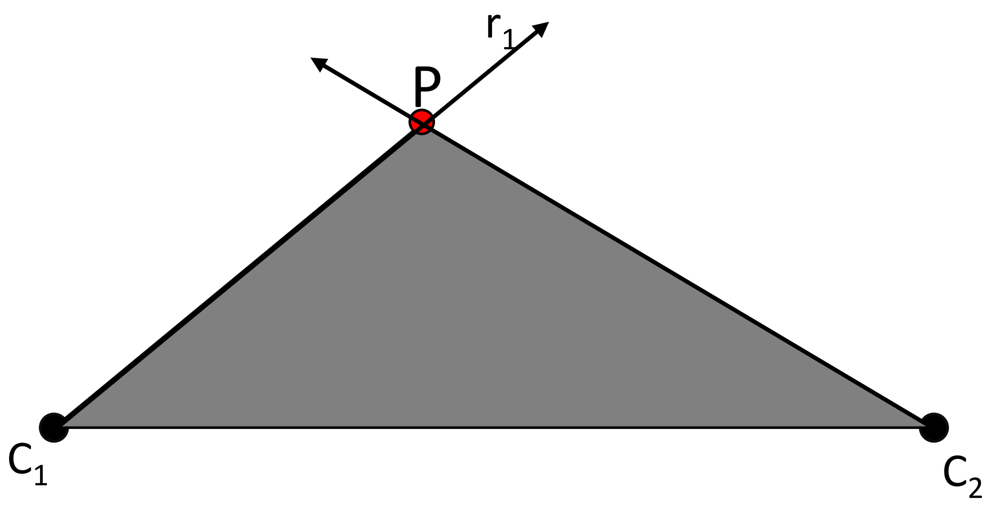
# Epipolar geometry

- The relations between views as appeared in the image



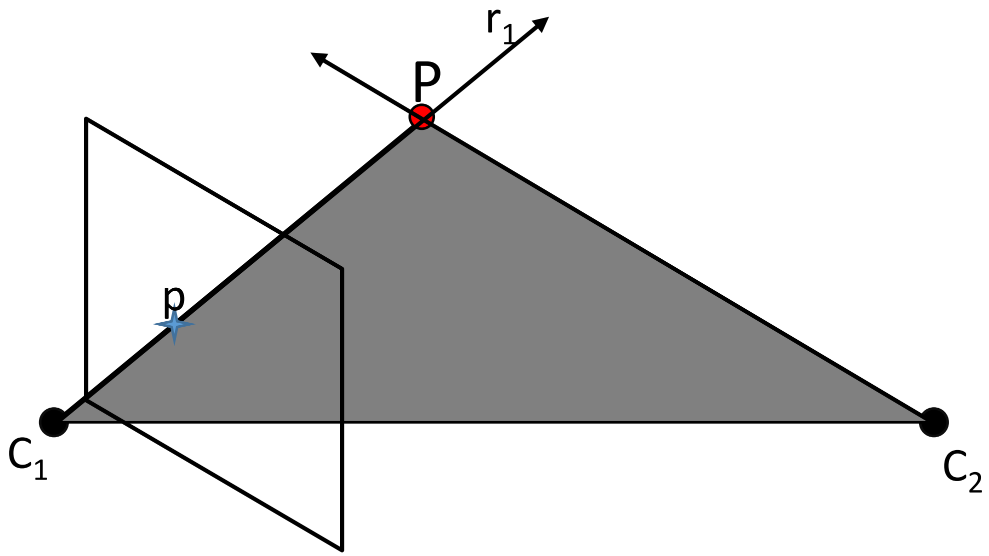
# Epipolar geometry

- The relations between views as appeared in the image
- **In 3D:** Ray $r_1$ intersects P



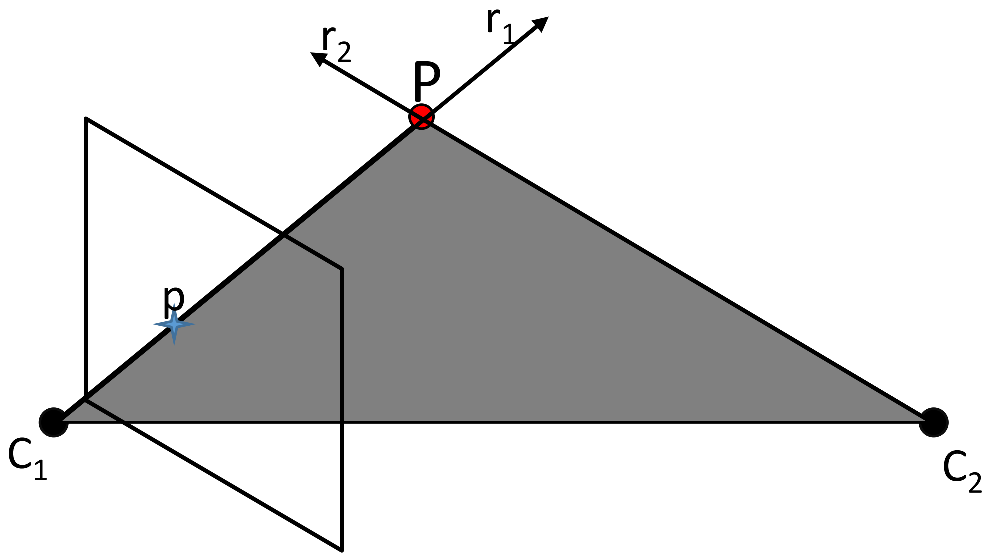
# Epipolar geometry

- The relations between views as appeared in the image
- **In 3D:** Ray $r_1$ intersects P **In image:** P is projected to p



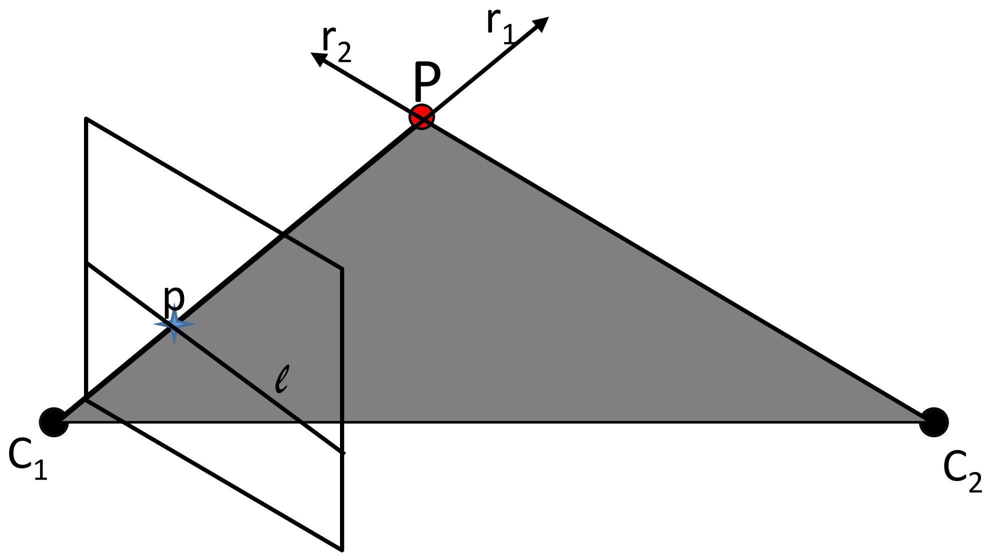
# Epipolar geometry

- The relations between views as appeared in the image
- **In 3D:** Ray $r_1$ intersects P  **In image:** P is projected to p
- **In 3D:** Ray $r_2$ intersect P



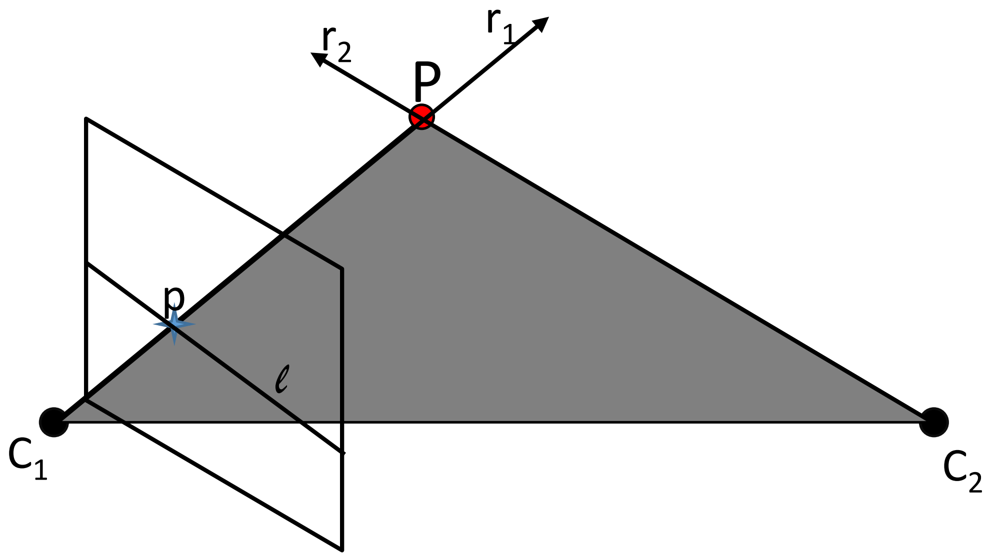
# Epipolar geometry

- The relations between views as appeared in the image
- **In 3D:** Ray $r_1$ intersects P **In image:** P is projected to p
- **In 3D:** Ray $r_2$ intersect P **In Image:** $r_2$ is projected to line $\ell$



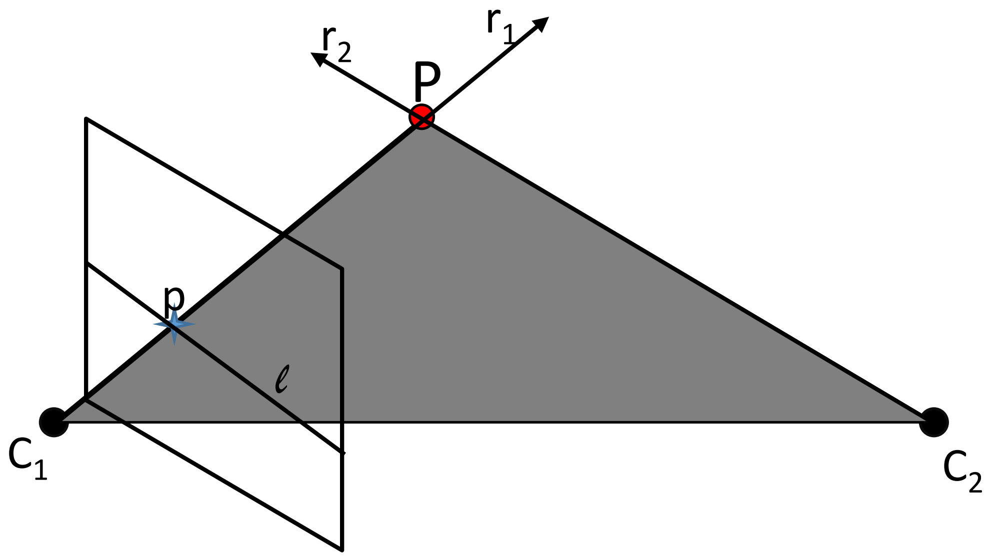
# Epipolar geometry

- The relations between views as appeared in the image
- **In 3D:** Ray $r_1$ intersects P **In image:** P is projected to p
- **In 3D:** Ray $r_2$ intersect P **In Image:** $r_2$ is projected to line $\ell$
- $\ell$ is the **epipolar line**



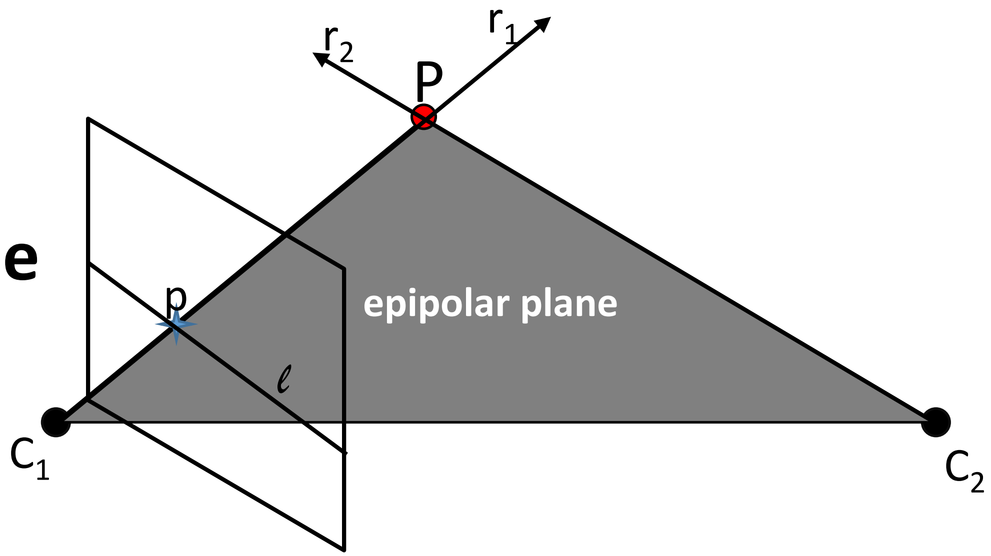
# Epipolar geometry

- The relations between views as appeared in the image
- **In 3D:** Ray $r_1$ intersects P **In image:** P is projected to p
- **In 3D:** Ray $r_2$ intersect P **In Image:** $r_2$ is projected to line $\ell$
- $\ell$ is the **epipolar line**
- $\ell$ intersects p in image 1



# Epipolar geometry

- The relations between views as appeared in the image
- **In 3D:** Ray $r_1$ intersects P **In image:** P is projected to p
- **In 3D:** Ray $r_2$ intersect P **In Image:** $r_2$ is projected to line $\ell$
- $\ell$ is the **epipolar line**
- $\ell$ intersects p in image 1
- $C_1$ $PC_2$ defines the **epipolar plane**

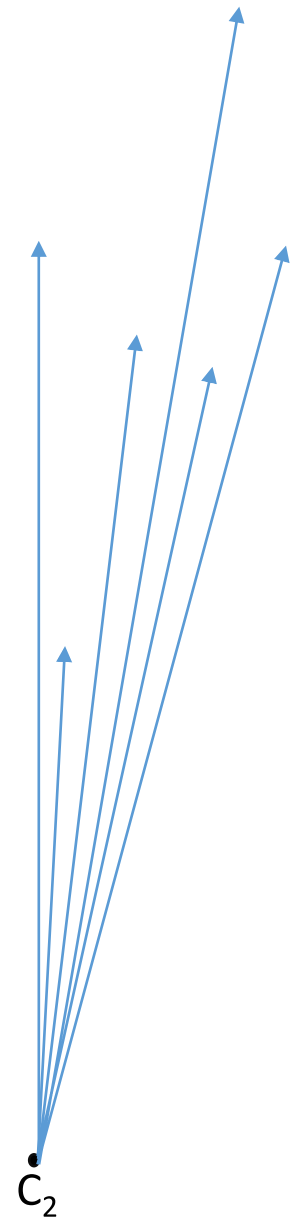


# Epipolar geometry

- What epipolar line is good for?
- If we search for a match for $p_1$
  - It will be on the epipolar line $\ell$
- If we suspect the match is wrong
  - We can decide it is an outlier if it's not on $\ell$

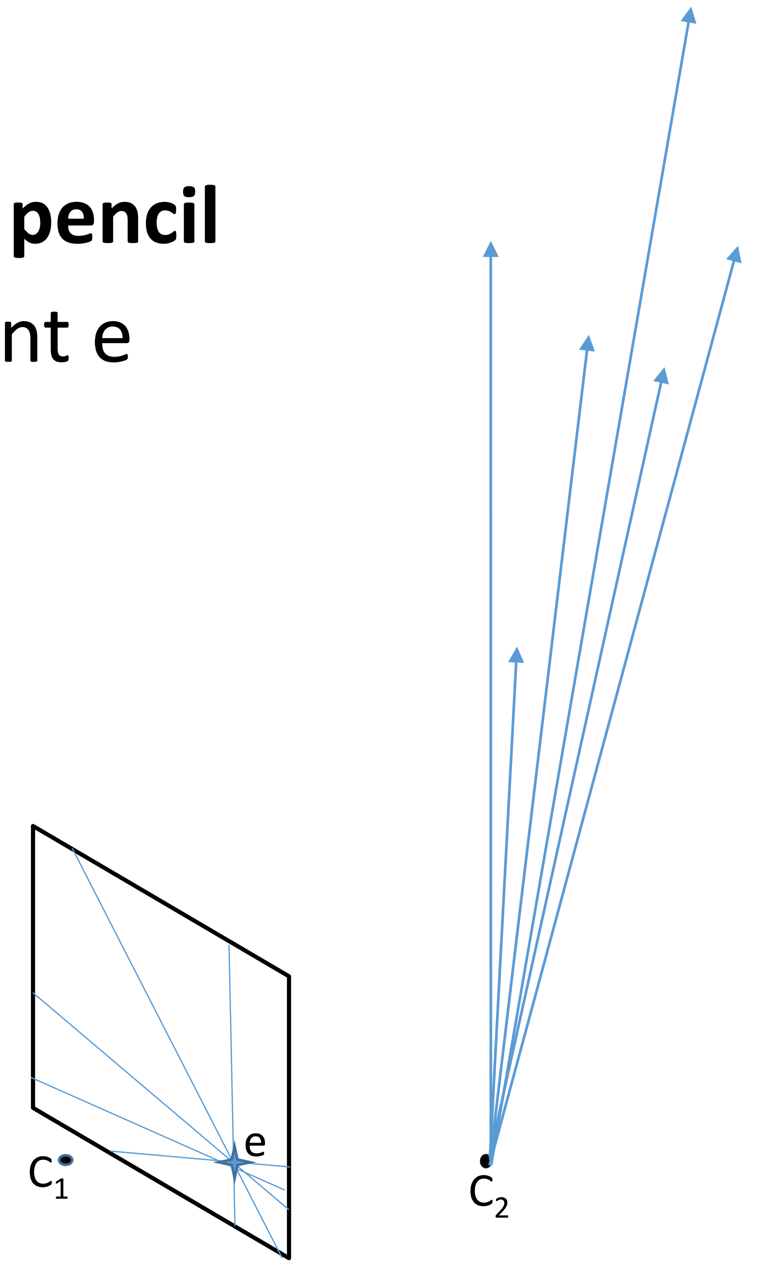
# Epipolar geometry

- **In 3D:** All 3D rays coming from $C_2$ create a **pencil**



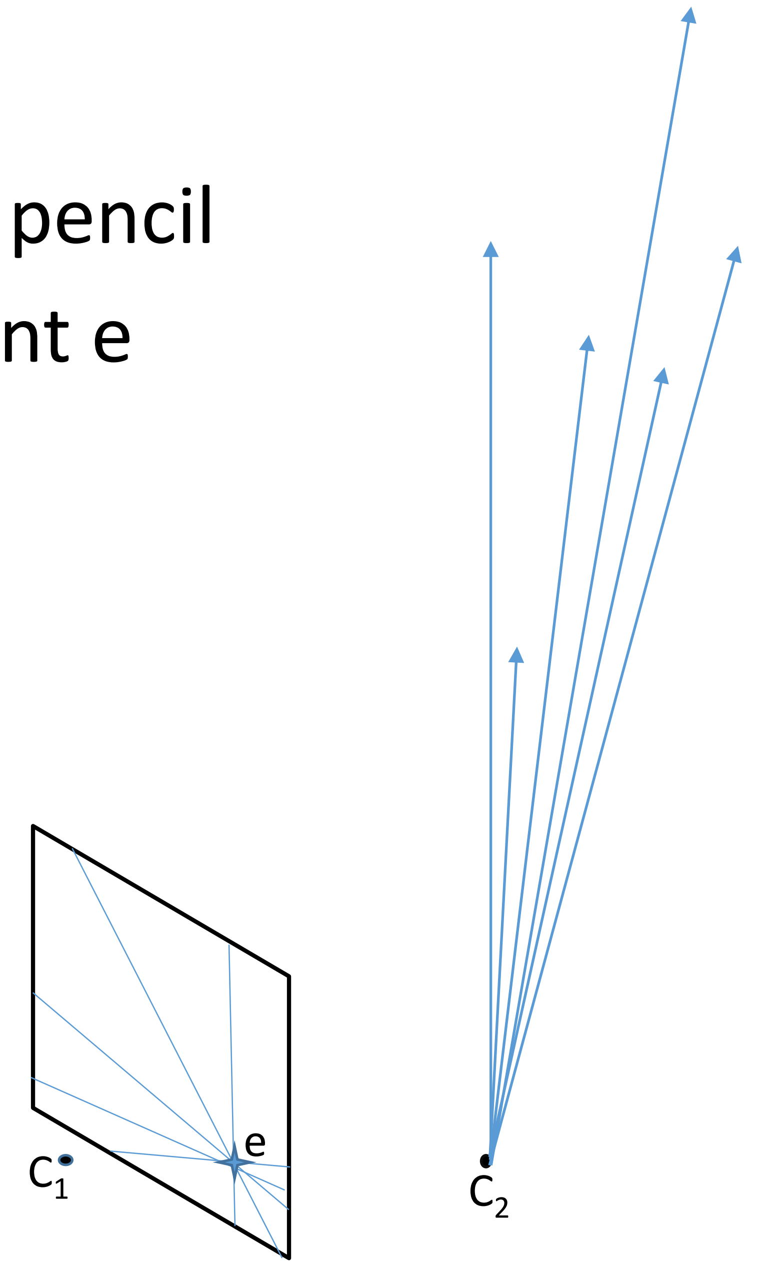
# Epipolar geometry

- **In 3D:** All 3D rays coming from $C_2$ create a **pencil**
- **In image**: all epipolar lines intersect at point e



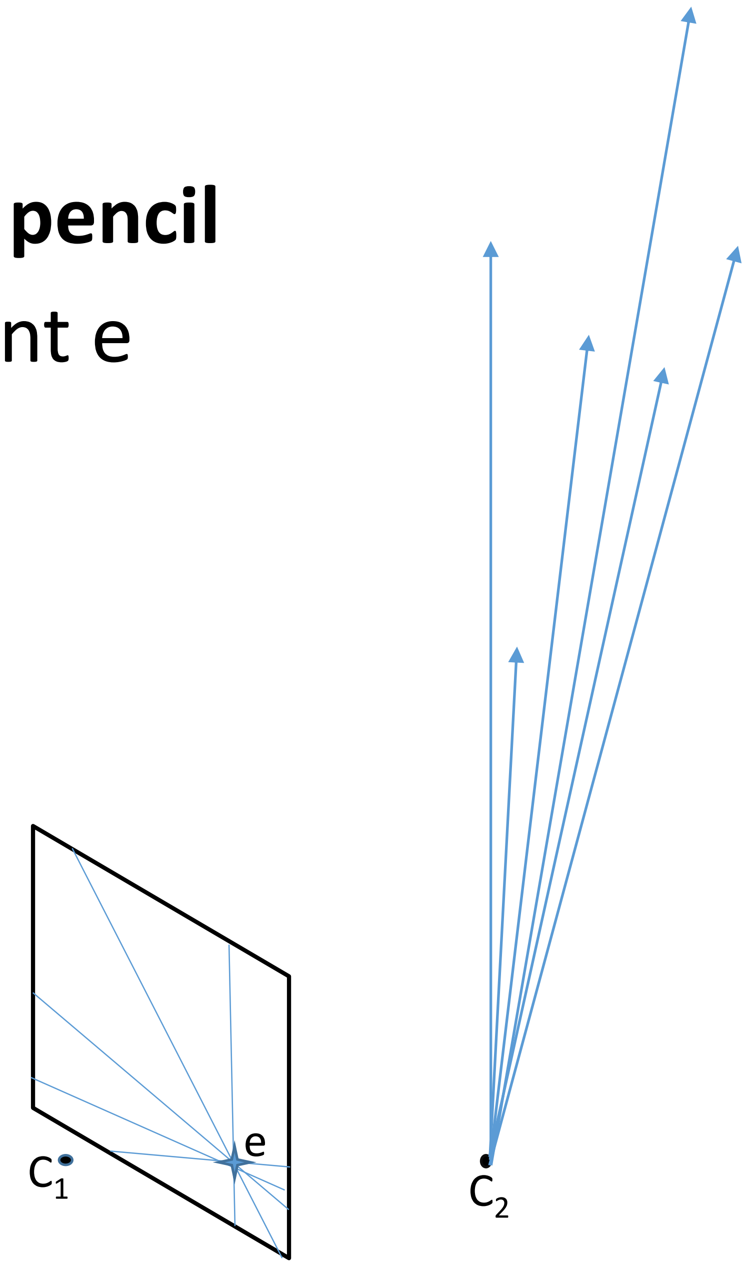
# Epipolar geometry

- **In 3D:** All 3D rays coming from $C_2$ create a pencil
- **In image**: all epipolar lines intersect at point e
- e is the **epipole**



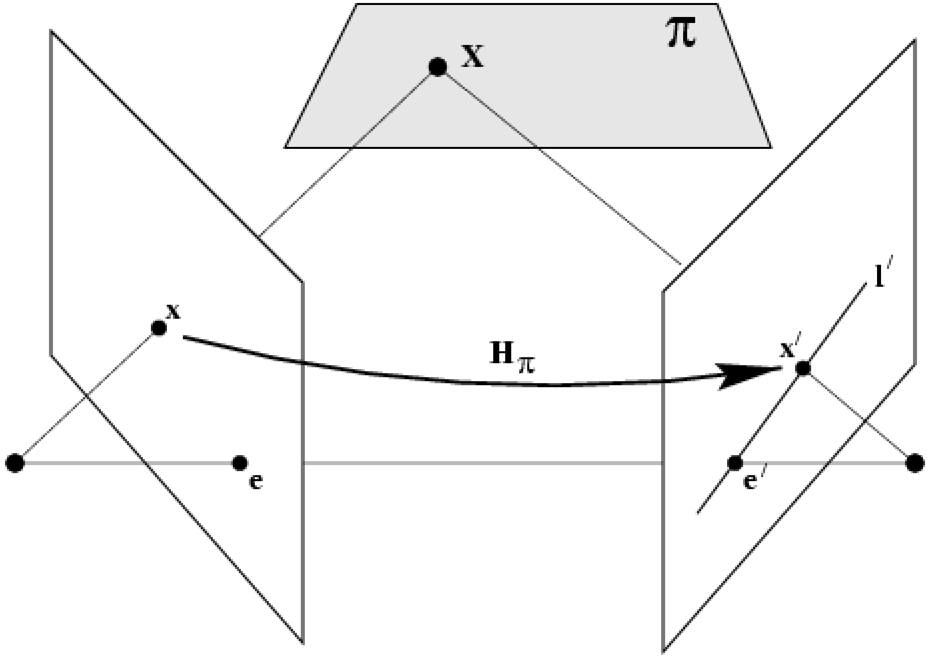
# Epipolar geometry

- **In 3D:** All 3D rays coming from $C_2$ create a **pencil**
- **In image**: all epipolar lines intersect at point e
- e is the **epipole**
- e is the projection of $c_2$ 3D location



# Epipolar geometry

geometric derivation of F:




$$x' = H_\pi x$$

$$l' = e' \times x' = [e']_\times H_\pi x = Fx$$

Courtesy of Marc Pollefeys

# Epipolar geometry

- If we don't know $\mathbf{K}_1$, $\mathbf{K}_2$, $\mathbf{R}$, or $\mathbf{t}$, can we still estimate $\mathbf{F}$?
- Yes, given enough correspondences.
- **Many algorithms:**
  - Linear (the normalized 8-point algorithm)
  - Minimal (7-point)
  - Robust (RANSAC)
  - Non-linear refinement (MLE, Algebraic minimization)
- We use 8-point algorithm
  - Although it's inaccurate
  - Because it's fast

# 8-point algorithm

- The fundamental matrix F is defined by

$$\mathbf{x}'^{\mathrm{T}}\mathbf{F}\mathbf{x} = 0$$

  for any pair of matches x and x' in two images.

- Let x=$(u,v,1)^{\mathrm{T}}$ and x'=$(u',v',1)^{\mathrm{T}}$,

$$\mathbf{F} = \begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix}$$

  each match gives a linear equation

$$uu' f_{11} + vu' f_{12} + u' f_{13} + uv' f_{21} + vv' f_{22} + v' f_{23} + uf_{31} + vf_{32} + f_{33} = 0$$

# 8-point algorithm

$$
\begin{bmatrix}
u_1u_1' & v_1u_1' & u_1' & u_1v_1' & v_1v_1' & v_1' & u_1 & v_1 & 1 \\
u_2u_2' & v_2u_2' & u_2' & u_2v_2' & v_2v_2' & v_2' & u_2 & v_2 & 1 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
u_nu_n' & v_nu_n' & u_n' & u_nv_n' & v_nv_n' & v_n' & u_n & v_n & 1
\end{bmatrix}
\begin{bmatrix}
f_{11} \\ f_{12} \\ f_{13} \\ f_{21} \\ f_{22} \\ f_{23} \\ f_{31} \\ f_{32} \\ f_{33}
\end{bmatrix} = 0
$$

Courtesy of Marc Pollefeys

# 8-point algorithm

$$\begin{bmatrix} u_1u_1' & v_1u_1' & u_1' & u_1v_1' & v_1v_1' & v_1' & u_1 & v_1 & 1 \\ u_2u_2' & v_2u_2' & u_2' & u_2v_2' & v_2v_2' & v_2' & u_2 & v_2 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ u_nu_n' & v_nu_n' & u_n' & u_nv_n' & v_nv_n' & v_n' & u_n & v_n & 1 \end{bmatrix} \begin{bmatrix} f_{11} \\ f_{12} \\ f_{13} \\ f_{21} \\ f_{22} \\ f_{23} \\ f_{31} \\ f_{32} \\ f_{33} \end{bmatrix} = 0$$

$$\mathbf{A}_{\mathbf{8x9}}\ \mathbf{f}_{\mathbf{9x1}} \mathbf{=0}$$

# 8-point algorithm

- We solve it as before, using SVD decomposition:

  $Af = 0$

  $A = U\Sigma V^T$

  $f = V^T{}_N$

- We can use more than 8 points. M>N=8
  - But now, instead of solving $\mathbf{Af} = 0$, we seek $\mathbf{f}$ to minimize $\|\mathbf{Af}\|$, least eigenvector of $\mathbf{A}^{\mathrm{T}}\mathbf{A}$.
  - Still, we take $f = V^T{}_N$

# 8-point algorithm

- **Problem: F** should have rank 2. It doesn't.
- To enforce that **F** is of rank 2, F is replaced by F' that minimizes $\|\mathbf{F} - \mathbf{F}'\|$ subject to the rank constraint.

- This too is achieved by SVD. Let $\mathbf{F} = \mathbf{U}\Sigma\mathbf{V}^{\mathrm{T}}$, where

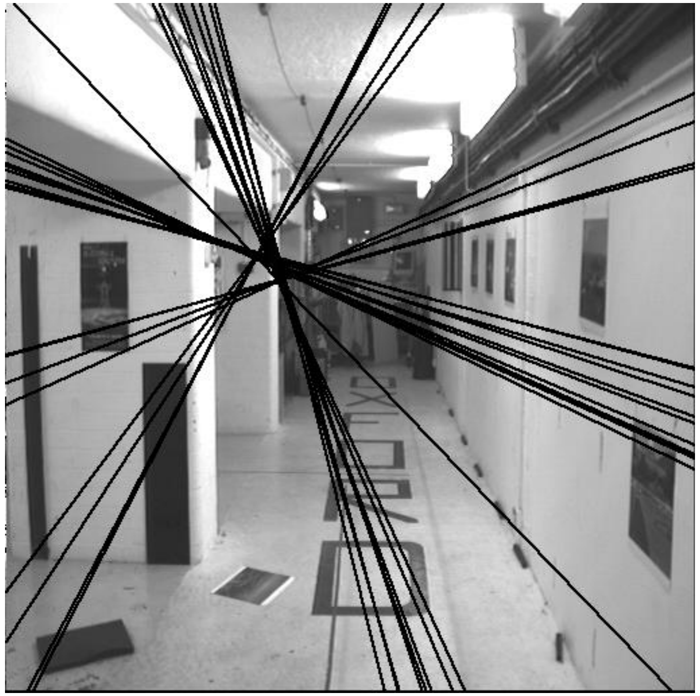
$$\Sigma = \begin{bmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \\ 0 & 0 & \sigma_3 \end{bmatrix} \qquad \Sigma' = \begin{bmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

then $\mathbf{F}' = \mathbf{U}\Sigma'\mathbf{V}^{\mathrm{T}}$ is the solution.

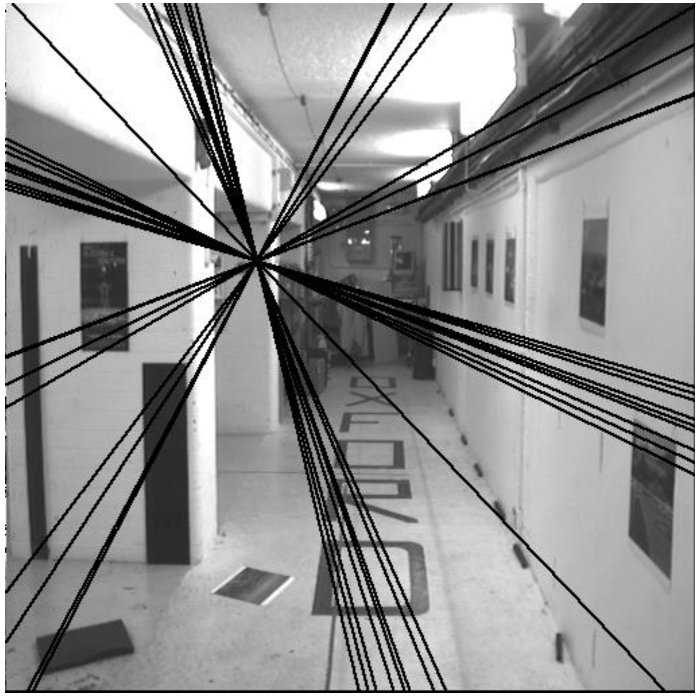
Courtesy of Marc Pollefeys

# 8-point algorithm

Before



After



Courtesy of Marc Pollefeys

# 8-point algorithm

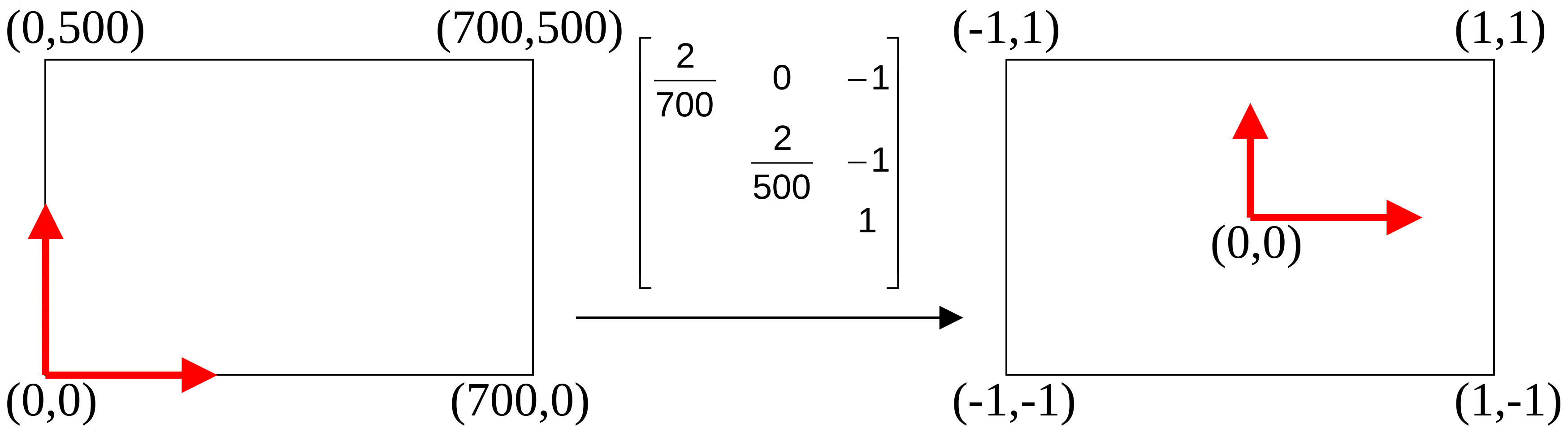
$$
\begin{array}{c}
\begin{bmatrix}
u_1u_1' & v_1u_1' & u_1' & u_1v_1' & v_1v_1' & v_1' & u_1 & v_1 & 1 \\
u_2u_2' & v_2u_2' & u_2' & u_2v_2' & v_2v_2' & v_2' & u_2 & v_2 & 1 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
u_nu_n' & v_nu_n' & u_n' & u_nv_n' & v_nv_n' & v_n' & u_n & v_n & 1
\end{bmatrix} \\
\begin{matrix} \sim 10000 & \sim 10000 & \sim 100 & \sim 10000 & \sim 10000 & \sim 100 & \sim 100 & \sim 100 & 1 \end{matrix}
\end{array}
\begin{bmatrix} f_{11} \\ f_{12} \\ f_{13} \\ f_{21} \\ f_{22} \\ f_{23} \\ f_{31} \\ f_{32} \\ f_{33} \end{bmatrix} = 0
$$

Orders of magnitude difference
between column of data matrix
$\rightarrow$ least-squares yields poor results

Courtesy of Marc Pollefeys

# 8-point algorithm

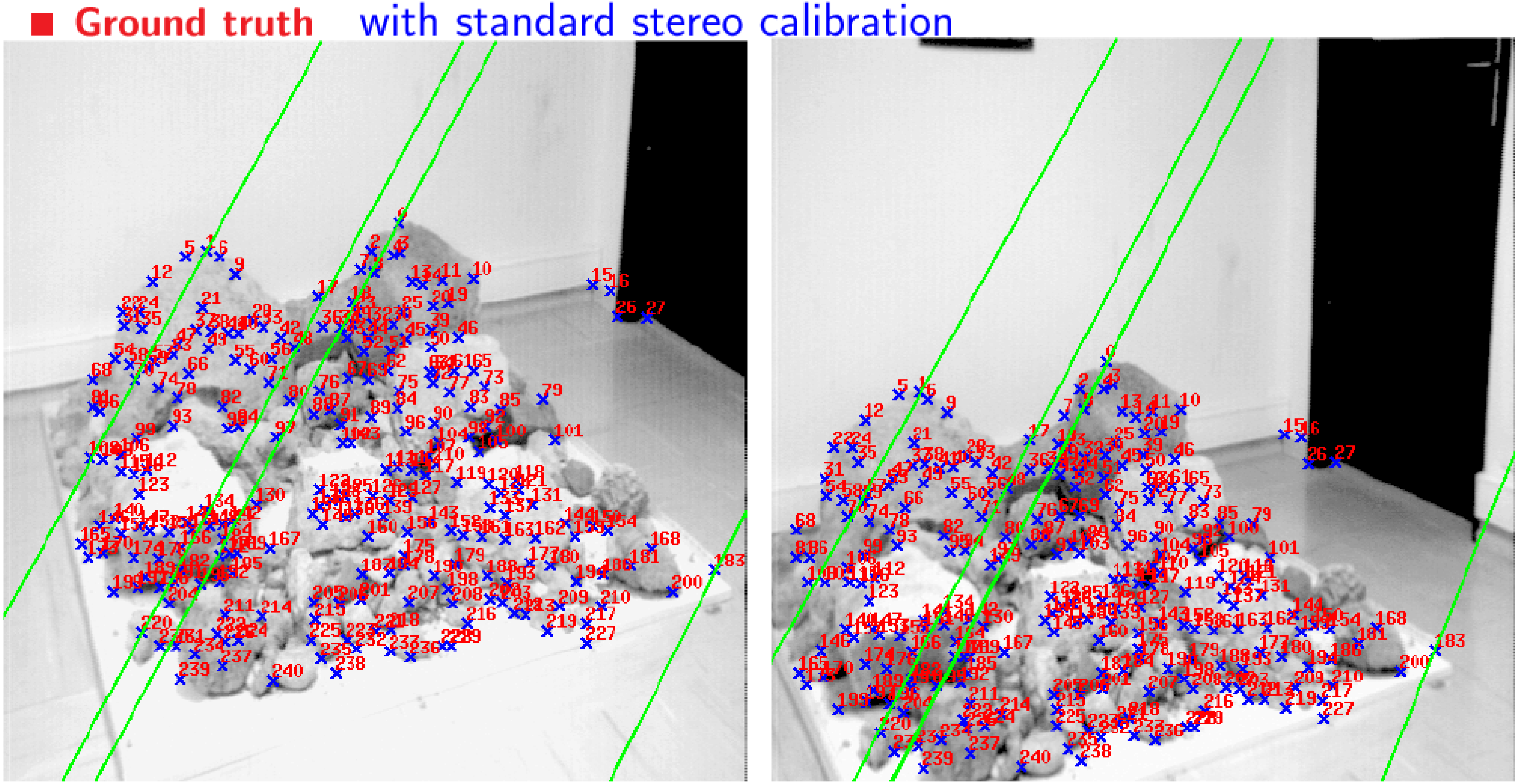
- normalized least squares yields good results
- Transform image to ~[-1,1]x[-1,1]




Courtesy of Marc Pollefeys

# 8-point algorithm

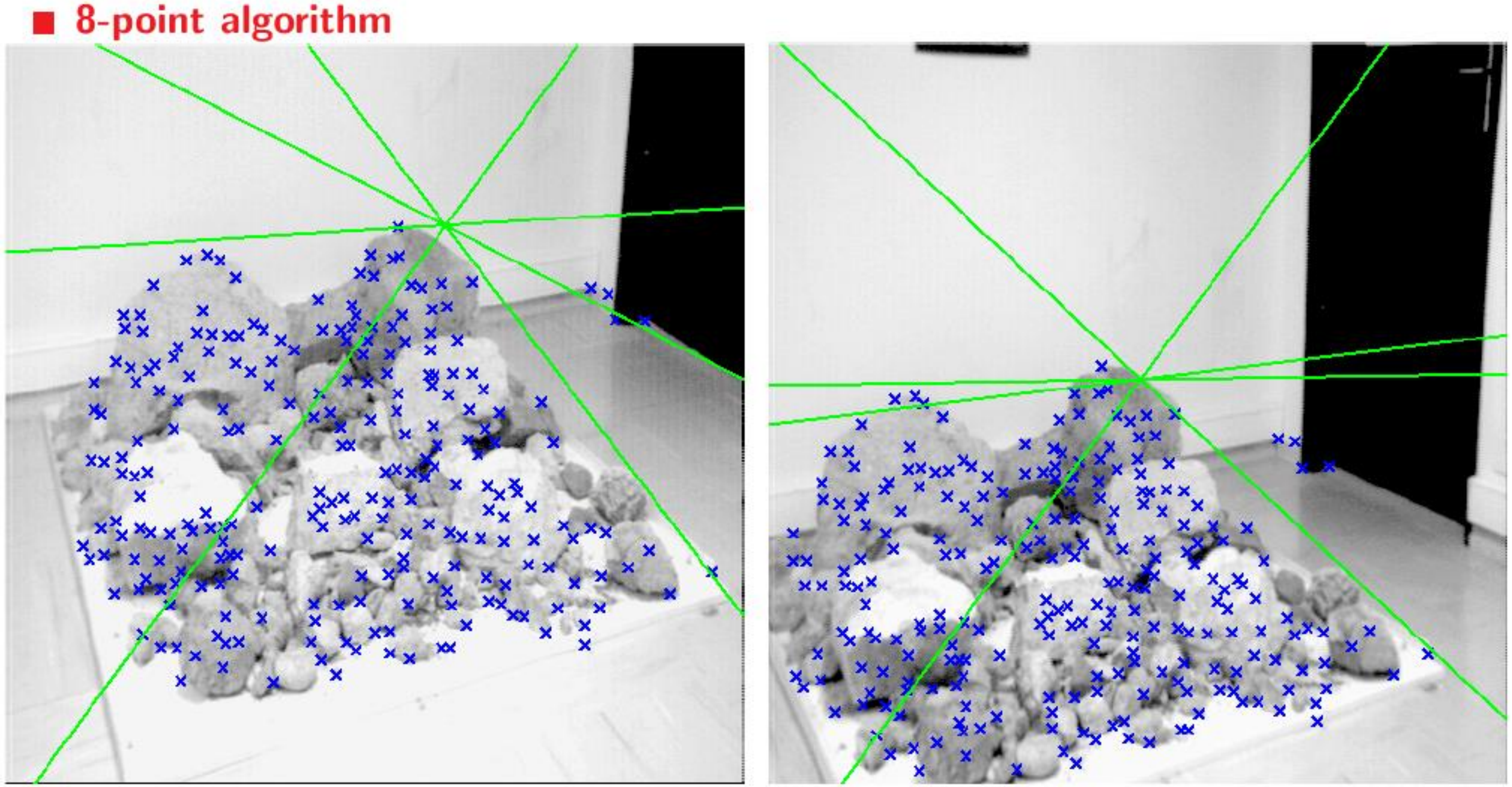
## Results (ground truth)




Courtesy of Marc Pollefeys

# 8-point algorithm

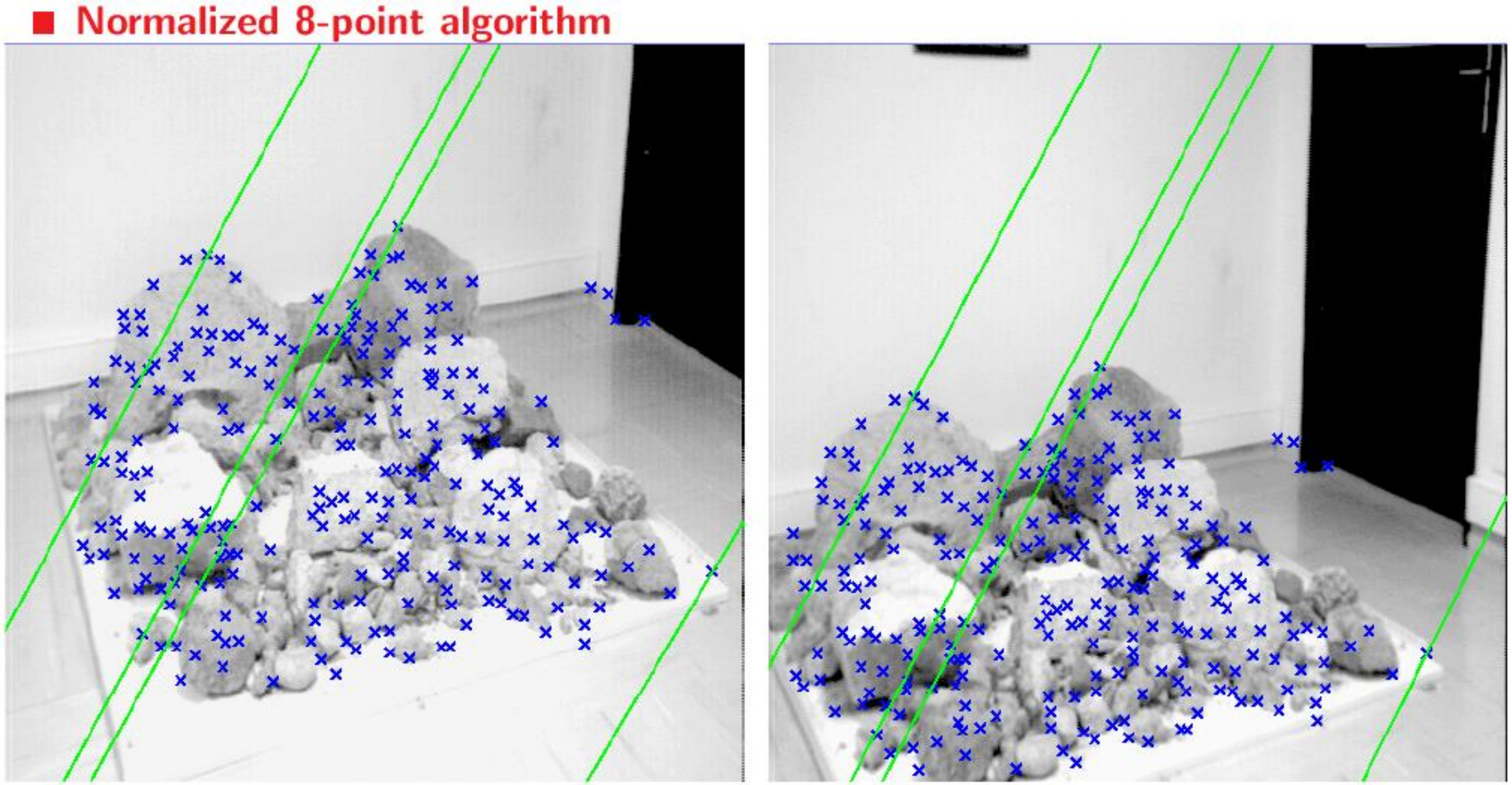
## Results (8-point algorithm)




Courtesy of Marc Pollefeys

# 8-point algorithm

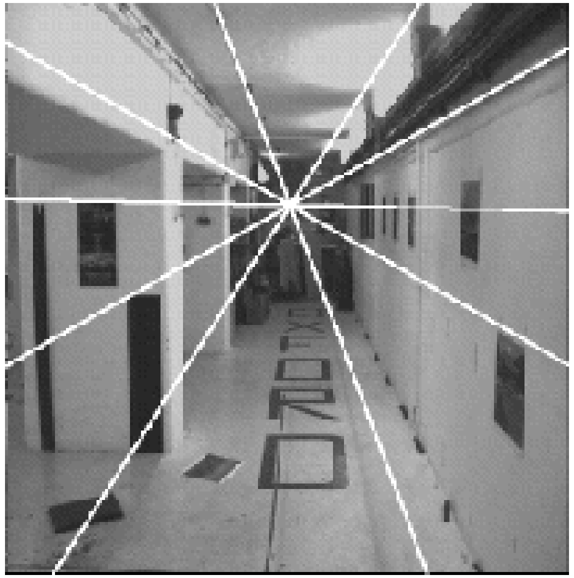
## Results (normalized 8-point algorithm)

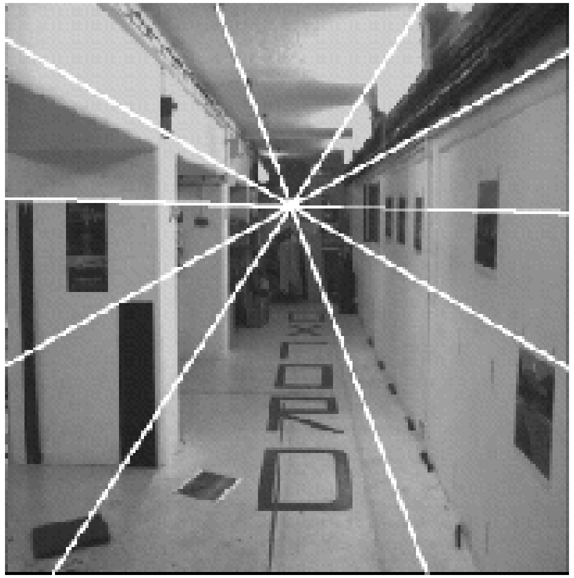



Courtesy of Marc Pollefeys

# Epipolar geometry

**Example: forward motion**

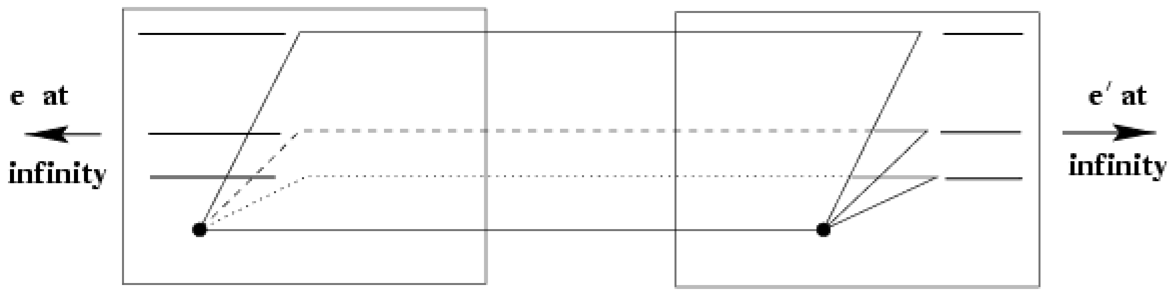



Courtesy of Marc Pollefeys

# Epipolar geometry

**Example: motion parallel with image plane**




Courtesy of Marc Pollefeys

# Epipolar geometry

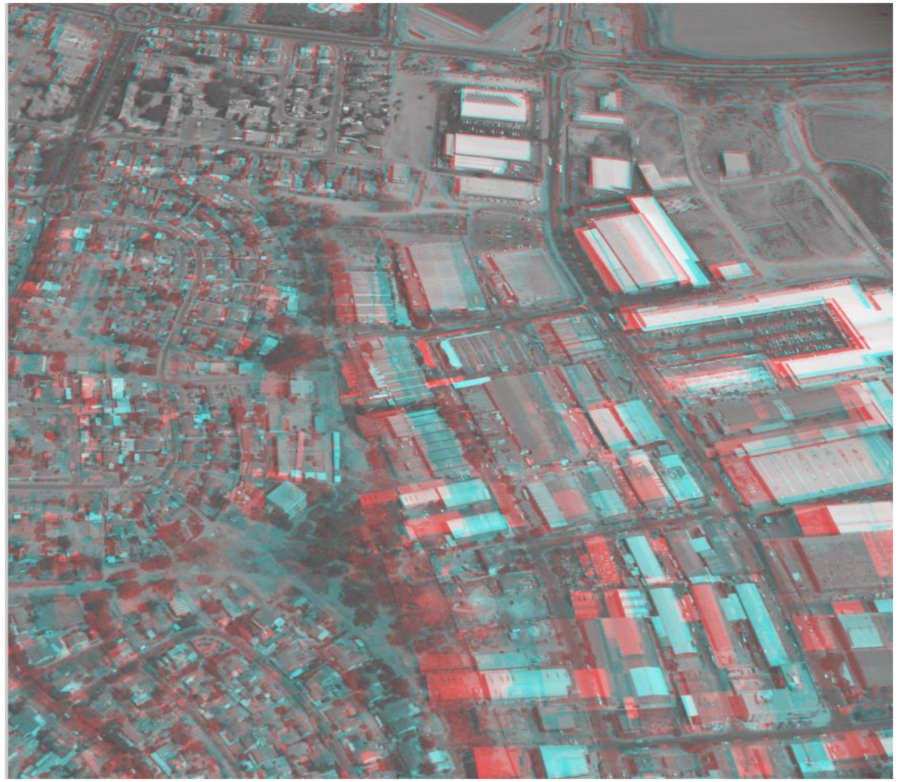
- In stereo rectification

  We wish all epipolar lines to be:

  $$F(x,y,1)^T \rightarrow (0,\ 1,\ -y)$$

  So F is from the shape:

  $$[t]_x = \begin{bmatrix}1\\0\\0\end{bmatrix}_x \begin{bmatrix} & & \\ & & 1\\ & -1 & \end{bmatrix}$$



# RANSAC

- RANdom SAmple Consensus
- Problem:
  - All inliers obey some model
  - But there are some unknown outliers.
- Example: inliers are on a curve
- Chicken and egg situation:
  - If we had the curve, we could spot the outliers
  - If we knew the inliers, we could estimate the curve
- Key to solution:
  - The model can be estimated using a small set


Courtesy of: ET Wales

# RANSAC

- Algorithm:
  1. Repeat:
     1. Sample a minimal set
     2. Estimate a model
     3. Check how many points obey the model
  2. Choose model with maximal #points
  3. Repeat:
     1. Estimate model from all inliers
     2. Calc inliers of new model
- Output: inliers, outliers, and model

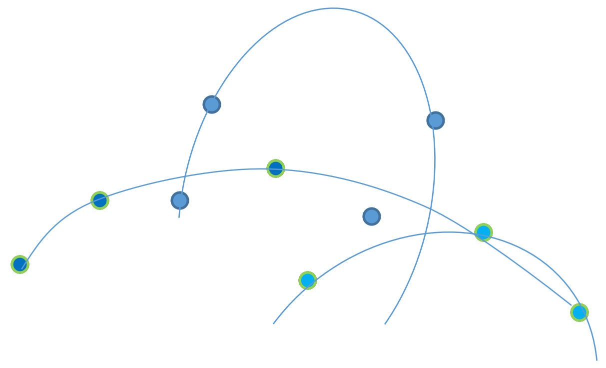
# RANSAC

- Example:
- #inliers = 3. max #inliers = 3
- #inliers = 5. **max #inliers = 5**
- #inliers = 3. max #inliers = 5

.

.

- Output: #inliers = 5



# RANSAC

- When to stop the first loop of RANSAC?
- Goal: one sample that will have only inliers, with high prob p.
- Prob of being an outlier: $\epsilon$
- P(being an inlier) = $1-\epsilon$
- P(all inliers-sample) = $(1-\epsilon)^{s}$
- P(bad sample)= $1-(1-\epsilon)^{s}$
- P(All samples are bad) = $(1-(1-\epsilon)^{s})^{I}$
- We wish it to be small: $(1-(1-\varepsilon)^{s})^{I} < 1-p$

$$\log(1-(1-\varepsilon)^{s})^{I} < \log(1-p)$$

$$I\log(1-(1-\varepsilon)^{s}) < \log(1-p)$$

$$I > \log(1-p)/\log(1-(1-\varepsilon)^{s})$$

# RANSAC

- This can be really high:

| **s \ $\epsilon$** | **25%** | **50%** | **60%** | **70%** | **80%** | **85%** |
|---|---|---|---|---|---|---|
| **2** | 6 | 16 | 26 | 49 | 113 | 202 |
| **3** | 8 | 34 | 70 | 168 | 573 | 1362 |
| **7** | 33 | 588 | 2808 | 21055 | 2.5E05 | 2.6E06 |

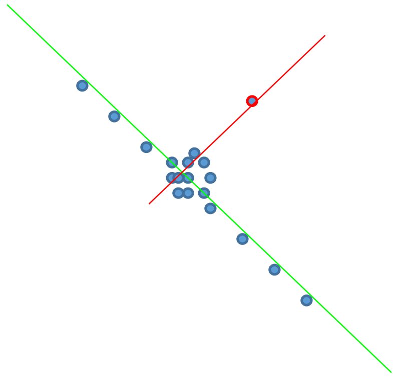
- What if we don't know $\varepsilon$?
- We can estimate it online:
  - We calc #inliers at each sample
  - This gives an ever-decreasing upper-bound on $\varepsilon$
  - Hence the needed iteration number $I$ is also decreasing

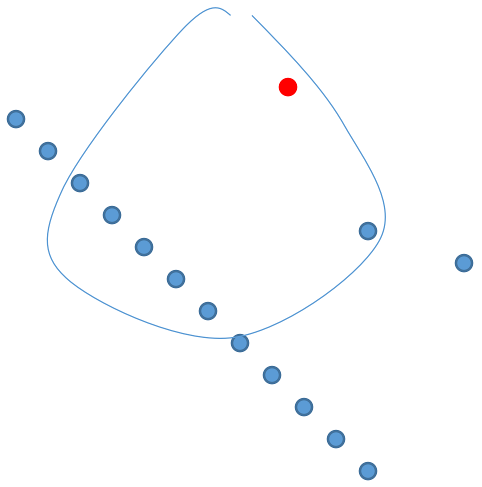
# RANSAC

- Which models are used with RANSAC?
- 2D points matching:
  - Fundamental matrix
  - Homography transformation
  - Essential Matrix
  - Trifocal Tensor
- 3D points:
  - Point cloud registration
  - Perspective-n-Point (PNP)
  - Plane fitting
  - Curve fitting

# RANSAC

- Limitations of RANSAC with FM:
  - Efficiency: unknown
    - because outliers ratio $\varepsilon$ in unknown
  - Accuracy
    - Even good sample may give a bad model
    - Sensitive to inlier threshold
  - Degeneracy
    - The plain+paralax problem
  - Many tricks and extensions:
    - PROSAC
    - USAC





# Thanks

# 3D points cloud registration

- In the matched case
  - Each 3D point in X have a correspondence in Y)

$$y_{3x1} = R_{3x3} x_{3x1} + t_{3x1}$$

$$\tilde{Y} = Y - \overline{Y}, \tilde{X} = X - \overline{X}$$

$$t = \overline{X} - \overline{Y}$$

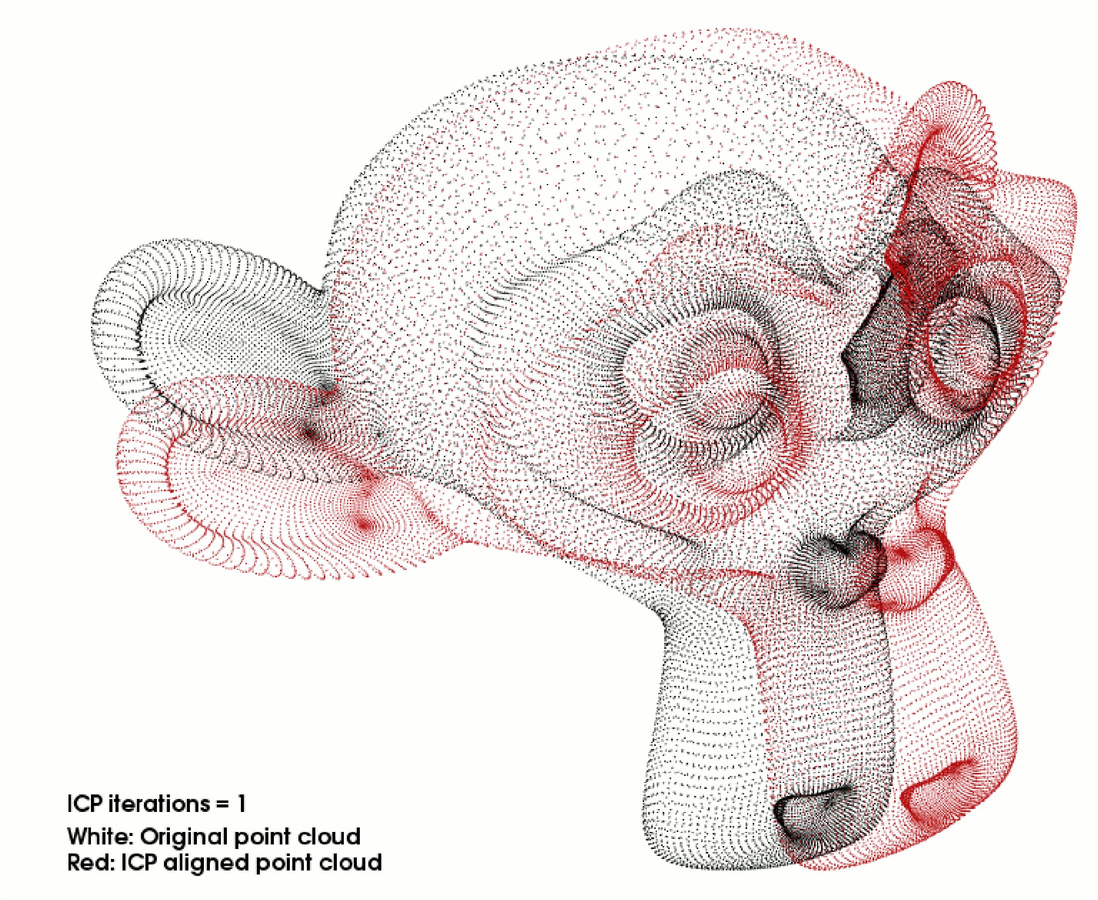
$$\tilde{Y}_{3xN} = R_{3x3} \tilde{X}_{3xN}$$

$$U_{3x3} \Sigma_{3x3} V^T_{3x3} = SVD(X^T{}_{Nx3} Y_{Nx3})$$

$$R = V^T U$$

# 3D points cloud registration

- In the not-matched case
- The ICP – Iterative Closest Point algorithm
  - Repeat until convergence:
    - Find temporary matches:
    - For each point in X:
    - Set the closest point in Y to match it
  - Calculate R and t using the SVD algorithm above
    - May use RANSAC for outlier removal
  - Transform X using R and t


ICP iterations = 1
White: Original point cloud
Red: ICP aligned point cloud