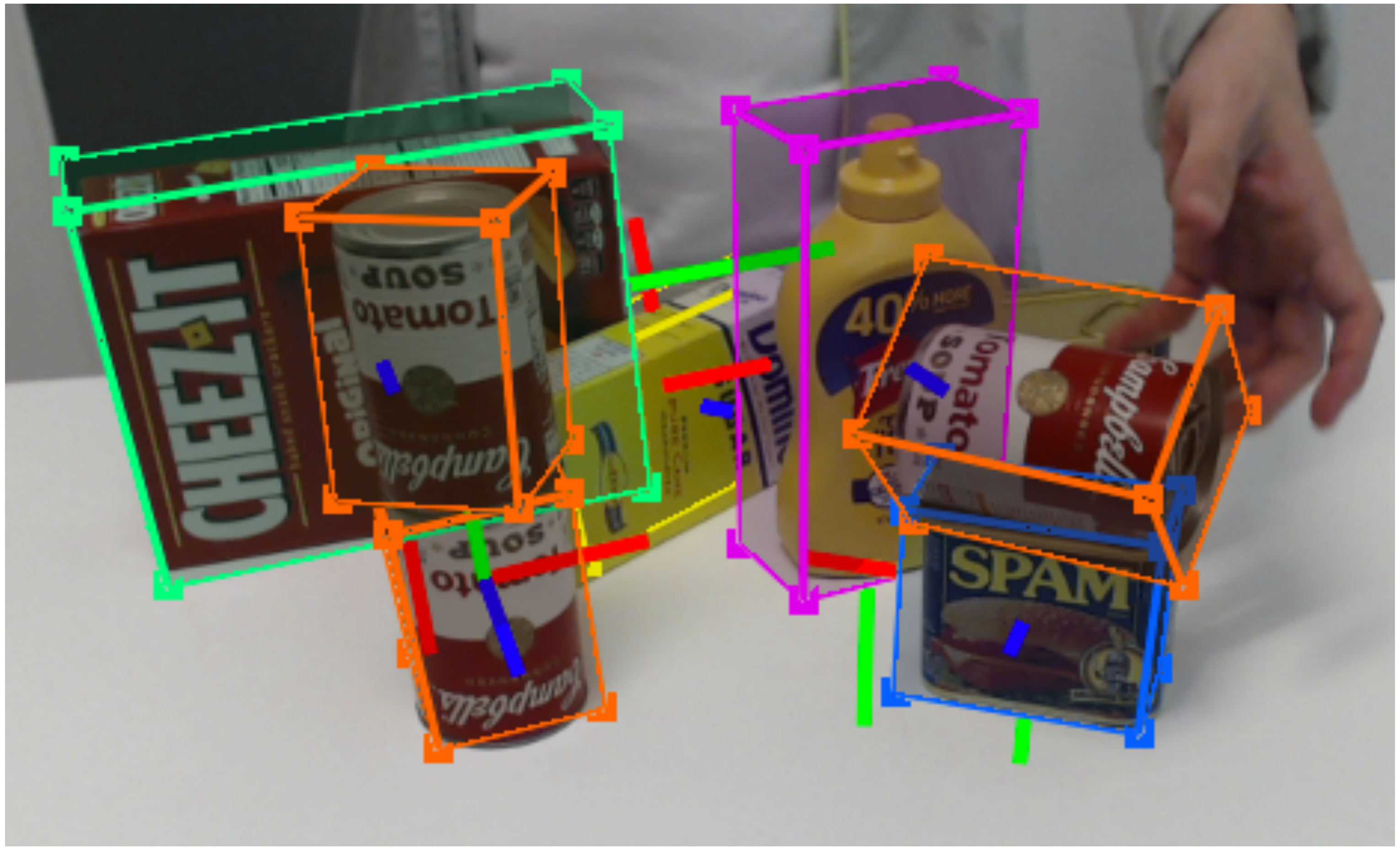
6.2 Pose Estimation

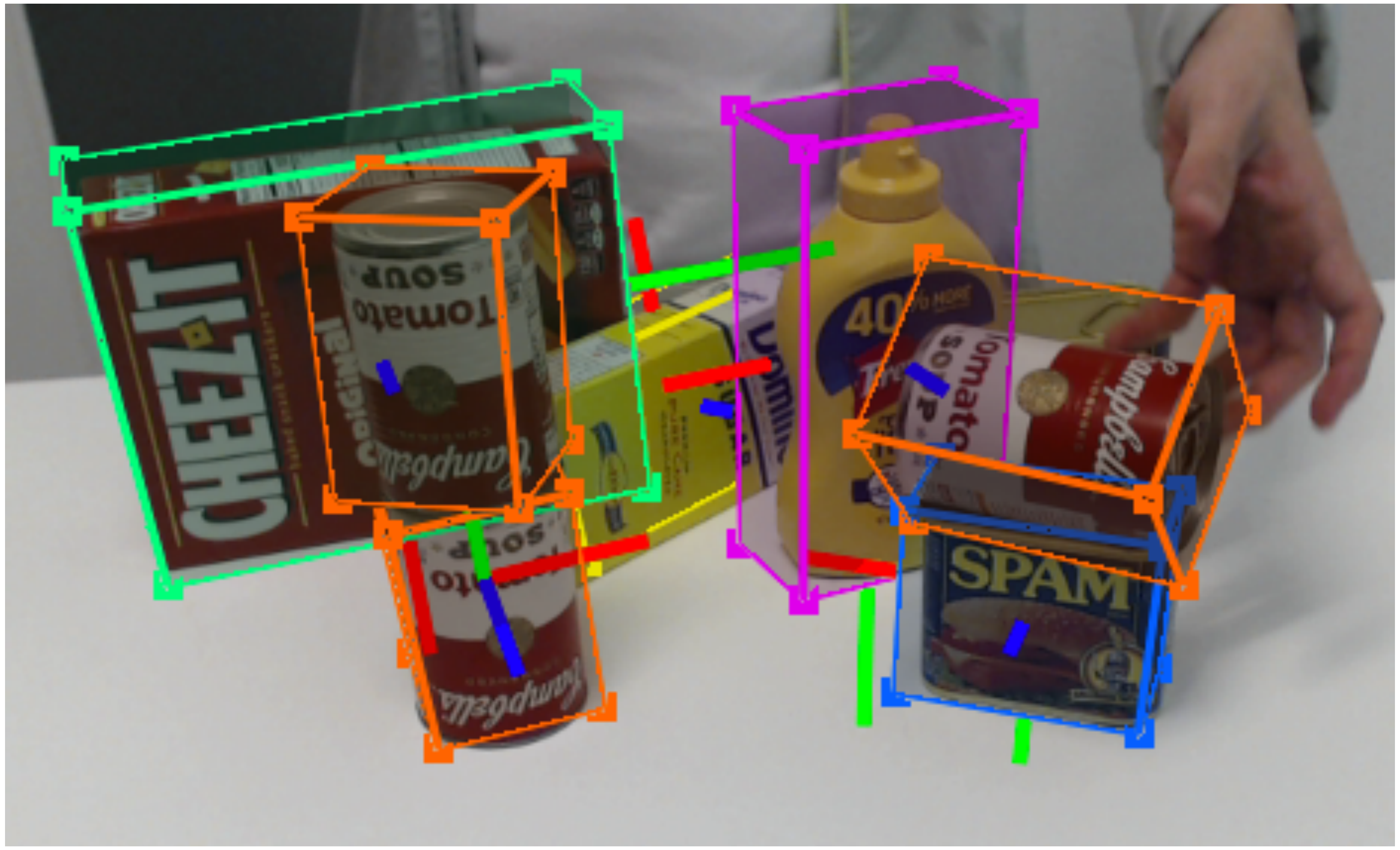
Problem Definition

- Given:
  - A 3D model of an object
  - An image of the object
- Estimate:
  - The 3D pose of the object relative to the camera



Perspective 3-Point Problem

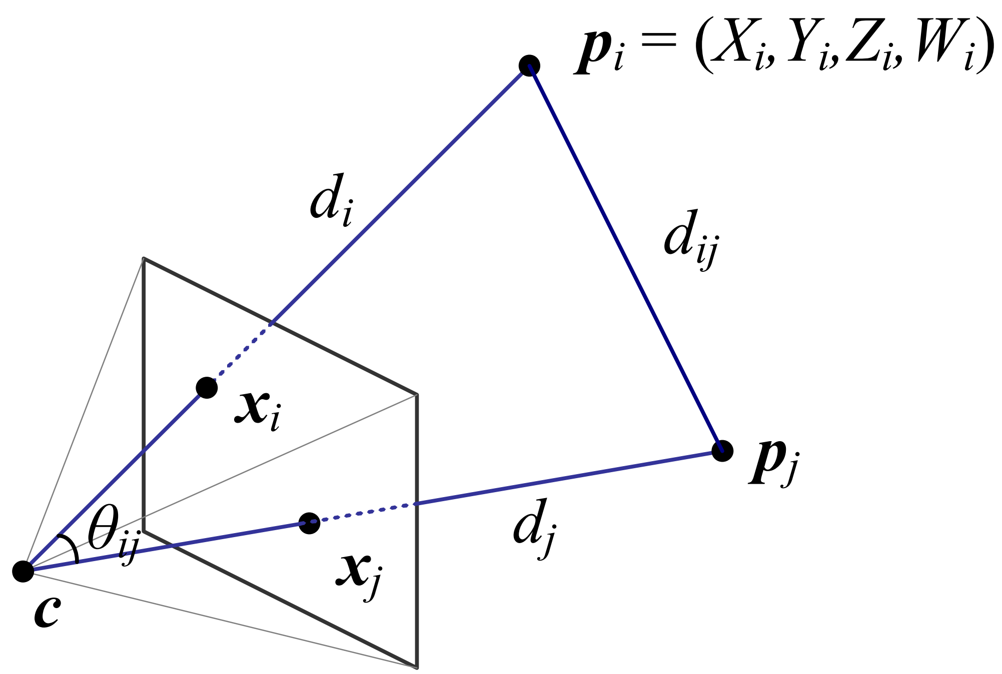
- How many degrees of freedom (parameters) are we estimating?
- How many point correspondences between 3D object and 2D image do we need?



Linear Algorithms

- 3 x 4 camera projection matrix $\boldsymbol{P}$

$$\tilde{\boldsymbol{x}}_s = \boldsymbol{K} \left[\begin{array}{c|c} \boldsymbol{R} & \boldsymbol{t} \end{array} \right] \bar{\boldsymbol{p}}_w = \boldsymbol{P} \bar{\boldsymbol{p}}_w$$



$$\boldsymbol{P} = \begin{bmatrix} p_{00} & p_{01} & p_{02} & p_{03} \\ p_{10} & p_{11} & p_{12} & p_{13} \\ p_{20} & p_{21} & p_{22} & p_{23} \end{bmatrix} \Rightarrow$$

$$x_i = \frac{p_{00}X_i + p_{01}Y_i + p_{02}Z_i + p_{03}}{p_{20}X_i + p_{21}Y_i + p_{22}Z_i + p_{23}}$$

$$y_i = \frac{p_{10}X_i + p_{11}Y_i + p_{12}Z_i + p_{13}}{p_{20}X_i + p_{21}Y_i + p_{22}Z_i + p_{23}}$$

Linear Algorithms

$$x_i = \frac{p_{00}X_i + p_{01}Y_i + p_{02}Z_i + p_{03}}{p_{20}X_i + p_{21}Y_i + p_{22}Z_i + p_{23}}$$

$$y_i = \frac{p_{10}X_i + p_{11}Y_i + p_{12}Z_i + p_{13}}{p_{20}X_i + p_{21}Y_i + p_{22}Z_i + p_{23}}$$

- As for estimation of 2D homographies, we can form a linear estimate of the parameters p_{ij} by multiplying through by the denominator, which yields

$$\begin{bmatrix} X_i & Y_i & Z_i & 1 & 0 & 0 & 0 & 0 & -x_iX_i & -x_iY_i & -x_iZ_i & -x_i \\ 0 & 0 & 0 & 0 & X_i & Y_i & Z_i & 1 & -y_iX_i & -y_iY_i & -y_iZ_i & -y_i \end{bmatrix} \begin{bmatrix} p_{00} \\ p_{01} \\ \vdots \\ p_{23} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

- How many pairs of matching points do we need?

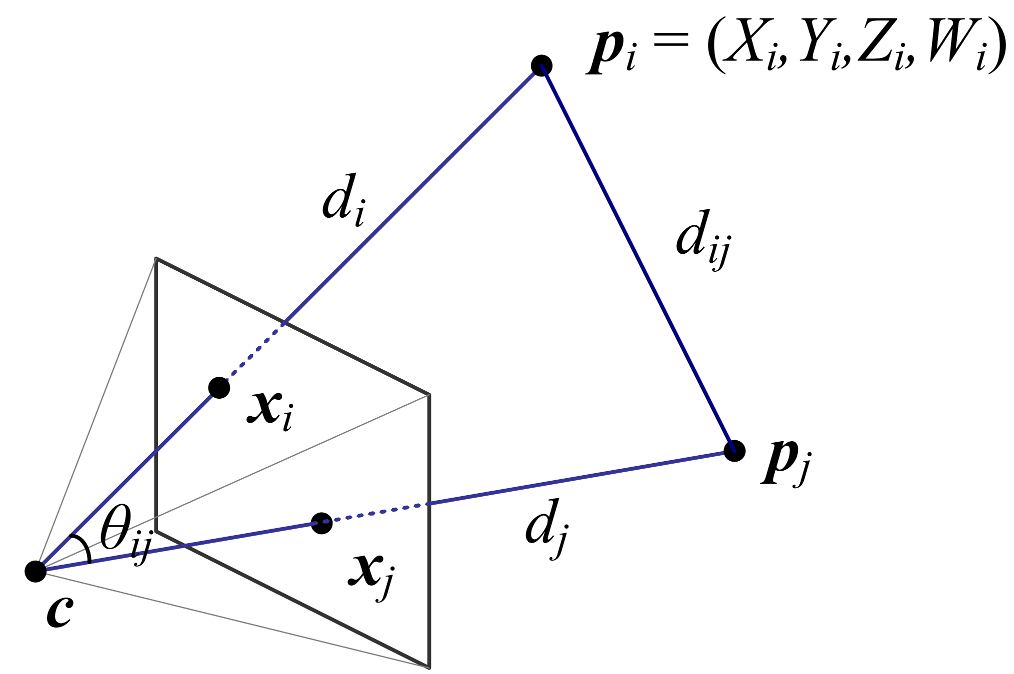
- Again, this estimate does not minimized the squared deviation but can be used as an initial guess for an iterative solution.

Linear Algorithms

❖ 3 x 4 camera projection matrix $\boldsymbol{P}$

$$\tilde{\boldsymbol{x}}_s = \boldsymbol{K} \left[\begin{array}{c|c} \boldsymbol{R} & \boldsymbol{t} \end{array} \right] \bar{\boldsymbol{p}}_w = \boldsymbol{P}\bar{\boldsymbol{p}}_w$$

$$\boldsymbol{P} = \begin{bmatrix} p_{00} & p_{01} & p_{02} & p_{03} \\ p_{10} & p_{11} & p_{12} & p_{13} \\ p_{20} & p_{21} & p_{22} & p_{23} \end{bmatrix}$$



❖ Once $\boldsymbol{P}$ has been estimated, its constituents $\boldsymbol{K}$, $\boldsymbol{R}$ and $\boldsymbol{t}$ can be recovered.

❖ Recall that $\boldsymbol{R}$ is orthonormal and $\boldsymbol{K}$ is normally treated as upper triangular:

$$\boldsymbol{K} = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

f_x and f_y: encode focal length and pixel spacing, which may be slightly different in x and y dimensions.

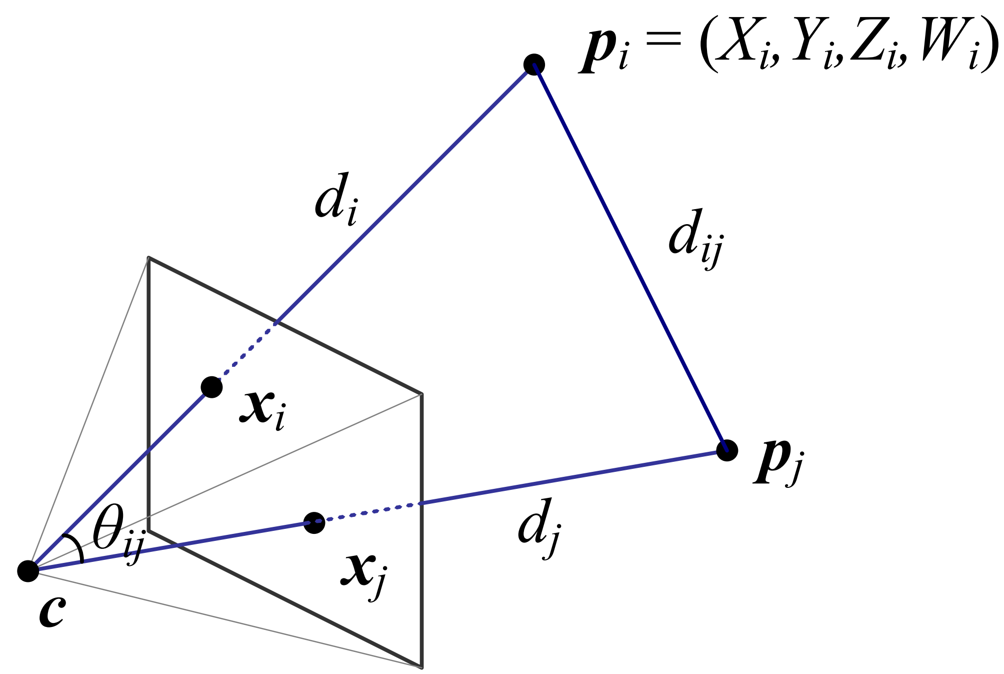
c_x and c_y: encode principal point (intersection of optic axis with sensor plane) - usually very close to centre of image

s: encodes possible skew between sensor axes (usually close to 0).

Linear Algorithms

$$\tilde{\boldsymbol{x}}_s = \boldsymbol{K} \left[\begin{array}{c|c} \boldsymbol{R} & \boldsymbol{t} \end{array} \right] \bar{\boldsymbol{p}}_w = \boldsymbol{P} \bar{\boldsymbol{p}}_w$$

$$\boldsymbol{P} = \begin{bmatrix} p_{00} & p_{01} & p_{02} & p_{03} \\ p_{10} & p_{11} & p_{12} & p_{13} \\ p_{20} & p_{21} & p_{22} & p_{23} \end{bmatrix} \qquad \boldsymbol{K} = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$



- Thus $\boldsymbol{K}$ and $\boldsymbol{R}$ can be recovered from the first 3 columns of $\boldsymbol{P}$ using QR decomposition.

  Complexity: $O\left(MN^2 + N^3\right)$ for an $M \times N$ matrix (3×4 in our case).

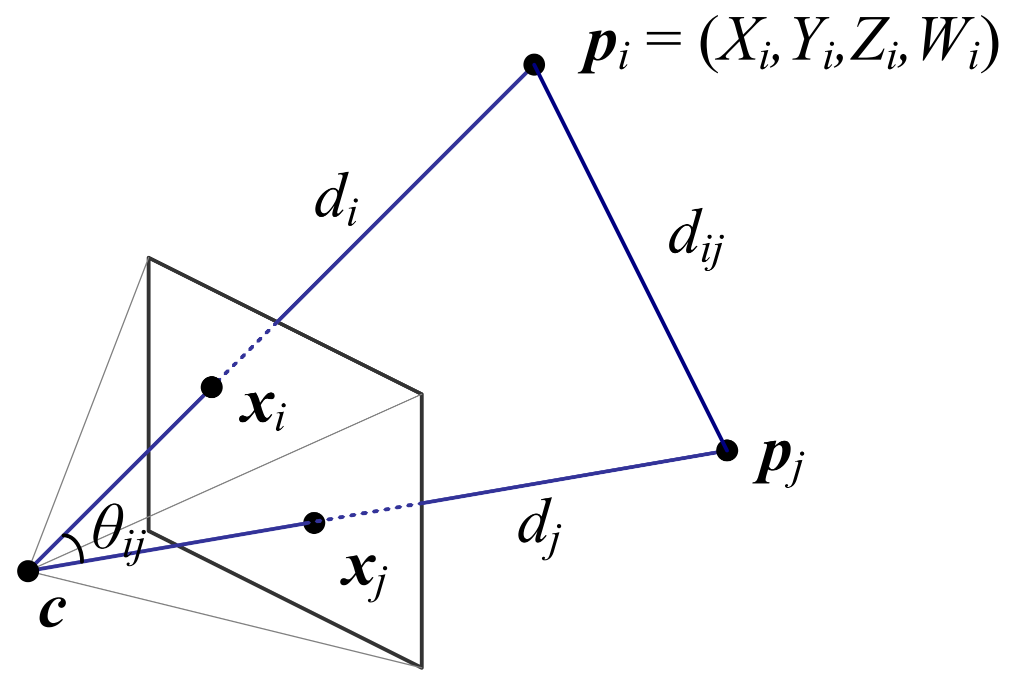
  $\boxed{\text{MATLAB function qr(A)}}$

Linear Algorithms

$$\tilde{\boldsymbol{x}}_s = \boldsymbol{K} \left[\begin{array}{c|c} \boldsymbol{R} & \boldsymbol{t} \end{array} \right] \bar{\boldsymbol{p}}_w = \boldsymbol{P}\bar{\boldsymbol{p}}_w$$

$$\boldsymbol{P} = \begin{bmatrix} p_{00} & p_{01} & p_{02} & p_{03} \\ p_{10} & p_{11} & p_{12} & p_{13} \\ p_{20} & p_{21} & p_{22} & p_{23} \end{bmatrix}$$

$$\boldsymbol{K} = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$



- Given a calibrated camera (***K*** known), ***R*** and ***t*** can be recovered with as few as 3 matched points
- Basic idea: visual angle between any pair of 2D points $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ in the image must be the same as the visual angle between their corresponding 3D points $\boldsymbol{p}_i$ and $\boldsymbol{p}_j$.

Linear Algorithms

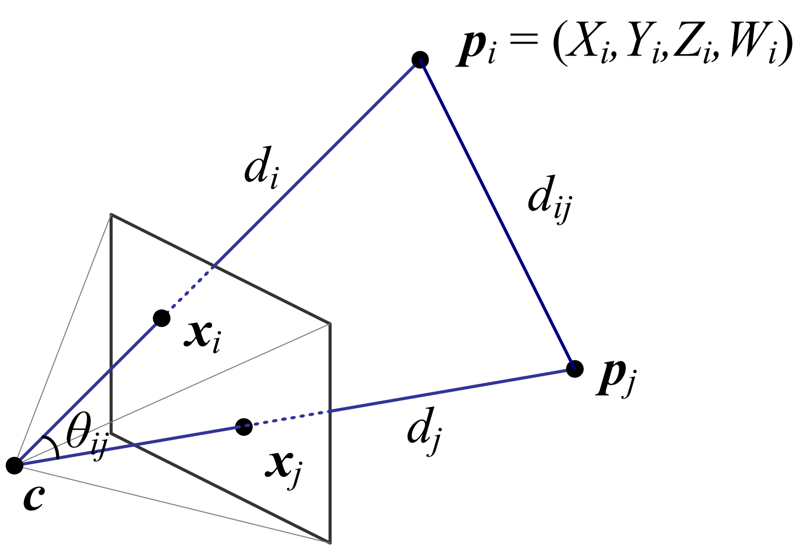
- Basic idea: visual angle between any pair of 2D points $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ in the image must be the same as the visual angle between their corresponding 3D points $\boldsymbol{p}_i$ and $\boldsymbol{p}_j$.

  Let $\hat{\boldsymbol{x}}_i$ represent the unit vector pointing to image point $\boldsymbol{x}_i$ from the camera centre $\boldsymbol{c}$:

  $\hat{\boldsymbol{x}}_i = \mathcal{N}(\boldsymbol{K}^{-1}\boldsymbol{x}_i) = \boldsymbol{K}^{-1}\boldsymbol{x}_i / \|\boldsymbol{K}^{-1}\boldsymbol{x}_i\|$

  the unknowns are the distances d_i from the camera origin $\boldsymbol{c}$ to the 3D points $\boldsymbol{p}_i$, where

$$\boldsymbol{p}_i = d_i \hat{\boldsymbol{x}}_i + \boldsymbol{c}$$



The cosine law for triangle $\Delta(\boldsymbol{c}, \boldsymbol{p}_i, \boldsymbol{p}_j)$ gives us

$$f_{ij}(d_i, d_j) = d_i^2 + d_j^2 - 2 d_i d_j c_{ij} - d_{ij}^2 = 0,$$

where

$$c_{ij} = \cos\theta_{ij} = \hat{\boldsymbol{x}}_i \cdot \hat{\boldsymbol{x}}_j$$

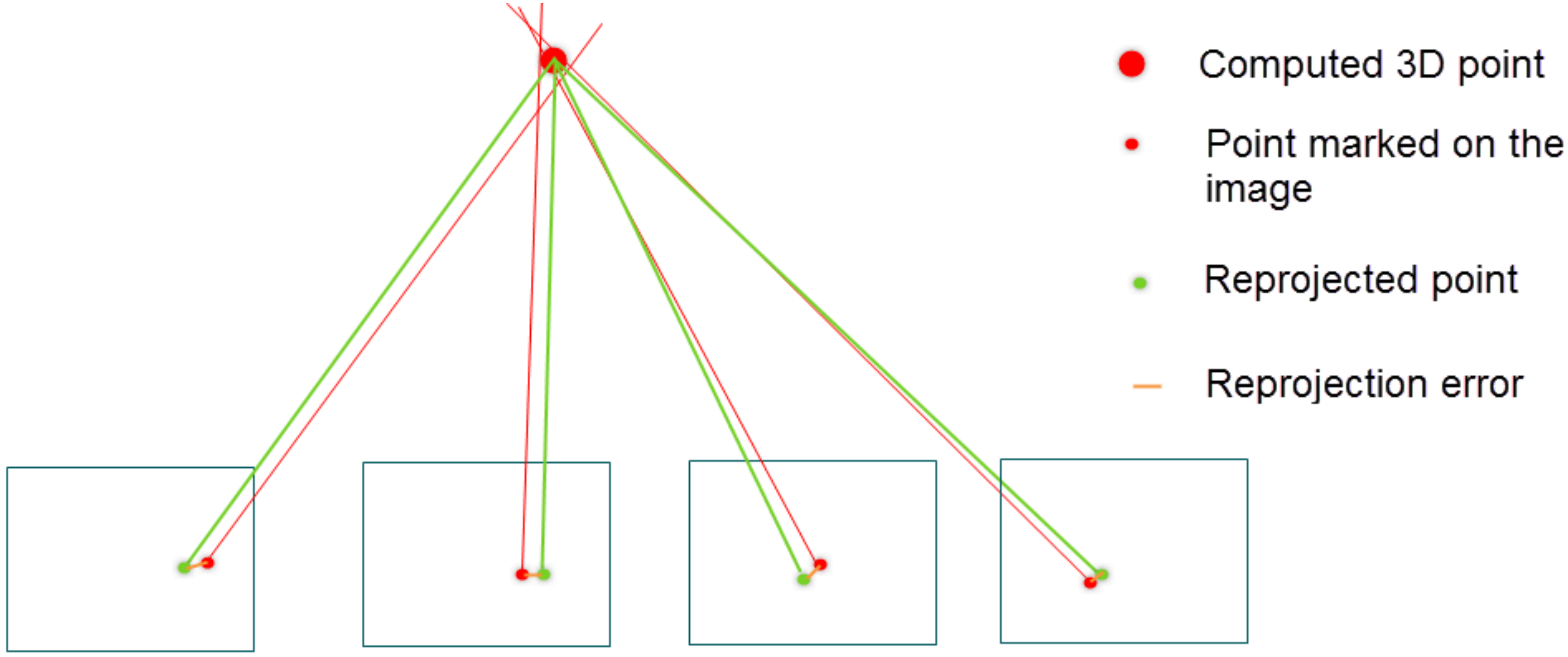
and

$$d_{ij}^2 = \|\boldsymbol{p}_i - \boldsymbol{p}_j\|^2.$$

Thus any triplet of constraints $f_{ij}(d_i, d_j), f_{ik}(d_i, d_k), f_{jk}(d_j, d_k)$ generates 3 equations in 3 unknowns.

Iterative Algorithms

- These minimal linear one-shot algorithms have limitations:
  - Noisy (few points)
  - Do not directly minimize error
- Given these limitations, they are most useful as a means to generate an initial guess that can then be refined iteratively to minimize the **reprojection error**.
- **Definition**: **Reprojection error**
  - The deviation in the image between 2D image points $\boldsymbol{x}_i$ and their corresponding 3D points $\boldsymbol{p}_i$, projected to the image.



Iterative Algorithms

- Let $\boldsymbol{f}$ now represent projection to the image:

$$\boldsymbol{x}_i = \boldsymbol{f}(\boldsymbol{p}_i; \boldsymbol{R}, \boldsymbol{t}, \boldsymbol{K})$$

- We now iteratively minimize a measure of the linearized reprojection error

$$E_{\mathrm{NLP}} = \sum_i \rho\left(\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{R}}\Delta\boldsymbol{R} + \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{t}}\Delta\boldsymbol{t} + \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{K}}\Delta\boldsymbol{K} - \boldsymbol{r}_i\right),$$

where $\boldsymbol{r}_i = \hat{\boldsymbol{x}}_i - \tilde{\boldsymbol{x}}_i$ is the current residual vector (2D error in predicted position)

and

Sign reversed in textbook.

$\hat{\boldsymbol{x}}_i$ is the 2D image point.

$\tilde{\boldsymbol{x}}_i$ is the current estimate of the projection of 3D point $\boldsymbol{p}_i$ to the image.