CORK INSTITUTE OF TECHNOLOGY
INSTITIÚID TEICNEOLAÍOCHTA CHORCAÍ

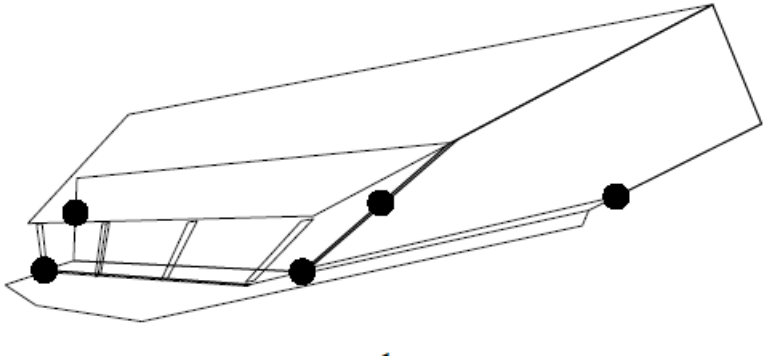
Computing@CIT

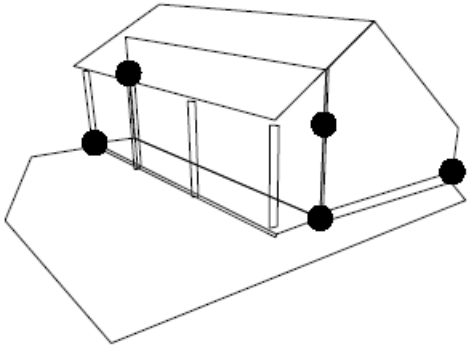
Machine Vision

Lecture 10: Metric scene reconstruction

Metric scene reconstruction

- If we have only image information available, we can only reconstruct a scene up to a 3d homography
- If we know the camera calibration we can upgrade this to a metric reconstruction defined up to a similarity
- This calibrated approach has the advantage, that not only angles in 3d are meaningful, but also the topology is preserved, as parallel lines do not intersect





Camera calibration

- To achieve a metric reconstruction we need to determine the camera calibration matrix $\boldsymbol{K}$, which enables the back-projection into rays and therefore the recovery of angles in the scene
- A projective camera projects 3d points according to the following transformation

$$\boldsymbol{x'} = \boldsymbol{PX} = \boldsymbol{KR^T}[\boldsymbol{I} \quad -\boldsymbol{t}]\boldsymbol{X}$$

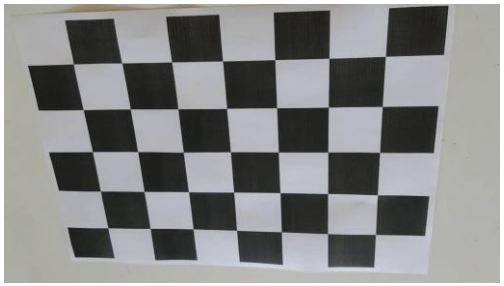
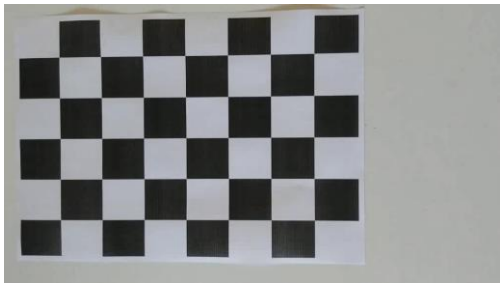
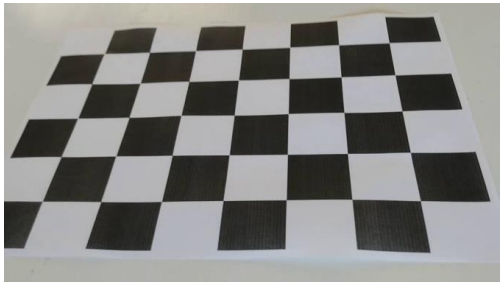
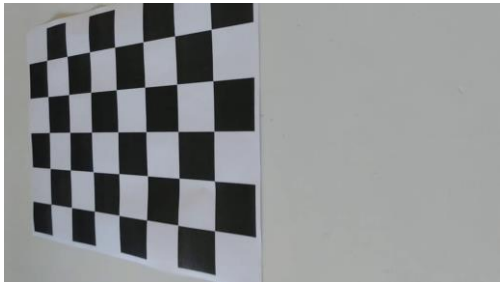
- While the position of the camera $(\boldsymbol{R}, \boldsymbol{t})$ is different for each picture, the 5 parameters of the calibration matrix

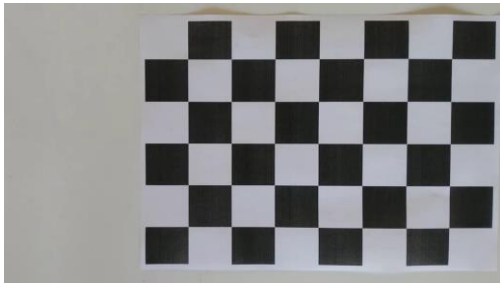
$$\boldsymbol{K} = \begin{pmatrix} c & s & x_0 \\ & \alpha c & y_0 \\ & & 1 \end{pmatrix}$$

- only depend on the camera itself can be pre-calibrated (assuming fixed focus optics)
- Because the calibration matrix does not change between different images of the same camera we can calibrate this using multiple images of a known calibration object

Camera calibration

- To determine the camera calibration we typically use multiple pictures of a planar calibration object, where we know the coordinates of points with high precision in a local coordinate system



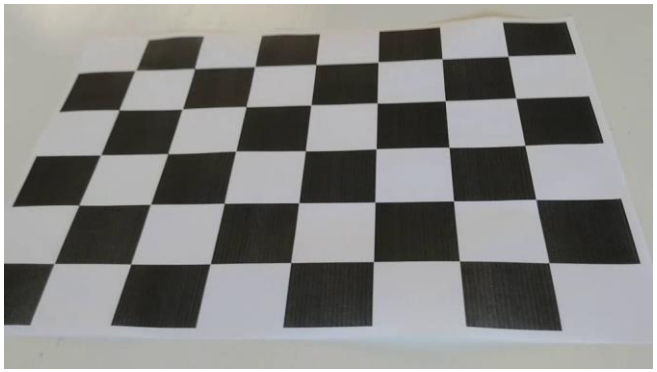
Subpixel accuracy

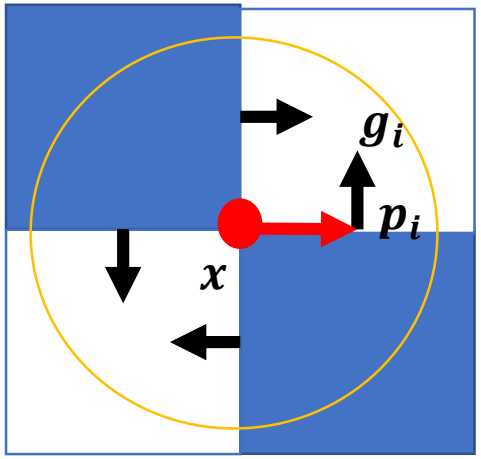
- For the calibration accuracy is extremely important
- For all points at the intersections of the checkerboard grid we observe, that the line connecting this point with every other point in the neighbourhood is orthogonal to the gradient (the gradient is zero inside the checkerboard squares), i.e.

$$\boldsymbol{g}_i^T(\boldsymbol{x} - \boldsymbol{p}_i) = 0$$

- Assuming this model the coordinate $\boldsymbol{x}$ can be much more accurately estimated using a least-squares approach as

$$\hat{\boldsymbol{x}} = \left(\sum_i \boldsymbol{g}_i \boldsymbol{g}_i^T\right)^{-1} \sum_i \boldsymbol{g}_i \boldsymbol{g}_i^T \boldsymbol{p}_i$$





Camera calibration

- Because the calibration object is planar, we cannot use the DLT approach to estimate the projection matrices directly
- We can assume that all points on the calibration pattern are on the plane $Z = 0$, which transform according to

$$\boldsymbol{x'} = \boldsymbol{PX} = \boldsymbol{KR^T}[\boldsymbol{I} \quad -\boldsymbol{t}]\begin{pmatrix} X \\ Y \\ 0 \\ 1 \end{pmatrix} = \boldsymbol{K}[\boldsymbol{r_1} \quad \boldsymbol{r_2} \quad -\boldsymbol{R^T t}]\begin{pmatrix} X \\ Y \\ 1 \end{pmatrix}$$

- This is a 2d homography

$$\boldsymbol{H} = \boldsymbol{K}[\boldsymbol{r_1} \quad \boldsymbol{r_2} \quad -\boldsymbol{R^T t}]$$

- We already know how to estimate a 2d homography from 4 point correspondences

Camera calibration

- A rotation matrix is always ortho-normal, i.e. $\boldsymbol{R^T R = I}$
- In particular $\boldsymbol{r_1^T r_1} = 1$ and $\boldsymbol{r_1^T r_2} = 0$

- The first two columns of the homography

$$\boldsymbol{H = [h_1 \quad h_2 \quad h_3] = K[r_1 \quad r_2 \quad -R^T t]}$$

- therefore have to satisfy the constraints

$$\boldsymbol{h_1^T K^{-T} K^{-1} h_2} = 0$$

$$\boldsymbol{h_1^T K^{-T} K^{-1} h_1 = h_2^T K^{-T} K^{-1} h_2}$$

Camera calibration

- With the substitution $\boldsymbol{b} = vec\left[\boldsymbol{K}^{-T}\boldsymbol{K}^{-1}\right]$ these constraints can be written as

$$\left(\boldsymbol{h}_2^T \otimes \boldsymbol{h}_1^T\right)\boldsymbol{b} = 0$$

$$\left(\boldsymbol{h}_1^T \otimes \boldsymbol{h}_1^T - \boldsymbol{h}_2^T \otimes \boldsymbol{h}_2^T\right)\boldsymbol{b} = 0$$

- Using homographies from more than $n \geq 3$ different images we can stack them into a matrix

$$\mathbf{Ab} = \begin{pmatrix} \boldsymbol{h}_{12}^T \otimes \boldsymbol{h}_{11}^T \\ \boldsymbol{h}_{11}^T \otimes \boldsymbol{h}_{11}^T - \boldsymbol{h}_{12}^T \otimes \boldsymbol{h}_{12}^T \\ \vdots \\ \boldsymbol{h}_{n2}^T \otimes \boldsymbol{h}_{n1}^T \\ \boldsymbol{h}_{n1}^T \otimes \boldsymbol{h}_{n1}^T - \boldsymbol{h}_{n2}^T \otimes \boldsymbol{h}_{n2}^T \end{pmatrix} \boldsymbol{b} = \mathbf{0}$$

- Because the matrix $\boldsymbol{B} = \boldsymbol{K}^{-T}\boldsymbol{K}^{-1}$ is symmetric, we can cut out the 6 columns from **A** corresponding to the upper right triangle

Camera calibration

- We can now solve the homogeneous equation system $\boldsymbol{Ab} = \mathbf{0}$ for the unknown $\boldsymbol{b} = vec\left[\boldsymbol{K^{-T}K^{-1}}\right]$ using the singular value decomposition of **A**

- To obtain the calibration matrix

$$\boldsymbol{K} = \begin{pmatrix} c & s & x_0 \\ & \alpha c & y_0 \\ & & 1 \end{pmatrix}$$

- We solve for the individual components of

$$\boldsymbol{K^T K^{-1}} = \begin{pmatrix} \frac{1}{c^2} & -\frac{s}{\alpha c^3} & \frac{sy_0 - \alpha c x_0}{\alpha c^3} \\ \cdot & \frac{s^2}{\alpha^2 c^4} + \frac{1}{\alpha^2 c^2} & \frac{-s(sy_0 - \alpha c x_0)}{\alpha^2 c^4} - \frac{y_0}{\alpha^2 c^2} \\ \cdot & \cdot & \frac{(sy_0 - \alpha c x_0)^2}{\alpha^2 c^4} + \frac{y_0^2}{\alpha^2 c^2} + 1 \end{pmatrix}$$

- If we assume $s = 0$ this is very straightforward

Camera calibration

- Knowing the calibration matrix $\boldsymbol{K}$ we can also determine each rotation matrix

$$\boldsymbol{r_1} = \lambda \boldsymbol{K^{-1} h_1}$$
$$\boldsymbol{r_2} = \lambda \boldsymbol{K^{-1} h_2}$$
$$\boldsymbol{r_3} = \boldsymbol{r_1} \times \boldsymbol{r_2}$$

Using the normalisation factor

$$\lambda = \left(\boldsymbol{h_1^T K^{-T} K^{-1} h_1}\right)^{-\frac{1}{2}} = \left(\boldsymbol{h_2^T K^{-T} K^{-1} h_2}\right)^{-\frac{1}{2}}$$

- Finally also the coordinate of the camera's projection centres can be determined from the homographies as

$$\boldsymbol{t} = -\lambda \boldsymbol{R K^{-1} h_3}$$

Camera calibration

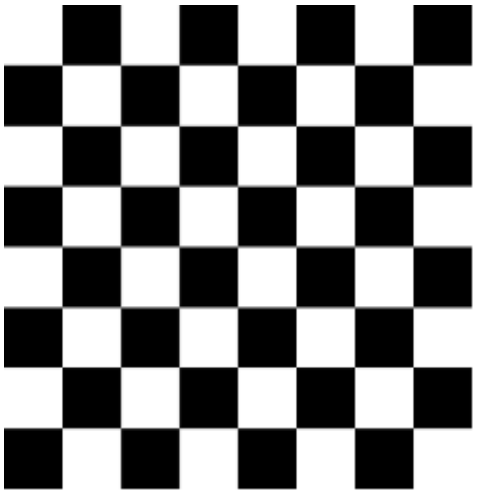
- So far we have discussed a direct solution for the camera parameters
- To obtain more accurate calibration results we need to estimate the parameters using a maximum likelihood approach with the model function

$$\boldsymbol{x'_{ij}} = \boldsymbol{K}\boldsymbol{R}_{\boldsymbol{j}}^{\boldsymbol{T}}[\boldsymbol{I} \quad -\boldsymbol{t_j}]\boldsymbol{X_i} \qquad i \in \{1, \dots, N\}, j \in \{1, \dots, M\}$$

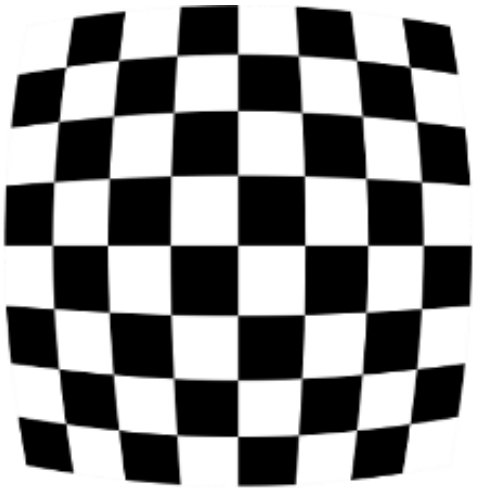
- The observations are all 2d point $\boldsymbol{x'_{ij}}$ where the i-th 3d point $\boldsymbol{X_i}$ is projected into image j measured to sub-pixel accuracy
- The parameters are the 5 common calibration parameters: c, α, s, x_0, y_0 and for each image j the 6 parameters of the camera positions $\boldsymbol{t_j}$ and the rotation in Rodrigues representation $\boldsymbol{m_j}$
- In total we get $2MN$ observations and $5 + 6M$ parameters, which enable the estimation of all parameters to very high accuracy using the Gauss-Helmert model estimation techniques

Radial distortion

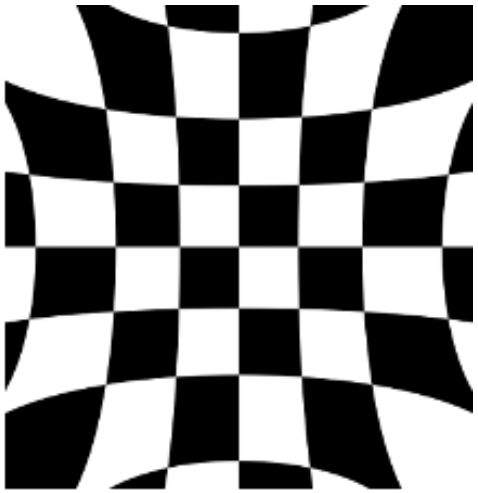
- Wide-angle camera lenses can introduce non-linear distortion
- These distortion parameters can be introduced in the estimation procedure
- This allows to compensate radial distortion effects by applying this transformation prior to any further processing



No distortion



Positive radial distortion
(Barrel distortion)



Negative radial distortion
(Pincushion distortion)

Calibration in OpenCV

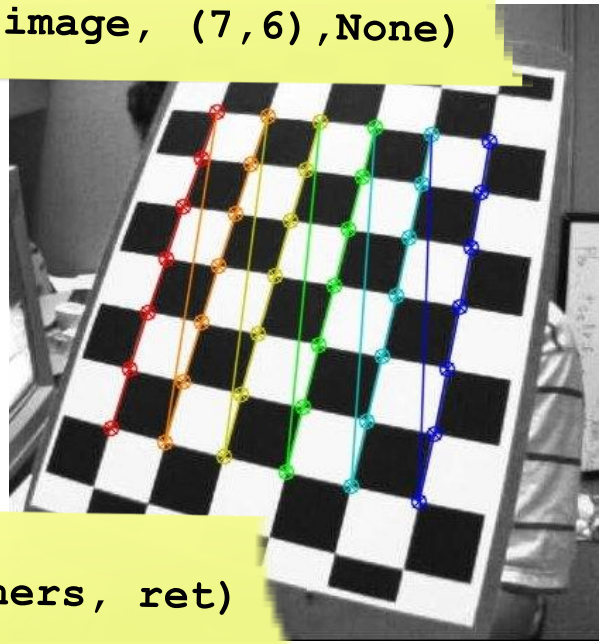
The first step is to find the 2d image coordinates of the checkerboard corners in each image

We have to specify the number of internal corners

```
ret, corners = cv2.findChessboardCorners(image, (7,6),None)
```

The result is a list of all internal corner points

The corners can be drawn into the images for verification



```
cv2.drawChessboardCorners(img, (7,6), corners, ret)
```

Calibration in OpenCV

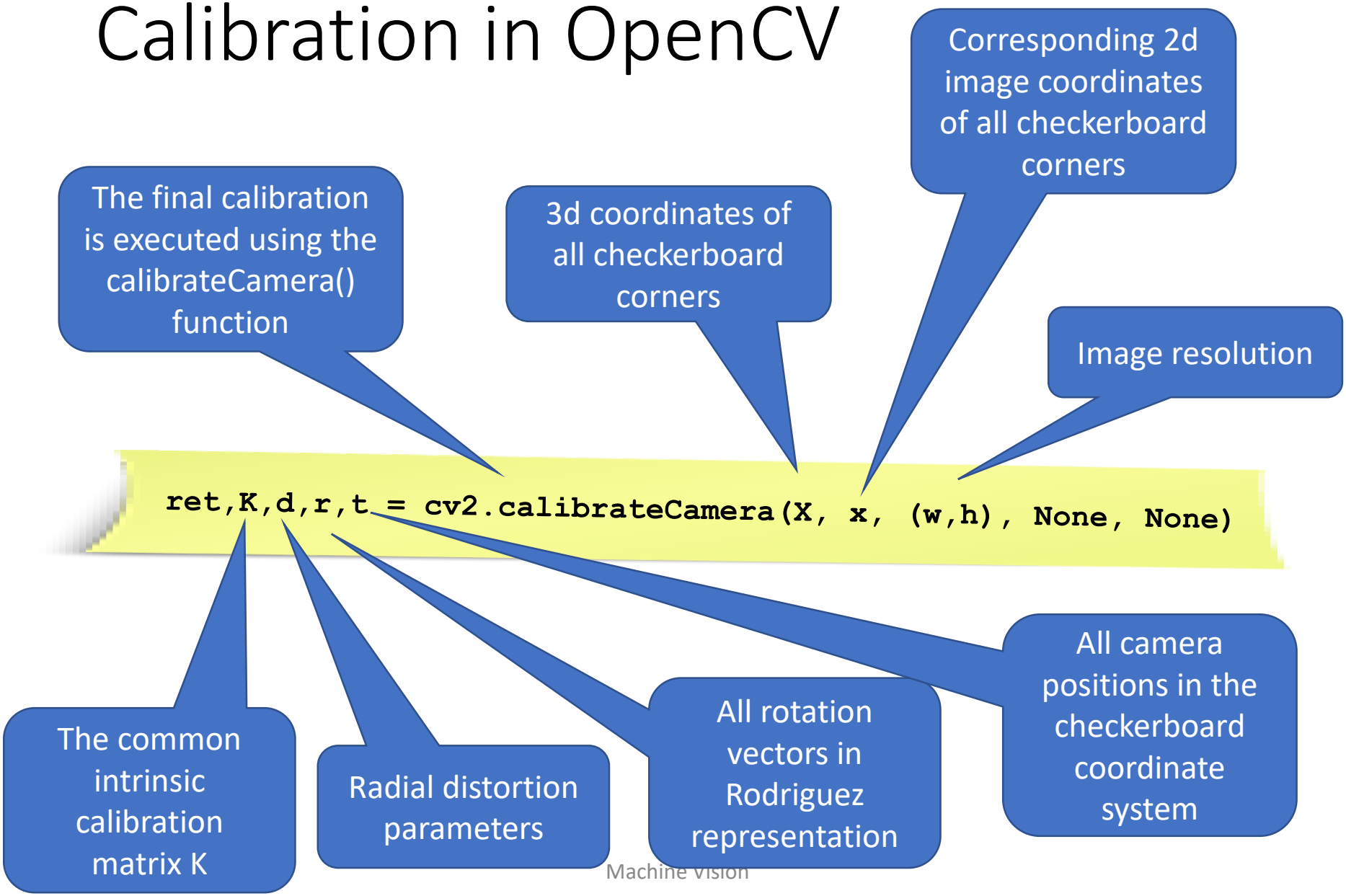
```
criteria = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER,
            30,
            0.001)
corners2 = cv2.cornerSubPix(image, corners,(11,11),(-1,-1),criteria)
```

Corners can be refined to sub-pixel accuracy

We have to provide a window size for the estimation

The termination criteria for the iterative least-squares estimation procedure need to be provided

Calibration in OpenCV



Essential matrix

- Two corresponding image points $\boldsymbol{x'} \leftrightarrow \boldsymbol{x}$ have to satisfy the epipolar constraint

$$\boldsymbol{x'^T F x} = 0$$

- Knowing the calibration matrices $\boldsymbol{K}$ and $\boldsymbol{K'}$ we can work with directions instead of image coordinates

$$\boldsymbol{m} = \boldsymbol{K}^{-1}\boldsymbol{x}$$
$$\boldsymbol{m'} = \boldsymbol{K'^{-1}}\boldsymbol{x'}$$

- And formulate the epipolar constraint very similar as

$$\boldsymbol{m'^T K'^T F K m} = \boldsymbol{m'^T E m} = 0$$

- The matrix $\boldsymbol{E} = \boldsymbol{K'^T F K}$ is called the **essential matrix** and is the calibrated equivalent to the fundamental matrix describing the epipolar geometry of two calibrated cameras

Essential matrix

- We derived the fundamental matrix to be

$$\boldsymbol{F} = \boldsymbol{S}[\boldsymbol{K}'\boldsymbol{R}^T\boldsymbol{t}]\boldsymbol{K}'\boldsymbol{R}^T\boldsymbol{K}^{-1} = \boldsymbol{K}'^{-\boldsymbol{T}}\boldsymbol{S}[\boldsymbol{R}^T\boldsymbol{t}]\boldsymbol{R}^T\boldsymbol{K}^{-1}$$

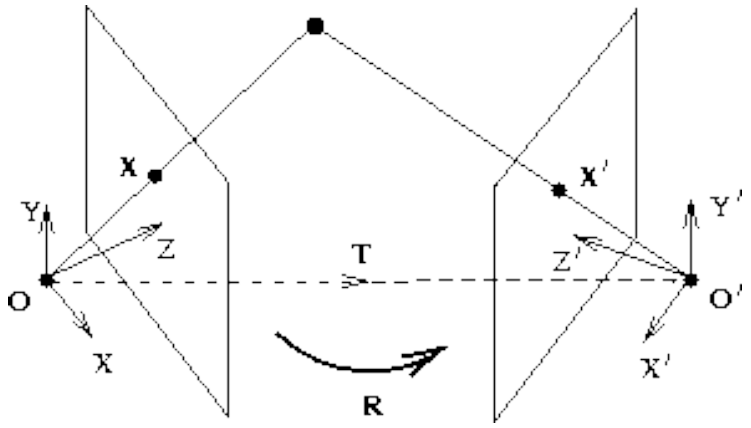
- Therefore the essential matrix is

$$\boldsymbol{E} = \boldsymbol{K}'^T\boldsymbol{F}\boldsymbol{K} = \boldsymbol{S}[\boldsymbol{R}^T\boldsymbol{t}]\boldsymbol{R}^T$$

- It only depends on the baseline and the rotation and is defined up to scale, therefore it has 5 degrees of freedom

Essential matrix

- The essential matrix describes the mutual position of two cameras up to scale, i.e. the rotation between the two camera centred coordinate systems and the baseline
- The length of the baseline, i.e. the distance between the two cameras, is unknown



Essential matrix

- Like the fundamental matrix the essential matrix is singular, i.e. its determinant is zero

$$\det \boldsymbol{E} = 0$$

- In addition to this the fundamental matrix satisfies the constraint

$$\boldsymbol{E}\boldsymbol{E}^T\boldsymbol{E} - \frac{1}{2} tr[\boldsymbol{E}\boldsymbol{E}^T]\boldsymbol{E} = \boldsymbol{0}$$

- This can be used to estimate it from fewer than 7 point correspondences
- Also, unlike the 7- and 8-point algorithm for the fundamental matrix the 5-point algorithm enables the estimation from co-planar points

Essential matrix

- Another way to enforce the constraints of the essential matrix is by noting that it's two non-zero singular value are equal, i.e. every valid essential matrix decomposes into

$$\boldsymbol{E} = \lambda \boldsymbol{U} \begin{pmatrix} 1 & & \\ & 1 & \\ & & 0 \end{pmatrix} \boldsymbol{V}^T$$

- To estimate the essential matrix we can therefore first estimate the fundamental matrix $\boldsymbol{F}$ from 7 or 8 points and apply the known calibration matrix to obtain

$$\widehat{\boldsymbol{E}} = \boldsymbol{K'}^{\boldsymbol{T}} \boldsymbol{F} \boldsymbol{K}$$

- We then calculate the singular value decomposition of

$$\widehat{\boldsymbol{E}} = \boldsymbol{U} \begin{pmatrix} \lambda_1 & & \\ & \lambda_2 & \\ & & 0 \end{pmatrix} \boldsymbol{V}^T$$

- Because of noise, the singular values are potentially unequal $\lambda_1 \neq \lambda_2$. To enforce the equality constraint we put the essential matrix together again using the average singular value instead

$$\lambda = \frac{\lambda_1 + \lambda_2}{2}$$

Essential matrix

- To recover the rotation and translation from $\boldsymbol{E} = \boldsymbol{S}[\boldsymbol{R}^T \boldsymbol{t}]\boldsymbol{R}^T$ we use the singular value decomposition

$$\boldsymbol{E} = \boldsymbol{U}\begin{pmatrix} \lambda & & \\ & \lambda & \\ & & 0 \end{pmatrix}\boldsymbol{V}^T$$

- And the fact that with the two matrices

$$W = \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \qquad Z = \begin{pmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

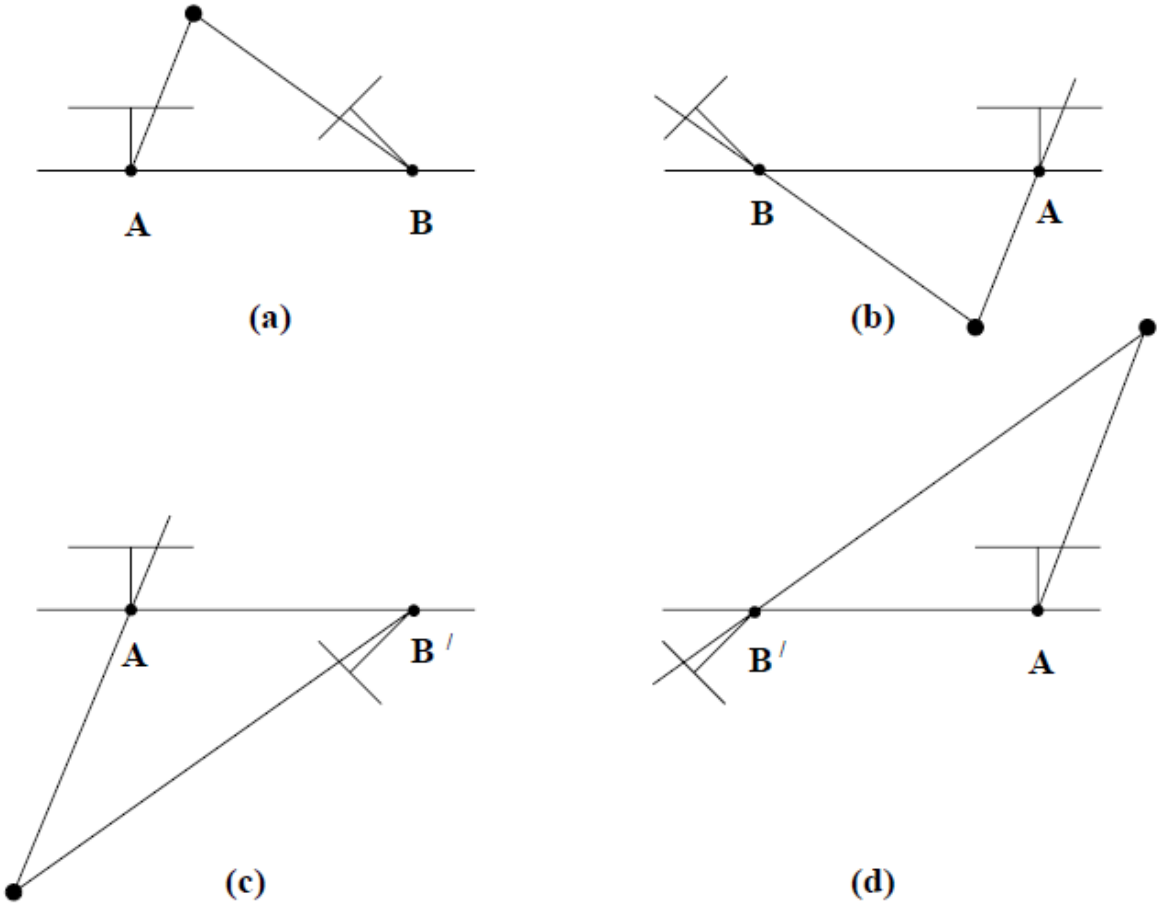
- The two constituent parts are any of the four possible solutions

$$\boldsymbol{S}[\boldsymbol{R}^T \boldsymbol{t}] = \pm\beta \boldsymbol{U}\boldsymbol{Z}\boldsymbol{U}^{\boldsymbol{T}}$$
$$\boldsymbol{R}^T = \boldsymbol{U}\boldsymbol{W}\boldsymbol{V}^T \quad or \quad \boldsymbol{R}^T = \boldsymbol{U}\boldsymbol{W}^T\boldsymbol{V}^T$$

- The scale of the baseline β is the distance between the two camera and can be used to introduce the correct scale (i.e. unit of measurement)

Essential matrix

- Out of the four possible solutions, only one of these has all points in front of both cameras, which can be used to select the correct one



Determinant of rotation matrices

- The determinant of a proper rotation matrix is always $\det \boldsymbol{R} = 1$

- Unfortunately, this is not the case for the Numpy implementation of the singular value decomposition
- Therefore, whenever you need to derive a proper rotation matrix from a singular value decomposition using Numpy, you need to test this condition and flip the rotation accordingly (this works because $\boldsymbol{E}$ is singular and the last singular value is 0)

```
U,S,V = np.linalg.svd(E)
if np.linalg.det(U)<0:
    U[:,2] *= -1
if np.linalg.det(V)<0:
    V[2,:] *= -1
```

Back-projection of points

- Knowing $\boldsymbol{t}$ and $\boldsymbol{R}$ we can determine the coordinates of all 3d scene points by intersecting the rays

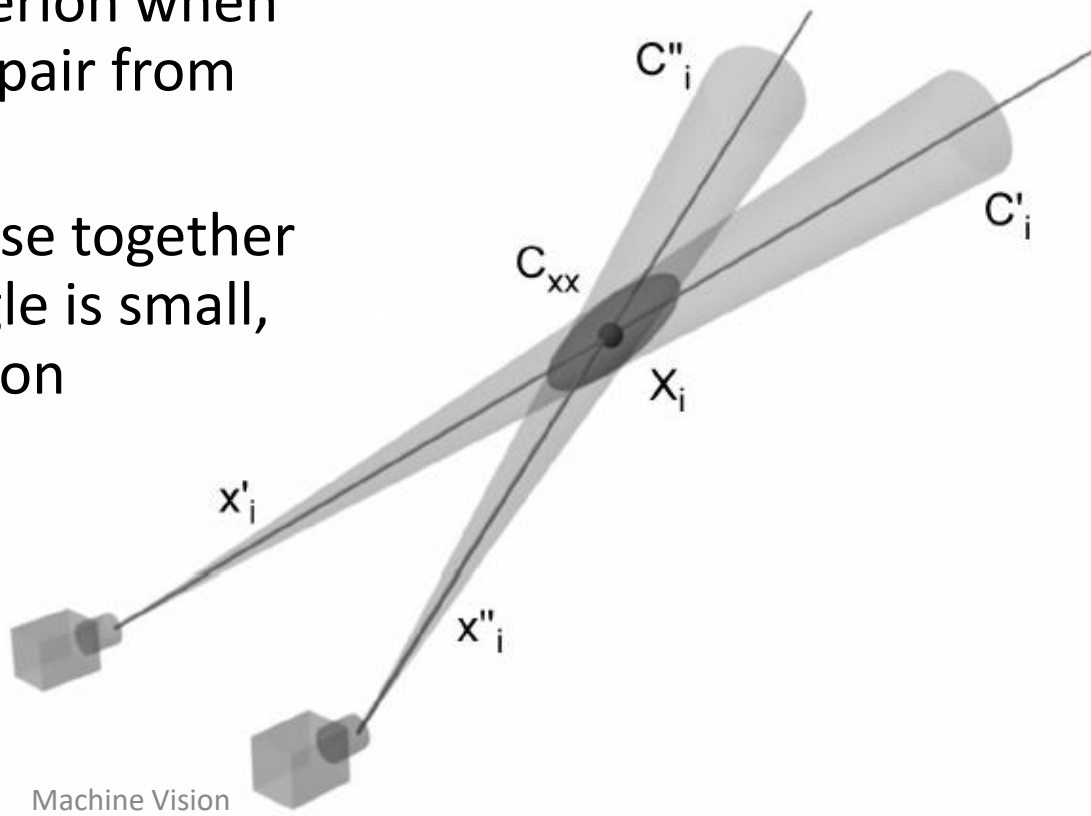
$$\boldsymbol{X}[\lambda] = \lambda \boldsymbol{m}$$
$$\boldsymbol{X}[\mu] = \boldsymbol{t} + \mu \boldsymbol{R}\boldsymbol{m}'$$

- The unknown distances λ and μ are calculated by solving the linear equation system

$$\begin{pmatrix} \boldsymbol{m}^T\boldsymbol{m} & -\boldsymbol{m}^T\boldsymbol{R}\boldsymbol{m}' \\ \boldsymbol{m}^T\boldsymbol{R}\boldsymbol{m}' & -\boldsymbol{m}'^T\boldsymbol{m}' \end{pmatrix} \begin{pmatrix} \lambda \\ \mu \end{pmatrix} = \begin{pmatrix} \boldsymbol{t}^T\boldsymbol{m} \\ \boldsymbol{t}^T\boldsymbol{R}\boldsymbol{m}' \end{pmatrix}$$

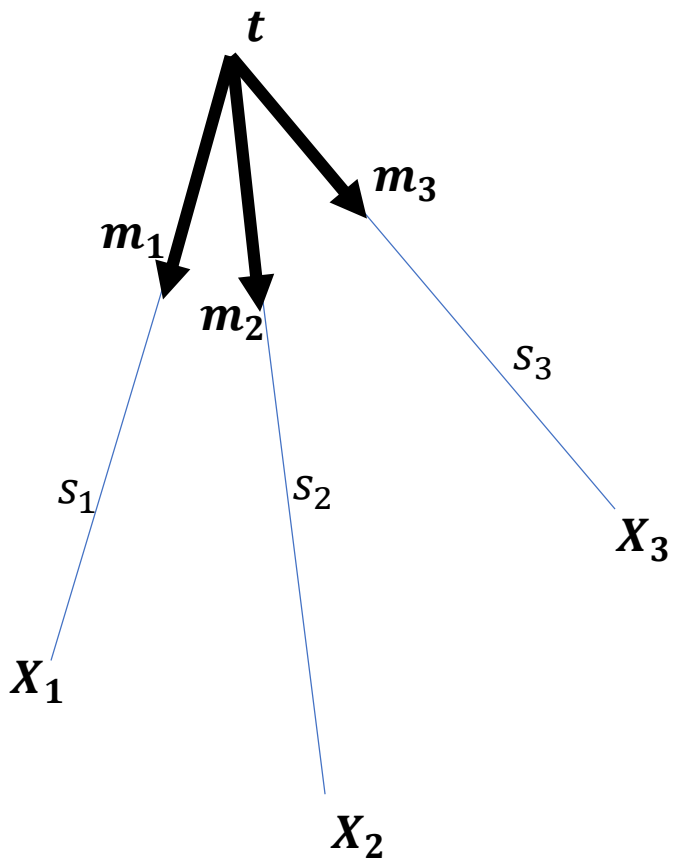
Back-projection of points

- The accuracy of the 3d points depends on how orthogonal the intersecting rays are
- This can be used as a criterion when selecting an initial image pair from an image sequence
- If the cameras are too close together and/or if the opening angle is small, the geometric configuration is more challenging



Spatial resection

- Additional frames can be connected by determining their position $\boldsymbol{t}$ and rotation $\boldsymbol{R}$ with respect to the reconstructed 3d scene points $\boldsymbol{X_i}$ from directions $\boldsymbol{m_i = K^{-1} x_i}$
- The minimal direct solution requires 3 point correspondences
- We start with calculating the unknown distances s_1, s_2, s_3 from $\boldsymbol{t}$ to $\boldsymbol{X_1}$, $\boldsymbol{X_2}$, $\boldsymbol{X_3}$



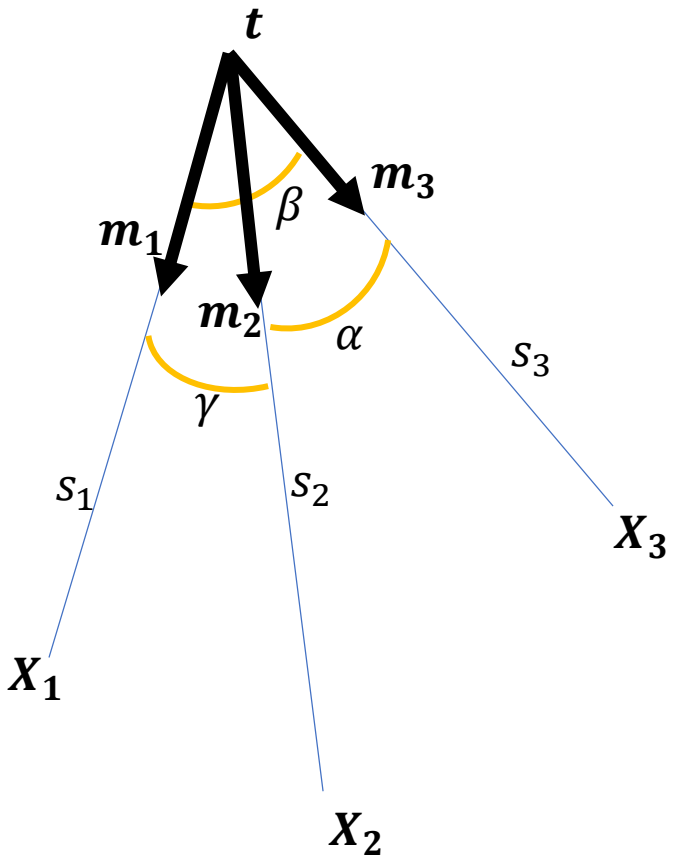
Spatial resection

- From the three directions $\boldsymbol{m_1}$, $\boldsymbol{m_2}$, $\boldsymbol{m_3}$ we can calculate the angels

$$\cos\alpha = \frac{\boldsymbol{m_2^T m_3}}{\sqrt{\boldsymbol{m_2^T m_2}}\sqrt{\boldsymbol{m_3^T m_3}}}$$

$$\cos\beta = \frac{\boldsymbol{m_2^T m_3}}{\sqrt{\boldsymbol{m_2^T m_2}}\sqrt{\boldsymbol{m_3^T m_3}}}$$

$$\cos\gamma = \frac{\boldsymbol{m_2^T m_3}}{\sqrt{\boldsymbol{m_2^T m_2}}\sqrt{\boldsymbol{m_3^T m_3}}}$$



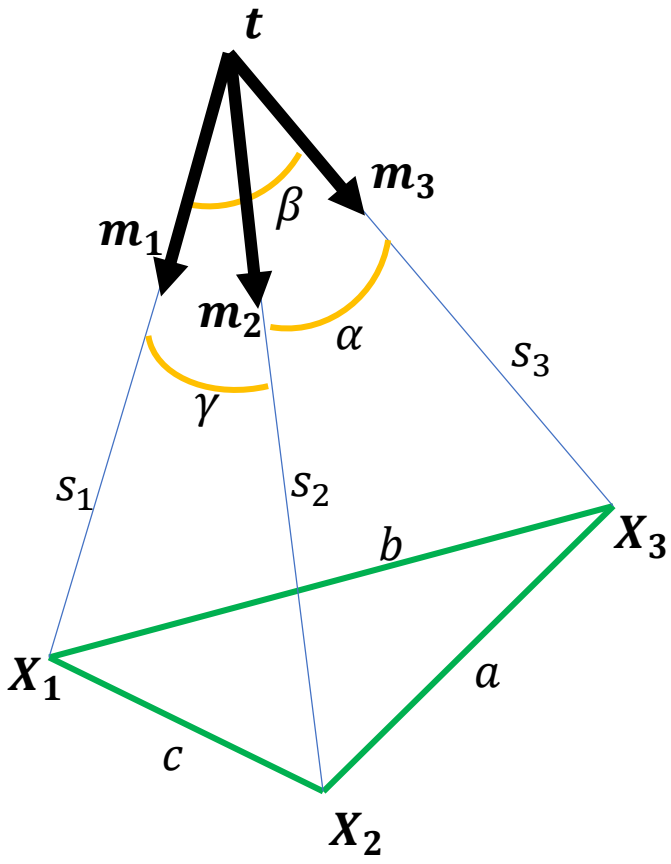
Spatial resection

- From the three points $\boldsymbol{X_1}, \boldsymbol{X_2}, \boldsymbol{X_3}$ we can calculate the distances

$$a = \sqrt{(\boldsymbol{X_3} - \boldsymbol{X_2})^T (\boldsymbol{X_3} - \boldsymbol{X_2})}$$

$$b = \sqrt{(\boldsymbol{X_1} - \boldsymbol{X_3})^T (\boldsymbol{X_1} - \boldsymbol{X_3})}$$

$$c = \sqrt{(\boldsymbol{X_2} - \boldsymbol{X_1})^T (\boldsymbol{X_2} - \boldsymbol{X_1})}$$



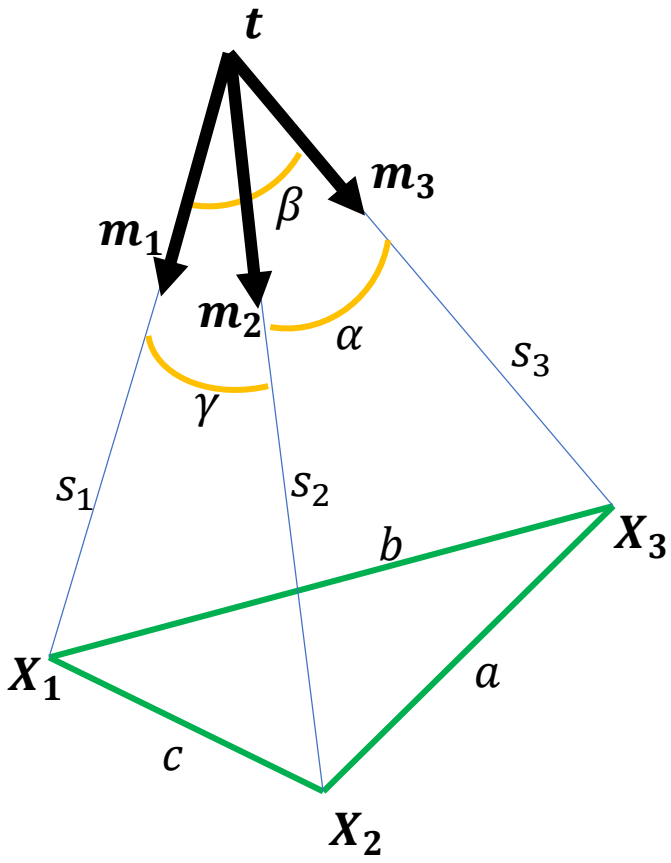
Spatial resection

- Now we observe that the distances s_1, s_2, s_3 from $\boldsymbol{t}$ to $\boldsymbol{X_1}$, $\boldsymbol{X_2}$, $\boldsymbol{X_3}$ have to satisfy

$$a^2 = s_2^2 + s_3^2 - 2s_2 s_3 \cos\alpha$$

$$b^2 = s_3^2 + s_1^2 - 2s_3 s_1 \cos\beta$$

$$c^2 = s_1^2 + s_2^2 - 2s_1 s_2 \cos\gamma$$



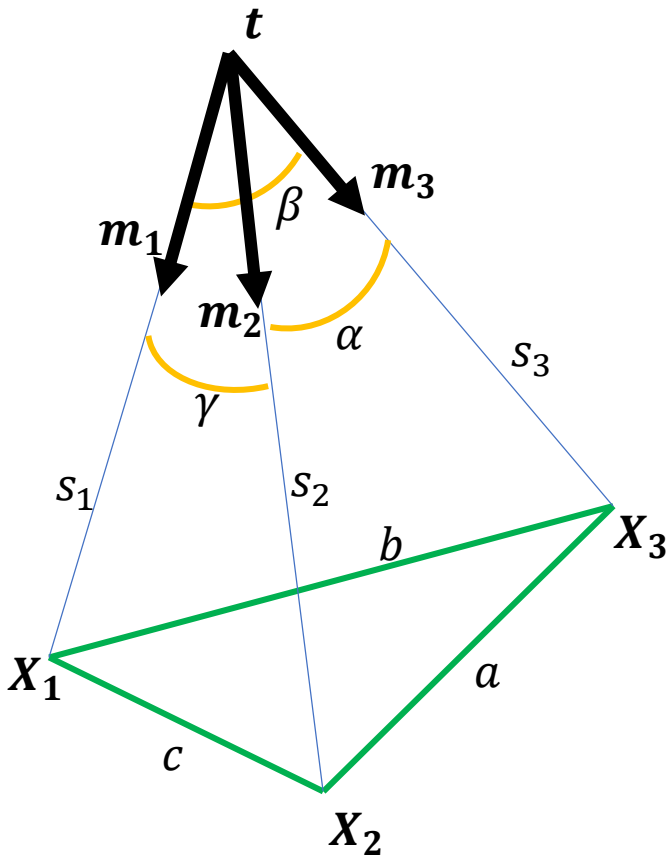
Spatial resection

- We substitute $u = \frac{s_2}{s_1}$, $v = \frac{s_3}{s_1}$ and solve for

$$s_1^2 = \frac{a^2}{u^2 + v^2 - 2uv\cos\alpha}$$

$$= \frac{b^2}{1 + v^2 - 2v\cos\beta}$$

$$= \frac{c^2}{1 + u^2 - 2u\cos\gamma}$$

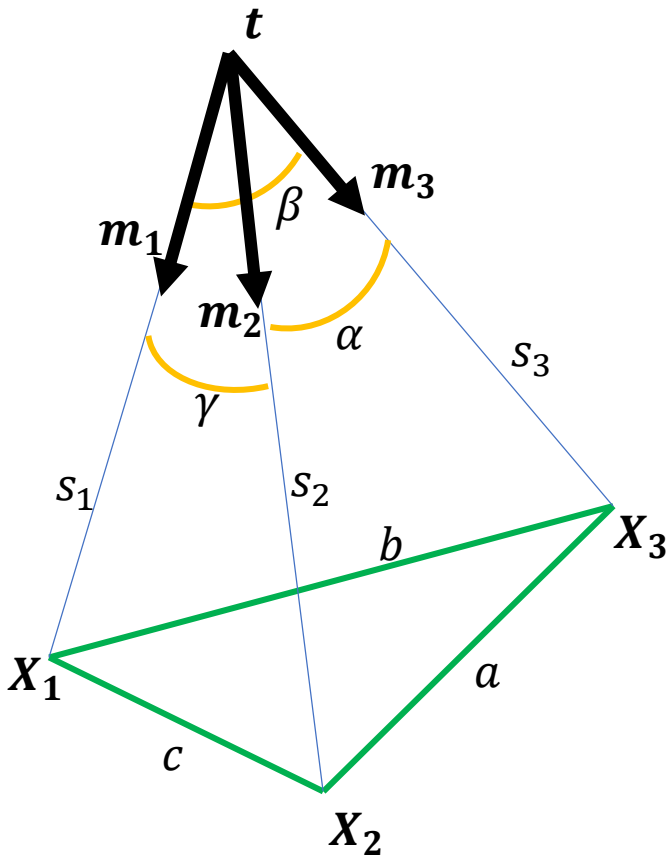


Spatial resection

- These two equations can be solved for u an then rewritten as a 4th degree polynomial in v

$$A_4 v^4 + A_3 v^3 + A_2 v^2 + A_1 v + A_0 = 0$$

- Solving this for v yields up to **4 valid solutions**, from which we can determine the three distances s_1, s_2, s_3 by substituting back into the previous equations



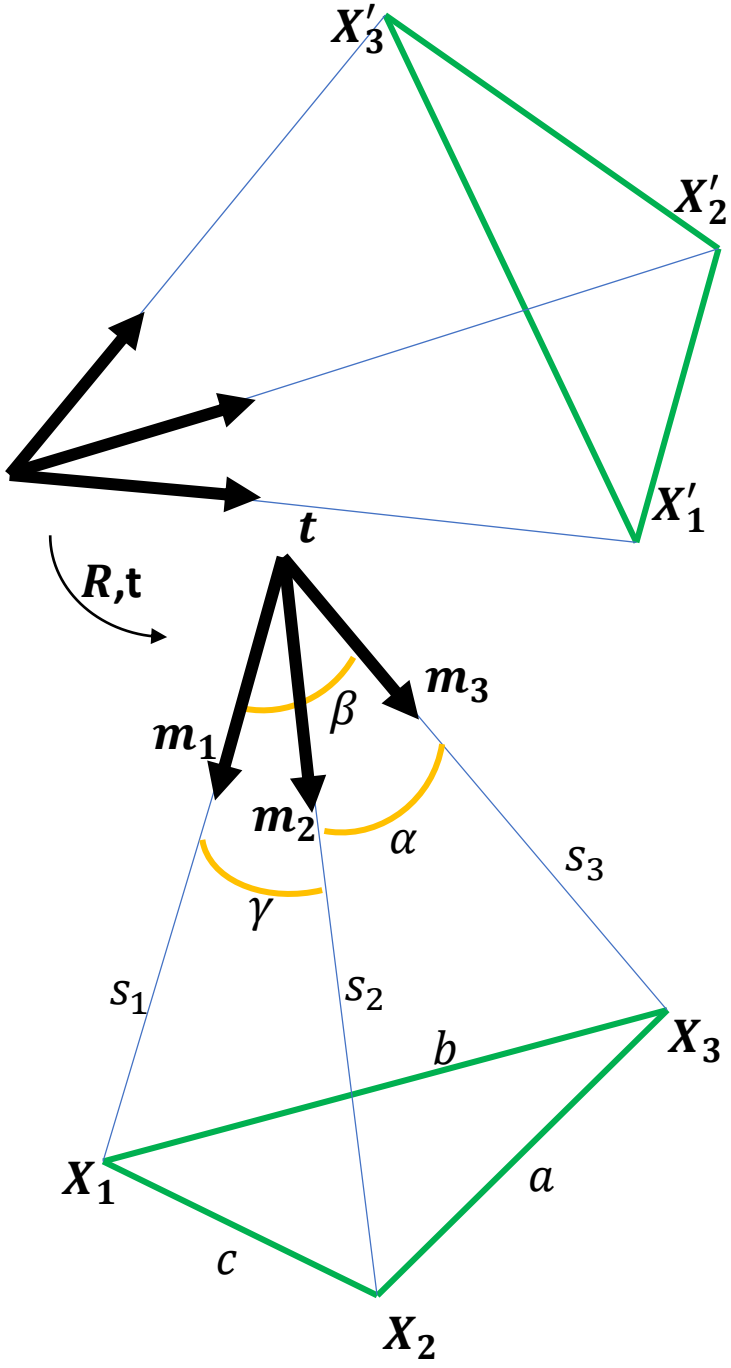
Spatial resection

- We can now calculate the coordinates of the 3d points in a camera centred coordinate system

$$\boldsymbol{X_i'} = s_i \frac{\boldsymbol{m_i}}{\sqrt{\boldsymbol{m_i^T m_i}}}$$

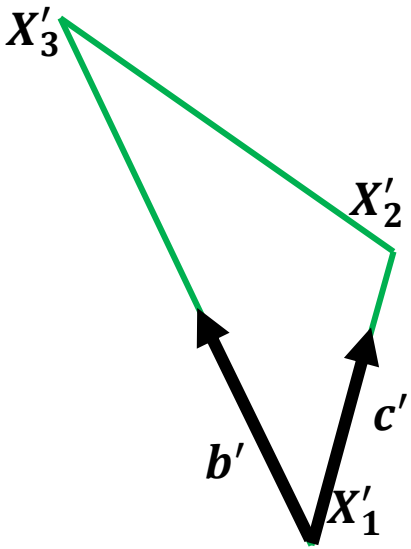
- All that is left is to determine the unknown rotation $\boldsymbol{R}$ and translation $\boldsymbol{t}$

$$\boldsymbol{X_i'} = \boldsymbol{R}^T(\boldsymbol{X_i} - \boldsymbol{t})$$



Spatial resection

- To obtain the mutual rotation we first calculate the direction vectors

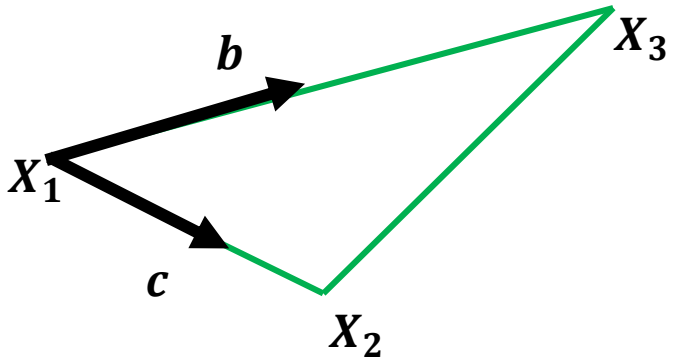


$$\boldsymbol{b} = \frac{\boldsymbol{X_3} - \boldsymbol{X_1}}{\sqrt{(\boldsymbol{X_3} - \boldsymbol{X_1})^T(\boldsymbol{X_3} - \boldsymbol{X_1})}}$$

$$\boldsymbol{c} = \frac{\boldsymbol{X_2} - \boldsymbol{X_1}}{\sqrt{(\boldsymbol{X_2} - \boldsymbol{X_1})^T(\boldsymbol{X_2} - \boldsymbol{X_1})}}$$

$$\boldsymbol{b'} = \frac{\boldsymbol{X_3'} - \boldsymbol{X_1'}}{\sqrt{(\boldsymbol{X_3'} - \boldsymbol{X_1'})^T(\boldsymbol{X_3'} - \boldsymbol{X_1'})}}$$

$$\boldsymbol{c'} = \frac{\boldsymbol{X_2'} - \boldsymbol{X_1'}}{\sqrt{(\boldsymbol{X_2'} - \boldsymbol{X_1'})^T(\boldsymbol{X_2'} - \boldsymbol{X_1'})}}$$



Spatial resection

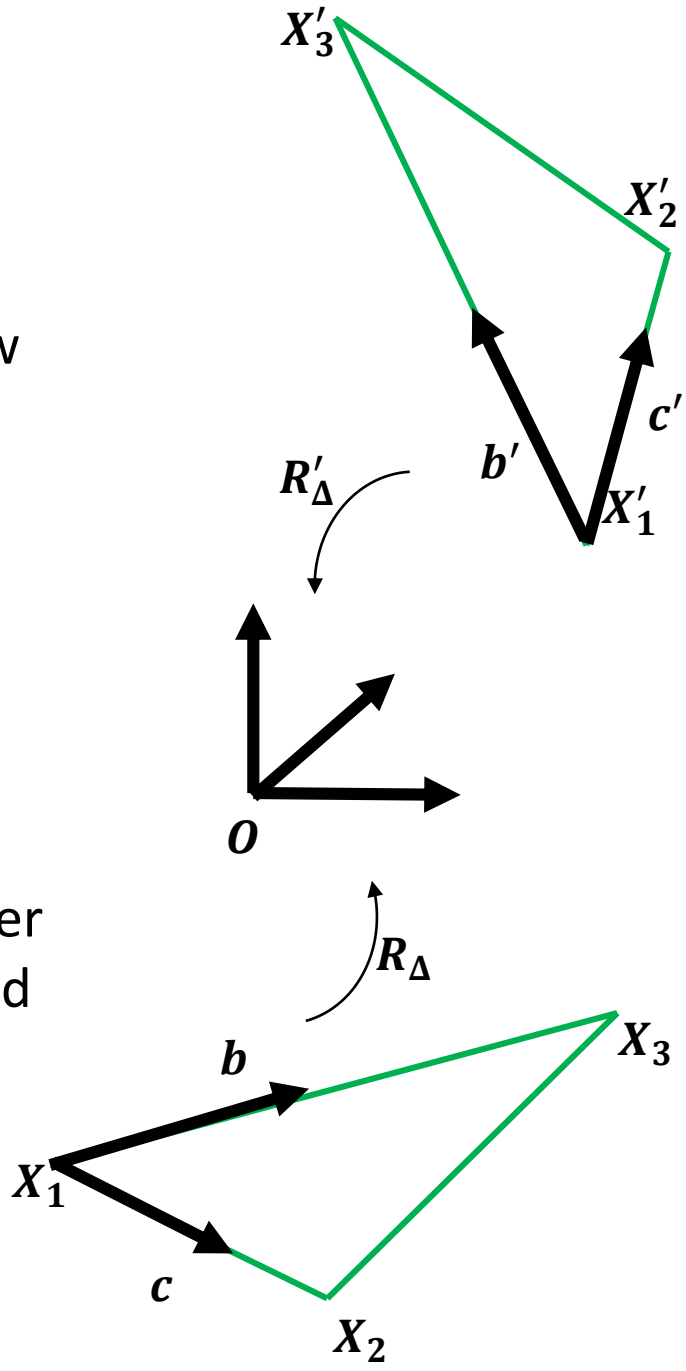
- The two rotation of the two triangles can now be calculated as

$$\boldsymbol{R_\Delta} = (\boldsymbol{b} \quad \boldsymbol{b} \times \boldsymbol{c} \quad \boldsymbol{b} \times (\boldsymbol{b} \times \boldsymbol{c}))$$

$$\boldsymbol{R'_\Delta} = (\boldsymbol{b'} \quad \boldsymbol{b'} \times \boldsymbol{c'} \quad \boldsymbol{b'} \times (\boldsymbol{b'} \times \boldsymbol{c'}))$$

- Therefore, to rotate one triangle into the other we need to concatenate the two rotations and finally get

$$\boldsymbol{R^T} = \boldsymbol{R'_\Delta R_\Delta^T}$$



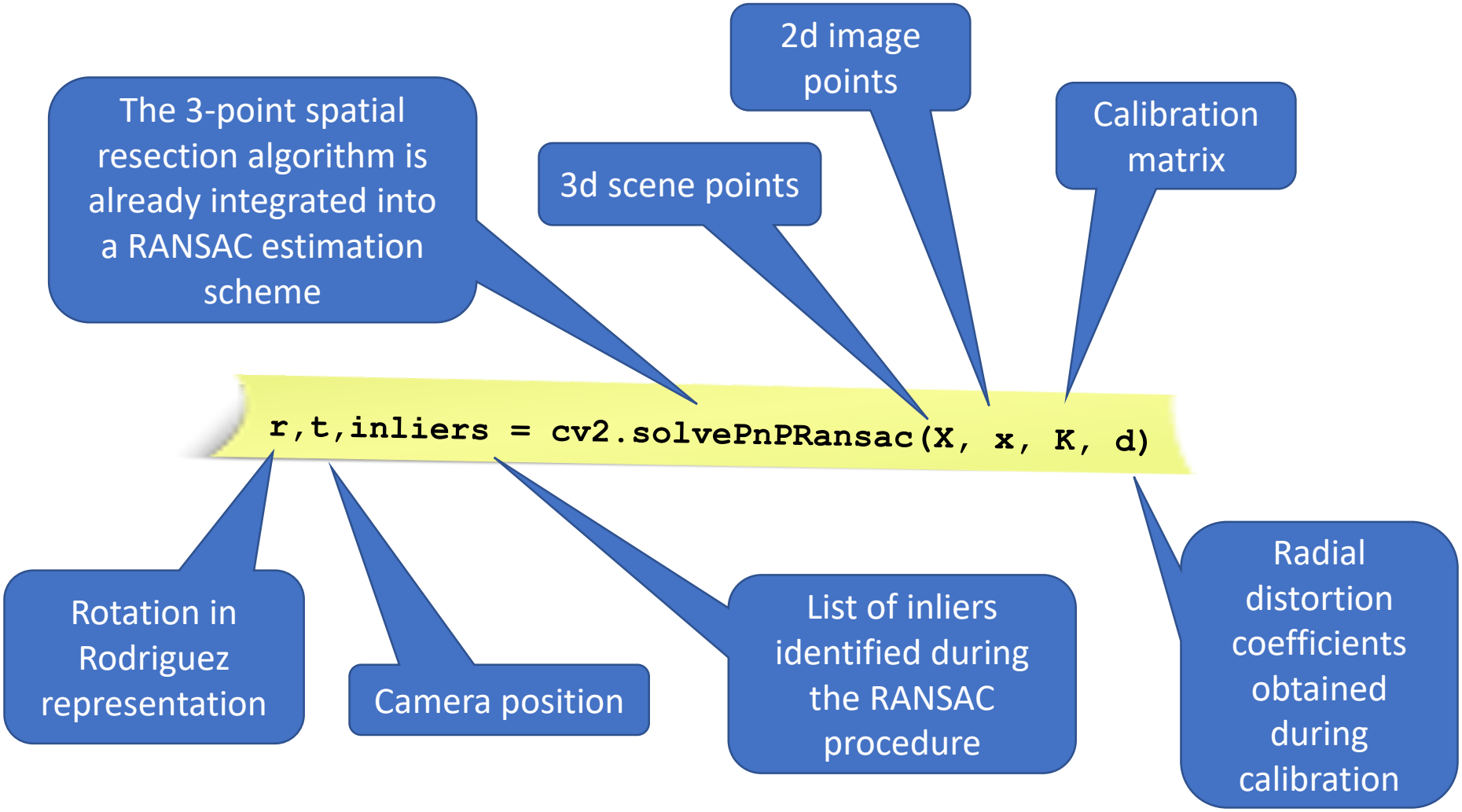
Spatial resection

- The last step is to calculate the position of the camera
- Now knowing the mutual rotation, this can be easily done from any of the three point correspondences

$$\boldsymbol{t} = \boldsymbol{X_i} - \boldsymbol{R}\boldsymbol{X_i'}$$

- Due to the 4th degree polynomial we had to solve, we finally obtain four solutions of the camera pose $(\boldsymbol{R}, \boldsymbol{t})$ with respect to three scene points
- Typically we use a fourth 3d point to determine which of the four solutions is the correct one

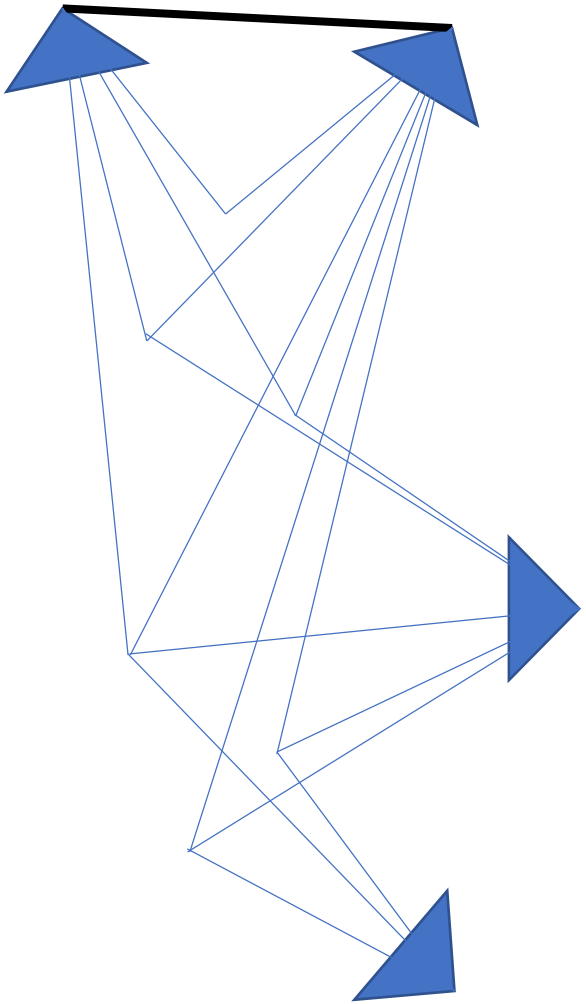
Pose estimation in OpenCV



Processing an image sequence

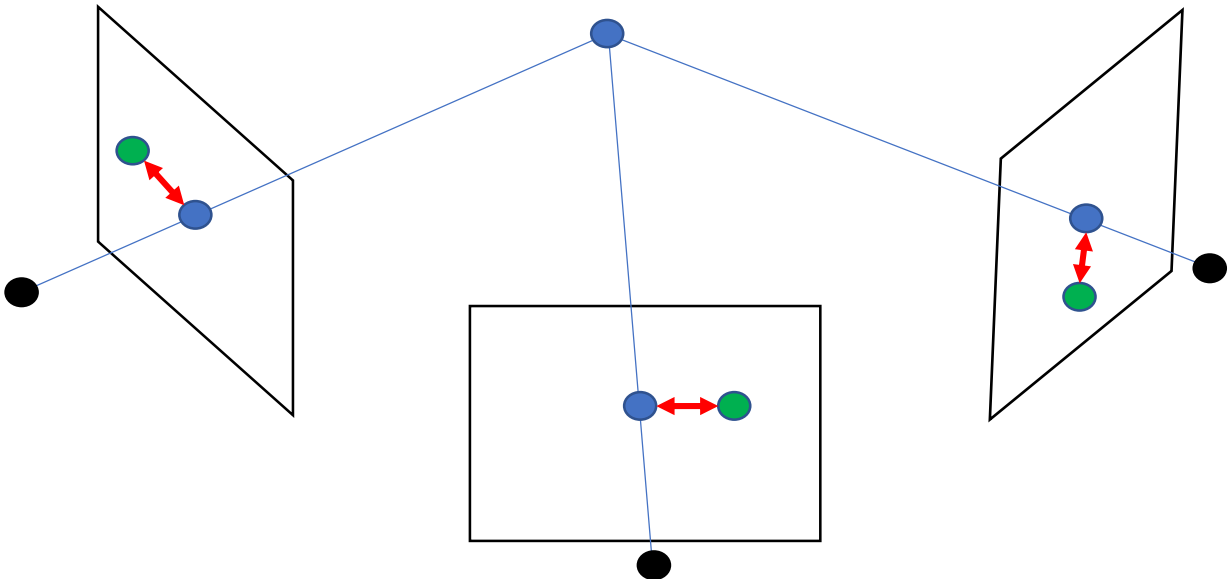
The following procedure should be used to obtain a metric reconstruction from an image sequence

1. Calibrate the camera to maximum accuracy
2. Select in initial image pair and calculate the essential matrix, use RANSAC to eliminate outliers, fix the length of the baseline to the desired unit
3. Determine 3d scene points in the coordinate system determined by the first camera
4. Estimate the pose of all cameras where enough 3d points are visible, use RANSAC to eliminate outliers
5. Iterate steps 3. and 4. until no cameras can be added



Reprojection error

- So far we have developed a robust (if we use RANSAC in every step), but only approximate solution for recovering scene points and camera poses from point correspondences across a sequence of images
- With this approximate solution we can now re-project all these 3d points into all images and determine the distance of the reprojection to the actual feature points
- This **reprojection error** should be minimised for an optimal solution



Bundle adjustment

- The reprojection of a point $\boldsymbol{X_i}$ into the image $\boldsymbol{P_j}$ is

$$\boldsymbol{x_{ij}} = \boldsymbol{P_j X_i}$$

- Or in non-homogeneous coordinates

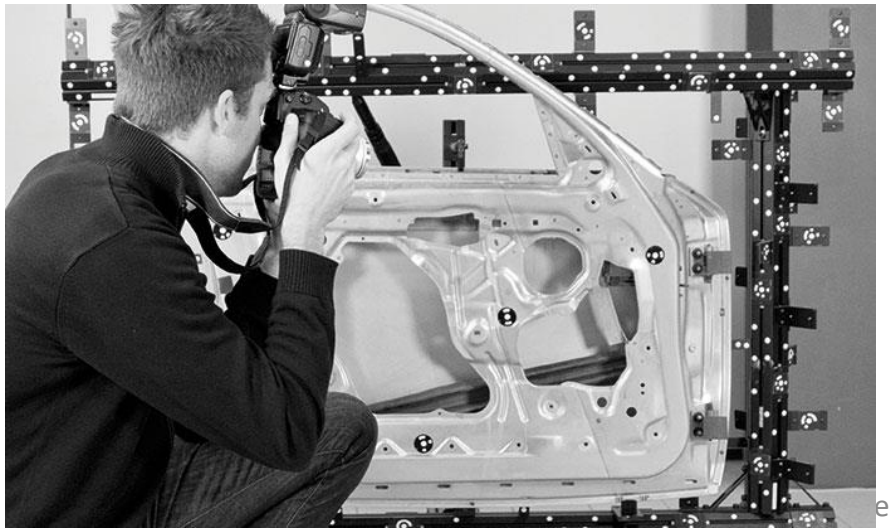
$$\boldsymbol{x_{ij}} = f\left[\boldsymbol{K R^T}\left[\boldsymbol{m_j}\right]\left(\boldsymbol{X_i} - \boldsymbol{t_j}\right)\right]$$

- using the normalisation function $f[x_1, x_2, x_3] = \begin{pmatrix} x_1/x_3 \\ x_2/x_3 \end{pmatrix}$
- The parameter estimation problem with observations $\boldsymbol{x_{ij}}$ and unknown parameters $\boldsymbol{X_i}$, $\boldsymbol{m_j}$, $\boldsymbol{t_j}$ is called **bundle adjustment**
- It minimises the reprojection error over all images

Bundle adjustment

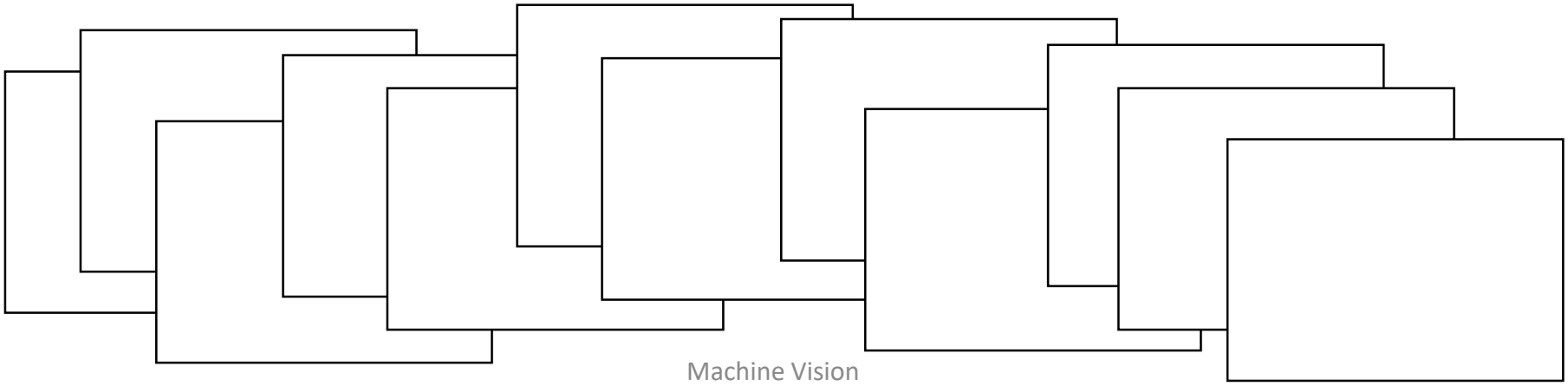
- Because a metric scene reconstruction is only defined up to 7 parameters of a common similarity transformation we need to define **gauge constraints** to fix the coordinate system
- We can either choose to fix one of the cameras and length of one baseline or we use more than three control points, which are known 3d coordinates of object points and introduce additional observations to fix the coordinate system

$$\overline{\boldsymbol{X_i}} = \boldsymbol{X_i}$$



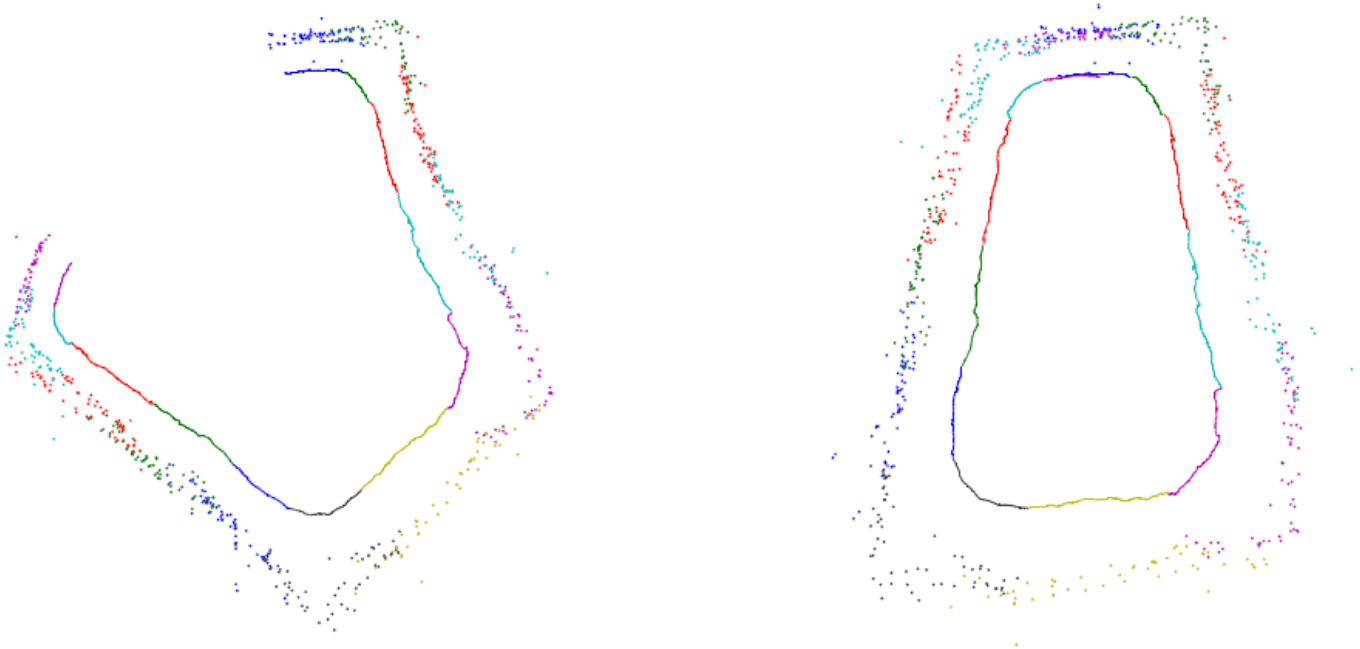
Bundle adjustment

- The matrix inversion in each iteration of the Gauss-Helmert-Model is roughly cubic in the number of images
- Therefore bundle-adjustment is a relatively costly operation, in particular for long image sequences
- The correlation between far away images in a sequence is usually small, resulting in very sparse normal equation matrices
- This sparsity needs to be exploited in order to be able to solve larger bundle adjustment systems

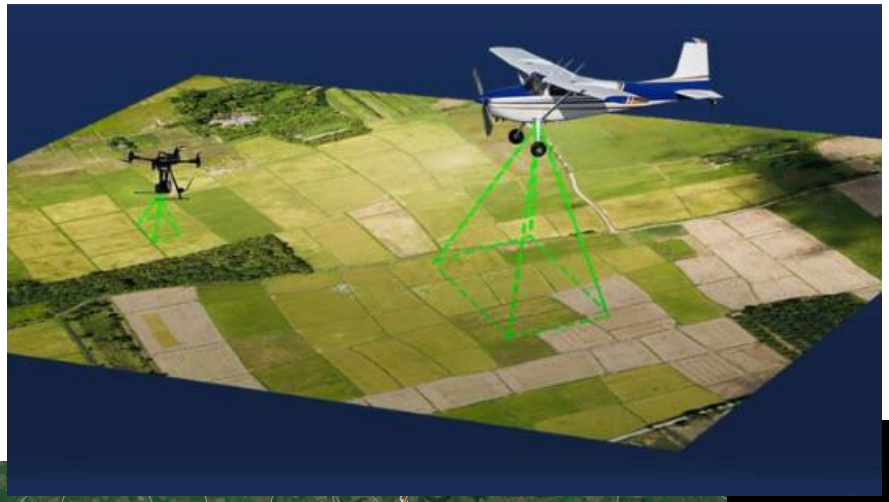


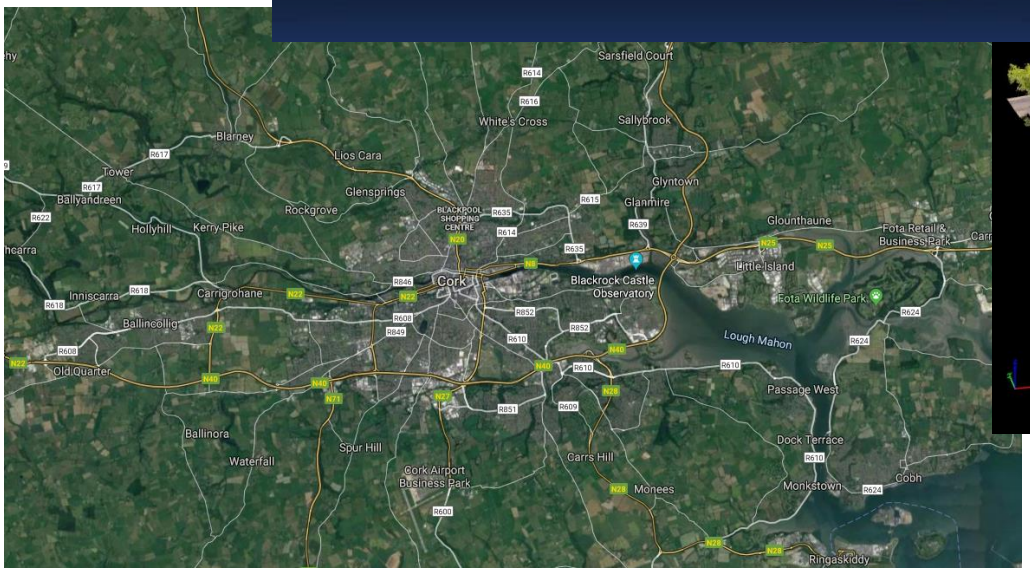
Bundle adjustment

- When building a metric reconstruction from tracked features, errors tend to accumulate over time
- It is therefore necessary to **close loops** using feature matching between far away frames using wide-baseline matching if possible

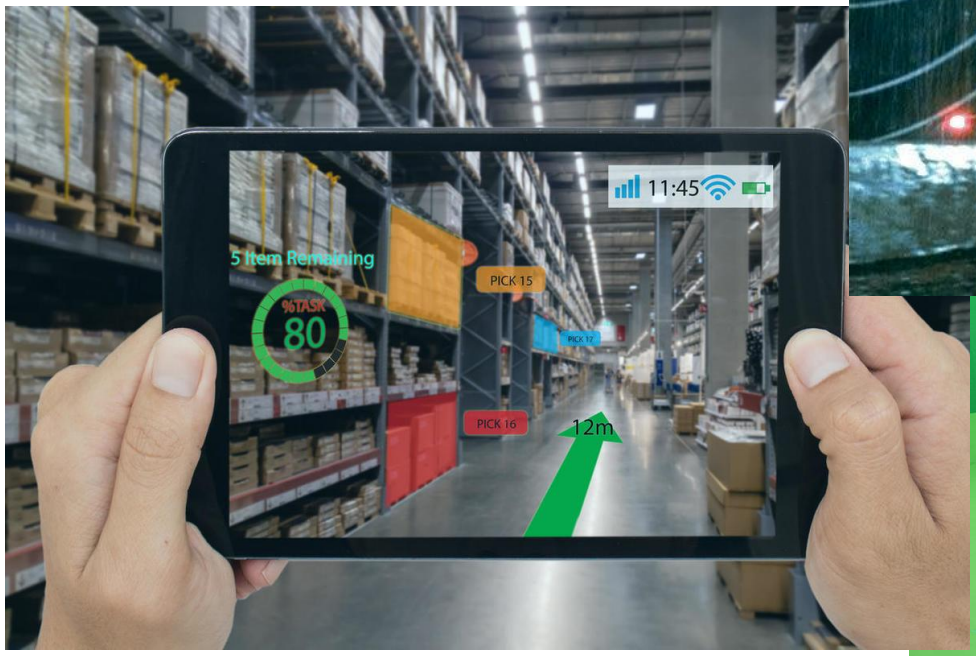


Application: Mapping





Application: Augmented Reality



Thank you for your attention!