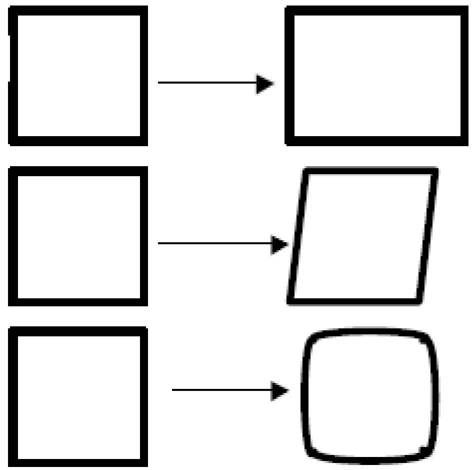
Camera model & calibration

Prof. Didier Stricker

Camera Calibration

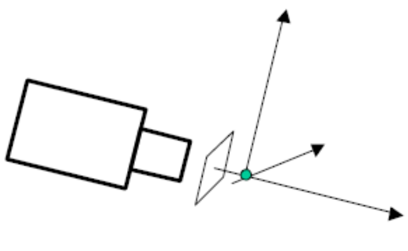
- Finding camera's internal parameters that effect the imaging process.
  1. Position of the image center on the image (usually not just (width/2, height/2).
  2. Focal length
  3. Scaling factors for row and column pixels.
  4. Lens distortion.

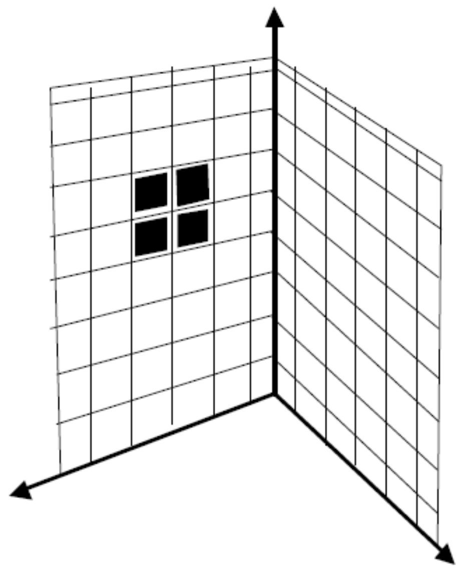


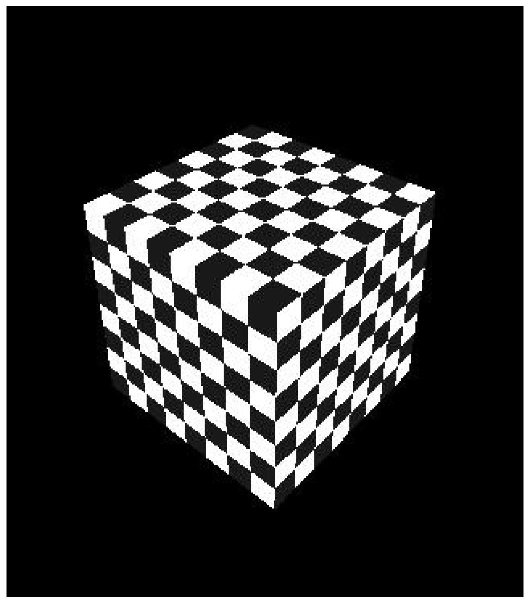
Calibration Procedure

Calibration: finding the function (defined by the camera) that maps 3D points to 2D image plane.

First thing required is to obtain pairs of corresponding 3D and 2D points. (X_1, Y_1, Z_1) (x_1, y_1), (X_2, Y_2, Z_2) (x_2, y_2), ….







Calibration Procedure

Calibration Target: Two perpendicular planes with chessboard pattern.

1. We know the 3D positions of the corners with respect to a coordinates system defined on the target.

2. Place a camera in front of the target and we can locate the corresponding corners on the image. This gives us the correspondences.

3. Recover the equation that describes imaging projection and camera's internal parameters. At the same time, also recover the relative orientation between the camera and the target (pose).

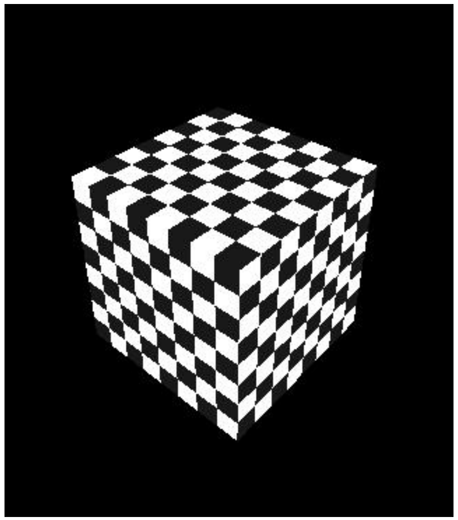
Finding Corners

1. Corner detector
2. Canny Edge detector plus fitting lines to the detected edges. Find the intersections of the lines.
3. Manual input.

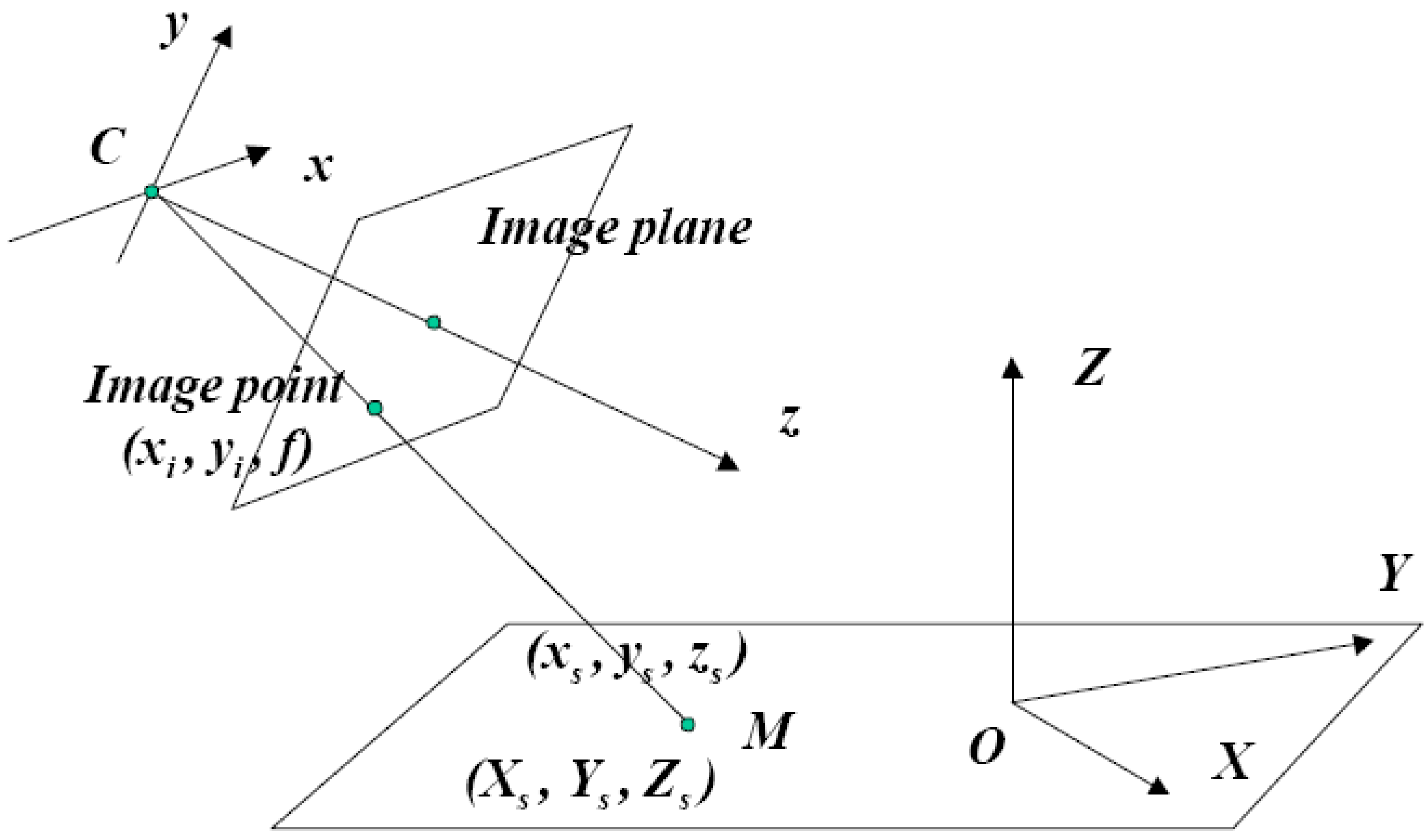
Matching 3D and 2D points (we know the number of corners) by counting. This gives corresponding pairs

(world point) < --- > (image point)

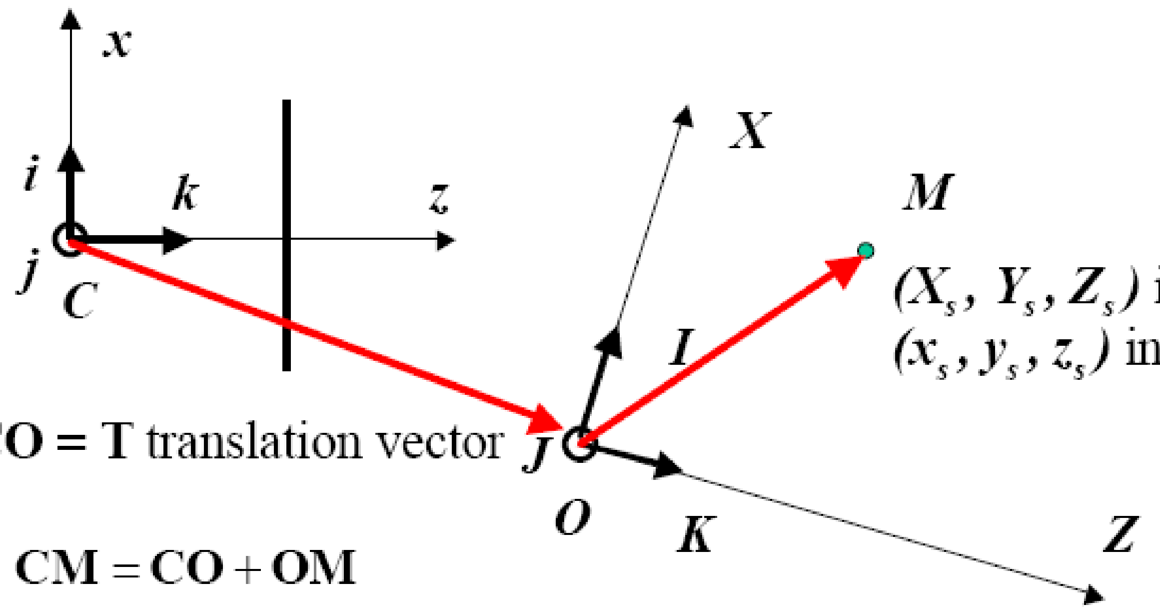
(X_1, Y_1, Z_1) (x_1, y_1),



World Coordinates and Camera Coordinates



Camera Frame to World Frame



(X_s, Y_s, Z_s) in world cordinate system
(x_s, y_s, z_s) in image cordinate system

CO = T translation vector

$$\mathbf{CM} = \mathbf{CO} + \mathbf{OM}$$

$$x_S\mathbf{i} + y_S\mathbf{j} + z_S\mathbf{k} = T_x\mathbf{i} + T_y\mathbf{j} + T_z\mathbf{k} + X_S\mathbf{I} + Y_S\mathbf{J} + Z_S\mathbf{K}$$

$$x_S = T_x + X_S\mathbf{I.i} + Y_S\mathbf{J.i} + Z_S\mathbf{K.i}$$

$$\begin{bmatrix} x_S \\ y_S \\ z_S \end{bmatrix} = \begin{bmatrix} T_x \\ T_y \\ T_z \end{bmatrix} + \begin{bmatrix} \mathbf{I.i} & \mathbf{J.i} & \mathbf{K.i} \\ \mathbf{I.j} & \mathbf{J.j} & \mathbf{K.j} \\ \mathbf{I.k} & \mathbf{J.k} & \mathbf{K.k} \end{bmatrix} \begin{bmatrix} X_S \\ Y_S \\ Z_S \end{bmatrix}$$

Rotation matrix

Definition: 3D-Rotation

- Linear Algebra

- Definition: a matrix R is a rotation matrix if and only if it is a orthogonal matrix with determinant +1

Orthogonal Matrix: a square matrix with real entries whose columns and rows are orthogonal vectors with length 1.

That means: $RR^T = I$

Or: $R^{-1} = R^T$

3D Transformations - Rotation

- Euler Angles for Rotation $R=R_zR_yR_x$:

rotation by ψ about the z axis: $$R_z = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

rotation by θ about the y axis: $$R_y = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}$$

rotation by ϕ about the x axis: $$R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix}$$

Sequence not standardized! Many different conventions!

Homogeneous Coordinates

$$\begin{bmatrix} x_S \\ y_S \\ z_S \end{bmatrix} = \begin{bmatrix} T_x \\ T_y \\ T_z \end{bmatrix} + \begin{bmatrix} \mathbf{I.i} & \mathbf{J.i} & \mathbf{K.i} \\ \mathbf{I.j} & \mathbf{J.j} & \mathbf{K.j} \\ \mathbf{I.k} & \mathbf{J.k} & \mathbf{K.k} \end{bmatrix} \begin{bmatrix} X_S \\ Y_S \\ Z_S \end{bmatrix}$$

$$\begin{bmatrix} x_S \\ y_S \\ z_S \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{I.i} & \mathbf{J.i} & \mathbf{K.i} & T_x \\ \mathbf{I.j} & \mathbf{J.j} & \mathbf{K.j} & T_y \\ \mathbf{I.k} & \mathbf{J.k} & \mathbf{K.k} & T_z \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} X_S \\ Y_S \\ Z_S \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} x_S \\ y_S \\ z_S \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R} & \mathbf{T} \\ \mathbf{0}_3^{\mathbf{T}} & 1 \end{bmatrix} \begin{bmatrix} X_S \\ Y_S \\ Z_S \\ 1 \end{bmatrix}$$

Let **C** = - $\mathbf{R}^t$ **T**

$$\begin{bmatrix} x_S \\ y_S \\ z_S \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R} & -\mathbf{R}\widetilde{\mathbf{C}} \\ \mathbf{0}_3^{\mathbf{T}} & 1 \end{bmatrix} \begin{bmatrix} X_S \\ Y_S \\ Z_S \\ 1 \end{bmatrix}$$

Perspective Projection

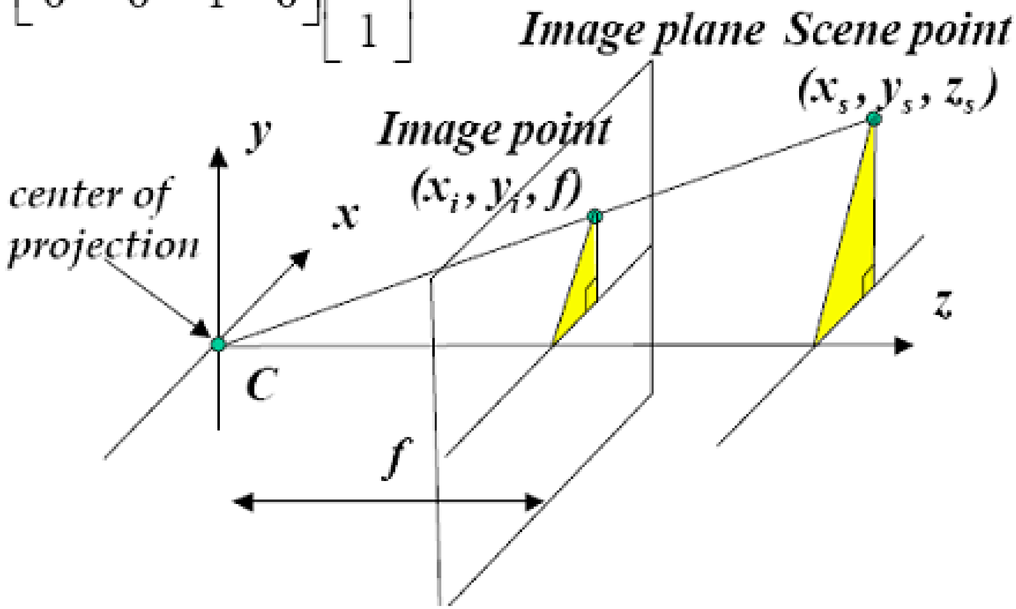
Use Homogeneous coordinates, the perspective projection becomes linear.

$$x_i = f\frac{x_s}{z_s}$$

$$y_i = f\frac{y_s}{z_s}$$

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_s \\ y_s \\ z_s \\ 1 \end{bmatrix}$$

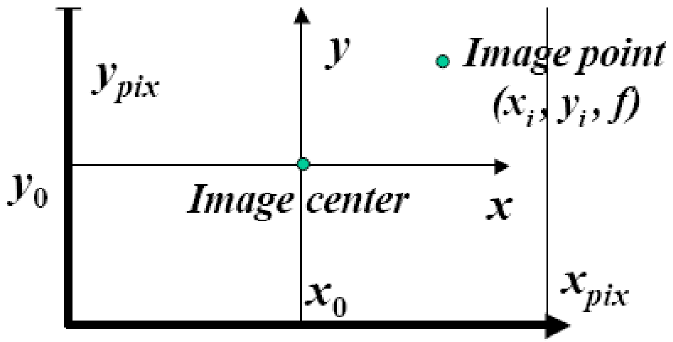
$$x_i = u / w, \quad y_i = v / w$$



Pixel Coordinates

Transformation uses:

- image center (x_0, y_0)
- scaling factors k_x and k_y



$$x_i = f\,\frac{x_s}{z_s} \qquad x_{pix} = k_x x_i + x_0 = f\,k_x\,\frac{x_s + z_s x_0}{z_s}$$

$$y_i = f\,\frac{y_s}{z_s} \qquad y_{pix} = k_y y_i + y_0 = f\,k_y\,\frac{y_s + z_s y_0}{z_s}$$

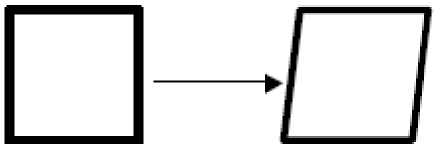
$$\begin{bmatrix} u' \\ v' \\ w' \end{bmatrix} = \begin{bmatrix} \alpha_x & 0 & x_0 & 0 \\ 0 & \alpha_y & y_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_s \\ y_s \\ z_s \\ 1 \end{bmatrix} \quad \text{with} \quad \begin{aligned} \alpha_x &= f\,k_x \\ \alpha_y &= f\,k_y \end{aligned} \quad \text{then} \quad \begin{aligned} x_{pix} &= u'/w' \\ y_{pix} &= v'/w' \end{aligned}$$

Calibration Matrix

$$\begin{bmatrix} u' \\ v' \\ w' \end{bmatrix} = \begin{bmatrix} \alpha_x & s & x_0 & 0 \\ 0 & \alpha_y & y_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_s \\ y_s \\ z_s \\ 1 \end{bmatrix} \quad \text{with} \quad \begin{aligned} \alpha_x &= f\,k_x \\ \alpha_y &= f\,k_y \end{aligned} \qquad \begin{aligned} x_{pix} &= u'/w' \\ y_{pix} &= v'/w' \end{aligned}$$

$$\begin{bmatrix} \alpha_x & s & x_0 & 0 \\ 0 & \alpha_y & y_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} \alpha_x & s & x_0 \\ 0 & \alpha_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} = \mathbf{K}\left[\mathbf{I}_3 \quad | \quad \mathbf{0}_3\right]$$

- α_x and α_y "focal lengths" in pixels
- x_0 and y_0 coordinates of image center in pixels
- Added parameter s is skew parameter



- **K** is called *calibration matrix.* **Five degrees of freedom.**
  - **K** is a 3x3 upper triangular matrix

Putting Everything Together

$$\begin{bmatrix} u' \\ v' \\ w' \end{bmatrix} = \mathbf{K}\left[\mathbf{I}_3 \quad | \quad \mathbf{0}_3\right] \begin{bmatrix} x_s \\ y_s \\ z_s \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} x_S \\ y_S \\ z_S \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R} & -\mathbf{R}\widetilde{\mathbf{C}} \\ \mathbf{0}_3^\mathbf{T} & 1 \end{bmatrix} \begin{bmatrix} X_S \\ Y_S \\ Z_S \\ 1 \end{bmatrix}$$

$$\Rightarrow \begin{bmatrix} u' \\ v' \\ w' \end{bmatrix} = \mathbf{K}\left[\mathbf{I}_3 \quad | \quad \mathbf{0}_3\right] \begin{bmatrix} \mathbf{R} & -\mathbf{R}\widetilde{\mathbf{C}} \\ \mathbf{0}_3^\mathbf{T} & 1 \end{bmatrix} \begin{bmatrix} X_S \\ Y_S \\ Z_S \\ 1 \end{bmatrix}$$

$$\Rightarrow \begin{bmatrix} u' \\ v' \\ w' \end{bmatrix} = \mathbf{P} \begin{bmatrix} X_S \\ Y_S \\ Z_S \\ 1 \end{bmatrix}$$

$$\mathbf{x} = \mathbf{P}\,\mathbf{X}$$

Calibration

1. Estimate matrix **P** using scene points and their images

2. Estimate the intrinsic and extrinsic parameters

$$\mathbf{P} = \mathbf{K}\,\mathbf{R}\left[\mathbf{I}_3 \quad | \quad -\tilde{\mathbf{C}}\right]$$

Left 3x3 sub-matrix is the product of an upper triangular matrix and an orthogonal matrix.

Computing the Matrix P

- Use corresponding image and scene points
  - 3D points $\mathbf{X_i}$ in world coordinate system
  - Images $\mathbf{x_i}$ of $\mathbf{X_i}$ in image
- Write $\mathbf{x_i} = \mathbf{P}\,\mathbf{X_i}$ for all i

Computing the Matrix P

- $\mathbf{x_i} = \mathbf{P}\ \mathbf{X_i}$ involves homogeneous coordinates, thus $\mathbf{x_i}$ and $\mathbf{P}\ \mathbf{X_i}$ just have to be proportional: $\mathbf{x}_i \times \mathbf{P}\,\mathbf{X}_i = 0$
- Let $\mathbf{p}_1{}^{\mathrm{T}}$, $\mathbf{p}_2{}^{\mathrm{T}}$, $\mathbf{p}_3{}^{\mathrm{T}}$ be the 3 row vectors of $\mathbf{P}$

$$\mathbf{P}\,\mathbf{X}_i = \begin{bmatrix} \mathbf{p}_1^{\mathrm{T}}\mathbf{X}_i \\ \mathbf{p}_2^{\mathrm{T}}\mathbf{X}_i \\ \mathbf{p}_3^{\mathrm{T}}\mathbf{X}_i \end{bmatrix} \qquad \mathbf{x}_i \times \mathbf{P}\,\mathbf{X}_i = \begin{bmatrix} v'_i\,\mathbf{p}_3^{\mathrm{T}}\mathbf{X}_i - w'_i\,\mathbf{p}_2^{\mathrm{T}}\mathbf{X}_i \\ w'_i\,\mathbf{p}_1^{\mathrm{T}}\mathbf{X}_i - u'_i\,\mathbf{p}_3^{\mathrm{T}}\mathbf{X}_i \\ u'_i\,\mathbf{p}_2^{\mathrm{T}}\mathbf{X}_i - v'_i\,\mathbf{p}_1^{\mathrm{T}}\mathbf{X}_i \end{bmatrix}$$

$$\Rightarrow \begin{bmatrix} \mathbf{0}_4^{\mathrm{T}} & -w'_i\,\mathbf{X}_i^{\mathrm{T}} & v'_i\,\mathbf{X}_i^{\mathrm{T}} \\ w'_i\,\mathbf{X}_i^{\mathrm{T}} & \mathbf{0}_4^{\mathrm{T}} & -u'_i\,\mathbf{X}_i^{\mathrm{T}} \\ -v'_i\,\mathbf{X}_i^{\mathrm{T}} & u'_i\,\mathbf{X}_i^{\mathrm{T}} & \mathbf{0}_4^{\mathrm{T}} \end{bmatrix} \begin{bmatrix} \mathbf{p}_1 \\ \mathbf{p}_2 \\ \mathbf{p}_3 \end{bmatrix} = 0 \qquad \begin{bmatrix} \mathrm{p}_1 \\ \mathrm{p}_2 \\ \mathrm{p}_3 \end{bmatrix} \text{is a } 12\times 1 \text{ vector}$$

Computing the Matrix P

- Third row can be obtained from sum of u'_i times first row - v'_i times second row

Rank 2

$$\begin{bmatrix} \mathbf{0}_4^T & -w'_i \mathbf{X}_i^T & v'_i \mathbf{X}_i^T \\ w'_i \mathbf{X}_i^T & \mathbf{0}_4^T & -u'_i \mathbf{X}_i^T \\ -v'_i \mathbf{X}_i^T & u'_i \mathbf{X}_i^T & \mathbf{0}_4^T \end{bmatrix} \begin{bmatrix} \mathbf{p}_1 \\ \mathbf{p}_2 \\ \mathbf{p}_3 \end{bmatrix} = 0$$

- So we get 2 independent equations in 11 unknowns (ignoring scale)
- With 6 point correspondences, we get enough equations to compute matrix **P**

$$\mathbf{A}\,\mathbf{p} = 0$$

Computing the Matrix P

- Linear system $\mathbf{A}\,\mathbf{p} = 0$
- When possible, have at least 5 times as many equations as unknowns (28 points)
- Minimize $\|\,\mathbf{A}\,\mathbf{p}\,\|$ with the constraint $\|\,\mathbf{p}\,\| = 1$
  - P is the unit singular vector of **A** corresponding to the smallest singular value (the last column of **V**, where $\mathbf{A} = \mathbf{U}\,\mathbf{D}\,\mathbf{V}^{\mathbf{T}}$ is the SVD of **A**)

Computing the translation component

- Find homogeneous coordinates of C in the scene
- **C** is the null vector of matrix **P**
  - $\mathbf{P}\,\mathbf{C} = 0$:

$$\mathbf{P} = \mathbf{K}\,\mathbf{R}\left[\mathbf{I}_3 \quad | \quad -\tilde{\mathbf{C}}\right]$$

$$\begin{bmatrix} 1 & 0 & 0 & X_c \\ 0 & 1 & 0 & Y_c \\ 0 & 0 & 1 & Z_c \end{bmatrix} \begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

- Find null vector **C** of **P** using SVD
  - **C** is the unit singular vector of **P** corresponding to the smallest singular value (the last column of **V**, where $\mathbf{P} = \mathbf{U}\,\mathbf{D}\,\mathbf{V}^{\mathbf{T}}$ is the SVD of **P**)

Camera matrix decomposition

Finding the camera orientation and internal parameters

A **QR decomposition** (also called a **QR factorization**) of a matrix is a decomposition of the matrix into an orthogonal and a right triangular matrix.

$$KR$$

(use **RQ decomposition** ~QR)

(if only QR, invert)

$$\boxed{} = (\boxed{Q}\ R)^{-1} = R^{-1}\ \boxed{Q}^{-1}$$

Computing the Matrix P

Further Improvement

Use as initialization for nonlinear minimization of $\sum_{\mathbf{i}} d(\mathbf{x_i}, \mathbf{PX_i})^2$

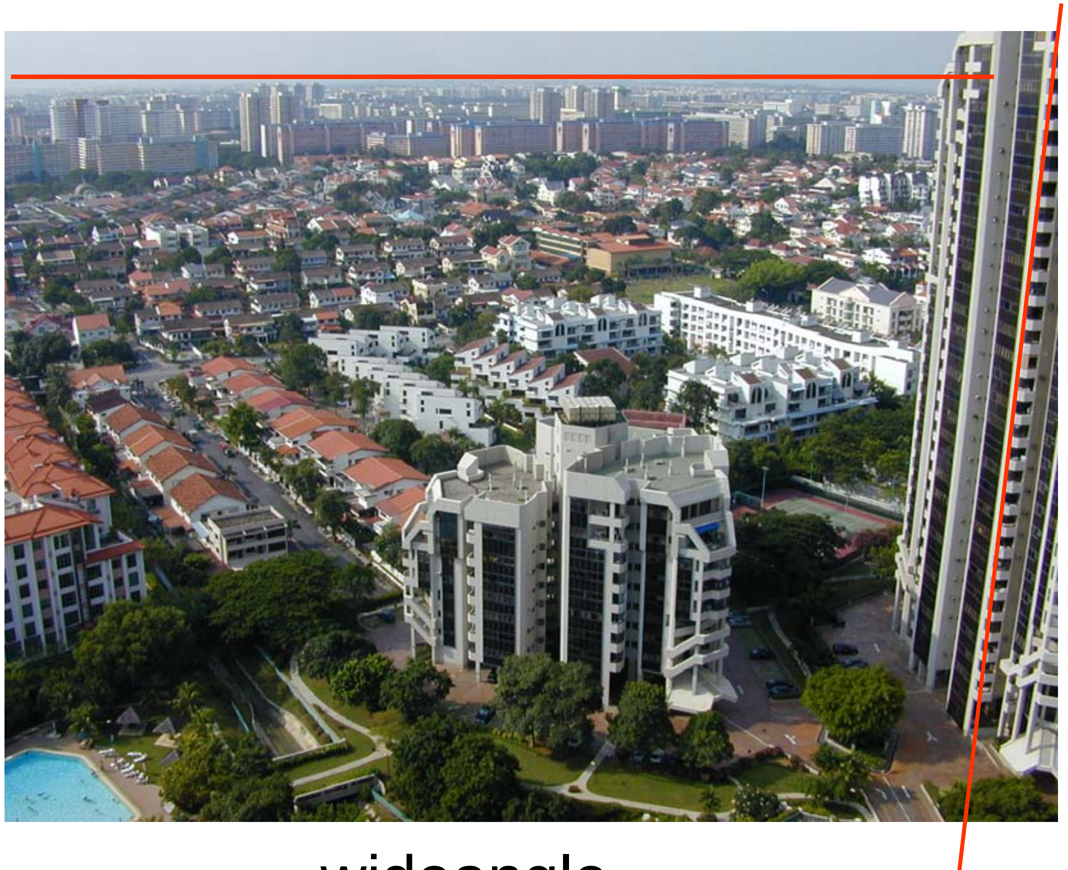
Most popular non-linear minimization algorithm is the **Levenberg-Marquart minimization**

LM is more robust to local minima than e.g. the **Gauss–Newton algorithm** and the method of **gradient descent**)

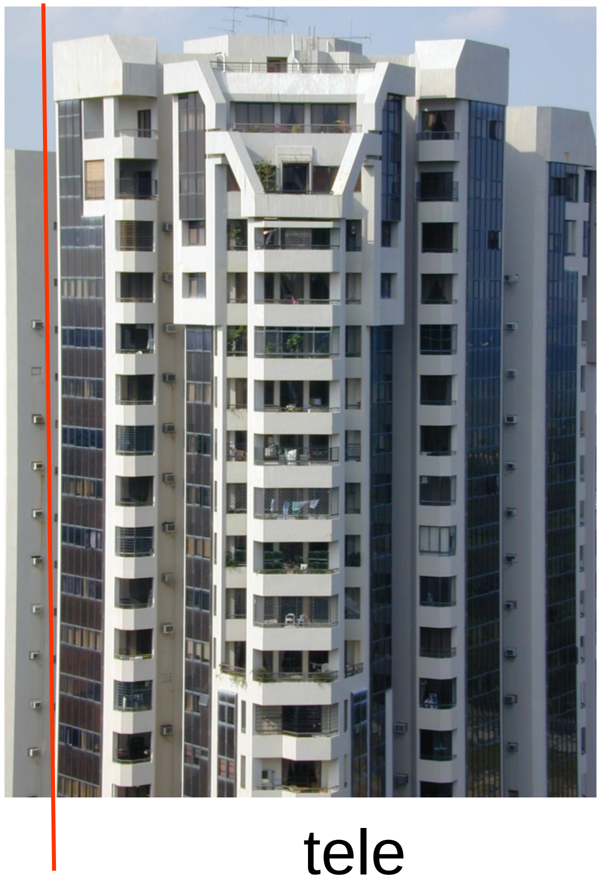
Advanced Calibration: Nonlinear Distortions

- Barrel and Pincushion
- Tangential

Barrel and Pincushion Distortion



wideangle



tele

Models of Radial Distortion

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} x_d \\ y_d \end{pmatrix} (1 + k_1 r^2 + k_2 r^4)$$

distance from center

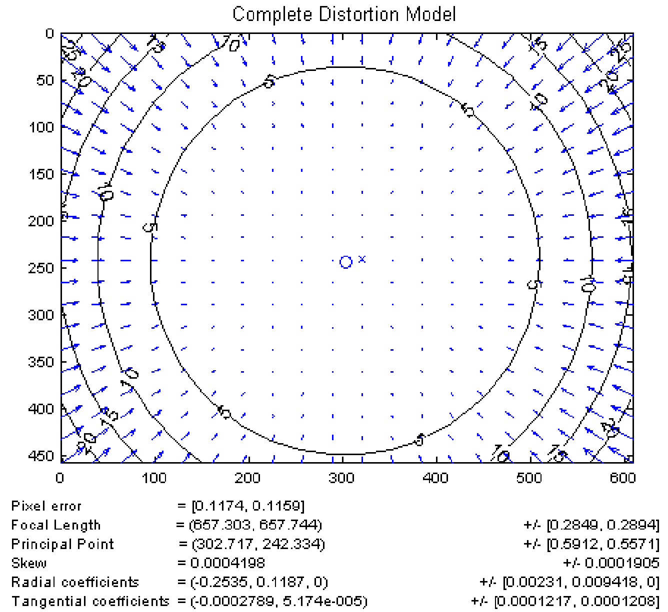
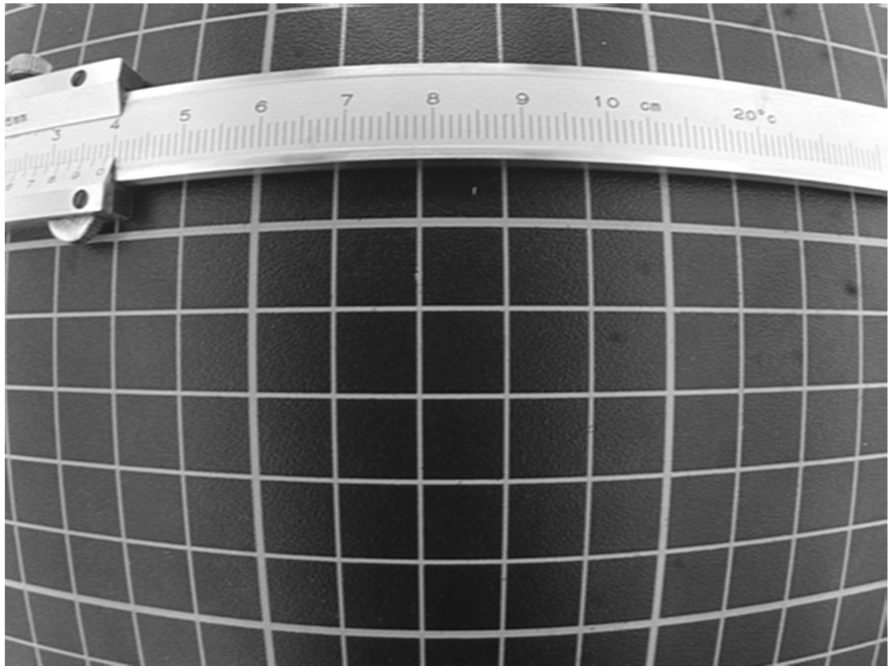
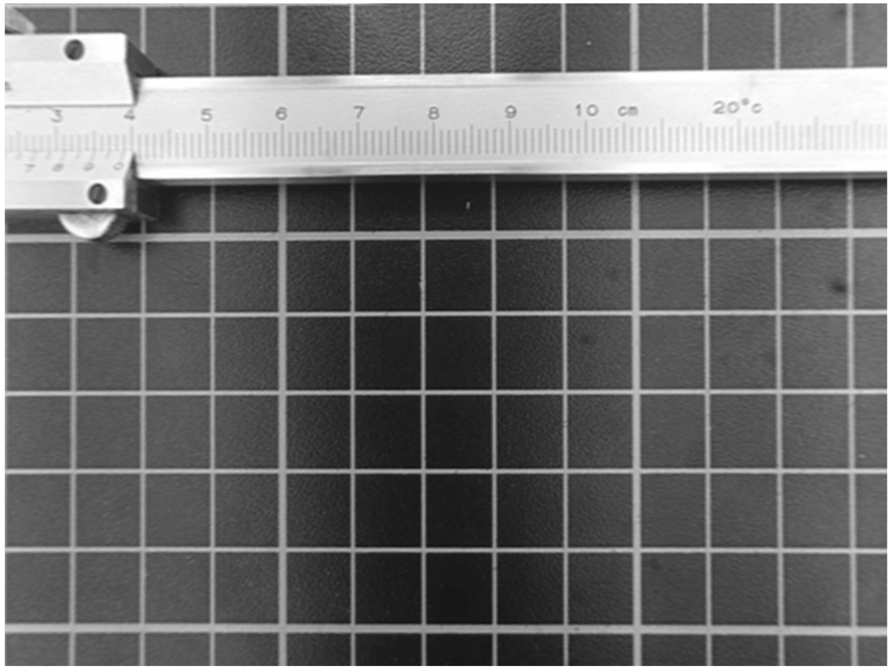
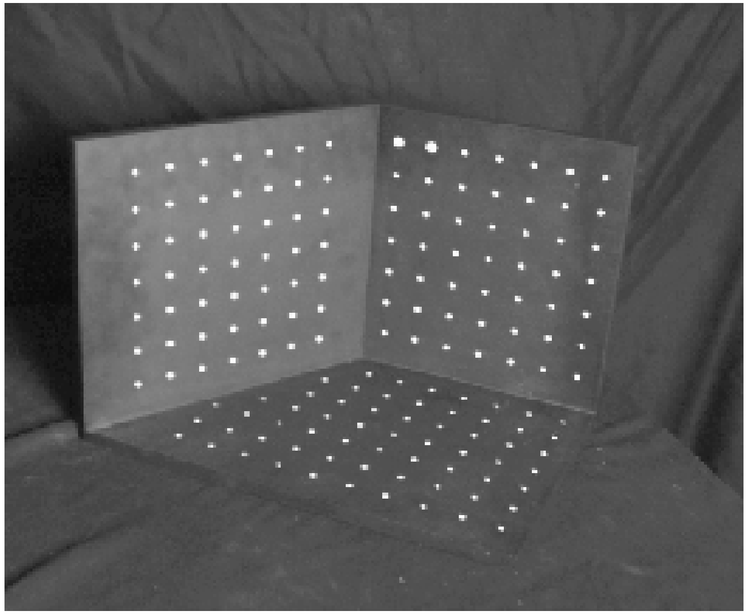


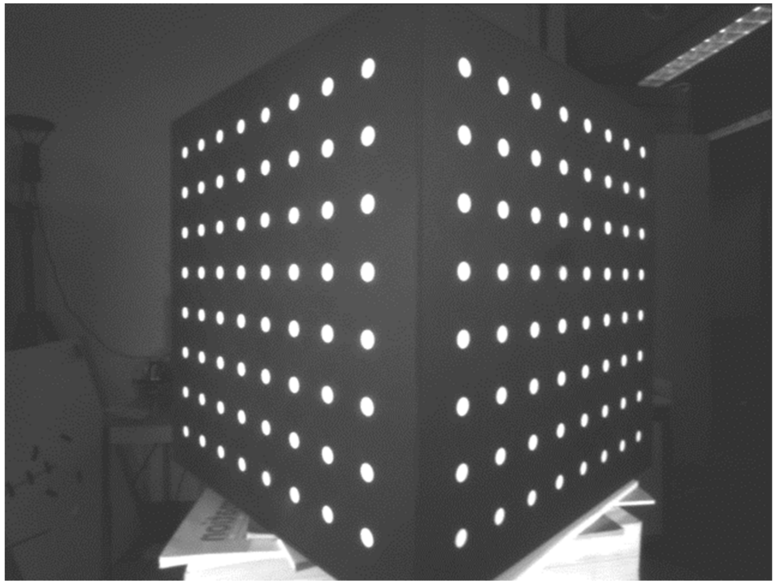
Image Rectification





Examplary calibration set-ups





Direct linear calibration

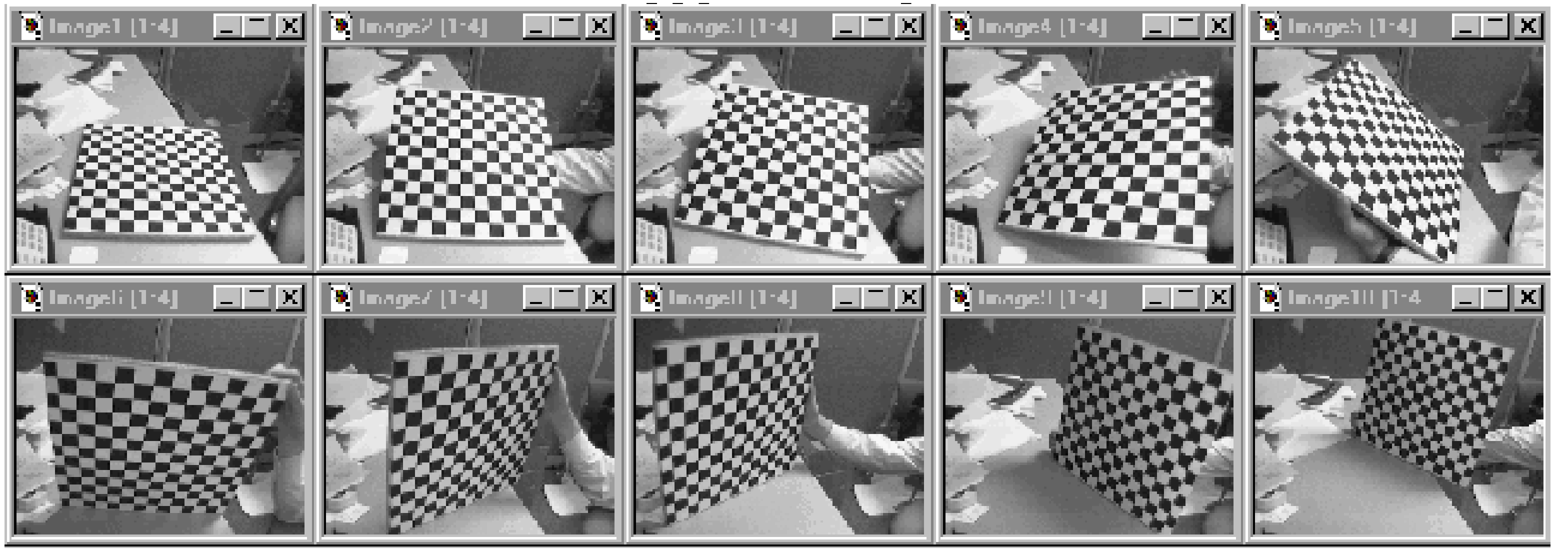
- Advantage:
  - Very simple to formulate and solve

- Disadvantages:
  - Doesn’t tell you the camera parameters
  - Doesn’t model radial distortion
  - Hard to impose constraints (e.g., known focal length)
  - Doesn’t minimize the right error function

For these reasons, *nonlinear methods* are preferred

- Define error function E between projected 3D points and image positions
  - E is nonlinear function of intrinsics, extrinsics, radial distortion
- Minimize E using nonlinear optimization techniques
  - e.g., variants of Newton’s method (e.g., Levenberg Marquart)

Alternative: multi-plane



Images courtesy Jean-Yves Bouguet, Intel Corp.

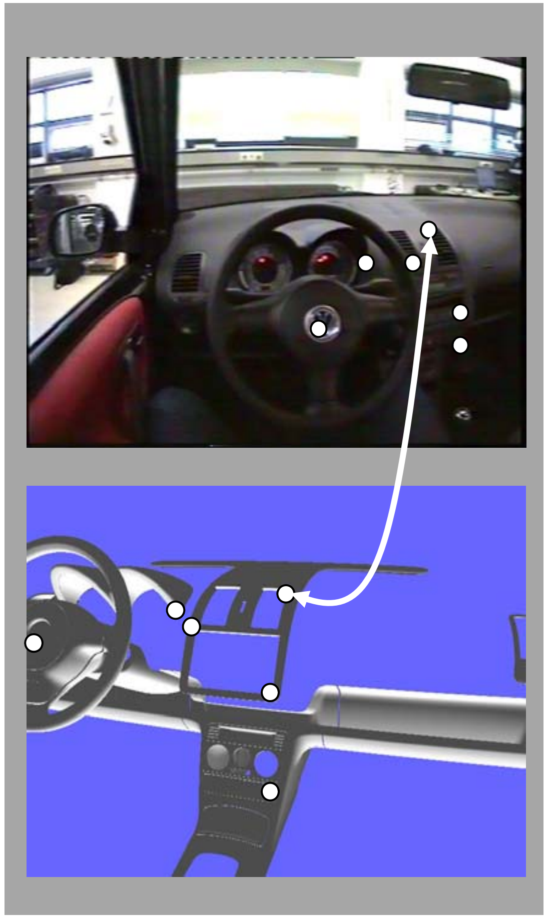
Advantage

- Only requires a plane
- Don’t have to know positions/orientations
- Good code available online!
  - Intel’s OpenCV library: http://www.intel.com/research/mrl/research/opencv/
  - Matlab version by Jean-Yves Bouget: http://www.vision.caltech.edu/bouguetj/calib_doc/index.html
  - Zhengyou Zhang’s web site: http://research.microsoft.com/~zhang/Calib/

Calibration or Pose estimation using a CAD model

- Reduce the number of interactions
- Pre-compute as much parameters as possible:
  - More stable
  - Less user-input required

Calibration or Pose estimation using a CAD model



THANK YOU!