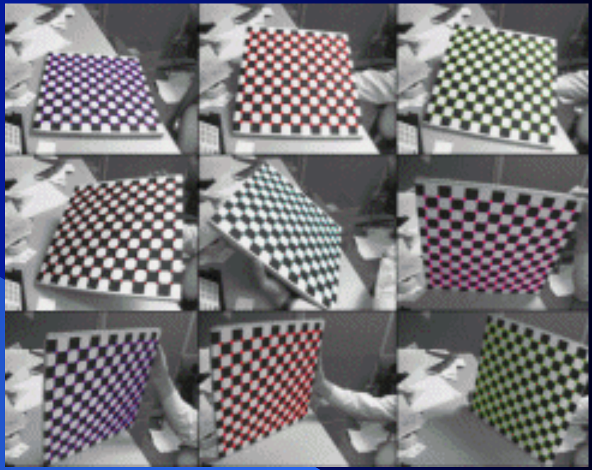
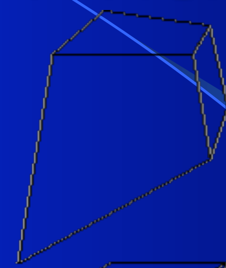
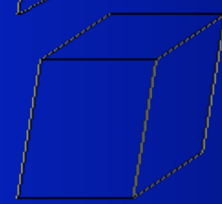
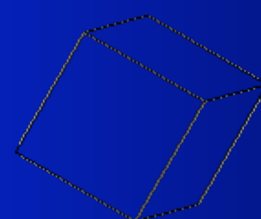
# Computer Vision

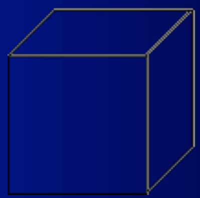
Lecture X

Camera Calibration



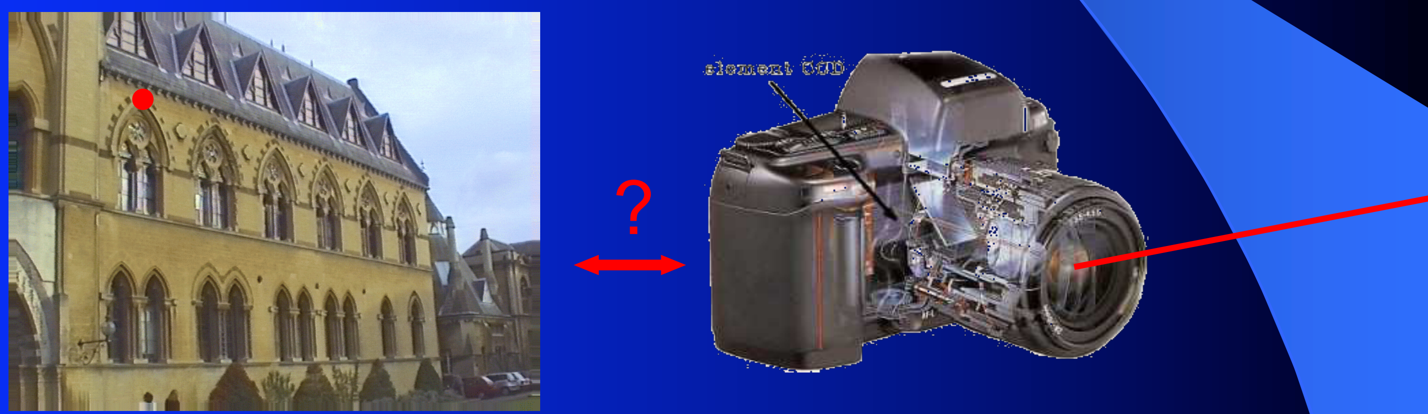
# 3D Transformations

| | | | |
|---|---|---|---|
| Projective<br>15dof | $\begin{bmatrix} A & t \\ v^{\mathsf{T}} & v \end{bmatrix}$ |  | Intersection and tangency |
| Affine<br>12dof | $\begin{bmatrix} A & t \\ 0^{\mathsf{T}} & 1 \end{bmatrix}$ |  | Parallelism of planes,<br>Volume ratios, centroids,<br>The plane at infinity $\pi_\infty$ |
| Similarity<br>7dof | $\begin{bmatrix} sR & t \\ 0^{\mathsf{T}} & 1 \end{bmatrix}$ |  | Angles, ratios of length<br>The absolute conic $\Omega_\infty$ |
| Euclidean<br>6dof | $\begin{bmatrix} R & t \\ 0^{\mathsf{T}} & 1 \end{bmatrix}$ |  | Volume |

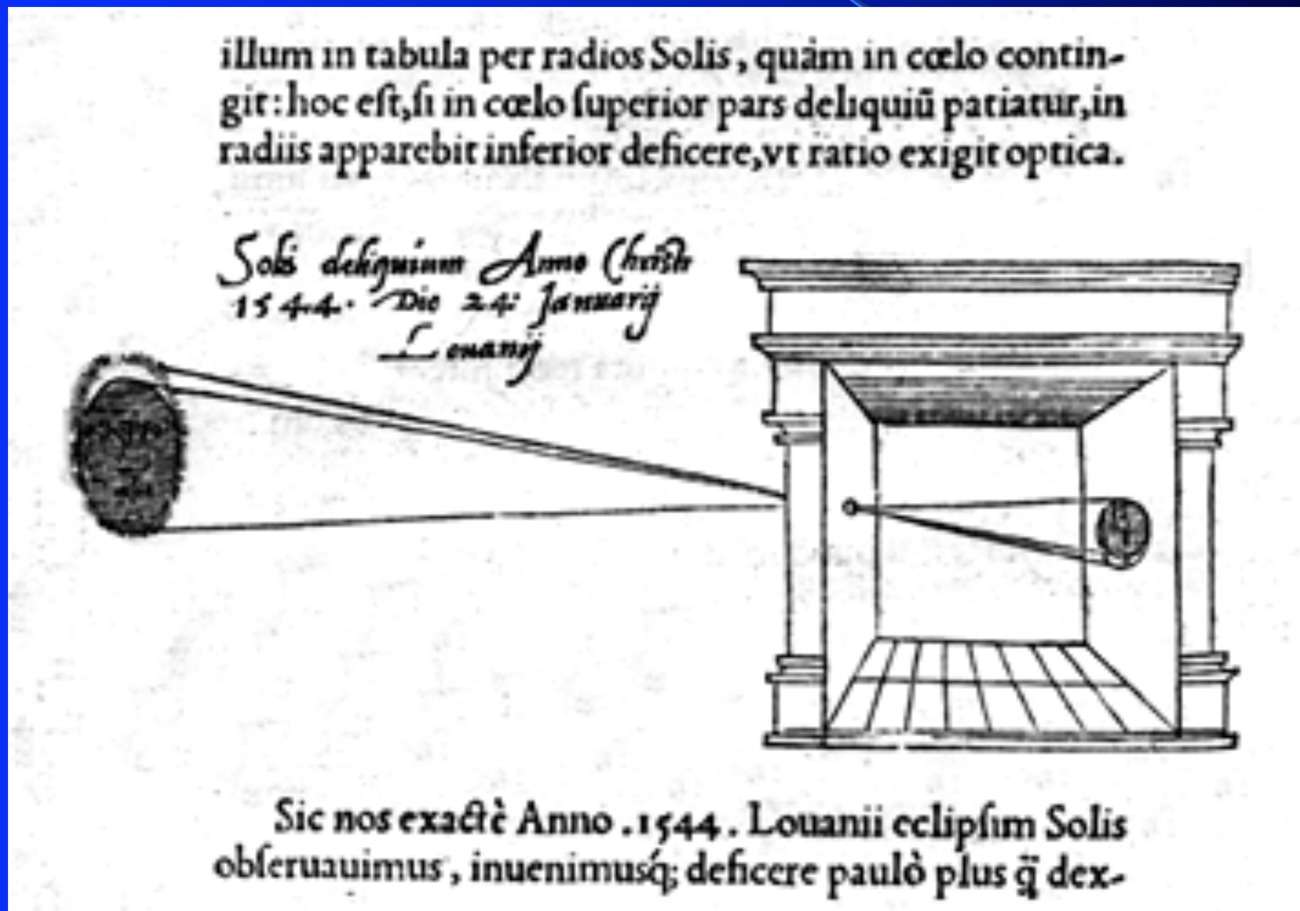


# Camera Calibration

Compute relation between pixels and rays in space



# Pinhole Camera

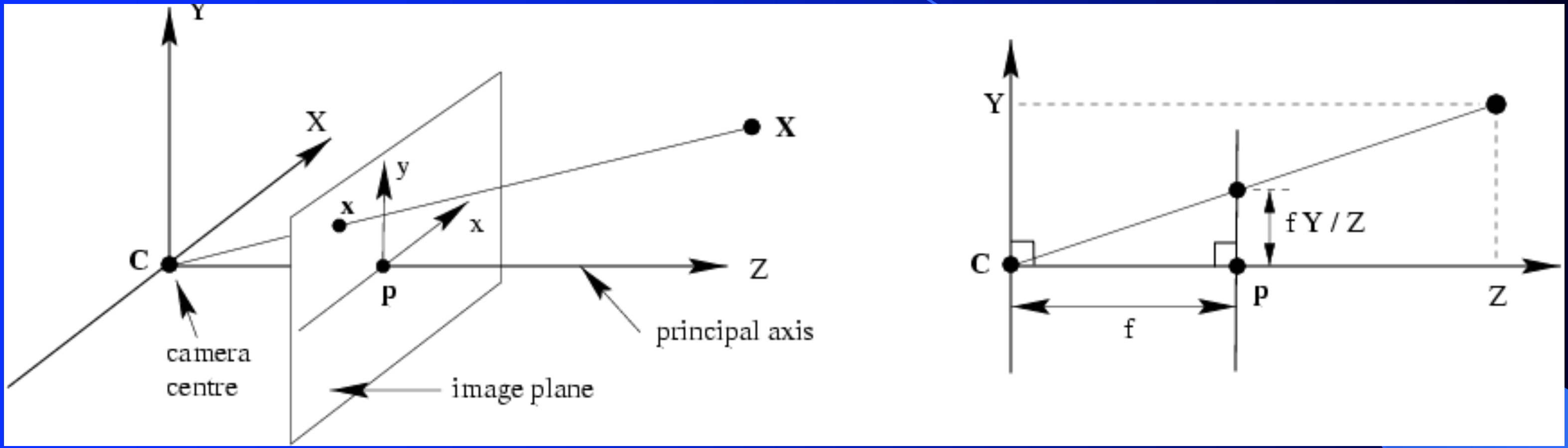


illum in tabula per radios Solis, quàm in cœlo contin-
git: hoc eſt, ſi in cœlo ſuperior pars deliquiũ patiatur, in
radiis apparebit inferior deficere, vt ratio exigit optica.



Sic nos exactè Anno .1544. Louanii eclipſim Solis
obſeruauimus, inuenimusq; deficere paulò plus q̃ dex-

# Pinhole Camera Model



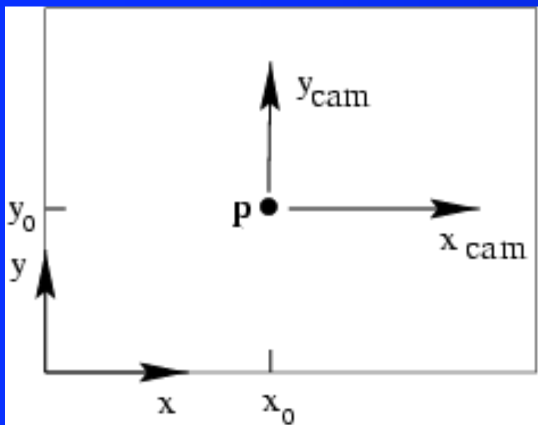

non-homogeneous coordinates

$$(X,Y,Z)^T \mapsto (fX/Z, fY/Z)^T$$

homogeneous coordinates

$$\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \mapsto \begin{pmatrix} fX \\ fY \\ Z \end{pmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

# Principal Point Offset



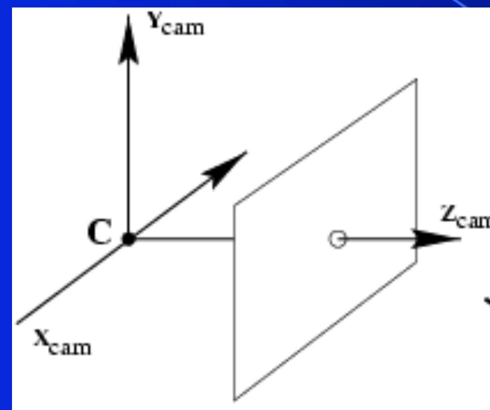

$$(X,Y,Z)^T \mapsto (fX/Z + c_x, fY/Z + c_y)^T$$

$(c_x, c_y)^T$ principal point

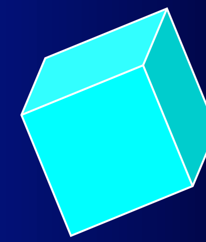
$$\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \mapsto \begin{pmatrix} fX + c_x Z \\ fY + c_x Z \\ Z \end{pmatrix} = \begin{bmatrix} f & 0 & c_x & 0 \\ 0 & f & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

# Object Motion

$$K = \begin{bmatrix} f & 0 & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{bmatrix}$$




$\mathbf{R}, \mathtt{t}$



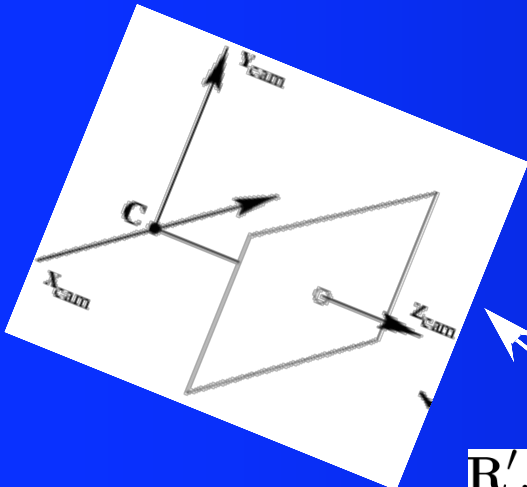
$$\mathtt{x} = \mathbf{K}\begin{bmatrix} \mathbf{I} & \mathtt{0} \end{bmatrix} \mathtt{X}_{\mathtt{cam}}$$

$$\mathtt{X}_{\mathtt{cam}} = \begin{bmatrix} \mathbf{R} & \mathtt{t} \\ \mathtt{0} & \mathtt{1} \end{bmatrix} \mathtt{X}_{\mathtt{obj}}$$

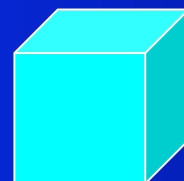
$$\mathtt{x} = \mathbf{K}\begin{bmatrix} \mathbf{I} & \mathtt{0} \end{bmatrix} \begin{bmatrix} \mathbf{R} & \mathtt{t} \\ \mathtt{0} & \mathtt{1} \end{bmatrix} \mathtt{X}_{\mathtt{obj}}$$

$$= \mathbf{K}\begin{bmatrix} \mathbf{R} & \mathtt{t} \end{bmatrix} \mathtt{X}_{\mathtt{obj}}$$

# Camera Motion




$\mathbf{R}', \mathtt{t}' = \mathtt{C}$



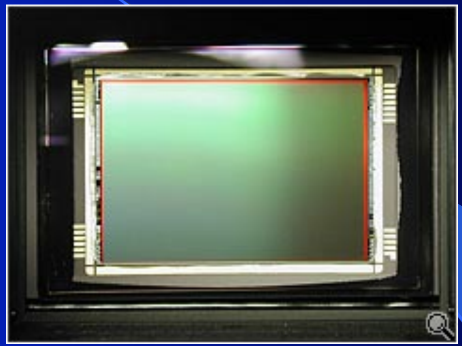
$$K = \begin{bmatrix} f & 0 & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

$$\mathtt{X}_{\mathtt{obj}} = \begin{bmatrix} \mathbf{R}' & \mathtt{t}' \\ 0 & 1 \end{bmatrix} \mathtt{X}_{\mathtt{cam}}$$

$$\mathtt{X}_{\mathtt{cam}} = \begin{bmatrix} \mathbf{R}'^{\top} & -\mathbf{R}'^{\top}\mathtt{t}' \\ 0 & 1 \end{bmatrix} \mathtt{X}_{\mathtt{obj}}$$

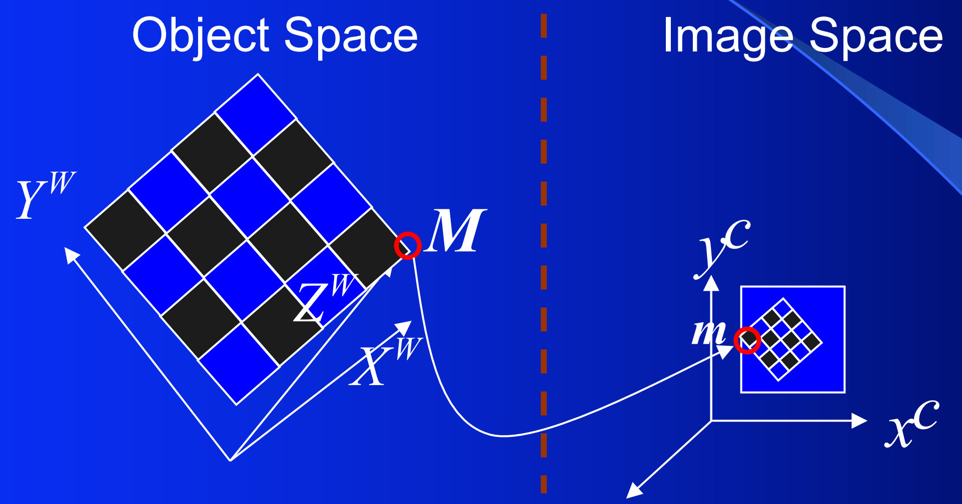
$$\mathtt{x} = \mathbf{K} \begin{bmatrix} \mathbf{R}'^{\top} & -\mathbf{R}'^{\top}\mathtt{t}' \end{bmatrix} \mathtt{X}_{\mathtt{obj}}$$

# CCD Camera

$$K = \begin{bmatrix} m_x & 0 & 0 \\ 0 & m_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f & 0 & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

# Camera Calibration




$$\boldsymbol{m} = [\text{Camera Projection Matrix}] \ \boldsymbol{M}$$

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

$$\mathbf{A} = \underbrace{[\mathbf{R}\ \mathbf{t}]}$$

intrinsic camera parameters

extrinsic camera parameters

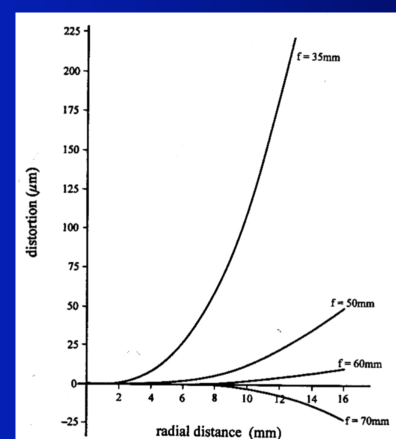
# Radial Distortion

- Due to spherical lenses (cheap)
- Model:

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \sim \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \mathrm{R}\left( \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{R}^\top & -\mathbf{R}^\top \mathbf{t} \\ 0_3^\top & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \right)$$

$$\mathrm{R}(x, y) = (1 + K_1(x^2 + y^2) + K_2(x^2 + y^2)^2 + \ldots) \begin{bmatrix} x \\ y \end{bmatrix}$$

straight lines are not straight anymore




barrel dist.

pincushion dist.

# Radial Distortion Example

# Some Calibration Algorithms

Tsai calibration

Tsai, Roger Y. (1986) "An Efficient and Accurate Camera Calibration Technique for 3D Machine Vision," *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, Miami Beach, FL, 1986, pp. 364–374.

Tsai, Roger Y. (1987) "A Versatile Camera Calibration Technique for High-Accuracy 3D Machine Vision Metrology Using Off-the-Shelf TV Cameras and Lenses," *IEEE Journal of Robotics and Automation*, Vol. RA–3, No. 4, August 1987, pp. 323–344.

Zhang calibration http://research.microsoft.com/~zhang/calib/

Z. Zhang. A flexible new technique for camera calibration. IEEE Transactions on Pattern Analysis and Machine Intelligence, 22(11):1330-1334, 2000.

Z. Zhang. Flexible Camera Calibration By Viewing a Plane From Unknown Orientations. International Conference on Computer Vision (ICCV'99), Corfu, Greece, pages 666-673, September 1999.

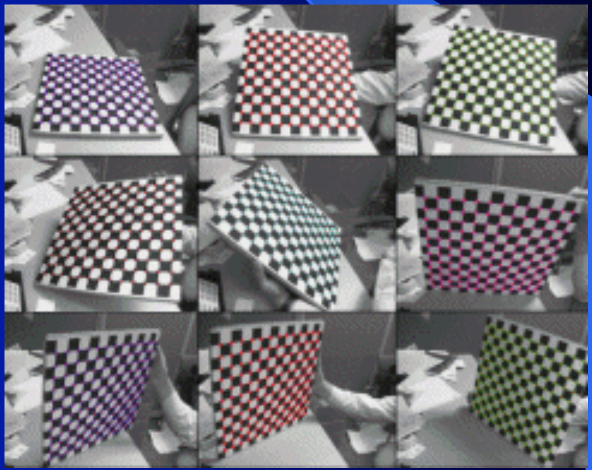
# Camera Calibration from Planar Patterns

- ICCV Zhang'99: "*Flexible Calibration by Viewing a Plane From Unknown Orientations*"

$$\boldsymbol{m} = \underbrace{[\text{Camera Projection Matrix}]}_{\mathbf{K}\ [\mathbf{R}\ \mathbf{t}]}\ \boldsymbol{M}$$

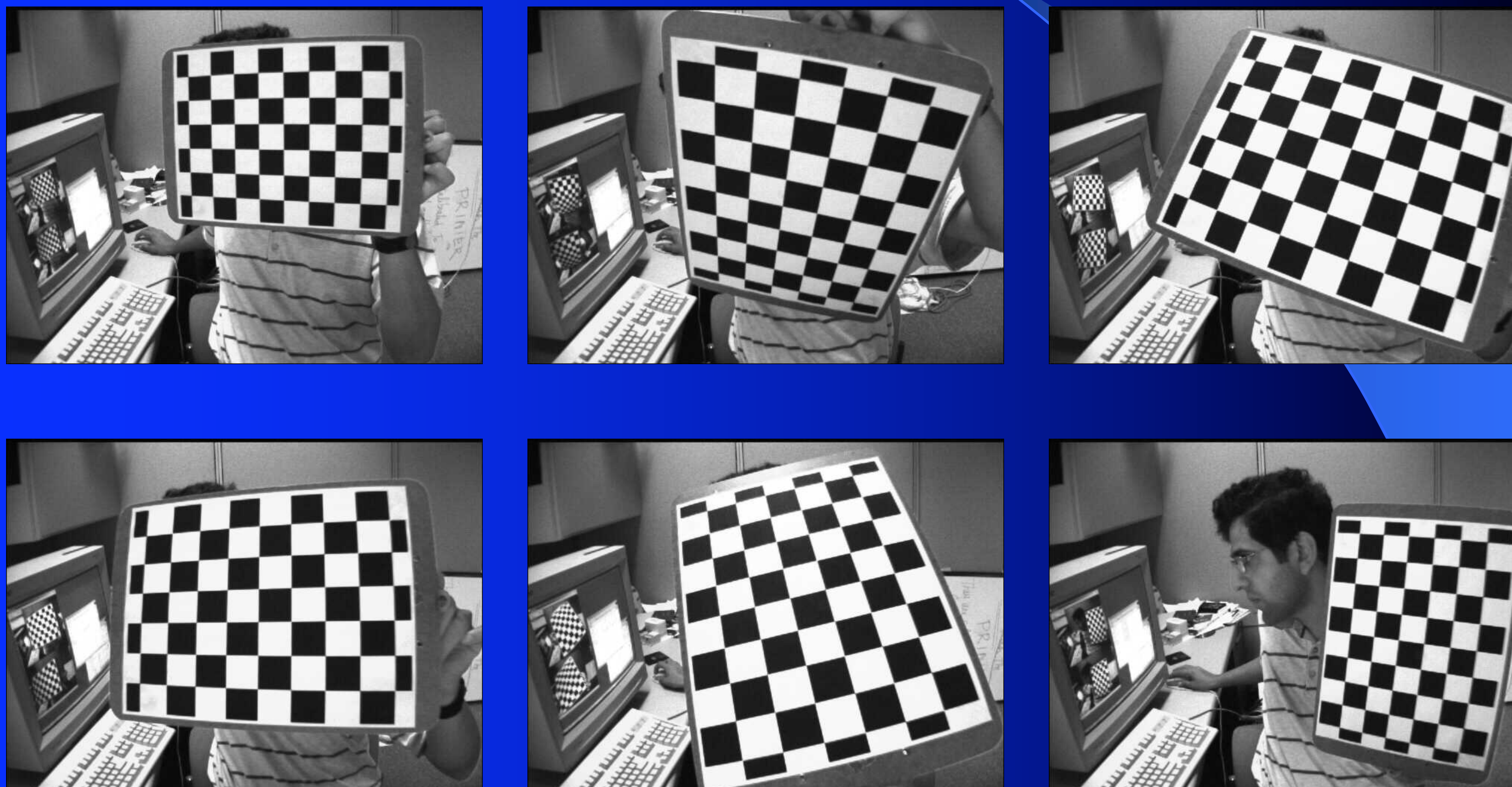
Minimize:

$$\sum_{i=1}^{n}\sum_{j=1}^{m} \|\underbrace{\mathbf{m}_{ij}}_{\textit{observed}} - \underbrace{\hat{\mathbf{m}}(\mathbf{K}, \mathbf{R}_i, \mathbf{t}_i, \mathrm{M}_j)}_{\textit{estimate}:\ \mathbf{K}\ [\mathbf{R}\ \mathbf{t}]\ \boldsymbol{M}}\|^2$$



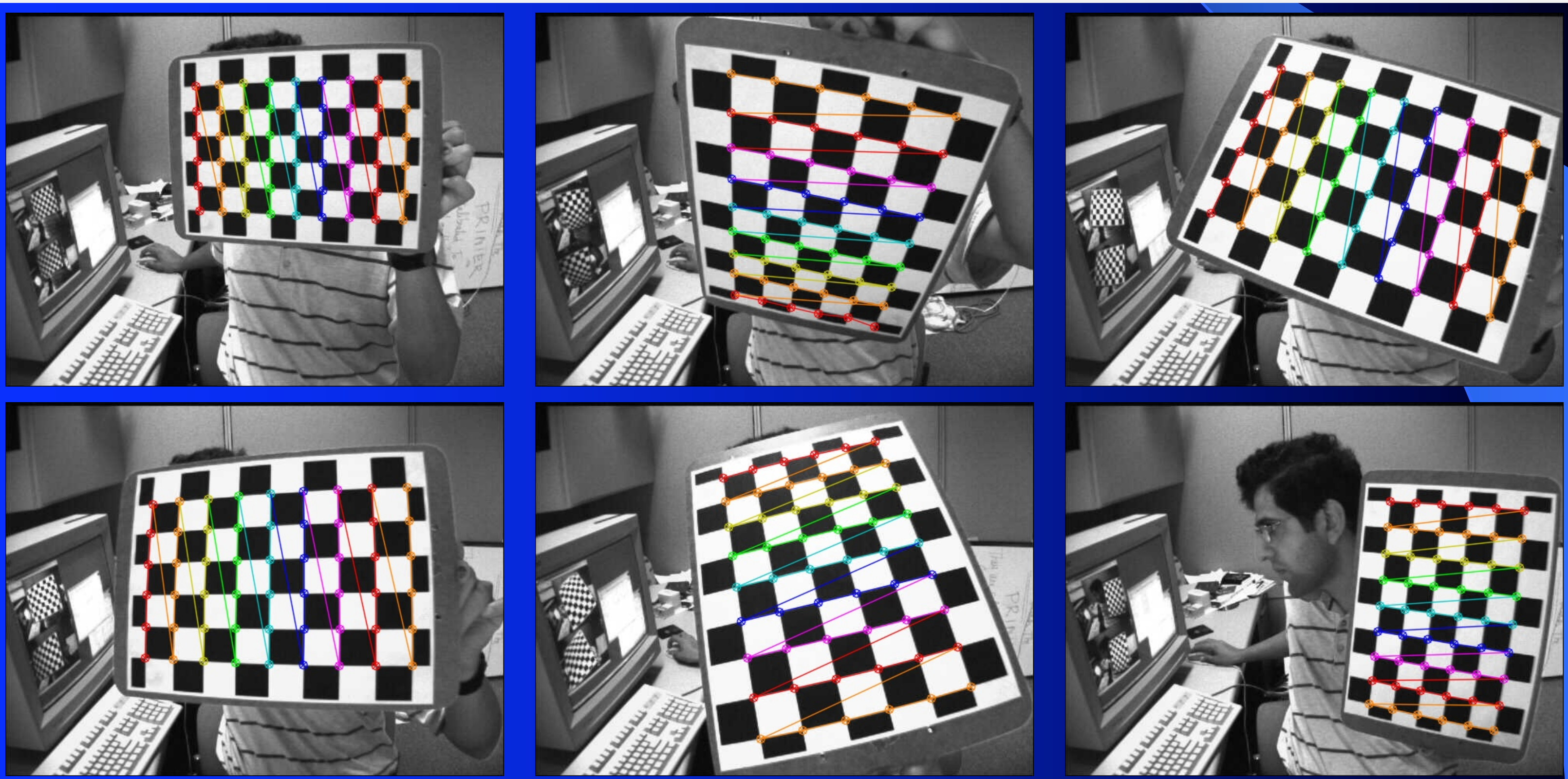
- Two steps:
  - Find an initial solution for K [R t]
  - Minimize the objective function using the initial solution

# Experiment (OpenCV)



# Find Chessboard Corners

```
bool findChessboardCorners( const Mat& image, Size patternSize,
          vector<Point2f>& corners,
          int flags=CV_CALIB_CB_ADAPTIVE_THRESH+
          CV_CALIB_CB_NORMALIZE_IMAGE );
```



# Calibrate Camera

```
double calibrateCamera( const vector<vector<Point3f> >& objectPoints,
          const vector<vector<Point2f> >& imagePoints,
          Size imageSize,
          Mat& cameraMatrix, Mat& distCoeffs,
          vector<Mat>& rvecs, vector<Mat>& tvecs,
          int flags=0 );
```

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = R \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + t$$

$$x' = x/z$$

$$y' = y/z$$

$$u = f_x * x' + c_x$$

$$v = f_y * y' + c_y$$

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = R \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + t$$

$$x' = x/z$$

$$y' = y/z$$

$$x'' = x' \frac{1+k_1r^2+k_2r^4+k_3r^6}{1+k_4r^2+k_5r^4+k_6r^6} + 2p_1x'y' + p_2(r^2 + 2x'^2)$$

$$y'' = y' \frac{1+k_1r^2+k_2r^4+k_3r^6}{1+k_4r^2+k_5r^4+k_6r^6} + p_1(r^2 + 2y'^2) + 2p_2x'y'$$

where $r^2 = x'^2 + y'^2$

$$u = f_x * x'' + c_x$$

$$v = f_y * y'' + c_y$$

# Undistort Image

```
void undistort( const Mat& src, Mat& dst, const Mat& cameraMatrix,
           const Mat& distCoeffs, const Mat& newCameraMatrix=Mat() );
```

# Homework

- ***References***

  [1] Tsai, Roger Y. (1987) "A Versatile Camera Calibration Technique for High-Accuracy 3D Machine Vision Metrology Using Off-the-Shelf TV Cameras and Lenses," IEEE Journal of Robotics and Automation, Vol. RA–3, No. 4, August 1987, pp. 323–344.

  [2] Zhengyou Zhang. Flexible camera calibration by viewing a plane from unknown orientations. In Proceedings of ICCV'99, pages 666–673

- ***Programs***

  [1] Camera Calibration Toolbox for Matlab

  http://www.vision.caltech.edu/bouguetj/calib_doc/

  [2] OpenCV

  http://sourceforge.net/projects/opencvlibrary/