ECE 1390/2390 Image Processing and Computer Vision – Fall 2021

Camera Calibration – Intrinsic calibration

Ahmed Dallal

Reading

- FP chapter 1.2 and 1.3
- Szeliski section 5.2, 5.3

- Today: Really using homogeneous systems to represent projection. And how to do calibration.

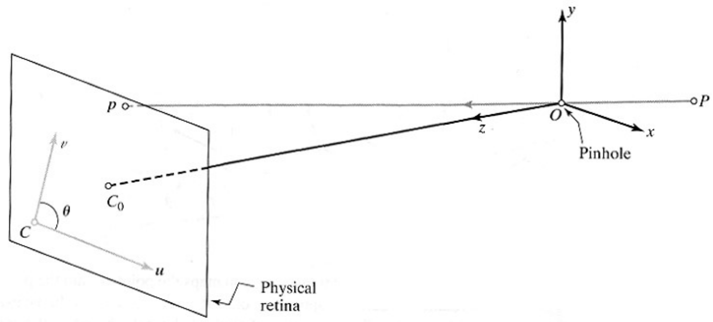
Intrinsic camera calibration

Geometric Camera calibration

Composed of 2 transformations:

- From some (arbitrary) world coordinate system to the camera's 3D coordinate system. *Extrinisic parameters (or camera pose)*
- From the 3D coordinates in the camera frame to the 2D image plane via projection. *Intrinisic parameters*

Camera 3D (x,y,z) to 2D (u,v) or (x',y'): Ideal intrinsic parameters



Ideal Perspective projection

$$u = f\,\frac{x}{z}$$

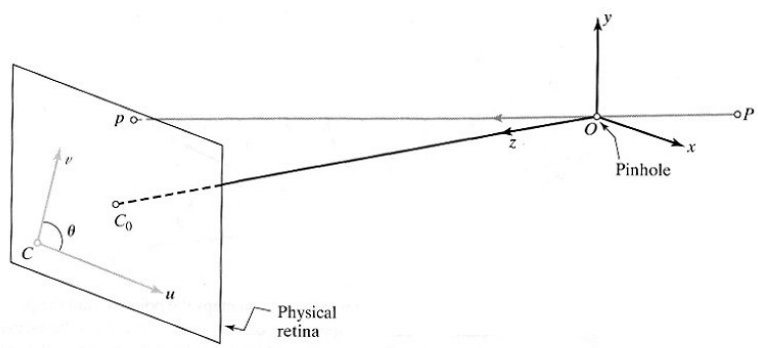
$$v = f\,\frac{y}{z}$$

Real intrinsic parameters(1)

But "pixels" are in some arbitrary spatial units

$$u = \alpha\,\frac{x}{z}$$

$$v = \alpha\,\frac{y}{z}$$



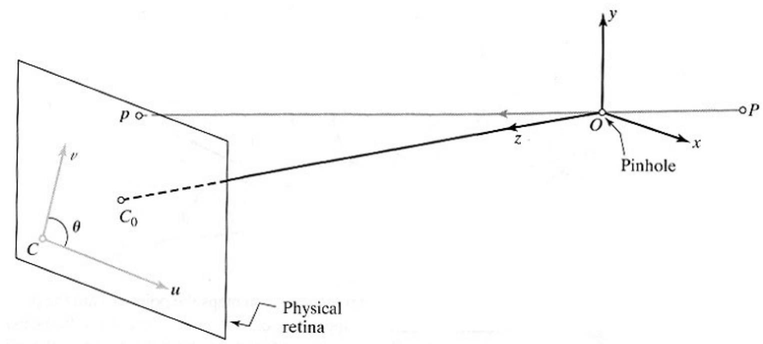
f is given in millimeter, but image locations are represented in pixels. α is related to how many pixels per mm

Real intrinsic parameters(2)

Maybe pixels are not square

$$u = \alpha \frac{x}{z}$$

$$v = \beta \frac{y}{z}$$

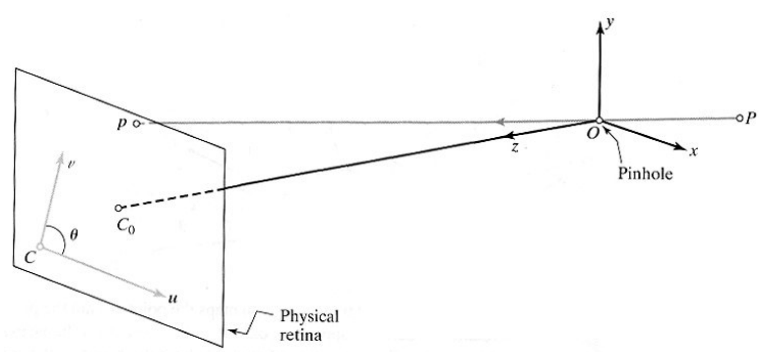


Real intrinsic parameters(3)

We don't know the origin of our camera pixel coordinates

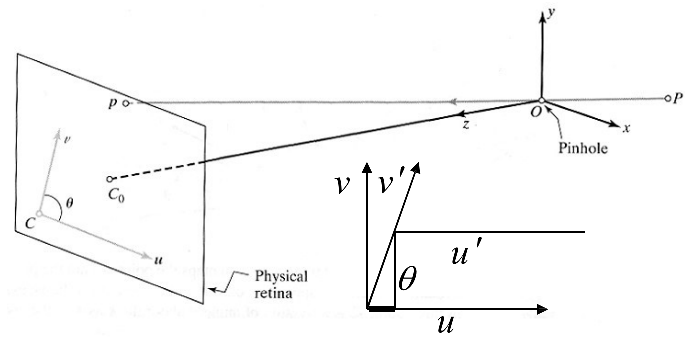
$$u = \alpha \frac{x}{z} + u_0$$

$$v = \beta \frac{y}{z} + v_0$$



Really ugly intrinsic parameters (4)

May be skew between camera pixel axes



$$v'\sin(\theta) = v$$

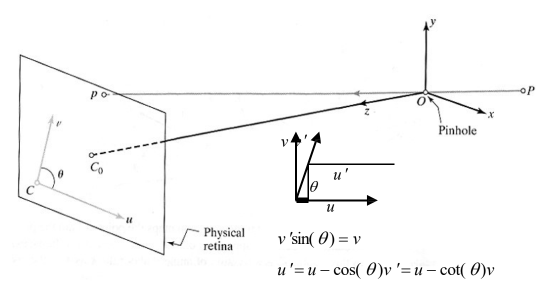
$$u' = u - \cos(\theta)v' = u - \cot(\theta)v$$

Really ugly intrinsic parameters (4)

May be skew between camera pixel axes

$$u = \alpha \frac{x}{z} - \alpha \cot(\theta) \frac{y}{z} + u_0$$

$$v = \frac{\beta}{\sin(\theta)} \frac{y}{z} + v_0$$



Intrinsic parameters, non-homogeneous coords

Notice division by z

$$u = \alpha \frac{x}{z} - \alpha \cot(\theta) \frac{y}{z} + u_0$$

$$v = \frac{\beta}{\sin(\theta)} \frac{y}{z} + v_0$$

Intrinsic parameters, *homogeneous* coords

$$\begin{pmatrix} z * u \\ z * v \\ z \end{pmatrix} = \begin{pmatrix} \alpha & -\alpha\cot(\theta) & u_0 & \boxed{0} \\ 0 & \frac{\beta}{\sin(\theta)} & v_0 & \boxed{0} \\ 0 & 0 & 1 & \boxed{0} \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

$$\vec{p}' = K \; {}^{C}\vec{p}$$

In homogeneous pixels

Intrinsic matrix

In camera-based 3D coords

Kinder, gentler intrinsics

- Can use simpler notation for intrinsics– remove last column which is zero:

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

- f –focal length
  s – skew
  a – aspect ratio
  c_x, c_y – offset

  (5 DOF)

Kinder, gentler intrinsics

- If square pixels, no skew, and optical center is in the center (assume origin in the middle):

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

In this case only one DOF, focal length ***f***

Quiz

The intrinsics have the following: a focal length, a pixel x size, a pixel y size, two offsets and a skew. That's 6. But we've said there are only 5 DOFS. What happened:

a) Because *f* always multiplies the pixel sizes, those 3 numbers are really only 2 DOFs.

b) In modern cameras, the skew is always zero so we don't count it.

c) In CCDs or CMOS cameras, the aspect is carefully controlled to be 1.0, so it is no longer modeled.

Combining extrinsic and intrinsic calibration parameters

Pixels

$$\vec{p}\,' = K\ {}^{C}\vec{p}$$

Intrinsic

World 3D coordinates

Camera 3D coordinates

$$\begin{pmatrix} \\ {}^{C}\vec{p} \\ \\ \\ \end{pmatrix} = \begin{pmatrix} - & - & - & | \\ - & {}^{C}_{W}R & - & {}^{C}_{W}\vec{t} \\ - & - & - & | \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \\ {}^{W}\vec{p} \\ \\ \\ \end{pmatrix}$$

Extrinsic

Combining extrinsic and intrinsic calibration parameters

$$p' = K \left({}^{C}_{W}R \quad {}^{C}_{W}\vec{t} \right) {}^{W}p$$

$$\underset{3\text{x}3}{\text{K}} \qquad \underbrace{\left({}^{C}_{W}R \quad {}^{C}_{W}\vec{t} \right)}_{3\text{x}4}$$

$$\begin{pmatrix} z*u \\ z*v \\ z \end{pmatrix} = p' = M \; {}^{W}p$$

Other ways to write the same equation

pixel coordinates

world coordinates

$$p' = \color{red}{M} \; {}^{W}p$$

Conversion back from homogeneous coordinates leads to:

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} \cong \begin{pmatrix} s*u \\ s*v \\ s \end{pmatrix} = \begin{pmatrix} . & m_1^T & . & . \\ . & m_2^T & . & . \\ . & m_3^T & . & . \end{pmatrix} \begin{pmatrix} {}^{W}p_x \\ {}^{W}p_y \\ {}^{W}p_z \\ 1 \end{pmatrix}$$

projectively similar

$$\begin{cases} u = \dfrac{m_1 \quad P}{m_3 \cdot P} \\ v = \dfrac{m_2 \cdot P}{m_3 \cdot P} \end{cases}$$

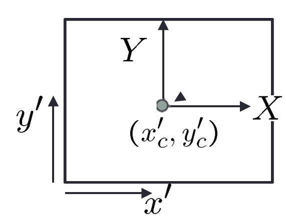
Finally: Camera parameters

- A camera (and its matrix) M (or Π) is described by several parameters
  - Translation **T** of the optical center from the origin of world coordinates
  - Rotation **R** of the camera system
  - focal length and aspect **(f, a)** [or pixel size **(s_x, s_y)**] , principle point **(x'_c, y'_c)**, and skew **(s)**
  - **blue** parameters are called "**extrinsics**," and red are "intrinsics"

Finally: Camera parameters

- Projection equation – the cumulative effect of all parameters:

$$\mathbf{x} = \begin{bmatrix} sx \\ sy \\ s \end{bmatrix} = \begin{bmatrix} * & * & * & * \\ * & * & * & * \\ * & * & * & * \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \mathbf{M}\,\mathbf{X}$$



- The projection matrix models the cumulative effect of all parameters
- Useful to decompose into a series of operations

Finally: Camera parameters

- Projection equation – the cumulative effect of all parameters:

$$\underset{(3\text{x}4)}{\mathbf{M}} = \begin{bmatrix} f & s & x'_c \\ 0 & af & y'_c \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{R}_{3x3} & \mathbf{0}_{3x1} \\ \mathbf{0}_{1x3} & 1 \end{bmatrix} \begin{bmatrix} \mathbf{I}_{3x3} & \mathbf{T}_{3x1} \\ \mathbf{0}_{1x3} & 1 \end{bmatrix}$$

intrinsics projection rotation translation

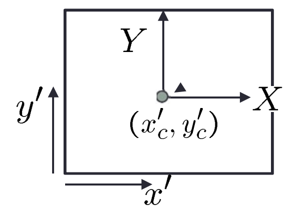
DoFs: 5+0+3+3 = 11

Calibrating cameras

Finally: Camera parameters

- Projection equation – the cumulative effect of all parameters:

$$\mathbf{x} = \begin{bmatrix} sx \\ sy \\ s \end{bmatrix} = \begin{bmatrix} * & * & * & * \\ * & * & * & * \\ * & * & * & * \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \mathbf{M}\,\mathbf{X}$$



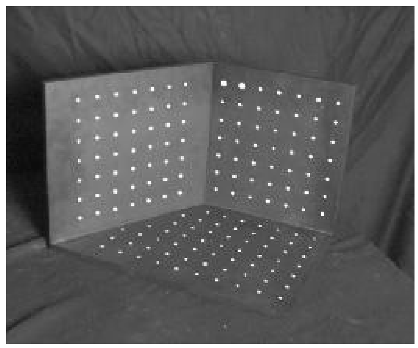
Calibration

- How to determine M?

Calibration using known points

Place a known object in the scene

- identify correspondence between image and scene
- compute mapping from scene to image



Resectioning

Estimating the camera matrix from known 3D points

Projective Camera Matrix:

$$p = K\begin{bmatrix} R & t \end{bmatrix} P = MP$$

$$\begin{bmatrix} w*u \\ w*v \\ w \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Direct linear calibration - homogeneous

$$\begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} \cong \begin{bmatrix} w * u_i \\ w * v_i \\ w \end{bmatrix} = \begin{bmatrix} m_{00} & m_{01} & m_{02} & m_{03} \\ m_{10} & m_{11} & m_{12} & m_{13} \\ m_{20} & m_{21} & m_{22} & m_{23} \end{bmatrix} \begin{bmatrix} X_i \\ Y_i \\ Z_i \\ 1 \end{bmatrix}$$

One pair of equations for each point

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

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

Direct linear calibration - homogeneous

One pair of equations for each point

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

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

$$u_i (m_{20} X_i + m_{21} Y_i + m_{22} Z_i + m_{23}) = m_{00} X_i + m_{01} Y_i + m_{02} Z_i + m_{03}$$

$$v_i (m_{20} X_i + m_{21} Y_i + m_{22} Z_i + m_{23}) = m_{10} X_i + m_{11} Y_i + m_{12} Z_i + m_{13}$$

Direct linear calibration - homogeneous

$$u_i(m_{20}X_i + m_{21}Y_i + m_{22}Z_i + m_{23}) = m_{00}X_i + m_{01}Y_i + m_{02}Z_i + m_{03}$$

$$v_i(m_{20}X_i + m_{21}Y_i + m_{22}Z_i + m_{23}) = m_{10}X_i + m_{11}Y_i + m_{12}Z_i + m_{13}$$

One pair of equations for each point

$$\begin{bmatrix} X_i & Y_i & Z_i & 1 & 0 & 0 & 0 & 0 & -u_iX_i & -u_iY_i & -u_iZ_i & -u_i \\ 0 & 0 & 0 & 0 & X_i & Y_i & Z_i & 1 & -v_iX_i & -v_iY_i & -v_iZ_i & -v_i \end{bmatrix} \begin{bmatrix} m_{00} \\ m_{10} \\ m_{02} \\ m_{03} \\ m_{10} \\ m_{11} \\ m_{12} \\ m_{13} \\ m_{20} \\ m_{21} \\ m_{22} \\ m_{23} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Direct linear calibration - homogeneous

- This is a homogenous set of equations.
- When over constrained, defines a least squares problem – minimize $\|\mathbf{A}\,\mathbf{m}\|$
- Since m is only defined up to scale, solve for unit vector m*
  - Solution: m* = eigenvector of A^TA with smallest eigenvalue
  - Works with 6 or more points

Direct linear calibration - homogeneous

$$\begin{bmatrix} X_1 & Y_1 & Z_1 & 1 & 0 & 0 & 0 & 0 & -u_1X_1 & -u_1Y_1 & -u_1Z_1 & -u_1 \\ 0 & 0 & 0 & 0 & X_1 & Y_1 & Z_1 & 1 & -v_1X_1 & -v_1Y_1 & -v_1Z_1 & -v_1 \\ & & & & & & \vdots & & & & & \\ X_n & Y_n & Z_n & 1 & 0 & 0 & 0 & 0 & -u_nX_n & -u_nY_n & -u_nZ_n & -u_n \\ 0 & 0 & 0 & 0 & X_n & Y_n & Z_n & 1 & -v_nX_n & -v_nY_n & -v_nZ_n & -v_n \end{bmatrix} \begin{bmatrix} m_{00} \\ m_{10} \\ m_{02} \\ m_{03} \\ m_{10} \\ m_{11} \\ m_{12} \\ m_{13} \\ m_{20} \\ m_{21} \\ m_{22} \\ m_{23} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}$$

A $\quad$ **m** $\quad$ **0**

2n × 12 $\quad$ **12** $\quad$ **2n**

The SVD (singular value decomposition) trick…

- Find the **m** that minimizes $\|A\mathbf{m}\|$ subject to $\|\mathbf{m}\|=1$.
- Let $A = UDV^T$ (singular value decomposition, D diagonal, U and V orthogonal)
- Therefor minimizing $\|UDV^T\mathbf{m}\|$
- But, $\|UDV^T\mathbf{m}\| = \|DV^T\mathbf{m}\|$ and $\|\mathbf{m}\|=\|V^T\mathbf{m}\|$
- Thus minimize $\|DV^T\mathbf{m}\|$ subject to $\|V^T\mathbf{m}\| = \mathbf{1}$

The SVD (singular value decomposition) trick…

- Thus minimize $||DV^T\mathbf{m}||$ subject to $||V^T\mathbf{m}|| = \mathbf{1}$
- Let $\mathbf{y} = V^T\mathbf{m}$ Now minimize $||D\mathbf{y}||$ subject to $||\mathbf{y}|| = 1$.
- But D is diagonal, with decreasing values. So $||D\mathbf{y}||$ minimum is when $\mathbf{y} = (0,0,0\ldots,0,1)^T$
- Since $\mathbf{y} = V^T\mathbf{m}$, $\mathbf{m} = V\mathbf{y}$ since V orthogonal
- Thus $\mathbf{m} = V\mathbf{y}$ is the last column in V.

The SVD (singular value decomposition) trick…

- Thus $\mathbf{m} = V\mathbf{y}$ is the last column in V.
- And, the singular values (D) of A are square roots of the eigenvalues of A^TA and the columns of V are the eigenvectors.
- Recap: Given Am=0, find the eigenvector of A^TA with smallest eigenvalue, that's m.

Direct linear calibration (transformation)

Advantages:

- Very simple to formulate and solve. Can be done, say, on a problem set
- These methods are referred to as "algebraic error" minimization.

Direct linear calibration (transformation)

Disadvantages:

- Doesn't directly tell you the camera parameters (more in a bit)
- Approximate: e.g. doesn't model radial distortion
- Hard to impose constraints (e.g., known focal length)
- *Mostly: Doesn't minimize the right error function*

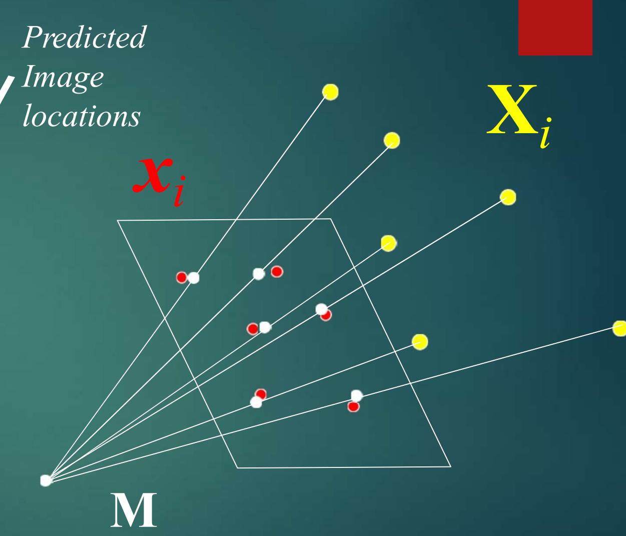
Direct linear calibration (transformation)

For these reasons, prefer nonlinear methods:

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

Geometric Error

$$\text{minimize } E = \sum_i d(x'_i, \hat{x}'_i)$$



$$\min_{\mathbf{M}} \sum_i d(x'_i, \mathbf{M}\mathbf{X}_i)$$

“Gold Standard” algorithm (Hartley and Zisserman)

Objective

Given n≥6 3D to 2D point correspondences $\{X_i \leftrightarrow x_i'\}$, determine the “Maximum Likelihood Estimation” of **M**

“Gold Standard” algorithm (Hartley and Zisserman)

Algorithm

(i) Linear solution:

(a) (Optional) Normalization: $\tilde{\mathbf{X}}_i = \mathbf{U}\mathbf{X}_i$ $\tilde{\mathbf{x}}_\mathbf{i} = \mathbf{T}\,\mathbf{x}_\mathbf{i}$

(b) Direct Linear Transformation minimization

(ii) Minimize geometric error: using the linear estimate as a starting point minimize the geometric error:

$$\min_{\mathbf{M}} \sum_i d(\tilde{\mathbf{x}}_i, \tilde{\mathbf{M}}\mathbf{X}_i)$$

“Gold Standard” algorithm (Hartley and Zisserman)

(iii) Denormalization: $\mathbf{M} = \mathbf{T}^{-1}\tilde{\mathbf{M}}\,\mathbf{U}$

Finding the 3D Camera Center from M

- M encodes all the parameters. So we should be able to find things like the camera center from M.
- Two ways: pure way and easy way

Finding the 3D Camera Center from M

- Slight change in notation. Let: $M = [Q \mid b]$
  M is(3x4) – *b* is last column of M
- The center C is the null-space camera of projection matrix. So if find C such that:

$$\mathbf{M}\,\mathbf{C} = \mathbf{0}$$

that will be the center. Really…

Finding the 3D Camera Center from M

- Proof: Let X be somewhere between any point P and C

$$\mathbf{X} = \lambda\,\mathbf{P} + (1-\lambda)\mathbf{C}$$

Finding the 3D Camera Center from M

- Proof: Let X be somewhere between any point P and C

$$\mathbf{X} = \lambda \mathbf{P} + (1 - \lambda)\mathbf{C}$$

- And the projection:

$$\mathbf{x} = \mathbf{M}\mathbf{X} = \lambda \mathbf{M}\mathbf{P} + (1 - \lambda)\mathbf{M}\mathbf{C}$$

Finding the 3D Camera Center from M

- Proof: Let X be somewhere between any point P and C

$$\mathbf{X} = \lambda \mathbf{P} + (1 - \lambda)\mathbf{C}$$

- And the projection:

$$\mathbf{x} = \mathbf{M}\mathbf{X} = \lambda \mathbf{M}\mathbf{P} + (1 - \lambda)\mathbf{M}\mathbf{C}$$

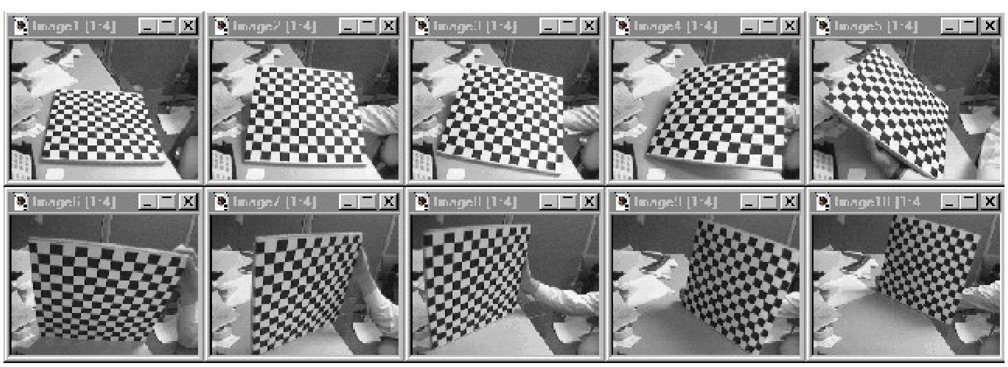
- For any P, all points on PC ray project on image of P, therefore MC must be zero. So the camera center has to be in the null space.

Finding the 3D Camera Center from M

- Now the easy way. A formula! If M =[Q|b] then:

$$\mathbf{C} = \begin{pmatrix} -\mathbf{Q}^{-1}\mathbf{b} \\ 1 \end{pmatrix}$$

Alternative: multi-plane calibration



Images courtesy Jean-Yves Bouguet, Intel Corp.

Alternative: multi-plane calibration

Advantages

- Only requires a plane
- Don't have to know positions/orientations
- Good code available online!
  - OpenCV library
  - 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/