# Computer Vision, Lecture 11

Professor Hager

http://www.cs.jhu.edu/~hager

Jason Corso

http://www.cs.jhu.edu/~jcorso

Notes from today's lecture are downloadable there.

# Outline for Today

- Camera Calibration
  - Motivation
  - Review of Projective Camera Model
  - Methods for Calibration
  - The Multiplane Method in Detail

# Motivation

- Computer Vision is …
  - A conglomerate of aspects from many fields.
  - A science whose main goal is to compute properties of the 3-D world from one or more digital images.
  - So far in the course has been largely spent in “image space.”
  - Move beyond the images and infer properties of the metric world.
  - To do so, we need to know the parameters of projection;  the camera parameters.

# Camera parameters

- Summary:
  - points expressed in external frame
  - points are converted to canonical camera coordinates
  - points are projected
  - points are converted to pixel units

$$\begin{pmatrix} U \\ V \\ W \end{pmatrix} = \begin{pmatrix} \text{Transformation} \\ \text{representing} \\ \text{intrinsic parameters} \end{pmatrix} \begin{pmatrix} \text{Transformation} \\ \text{representing} \\ \text{projection model} \end{pmatrix} \begin{pmatrix} \text{Transformation} \\ \text{representing} \\ \text{extrinsic parameters} \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ T \end{pmatrix}$$

point in pixel coords.

point in metric image coords.

point in cam. coords.

point in world coords.

# Extrinsic Parameters

Using the idea of homogeneous coordinates,
we can write:

$$p' = \begin{pmatrix} & R & & T \\ 0 & 0 & 0 & 1 \end{pmatrix} p$$

R and T both require 3 parameters. These correspond to the 6 extrinsic parameters needed for camera calibration

# Intrinsic Parameters

Intrinsic Parameters describe the conversion from unit focal length metric to pixel coordinates (and the reverse)

$$x_{mm} = - (x_{pix} – o_x)\, s_x \;\rightarrow\; -/s_x\, x_{mm} – o_x = -x_{pix}$$
$$y_{mm} = - (y_{pix} – o_y)\, s_y \;\rightarrow\; -/s_y\, y_{mm} – o_y = -y_{pix}$$

or

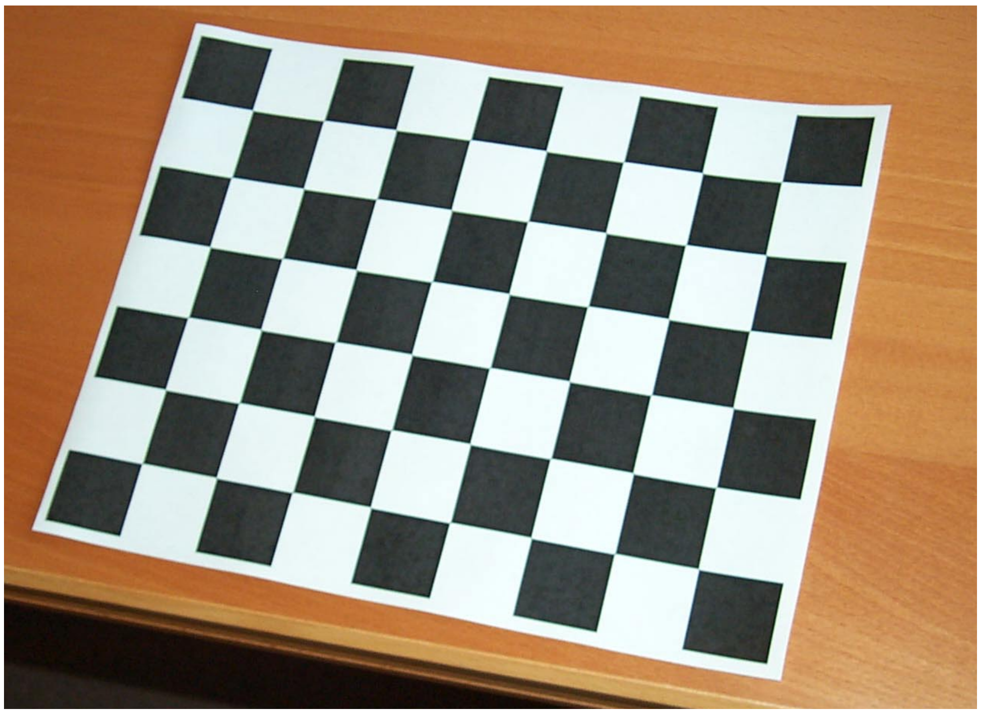
$$\begin{pmatrix} x \\ y \\ w \end{pmatrix}_{pix} = \begin{pmatrix} -1/s_x & 0 & o_x \\ 0 & -1/s_y & o_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ w \end{pmatrix}_{mm} = M_{\text{int}} p$$

It is common to combine scale and focal length together as the are both scaling factors; note projection is unitless in this case!

# Calibration – Problem Statement

**The problem:**

Compute the camera intrinsic (4 or 5) and extrinsic parameters (6) using only observed camera data.



# Types of Calibration

- Photogrammetric Calibration
- Self Calibration
- Multi-Plane Calibration
- Others not discussed:
  - Pure Rotation
  - Vanishing Points

# Photogrammetric Calibration

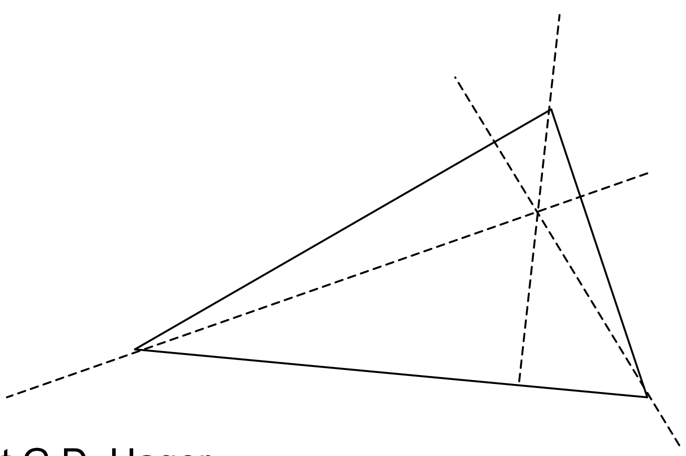
- Calibration is performed through imaging a pattern whose geometry in 3d is known with high precision.

- PRO: Calibration can be performed very efficiently
- CON: Expensive set-up apparatus is required; multiple orthogonal planes.

- Approach 1: Direct Parameter Calibration
- Approach 2: Projection Matrix Estimation

# The General Case

- Affine is “easy” because it is linear and unconstrained (note orthographic is harder because of constraints)
- Perspective case is also harder because it is both nonlinear and constrained
- Observation: optical center can be computed from the *orthocenter* of vanishing points of orthogonal sets of lines.



# Basic Equations

$$^{c}T_w = (T_x, T_y, T_z)'$$

$$^{c}R_w = (R_x, R_y, R_z)'$$

$$^{c}p = {}^{c}R_w{}^{w}p +^{c} T_w$$

$$u = -f\frac{R_x p + T_x}{R_z p + T_z}$$

$$v = -f\frac{R_y p + T_y}{R_z p + T_z}$$

# Basic Equations

$$u_{pix} = \frac{1}{s_x} u + o_x$$

$$v_{pix} = \frac{1}{s_y} v + o_y$$

$$\bar{u} = u_{pix} - o_x = -f_x \frac{R_x p + T_x}{R_z p + T_z}$$

$$\bar{v} = v_{pix} - o_y = -f_y \frac{R_y p + T_y}{R_z p + T_z}$$

# Basic Equations

$$\bar{u_i} f_y(R_y p_i + T_y) = \bar{v_i} f_x(R_x p_i + T_x)$$
$$\bar{u_i}(R_y p_i - T_y) - \bar{v_i}\alpha(R_x p_i + T_x) = 0$$

$r = \alpha R_x$ and $w = \alpha T_x$

$t = R_y$ and $s = T_y$

one of these for each point

$A_i = (u_i p_i, u_i, -v_i p_i, -v_i)$ and $A[t, s, w, r]' = 0$

# Basic Equations

$A_i = (u_i p_i, u_i, -v_i p_i, -v_i)$ and
$A[t, s, w, r]' = Am = 0$

Note that m is defined up a scale factor!

$A = UDV'$ and choose m as column of V corresponding to the smallest singular value

# Properties of SVD Again

- Recall the singular values of a matrix are related to its rank.
- Recall that Ax = 0 can have a nonzero x as solution only if A is singular.
- Finally, note that the matrix V of the SVD is an orthogonal basis for the domain of A; in particular the zero singular values are the basis vectors for the null space.

- Putting all this together, we see that A must have rank 7 (in this particular case) and thus x must be a vector in this subspace.

- Clearly, x is defined only up to scale.

# Basic Equations

$$A_i = (u_i p_i, u_i, -v_i p_i, -v_i) \text{ and}$$
$$A[t, s, w, r]' = Am = 0$$

$$\|t\| = |\gamma| \text{ gives scale factor for solution}$$
$$\|w\| = |\gamma|\alpha$$

We now know $R_x$ and $R_y$ up to a sign and gamma.
$R_z = R_x \times R_y$

We will probably use another SVD to orthogonalize this system (R = U D V'; set D to I and multiply).

# Last Details

- We still need to compute the correct sign.
  - note that the denominator of the original equations must be positive (points must be in front of the cameras)
  - Thus, the numerator and the projection must disagree in sign.
  - We know everything in numerator and we know the projection, hence we can determine the sign.

- We still need to compute $T_z$ and $f_x$
  - we can formulate this as a least squares problem on those two values using the first equation.

$$\bar{u} = -f_x \frac{R_x p + T_x}{R_z p + T_z} \rightarrow$$
$$\bar{u}(R_z p + T_z) = -f_x(R_x p + T_x)$$
$$f_x(R_x p + T_x) + \bar{u} T_z = -\bar{u} R_z p$$
$$A(f_x, T_z)' = b \rightarrow (f_x, T_z)' = (A'A)^{-1} A' b$$

# Direct Calibration: The Algorithm

1. Compute image center from orthocenter
2. Compute the A matrix (6.8)
3. Compute solution with SVD
4. Compute gamma and alpha
5. Compute R (and normalize)
6. Compute $f_x$ and and $T_z$

# Indirect Calibration: The Basic Idea

- We know that we can also just write
  - $\mathbf{u}_h = M\,\mathbf{p}_h$
  - x = (u/w) and y = (v/w), $\mathbf{u}_h = (u,v,1)'$
  - As before, we can multiply through (after plugging in for u,v, and w)
- Once again, we can write
  - A m = 0
- Once again, we use an SVD to compute m up to a scale factor.

# Getting The Camera Parameters

$$M = \begin{bmatrix} -f_x R_x + o_x R_z & -f_x T_x + o_x T_z \\ -f_y R_y + o_y R_z & -f_y T_y + o_y T_z \\ R_z & T_z \end{bmatrix}$$

We'll write

$$M = \begin{bmatrix} q_1 & \\ q_2 & q_4' \\ q_3 & \end{bmatrix}$$

# Getting The Camera Parameters

$$M = \begin{bmatrix} -f_x R_x + o_x R_z & -f_x T_x + o_x T_z \\ -f_y R_y + o_y R_z & -f_y T_y + o_y T_z \\ R_z & T_z \end{bmatrix}$$

We'll write

$$M = \begin{bmatrix} q_1 & \\ q_2 & q'_4 \\ q_3 & \end{bmatrix}$$

FIRST:

$|q_3|$ is scale up to sign; divide by this value

$M_{3,4}$ is $T_z$ up to sign, but $T_z$ must be positive; if not divide M by -1

$o_x = q_1 \,.\, q_3$
$o_y = q_2 \,.\, q_3$
$f_x = (q_1 \,.\, q_1 - o_x^2)^{1/2}$
$f_y = (q_2 \,.\, q_2 - o_y^2)^{1/2}$

THEN:

$R_y = (q_2 - o_y R_z)/f_y$
$R_x = R_y \times R_z$
$T_x = -(q_{4,1} - o_x T_z)/f_x$
$T_y = -(q_{4,2} - o_y T_z)/f_y$

Finally, use SVD to orthogonalize the rotation,

# Self-Calibration

- Calculate the intrinsic parameters solely from point correspondences from multiple images.
- Static scene and intrinsics are assumed.
- No expensive apparatus.
- Highly flexible but not well-established.
- Projective Geometry – image of the absolute conic.

# Multi-Plane Calibration

- Hybrid method: Photogrammetric and Self-Calibration.
- Uses a planar pattern imaged multiple times (inexpensive).
- Used widely in practice and there are many implementations.
- Based on a group of projective transformations called homographies.

- m be a 2d point [u v 1]' and M be a 3d point [x y z 1]'.

- Projection is

$$s\tilde{m} = A[R \quad T]\tilde{M}$$

# Planar Homographies

- First Fundamental Theorem of Projective Geometry:
  - There exists a unique homography that performs a change of basis between two projective spaces of the same dimension.

$$
\begin{aligned}
s[u \quad v \quad 1]^T &= A[r_1 \quad r_2 \quad r_3 \quad t][X \quad Y \quad Z \quad 1]^T \\
s[u \quad v \quad 1]^T &= A[r_1 \quad r_2 \quad r_3 \quad t][X \quad Y \quad 0 \quad 1]^T \\
s[u \quad v \quad 1]^T &= A[r_1 \quad r_2 \quad t][X \quad Y \quad 1]^T \\
s[u \quad v \quad 1]^T &= H[X \quad Y \quad 1]^T
\end{aligned}
$$

  - Projection Becomes

$$s\tilde{m} = H\tilde{M}$$

  - Notice that the homography is defined up to scale (s).

# Computing the Intrinsics

- We know that $[h_1 \quad h_2 \quad h_3] = sA[r_1 \quad r_2 \quad t]$
- From one homography, how many constraints on the intrinsic parameters can we obtain?
  - Extrinsics have 6 degrees of freedom.
  - The homography has 8 degrees of freedom.
  - Thus, we should be able to obtain 2 constraints per homography.
- Use the constraints on the rotation matrix columns…

# Computing Intrinsics

- Rotation Matrix is orthogonal….

$$r_i^T r_j = 0$$

$$r_i^T r_i = r_j^T r_j$$

- Write the homography in terms of its columns…

$$\begin{aligned} h_1 &= sAr_1 \\ h_2 &= sAr_2 \\ h_3 &= sAt \end{aligned}$$

# Computing Intrinsics

- Derive the two constraints:

$$
\begin{aligned}
h_1 &= sAr_1 \\
\frac{1}{s}A^{-1}h_1 &= r_1 \\
\frac{1}{s}A^{-1}h_2 &= r_2
\end{aligned}
$$

$$
\begin{aligned}
r_1^T r_2 &= 0 \\
h_1^T A^{-T} A^{-1} h_2 &= 0
\end{aligned}
$$

$$
\begin{aligned}
r_1^T r_1 &= r_2^T r_2 \\
h_1^T A^{-T} A^{-1} h_1 &= h_2^T A^{-T} A^{-1} h_2
\end{aligned}
$$

# Closed-Form Solution

$$\text{Let } B = A^{-T}A^{-1} = \begin{bmatrix} \frac{1}{\alpha^2} & -\frac{\gamma}{\alpha^2\beta} & \frac{v_0\gamma - u_0\beta}{\alpha^2\beta} \\ -\frac{\gamma}{\alpha^2\beta} & \frac{\gamma^2}{\alpha^2\beta^2} + \frac{1}{\beta^2} & -\frac{\gamma(v_0\gamma - u_0\beta))}{\alpha^2\beta^2} - \frac{v_0}{\beta^2} \\ \frac{v_0\gamma - u_0\beta}{\alpha^2\beta} & -\frac{\gamma(v_0\gamma - u_0\beta))}{\alpha^2\beta^2} - \frac{v_0}{\beta^2} & \frac{(v_0\gamma - u_0\beta)^2}{\alpha^2\beta^2} + \frac{v_0^2}{\beta^2} + 1 \end{bmatrix}$$

- Notice B is symmetric, 6 parameters can be written as a vector b.
- From the two constraints, we have $h_i^T B h_j = v_{ij}^T$

$$\begin{bmatrix} v_{ij}^T \\ (v_{11} - v_{22})^T \end{bmatrix} b = 0;$$

- Stack up n of these for n images and build a 2n*6 system.
- Solve with SVD (yet again).
- Extrinsics “fall-out” of the result easily.

# Non-linear Refinement

- Closed-form solution minimized algebraic distance.
- Since full-perspective is a non-linear model
  - Can include distortion parameters (radial, tangential)
  - Use maximum likelihood inference for our estimated parameters.

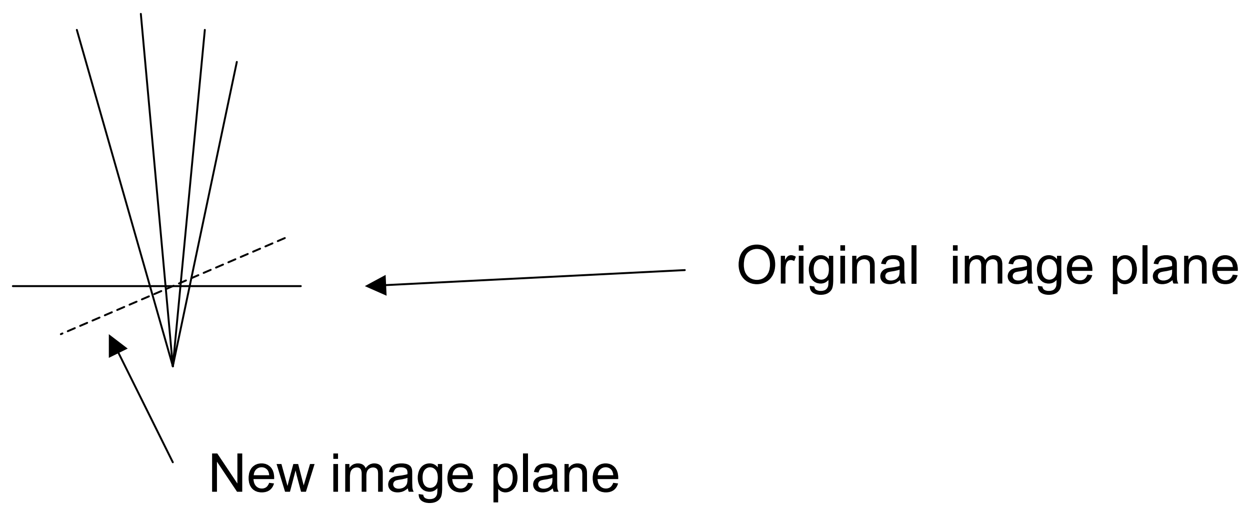
$$\sum_{i=1}^{n} \sum_{j=1}^{m} ||m_{ij} - \hat{m}(A, R_k, T_k, M_j)||^2$$

# Multi-Plane Approach In Action

- …if we can get matlab to work…

# An Example of Using Calibration

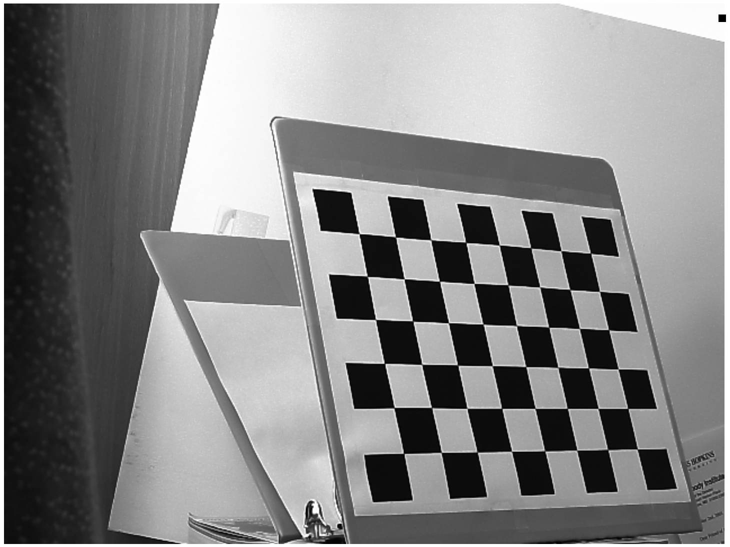
- Image rectification is the computation of an image as seen by a rotated camera
  - we'll show later that depth doesn't matter when rotating; for now we'll just use intuition



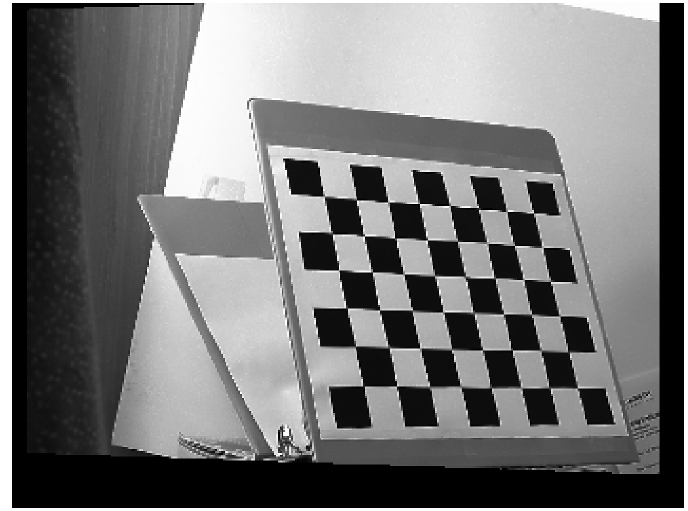
# Rectification: Basic Algorithm

- 1. Create a mesh of pixel coordinates for the rectified image
- 2. Turn the mesh into a list of homogeneous points
- 3. Project *backwards* through the intrinsic parameters to get unit focal length values
- 4. Rotate these values back to the current camera coordinate system.
- 5. Project them *forward* through the intrinsic parameters to get pixel coordinates again.
- 6. Sample at these points to populate the rectified image.

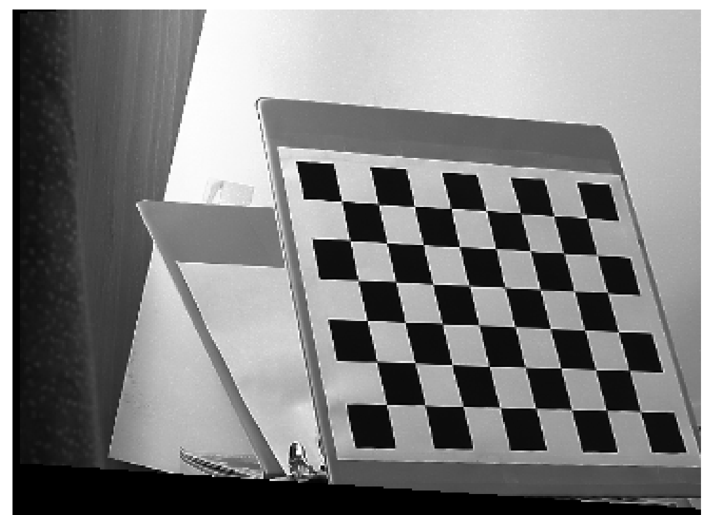
# Rectification Results

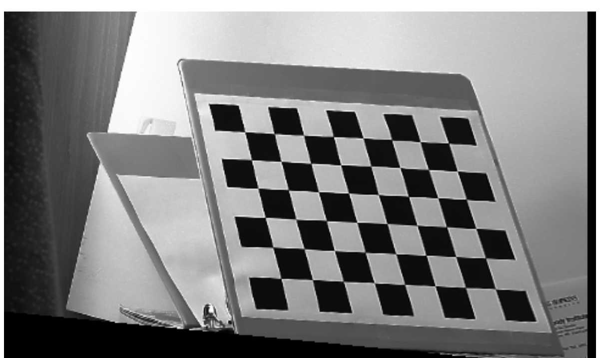


.2 rad



.4 rad





.6 rad

# Calibration Summary

- Two groups of parameters:
  - internal (intrinsic) and external (extrinsic)
- Many methods
  - direct and indirect, flexible/robust
- The form of the equations that arise here and the way they are solved is common in vision:
  - bilinear forms
  - $Ax = 0$
  - Orthogonality constraints in rotations
- Most modern systems use the method of multiple planes (matlab demo)
  - more difficult optimization over a large # of parameters
  - more convenient for the user