Computer Vision
Projective Geometry and Calibration

Professor Hager

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

Jason Corso

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

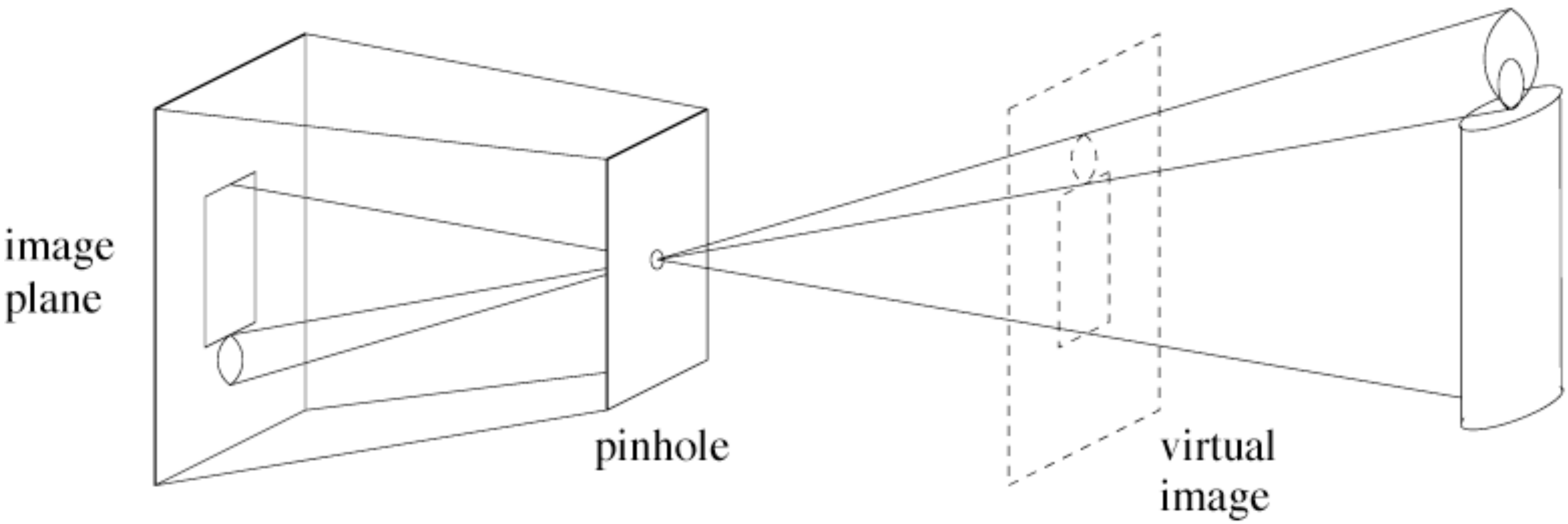
.

Topics

- Camera projection models
- Spatial transformations
- Projective coordinates
- Camera calibration

Pinhole cameras

- Abstract camera model - box with a small hole in it
- Pinhole cameras work in practice

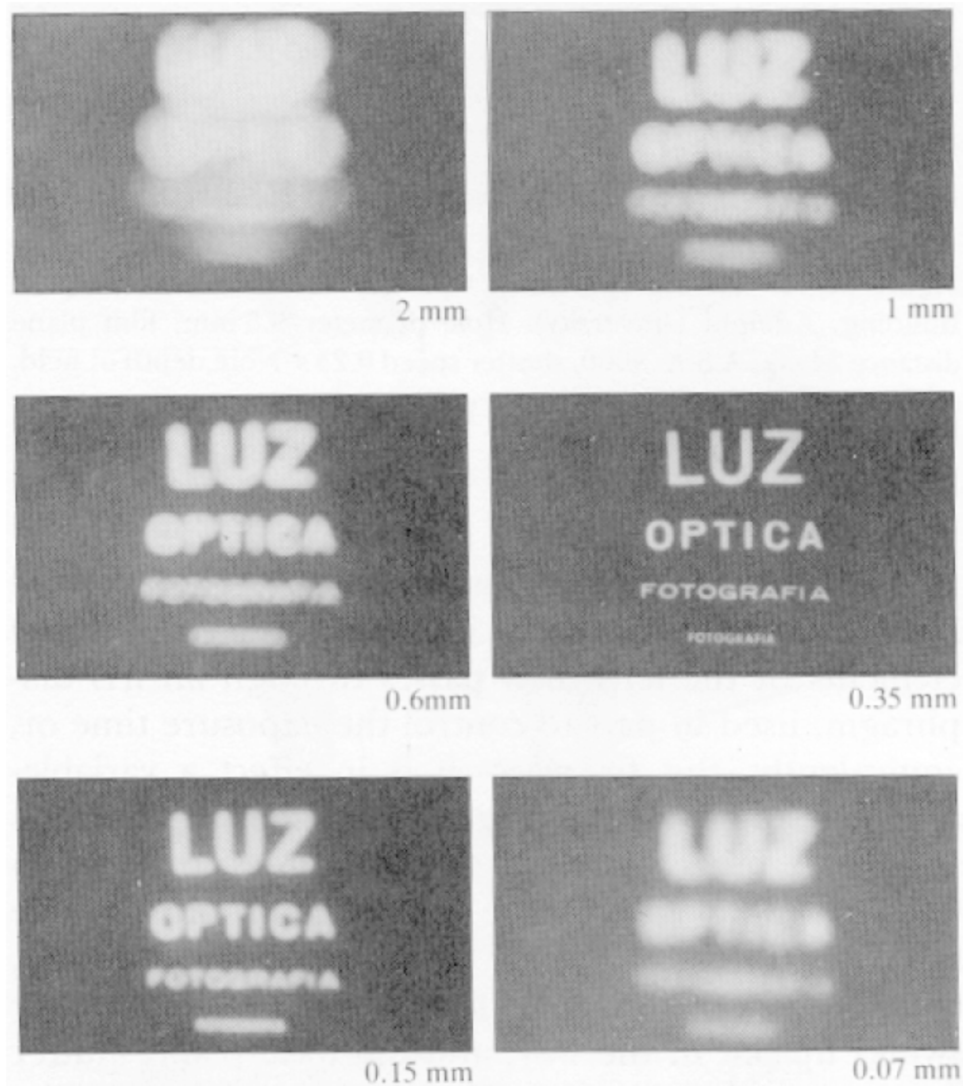


Real Pinhole Cameras

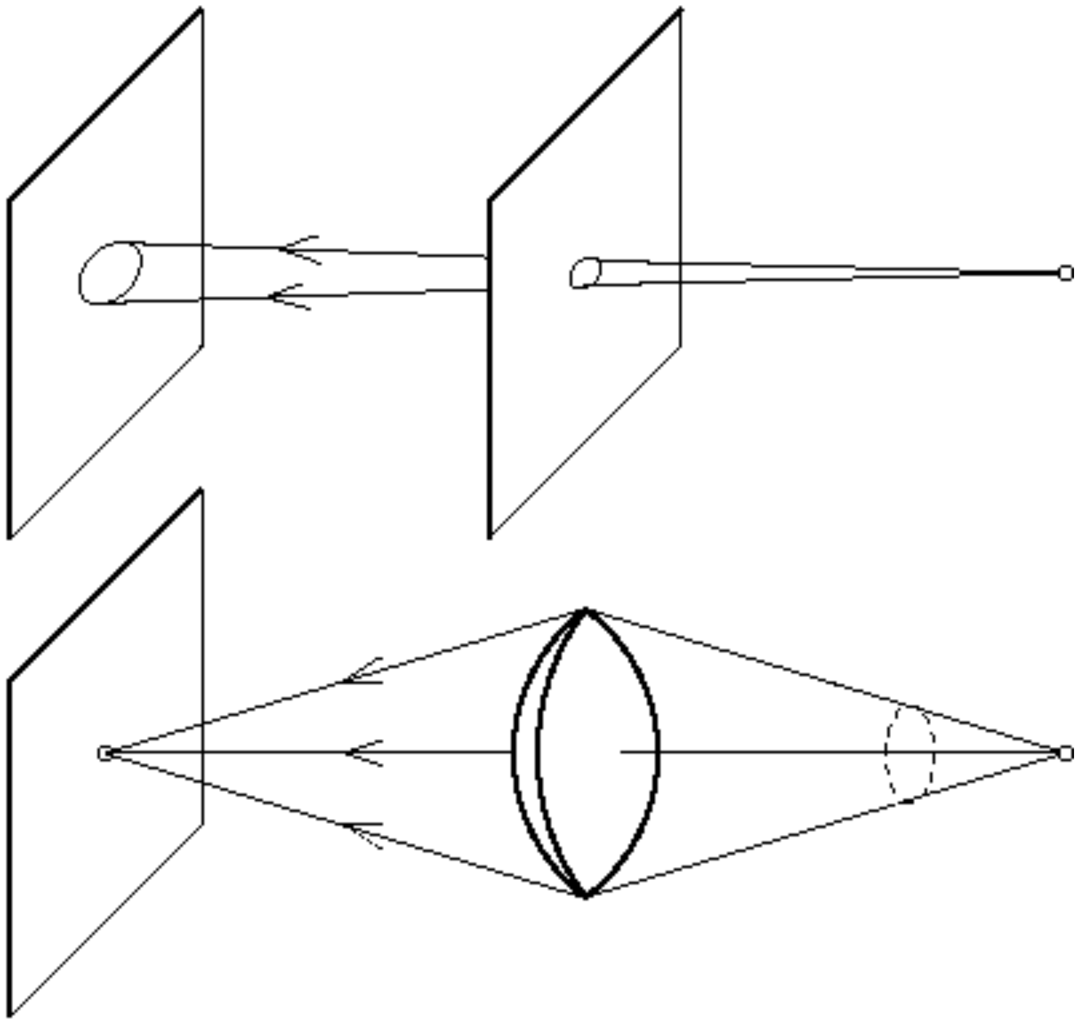
Pinhole too big -
many directions are averaged, blurring the image

Pinhole too small-
diffraction effects blur the image

Generally, pinhole cameras are *dark*, because a very small set of rays from a particular point hits the screen.

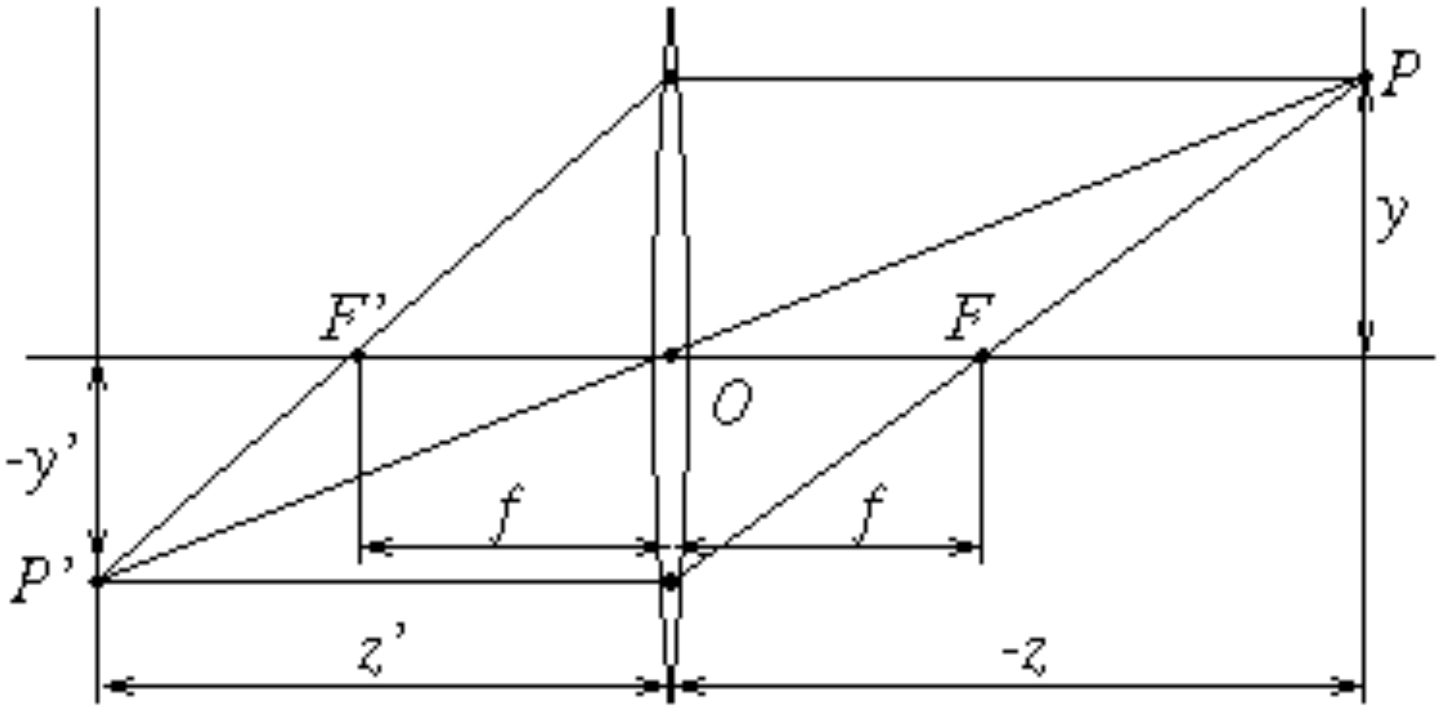


The reason for lenses



Lenses gather and focus light, allowing for brighter images.

The thin lens

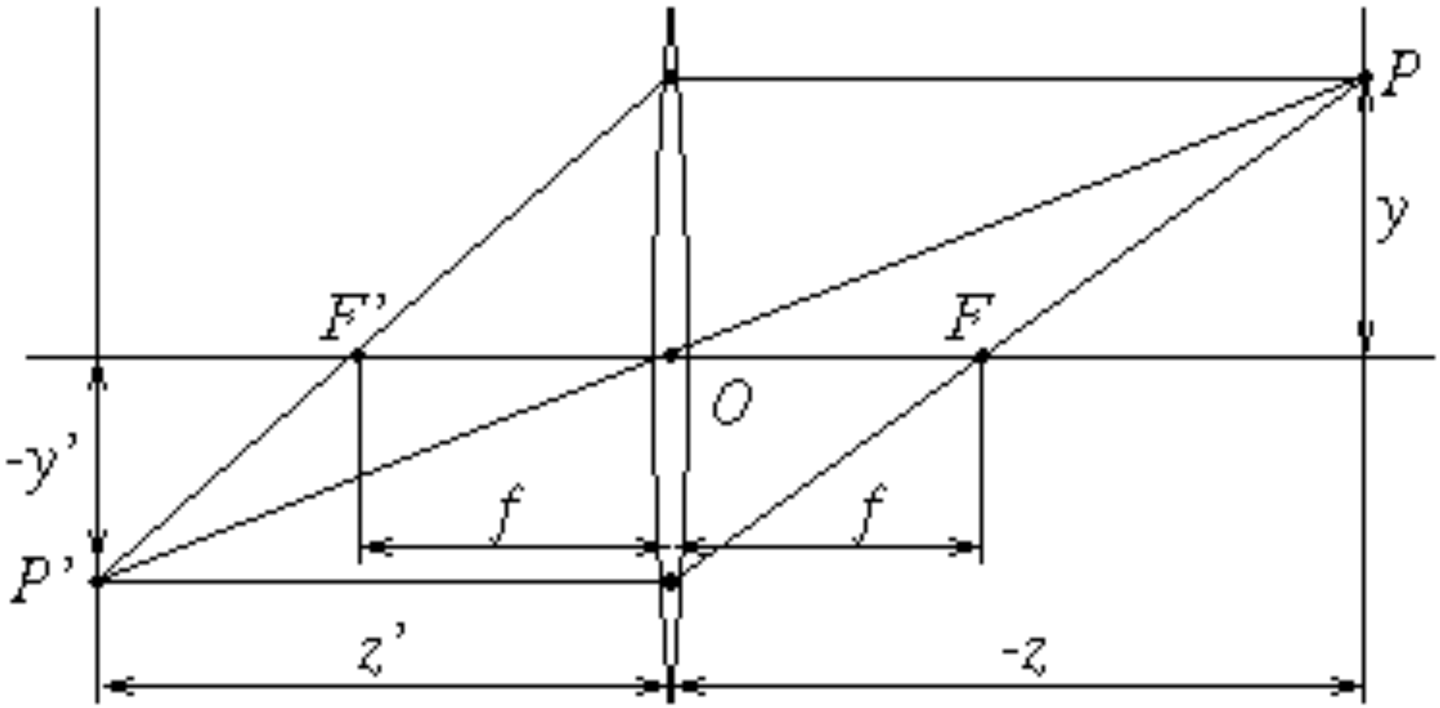


Thin Lens Properties:

1. A ray entering parallel to optical axis goes through the focal point.
2. A ray emerging from focal point is parallel to optical axis
3. A ray through the optical center is unaltered

$$\frac{1}{z'} - \frac{1}{z} = \frac{1}{f}$$

The thin lens



$$\frac{1}{z'} - \frac{1}{z} = \frac{1}{f}$$

Note that, if the image plane is very small and/or z >> z', then z' is approximately equal to f

Field of View

- The *effective diameter* of a lens (d) is the portion of a lens actually reachable by light rays.

- The effective diameter and the focal length determine the field of view:

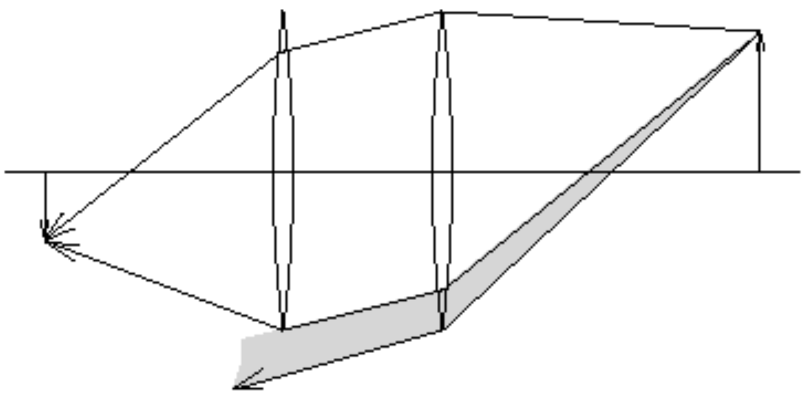
$$\tan w = d/(2f) \quad --> \quad FOV = 2 \ a\tan(d/(2f))$$

- Another fact is that in practice points at different distances are imaged, leading to so-called "circles of confusion" of size d/z | z'-z| where z is the nominal image plane and z' is the focusing distance given by the thin lens equation.

- The "depth of field" is the range of distances that produce acceptably focused images.
  - Depth of field varies inversely with focal length and lens diameter.

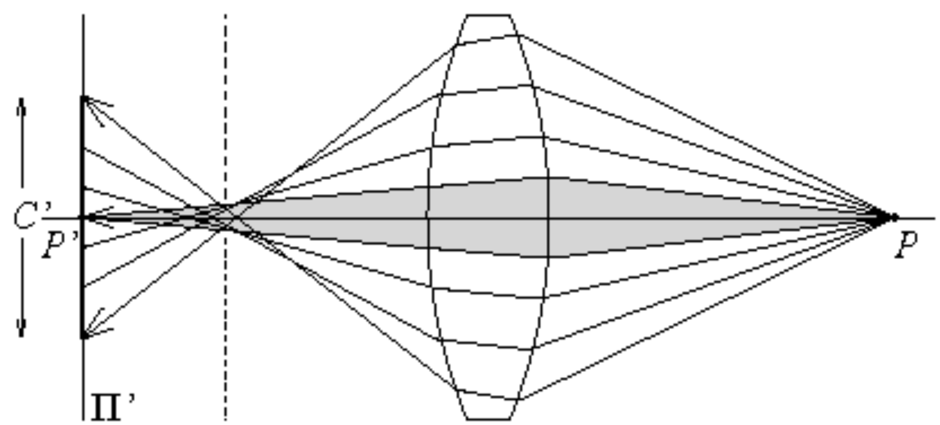
Lens Realities

Real lenses have a finite depth of field, and usually suffer from a variety of defects



vignetting

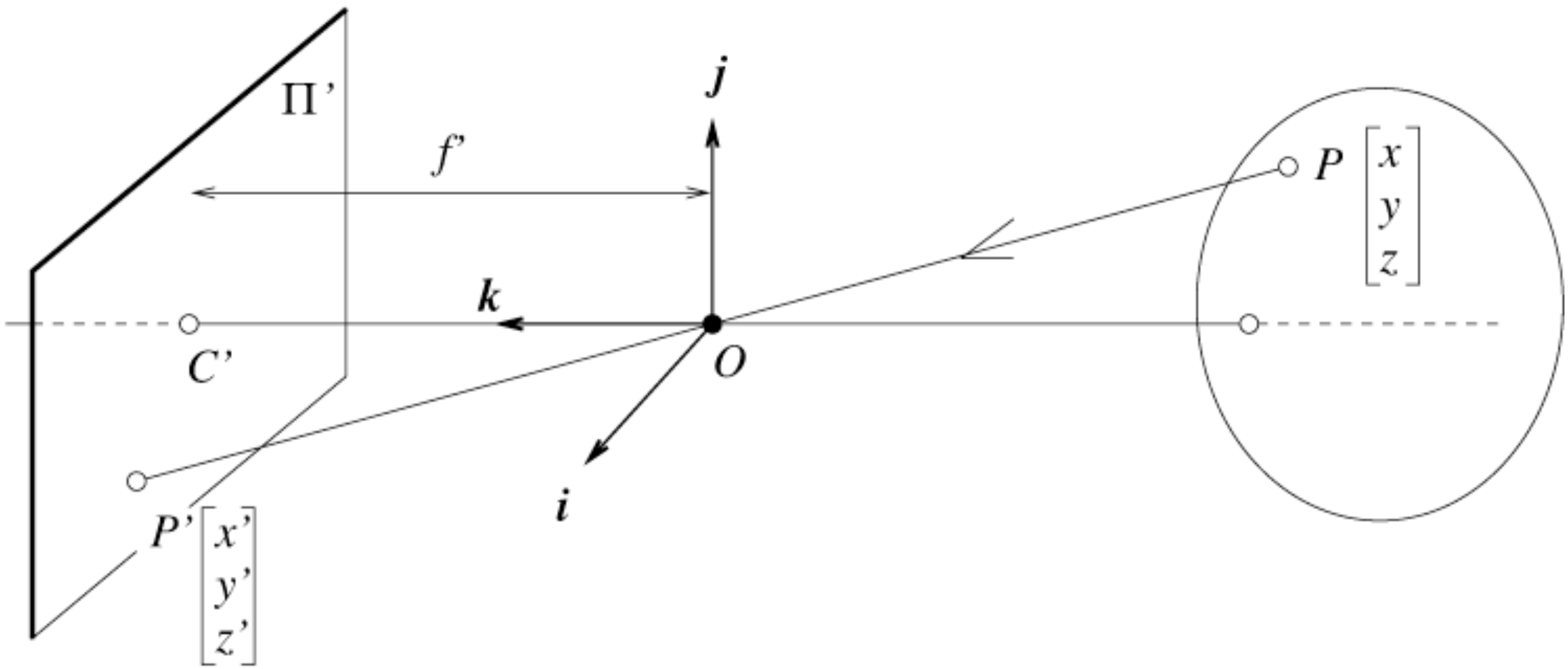
Spherical Aberration



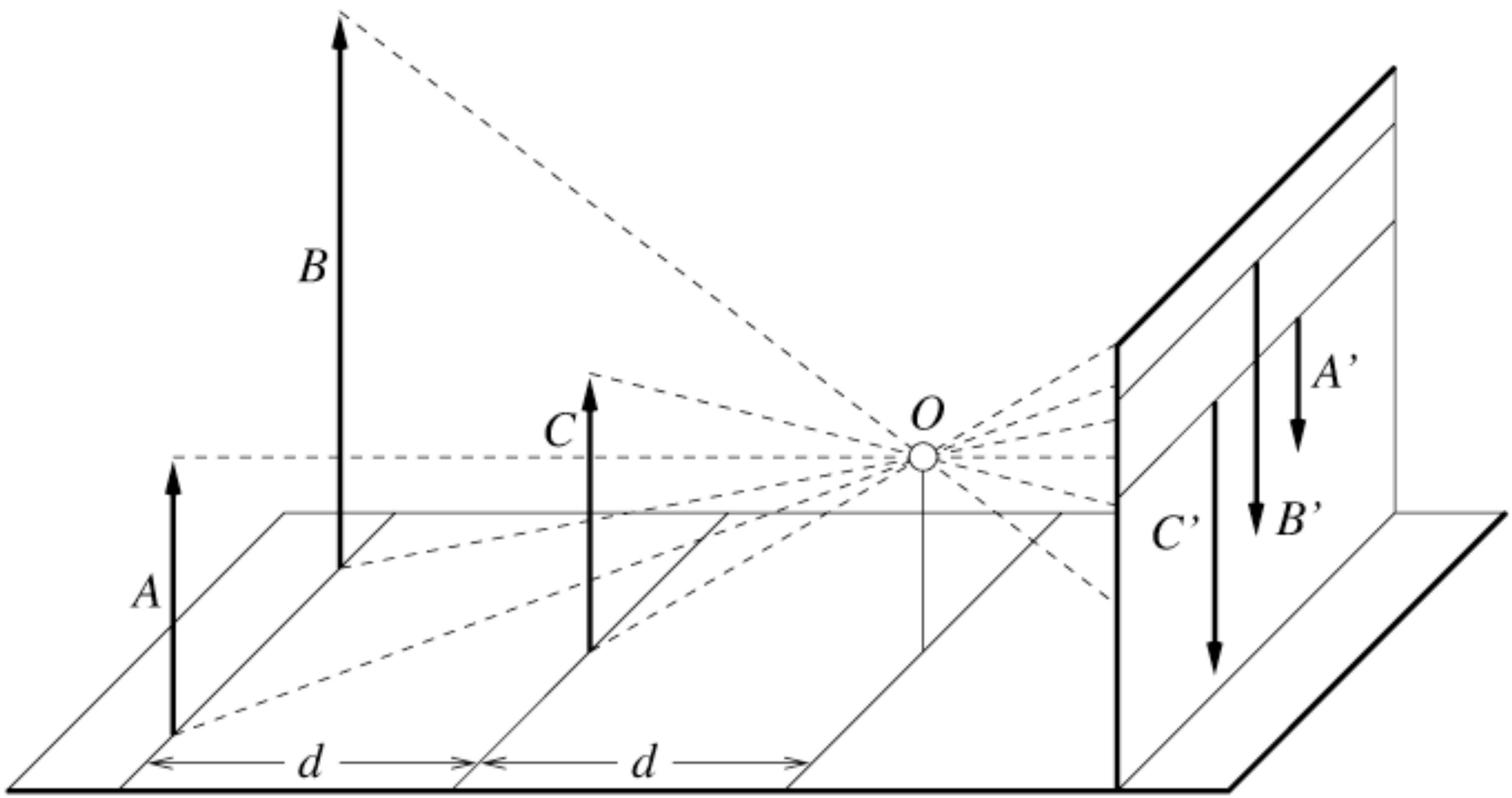
Perspective Projection

- Equating z’ and f
  - We have, by similar triangles, that (x, y, z) -> (-f x/z, -f y/z, -f)
  - Ignore the third coordinate, and flip the image around to get:

$$(x, y, z) \rightarrow (f\frac{x}{z}, f\frac{y}{z})$$

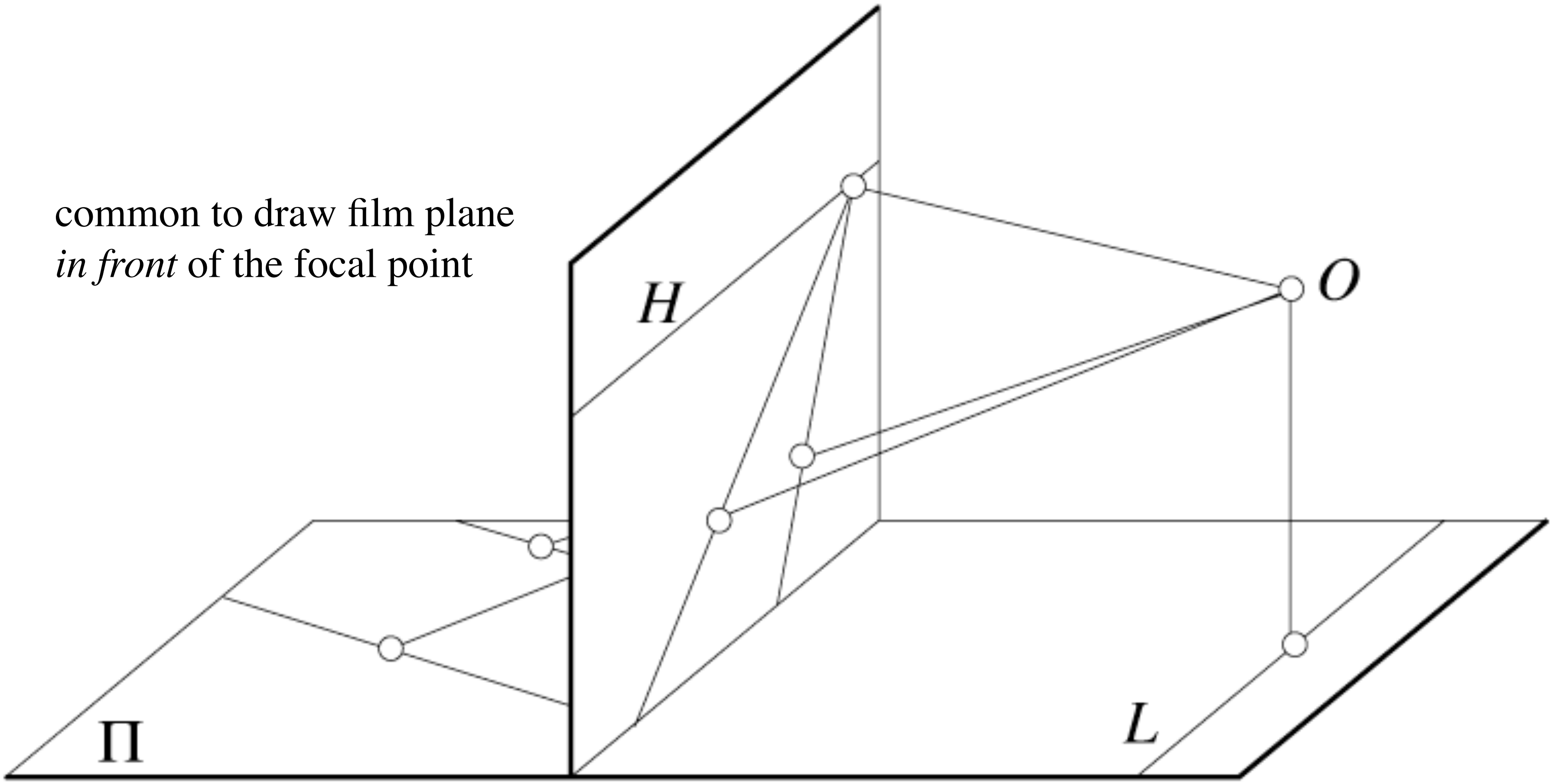


Distant objects are smaller



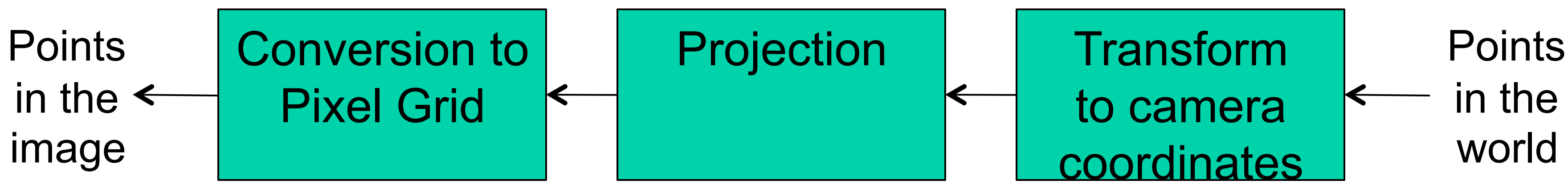
Parallel Lines Meet at a Point

common to draw film plane
in front of the focal point



A Good Exercise: Show this is the case!

The Projection "Chain"



Intrinsic Parameters

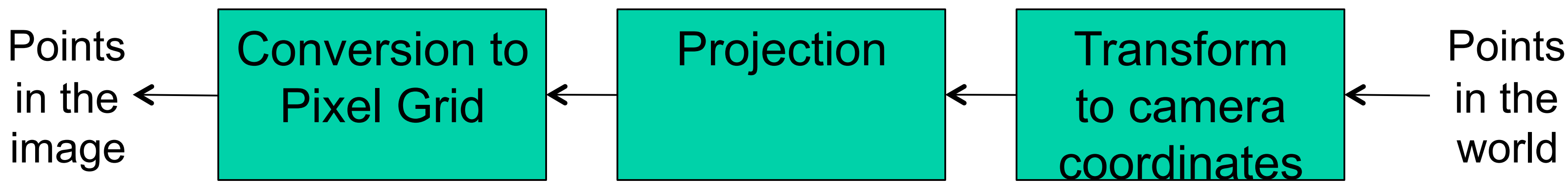
Intrinsic Parameters describe the conversion from unit focal length metric to pixel coordinates (and the reverse)

$$u_{mm} = f\ x/z$$
$$v_{mm} = f\ y/z$$

$$u_{mm} = (u_{pix} - o_x)\ s_u \ \text{-->}\ 1/s_u\ u_{mm} + o_u = u_{pix}$$
$$v_{mm} = (v_{pix} - o_y)\ s_v \ \text{-->}\ 1/s_v\ v_{mm} + o_v = v_{pix}$$

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

The Projection "Chain"



Projection Geometry: Standard Camera Coordinates

- By convention, we place the image in front of the optical center
  - typically we approximate by saying it lies one focal distance from the center
  - in reality this can't be true for a finite size chip!
- Optical axis is z axis pointing outward
- X axis is parallel to the scanlines (rows) pointing to the right!
- By the right hand rule, the Y axis must point downward
- Note this corresponds with indexing an image from the upper left to the lower right, where the X coordinate is the column index and the Y coordinate is the row index.

An Aside: Geometric Transforms

In general, a point in n-D space transforms rigidly by

newpoint = rotate(point) + translate(point)

In 2-D space, this can be written as a matrix equation:

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} Cos(\theta) & -Sin(\theta) \\ Sin(\theta) & Cos(\theta) \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} tx \\ ty \end{pmatrix}$$

In 3-D space (or n-D), this can generalized as a matrix equation:

$p' = R\,p + T$ or $p = R^t\,(p' - T)$

Transforming the world frame   Transforming the observer frame

Properties of Rotations

- In general, a rotation matrix satisfies two properties:
  - $R^t R = R R^t = I$
  - $\det(R) = 1$

- What does this make the inverse of a rotation?

- Note that this defines some properties of the component vectors of the matrix.

- A 3D rotation can be expressed in many ways:
  - as a composition of individual rotations $R_{3d}(\theta_1, \theta_2, \theta_3) = R_{2d}(\theta_1)\, R_{2d}(\theta_2)\, R_{2d}(\theta_3)$
  - as an angle-axis n, θ
    $R = I \cos(\theta) + (1-\cos(\theta))\,(n\, n^t) + \sin(\theta)\, sk(n)$

Homogeneous Transforms

Often, we want to *compose* transformations, but using separate translations and rotations makes that clumsy.

Instead, we embed points in a higher-dimensional space by appending a 1 to the end (now a 4d vector)

Now, using the idea of *homogeneous transforms*, 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

How Do We Combine Projection and Transformation?

- Step 0: Points are expressed in some coordinate system that is not the cameras (e.g. a model or a robot):
  - $p = (x, y, z ,1)$
- Step 1: Transform the points into camera coordinates
  - $q= (x', y', z', 1) = T\ p$
- Step 2: Project the points
  - $u = f\ x'/z'$; $v = f\ y'/z'$
- A linear step followed by a nonlinear step …

Basic Projective Concepts

- We have seen homogeneous coordinates already; projective geometry makes use of these types of coordinates, but generalizes them
- The vector p = (x,y,z,w)' is equivalent to the vector k p for nonzero k
  - note the vector p = 0 is disallowed from this representation
- The vector v = (x,y,z,0)' is termed a "point at infinity"; it corresponds to a direction
- We can embed real points into projective space, and always recover the "real" point by normalizing by the third coordinate provided it is not a point at infinity.
- A good model for P(n) embedded in R(n+1) is the set of all lines passing through the origin – each line corresponds to a point

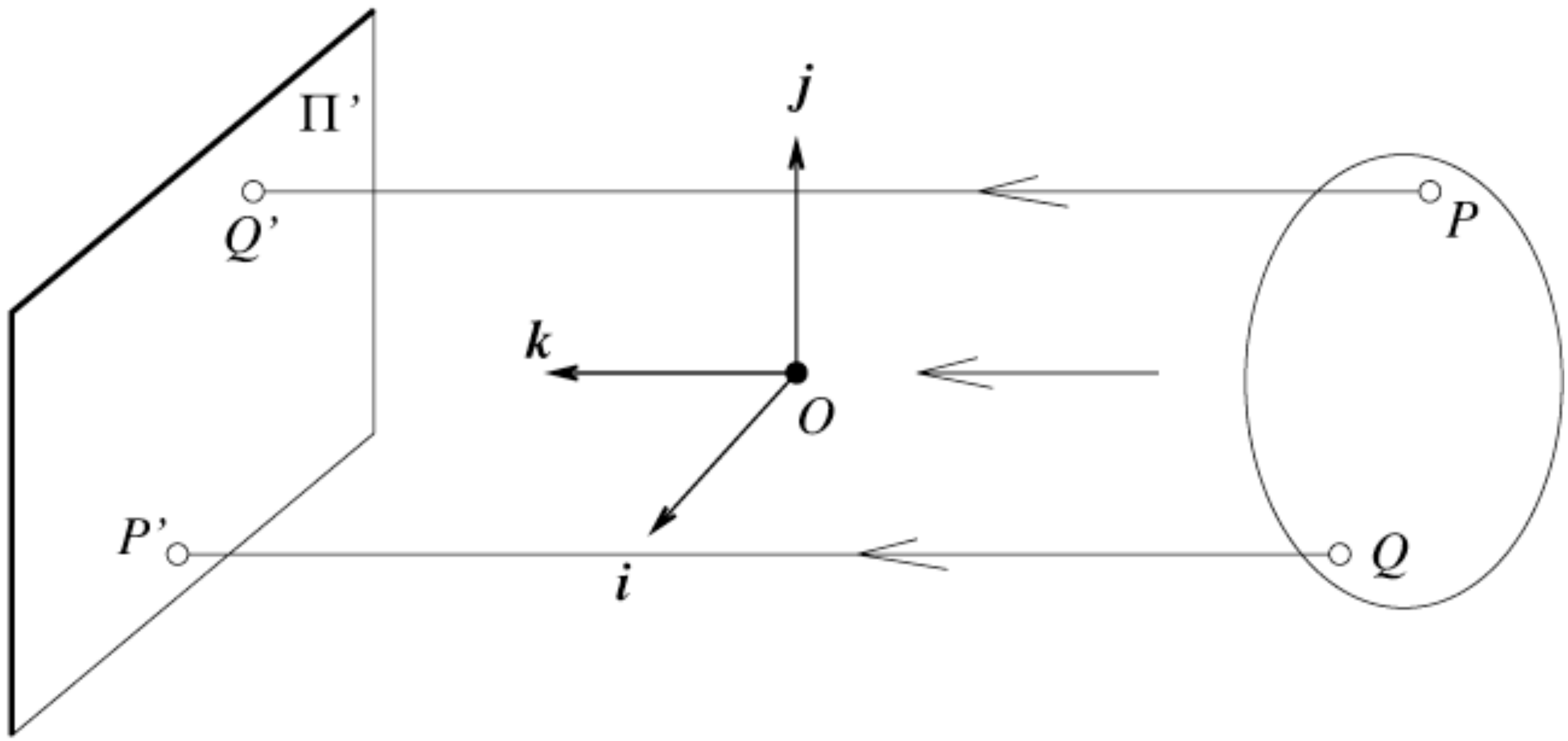
The Camera Matrix

- Homogenous coordinates for 3D
  - four coordinates for 3D point
  - equivalence relation (X,Y,Z,T) is the same as (k X, k Y, k Z,k T)
- Turn previous expression into HC's
  - HC's for 3D point are (X,Y,Z,T)
  - HC's for point in image are (U,V,W)

$$\begin{pmatrix} U \\ V \\ W \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1/f & 0 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ T \end{pmatrix}$$

$$(U,V,W) \rightarrow (\frac{U}{W},\frac{V}{W}) = (u,v)$$

Orthographic projection



Suppose I let f go to infinity; then

$$u = x$$

$$v = y$$

The model for orthographic projection

$$\begin{pmatrix} U \\ V \\ W \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ T \end{pmatrix}$$

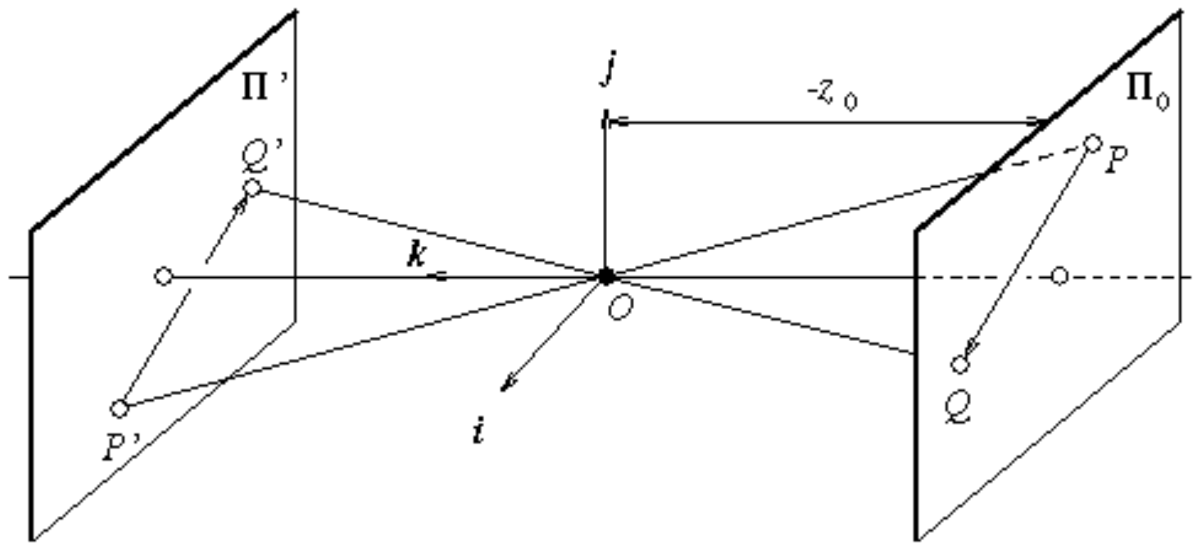
Scaled Orthography (Weak Perspective)

- Issue
  - perspective effects, but not over the scale of individual objects
  - collect points into a group at about the same depth, then divide each point by the depth of its group
  - Adv: easy
  - Disadv: wrong

$$u = sx$$

$$v = sy$$

$$s = f / Z^*$$



The Model for Scaled Orthography

$$\begin{pmatrix} U \\ V \\ W \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & Z^*/f \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ T \end{pmatrix}$$

Affine Projection

Pick an arbitrary point p0 = $(x_0,y_0,z_0)^t$

Recall u = f x/z and v = f y /z

Linearize about p → u = $f/z_0\,(x-x_0) - f\,x_0/z_0^2\,(z-z_0)$

u = [f/z_0 0 $f\,x_0/z_0^2$ $f\,x_0/z_0 - f\,x_0/z_0$]

$$\begin{pmatrix} U \\ V \\ W \end{pmatrix} = \begin{pmatrix} 1 & 0 & x_0/z_0 & 0 \\ 0 & 1 & y_0/z_0 & 0 \\ 0 & 0 & 0 & z_0/f \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ T \end{pmatrix}$$

Intrinsic Parameters

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

$x_{mm} = -(x_{pix} - o_x)\, s_x \;\; \text{-->} \;\; -1/s_x\, x_{mm} + o_x = x_{pix}$

$y_{mm} = -(y_{pix} - o_y)\, s_y \;\; \text{-->} \;\; -1/s_y\, y_{mm} + o_y = y_{pix}$

or

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

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

Putting it All Together

Now, using the idea of *homogeneous transforms*, 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

Then we can write

q = Π p’ for some projection model Π

Finally, we can write

u = K q for intrinsic parameters K

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.

Some Things to Point Out

General projection model: $q = \lambda K \Pi H p = \lambda Mp$

If M is perspective, 3x4 values-> how many independent parameters?

If M is affine, how many independent parameters?

If M is orthographic, how many independent parameters?

What Is Preserved?

- We are used to rigid body transformations (homogeneous transforms) which preserve:
  - Distance
  - Angle
  - Area

- With Affine, we lose distance and length; what is preserved is
  - Ratios of distances/lengths
  - Area

- With Projective, we also loose preservation of area; what is preserved is:
  - Ratios of ratios of distances (the so-call cross-ratio)
  - Intersection/coincidence of lines/points

Model Stratification

	Euclidean	Similarity	Affine	Projective
Transforms				
rotation	x	x	x	x
translation	x	x	x	x
uniform scaling		x	x	x
nonuniform scaling			x	x
shear			x	x
perspective				x
composition of proj.				x
Invariants				
length	x			
angle	x	x		
ratios	x	x		
parallelism	x	x	x	
incidence/cross rat.	x	x	x	x

Projection and 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 H is defined up to scale (s).

Estimating A Homography

- Here is what looks like a reasonable recipe for computing homographies:
  - Planar pts $(x_1;y_1;1, x_2; y_2; 1, \ldots. x_n;y_n;1) = X$
  - Corresponding pts $(u_1;v_1;1,u_2;v_2;1,\ldots u_n;v_n;1) = U$
  - $U = H X$
  - $U X' (X X')^{-1} = H$

- This will not work: the problem is really $\lambda_i U_i = H X_i$
- So we'll have to work a little harder ...
  - hint: work out algebraically eliminating λ_i

Properties of SVD

- SVD: $A = U D V^t$
  - U and V are unitary (unit columns mutually orthogonal, but not rotations!)
  - D is diagonal
  - In general if A = m x n, U is mxm, D is m x n, V is n x n
  - If m > n, there will be many zeros in D; can make U mxn and D nxn
  - Eigenvalues are squares of elements of D; eigenvectors are columns of V
- 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
  - We can show eigenvectors with null eigenvalue are the only nontrivial solution here
- 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 n-1 (in this particular case) and thus x must be a vector in this subspace.
- Clearly, x is defined only up to scale.

Basic Projective Concepts

- In 2D space
  - points:
    - Cartesian point (x,y)
    - Projective pt (x,y,w) with convention that w is a scale factor
  - lines
    - a point p on the line and a unit *normal* n s.t. n • (p' - p) = 0
    - multiplying through, also n • p' - d = 0, where d is distance of closest pt to origin.
    - any vector n • q = 0 where q is a projective pt
      - note, for two lines, the intersection is two equations in 3 unknowns up to scale --- i.e. a one-dimensional subspace, or a *point*
    - note that points and lines are *dual* --- I can think of n or q as the normal (resp. point) e.g.
      - two points determine a line
      - two lines determine a point

Basic Projective Concepts

- In 3D space
  - points:
    - Cartesian point (x,y,z)
    - Projective pt (x,y,z,w) with convention that w is a scale factor
  - lines:
    - a point p on the line and unit vector v for direction
      - for minimal parameterization, p is closest point to origin
    - Alternative, a line is the intersection of two planes (see below)
  - planes
    - a point p on the plane and a unit *normal* n s.t. n • (p' - p) = 0
    - multiplying through, also n • p' - d = 0, where d is distance of closest pt to origin.
    - any vector n • q = 0 where q is a projective pt
      - note, for two planes, the intersection is two equations in 4 unknowns up to scale --- i.e. a one-dimensional subspace, or a *line*
    - Note that planes and points are *dual* --- in the above, I can equally think of n or q as the normal (resp. point).

Properties of SVD

- Recall the Singular Value Decomposition of a matrix M (m by n) is
  $M = U\ D\ V^t$ where
  - U is m by n and has unit orthogonal columns (unitary)
  - D is n by n and has the *singular values* on the diagonal
  - V is n by n and has unit orthogonal columns (unitary)

- Interpretation:
  - V is the “input space”
  - D provides a “gain” for each input direction
  - U is a projection into the “output space”

- As a result:
  - The null space of M corresponds to the zero singular values in D
  - In most cases (e.g. Matlab) the singular values are sorted largest to smallest, so the null space is the right-most columns of V
  - Matlab functions are
    - [u,d,v] = SVD(m)
    - [u,d,v] = SVD(m,0) (“economy svd”)
      - if m > n, only the first n singular values are computed and D is n by n
      - useful when solving overconstrained systems of equations

Some Projective Concepts

- The vector p = (x,y,z,w)' is equivalent to the vector k p for nonzero k
  - note the vector p = 0 is disallowed from this representation
- The vector v = (x,y,z,0)' is termed a "point at infinity"; it corresponds to a direction
- In P^2,
  - given two points p_1 and p_2, $l = p_1 \times p_2$ is the line containing them
  - given two lines, l_1, and l_2, $p = l_1 \times l_2$ is point of intersection
  - A point p lies on a line l if $p \bullet l = 0$ (note this is a consequence of the triple product rule)
  - l = (0,0,1) is the "line at infinity"
  - it follows that, for any point p at infinity, $l \bullet p = 0$, which implies that points at infinity lie on the line at infinity.

Some Projective Concepts

- The vector p = (x,y,z,w)' is equivalent to the vector k p for nonzero k
  - note the vector p = 0 is disallowed from this representation
- The vector v = (x,y,z,0)' is termed a "point at infinity"; it corresponds to a direction
- In P^3,
  - A point p lies on a plane l if p • l = 0 (note this is a consequence of the triple product rule; there is an equivalent expression in determinants)
  - l = (0,0,0,1) is the "plane at infinity"
  - it follows that, for any point p at infinity, l• p = 0, which implies that points at infinity lie on the line at infinity.

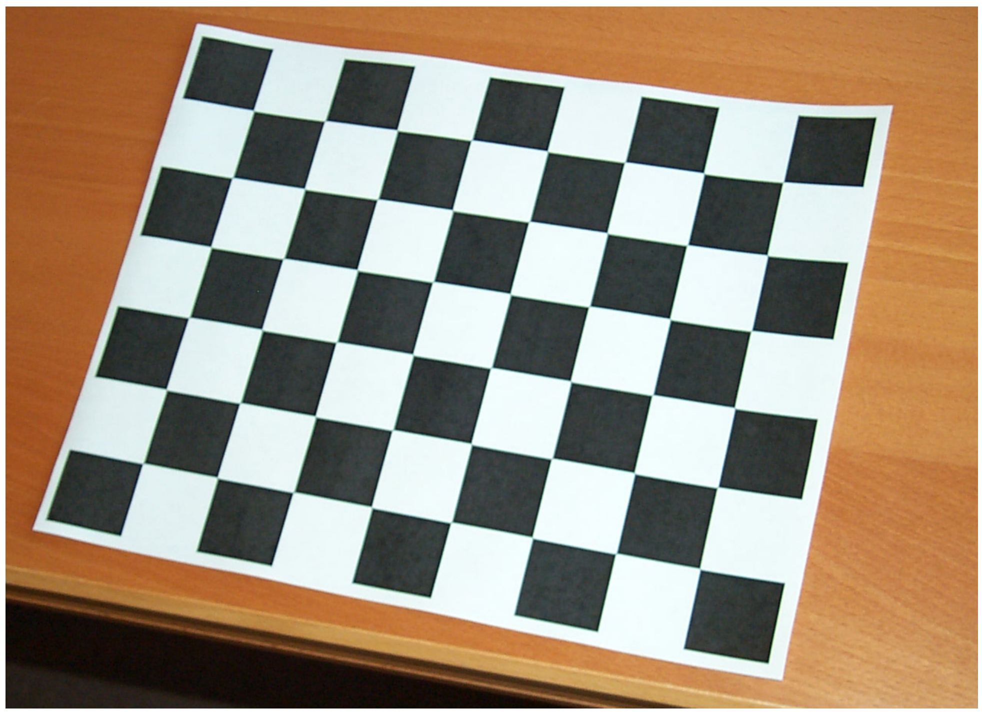
Parallel lines meet

- First, show how lines project to images.
- Second, consider lines that have the same direction (are parallel) but are not parallel to the imaging plane
- Third, consider the degenerate case of lines parallel to the image plane
  - (by convention, the vanishing point is at infinity!)

A Good Exercise: Show this is the case!

Camera Calibration: Problem Statement

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



Camera Calibration

Calibration = the computation of the camera intrinsic and extrinsic parameters

- General strategy:
  - view calibration object
  - identify image points
  - obtain camera matrix by minimizing error
  - obtain intrinsic parameters from camera matrix
- Most modern systems employ the multi-plane method
  - avoids knowing absolute coordinates of calibration poitns
- Error minimization:
  - Linear least squares
    - easy problem numerically
    - solution can be rather bad
  - Minimize image distance
    - more difficult numerical problem
    - solution usually rather good, but can be hard to find
      - start with linear least squares
  - Numerical scaling is an issue

A Quick Aside: Least Squares

- Total least squares: $a\, x_i + b\, y_i = z_i$
  - Leads to $\min_{a,b} \sum_i (a\, x_i + b\, y_i - z_i)^2$
  - Equivalent to $\min_d \sum_i \| d^t u_i - z_i \|^2$
  - Equivalent to $\min_U \| \mathbf{U}\, d - \mathbf{z} \|^2$
  - Solution is given by taking derivatives yielding $U^t U d = U^t z$
    - This implies that $U^t U$ must be full rank!

- Suppose I have $f(p, x_i) = z_i$
  - $\min_p \sum_i \| f(p, x_i) - z_i \|^2$
  - Many solutions, however notice if we Taylor series expand f, we get
  - $\min_{\Delta p} \sum_i \| f(p_0, x_i) + J_f(p_0,x_i)\, \Delta p - z_i \|^2$
  - Define "innovation" $b_i = f(p_0,x_i) - z_i$
  - Define $J = [J_f(p_0,x_1); J_f(p_0,x_2) \ldots; J_f(p_0,x_n)]$
  - Solve $\min_{M\, p} \| J\, \Delta p - b \|^2$
    - This is now the same as the previous problem

- Finally, suppose we have $A\, u_i = b_i$ and we are looking for A
  - $\min_A \sum_i \| A\, u_i - b_i \|^2$
  - Equivalent to $\min_A \| A\, U - B \|^2_F$
  - Equivalent to solving A U = B in the least squares sense
  - Solution is to write $A\, U\, U^t = B\, U^t \Longrightarrow A = (U\, U^t)^{-1} B$

Homogeneous systems

- Suppose I have a_i u = 0
  - $\min_A$ || A u $||^2$ with A = [a_1;a_2; … a_n]
  - Clearly u = 0 is a minimum, so add constraint ||u|| = 1
  - Langrangian: $\min_{A,l}$ || A u $||^2$ + l ||u -1$||^2$
  - Take gradient w.r.t u: A^tA u + l u = 0
  - Result --- u is eigenvector e of A^tA
  - A e is eigenvalue; plugging into original optimization, choose eigenvector with minimum eigenvalue
  - Done efficiently using SVD

Calibration: A Warmup

- Suppose we want to calibrate the affine camera and we know $u_i = A\ p_i + d$ for many pairs i
- m is mean of u's and q is mean of p's; note m = A q + d
- $U = [u_1 - m, u_2 - m, \ldots u_n - m]$ and $P = [p_1 - q, p_2 - q, \ldots p_n - q]$
- $U = A\ P \rightarrow U\ P'\ (P\ P')^{-1} = A$
- d is now mean of $u_i - A\ p_i$

Types of Calibration

- Photogrammetric Calibration
- Self Calibration
- **Multi-Plane Calibration**

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.

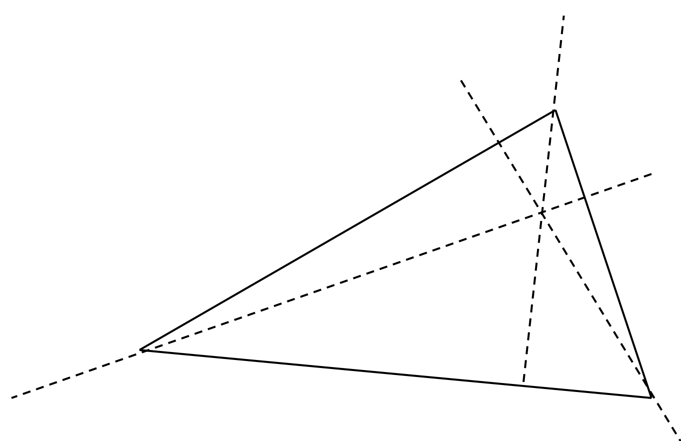
- Approach 1: Direct Parameter Calibration
- Approach 2: Projection Matrix Estimation

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

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

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

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

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

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

$$\bar{u} = (u_{pix} - c_u) = s_x u$$

$$\bar{v} = (v_{pix} - c_v) = s_y u$$

Known values

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

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

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

Basic Equations

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

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

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

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.

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

Direct Calibration: The Algorithm

1. Compute image center from orthocenter
2. Compute the A matrix
3. Compute solution with SVD
4. Compute gamma and alpha
5. Compute R (and normalize)
6. Compute f_x and and T_z
7. If necessary, solve a nonlinear regression to get distortion parameters

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.

- We can again use algebra to recover the actual camera parameters from the matrix.

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.

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

- Matlab implementation:
  - http://www.vision.caltech.edu/bouguetj/calib_doc/index.html

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

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

Review: Projection Model

$$\begin{pmatrix} U \\ V \\ W \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1/f & 0 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ T \end{pmatrix} \Rightarrow \begin{pmatrix} U \\ V \\ W \end{pmatrix} = \begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ T \end{pmatrix}$$

$$\begin{pmatrix} U \\ V \\ W \end{pmatrix}_{pix} = \begin{pmatrix} s_u & 0 & o_u \\ 0 & s_v & o_v \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} U \\ V \\ W \end{pmatrix}_{mm} \Rightarrow$$

$$\begin{pmatrix} U \\ V \\ W \end{pmatrix}_{pix} = \begin{pmatrix} fs_u & 0 & o_u \\ 0 & fs_v & o_v \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} U \\ V \\ W \end{pmatrix}_{mm} = \begin{pmatrix} \alpha & \gamma & u_0 \\ 0 & \beta & v_0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} U \\ V \\ W \end{pmatrix} Ap$$

Result

- Recall A = [r_1 r_2 t]
- Given matching point pairs, we can compute an homography H
- 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 supplies 8 values.
  - 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 $h_1^T B h_2 = v_{12} b$

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

- Stack up n of these for n images and build a 2n*6 system.
- Solve with SVD.
- Intrinsic parameters “fall-out” of the result easily using algebra

Computing Extrinsics

$$s[u \quad v \quad 1]^T = A[r_1 \quad r_2 \quad t][X \quad Y \quad 1]^T$$
$$s[u \quad v \quad 1]^T = H[X \quad Y \quad 1]^T$$

First, compute H' = A^{-1} H

Note that first two columns of H' should be rotation

use this to determine scaling factor

Orthogonalize using SVD to get rotation

Pull out translation as scaled last column

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…

Camera parameters

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

$$\underbrace{\begin{pmatrix} U \\ V \\ W \end{pmatrix}}_{\text{point in pixel coords.}} = \begin{pmatrix} \text{Transformation} \\ \text{representing} \\ \text{intrinsic parameters} \end{pmatrix} \underbrace{\begin{pmatrix} \text{Transformation} \\ \text{representing} \\ \text{projection model} \end{pmatrix} \underbrace{\begin{pmatrix} \text{Transformation} \\ \text{representing} \\ \text{extrinsic parameters} \end{pmatrix} \underbrace{\begin{pmatrix} X \\ Y \\ Z \\ T \end{pmatrix}}_{\text{point in world coords.}}}_{\text{point in cam. coords.}}}_{\text{point in metric image coords.}}$$

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

Lens Distortion

- In general, lens introduce minor irregularities into images, typically radial distortions:

$$x = x_d(1 + k_1 r^2 + k_2 r^4)$$
$$y = y_d(1 + k_1 r^2 + k_2 r^4)$$
$$r^2 = x_d^2 + y_d^2$$

- The values k_1 and k_2 are additional parameters that must be estimated in order to have a model for the camera system.

  – The complete model is then:

    $q = \mathrm{distort}(k_1, k_2, K(s_x, s_y, o_x, o_y) * (\Pi(p; R, t)))$

The Final Iterations

- Recall scalar linear least squares:
  - $\min_x \text{sum}_i (y - a\,x)^2$
- To go to multiple dimensions
  - $\min_x \text{sum}_i \|y - A\,x\|^2$
- What if we have a nonlinear problem?
  - $\min_x \text{sum}_i \|y - F(x)\|^2$

Multi-Camera Calibration

- Note that I might observe a target simultaneously in two or more cameras
  - For any given pair, I can solve for the transformation between them
  - For multiple pairs, I can optimize the relative location

- This is the natural lead-in to computational stereo …

Resampling Using Homographies

- Pick a rotation matrix R from old to new image

- Consider all points in the image *you want to compute*; then
  - construct pixel coordinates x = (u,v,1)
  - K maps unit focal length metric coordinates to pixel (normalized camera)
  - $x' = K\, R^t\, K^{-1}\, x \quad \rightarrow x' = H\, x$

pixel coordinates

pixel coordinates

rotation

pixel to euclidean coords

euclidean to pixel coords

- Sample a point x' in the original image for each point x in the new.

Rectification

- The goal of rectification is to turn a verged camera system into a non-verged system.
  - Let us assume we know $p_r = {}^rR_l\, p_l + T$
    - how would we get this out of camera calibration?

- Observation:
  - consider a coordinate system where the stereo baseline defines the x axis, z is any axis orthogonal to it, and y is z cross x.
    - x = T/||T||
    - y = ([0,0,1] £ x)/|| [0,0,1] £ x ||
    - z = x cross y

  - Note that both the left and right camera can now be rotated to be parallel to this frame
    - ${}^lR_u = [x\ y\ z]$ is the rotation from unverged to verged frame for left camera
    - ${}^rR_u = {}^rR_l\, {}^lR_u$

Bilinear Interpolation

- A minor detail --- new value $x' = (u',v',1)$ may not be integer
- let $u' = i + f_u$ and $v' = j+f_v$
- New image value $b = (1-f_u)((1-f_v)I(j,i) + f_v\, I(j+1,i)) +$
  $f_u((1-f_v)I(j,i+1) + f_v\, I(j+1,i+1))$

Estimating Changes of Coordinates

- Affine Model: y = A x + d
- How do we solve this given matching pairs of y's and x's?
- How many points do we need for a unique solution?
- Note this can be written sum_i || y – Ax –d $||^2$
  - It can also be written using the Frobenius norm
- Answer:

Estimating Changes of Coordinates

- Consider a 2D Euclidean model
  - $y = R x + t$
- How many points to solve for this transformation?
- Consider a 3D Euclidean model
  - $y = R x + t$
- How many points to solve for this transformation?
- Here, SVD will come to the rescue!
  - Compute barycentric y' and x'
  - Compute $M = Y' X'^T$
  - $M = U D V^T$
  - $R = V U^T$

An Approximation: The Affine Camera

- Choose a nominal point x_0, y_0, z_0 and describe projection relative to that point

- $u = f\,[x_0/z_0 + (x-x_0)/z_0 - x_0/z^2_0\,(z - z_0)\,] = f\,(a_1\,x + a_2\,z + d_1)$
- $v = f\,[y_0/z_0 + (y - y_0)/z_0 - y_0/z^2_0\,(z - z_0) = f\,(a_3\,y + a_4\,z + d_2)$

- gathering up

alternatively:

- $A = [a_1\ 0\ a_2;\ 0\ a_3\ a_4]$
- $d = [d_1;\ d_2]$

$$\begin{pmatrix} U \\ V \\ W \end{pmatrix} = \begin{pmatrix} a_1 & 0 & a_2 & d_1 \\ 0 & a_3 & a_4 & d_2 \\ 0 & 0 & 0 & 1/f \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ T \end{pmatrix}$$

- $\mathbf{u} = A\,\mathbf{P} + d$

- add external transform

alternatively:

- u = A (RP + T) + d --> u = A* Q   where A* is 2x4 rank 2 Matrix and Q is homogeneous version of P

Summary: Other Models

- The *orthographic* and *scaled orthographic* cameras (also called *weak perspective*)
  - simply ignore z
  - differ in the scaling from x/y to u/v coordinates
  - preserve Euclidean structure to a great degree

- The *affine camera* is a generalization of orthographic models.
  - u = A p + d
  - A is 2 x 3 and d is 2x1
  - This can be derived from scaled orthography or by linearizing perspective about a point not on the optical axis

- The *projective camera* is a generalization of the perspective camera.
  - u' = M p
  - M is 3x4 nonsingular defined up to a scale factor
  - This just a generalization (by one parameter) from "real" model

- Both have the advantage of being linear models on real and projective spaces, respectively.

Model Stratification

	Euclidean	Similarity	Affine	Projective
Transforms				
rotation	x	x	x	x
translation	x	x	x	x
uniform scaling		x	x	x
nonuniform scaling			x	x
shear			x	x
perspective				x
composition of proj.				x
Invariants				
length	x			
angle	x	x		
ratios	x	x		
parallelism	x	x	x	
incidence/cross rat.	x	x	x	x

Why Projective (or Affine or ...)

- Recall in Euclidean space, we can define a change of coordinates by choosing a new origin and three orthogonal unit vectors that are the new coordinate axes
  - The class of all such transformation is SE(3) which forms a group
  - One rendering is the class of all homogeneous transformations
  - This *does not* model what happens when things are imaged (why?)
- If we allow a change in scale, we arrive at similarity transforms, also a group
  - This sometimes can model what happens in imaging (when?)
- If we allow the 3x3 rotation to be an arbitrary member of GL(3) we arrive at affine transformations (yet another group!)
  - This also sometimes is a good model of imaging
  - The basis is now defined by three arbitrary, non-parallel vectors

- The process of perspective projection **does not** form a group
  - that is, a picture of a picture cannot in general be described as a perspective projection

- Projective systems include perspectivities as a special case and **do** form a group
  - We now require 4 basis vectors (three axes plus an additional independent vector)
  - A model for linear transformations (also called collineations or homographies) on P^n is GL(n+1) which is, of course, a group

Basic Equations

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

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

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

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

A Quick Aside: Least Squares

- Familiar territory is $y_i = a\, x_i + b$ ==> $\min_{a,b} \sum_i (a\, x_i + b - y_i)$

- Total least squares: $a\, x_i + b\, y_i = z_i$
  - Leads to $\min_{a,b} \sum_i (a\, x_i + b\, y_i - z_i)^2$
  - Equivalent to $\min_d \sum_i \| d^t u_i - z_i \|^2$
  - Equivalent to $\min_U \| \mathbf{U}\, d - \mathbf{z} \|^2$
  - Solution is given by taking derivatives yielding $U^t U d = U^t z$
    - This implies that $U^t U$ must be full rank!

- Another way to think of this:
  - let $d = (a, b, 1)$
  - Let $w_i = (x_i, y_i, z_i)$, W the matrix with rows w_i
  - Then we can write
    - $\min_d \| Wd \|^2_F$ with $\|d\| = 1$ or $W^t W d = 0$ with $\|d\| = 1$
    - Another way to state this is solve $W d = 0$ in least squares sense with $\|d\| = 1$
    - How do we solve this?