Issues in Project #1

What is the goal of camera calibration?

Estimate the intrinsic parameters, which do not depend on camera position

1. $f_u = f k_u$
2. $f_v = f k_v$
3. u_0
4. v_0

The projection matrix P is not the answer!

Issues in Project #1

In addition to projection error, you should check if the calibration result does make sense

Meanings and reasonable range for intrinsic parameters

$f_u = f * k_u \quad f_v = f * k_v$

k_u and k_v are spatial sampling rate (pixels/mm)

u_0 and v_0 are the x-y coordinates of the principal point, which is almost at the image center

Issues in Project #1

Grading is primarily based on your report.

Your report should be complete and contain all the required information:

- a brief introduction on the addressed problem,
- a succinct description on the methods you implemented with the major steps, e.g., the equations to calculate those intrinsic parameters
- **the experimental results and analysis,**
- conclusion, and
- reference.

Fix the Issue in Project #1

- Revise and resubmit your project #1 by **11:59pm, Wednesday, March 8**. Project 1 will be graded based on your revised submission.

- If you have done project #1 and do not need/want a revision, you will get **20 extra points** for Project 1. Project 1 will be graded based on your original submission.

Midterm Exam

In-person exam

2:20pm – 3:35pm, Monday, March 13th

A calculator is allowed

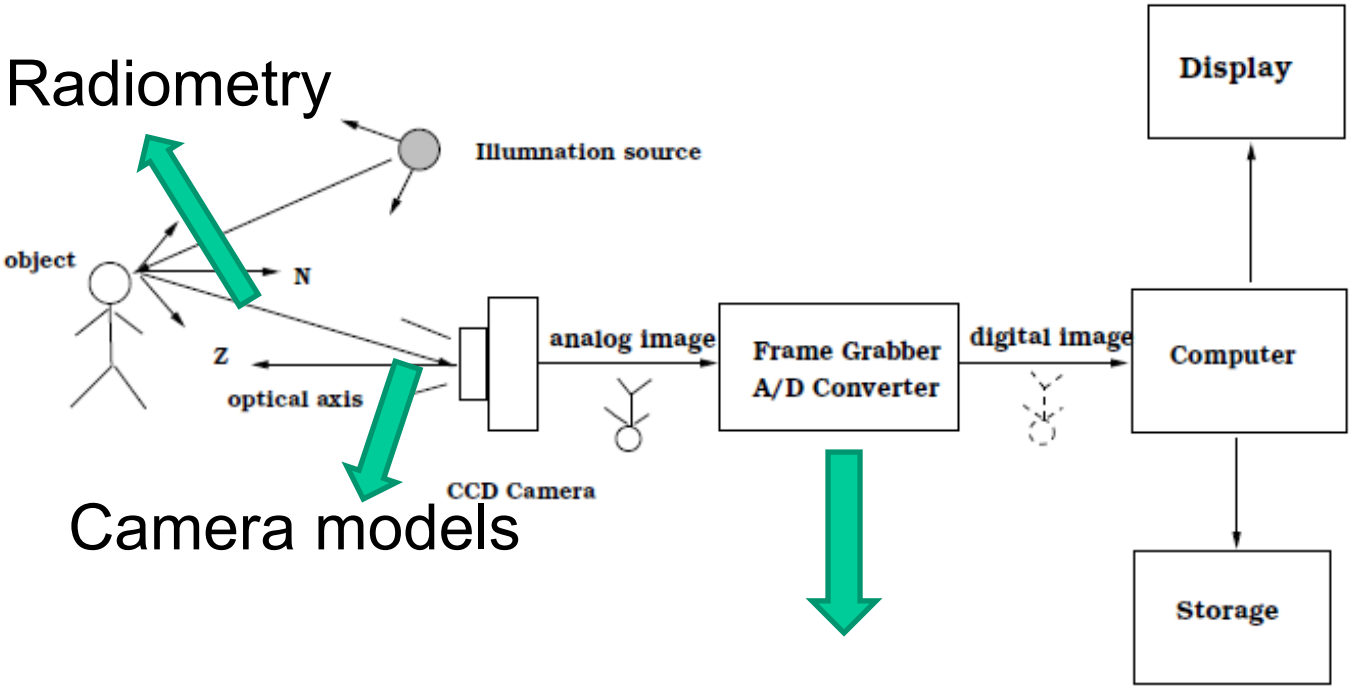
Cover everything till the class on Wednesday, March 1st

Open book and open notes

Review for Midterm

Digital Image Acquisition

Radiometry



Camera models

Sampling: Digitizing the coordinate values

- sampling frequency (pixels/mm)

Quantization: Digitizing the amplitude values

- Intensity range and levels

Points, Lines, and Planes

Point: represented as a vector $\mathbf{p} = [p_1, p_2, \cdots, p_N]$ in Euclidean space $\mathbb{R}^N$

Line:

- **Lines in 2D**
  - Slope-intercept form: $y = ax + b$
  - $\mathbf{l} \cdot \mathbf{p} = 0$, where $\mathbf{l}=[a, b, c]$, and $\mathbf{p}=[x, y, 1]$
- **Lines in general**
  - Parametric form: $\mathbf{p}(t) = \mathbf{p}_0 + t\,\mathbf{d}$, where $\mathbf{p}_0$ is any point on the line and $\mathbf{d}=[d_1, d_2, \cdots, d_N]$ is a unit vector - the direction of the line
    - Parallel lines have the same $\mathbf{d}$ with different $\mathbf{p}_0$
  - Point-normal: $\mathbf{m} \cdot (\mathbf{p} - \mathbf{p}_0) = 0$, where $\mathbf{p}_0$ is any point on the line and $\mathbf{m}$ is normal to the line

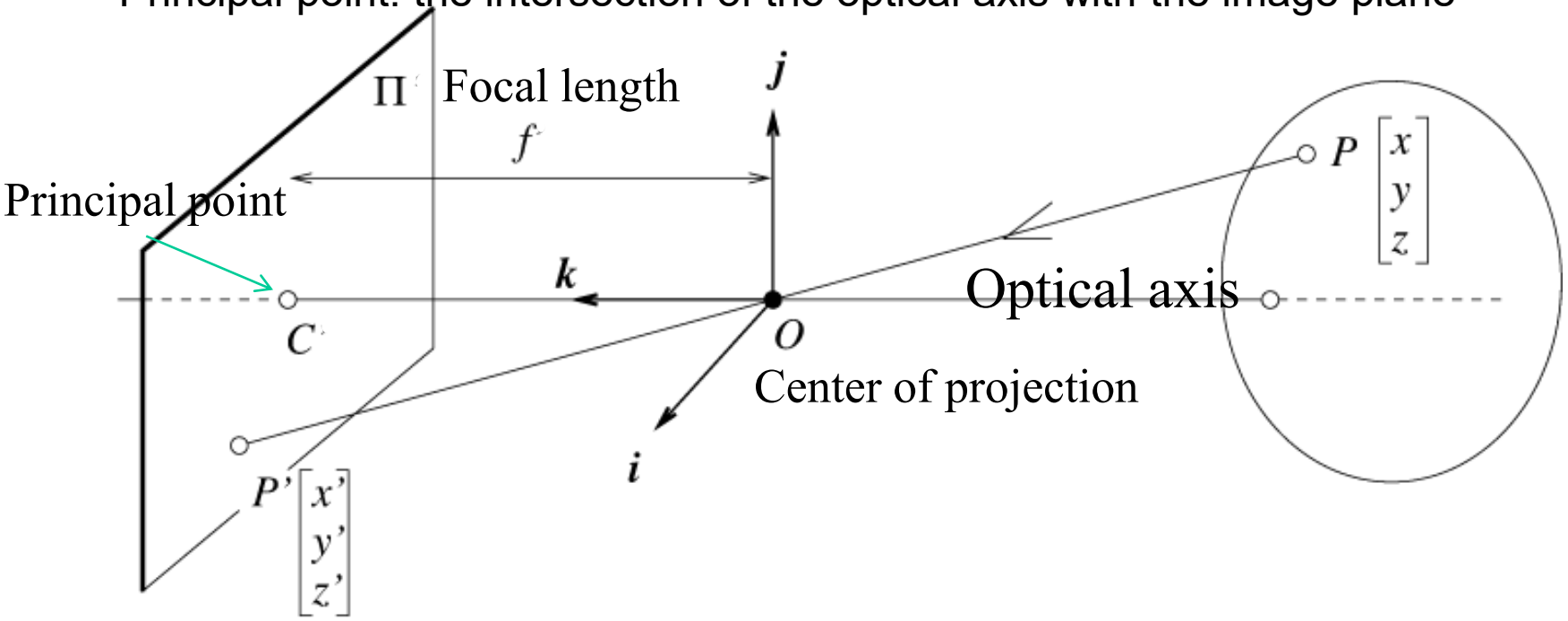
Plane: $\mathbf{n} \cdot (\mathbf{p} - \mathbf{p_0}) = \mathbf{0}$

$\mathbf{n}$ is the normal vector – perpendicular to the plane

You should know how to represent a point, a line, and a plane

The Equation of Projection

- Center of projection (O)
- Focal length (f): the distance between the image plane to O
- Optical axis: the line passing through O and perpendicular to the image plane
- Principal point: the intersection of the optical axis with the image plane



Only one coordinate system – camera coordinate system

The Equation of Perspective Projection

Cartesian coordinates:

- We have, by similar triangles, that

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

- Ignore the third coordinate, we get

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

Isotropic scaling

3D object point → 2D image point

The perspective projection is non-linear!

Projection of a Line

Lines project to lines

You should know how to calculate the 2D line from the 3D line

$$\mathbf{p}(t) = \mathbf{p_0} + t\mathbf{d}$$

A 3D line $\mathbf{p_0} = [x_0, y_0, z_0]$

$\mathbf{d} = [d_x, d_y, d_z]$

$$[u(t)\ v(t)] = \left[\frac{f(x_0 + td_x)}{z_0 + td_z}, \frac{f(y_0 + td_y)}{z_0 + td_z}\right]$$

When the 3D line goes to infinity, $t \to \infty$

$$[u(t)\ v(t)] = \left[\frac{f d_x}{d_z}, \frac{f d_y}{d_z}\right]$$  The vanishing point

Properties of Perspective Projection

Points project to points; lines project to lines; planes project to the whole or half image

Degenerate cases

- Line through focal point projects to a point.
- Plane through focal point projects to line

Scaling and foreshortening

Angles are not preserved

- Each set of parallel lines meets at a vanishing point unless they are parallel to the image plane
- Sets of parallel lines on the same plane lead to collinear vanishing points (horizon for that plane)

Cross-ratio is preserved

You should know how to apply these properties to solve real problem

Other Projection Models

Weak Perspective Projection Model

$$x = \frac{f}{\bar{Z}} X \qquad y = \frac{f}{\bar{Z}} Y$$

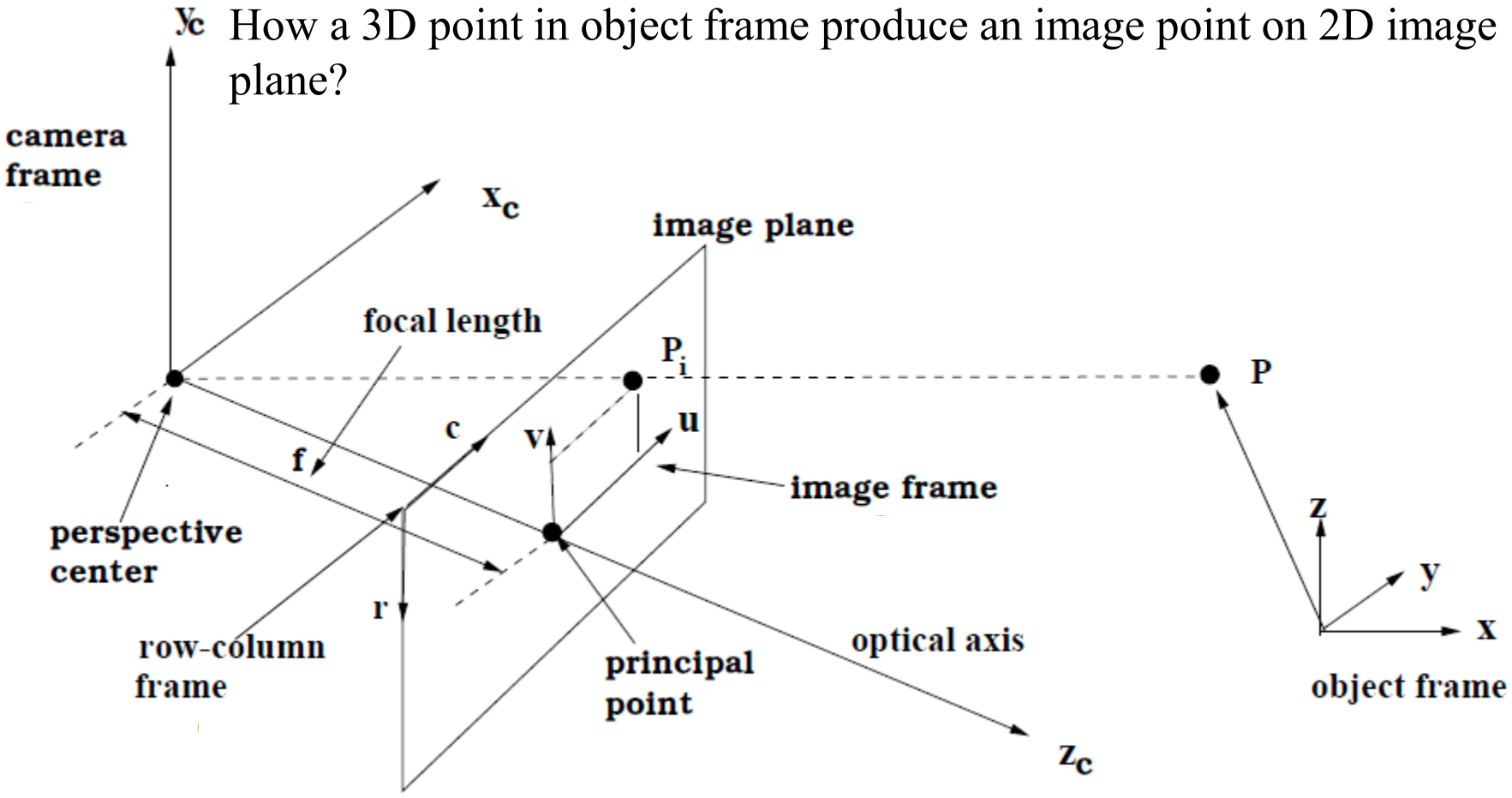
Orthographic Projection Model

$$x = X \qquad y = Y$$

You should know how and when these models work

Perspective Projection Geometry

How a 3D point in object frame produce an image point on 2D image plane?



You should know the relationships among different frames/coordinate system

Project a 3D Point in Camera Frame to a 2D Point in Row-Column Image Frame

Intrinsic Parameters (Do not depend on camera position):

$$\lambda \begin{bmatrix} u^{(I)} \\ v^{(I)} \\ 1 \end{bmatrix} = \begin{bmatrix} f_u & 0 & u_0 & 0 \\ 0 & f_v & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X^{(C)} \\ Y^{(C)} \\ Z^{(C)} \\ 1 \end{bmatrix}$$

$$\lambda = Z^{(C)}$$

You should know how to calculate the 2D point given the camera parameters

Full Perspective Camera Model

$$\begin{bmatrix} f_u & 0 & u_0 & 0 \\ 0 & f_v & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} R_{3\times3} & T_{3\times1} \\ 0_{1\times3} & 1 \end{bmatrix} = \begin{bmatrix} f_u \mathbf{r}_1^t + u_0 \mathbf{r}_3^t & f_u t_X + u_0 t_z \\ f_v \mathbf{r}_2^t + v_0 \mathbf{r}_3^t & f_v t_Y + v_0 t_z \\ \mathbf{r}_3^t & t_z \end{bmatrix}$$

The rotation matrix is an orthonormal matrix

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

Projection matrix P

$$\lambda \begin{bmatrix} u^{(I)} \\ v^{(I)} \\ 1 \end{bmatrix} = \begin{bmatrix} f_u \mathbf{r}_1^t + u_0 \mathbf{r}_3^t & f_u t_X + u_0 t_z \\ f_v \mathbf{r}_2^t + v_0 \mathbf{r}_3^t & f_v t_Y + v_0 t_z \\ \mathbf{r}_3^t & t_z \end{bmatrix} \begin{bmatrix} X^{(W)} \\ Y^{(W)} \\ Z^{(W)} \\ 1 \end{bmatrix}$$

$$\lambda = Z^{(C)} = r_3{}^T M^{(W)} + t_Z$$

Weak Perspective Camera Model

Overall Weak Perspective Camera Model:

$$\lambda \begin{bmatrix} u^{(I)} \\ v^{(I)} \\ 1 \end{bmatrix} = \begin{bmatrix} f_u & 0 & u_0 \\ 0 & f_v & v_0 \\ 0 & 0 & f \end{bmatrix} \begin{bmatrix} \mathbf{r}_1^t & t_x \\ \mathbf{r}_2^t & t_y \\ 0_{1\times 3} & \frac{\bar{Z}^{(C)}}{f} \end{bmatrix} \begin{bmatrix} X^{(W)} \\ Y^{(W)} \\ Z^{(W)} \\ 1 \end{bmatrix}$$

Projection matrix P_{weak}

$$P_{weak} = \begin{bmatrix} f_u \mathbf{r}_1^t & f_u t_X + u_0 \bar{Z}^{(C)} \\ f_v \mathbf{r}_2^t & f_v t_Y + v_0 \bar{Z}^{(C)} \\ 0_{1\times 3} & \bar{Z}^{(C)} \end{bmatrix} \quad \text{and} \quad \lambda = \bar{Z}^{(C)} \approx \mathrm{r}_3{}^t \bar{M}^{(W)} + t_Z$$

Conventional Camera Calibration

General strategy:

- Take pictures from calibration patterns with known coordinates of 3D features (e.g., points, lines, or curves)
- Identify the corresponding image features
- obtain the projection matrix by minimizing projection error
- obtain intrinsic (and extrinsic) parameters from the projection matrix

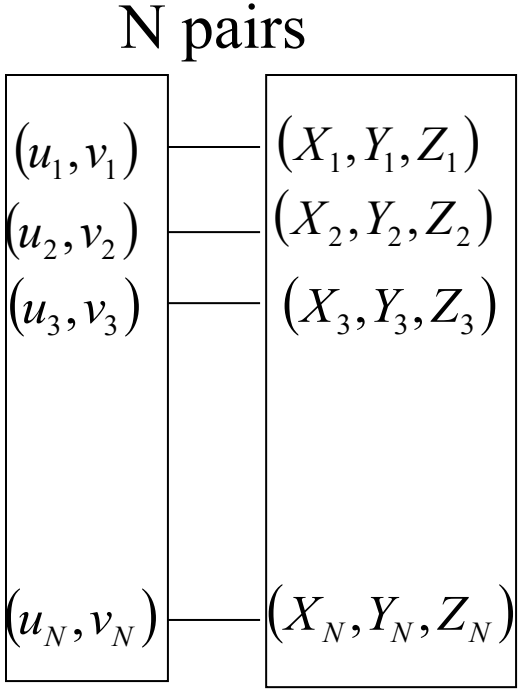
Projection matrix:

$$P = \begin{bmatrix} f_u \mathbf{r}_1^t + u_0 \mathbf{r}_3^t & f_u t_X + u_0 t_z \\ f_v \mathbf{r}_2^t + v_0 \mathbf{r}_3^t & f_v t_Y + v_0 t_z \\ \mathbf{r}_3^t & t_z \end{bmatrix}$$

Camera Calibration from Projection Matrix P: A Linear Method

For every pair of 3D-2D points, we have a pair of linear equations:

$$\mathbf{p}_1^t M'^{(W)} + p_{14} - \mathbf{p}_3^t M'^{(W)} u^{(I)} + p_{34} u^{(I)} = 0$$
$$\mathbf{p}_2^t M'^{(W)} + p_{24} - \mathbf{p}_3^t M'^{(W)} v^{(I)} + p_{34} v^{(I)} = 0$$



We can setup a linear equation systems for N corresponding pairs with 2N equations:

- Known: 2D and 3D coordinates
- Unknown: 4x3 parameters of **P**

$$\mathbf{Av} = 0$$

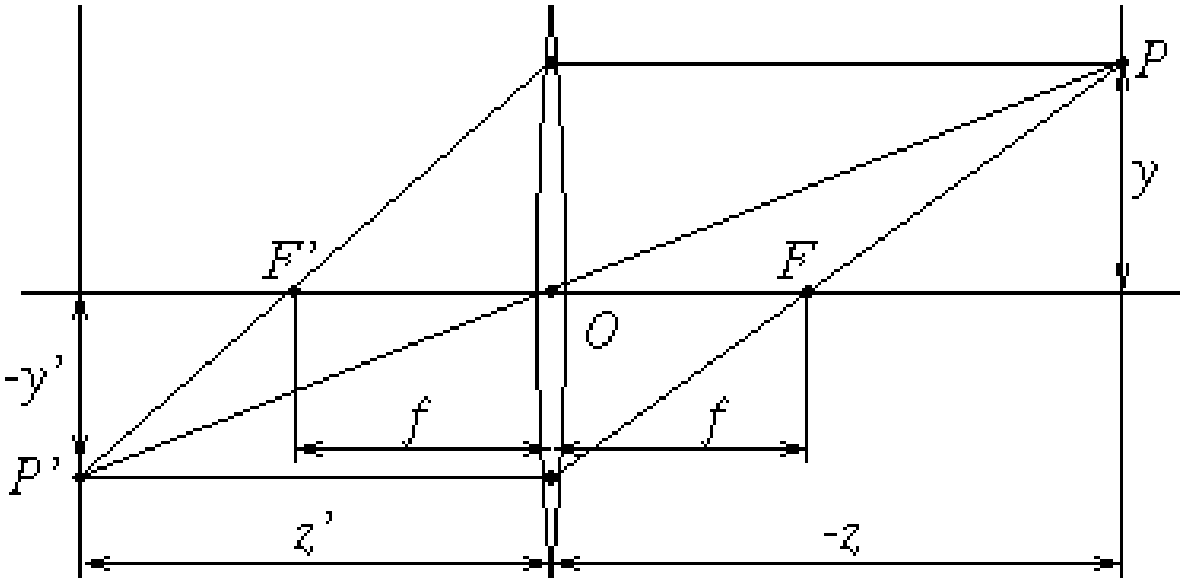
A is a $2N \times 12$ matrix containing the known

v is a 12×1 vector containing the unknown

$$\mathbf{v} = \begin{pmatrix} \mathbf{p}_1^t & p_{14} & \mathbf{p}_2^t & p_{24} & \mathbf{p}_3^t & p_{34} \end{pmatrix}^t$$

If rank(**A**) = 11, the solution to **v** is unique up to a scaling factor

Basic Optics: Thin Lens Model



$$\frac{1}{z'} + \frac{1}{|z|} = \frac{1}{f}$$

Field of View: $\omega = 2arctan\frac{d}{f}$

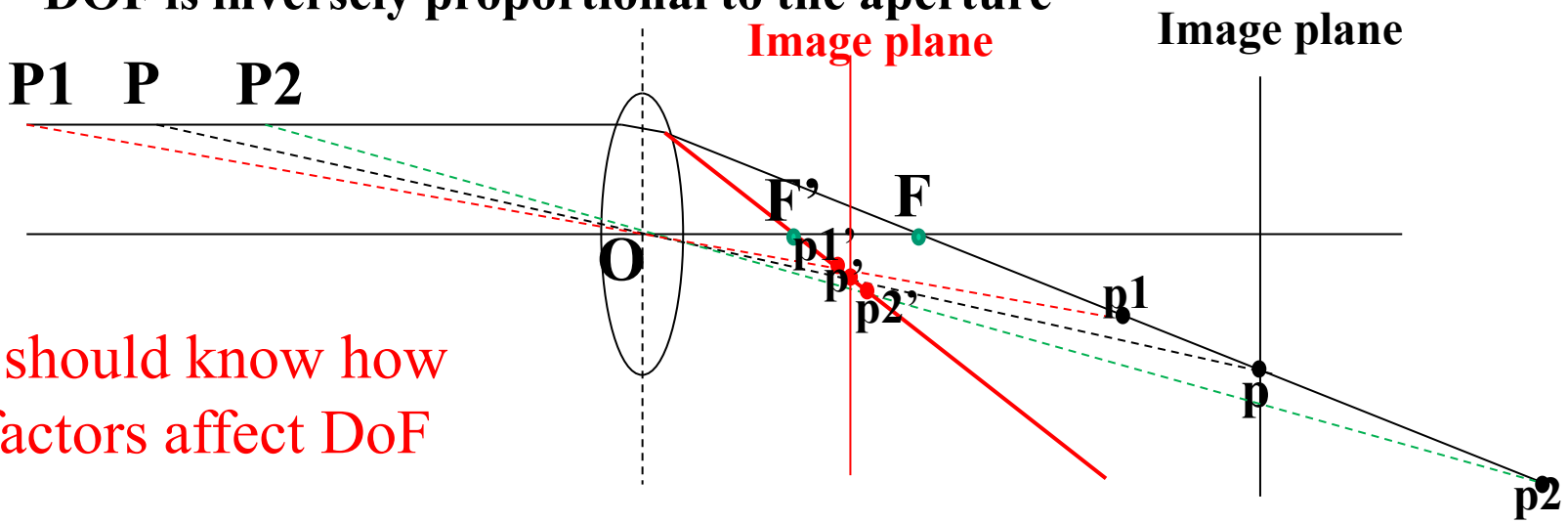
Increase focal length:

- Increase image size
- decrease the FOV
- decrease the DoF

You should know how the model works

Depth of Field & Out of Focus

- DOF is proportional to object distance
  - Objects far away from the camera have a large DOF
- DOF is inversely proportional to the focus length
- DOF is inversely proportional to the aperture



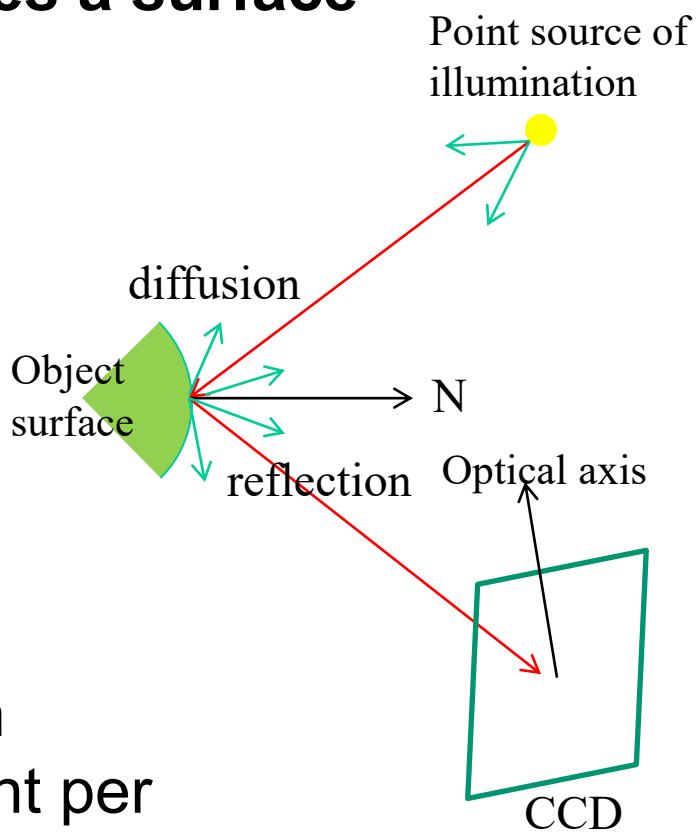
You should know how the factors affect DoF

More information on DoF: http://www.azuswebworks.com/photography/dof.html

Brightness

Many effects when light strikes a surface

- Absorbed;
- Transmitted;
- Reflected (specular);
- Scattered



The brightness is affected by

- Illumination
- Surface radiance: amount of radiation in a specific direction
- Image irradiance: power of light per unit area a CCD array element receives

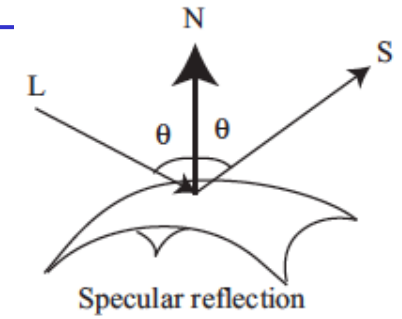
Important Reflection Modes

Specular reflection (mirror like)

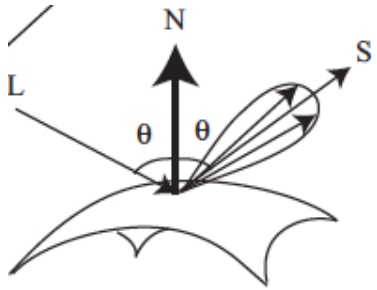
- Pure mirror:
  - incoming, outgoing directions and normal are coplanar
  - incoming, outgoing angles to normal are equal
- Most specular surfaces:
  - some light leaves the surface along directions near to the specular direction as well

Diffuse reflection

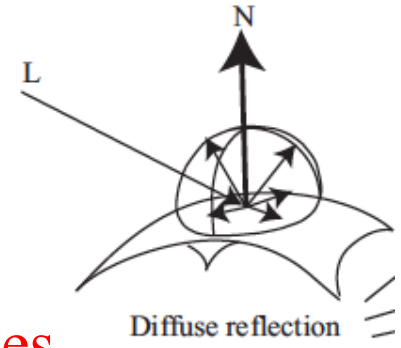
- Light leaves in equal amounts in each direction
- Described by one parameter: Albedo (reflection coefficient)



Specular reflection



Specularity



Diffuse reflection

You should know the difference between the two modes

How much light arrives: Lambertian Reflection

Assume source is far away

- So light travels in parallel rays
- (Light arriving) proportional to (number of rays striking surface)

Lambertian reflection: Ideal diffuse reflection

- Surface radiance follows the Lambert's cosine law

$$\boldsymbol{I_R = \rho L \cdot N}$$

$\boldsymbol{I_R}$: surface radiance; $\boldsymbol{\rho}$: surface albedo;

$\boldsymbol{L}$: intensity of incoming light; $\boldsymbol{N}$: surface normal

Fundamental Equation of Radiometric Image Formation

illumination ⟹ **intensity**

$$I = \beta \boldsymbol{\rho} \frac{\boldsymbol{\pi}}{\mathbf{4}} \left(\frac{\boldsymbol{d}}{\boldsymbol{f}} \right)^{\mathbf{2}} \boldsymbol{cos^4 \alpha L \cdot N}$$

Sensor determined

Photometric Stereo

Recover shape and albedo of surfaces from multiple shaded images

Assume:

- A set of point sources that are infinitely distant
- A set of pictures of an object, obtained in exactly the same camera/object configuration but using different illumination sources
- Pictures taken based on an orthographic camera model
- A Lambertian object (or the specular component has been identified and removed)

What do we want

- Reconstruct a patch of surface in 3D space

Major steps:

You should know how photometric stereo works

- Preprocessing
  - Points in shade
  - Specularity
- Estimate the surface normal at a point
- Estimate 3D surface from normals

Shadows

Most shadows aren't dark because shadow points get light from other surfaces, not just light source

Area sources are large and bright areas and yield smooth and blurry shadows

- Points that can see the whole source are brighter
- Points that can see only part of the source are darker (**penumbra**)
- Points that can see no part of the source are darkest (**umbra**)

Important Noise Models

Gaussian noise

Rayleigh noise

Gamma noise

Exponential noise

Uniform noise

Continuous noise model

Image denoise by Gaussian filter, averaging, and other mean filters

Impulse noise (salt and/or pepper)

- Image denoise by order statistic filters such as median, min, and max filters

Image Filtering

Modifying the pixels in an image based on some function of a local neighborhood of the pixels

- **Linear function**
  - Correlation $f(p) = \sum_{q_i \in N(p)} a_i q_i$
  - Convolution $f(i,j) = I \otimes H = \sum_k \sum_l I(k,l) H(i-k, j-l)$
- **Nonlinear function**
  - Order statistic (median)

You should know how to perform image filtering

Edges

Points of sharp change in an image:

- change in reflectance
- change in object
- change in illumination
- Noise

Edge detectors

$$\text{The first-order derivative}\ \frac{\partial f}{\partial x} = f'(x) = f(x+1) - f(x)$$

$$\text{The second-order derivative}\ \frac{\partial^2 f}{\partial x^2} = f''(x) = f(x+1) - 2f(x) + f(x-1)$$

You should know how to get edge

First-order Derivative based Edge Detection

Gradient

$$\nabla f(x,y) = grad(f) = \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \begin{bmatrix} \partial f/\partial x \\ \partial f/\partial y \end{bmatrix}$$

Edge strength $M(x,y) = \sqrt{g_x^2 + g_y^2}$

$$\approx |g_x| + |g_y|$$

Edge direction $\alpha(x,y) = \tan^{-1}\left[\frac{g_y}{g_x}\right]$

First-order derivatives generally produce thicker edges

Prewitt

−1	−1	−1
0	0	0
1	1	1

−1	0	1
−1	0	1
−1	0	1

0	1	1
−1	0	1
−1	−1	0

−1	−1	0
−1	0	1
0	1	1

Sobel

−1	−2	−1
0	0	0
1	2	1

−1	0	1
−2	0	2
−1	0	1

0	1	2
−1	0	1
−2	−1	0

−2	−1	0
−1	0	1
0	1	2

You should know how to calculate gradient

Second-order based Edge Detection: Laplacian

Second-order derivatives -- Laplacian filter

- have a stronger response to fine details, such as thin lines, isolated points, and noise
- produce a double-edge response at ramp and step transitions in intensity

-1	-1	-1
-1	8	-1
-1	-1	-1

1	1	1
1	-8	1
1	1	1

Laplacian of Gaussian filter – a combination of smoothing and second-order derivative

You should know what Laplacian filter does

Corner Detection

Corners are located in the region with large intensity variations in a region

Build a matrix for a point *p* in its neighborhood

$$H = \begin{bmatrix} \sum_{N(p)} G_x^2 & \sum_{N(p)} G_x G_y \\ \sum_{N(p)} G_x G_y & \sum_{N(p)} G_y^2 \end{bmatrix}$$

λ_1 **and** λ_2 **are two eigenvalues of *H***

If $\min(\lambda_1, \lambda_2) > threshold$, **the point *p* is a corner**

You should know how it works and what the values of eigenvalues mean

SIFT Feature

Scale Invariant Feature Transform (SIFT)

- Invariant to uniform scaling and orientation
- Partially invariant to affine transformation and illumination changes

Algorithm overview

- Scale-space extrema detection based on Difference of Gaussian
- Keypoint localization
  - Eliminate low contrast points
  - Eliminate edge points
  - Assign major orientation to the keypoint
- Keypoint descriptor
  - Align the neighborhood with major orientation
  - Histogram of orientations in a neighborhood

Texture

Textures are made up of stylized subelements spatially repeated in meaningful ways

Representation: subelements and their statistics

Find subelements by applying filters

- Each filter corresponds to one pattern element
- **spots** and **oriented bars**
- Form an oriented pyramid (or equivalent set of responses to filters at different scales and orientations)
- Take statistics of responses

Representing a Region

Build a dictionary of subelements from pictures

- Clustering
- Vector quantization
- Summarize the pattern of patches

Describe the image using this dictionary

You should know how to represent a region

Texture synthesis

Simple case -- Fill holes

- Synthesize a single pixel in a large image
- Approach:
  - Match the window around that pixel to other windows in the image
  - Choose a value from the matching windows
    - most likely, uniformly and at random

Expand to large images

- Start: take a piece of the example image
- Fill in pixels on the boundary
  - But these are missing more than the center
    - Discount missing pixels when matching
- Each time you fill in a pixel, you can use that to match