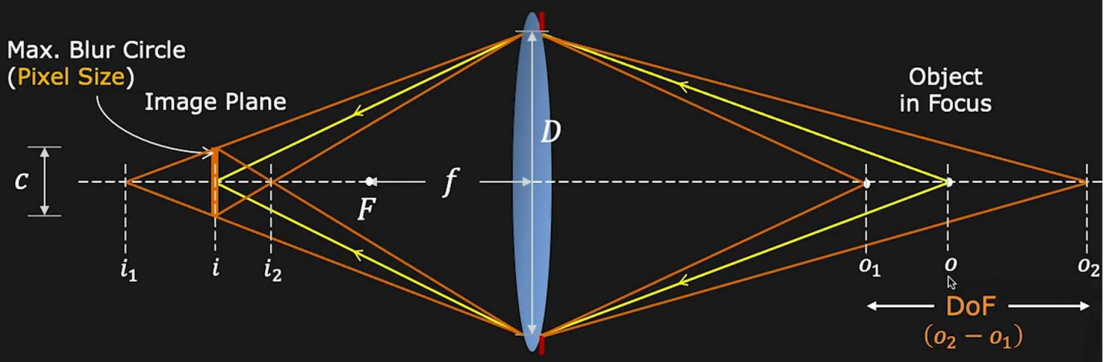
Depth of Field (DoF)

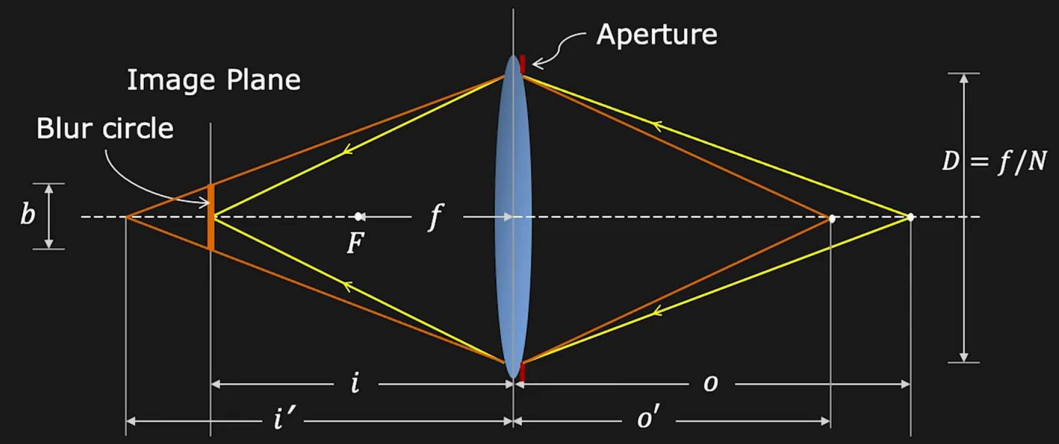


If o_1 and o_2 are the nearest and farthest distances respectively for which blur circle is maximum c, then:

$$c = \frac{f^2(o - o_1)}{No_1(o - f)} \qquad c = \frac{f^2(o_2 - o)}{No_2(o - f)}$$

Depth of Field: $$o_2 - o_1 = \frac{2of^2cN(o - f)}{f^4 - c^2N^2(o - f)^2}$$

Lens Defocus



From similar triangles:

$$\frac{b}{D} = \frac{|i' - i|}{i'}$$

Blur circle diameter:

$$b = \frac{D}{i'}|i' - i| \qquad b \propto D \propto \frac{1}{}$$

Blur Circle (Defocus)

Focused Point	Defocused Point
$\frac{1}{i}+\frac{1}{o}=\frac{1}{f}$	$\frac{1}{i'}+\frac{1}{o'}=\frac{1}{f}$
$i=\frac{of}{o-f}$	$i'=\frac{o'f}{o'-f}$

(Gaussian Lens Law)

$$i'-i=\frac{f}{(o'-f)}\cdot\frac{f}{(o-f)}\cdot(o-o')$$

$$b=Df\left|\frac{(o-o')}{o'(o-f)}\right| \qquad \boxed{b=\frac{f^2}{N}\left|\frac{(o-o')}{o'(o-f)}\right|}$$

$$z_o=\frac{f^2}{Nc}+f$$

Shape From Shading

Method for recovering 3D shape information from a single image using shading.

Topics:

(1) Human Perception of Shading

(2) Shape From Shading Algorithm

Shape from a Single Image?

Given image I, source direction $\mathbf{s}$ and surface reflectance

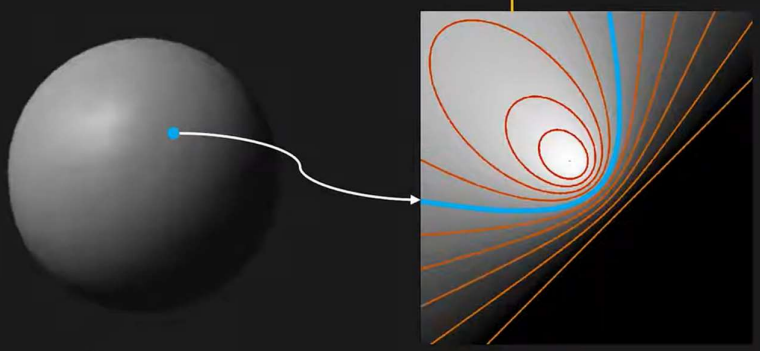
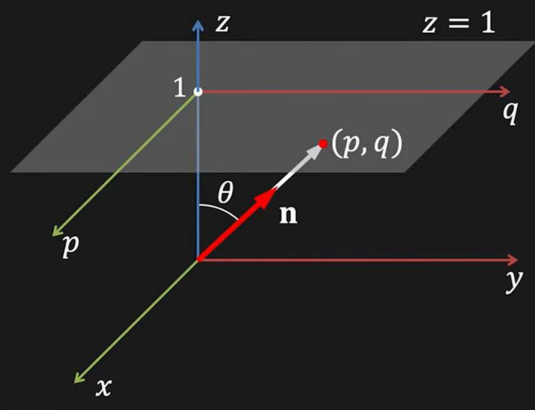


Image I Reflectance Map $R(p,q)$

Can we estimate surface gradient (p,q) from single intensity?

Stereographic Projection: fg Space

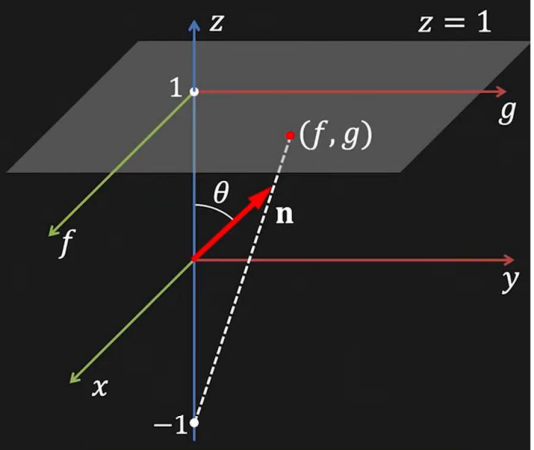
pq Space



Problem:

p or q is infinite when $\theta = 90°$

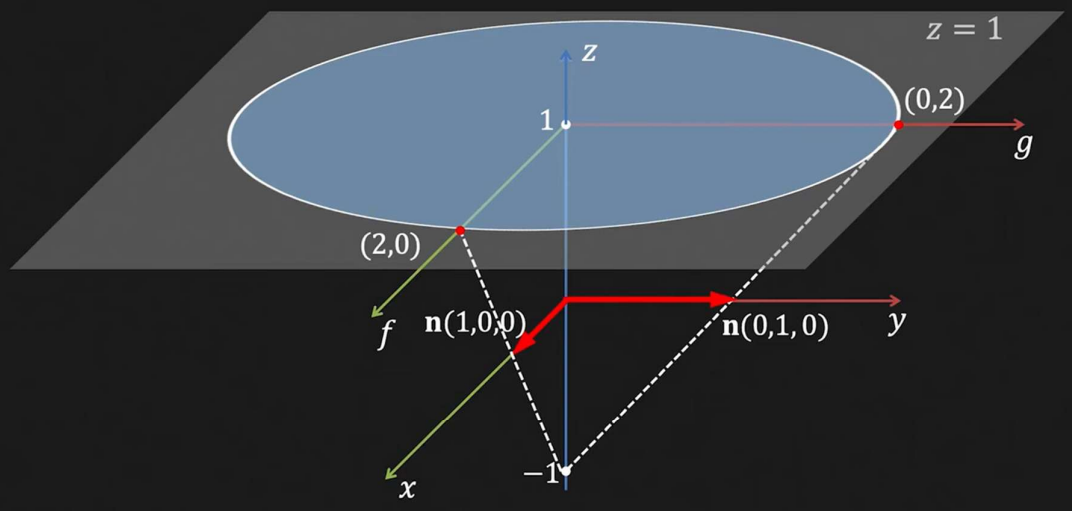
fg Space



$$f = \frac{2p}{1 + \sqrt{p^2 + q^2 + 1}}$$

$$g = \frac{2q}{1 + \sqrt{p^2 + q^2 + 1}}$$

Stereographic Projection: fg Space



All possible values of surface gradients
lie within a circle of radius 2 on the fg Plane.

For a given source direction $\mathbf{s}$ and surface reflectance, reflectance map relates image intensity to its surface gradients:

$$I = R_s(f, g)$$

Image Intensity

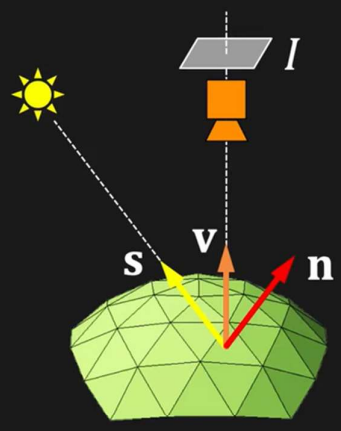
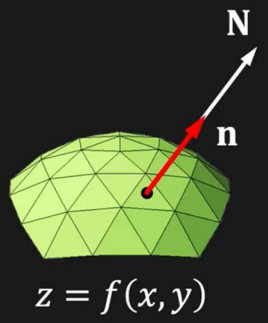


Image Intensity $I = \mathcal{F}$(Source,
Surface Normal $\mathbf{n}$,
Surface Reflectance)

Surface Gradient and Normal

Let $z = f(x, y)$ represent a 3D surface.



$z = f(x, y)$

Surface gradient:

$$\left(-\frac{\partial z}{\partial x}, -\frac{\partial z}{\partial y}\right) = (p, q)$$

Surface normal:

$$\mathbf{N} = \left(-\frac{\partial z}{\partial x}, -\frac{\partial z}{\partial y}, 1\right) = (p, q, 1)$$

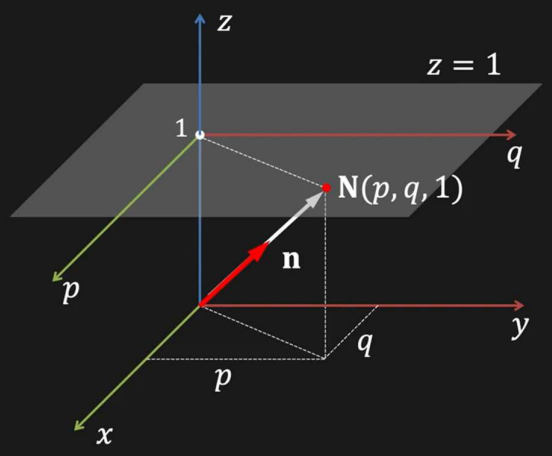
Unit surface normal:

$$\mathbf{n} = \frac{\mathbf{N}}{\|\mathbf{N}\|} = \frac{(p, q, 1)}{\sqrt{p^2 + q^2 + 1}}$$

Surface Normal represented with only two parameters (p

Gradient Space

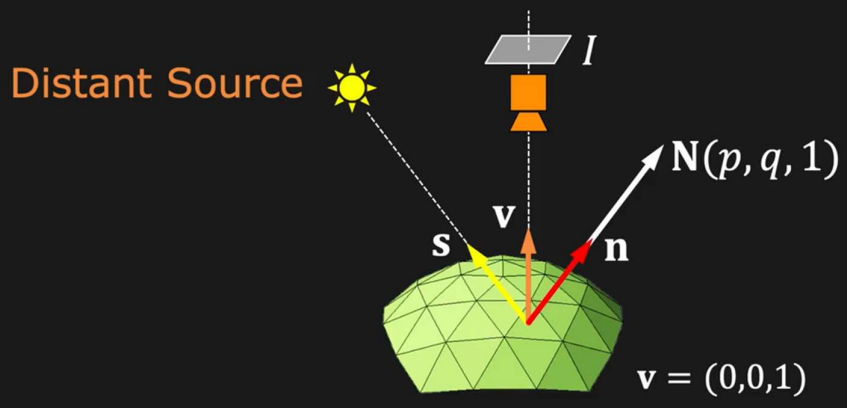
Plane $z = 1$ is called the Gradient Space or pq Plane



Surface normal:

$$\mathbf{n} = \frac{\mathbf{N}}{\|\mathbf{N}\|} = \frac{(p, q, 1)}{\sqrt{p^2 + q^2 + 1}}$$

Reflectance Map $R(p,q)$



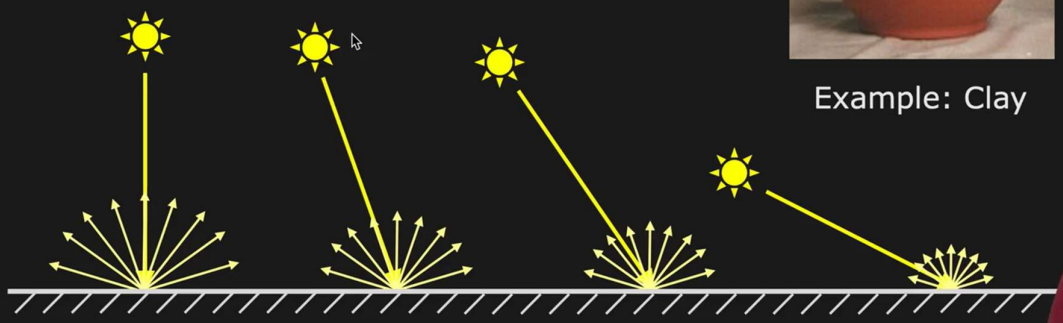
For a given source direction $\mathbf{s}$ and surface reflectance, image intensity at a point (x,y):

$$\boxed{I = R(p,q)}$$ Reflectance Map

Review: Lambertian Surface

Image intensity I is independent of viewing direction.

Example: Clay



Lambertian (Diffuse or Matte) Surface

Reflectance Map: Lambertian Surface

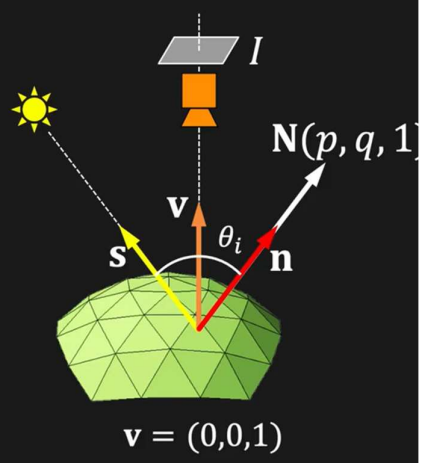
Image Intensity:

$$I = c\frac{\rho}{\pi}\frac{J}{r^2}\cos\theta_i = c\frac{\rho}{\pi}k(\mathbf{n}\cdot\mathbf{s})$$

where k: Source "Brightness"

ρ: Surface Albedo (Reflectance)

c: Constant (Camera Gain)



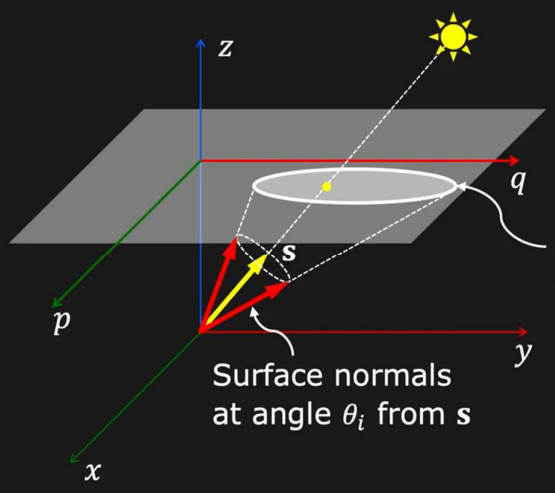
Let $c\frac{\rho}{\pi}k = 1$ then, $\boxed{I = \cos\theta_i = \mathbf{n}\cdot\mathbf{s}}$

Reflectance Map: Lambertian Surface

$$I = \mathbf{n}\cdot\mathbf{s} = \frac{pp_s + qq_s + 1}{\sqrt{p^2 + q^2 + 1}\sqrt{{p_s}^2 + {q_s}^2 + 1}} = R(p,q)$$

Reflectance Map: Iso-Brightness Contours

$$I = \mathbf{n} \cdot \mathbf{s} = \frac{pp_s + qq_s + 1}{\sqrt{p^2 + q^2 + 1}\sqrt{{p_s}^2 + {q_s}^2 + 1}} = R(p, q)$$



Shape from a Single Image?

Given image I, source direction $\mathbf{s}$ and surface reflectance

Reflectance Map $R(p, q)$

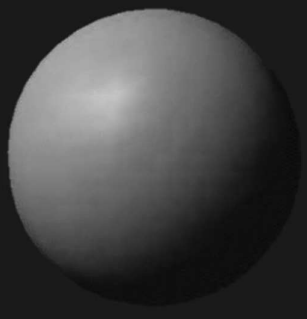
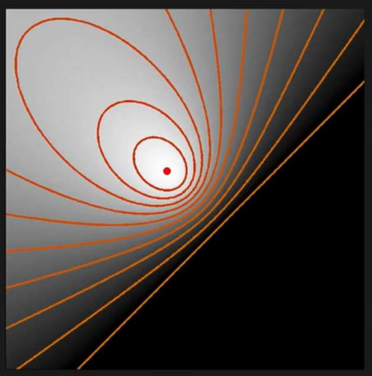


Image I



Reflectance Map $R(p, q)$

Can we estimate surface gradients (p, q) at each pixel? NO

Photometric Stereo: Basic Idea

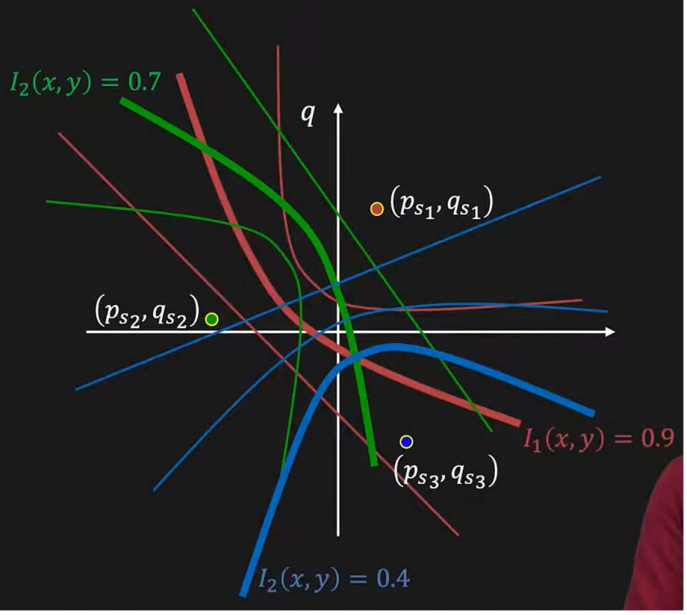
Capture Image I_3 under light source $\mathbf{s}_3(p_{s_3}, q_{s_3})$.

For Example:

Let $I_1(x,y) = 0.9$

$I_2(x,y) = 0.7$

$I_3(x,y) = 0.4$



Photometric Stereo: Basic Idea

Step 1: Acquire K images with K known light sources.

Step 2: Using known source direction and BRDF, construct reflectance map for each source direction.

Step 3: For each pixel location (x,y), find (p,q) as the intersection of K curves. This (p,q) gives the surface normal at pixel (x,y).

Smallest K needed depends on the material properties.

$$\begin{bmatrix} I_1 \\ I_2 \\ I_3 \end{bmatrix} = \frac{\rho}{\pi} \underbrace{\begin{bmatrix} s_{x_1} & s_{y_1} & s_{z_1} \\ s_{x_2} & s_{y_2} & s_{z_2} \\ s_{x_3} & s_{y_3} & s_{z_3} \end{bmatrix}}_{S_{3\times3} \text{ (Known)}} \mathbf{n} \quad \Rightarrow \quad \begin{cases} I = S\mathbf{N} \\ \text{where: } \mathbf{N} = \frac{\rho}{\pi}\mathbf{n} \end{cases}$$

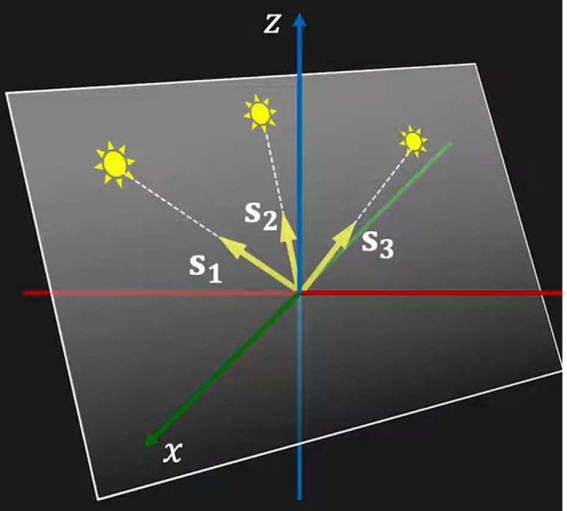
Solution: $\mathbf{N} = (S)^{-1} I$

Surface Normal: $\mathbf{n} = \frac{\mathbf{N}}{\|\mathbf{N}\|}$ Albedo: $\frac{\rho}{\pi} = \|\mathbf{N}\|$

When $S_{3\times3}$ is not invertible.

That is, when one source direction can be represen as a linear combination of the other two.

$\mathbf{s_3} = \alpha\mathbf{s_1} + \beta\mathbf{s_2}$



More Sources than Minimum Needed

Better results by using more ($K > 3$) light sources

$$\underbrace{\begin{bmatrix} I_1 \\ I_2 \\ \vdots \\ I_K \end{bmatrix}}_{I_{K\times 1}} = \frac{\rho}{\pi} \underbrace{\begin{bmatrix} s_{x_1} & s_{y_1} & s_{z_1} \\ s_{x_2} & s_{y_2} & s_{z_2} \\ \vdots & \vdots & \vdots \\ s_{x_K} & s_{y_K} & s_{z_K} \end{bmatrix}}_{S_{K\times 3}} \mathbf{n} \Rightarrow \begin{cases} I = S\mathbf{N} \\ \text{where: } \mathbf{N} = \frac{\rho}{\pi}\mathbf{n} \end{cases}$$

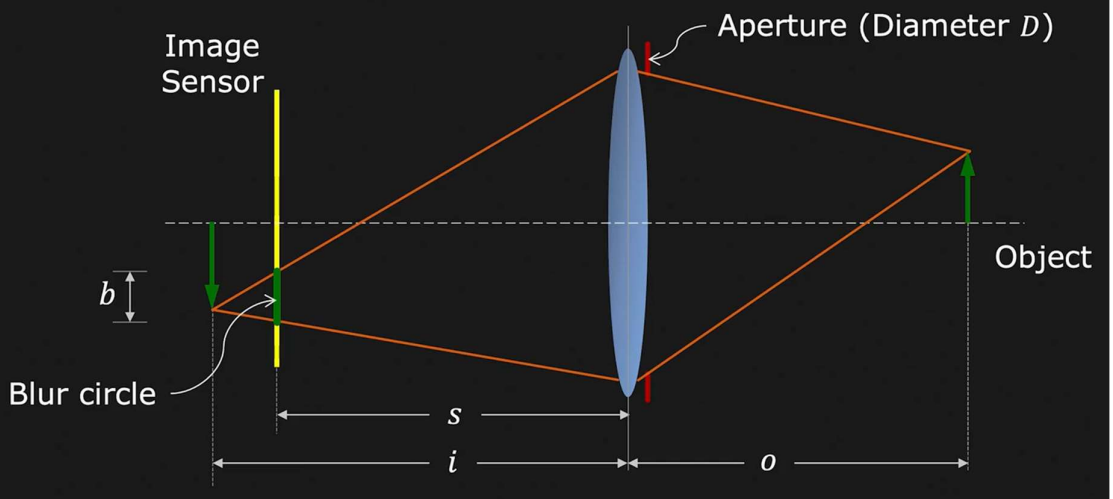
$S_{K\times 3}$ is not a square matrix and hence not invertible.

Solution: Use Least Squares Estimation

$$S^T I = \underbrace{S^T S}_{3\times 3}\mathbf{N}$$

$$\mathbf{N} = \underbrace{(S^T S)^{-1} S^T}_{\text{Pseudo Inverse}} I$$

Image Defocus and Sensor Location



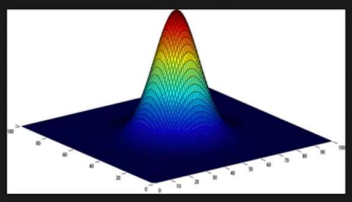
Blur circle diameter: $b = D\left|1 - \frac{s}{i}\right|$

Point Spread Function (PSF)

In practice, due to diffraction, lens aberrations and image sensing, the PSF often appears like a Gaussian function.

Gaussian PSF:

$$h(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

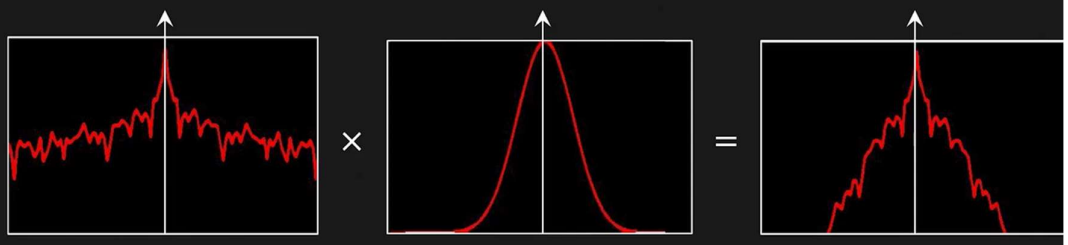


$h(x,y)$

$\sigma = b/2$

(approximation)

Defocus can be represented as product of Fourier transform



$F_0(u,v)$ $H(u,v)$ $F(u,v)$

(1D slice of Fourier transform)

Defocus is a Low-Pass Filter

Obtain scene depth using gaussian lens law.

$$\frac{1}{f} = \frac{1}{s} + \frac{1}{o} \quad \Rightarrow \quad \boxed{o = \frac{sf}{s - f}}$$

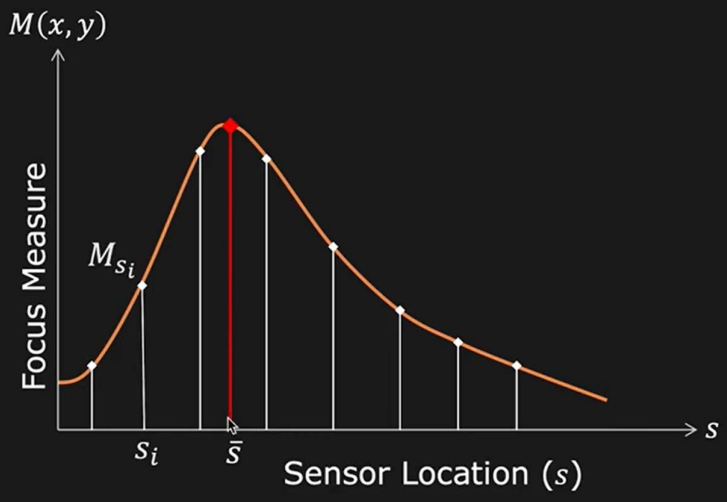
Ex: $s = 51.25\ mm$

$f = 50\ mm$

$o = 2.05\ m$

How to find the best focused image?

Peak of focus measure curve is a Gaussian-like function



Mean of the Gaussian may be used as the sensor location corresponding to the "best focus."

$$M_s = M_p \exp\left\{-\frac{1}{2}\left(\frac{s-\bar{s}}{\sigma_M}\right)^2\right\}$$

Gaussian Interpolation

Linearize problem to solve for $(\bar{s},\ M_p,\ \sigma_M)$.

Gaussian: $M_s = M_p \exp\left\{-\frac{1}{2}\left(\frac{s-\bar{s}}{\sigma_M}\right)^2\right\}$

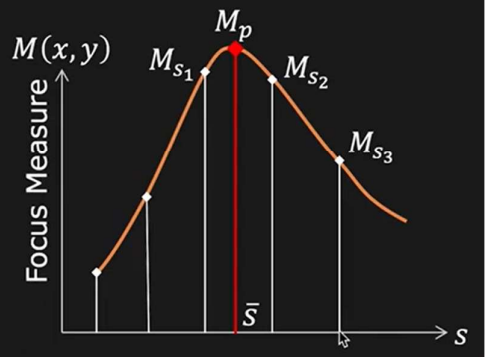
Taking the natural logarithm: $\ln M_s = \ln M_p - \frac{1}{2}\left(\frac{s-\bar{s}}{\sigma_M}\right)^2$

Three sensor positions:

$$\ln M_{s_1} = \ln M_p - \frac{1}{2}\left(\frac{s_1-\bar{s}}{\sigma_M}\right)^2$$

$$\ln M_{s_2} = \ln M_p - \frac{1}{2}\left(\frac{s_2-\bar{s}}{\sigma_M}\right)^2$$

$$\ln M_{s_3} = \ln M_p - \frac{1}{2}\left(\frac{s_3-\bar{s}}{\sigma_M}\right)^2$$



Where, $M_{s_1},\ M_{s_2}, M_{s_3}$ are the three largest values.

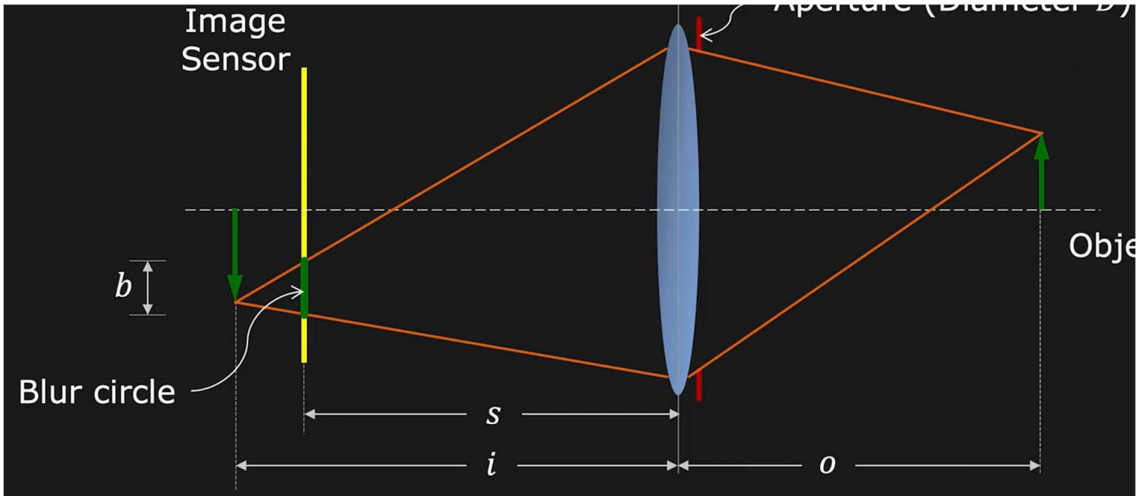
Solving for $\bar{s}$:

$$\bar{s} = +\frac{\left(\ln M_{s_2} - \ln M_{s_3}\right)\left(s_2^2 - s_1^2\right)}{2(s_3 - s_2)\left\{\left(\ln M_{s_2} - \ln M_{s_1}\right) + \left(\ln M_{s_2} - \ln M_{s_3}\right)\right\}}$$

$$-\frac{\left(\ln M_{s_2} - \ln M_{s_1}\right)\left(s_2^2 - s_3^2\right)}{2(s_3 - s_2)\left\{\left(\ln M_{s_2} - \ln M_{s_1}\right) + \left(\ln M_{s_2} - \ln M_{s_3}\right)\right\}}$$

Obtain scene depth using Gaussian Lens Law:

$$o = \frac{\bar{s}f}{\bar{s} - f}$$



We know that: $\frac{b}{D} = \frac{i-s}{i}$ and $o = \frac{if}{i-f}$

$\Rightarrow i = \frac{Ds}{D-b}$ $\Rightarrow o = \frac{sf}{s-f+b(f/D)}$

Can we estimate blur size b from a single image?

$*$

$=$

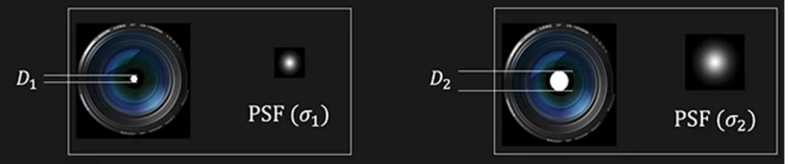
Scene
$f(x,y)$

Defocus PSF
$h(x,y)$

Image Patch
$g(x,y)$

Impossible: One equation, two unknowns

If the two images were taken with two known apertures then their blur sizes are related.



Blur circles:

$$b_1 = D_1 \left|1 - \frac{s}{i}\right| \qquad b_2 = D_2 \left|1 - \frac{s}{i}\right|$$

$$\sigma_1 = b_1/2 \qquad \sigma_2 = b_2/2$$

$$\frac{\sigma_1}{\sigma_2} = \frac{D_1}{D_2}$$

Three equations

$$g_1(x,y) = f(x,y) * h_{\sigma_1}(x,y)$$

$$g_2(x,y) = f(x,y) * h_{\sigma_2}(x,y)$$

$$\sigma_1/\sigma_2 = D_1/D_2$$

In Fourier Domain

$$G_1(u,v) = F(u,v) \times H_{\sigma_1}(u,v)$$

$$G_2(u,v) = F(u,v) \times H_{\sigma_2}(u,v)$$

$$\sigma_1/\sigma_2 = D_1/D_2$$

A Naïve DFD Algorithm

Cancel out $F(u,v)$:

$$\frac{G_1(u,v)}{G_2(u,v)} = \frac{F(u,v)\times H_{\sigma_1}(u,v)}{F(u,v)\times H_{\sigma_2}(u,v)} = \frac{H_{\sigma_1}(u,v)}{H_{\sigma_2}(u,v)}$$

Substitute for $H_{\sigma_1}(u,v)$ and $H_{\sigma_2}(u,v)$:

$$\frac{G_1(u,v)}{G_2(u,v)} = \frac{\exp(-2\pi^2(u^2+v^2)\sigma_1^2)}{\exp(-2\pi^2(u^2+v^2)\sigma_2^2)}$$

Taking the natural logarithm on both sides:

$$\sigma_1^2 - \sigma_2^2 = \frac{\ln G_2(u,v) - \ln G_1(u,v)}{2\pi^2(u^2+v^2)}$$

$$\sigma_1/\sigma_2 = D_1/D_2$$

Solve the above to get σ_1 and σ_2. Use either one to obtain object distance (depth).

Using σ_1 compute size of blur circle: $b_1 = 2\sigma_1$

Object distance (depth):

$$o = \frac{s_1 f}{s_1 - f + b_1(f/D)}$$

Forward Imaging Model: 3D to 2D

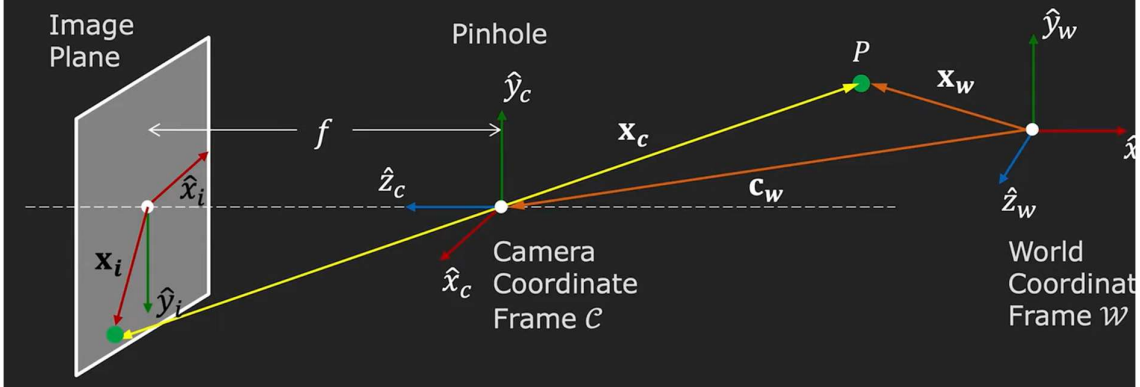


Image Coordinates

$$\mathbf{x}_i = \begin{bmatrix} x_i \\ y_i \end{bmatrix}$$

Perspective Projection

Camera Coordinates

$$\mathbf{x}_c = \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix}$$

Coordinate Transformation

World Coordinates

$$\mathbf{x}_w = \begin{bmatrix} x_w \\ y_w \\ z_w \end{bmatrix}$$

If m_x and m_y are the pixel densities (pixels/mm) in x and y directions, respectively, then pixel coordinates are:

$$u = m_x x_i = m_x f \frac{x_c}{z_c} \qquad v = m_y y_i = m_y f \frac{y_c}{z_c}$$

We usually treat the top-left corner of the image sensor as its origin (easier for indexing). If pixel (o_x, o_y) is the Principle Point where the optical axis pierces the sensor, then:

$$u = m_x f \frac{x_c}{z_c} + o_x \qquad v = m_y f \frac{y_c}{z_c} + o_y$$

$$u = m_x f \frac{x_c}{z_c} + o_x \qquad v = m_y f \frac{y_c}{z_c} + o_y$$

$$\boxed{u = f_x \frac{x_c}{z_c} + o_x \qquad v = f_y \frac{y_c}{z_c} + o_y}$$

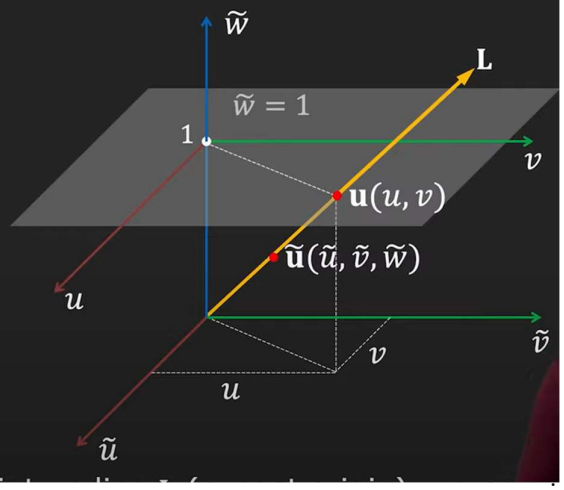
where: $(f_x, f_y) = (m_x f, m_y f)$ are the focal lengths in pixels in the x and y directions.

(f_x, f_y, o_x, o_y): Intrinsic parameters of the camera. They represent the camera's internal geometry.

The homogenous representation of a 2D point $\mathbf{u} = (u, v)$ is a 3D point $\tilde{\mathbf{u}} = (\tilde{u}, \tilde{v}, \tilde{w})$. The third coordinate $\tilde{w} \neq 0$ is fictitious such that:

$$u = \frac{\tilde{u}}{\tilde{w}} \qquad v = \frac{\tilde{v}}{\tilde{w}}$$

$$\boxed{\mathbf{u} \equiv \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \equiv \begin{bmatrix} \tilde{w}u \\ \tilde{w}v \\ \tilde{w} \end{bmatrix} \equiv \begin{bmatrix} \tilde{u} \\ \tilde{v} \\ \tilde{w} \end{bmatrix} = \tilde{\mathbf{u}}}$$



Perspective projection equations:

$$u = f_x \frac{x_c}{z_c} + o_x \qquad v = f_y \frac{y_c}{z_c} + o_y$$

Homogenous coordinates of (u, v):

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \equiv \begin{bmatrix} \tilde{u} \\ \tilde{v} \\ \tilde{w} \end{bmatrix} \equiv \begin{bmatrix} z_c u \\ z_c v \\ z_c \end{bmatrix} = \begin{bmatrix} f_x x_c + z_c o_x \\ f_y y_c + z_c o_y \\ z_c \end{bmatrix} = \begin{bmatrix} f_x & 0 & o_x & 0 \\ 0 & f_y & o_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix}$$

where: $(u, v) = (\tilde{u}/\tilde{w}, \tilde{v}/\tilde{w})$

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \equiv \begin{bmatrix} \tilde{u} \\ \tilde{v} \\ \tilde{w} \end{bmatrix} = \begin{bmatrix} f_x & 0 & o_x & 0 \\ 0 & f_y & o_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix}$$

Calibration Matrix:

$$K = \begin{bmatrix} f_x & 0 & o_x \\ 0 & f_y & o_y \\ 0 & 0 & 1 \end{bmatrix}$$

Upper Right Triangular Matrix

Intrinsic Matrix:

$$M_{int} = [K|0] = \begin{bmatrix} f_x & 0 & o_x & 0 \\ 0 & f_y & o_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

$$\tilde{\mathbf{u}} = [K|0]\,\tilde{\mathbf{x}}_c = M_{int}\,\tilde{\mathbf{x}}_c$$

Orthonormal Vectors and Matrices

Orthonormal Vectors: Two vectors $\mathbf{u}$ and $\mathbf{v}$ are orthonorma if and only if:

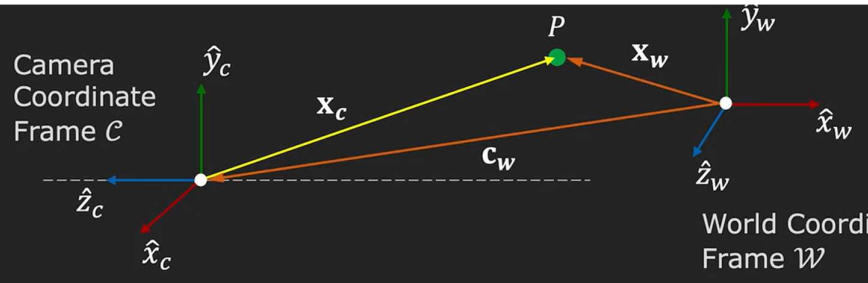
$$dot(\mathbf{u},\mathbf{v}) = \mathbf{u}^T\mathbf{v} = 0 \quad \text{and} \quad \mathbf{u}^T\mathbf{u} = \mathbf{v}^T\mathbf{v} = 1$$

(Orthogonality) (Unit length)

Example: The x-, y- and z-axes of $\mathbb{R}^3$ Euclidean space

Orthonormal Matrix: A square matrix R whose row (or column) vectors are orthonormal. For such a matrix:

$$R^{-1} = R^T \qquad R^TR = RR^T = I$$



Given the extrinsic parameters $(R, \mathbf{c}_w)$ of the camera, the camera-centric location of the point P in the world coordinate frame is:

$$\mathbf{x}_c = R(\mathbf{x}_w - \mathbf{c}_w) = R\mathbf{x}_w - R\mathbf{c}_w = R\mathbf{x}_w + \mathbf{t} \qquad \boxed{\mathbf{t} = -R\mathbf{c}_w}$$

$$\mathbf{x_c} = \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \begin{bmatrix} x_w \\ y_w \\ z_w \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix}$$

Rewriting using homogenous coordinates:

$$\tilde{\mathbf{x}}_c = \begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix}$$

Extrinsic Matrix: $M_{ext} = \begin{bmatrix} R_{3\times3} & \mathbf{t} \\ \mathbf{0}_{1\times3} & 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$

$$\tilde{\mathbf{x}}_c = M_{ext}\tilde{\mathbf{x}}_w$$

Camera to Pixel

$$\begin{bmatrix} \tilde{u} \\ \tilde{v} \\ \tilde{w} \end{bmatrix} = \begin{bmatrix} f_x & 0 & o_x & 0 \\ 0 & f_y & o_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix}$$

$$\tilde{\mathbf{u}} = M_{int}\,\tilde{\mathbf{x}}_c$$

World to Camera

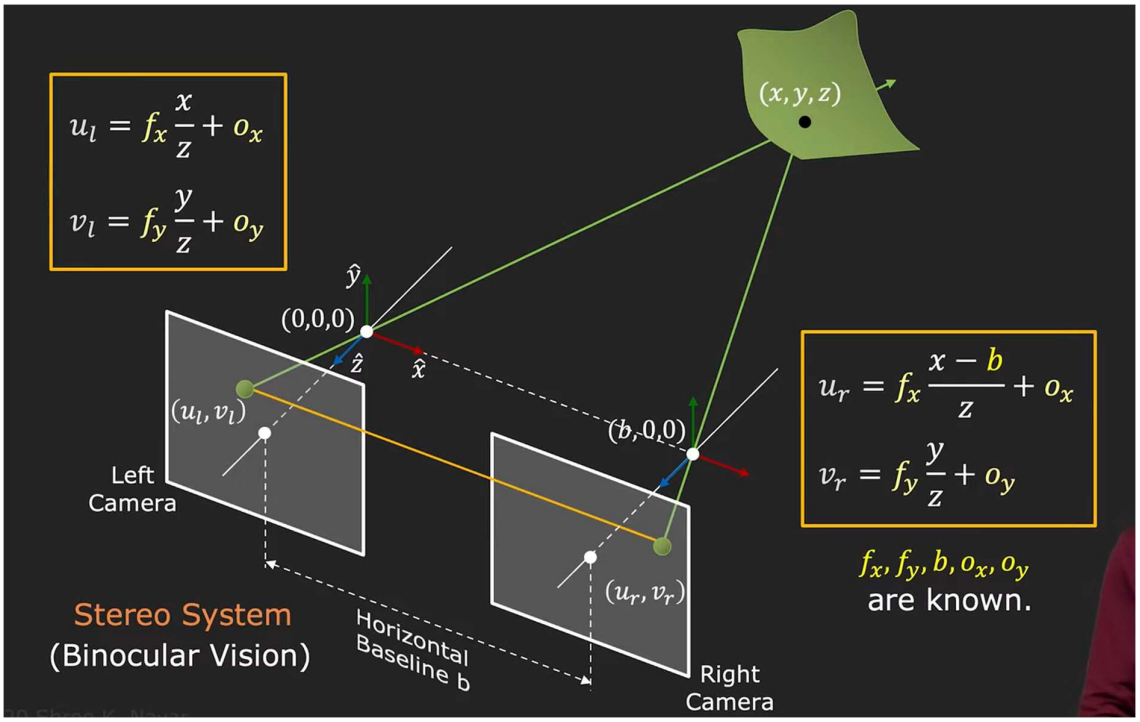
$$\begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix}$$

$$\tilde{\mathbf{x}}_c = M_{ext}\tilde{\mathbf{x}}_w$$

Combining the above two equations, we get the full projection matrix P:

$$\tilde{\mathbf{u}} = M_{int}M_{ext}\,\tilde{\mathbf{x}}_w = P\,\tilde{\mathbf{x}}_w$$

$$\begin{bmatrix} \tilde{u} \\ \tilde{v} \\ \tilde{w} \end{bmatrix} = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix} \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix}$$



From perspective projection:

$$(u_l, v_l) = \left(f_x \frac{x}{z} + o_x, f_y \frac{y}{z} + o_y\right) \qquad (u_r, v_r) = \left(f_x \frac{x-b}{z} + o_x, f_y \frac{y}{z} + o_y\right)$$

Solving for (x, y, z):

$$x = \frac{b(u_l - o_x)}{(u_l - u_r)} \qquad y = \frac{b f_x (v_l - o_y)}{f_y (u_l - u_r)} \qquad \boxed{z = \frac{b f_x}{(u_l - u_r)}}$$

where $(u_l - u_r)$ is called Disparity.

Depth z is inversely proportional to Disparity.

Disparity is proportional to Baseline.

Image velocity of a point that is moving in the scene

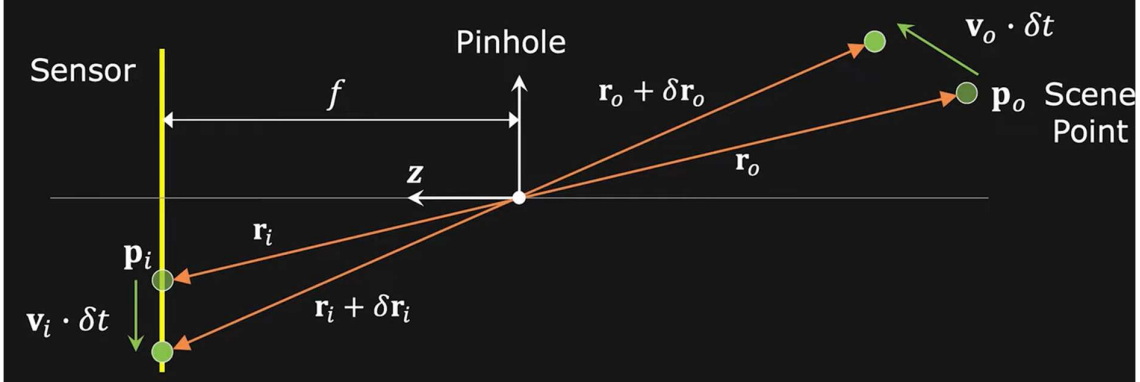


Image Point Velocity: $\mathbf{v}_i = \frac{d\mathbf{r}_i}{dt}$ (Motion Field)

Scene Point Velocity: $\mathbf{v}_o = \frac{d\mathbf{r}_o}{dt}$

Perspective projection: $\frac{\mathbf{r}_i}{f} = \frac{\mathbf{r}_o}{\mathbf{r}_o \cdot \mathbf{z}}$

Image Point Velocity: $\mathbf{v}_i = \frac{d\mathbf{r}_i}{dt} = f\frac{(\mathbf{r}_o \cdot \mathbf{z})\mathbf{v}_0 - (\mathbf{v}_o \cdot \mathbf{z})\mathbf{r}_0}{(\mathbf{r}_o \cdot \mathbf{z})^2}$ (Motion Field)

$$\mathbf{v}_i = f\frac{(\mathbf{r}_o \times \mathbf{v}_0) \times \mathbf{z}}{(\mathbf{r}_o \cdot \mathbf{z})^2}$$