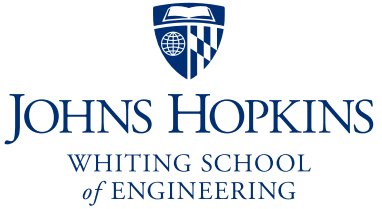
Johns Hopkins Engineering

Computer Vision

Camera Calibration and Photogrammetry



Camera Calibration and Photogrammetry

Method to find a camera's parameters and a method to estimate 3D structure using two cameras.

Topics:

(1) Linear Camera Model

(2) Camera Calibration

(3) Photogrammetry and Stereo

Camera Calibration and Photogrammetry

Method to find a camera's parameters and a method to estimate 3D structure using two cameras.

Topics:

(1) Linear Camera Model

Forward Imaging Model: 3D to 2D

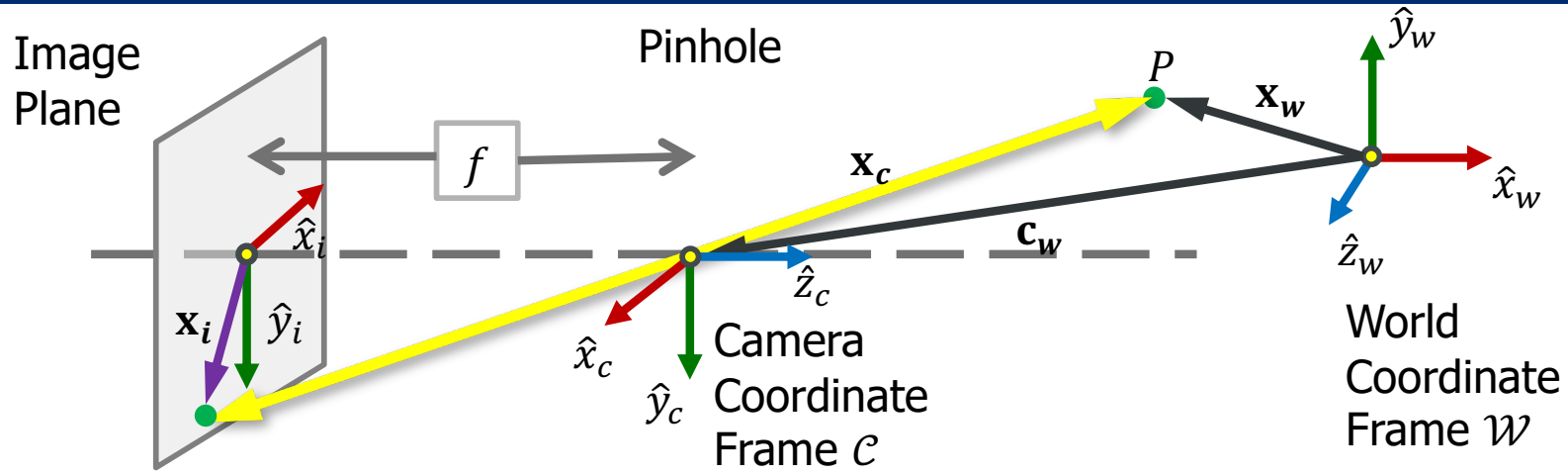


Image Plane Coordinates

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

Perspective Projection

Camera Coordinates

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

Coordinate Transformation

World Coordinates

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

Forward Imaging Model: 3D to 2D

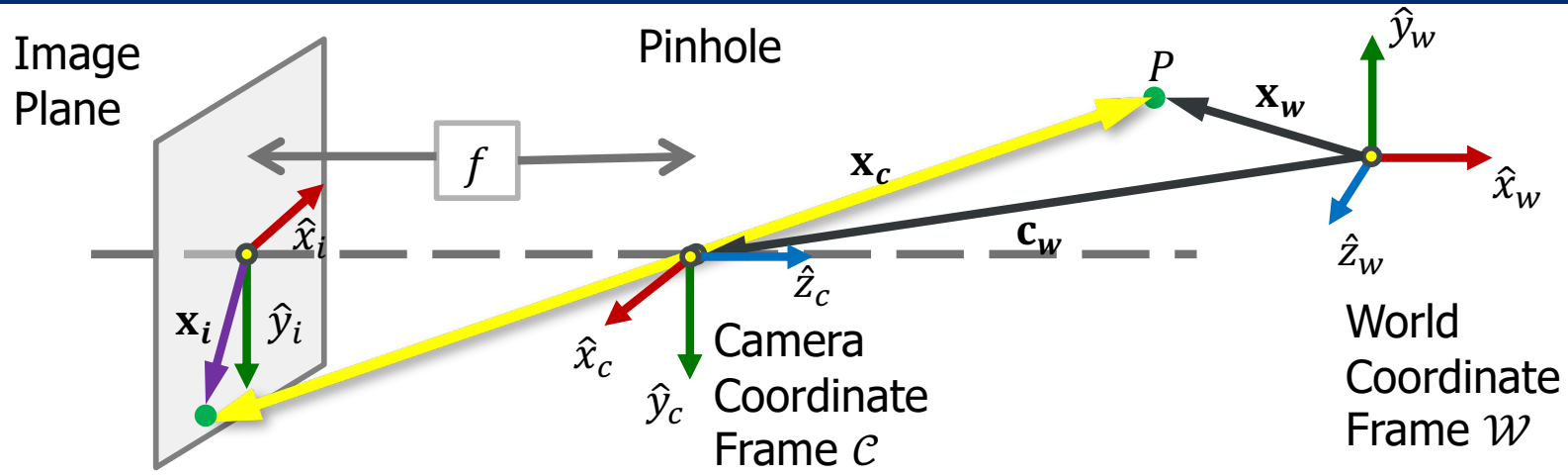


Image Plane Coordinates

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

Perspective Projection

Camera Coordinates

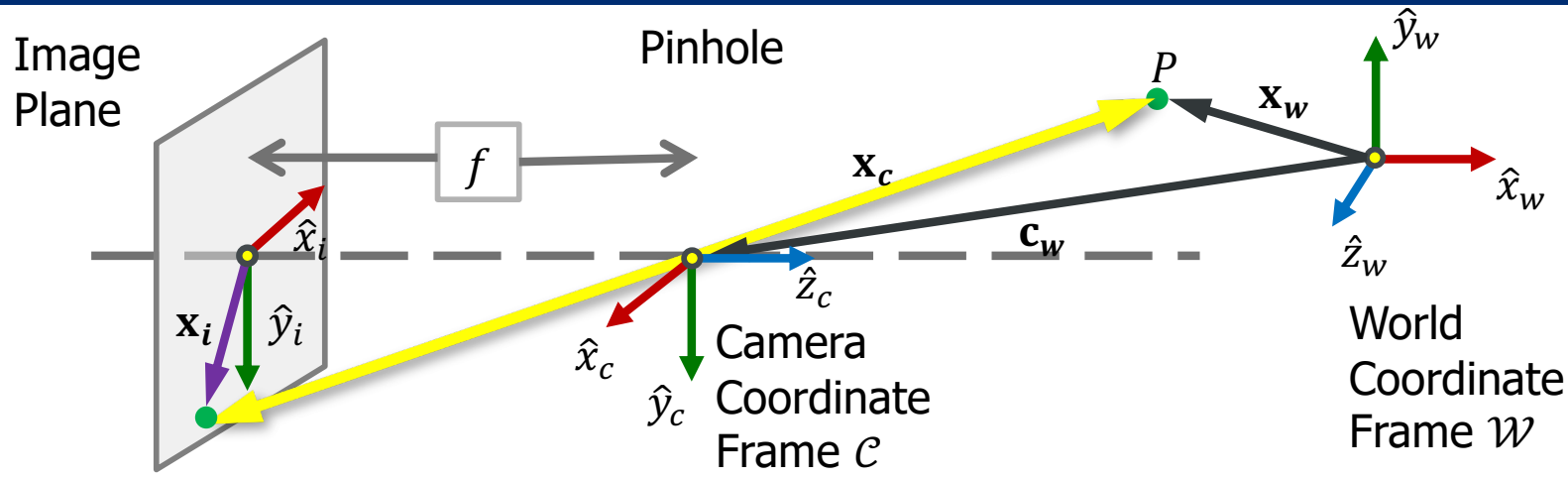
$$\mathbf{x}_c = \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix}$$

Coordinate Transformation

World Coordinates

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

Perspective Projection



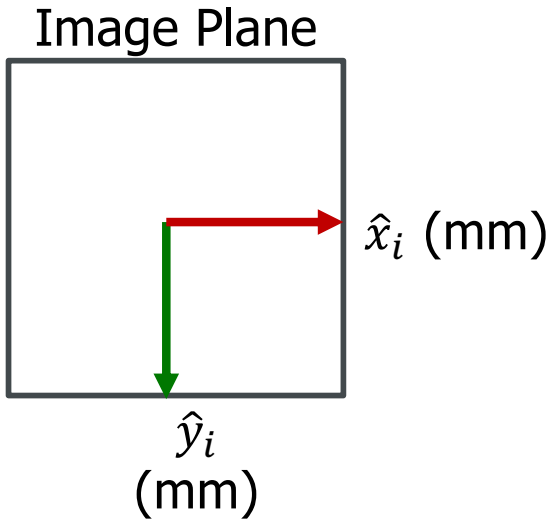
We know that: $\frac{x_i}{f} = \frac{x_c}{z_c}$ and $\frac{y_i}{f} = \frac{y_c}{z_c}$

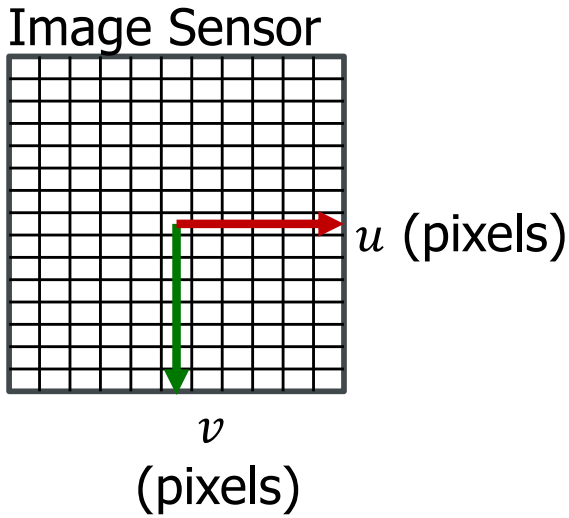
Therefore: $x_i = f\frac{x_c}{z_c}$ and $y_i = f\frac{y_c}{z_c}$

What are the units of x_i?

Image Plane to Image Sensor Mapping

Pixels may be rectangular.



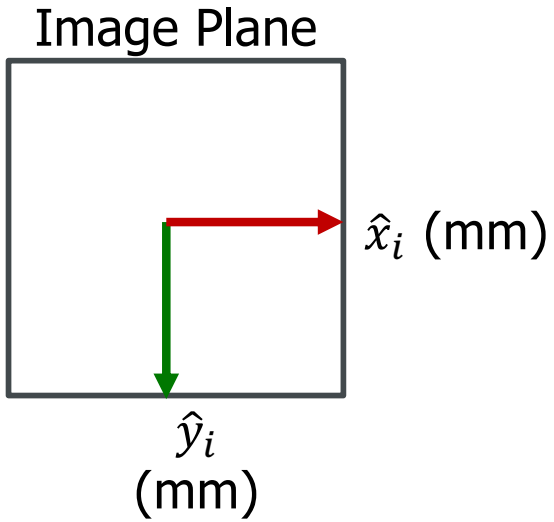


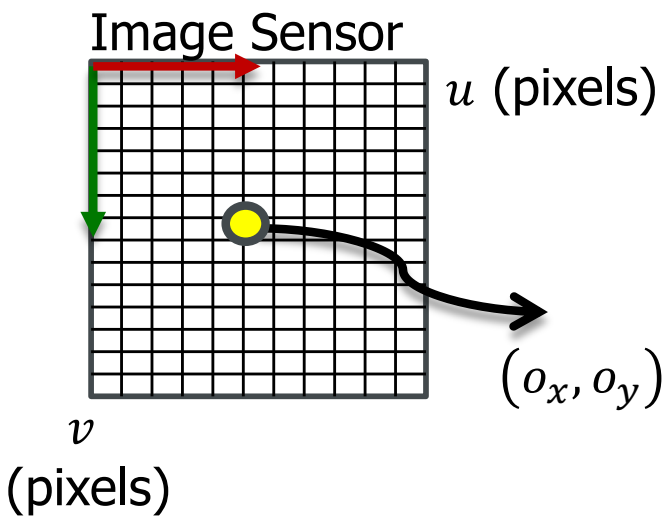
If m_x and m_y are the pixel densities (ex: pixels/mm) in x and y directions respectively, then pixel coordinates are:

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

Image Plane to Image Sensor Mapping (cont.)

Pixels may be rectangular.





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

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

Perspective Projection

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

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

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

$(\boldsymbol{f_x}, \boldsymbol{f_y}, \boldsymbol{o_x}, \boldsymbol{o_y})$**: Intrinsic parameters** of the camera. They represent the **camera's internal geometry**.

Perspective Projection (cont.)

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

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

Equations for Perspective projection are Non-Linear.

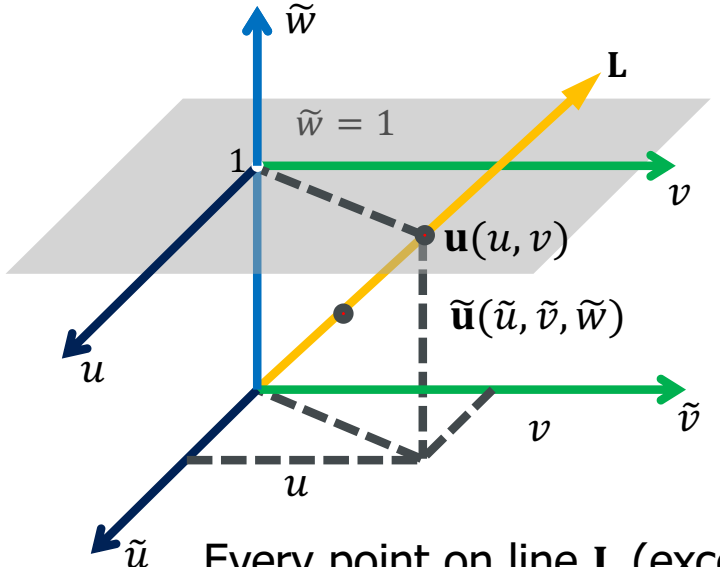
It is often convenient to express them as linear equations.

Homogenous Coordinates

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:

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

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



Every point on line $\mathbf{L}$ (except origin) represents the homogenous coordinate of $\mathbf{u}(u, v)$

Homogenous Coordinates (cont.)

- The homogenous representation of a 3D point $\mathbf{x} = (x, y, z)$ is a 4D point $\tilde{\mathbf{x}} = (\tilde{x}, \tilde{y}, \tilde{z}, \tilde{w})$. The fourth coordinate $\tilde{w} \neq 0$ is fictitious such that:

$$x = \frac{\tilde{x}}{\tilde{w}} \qquad y = \frac{\tilde{y}}{\tilde{w}} \qquad z = \frac{\tilde{z}}{\tilde{w}}$$

$$\mathbf{x} \equiv \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \equiv \begin{bmatrix} \tilde{w}x \\ \tilde{w}y \\ \tilde{w}z \\ \tilde{w} \end{bmatrix} \equiv \begin{bmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{z} \\ \tilde{w} \end{bmatrix} = \tilde{\mathbf{x}}$$

Perspective Projection in Homogenous Coordinates

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

Intrinsic Matrix

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

Calibration Matrix:

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

Intrinsic Matrix:

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

Upper Right Triangular Matrix

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

Forward Imaging Model: 3D to 2D

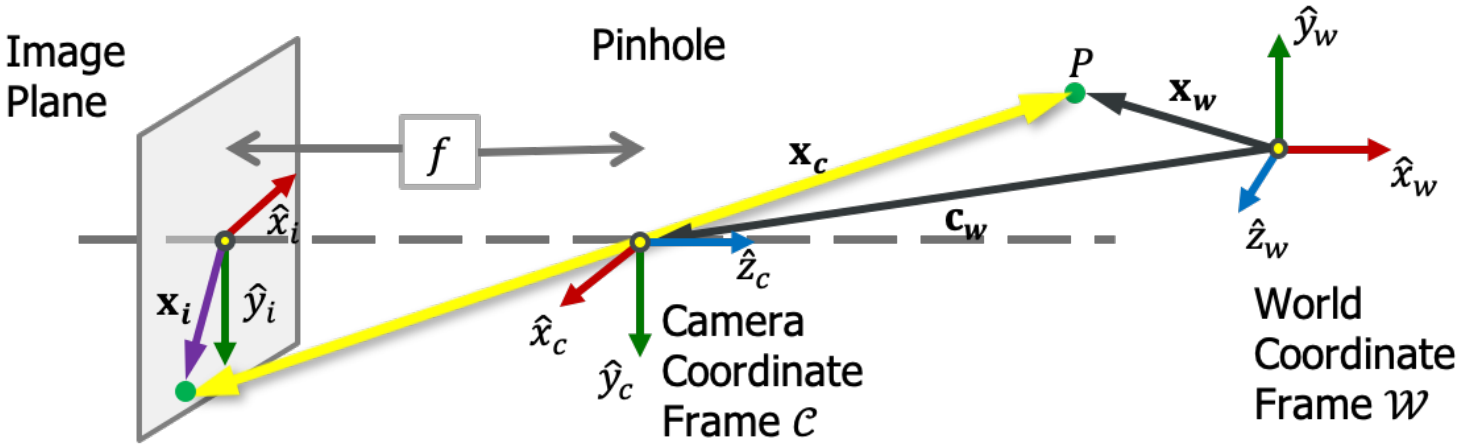


Image Coordinates

$$\mathbf{u} = \begin{bmatrix} u \\ v \end{bmatrix}$$

M_{int}

Perspective Projection

Camera Coordinates

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

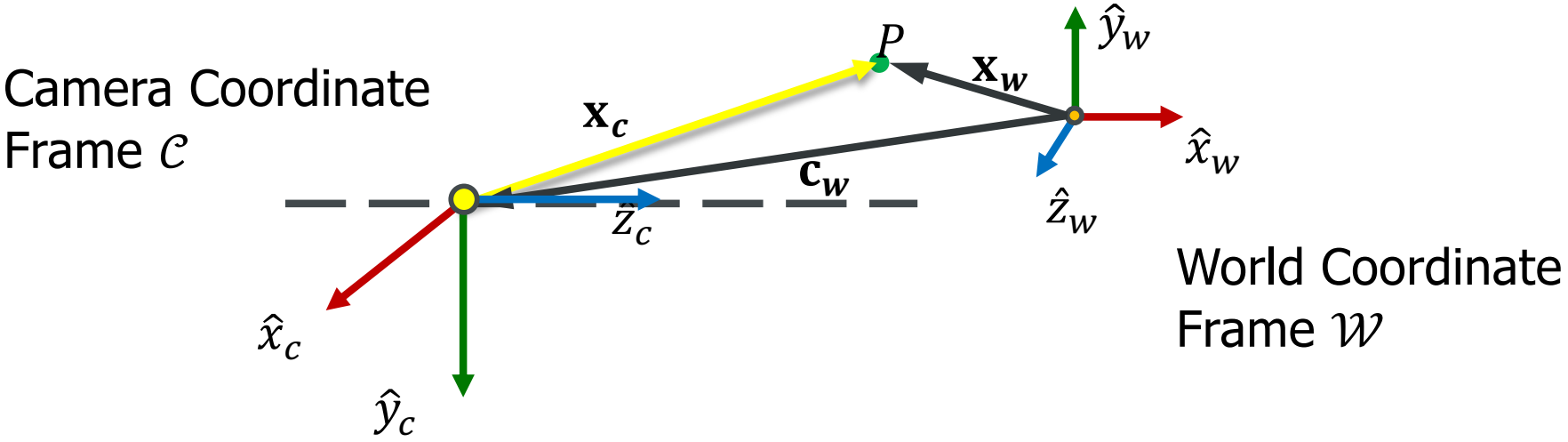
?

Coordinate Transformation

World Coordinates

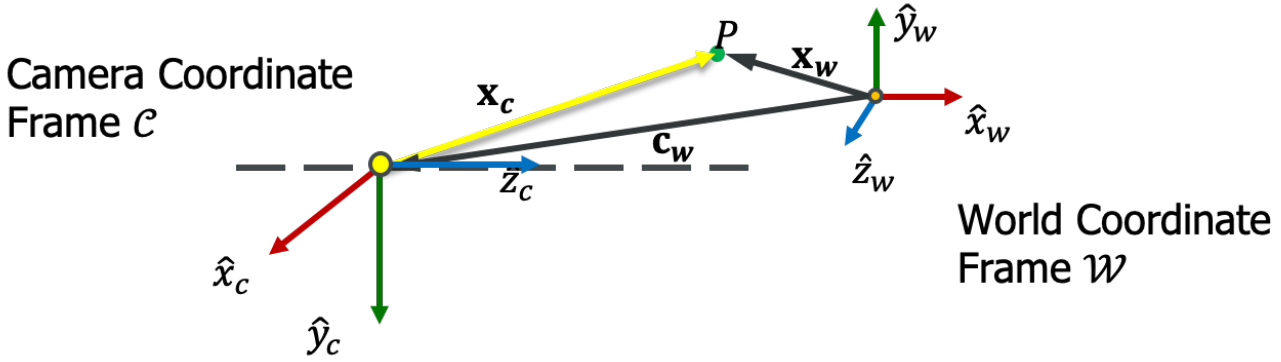
$$\mathbf{x}_w = \begin{bmatrix} x_w \\ y_w \\ z_w \end{bmatrix}$$

Extrinsic Parameters



Position $\mathbf{c}_w$ and Orientation R of the camera in the world coordinate frame $\mathcal{W}$ are the camera's **Extrinsic Parameters.**

World-to-Camera Transformation



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

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

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

World-to-Camera Transformation (cont.)

Rewriting using homogenous coordinates:

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

Extrinsic Matrix: $M_{ext} = \begin{bmatrix} R_{3\times3} & \mathbf{t}_{3\times1} \\ \mathbf{0}_{1\times3} & 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$

$$\tilde{\mathbf{x}}_{\boldsymbol{c}} = M_{ext}\tilde{\mathbf{x}}_{\boldsymbol{w}}$$

Forward Imaging Model: 3D to 2D (cont.)

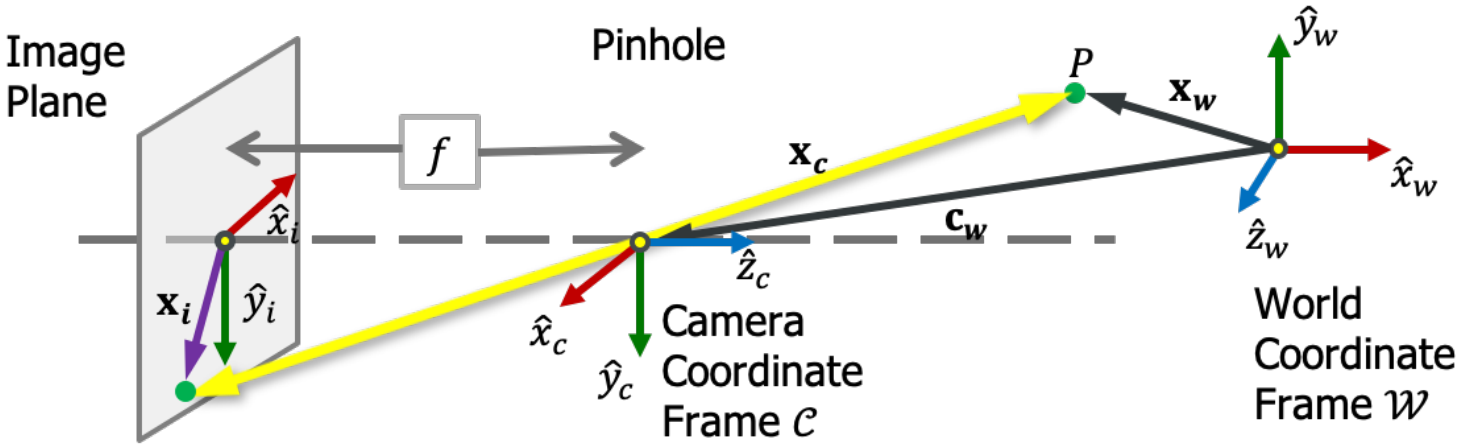


Image Coordinates

$$\mathbf{u} = \begin{bmatrix} u \\ v \end{bmatrix}$$

M_{int}

Perspective Projection

Camera Coordinates

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

M_{ext}

Coordinate Transformation

World Coordinates

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

Linear Camera Model

Camera to Pixel

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

$$\tilde{\mathbf{u}} = M_{int}\, \tilde{\mathbf{x}}_{\boldsymbol{c}}$$

World to Camera

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

$$\tilde{\mathbf{x}}_{\boldsymbol{c}} = M_{ext} \tilde{\mathbf{x}}_{\boldsymbol{w}}$$

Combining the above two equations, we get the Projection Matrix P:

$$\tilde{\mathbf{u}} = M_{int} M_{ext}\, \tilde{\mathbf{x}}_{\boldsymbol{w}} = P\, \tilde{\mathbf{x}}_{\boldsymbol{w}}$$

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

Scale of Projection Matrix

Projection matrix acts on homogenous coordinates.

We know that: $\begin{bmatrix} \tilde{u} \\ \tilde{v} \\ \tilde{w} \end{bmatrix} \equiv k \begin{bmatrix} \tilde{u} \\ \tilde{v} \\ \tilde{w} \end{bmatrix}$ ($k \neq 0$ is any constant)

That is: $\begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix} \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix} \equiv k \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix} \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix}$

Therefore, Projection Matrix P and kP produce the same homogenous pixel coordinates.

Projection Matrix P needs to be determined only up to a scale factor.

Summary

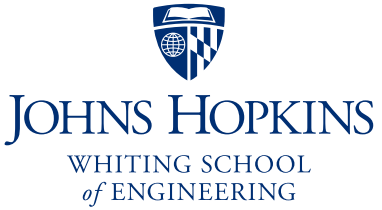
Camera Calibration and Photogrammetry: Method to find a camera's parameters and estimate 3D structure using two cameras.

- Essential concepts in this lecture:
  - 3D homogeneous transforms
  - Internal camera parameters
  - Full 3D linear projection model

Johns Hopkins Engineering

Computer Vision

Camera Calibration and Photogrammetry



Camera Calibration and Photogrammetry

Method to find a camera's parameters and a method to estimate 3D structure using two cameras.

Topics:

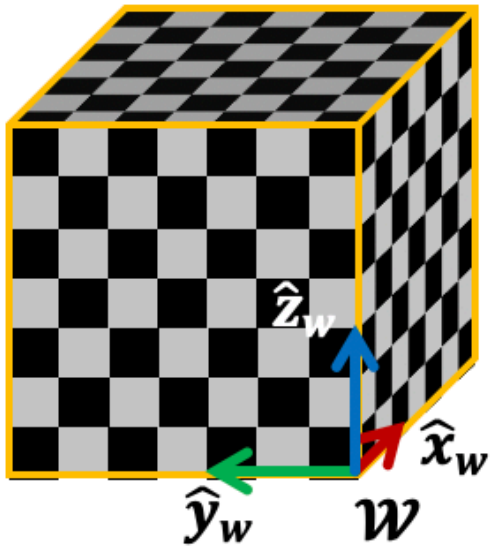
(1) Camera Calibration

Camera Calibration

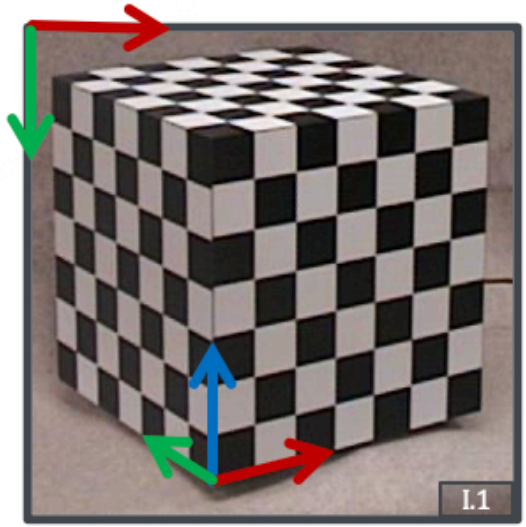
Most vision applications require the knowledge of intrinsic (f_x, f_y, o_x, o_y) and extrinsic $(R, \mathbf{t})$ parameters of the cameras being used.

We "**Calibrate**" the cameras to determine these.

Camera Calibration Procedure (1)



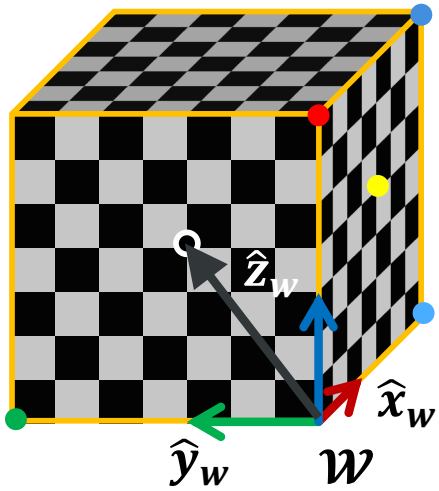
Object whose precise geometry is known

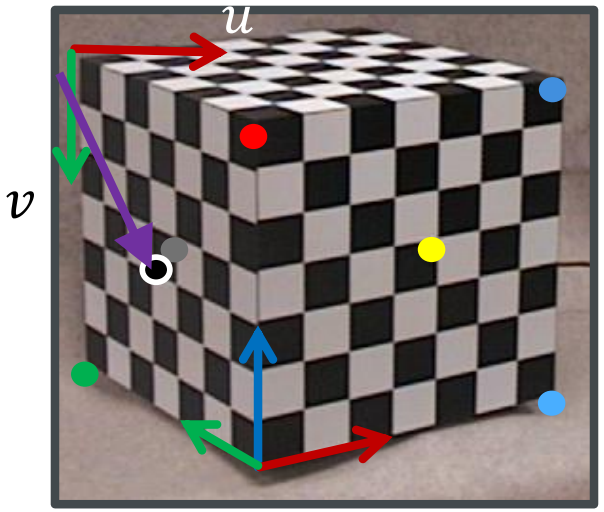


Captured Image

Camera Calibration Procedure (2)

Object whose precise geometry is known





Captured Image

- $\mathbf{x}_w = \begin{bmatrix} x_w \\ y_w \\ z_w \end{bmatrix} = \begin{bmatrix} 0 \\ 3 \\ 4 \end{bmatrix}$

- $\mathbf{u} = \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} 56 \\ 115 \end{bmatrix}$

Camera Calibration Procedure (3)

Step 3: For each corresponding point i in scene and image:

$$\underbrace{\begin{bmatrix} u^{(i)} \\ v^{(i)} \\ 1 \end{bmatrix}}_{\text{Known}} \equiv \underbrace{\begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix}}_{\text{Unknown}} \underbrace{\begin{bmatrix} x_w^{(i)} \\ y_w^{(i)} \\ z_w^{(i)} \\ 1 \end{bmatrix}}_{\text{Known}}$$

Expanding the matrix as linear equations:

$$u^{(i)} = \frac{p_{11}x_w^{(i)} + p_{12}y_w^{(i)} + p_{13}z_w^{(i)} + p_{14}}{p_{31}x_w^{(i)} + p_{32}y_w^{(i)} + p_{33}z_w^{(i)} + p_{34}}$$

$$v^{(i)} = \frac{p_{21}x_w^{(i)} + p_{22}y_w^{(i)} + p_{23}z_w^{(i)} + p_{24}}{p_{31}x_w^{(i)} + p_{32}y_w^{(i)} + p_{33}z_w^{(i)} + p_{34}}$$

Camera Calibration Procedure (4)

Step 4: Rearranging the terms:

$$\underbrace{\begin{bmatrix} x_w^{(1)} & y_w^{(1)} & z_w^{(1)} & 1 & 0 & 0 & 0 & 0 & -u_1 x_w^{(1)} & -u_1 y_w^{(1)} & -u_1 z_w^{(1)} & -u_1 \\ 0 & 0 & 0 & 0 & x_w^{(1)} & y_w^{(1)} & z_w^{(1)} & 1 & -v_1 x_w^{(1)} & -v_1 y_w^{(1)} & -v_1 z_w^{(1)} & -v_1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_w^{(i)} & y_w^{(i)} & z_w^{(i)} & 1 & 0 & 0 & 0 & 0 & -u_i x_w^{(i)} & -u_i y_w^{(i)} & -u_i z_w^{(i)} & -u_i \\ 0 & 0 & 0 & 0 & x_w^{(i)} & y_w^{(i)} & z_w^{(i)} & 1 & -v_i x_w^{(i)} & -v_i y_w^{(i)} & -v_i z_w^{(i)} & -v_i \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_w^{(n)} & y_w^{(n)} & z_w^{(n)} & 1 & 0 & 0 & 0 & 0 & -u_n x_w^{(n)} & -u_n y_w^{(n)} & -u_n z_w^{(n)} & -u_n \\ 0 & 0 & 0 & 0 & x_w^{(n)} & y_w^{(n)} & z_w^{(n)} & 1 & -v_n x_w^{(n)} & -v_n y_w^{(n)} & -v_n z_w^{(n)} & -v_n \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} p_{11} \\ p_{12} \\ p_{13} \\ p_{14} \\ p_{21} \\ p_{22} \\ p_{23} \\ p_{24} \\ p_{31} \\ p_{32} \\ p_{33} \\ p_{34} \end{bmatrix}}_{\mathbf{p}} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

A (Known)

$\mathbf{p}$ (Unknown)

Step 5: Solve for $\mathbf{p}$:

$$A\,\mathbf{p} = \mathbf{0}$$

Least Squares Solution for P

$$A\,\mathbf{p} = \mathbf{0}$$

If $\bar{\mathbf{p}}$ is a solution, so is $k\bar{\mathbf{p}}$ for any constant k.

But, Projection Matrix P needs to be determined only up to a scale factor. We can assume any scale for $\mathbf{p}$.

Set scale so that: $\|\mathbf{p}\|^2 = 1$

We want $A\mathbf{p}$ as close to 0 as possible and $\|\mathbf{p}\|^2 = 1$:

$$\min_{\mathbf{p}} \|A\mathbf{p}\|^2 \quad \text{such that } \|\mathbf{p}\|^2 = 1$$

See Appendix A

Find the eigenvector of A^tA with zero eigenvalue (or "SVD trick" to solve this)

Rearrange solution $\mathbf{p}$ to form the projection matrix P.

Extracting Intrinsic and Rotation Parameters

We know that:

$$P = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix} = \underbrace{\begin{bmatrix} f_x & s & o_x & 0 \\ 0 & f_y & o_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{\text{(Intrinsic)}} \underbrace{\begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{\text{(Extrinsic)}}$$

That is:

$$\begin{bmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ p_{31} & p_{32} & p_{33} \end{bmatrix} = \begin{bmatrix} f_x & 0 & o_x \\ 0 & f_y & o_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} = KR$$

Given that K is an Upper Right Triangle matrix and R is an Orthonormal matrix, it is possible to "decouple" K and R from their product using RQ factorization.

(See Appendix B)

Extracting Translation Parameters

We know that: $P = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix} = \underbrace{\begin{bmatrix} f_x & s & o_x & 0 \\ 0 & f_y & o_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{\text{(Intrinsic)}} \underbrace{\begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{\text{(Extrinsic)}}$

That is: $\begin{bmatrix} p_{14} \\ p_{24} \\ p_{34} \end{bmatrix} = \begin{bmatrix} f_x & s & o_x \\ 0 & f_y & o_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} = K\mathbf{t} \quad = -KR\mathbf{c}_w \qquad (\mathbf{t} = -R\mathbf{c}_w)$

Therefore:

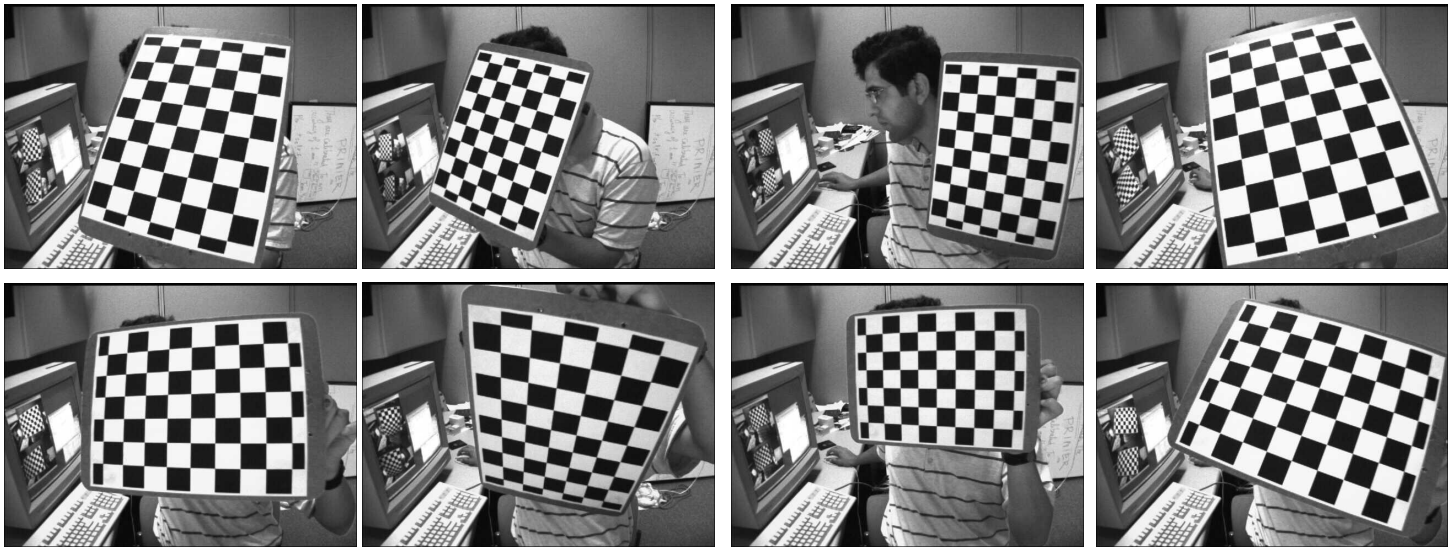
$$\mathbf{t} = K^{-1} \begin{bmatrix} p_{14} \\ p_{24} \\ p_{34} \end{bmatrix}$$

$$\mathbf{c}_w = -R^T K^{-1} \begin{bmatrix} p_{14} \\ p_{24} \\ p_{34} \end{bmatrix}$$

Camera Calibration (1)

So what's a practical procedure to calibrate a camera?
Turns out, we don't need a full 3D object – just planar ones(!)

First, we generally collect "checkerboard" images:



Camera Calibration (2)

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & s & o_x \\ 0 & f_y & o_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix} \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix}$$

Since all points lie in a plane: $z_w = 0$

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & s & o_x \\ 0 & f_y & o_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & \cancel{r_{13}} & t_x \\ r_{21} & r_{22} & \cancel{r_{23}} & t_y \\ r_{31} & r_{32} & \cancel{r_{33}} & t_z \end{bmatrix} \begin{bmatrix} x_w \\ y_w \\ \cancel{0} \\ 1 \end{bmatrix}$$

Thus, we can delete the 3rd column of the Extrinsic parameter matrix

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \underbrace{\begin{bmatrix} f_x & s & o_x \\ 0 & f_y & o_y \\ 0 & 0 & 1 \end{bmatrix}}_{K} \underbrace{\begin{bmatrix} r_{11} & r_{12} & t_x \\ r_{21} & r_{22} & t_y \\ r_{31} & r_{32} & t_z \end{bmatrix}}_{\mathbf{r_1}\ \mathbf{r_2}\ \mathbf{r_3}} \begin{bmatrix} x_w \\ y_w \\ 1 \end{bmatrix}$$

$$\mathbf{H} = [\mathbf{h_1}, \mathbf{h_2}, \mathbf{h_3}]$$

Camera Calibration (3)

$$\mathbf{h_1} = K\mathbf{r_1}, \mathbf{h_2} = K\mathbf{r_2} \quad \Rightarrow \quad \mathbf{r_1} = K^{-1}\mathbf{h_1}, \mathbf{r_2} = K^{-1}\mathbf{h_2}$$

$$\mathbf{r_1}^T\mathbf{r_1} = \mathbf{r_2}^T\mathbf{r_2} = \mathbf{1}, \mathbf{r_1}^T\mathbf{r_2} = \mathbf{0}$$

$$\begin{cases} \mathbf{h_1}^T K^{-T}K^{-1}\mathbf{h_1} - \mathbf{h_2}^T K^{-T}K^{-1}\mathbf{h_2} = \mathbf{0} \\ \mathbf{h_1}^T K^{-T}K^{-1}\mathbf{h_2} = \mathbf{0} \end{cases}$$

Define $B = K^{-T}K^{-1}$

Note that B is symmetric and positive definite

K can be calculated from B using **Cholesky factorization**

We now have a form that let's us solve **linearly** for B using the two equations from the homography that relates the images of the left and right cameras!

Camera Calibration (4)

$$\mathbf{h_1} = K\mathbf{r_1}, \mathbf{h_2} = K\mathbf{r_2} \quad \Rightarrow \quad \mathbf{r_1} = K^{-1}\mathbf{h_1}, \mathbf{r_2} = K^{-1}\mathbf{h_2}$$

$$\mathbf{r_1}^T\mathbf{r_1} = \mathbf{r_2}^T\mathbf{r_2} = \mathbf{1}, \mathbf{r_1}^T\mathbf{r_2} = \mathbf{0}$$

$$\begin{cases} \mathbf{h_1}^T K^{-T}K^{-1}\mathbf{h_1} - \mathbf{h_2}^T K^{-T}K^{-1}\mathbf{h_2} = \mathbf{0} \\ \mathbf{h_1}^T K^{-T}K^{-1}\mathbf{h_2} = \mathbf{0} \end{cases}$$

$B = K^{-T}K^{-1}$ is symmetric and positive definite

Each plane gives us two equations for B

Since B has 6 degrees of freedom (why?), we need at least 3 different homographies (i.e. 3 image pairs)

We need at least 4 points per plane to compute the homography

Camera Calibration (4)

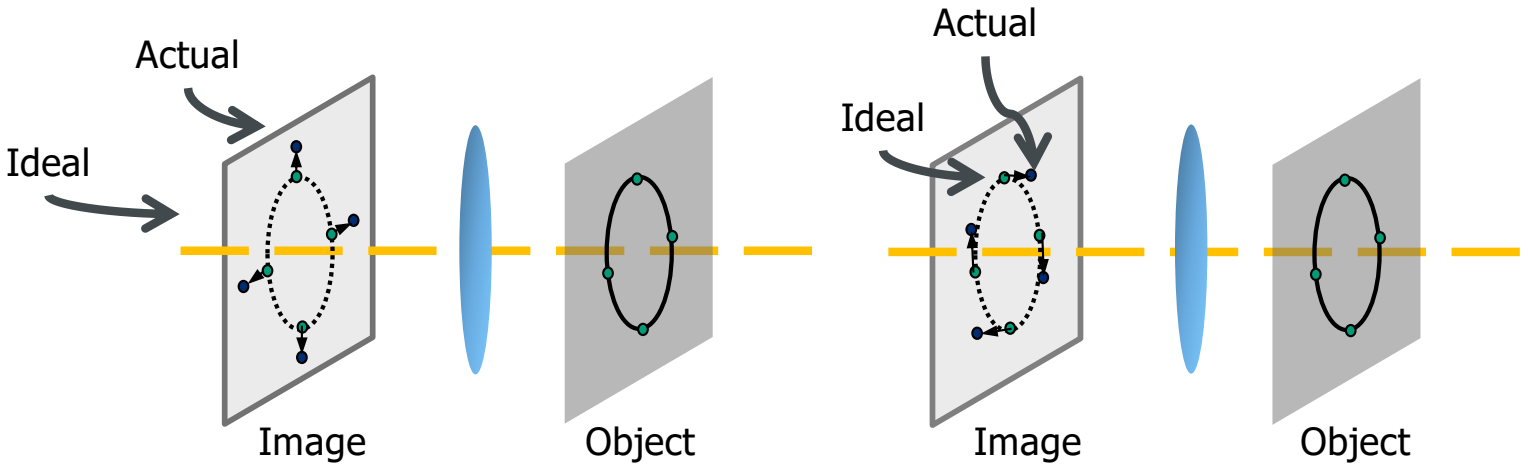
So what's the procedure to calibrate a camera?

Then, we:

- Extract corners in each checkerboard
- Associate each corner to a 3D location ON the checkerboard
- Collect across different views
- Solve for camera intrinsics: focal length (in pixels), principal point (in pixels), distortion parameters

Other Intrinsic Parameters: Distortion

Pinholes do not exhibit image distortions. Lenses do.



Radial distortion

Tangential distortion

The mathematical model of the camera will need to incorporate the distortion coefficients.

Distortion Parameters

How do we use the distortion parameters? Given radial distortion coefficients k_1, k_2, k_3 and tangential distortion coefficients p_1 and p_2:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = R \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + t$$

$$x' = \frac{x}{z} \qquad y' = \frac{y}{z}$$

where:

$$x'' = x'(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) + 2p_1 x' y' + p_2(r^2 + 2x'^2)$$
$$y'' = y'(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) + 2p_2 x' y' + p_1(r^2 + 2y'^2)$$
$$r^2 = x'^2 + y'^2$$

finally: $u = f_x x'' + c_x$ $\quad$ $v = f_y y'' + c_y$

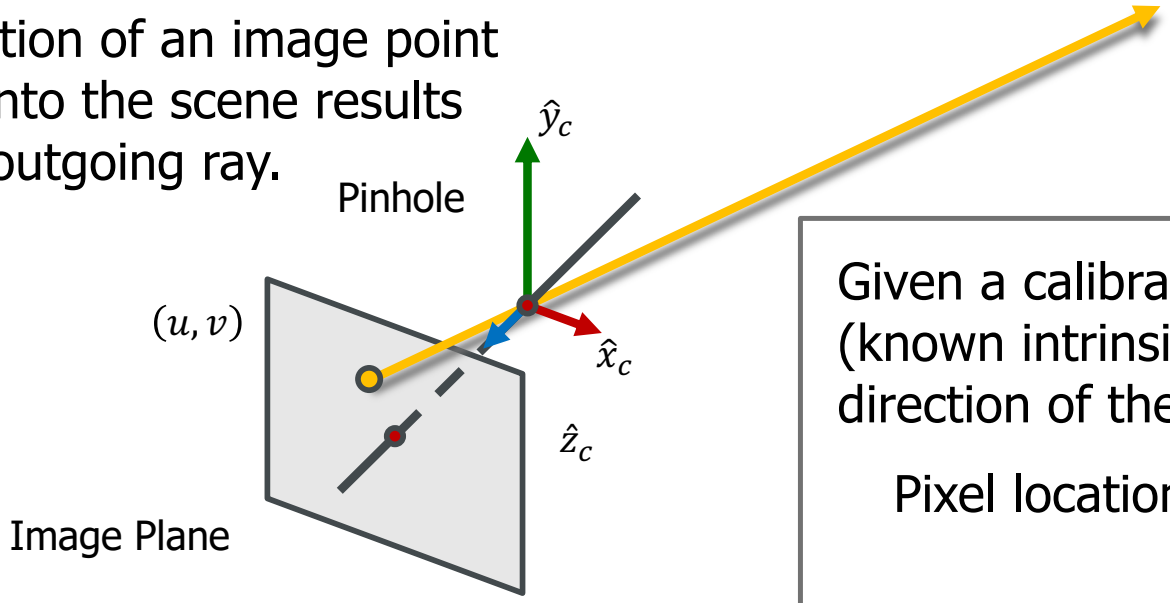
Non-linear Refinement

- 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.

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

Backward Projection: From 2D to 3D

Projection of an image point back into the scene results in an outgoing ray.



Given a calibrated camera (known intrinsics), what is the direction of the ray?

Pixel location: $u = m_x x_i + o_x$
$v = m_y y_i + o_y$

Image point: $\mathbf{x}_i = (x_i, y_i, f)$

Direction of ray: $\frac{-\mathbf{x}_i}{\|\mathbf{x}_i\|}$

Summary

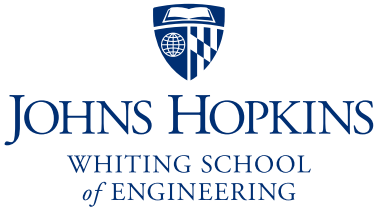
Camera Calibration and Photogrammetry: Method to find a camera's parameters and estimate 3D structure using two cameras.

- Essential concepts in this lecture:
  - Setting up calibration as a series of homographies
  - Extracting camera parameters
  - Including distortion and nonlinear refinement

Johns Hopkins Engineering

Computer Vision

Camera Calibration and Photogrammetry



Camera Calibration and Photogrammetry

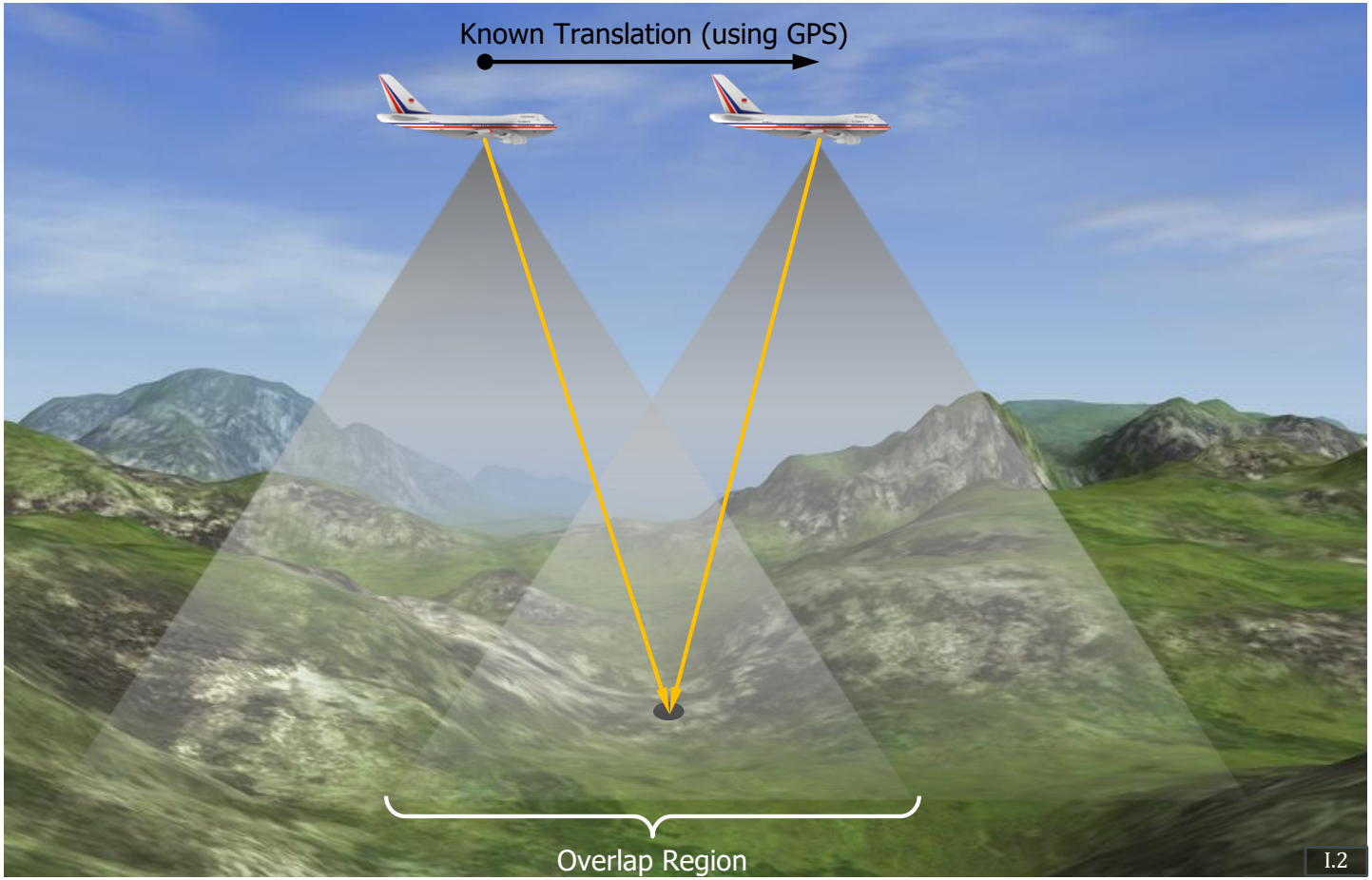
Method to find a camera's parameters and a method to estimate 3D structure using two cameras.

Topics:

(1) 3D reconstruction

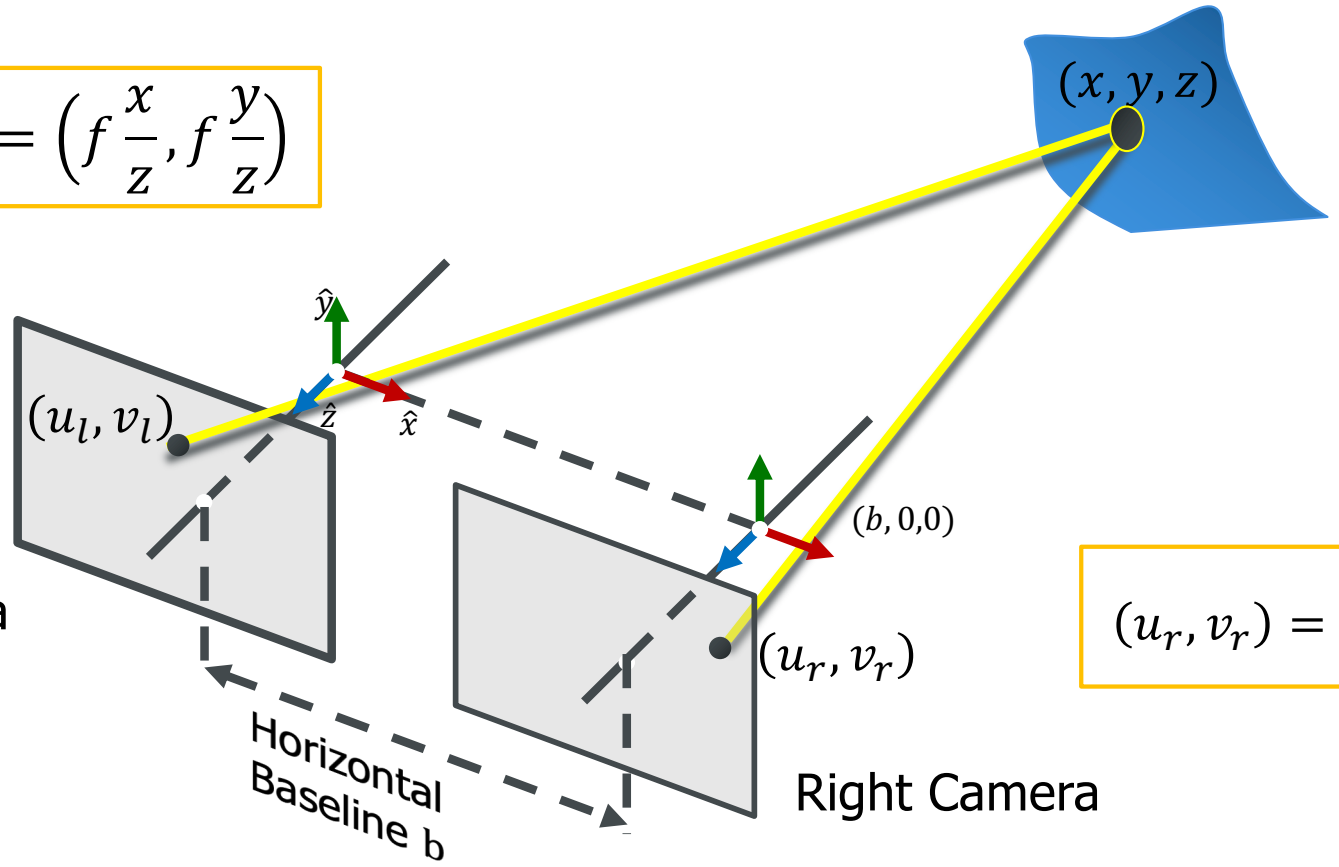
(2) Window-based disparity search

Aerial Photogrammetry



Simple Stereo

$$(u_l, v_l) = \left(f\frac{x}{z}, f\frac{y}{z}\right)$$



$$(u_r, v_r) = \left(f\frac{x-b}{z}, f\frac{y}{z}\right)$$

Simple Stereo: Depth and Disparity

For two pixels corresponding to the same scene point in the left and right images:

$$(u_l, v_l) = \left(f\frac{x}{z}, f\frac{y}{z}\right) \qquad (u_r, v_r) = \left(f\frac{x-b}{z}, f\frac{y}{z}\right)$$

Unknown:

Scene point position (x, y, z)

Known:

Pixel positions (u_l, v_l) and (u_r, v_r)

Baseline b from construction (or other calibration methods)

Focal length f from calibration

Simple Stereo: Depth and Disparity (cont.)

For two pixels corresponding to the same scene point in the left and right images:

$$(u_l, v_l) = \left(f\frac{x}{z}, f\frac{y}{z}\right) \qquad (u_r, v_r) = \left(f\frac{x-b}{z}, f\frac{y}{z}\right)$$

Solve for z: $\boxed{z = \frac{bf}{(u_l - u_r)}}$ $\qquad x = \frac{zu_l}{f}$, $\quad y = \frac{zv_l}{f}$

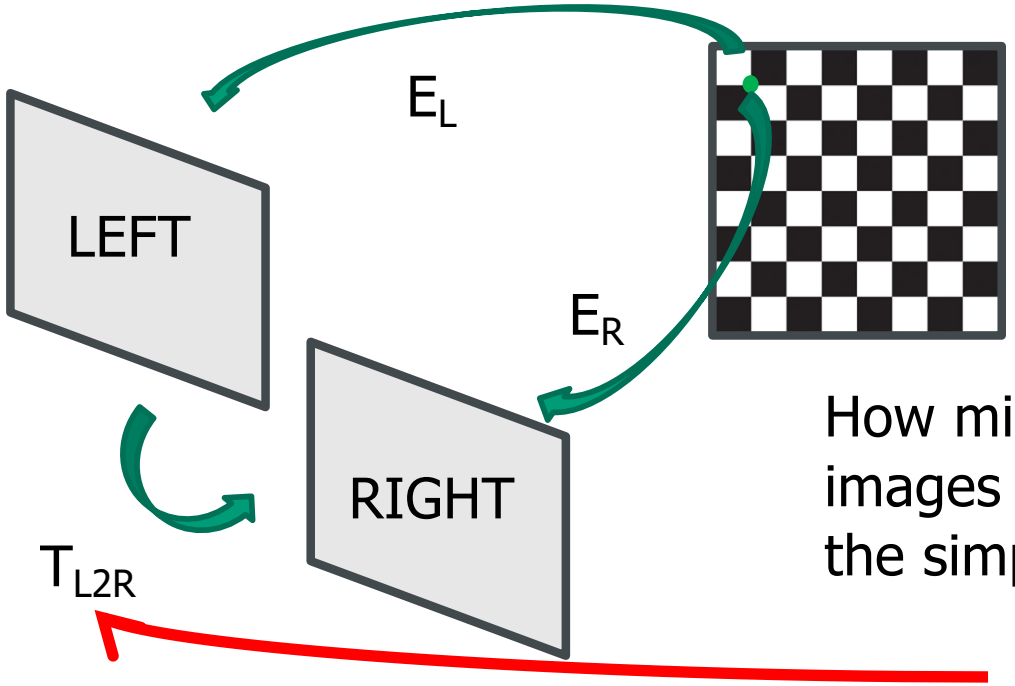
where $(u_l - u_r)$ is called the Disparity.

Given baseline b and focal length f, the scene depth for each pixel can be found by determining the disparity.

Depth is inversely proportional to Disparity.

More General Stereo

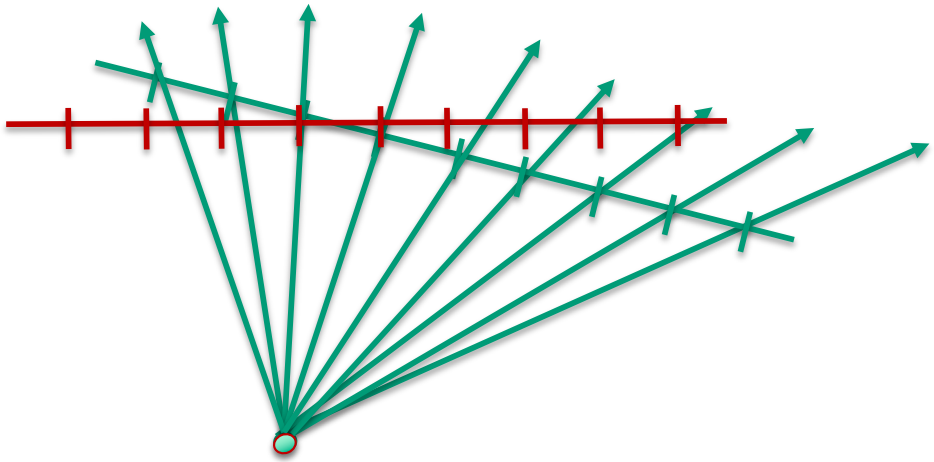
Stereo cameras differ from each other by a rigid body transform: a 3x3 rotation matrix and a 3x1 translation vector (e.g., the stereo extrinsics)



How might we change the images to make them like the simple, nonverged, case

More General Stereo

A simpler case – 2D rectification



$$R(\theta) = \begin{bmatrix} cos(\theta) & -sin(\theta) \\ sin(\theta) & cos(\theta) \end{bmatrix}$$

$$K_{old} = \begin{bmatrix} 10 & 4 \\ 0 & 1 \end{bmatrix}$$

$$p_{old} = \begin{bmatrix} 0 & 1 & 2 & 3 & 4 & \dots \\ 1 & 1 & 1 & 1 & 1 & \end{bmatrix}$$

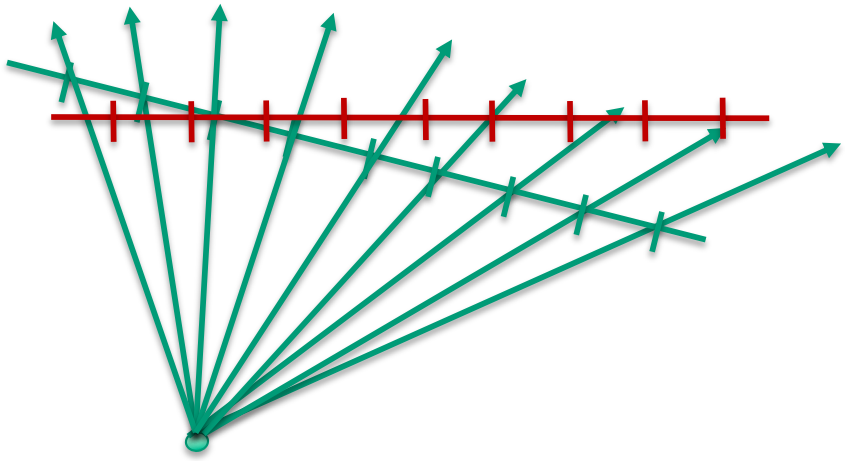
$$H_1 = K_{old} R(\theta) (K_{old})^{-1}$$

For $\theta = 0.6$, $p_{new} = (H_1)^{-1} p_{old}$ (sample backwards)

[-1.1,-2.1, -3.1, -4.1, -5.1, -6.1, -7.1, -8.1, -9.1]

More General Stereo

A simpler case – 2D rectification



$$R(\theta) = \begin{bmatrix} cos(\theta) & -sin(\theta) \\ sin(\theta) & cos(\theta) \end{bmatrix}$$

$$K_{old} = \begin{bmatrix} 10 & 4 \\ 0 & 1 \end{bmatrix}$$

$$p_{old} = \begin{bmatrix} 0 & 1 & 2 & 3 & 4 & \dots \\ 1 & 1 & 1 & 1 & 1 & \end{bmatrix}$$

$$K_{new} = \begin{bmatrix} 10 & 0 \\ 0 & 1 \end{bmatrix}$$

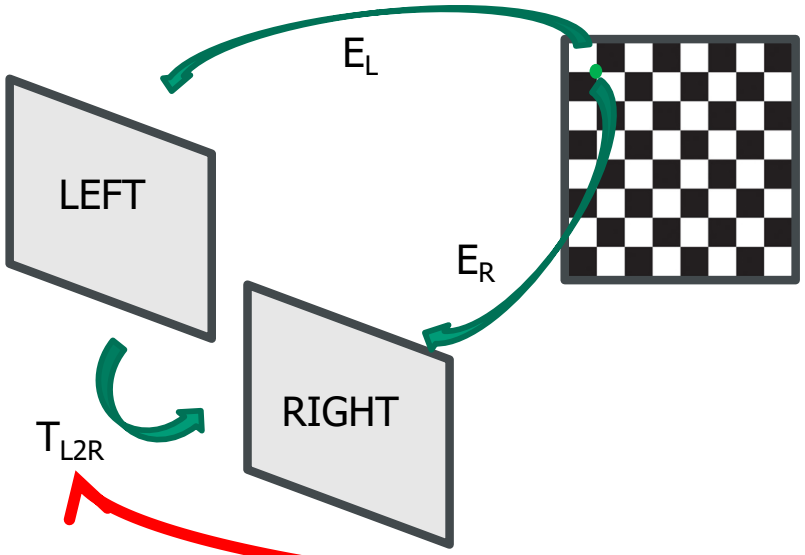
$$H_2 = K_{new} R(\theta) (K_{old})^{-1}$$

For $\theta = 0.6$, $p_{new} = (H_2)^{-1} p_{old}$

[0,1, 2, 3, 4, 5, 6, 7, 9]

More General Stereo

Note there is always a pair of rotations that will align the camera image planes and the image rows (!)



(One) Solution:

Make x axis align to baseline t_{l2r}

Make y= $\frac{y}{||y'||} where\ y' = (0,0,1) \times x$

Compute z = x x y

Pick scales as average of image scales

Adjust image center to account for resampling

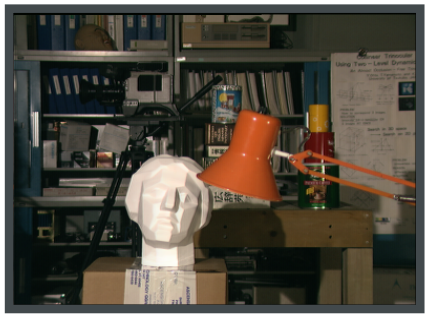
$R = [x \quad y \quad z]^T$

Resample camera 1 by $H_1 = K_{new}\ R\ (K_{old})^{-1}$

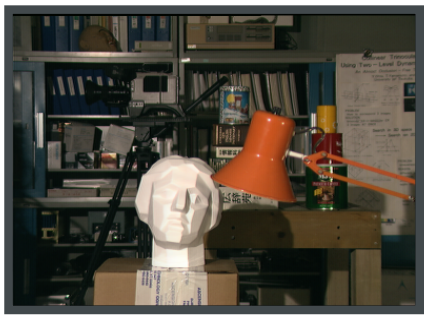
Resample camera 2 by $H_2 = K_{new}\ R\ R_{R2L}\ (K_{old})^{-1}$

Finding Correspondences

Goal: Find the disparity between left and right stereo pairs.



Left Image



Right Image

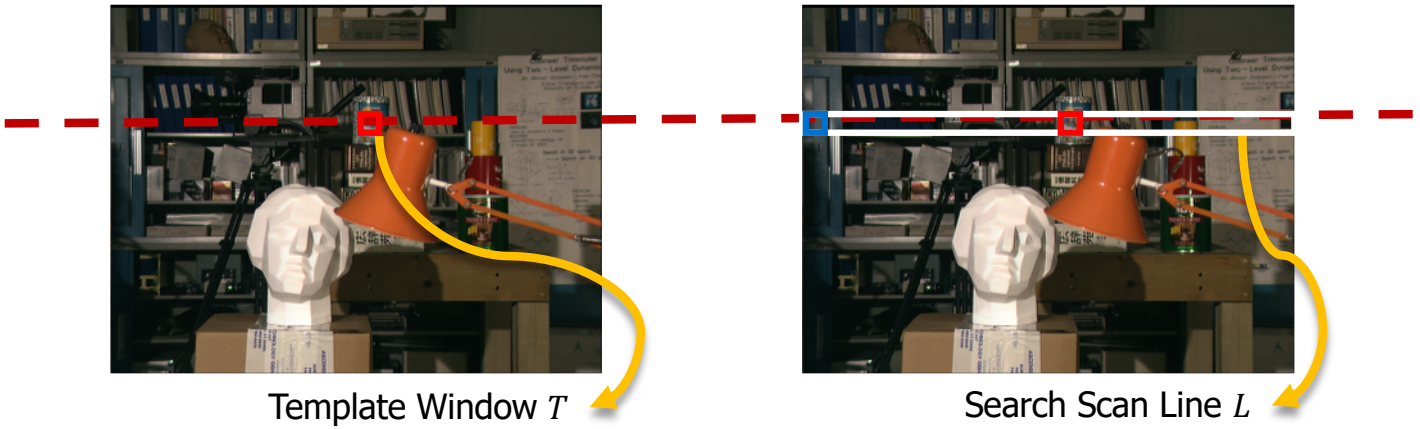
Disparity

From perspective projection: $v_l = v_r = f\frac{y}{z}$

Corresponding scene points lie on the same horizontal scan line.

Window based Methods

Determine Disparity using **Template Matching**



Left Camera Image E_l Right Camera Image E_r

Disparity, $d = u_l - u_r$

Similarity Metrics for Template Matching

Find pixel $(k, l) \in L$ with Minimum Sum of Absolute Differences:

$$SAD(k,l) = \sum_{(i,j)\in T} |E_l(i,j) - E_r(i+k, j+l)|$$

Find pixel $(k, l) \in L$ with Minimum Sum of Squared Differences:

$$SSD(k,l) = \sum_{(i,j)\in T} |E_l(i,j) - E_r(i+k, j+l)|^2$$

Find pixel $(k, l) \in L$ with Minimum Normalized Cross-Correlation:

$$NCC(k,l) = \frac{\sum_{(i,j)\in T} |E_l(i,j) - E_r(i+k, j+l)|^2}{\sqrt{\sum_{(i,j)\in T} |E_l(i,j)|^2 \sum_{(i,j)\in T} |E_r(i+k, j+l)|^2}}$$

(See Image Processing Lecture 1)

Issues with Window Based Methods

How large should the window be?

Window size = 5 pixels

Window size = 30 pixels

Adaptive Window Method Solution: For each point, match using windows of multiple sizes and use the disparity that is a result of the best similarity measure.

Implementing Region Based Matching

- **Deciding the Range**: The first step in correspondence search is to compute the range of disparities to search
  - Horopter

- Assume a non-verged (rectified) system. Therefore, we have
  - $d = f\ b/z$
  - given a range z_{min} to z_{max}, we calculate
    - $d_{min} = f\ b/z_{max}$
    - $d_{max} = f\ b/z_{min}$

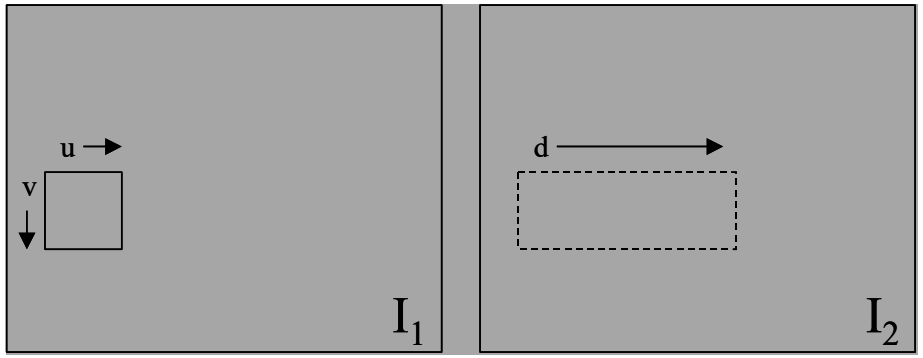
- Thus, for each point u_l in the left image, we will search points u_l+d_{min} to u_l+d_{max} in the right.

- Note we can turn this around and start at a point u_r and search from u_r-d_{max} to u_r-d_{min}

Region Based Approach Algorithm

- For each pixel (i,j) of the left image and offset o in disparity range
  - Compute $d(i,j,o) = \sum_{k,l} \psi(I_l(i+k,j+l),I_r(i+k,j+l+o))$
  - The disparity at (i,j) is the value o that minimizes d
- This can be visualized as a *volume* of image location and disparity
- The result of performing this search over every pixel is the *disparity map.*

Correspondence Search Algorithm (simple version for CC)

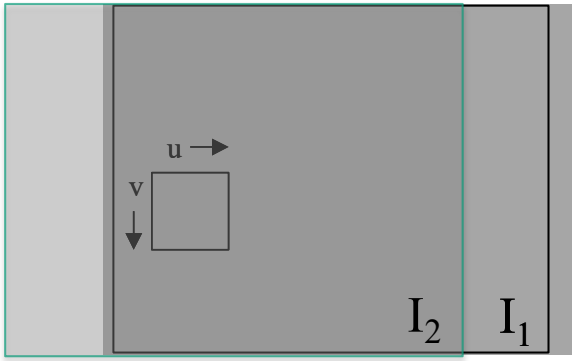



```
For  i = 1:nrows
  for j=1:ncols
        best(i,j) = -1
        for k = mindisparity:maxdisparity
          c = CC(I1(i,j),I2(i,j+k),winsize)
          if (c > best(i,j))
                best(i,j) = c
                disparities(i,j) = k
            end
         end
  end
end
                O(nrows * ncols * disparities * winx * winy)
```

Correspondence Search Algorithm (efficient version for CC)

```
best = -ones(size(im))
disp = zeros(size(im))
for k = mindisparity:maxdisparity
        prod = I1(:,overlap) .* I2(:,k+overlap)
        CC = conv2(prod,fspecial('average',winsize))
        better = CC > best;
        disp = better .* k + (1-better).*disp;
        best = better .*CC + (1-better).*best;
end
```

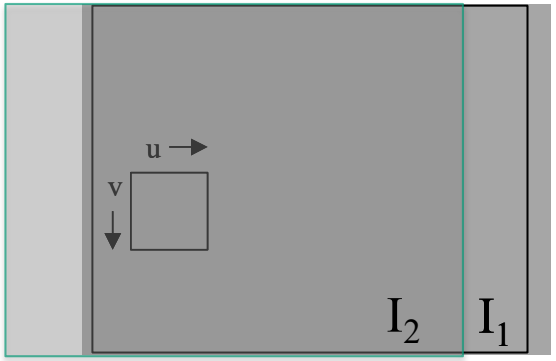
O(disparities * nrows * ncols)



Correspondence Search Algorithm (efficient version for CC)

```
best = -ones(size(im))
disp = zeros(size(im))
for k = mindisparity:maxdisparity
        prod = I1(:,overlap) .* I2(:,k+overlap)
        CC = conv2(prod,fspecial('average',winsize))
        better = CC > best;
        disp = better .* k + (1-better).*disp;
        best = better .*CC + (1-better).*best;
end
```

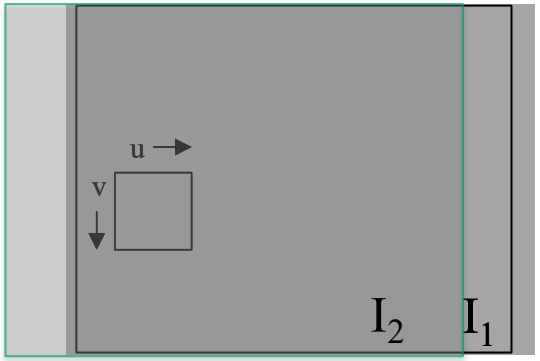
O(disparities * nrows * ncols)



Correspondence Search Algorithm (efficient version for CC)

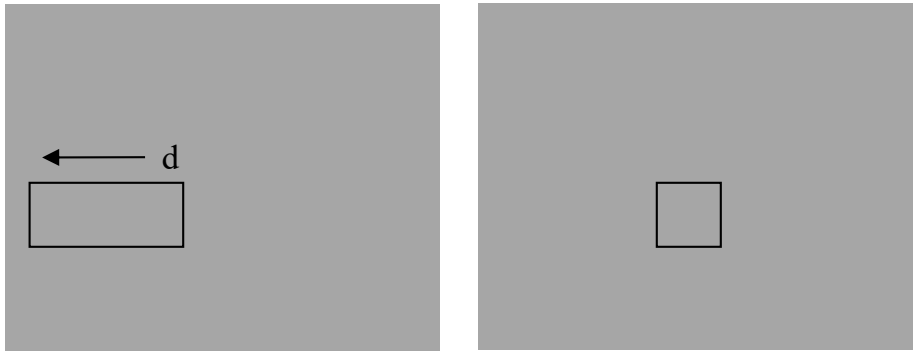
```
best = -ones(size(im))
disp = zeros(size(im))
for k = mindisparity:maxdisparity
        prod = I1(:,overlap) .* I2(:,k+overlap)
        CC = conv2(prod,fspecial('average',winsize))
        better = CC > best;
        disp = better .* k + (1-better).*disp;
        best = better .*CC + (1-better).*best;
end
```

O(disparities * nrows * ncols)



An Additional Twist

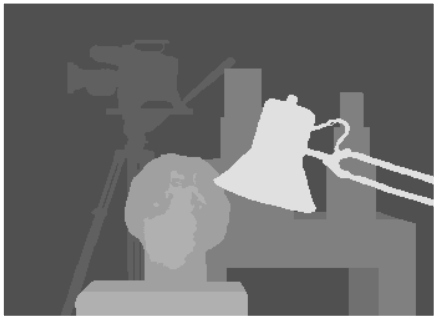
- Note that searching from left to right *is not the same* as searching from right to left.
- As a result, we can obtain a somewhat independent disparity map by flipping the images around.
- The results should be the same map up to sign.
- LRCheck: $disp_{lr}(i,j) = - disp_{rl}(i,j+disp_{lr}(i,j))$



Stereo Matching Results

Left Image

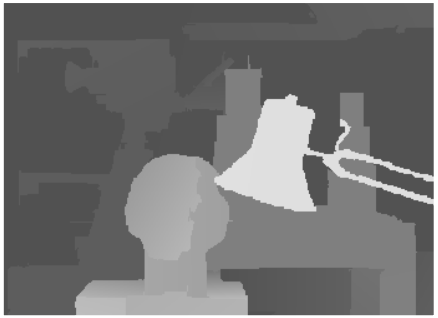
Right Image



Ground Truth

SSD (Window size=21)

SSD – Adaptive Window



State of the Art

Summary

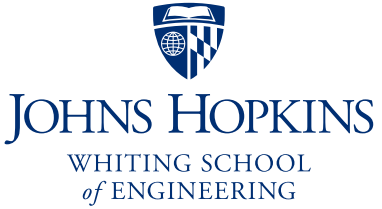
Camera Calibration and Photogrammetry: Method to find a camera's parameters and estimate 3D structure using two cameras.

- Essential concepts in this lecture:
  - The geometry of reconstruction
  - Non-verged stereo matching
  - Rectification to create a non-verged system
  - Similarity measures for matching

Johns Hopkins Engineering

Computer Vision

Camera Calibration and Photogrammetry



Camera Calibration and Photogrammetry

Method to find a camera's parameters and a method to estimate 3D structure using two cameras.

Topics:

(1)Advanced disparity search

Challenges of Correspondence

- Photometric issues
  - Specularities
  - Strongly non-Lambertian BRDF's
- Surface structure
  - Lack of texture
  - Repeating texture within horopter bracket
- Geometric ambiguities
  - As surfaces turn away, difficult to get accurate reconstruction (affine approximation can help)
  - At the occluding contour, likelihood of good match but incorrect reconstruction
- Consistency
  - Scanline Consistency
  - Global Optimization

Global Optimization Methods

- Assume that the surfaces are smooth
- Can pose the problem of finding the corresponding points as an energy (or cost) minimization:

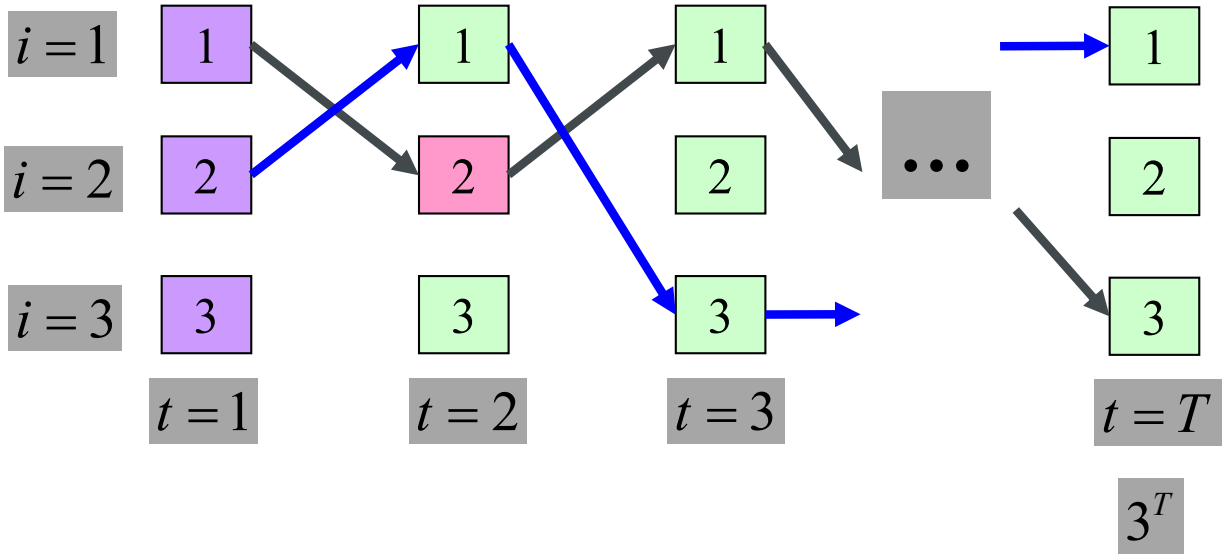
$$E([d_1 \dots d_n]) = \Sigma_i M(I_L(x), I_R(x+d)) + \Sigma_{j \in N(x)} R(d_i, d_j)$$

Pixel matching cost — *neighboring pixels have similar disparities*

- However, this is NP-hard
- Instead use approximations

Dynamic Programming

- Efficient algorithm for solving sequential decision (optimal path) problems – a form of **global optimization.**



Note this works line by line, so we still have artifacts

(Rehg)

Graph Cuts: Solving an MRF for stereo

Disparity estimate at each pixel

Local evidence for each disparity based on intensity match to neighboring frame

$$P(x_1,x_2,\ldots,x_N,y_1,y_2,\ldots,y_N) = \prod_{(i,j)} \Psi(x_i,x_j) \prod_{p} \Phi(x_p,y_p) \quad (1)$$

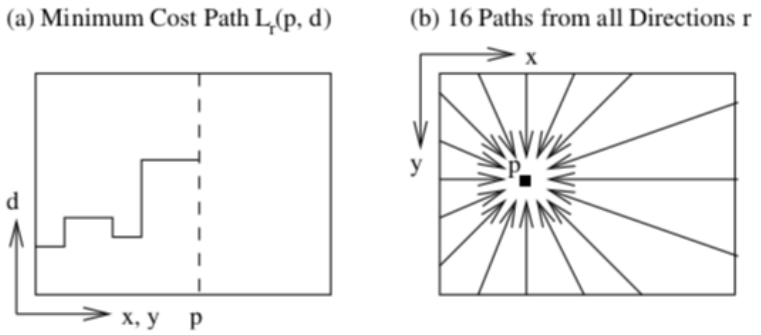
where N is the number of nodes, (i,j) represent a pair of neighboring nodes, x_n is the variable at location n, and y_n is the variable representing the intensity differences. The y

Graph cuts

- Algorithm: uses node label swaps or expansions as moves in the algorithm to reduce the energy. Swaps many labels at once, not just one at a time, as with ICM.
- Find which pixel labels to swap using min cut/max flow algorithms from network theory.
- Can offer bounds on optimality.
- See Boykov, Veksler, Zabih, IEEE PAMI 23 (11) Nov. 2001 (available on web).

Semi-Global Matching

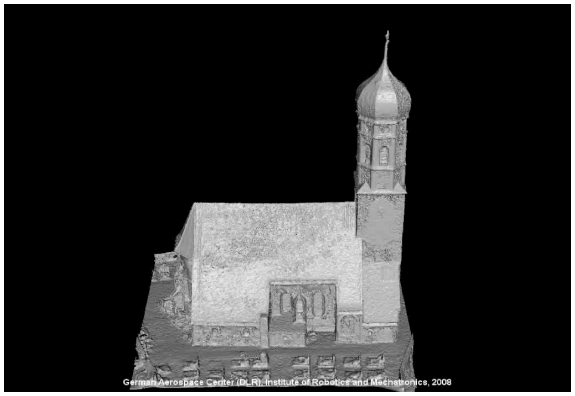
- Compute many paths and record optimal cost
- Choose match that agrees over the most paths
- Pretty good heuristic approximation to global optimization, but fast enough to run in real time



Accurate and efficient stereo processing by semi-global matching and mutual information. H Hirschmuller , TPAMI 2005

Stereo processing by semiglobal matching and mutual information. H Hirschmuller , TPAMI 2007

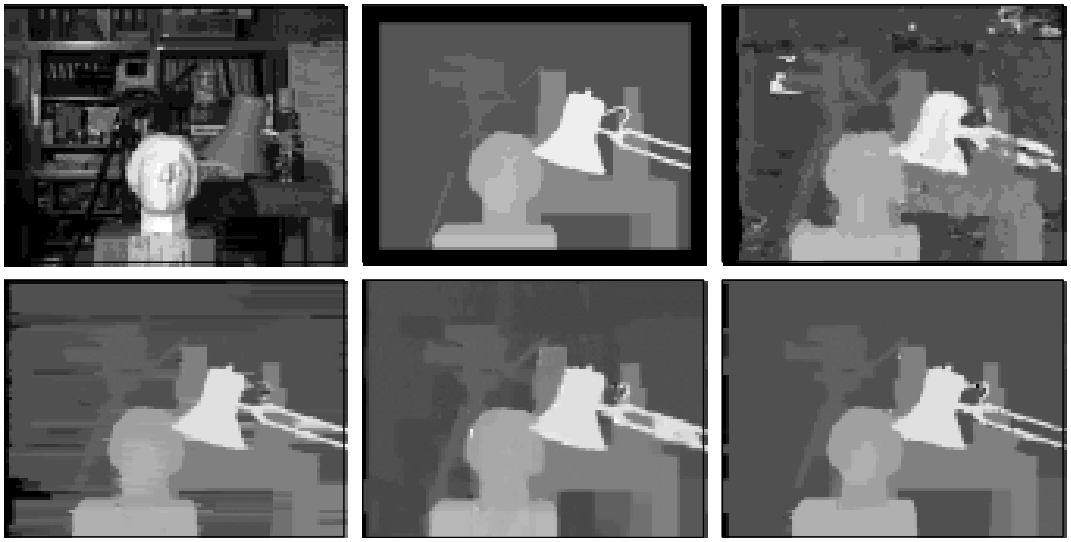
Semi-Global Matching Results



Accurate and efficient stereo processing by **semi**-**global** matching and mutual information.
H **Hirschmuller** , PAMI 2005

Local vs. Global Matching

Comparative results on images from the University of Tsukuba, provided by Scharstein and Szeliski [69]. Left to right: left stereo image, ground truth, Muhlmann et al.'s area correlation algorithm [57], dynamic programming (similar to Intille and Bobick [36]), Roy and Cox's maximum flow [65] and Komolgorov and Zabih's graph cuts [45].



State of the Art Algorithms

Middlebury Stereo Datasets

2001 datasets - 6 datasets of piecewise planar scenes [1]
(Sawtooth, Venus, Bull, Poster, Barn1, Barn2)

2003 datasets - 2 datasets with ground truth obtained using structured light [2]
(Cones, Teddy)

2005 datasets - 9 datasets obtained using the technique of [2], published in [3, 4]
(Art, Books, Dolls, Laundry, Moebius, Reindeer, Computer, Drumsticks, Dwarves)

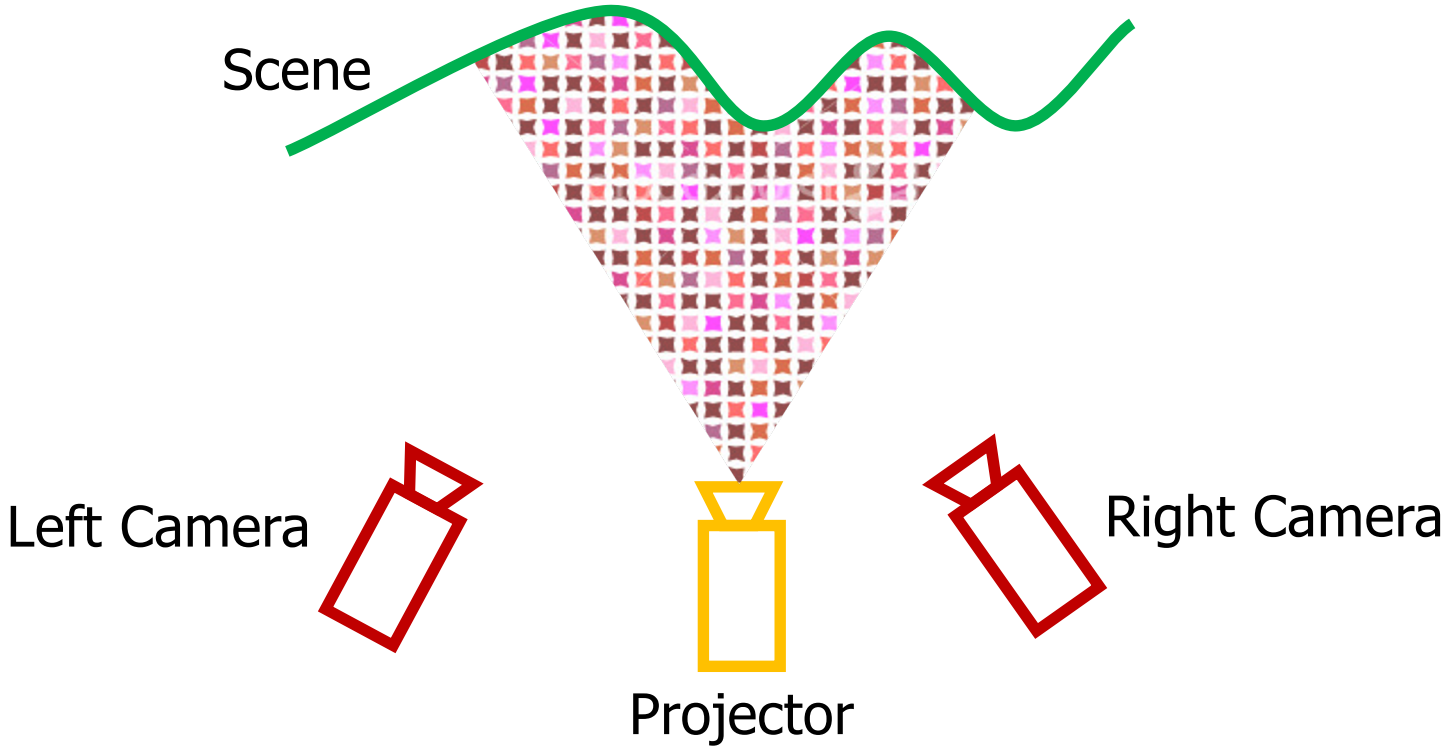
2006 datasets - 21 datasets obtained using the technique of [2], published in [3, 4]
(Aloe, Baby1-3, Bowling1-2, Cloth1-4, Flowerpots, Lampshade1-2, Midd1-2, Monopoly, Plastic, Rocks1-2, Wood1-2)

2014 datasets - 33 datasets obtained using the technique of [5]

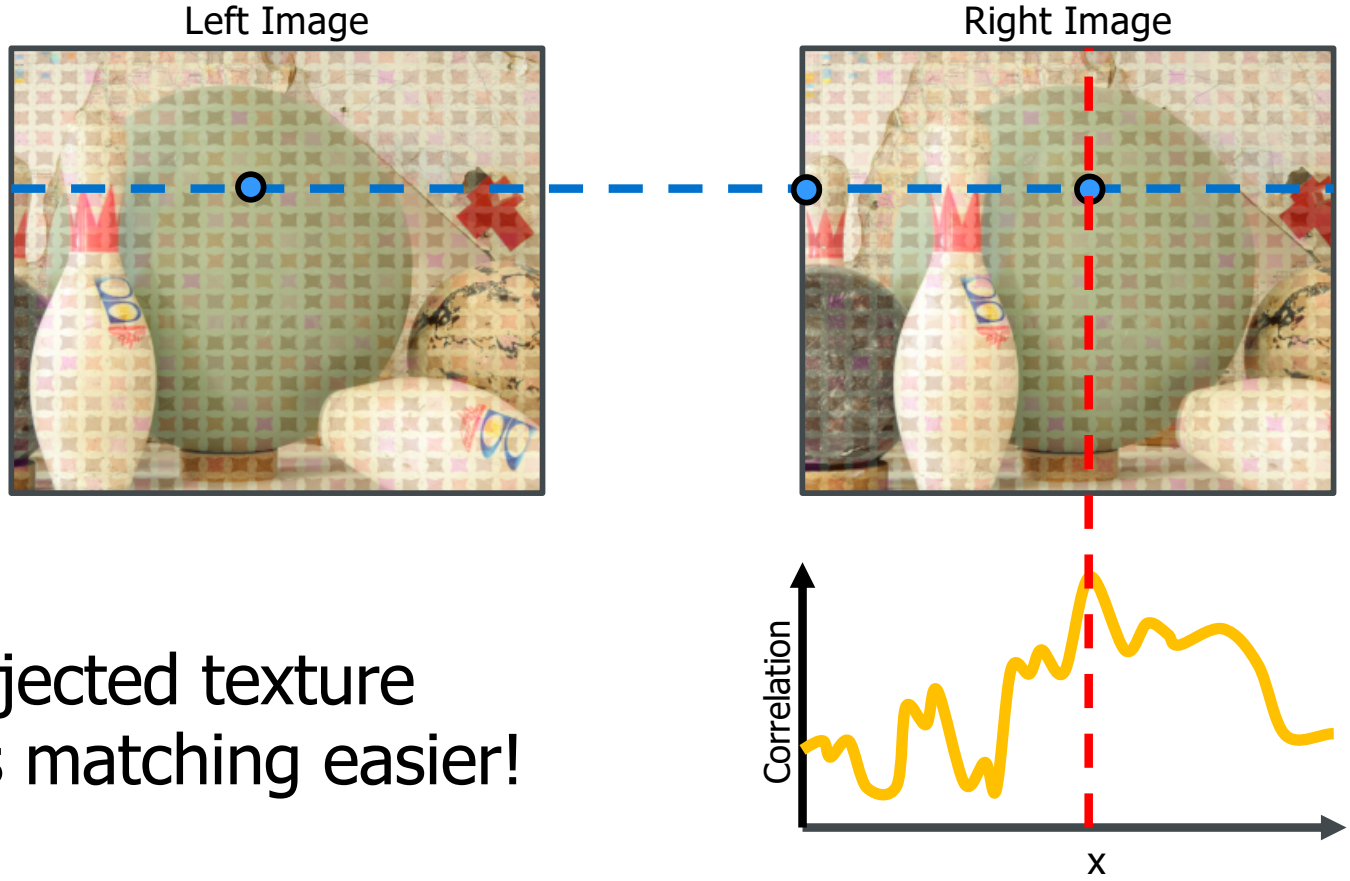
Current Leader:
Jie Li, Penglei Ji, and Xinguo Liu. Superpixel alpha-expansion and normal adjustment for stereo matching. Proceeding of CAD/Graphics 2019

https://vision.middlebury.edu/stereo/

Binocular Stereo… and a Projector

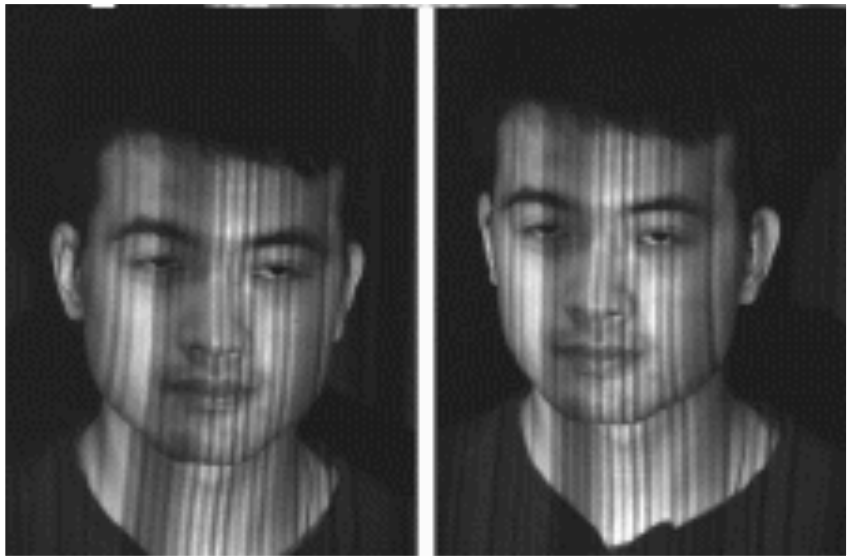


Active Stereo



Projected texture
makes matching easier!

Active Stereo Results

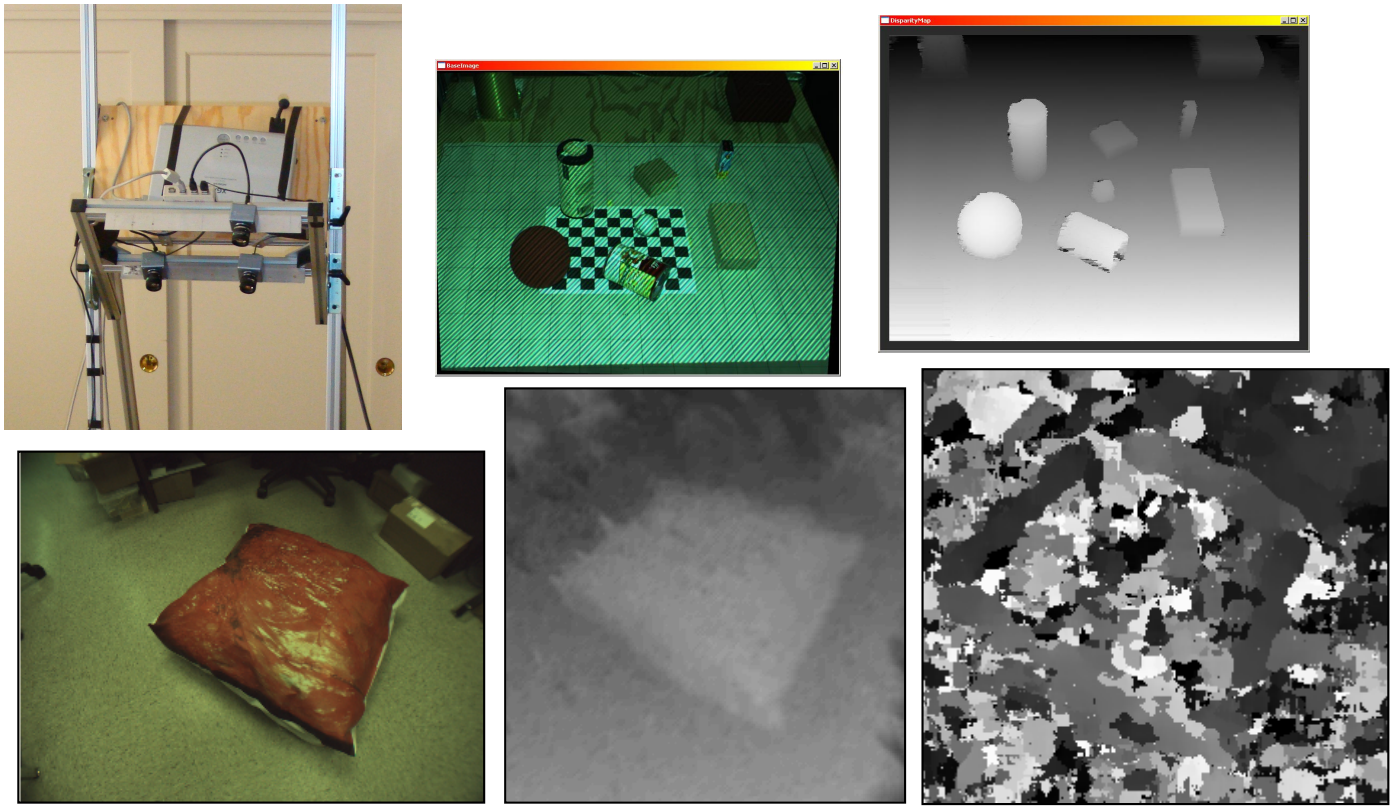


Left Image

Right Image

3D Structure

Trinocular Structured Light Stereo w DP



Hager, Gregory D., and Ben Wegbreit. "Scene parsing using a prior world model." *The International Journal of Robotics Research* 30, no. 12 (2011): 1477-1507.

Summary

Camera Calibration and Photogrammetry: Method to find a camera's parameters and estimate 3D structure using two cameras.

- Essential concepts in this lecture:
  - Global optimization for stereo
  - Some popular approaches
  - Evaluation frameworks
  - Structured light stereo

Code Packages for Calibration

Two Most Popular

OpenCV (C++, Python): http://opencv.willowgarage.com/wiki/

CalTech Camera Calibration Toolbox (MATLAB):
http://www.vision.caltech.edu/bouguetj/calib_doc/

Appendix A: Least Squares Solution for P

$$\min_{\mathbf{p}} \|A\mathbf{p}\|^2 \text{ such that } \|\mathbf{p}\|^2 = 1$$

We know that:

$$\|A\mathbf{p}\|^2 = (A\mathbf{p})^T(A\mathbf{p}) = \mathbf{p}^T A^T A\mathbf{p} \quad \text{and} \quad \|\mathbf{p}\|^2 = \mathbf{p}^T\mathbf{p} = 1$$

Create a Loss function $L(\mathbf{p})$ and find $\mathbf{p}$ that minimizes it.

$$\min_{\mathbf{p}}\{L(\mathbf{p}) = \mathbf{p}^T A^T A\mathbf{p} + \lambda(\mathbf{p}^T\mathbf{p} - 1)\}$$

Taking derivatives w.r.t $\mathbf{p}$ and λ: $A^T A\mathbf{p} + \lambda\mathbf{p} = 0$ Eigenvalue Problem

Clearly, eigenvector $\mathbf{p}$ with smallest eigenvalue λ of matrix $A^T A$ minimizes the loss function $L(\mathbf{p})$.

Appendix B: RQ Matrix Factorization

Definition: Any Invertible Matrix A can be uniquely decomposed as a product of Upper Right Triangle matrix R and Orthonormal matrix Q.

$$A = RQ$$

Note that according to our camera model (projection matrix) notation:

$$\underbrace{\begin{bmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ p_{31} & p_{32} & p_{33} \end{bmatrix}}_{P_{3\times3}} = \underbrace{\begin{bmatrix} f_x & s & o_x \\ 0 & f_y & o_y \\ 0 & 0 & 1 \end{bmatrix}}_{K} \underbrace{\begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}}_{R}$$

where K is the upper right triangle matrix and R is the orthonormal matrix.

Appendix B: RQ Matrix Factorization (cont.)

Step 1:

Let $A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} = \begin{bmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ p_{31} & p_{32} & p_{33} \end{bmatrix}$

Step 2:

Let $R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c_x & s_x \\ 0 & -s_x & c_x \end{bmatrix}$ where $c_x = \frac{a_{33}}{\sqrt{{a_{32}}^2 + {a_{33}}^2}}$ $s_x = \frac{a_{32}}{\sqrt{{a_{32}}^2 + {a_{33}}^2}}$

Set $A = AR_x$ (This makes $a_{32} = 0$)

Appendix B: RQ Matrix Factorization

Step 3:

Let $R_y = \begin{bmatrix} c_y & 0 & -s_y \\ 0 & 1 & 0 \\ s_y & 0 & c_y \end{bmatrix}$ where $c_y = \frac{a_{33}}{\sqrt{{a_{31}}^2 + {a_{33}}^2}}$ $s_y = \frac{-a_{31}}{\sqrt{{a_{31}}^2 + {a_{33}}^2}}$

Set $A = AR_y$ (This makes $a_{31} = 0$)

Step 4:

Let $R_z = \begin{bmatrix} c_z & s_z & 0 \\ -s_z & c_z & 0 \\ 0 & 0 & 1 \end{bmatrix}$ where $c_z = \frac{a_{22}}{\sqrt{{a_{21}}^2 + {a_{22}}^2}}$ $s_z = \frac{a_{21}}{\sqrt{{a_{21}}^2 + {a_{22}}^2}}$

Set $K = AR_z$ (This makes $a_{21} = 0$)

Step 5: $R = R_z^T R_y^T R_x^T$

References: Textbooks

Robot Vision (Chapter 13)
Horn, B. K. P., MIT Press

Computer Vision: A Modern Approach (Chapter 10)
Forsyth, D and Ponce, J., Prentice Hall

Multiple View Geometry (Chapters 8-10)
Hartley, R. and Zisserman, A., Cambridge University Press

Computer Vision: Algorithms and Applications (Chapter 7)
Szeliski, R., Springer

An introduction to 3D Computer Vision (Chapter 3)
Cyganek, B., Siebert, J. P., Wiley Pub

Image Credits

- I.1 http://www.cs.rochester.edu/~sanders/cali_box/box_offlights.jpg
- I.2 http://www.orderlymayhem.com/images/terrain_scene2.jpg
- I.3 http://vision.middlebury.edu/stereo/data/scenes2001/data/imagehtml/tsukuba.html

References

[Carrihill 1985] B. Carrihill and R. Hummel. *Experiments with the intensity ratio depth sensor.* Computer Vision, Graphics and Image Processing, vol.32, 1985, pp.337-358.

[Caspi 1998] D. Caspi, N. Kiryati, and J. Shamir. Range imaging with adaptive color structured light. IEEE Trans. on PAMI, 20(5), 1998, pp.470-480.

[Inokuchi 1984] S. Inokuchi, K. Sato and F. Matsuda. Range imaging system for 3-D object recognition. ICPR, 1984, pp.806-808.

[Levoy 2000] M. Levoy et al., The Digital Michelangelo Project: 3D Scanning of Large Statues. Siggraph, 2000.

[Posdamer 1981] J. L. Posdamer and M. D. Altschuler. Surface measurement by space-encoded projected beam systems. Computer Graphics and Image Processing, 18(1), pp.1-17. 1981.

[Salvi 2004] J. Salvi, J. Pages, and J. Batlle. *Pattern codification strategies in structured light systems.* Pattern Recognition, vol.37, no.4, 2004, pp.827-849.

[Wust 1991] C. Wust and D. W. Capson. Surface profile measurement using color fringe projection. Machine Vision and Applications, vol.4, 1991, pp.193-203.

Image Credits (cont.)

- I.1 http://vision.middlebury.edu/stereo/data/scenes2006/
- I.2 http://grail.cs.washington.edu/projects/ststereo/
- I.3 http://jordipages.webs.com/codedlight/examples/examples.html
- I.4 http://jordipages.webs.com/codedlight/examples/examples.html
- I.5 http://graphics.stanford.edu/projects/mich/
- I.6 http://graphics.stanford.edu/papers/marble-assessment/laser-striking-marble.jpg
- I.7 http://www.cs.columbia.edu/CAVE/projects/struc_light/
- I.8 http://www.artisanti.com/ekmps/shops/artisanti/images/electro-plate-buddha-head-10876-p.jpg
- I.9 http://www.flickr.com/photos/mattbell/1591769889/
- I.10 http://www.flipwallpapers.com/wallpapers/the_curly_hair_girl-2560x1600.jpg

JOHNS HOPKINS

WHITING SCHOOL
of ENGINEERING