Camera Calibration

Shree K. Nayar

Monograph: FPCV-4-1

Module: Reconstruction II

Series: First Principles of Computer Vision

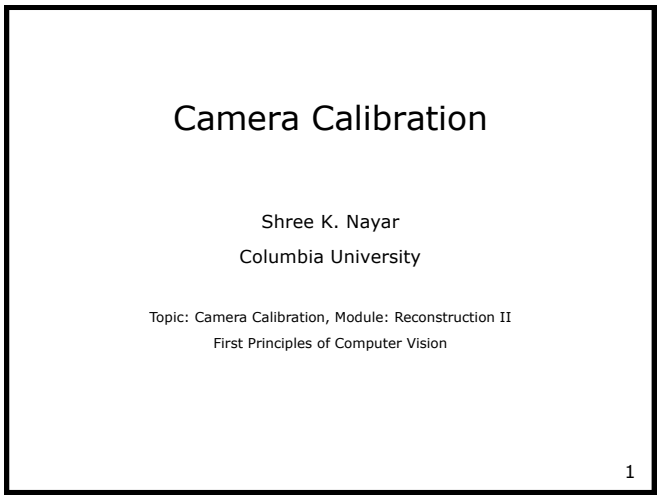
Computer Science, Columbia University

April, 2025

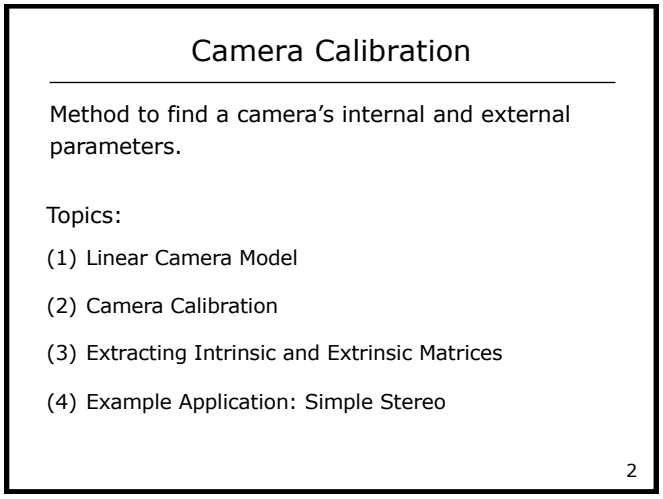
FPCV Channel

FPCV Website

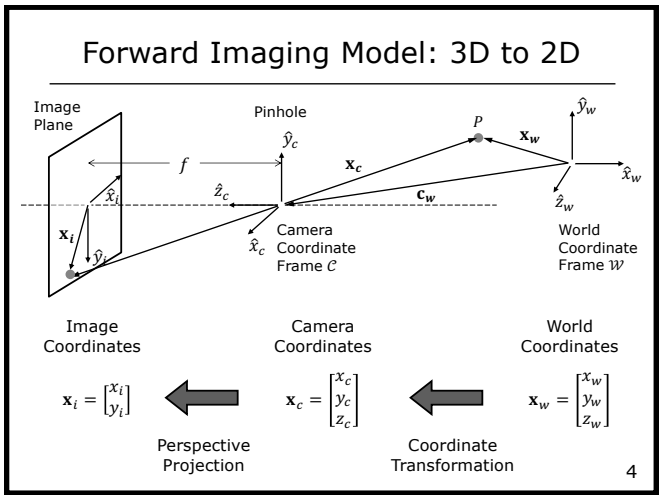
A key problem in computer vision is recovering the 3D structure of a scene from its images. For an intelligent system, such as a robot, to navigate around, or interact with, a scene, we need scene reconstruction to be metric, i.e., we want the measured coordinates of each scene point to be in physical units (e.g., in millimeters) in a fixed world coordinate frame. However, in a camera image, the projection of each scene point is measured in terms of pixels. To go from 2D images to a full metric 3D scene reconstruction, we need to know the position and orientation of the camera with respect the world coordinate frame (i.e., the camera's external parameters), and we need to know how the camera projects points in the scene to its image plane (i.e., the camera's internal parameters). The process of finding the external and internal parameters of a camera is called camera calibration.



We are going to develop a calibration method for estimating the camera's internal and external parameters. To achieve that, first, we will develop a camera model, called the forward imaging model, which maps the 3D coordinates of a scene point to pixels in the image. We show that this camera model is linear in that it is defined by a single matrix, called the projection matrix. Then, we will develop a camera calibration method that can estimate all the elements of the projection matrix, from a single image of an object with known dimensions. Interestingly, from the projection matrix, we can extract both the internal and external parameters of the camera, which are given by the intrinsic and extrinsic matrices. Finally, we will show how to use two identical calibrated cameras, displaced with respect to each other along one of the two image coordinates, to reconstruct a 3D scene. We refer to scene recovery using such a two-camera system as simple stereo.



Let us now develop a comprehensive model of the camera. We will describe the forward imaging model, which takes us from the 3D metric coordinates of a scene point in a world coordinate frame to its 2D image coordinates.



Shown here is a single point P in the world coordinate frame $\mathcal{W}$. In this coordinate frame, we have a camera defined by its own coordinate frame $\mathcal{C}$, where the z-axis of $\mathcal{C}$ is aligned with the camera's optical axis. The effective focal length, which is the distance between the effective center of projection of the camera and the image plane of the camera, is denoted by f. If we know the position and orientation of the camera frame $\mathcal{C}$ with respect to the world frame $\mathcal{W}$, then we can map a point P to its image coordinates. To do this, we first map the 3D world coordinates $\mathbf{x}_w$ of P to its 3D camera coordinates $\mathbf{x}_c$. Then, we use perspective projection to map $\mathbf{x}_c$ to the 2D image coordinates $\mathbf{x}_i$. This complete mapping from 3D world coordinates to 2D image coordinates is called the forward imaging model. We will use it to develop a comprehensive linear model for the camera.

We first discuss the perspective projection component of the forward image model. Shown here are the perspective projection equations we are familiar with. Consider the image coordinates $\mathbf{x}_i$ of the point P with coordinates $\mathbf{x}_c$ in the camera frame. We have x_i divided by f is equal to x_c divided by z_c, and y_i divided by f is equal to y_c divided by z_c. We can rewrite these to get the expressions for x_i and y_i, which are the coordinates of the projection of P onto the image plane.

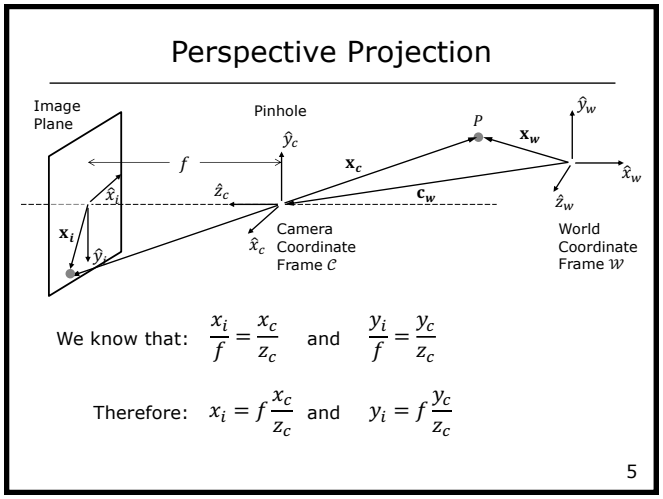
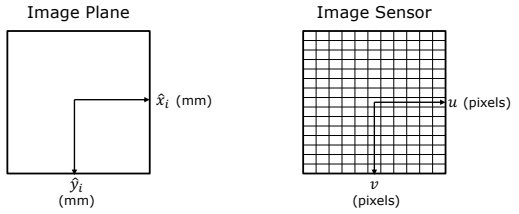


Image Plane to Image Sensor Mapping



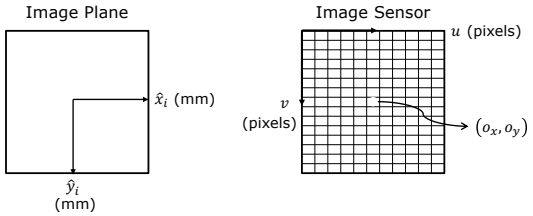
Pixels may be rectangular.

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

$$u = m_x x_i = m_x f \frac{x_c}{z_c} \quad \boxed{1} \qquad v = m_y y_i = m_y f \frac{y_c}{z_c} \quad \boxed{2}$$

6

Image Plane to Image Sensor Mapping



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 Principal Point where the optical axis pierces the sensor, then:

$$u = m_x f \frac{x_c}{z_c} + o_x \quad \boxed{3} \qquad v = m_y f \frac{y_c}{z_c} + o_y \quad \boxed{4}$$

7

Thus far, we have assumed that the image plane coordinates are expressed in terms of physical units (e.g., millimeters), which are the same units used to define the scene point P in the camera coordinate frame. In reality, however, the image sensor measures a projected point's coordinates in pixels. Thus, we need to account for the mapping from the physical coordinates x_i and y_i, to pixel coordinates u and v. We define m_x and m_y to be the pixel densities (pixels per millimeter) in the x and y directions, respectively. Since, in general, pixels don't have to be square (they could be rectangular), m_x and m_y do not have to be equal. The pixel coordinates u and v are then given by equations $\boxed{1}$ and $\boxed{2}$.

The point where the optical axis of the camera pierces the image plane is called the principal point. We will assume that we do not know the coordinates (o_x, o_y) of the principal point. We can therefore update our expressions for u and v to account for the principal point as shown in equations $\boxed{3}$ and $\boxed{4}$.

Our model for perspective projection is given by the two equations shown at the top. Since the pixel densities m_x and m_y and the effective focal length f are all unknown, we can combine them into two parameters: f_x and f_y. These parameters can be seen as effective focal lengths in the x and y direction, respectively. Therefore, the perspective projection model of the camera has four unknowns, (f_x, f_y, o_x, o_y), which are called the intrinsic parameters of the camera. Note that this projection model is non-linear, since z_c appears in the denominator. We would like a linear model to make the estimation of the intrinsic parameters easier.

Perspective Projection

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

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

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.

Equations for perspective projection are Non-Linear. It is convenient to express them as linear equations.

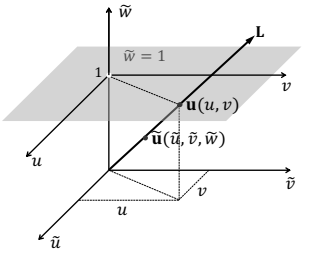
8

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 $\mathbf{u}(u, v)$

REVIEW 9

Homogenous Coordinates

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

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

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

10

As we have seen before, we can use homogenous coordinates to turn a non-linear model to a linear one. Recall that the homogeneous representation of a 2D point $\boldsymbol{u}$ is a 3D point $\tilde{\boldsymbol{u}}$ where the third coordinate $\tilde{w}$ is fictitious—it is used for scaling the other coordinates. As shown on the left, when expressed in their homogeneous coordinates, both $\boldsymbol{u}$ and $\tilde{\boldsymbol{u}}$ are equivalent. Using the same approach, as shown on the right, a 3D point $\mathbf{x}$ can be represented as a 4D point $\tilde{\boldsymbol{x}}$.

Let us now return to our perspective projection model. We can express the pixel coordinates u and v using the homogenous coordinates $(u, v, 1)$, which is equivalent to $(u, v, 1)$ multiplied by any constant. If $(u, v, 1)$ is multiplied by z_c, we can express the result as the product of a 3×4 matrix, which includes all the internal parameters of the camera (f_x, f_y, o_x, o_y), and the homogeneous coordinates $(x_c, y_c, z_c, 1)$ of the scene point in the camera coordinate frame. We now have a linear model for perspective projection.

Perspective Projection

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

Linear Model for Perspective Projection

11

The above 3×4 matrix is called the intrinsic matrix M_{int}. We call the left 3×3 submatrix of M_{int} the calibration matrix K, which is an upper right triangular matrix (the elements below the diagonal are zero). The intrinsic matrix M_{int} is a concatenation of K with a column of zeros. To summarize, M_{int} maps a scene point $\tilde{\boldsymbol{x}}_c$ to its image projection $\tilde{\boldsymbol{u}}$.

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:

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

Upper Right Triangular Matrix

Intrinsic Matrix:

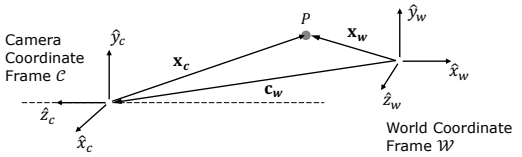
$$M_{int} = [K|0] = \begin{bmatrix} f_x & 0 & o_x & 0 \\ 0 & f_y & o_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

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

12

We now have a model that maps a scene point in the camera frame to the image. Since our scene points are defined in a world frame, we need a mapping (3D to 3D) from world coordinates to camera coordinates. This mapping is defined by the position and orientation of the camera frame $\mathcal{C}$ with respect to the world frame $\mathcal{W}$. Let us denote the camera's position by the vector $\boldsymbol{c}_w$, and its orientation by a 3×3 rotation matrix R. The rows of R correspond to the directions of $\hat{x}_c$, $\hat{y}_c$, and $\hat{z}_c$ in the world frame. The rotation matrix R, by definition, is an orthonormal matrix. Let us now take a look at the properties of an orthonormal matrix.

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.

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$$

- Row 1: Direction of $\hat{x}_c$ in world coordinate frame
- Row 2: Direction of $\hat{y}_c$ in world coordinate frame
- Row 3: Direction of $\hat{z}_c$ in world coordinate frame

Orientation/Rotation Matrix R is Orthonormal

14

Two vectors $\mathbf{u}$ and $\mathbf{v}$ are orthonormal if and only if the dot product of the two vectors is equal to zero (i.e., $\mathbf{u}^T\mathbf{v} = 0$) and both are unit vectors. An orthonormal matrix is a square matrix where the row vectors (or column vectors) are orthonormal. Our rotation matrix R is therefore an orthonormal matrix. Any orthonormal matrix has the property that its inverse is equal to its transpose.

Orthonormal Vectors and Matrices

Orthonormal Vectors: Two vectors $\mathbf{u}$ and $\mathbf{v}$ are orthonormal 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:

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

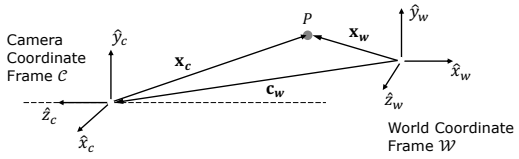
A Rotation Matrix is an Orthonormal Matrix

MATH PRIMER

15

We can now model the transformation from world coordinates to camera coordinates using the rotation matrix R and the camera position vector c_w. In the world frame, the vector $\mathbf{x}_c$ can be expressed as $\mathbf{x}_w - \mathbf{c}_w$. Now, $\mathbf{x}_c$ can be found in the camera frame by multiplying $\mathbf{x}_w - \mathbf{c}_w$ by the rotation matrix R. Therefore, $\mathbf{x}_c$ equals the rotation matrix R times $\mathbf{x}_w$ plus a vector $\mathbf{t}$, which is called the translation vector. In matrix form, (x_c, y_c, z_c) in the camera frame equals the rotation matrix R times the world coordinates (x_w, y_w, z_w) plus the translation vector (t_x, t_y, t_z).

World-to-Camera Transformation



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:

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

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

16

We would like to represent the above mapping using a single matrix. To achieve this, we can rewrite our world-to-camera mapping using homogenous coordinates, as shown here. Now, we have a mapping from the homogeneous coordinates $(x_w, y_w, z_w, 1)$ of $\mathbf{x}_w$ to the homogeneous coordinates $(x_c, y_c, z_c, 1)$ of $\mathbf{x}_c$ through a single 4×4 matrix $\mathrm{M}_{\mathrm{ext}}$ which includes both the rotation matrix R and the translation vector $\mathbf{t}$. $\mathrm{M}_{\mathrm{ext}}$ is called the extrinsic matrix.

Extrinsic Matrix

Rewriting using homogenous coordinates:

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

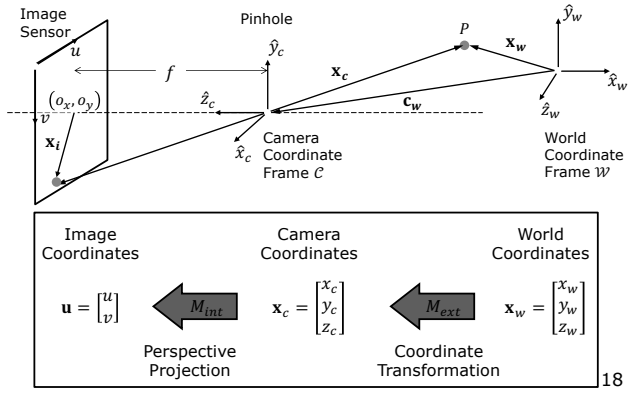
Extrinsic Matrix: $M_{ext} = \begin{bmatrix} R_{3\times3} & \mathbf{t} \\ \mathbf{0}_{1\times3} & 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$

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

17

Now, we have all the components necessary to model the mapping of a point in the world frame to pixels in the image.

Forward Imaging Model: 3D to 2D



18

The extrinsic matrix M_{ext} maps a world point to the camera frame, and the intrinsic matrix M_{int} maps the point in the camera frame to pixel coordinates in the image. We can multiply these two matrices to get a single 3×4 matrix, called the projection matrix P, which directly maps a point in the world frame to its pixel coordinates. To calibrate the camera, we need to find the projection matrix *P*.

Projection Matrix P

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:

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

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

19

Camera Calibration

Shree K. Nayar

Columbia University

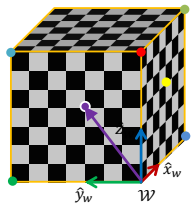
Topic: Camera Calibration, Module: Reconstruction II

First Principles of Computer Vision

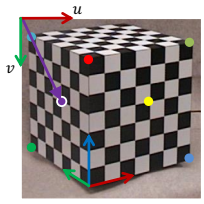
20

Camera Calibration Procedure

Step 2: Identify correspondences between 3D scene points and image points.



Object of Known Geometry



Captured Image

- $\mathbf{x}_w = \begin{bmatrix} x_w \\ y_w \\ z_w \end{bmatrix} = \begin{bmatrix} 0 \\ 3 \\ 4 \end{bmatrix}$ (inches)
- $\mathbf{u} = \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} 56 \\ 115 \end{bmatrix}$ (pixels)

21

We can estimate the projection matrix *P* of a camera by using an object of known geometry, such as the cube shown on the right. Assume that we know the dimensions of the cube and the locations of all the features (say, the corners of the checkerboard pattern). First, we choose our world frame to be aligned with one of the corners of the cube.

We capture a single image of this cube. Consider a single point on the cube, with world coordinates $(x_w, y_w, z_w) = (0, 3, 4)$ inches. In the image, this point has the 2D coordinates $(u, v) = (56, 115)$ pixels. The correspondences between points in the world and in the image can be established either manually, or automatically by using an algorithm that can uniquely identify scene points in the image based on their features and/or locations within a known spatial configuration. In the end, we get a set of correspondences between 3D scene points and their 2D image coordinates.

For each corresponding pair i, we can multiply our unknown projection matrix P with the known scene point to get the known image point. Upon expanding this matrix form, we get the two equations at the bottom—one for u and one for v.

Camera Calibration Procedure

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:

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

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

22

We can rewrite the above equations for u and v for the entire set of corresponding pairs, as shown here. The elements of the matrix A are functions of all the known coordinates, and the vector $\mathbf{p}$ has the 12 unknown elements of the projection matrix P. Our goal is to solve for the vector $\mathbf{p}$.

Camera Calibration Procedure

Step 4: Rearranging the terms

$$\underbrace{\begin{bmatrix} x_w^{(1)} & y_w^{(1)} & z_w^{(1)} & 1 & 0 & 0 & 0 & 0 & -u_1x_w^{(1)} & -u_1y_w^{(1)} & -u_1z_w^{(1)} & -u_1 \\ 0 & 0 & 0 & 0 & x_w^{(1)} & y_w^{(1)} & z_w^{(1)} & 1 & -v_1x_w^{(1)} & -v_1y_w^{(1)} & -v_1z_w^{(1)} & -v_1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_w^{(i)} & y_w^{(i)} & z_w^{(i)} & 1 & 0 & 0 & 0 & 0 & -u_ix_w^{(i)} & -u_iy_w^{(i)} & -u_iz_w^{(i)} & -u_i \\ 0 & 0 & 0 & 0 & x_w^{(i)} & y_w^{(i)} & z_w^{(i)} & 1 & -v_ix_w^{(i)} & -v_iy_w^{(i)} & -v_iz_w^{(i)} & -v_i \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_w^{(n)} & y_w^{(n)} & z_w^{(n)} & 1 & 0 & 0 & 0 & 0 & -u_nx_w^{(n)} & -u_ny_w^{(n)} & -u_nz_w^{(n)} & -u_n \\ 0 & 0 & 0 & 0 & x_w^{(n)} & y_w^{(n)} & z_w^{(n)} & 1 & -v_nx_w^{(n)} & -v_ny_w^{(n)} & -v_nz_w^{(n)} & -v_n \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} p_{11} \\ p_{12} \\ p_{13} \\ p_{14} \\ p_{21} \\ p_{22} \\ p_{23} \\ p_{24} \\ p_{31} \\ p_{32} \\ p_{33} \\ p_{34} \end{bmatrix}}_{\mathbf{p}} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

A: Known; $\mathbf{p}$: Unknown

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

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

23

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:

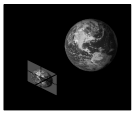
$$\begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix} \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix} \equiv k \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} &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

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

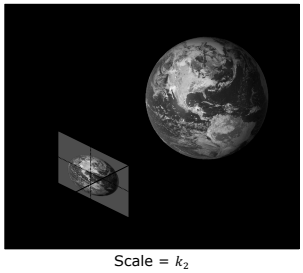
Projection Matrix P is defined only up to a scale.

24

Scale of Projection Matrix



Scale = k_1



Scale = k_2

Scaling projection matrix, implies simultaneously scaling the world and camera, which does not change the image.

Set projection matrix to some arbitrary scale!

25

Before solving for the vector $\mathbf{p}$, let us discuss an important property of the projection matrix P. Recall that homogenous coordinates are scale invariant, meaning that multiplying a homogenous coordinate by a non-zero factor k results in an equivalent coordinate. Therefore, the projection matrices P and kP produce the same homogenous pixel coordinates. In other words, the projection matrix is only defined up to a scale factor.

What does this scale ambiguity mean in terms of imaging? Imagine first taking an image of a scene. Now, if we double the size of the scene and the camera, the resulting image would be identical to the first one. Therefore, we can take the liberty of arbitrarily fixing the scale of the projection matrix.

How do we fix the scale of the projection matrix? One way is to set one of the 12 elements of $\mathbf{p}$ to be equal to 1. Or, we could just set the square of the magnitude of the vector $\mathbf{p}$ equal to one, which is what we do. Thus, we want $A\mathbf{p} = 0$ and $\|\mathbf{p}\|^2 = 1$. In other words, we want to find the $\mathbf{p}$ that minimizes $\|A\mathbf{p}\|^2$ such that $\|\mathbf{p}\|^2 = 1$. This is the classical constrained least squares problem of the type we solved to estimate the homography matrix in the lecture on image stitching. To solve for $\mathbf{p}$, we define the loss function L shown here.

Least Squares Solution for P

Option 1: Set scale so that: $p_{34} = 1$

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

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

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

$$\min_{\mathbf{p}} (\mathbf{p}^T A^T A \mathbf{p}) \text{ such that } \mathbf{p}^T\mathbf{p} = 1$$

Define Loss function $L(\mathbf{p}, \lambda)$:

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

(Similar to Solving Homography in Image Stitching)

26

To find the $\mathbf{p}$ that minimizes L, we compute the derivative of L with respect to $\mathbf{p}$ and set it equal to zero [1]. Finding the $\mathbf{p}$ that minimizes L is therefore equivalent to solving the eigenvalue problem. In other words, the optimal $\mathbf{p}$ is the eigenvector corresponding to the smallest eigenvalue of A^TA. Once we have found $\mathbf{p}$, we can rearrange its elements to get our projection matrix P.

Constrained Least Squares Solution

Taking derivatives of $L(\mathbf{p}, \lambda)$ w.r.t $\mathbf{p}$: $2A^TA\mathbf{p} - 2\lambda\mathbf{p} = \mathbf{0}$ [1]

$\boxed{A^TA\mathbf{p} = \lambda\mathbf{p}}$ Eigenvalue Problem

Eigenvector $\mathbf{p}$ with smallest eigenvalue λ of matrix A^TA minimizes the loss function $L(\mathbf{p})$.

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

27

Intrinsic and Extrinsic Matrices

Shree K. Nayar
Columbia University

Topic: Camera Calibration, Module: Reconstruction II
First Principles of Computer Vision

28

Extracting Intrinsic/Extrinsic Parameters

We know that:

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

M_{int} M_{ext}

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 Triangular matrix and R is an Orthonormal matrix, it is possible to uniquely "decouple" K and R from their product using "QR factorization".

29

The above calibration method allows us to estimate the projection matrix of a camera. We can go one step further and decompose the projection matrix into the intrinsic matrix M_{int}, which includes all the internal parameters of the camera, and the extrinsic matrix M_{ext}, which has all the external parameters of the camera. Consider the 3×3 submatrix of P outlined at the top. This submatrix is the product of the calibration matrix K in slide 12, which includes all the internal parameters of the camera, and the rotation matrix R. We know that K is an upper right triangular matrix, and the rotation matrix is orthonormal. If a matrix is the product of an upper right triangular matrix and an orthonormal matrix, we can use a linear algebra method called QR factorization to compute the two matrices from their product. Therefore, from our estimated P we can compute K and R.

We have yet to find the translation vector **t**. The last column of P, outlined at the top, equals K times the translation vector **t**. Therefore, we can find the translation vector by computing the product of K^{-1} with the last column of P. At this point, our camera is fully calibrated—we have found all its internal and external parameters.

Extracting Intrinsic/Extrinsic Parameters

We know that:

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

That is:

$$\begin{bmatrix} p_{14} \\ p_{24} \\ p_{34} \end{bmatrix} = \begin{bmatrix} f_x & 0 & o_x \\ 0 & f_y & o_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} = K\mathbf{t}$$

Therefore:

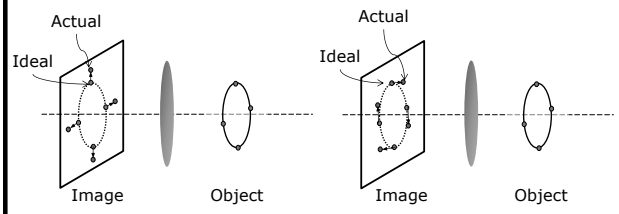
$$\mathbf{t} = K^{-1} \begin{bmatrix} p_{14} \\ p_{24} \\ p_{34} \end{bmatrix}$$

30

It is important to note that our linear camera model given by the projection matrix P does not account for some of the effects that may be found in real imaging systems. As discussed in our lecture on image formation, a lens could exhibit a variety of aberrations. For instance, if the lens has radial distortion, the projection model is no longer perspective. The image projection of each scene point has an additional displacement away from the center of the image, and this displacement increases with the distance of the scene point from the optical axis. In the case of tangential distortion, the imaged points are subjected to a twisting effect. Our camera model would require additional terms to account for these distortions.

Other Intrinsic Parameters

Pinholes do not exhibit image distortions. But, lenses do!



The intrinsic model of the camera will need to include the distortion coefficients. We ignore distortions here.

31

Now, we will present simple stereo—a widely used method for recovering the 3D structure of a scene using two identical calibrated cameras that are displaced with respect to each other.

Simple Stereo

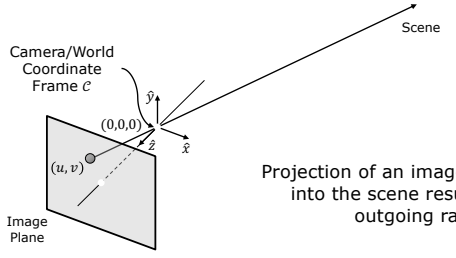
Shree K. Nayar
Columbia University

Topic: Camera Calibration, Module: Reconstruction II
First Principles of Computer Vision

32

Backward Projection: From 2D to 3D

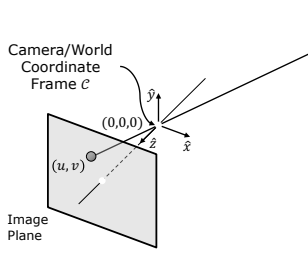
Given a calibrated camera, can we find the 3D scene point from a single 2D image?



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

33

Computing 2D-to-3D Outgoing Ray



3D-to-2D: (Point)

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

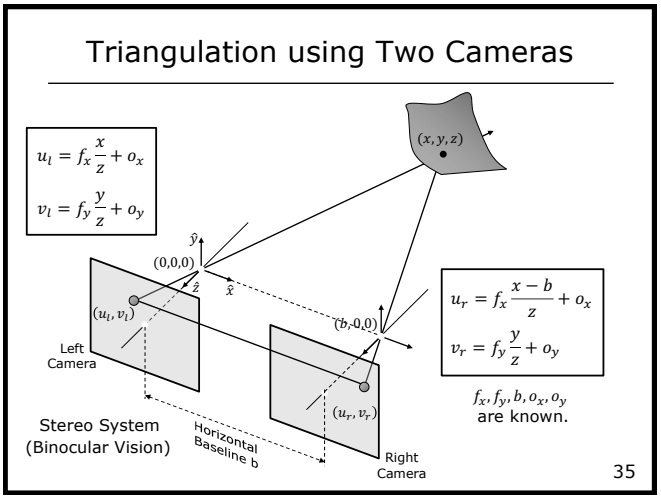
2D-to-3D: (Ray)

$$x = {}^{z}/_{f_x}\,(u - o_x)$$
$$y = {}^{z}/_{f_y}\,(v - o_y)$$
$$z > 0$$

34

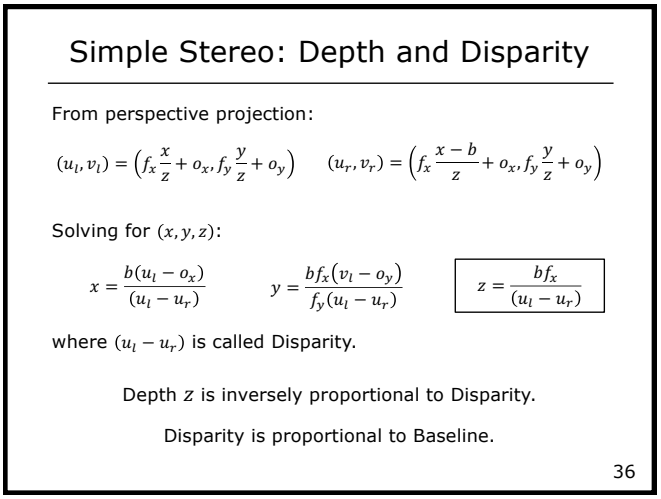
We know that we cannot find the 3D coordinates of a scene point from its 2D image coordinates. All we know is that the 3D point must lie on the ray that emanates from the image point and passes through the center of projection of the camera. Given a calibrated camera, we know the equation of this ray, which is given in the slide on the right.

To find the 3D coordinates of a scene point, we need more information. To this end, we take two images of the scene using two cameras that are displaced with respect to each other. Here, the right camera is identical to the left one but displaced along the horizontal direction by a distance b, called the baseline. This system is called a simple stereo or simple binocular system. It is similar in its working principle to the way we humans use two eyes to perceive depth.



Let us assume that the world frame is located at the center of projection of the left camera with its z −axis aligned with the optical axis. A scene point (x, y, z) is projected to the point (u_l, v_l) in the left camera and the point (u_r, v_r) in the right camera. Let us assume that by some means of feature matching, we know that (u_l, v_l) and (u_r, v_r) correspond to the same scene point. Then, we have four equations—the perspective projection equations for the left and right cameras. Note that the equation for u_r has $x - b$ instead of x, because of the displacement of the right camera. In these equations, the parameters f_x, f_y, o_x, o_y, b are all known to us since our cameras are calibrated.

Given these four equations, we can solve for the scene point (x, y, z), where z is referred to as the depth of the scene point. Note that the denominators of the expressions for x, y, and z contain the term $(u_l - u_r)$. This is the difference in the u coordinates of a scene point in the left and right images, and is called the disparity.



Disparity is inversely proportional to the depth z of the scene point. Therefore, the farther a point is from the stereo system, the smaller its disparity. As the point approaches infinity, its disparity approaches zero. Furthermore, disparity is directly proportional to the baseline of the stereo system.

Since we are dealing with discrete images with finite resolution, the accuracy of depth estimation increases with baseline. Therefore, while designing a stereo system, it makes sense to use the largest baseline the application permits.

Here is an example of a simple stereo camera from Fuji Corp. Stereo camera systems are also commonly found in smartphones today. Scene depth computed by these systems is used for various photographic effects. One example is the use of the depth map to modify a photo of the scene to make it appear like it was shot with a lens with a narrower depth of field.

A Simple Stereo Camera

L2

Fujifilm FinePix REAL 3D W3

37

There is one key step in the stereo pipeline that we have yet to discuss—finding the correspondence between points in the left and right images. This process is known as stereo matching. Shown on the left is one of the two stereo images of a scene, and on the right is a precise disparity map of the scene computed using an active illumination method. In this map, the brighter the point is, the greater its disparity. This disparity map is used as ground truth while evaluating the accuracy of a stereo system.

Stereo Matching: Finding Disparities

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

L3

Left/Right Camera Images

Disparity Map (Ground Truth)

From perspective projection: $v_l = v_r = f_y \frac{y}{z} + o_y$

Corresponding scene points lie on the same horizontal scan line.

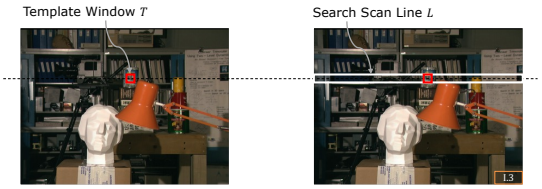
38

Now let us discuss the problem of stereo matching. Note that since in our stereo system the right camera is displaced in the horizontal direction with respect to the left camera, the disparity between corresponding points is only in the horizontal (u) direction. In other words, there is no disparity in the vertical (v) direction, as evidenced by the expressions for v_l and v_r shown at the bottom. That means that corresponding image points must lie on the same horizontal line in the left and right images. This is called scan-line correspondence and it can be used to reduce the search space when finding stereo matches.

Consider a small window in the left image. We can find its matching window in the right image using template matching. Thanks to the scan-line correspondence discussed above, we do not have to search for the matching window in the entire right image. We only need to look for it along the same horizontal line in the right image. The coordinates u_l and u_r of the original window in the left image and its best match in the right image are used to compute disparity. Using this disparity, we can compute the depth z as well as the x and y coordinates, using the expressions in slide 36.

Window Based Methods

Determine Disparity using Template Matching



Left Camera Image E_l Right Camera Image E_r

Disparity: $d = u_l - u_r$ Depth: $z = \frac{bf_x}{(u_l - u_r)}$

39

As discussed in previous lectures, there are several similarity metrics that can be used during template matching. These include the sum of absolute differences (SAD), the sum of squared differences (SSD), and the normalized cross-correlation (NCC). Note that normalized cross-correlation is particularly effective when the two images have radiometric differences due to factors such as vignetting and camera gain.

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 Maximum Normalized Cross-Correlation:

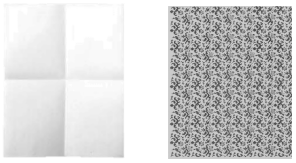
$$NCC(k,l) = \frac{\sum_{(i,j)\in T} E_l(i,j)\, E_r(i+k, j+l)}{\sqrt{\sum_{(i,j)\in T} E_l(i,j)^2 \sum_{(i,j)\in T} E_r(i+k, j+l)^2}}$$

40

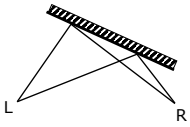
Let's discuss a few important issues related to stereo matching. First, consider the texture-less sheet of paper shown on the left. Its left and right images will be texture-less as well. Note that a texture-less window in the left image will match all texture-less windows in the right image equally well, making stereo matching impossible in this case. Therefore, for stereo to work, the scene must be textured. In addition, the texture must be non-repetitive. Consider the repetitive pattern shown on the right. In this case, each window in the left

Issues with Stereo Matching

- Surface must have (non-repetitive) texture



- Foreshortening effect makes matching challenging

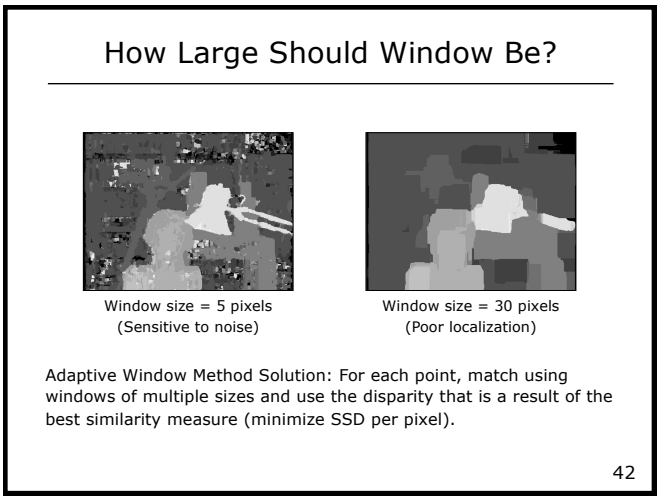


41

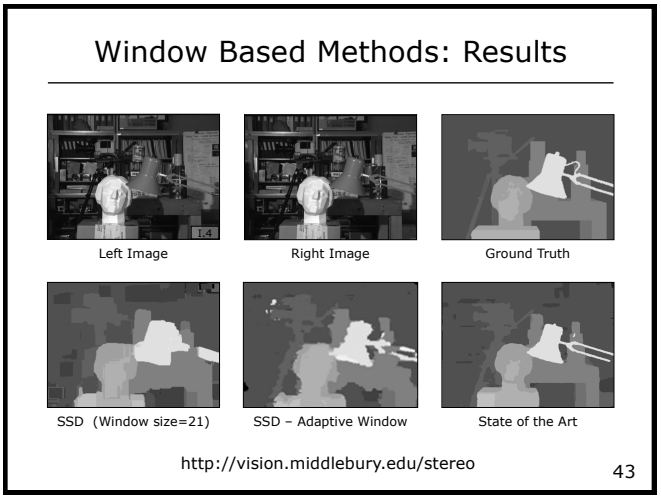
image will match multiple windows in the right image, each match resulting in a different scene depth.

Finally, there is the issue of foreshortening, which is an inherent problem in stereo. In order to find a match, we need to use a window of finite size since a single pixel intensity is not a unique enough feature for matching. Note that a window in the left image corresponds to a patch in the scene. Unless this patch is planar and parallel to the image planes of the two cameras, it will be projected differently in the left and right images. In other words, we are almost always trying to match differently distorted versions of each scene patch. For this reason, some stereo matching algorithms incorporate warping techniques to make the matching process more robust.

A key question during matching is how large the window should be, irrespective of similarity metric. A small window results in good localization but a greater probability of finding incorrect matches, because the smaller the window, the less unique the pattern within it. The result of using a small window size (5 pixels) is shown on the left, and can be seen to be noisy. In contrast, on the right, a large window (30 pixels) is used which produces more robust matches but poorer localization. As a result, the disparity map is more blurred, especially around the edges of objects. One approach to addressing this issue with window size is called the adaptive window method. It uses multiple window sizes for matching, and for each point in the left image, the window size that provides the best match is chosen.



Here are some results using different window-based matching methods. Using the sum of squared difference metric with a window size of 21, we get the disparity map on the bottom left, which may be good enough for some applications. The adaptive window technique produces the slightly improved disparity map shown in the middle. Recently, significant advances have been made with respect to stereo matching. The disparity map produced by a state of art method is shown on the right and is seen to be very close to the ground truth in slide 38.



References: Textbooks

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

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

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

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

45

References: Papers

R.Y. Tsai, An Efficient and Accurate Camera Calibration Technique for 3D Machine Vision. CVPR, 1986.

46

Image Credits

I.1 http://en.wikipedia.org/wiki/File:NASA_Mars_Rover.jpg. NASA. Public Domain.
I.2 commons.wikimedia.org/wiki/File:Fujiw3.jpg. Licensed under CC BY-SA 4.0.
I.3 "A Taxonomy and Evaluation of Dense Two-frame Stereo Correspondence Algorithms". D. Scharstein and R. Szeliski, IJCV, 2002. Used with permission.
I.4 "A Taxonomy and Evaluation of Dense Two-frame Stereo Correspondence Algorithms". D. Scharstein and R. Szeliski, IJCV, 2002. Used with permission.

47

Acknowledgements: Thanks to Pranav Sukumar, Tracy Cui and Kevin Chen for their help with transcription, editing and proofreading.

References

[Szeliski 2022] Computer Vision: Algorithms and Applications, Szeliski, R., Springer, 2022.

[Forsyth and Ponce 2003] Computer Vision: A Modern Approach, Forsyth, D. and Ponce, J., Prentice Hall, 2003.

[Horn 1986] Robot Vision, Horn, B. K. P., MIT Press, 1986.

[Hartley and Zisserman 2000] Multiple View Geometry, Hartley, R. and Zisserman, A., Cambridge University Press, 2000.

[Cyganek and Siebert 2009] An introduction to 3D Computer Vision, Cyganek, B. and Siebert, J. P., Wiley, 2009.

[Tsai 1986] An Efficient and Accurate Camera Calibration Technique for 3D Machine Vision, Tsai, R. Y., CVPR, 1986.