Sensors & Measurement

"Seeing is forgetting the name of what one sees"

– L. Weschler

METR 4202: Advanced Control & Robotics

Dr Surya Singh

Lecture # 6

August 30, 2013

metr4202@itee.uq.edu.au

http://itee.uq.edu.au/~metr4202/

Schedule

Week	Date	Lecture (F: 9-10:30, 42-212)
1	26-Jul	Introduction
2	2-Aug	Representing Position & Orientation & State (Frames, Transformation Matrices & Affine Transformations)
3	9-Aug	Robot Kinematics
4	16-Aug	Robot Dynamics & Control
5	23-Aug	Robot Trajectories & Motion
6	**30-Aug**	**Sensors & Measurement**
7	6-Sep	Perception / Computer Vision
8	13-Sep	Localization and Navigation
9	20-Sep	State-Space Modelling
	27-Sep	State-Space Control
10	4-Oct	*Study break*
11	11-Oct	Motion Planning
12	18-Oct	Vision-based control (+ Prof. P. Corke or Prof. M. Srinivasan)
13	25-Oct	Applications in Industry (+ Prof. S. LaValle) & Course Review

Announcements:

- Adam Keyes Needs More Volunteers

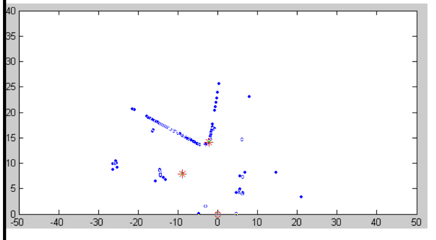
- UQ Summer Research Program:
  Applications Due August 30th

Quick Outline

1. **Kinematics Lab Recap**

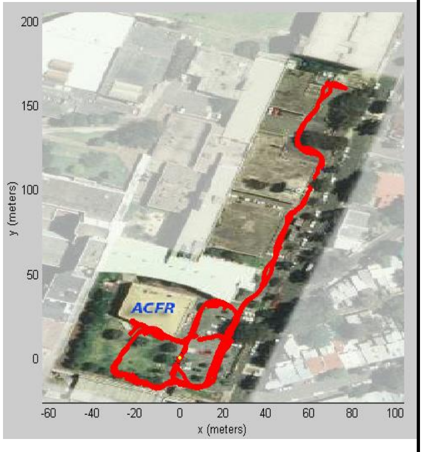
2. **Perception → Camera Sensors**
   1. Image Formation
      → "Computational Photography"
   2. Calibration
   3. Feature extraction
   4. Stereopsis and depth
   5. Optical flow

Sensor Information



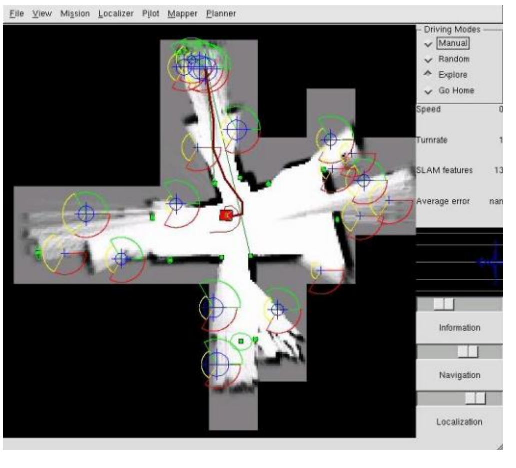
Laser

Vision

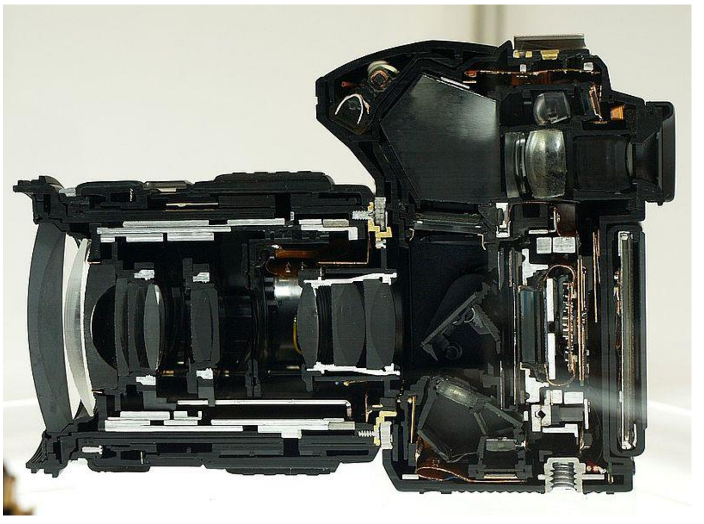


GPS

Mapping: Indoor robots

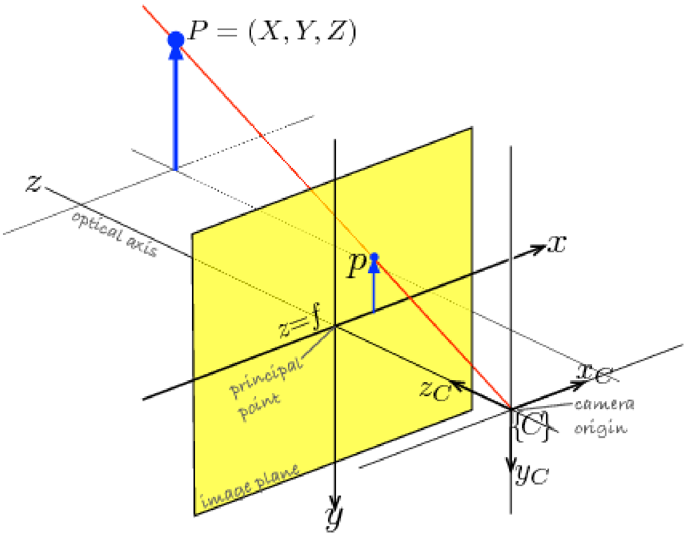


Cameras



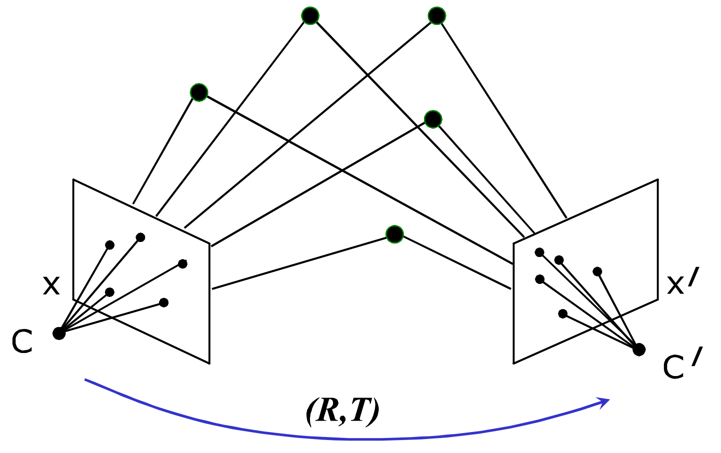
Wikipedia, E-30-Cutmodel

Image Formation



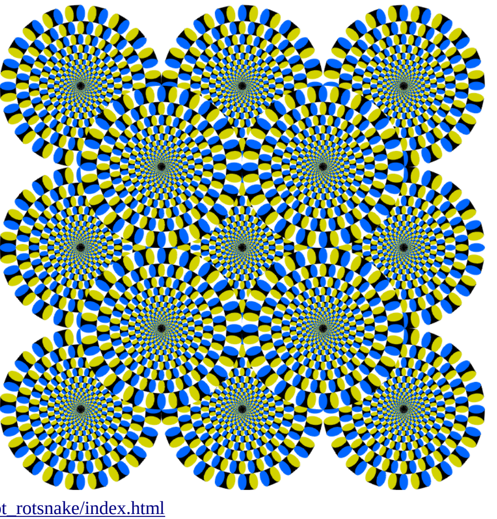
Corke, Ch. 11

Stereopsis



Perception

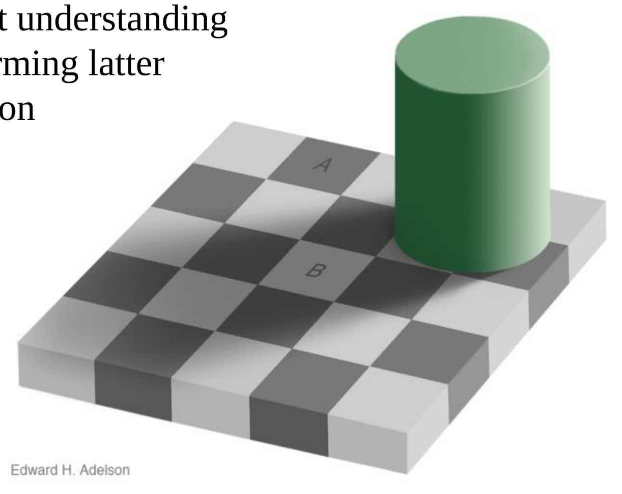
- Making Sense from Sensors



http://www.michaelbach.de/ot/mot_rotsnake/index.html

Perception

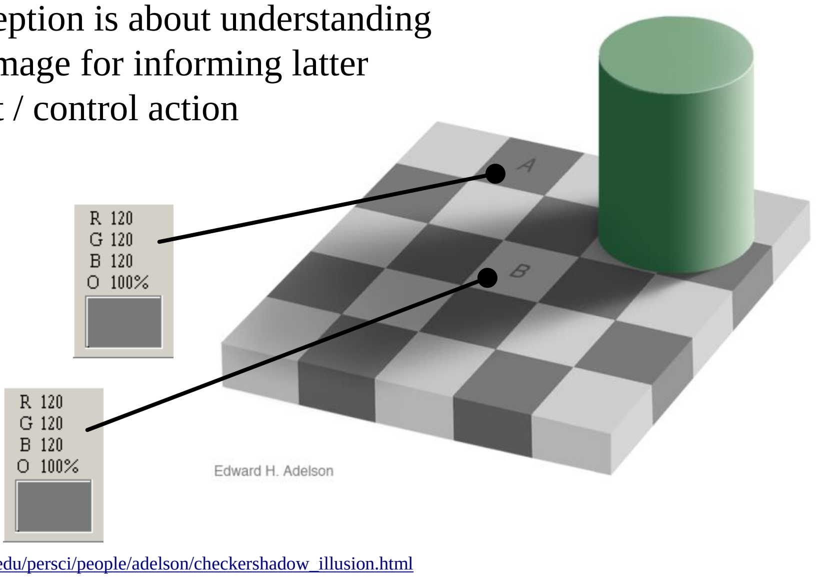
- Perception is about understanding the image for informing latter robot / control action



http://web.mit.edu/persci/people/adelson/checkershadow_illusion.html

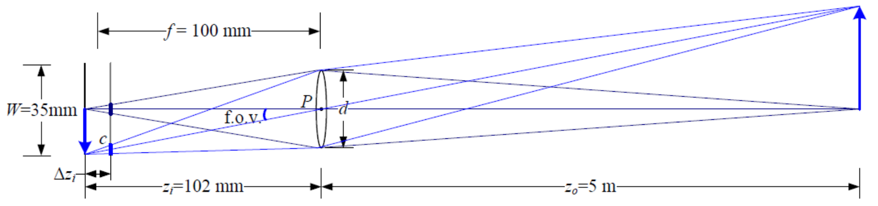
Perception

- Perception is about understanding the image for informing latter robot / control action



http://web.mit.edu/persci/people/adelson/checkershadow_illusion.html

Image Formation: Lens Optics

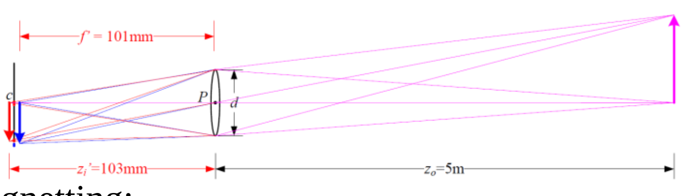


$$\frac{1}{z_0} + \frac{1}{z_1} = \frac{1}{f}$$

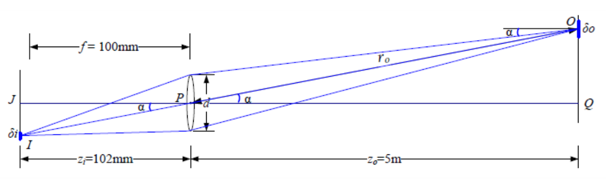
Sec. 2.2 from Szeliski, *Computer Vision: Algorithms and Applications*

Image Formation: Lens Optics (Chromatic Aberration & Vignetting)

- Chromatic Aberration:



- Vignetting:



$$E = L\frac{\pi}{4}\left(\frac{d}{f}\right)^2 \cos^4 \alpha$$

Sec. 2.2 from Szeliski, *Computer Vision: Algorithms and Applications*

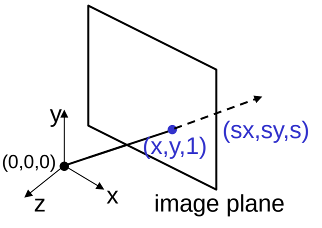
Image Formation:
Lens Optics (Aperture / Depth of Field)

$$N = \frac{f}{\#} = \frac{f}{d}$$

http://en.wikipedia.org/wiki/File:Aperture_in_Canon_50mm_f1.8_II_lens.jpg

The Projective Plane

- Why do we need homogeneous coordinates?
  - Represent points at infinity, homographies, perspective projection, multi-view relationships
- What is the geometric intuition?
  - A point in the image is a ray in projective space

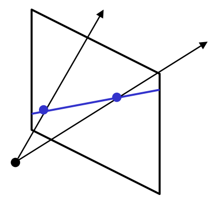


- Each *point* (x,y) on the plane is represented by a *ray* (sx,sy,s)
  - all points on the ray are equivalent: (x, y, 1) ≅ (sx, sy, s)

Slide from Szeliski, *Computer Vision: Algorithms and Applications*

Projective Lines

- What is a line in projective space?



- A line is a *plane* of rays through origin
  - all rays (x,y,z) satisfying: ax + by + cz = 0

in vector notation: $$0 = \underbrace{\begin{bmatrix} a & b & c \end{bmatrix}}_{\mathbf{l}^T} \underbrace{\begin{bmatrix} x \\ y \\ z \end{bmatrix}}_{\mathbf{p}}$$

- A line is represented as a homogeneous 3-vector **l**

Slide from Szeliski, *Computer Vision: Algorithms and Applications*

2-D Transformations

- Translation x' = x + t
- Rotation x' = R x + t
- Similarity x' = sR x + t
- Affine x' = A x
- Projective x' = A x

here, x is an inhomogeneous pt (2-vector)

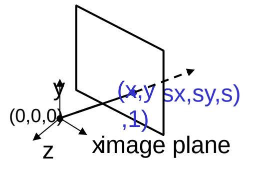
x' is a homogeneous point

2-D Transformations

Name	Matrix	# D.O.F.	Preserves:	Icon
translation	$\begin{bmatrix} I \mid t \end{bmatrix}_{2\times 3}$	2	orientation $+\cdots$	
rigid (Euclidean)	$\begin{bmatrix} R \mid t \end{bmatrix}_{2\times 3}$	3	lengths $+\cdots$	
similarity	$\begin{bmatrix} sR \mid t \end{bmatrix}_{2\times 3}$	4	angles $+\cdots$	
affine	$\begin{bmatrix} A \end{bmatrix}_{2\times 3}$	6	parallelism $+\cdots$	
projective	$\begin{bmatrix} \tilde{H} \end{bmatrix}_{3\times 3}$	8	straight lines	

Planar Projective Transformations

- Perspective projection of a plane
  - lots of names for this:
    - homography, colineation, planar projective map
  - Easily modeled using homogeneous coordinates



$$\begin{bmatrix} sx' \\ sy' \\ s \end{bmatrix} = \begin{bmatrix} * & * & * \\ * & * & * \\ * & * & * \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

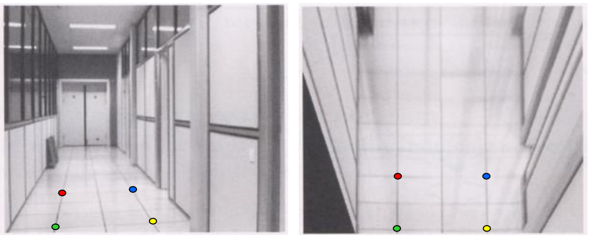
p’ **H** **p**

To apply a homography **H**

- compute **p’** = **Hp**
- **p”** = **p’**/s normalize by dividing by third component

Slide from Szeliski, *Computer Vision: Algorithms and Applications*

Image Rectification



To unwarp (rectify) an image

- solve for **H** given **p”** and **p**
- solve equations of the form: s**p”** = **Hp**
  - linear in unknowns: s and coefficients of **H**
  - need at least 4 points

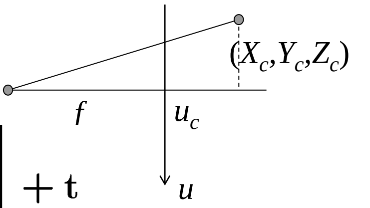
Slide from Szeliski, *Computer Vision: Algorithms and Applications*

3D Projective Geometry

- These concepts generalize naturally to 3D
  - Homogeneous coordinates
    - Projective 3D points have four coords: P = (X,Y,Z,W)
  - Duality
    - A plane L is also represented by a 4-vector
    - Points and planes are dual in 3D: L P=0
  - Projective transformations
    - Represented by 4x4 matrices T: P’ = TP, L’ = L T-1
  - Lines are a special case…

Slide from Szeliski, *Computer Vision: Algorithms and Applications*

3D → 2D Perspective Projection



$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = \begin{bmatrix} \mathbf{R} \end{bmatrix}_{3\times 3} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \mathbf{t}$$

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \sim \begin{bmatrix} U \\ V \\ W \end{bmatrix} = \begin{bmatrix} f & 0 & u_c \\ 0 & f & v_c \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix}$$

Slide from Szeliski, *Computer Vision: Algorithms and Applications*

3D → 2D Perspective Projection

- Matrix Projection (camera matrix):

$$\mathbf{p} = \begin{bmatrix} sx \\ sy \\ s \end{bmatrix} = \begin{bmatrix} * & * & * & * \\ * & * & * & * \\ * & * & * & * \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \mathbf{\Pi P}$$

It's useful to decompose $\prod$ into **T** → **R** → project → **A**

$$\mathbf{\Pi} = \underbrace{\begin{bmatrix} s_x & 0 & -t_x \\ 0 & s_y & -t_y \\ 0 & 0 & 1/f \end{bmatrix}}_{\text{intrinsics}} \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{\text{projection}} \underbrace{\begin{bmatrix} \mathbf{R}_{3x3} & \mathbf{0}_{3x1} \\ \mathbf{0}_{1x3} & 1 \end{bmatrix}}_{\text{orientation}} \underbrace{\begin{bmatrix} \mathbf{I}_{3x3} & \mathbf{T}_{3x1} \\ \mathbf{0}_{1x3} & 1 \end{bmatrix}}_{\text{position}}$$

Slide from Szeliski, *Computer Vision: Algorithms and Applications*

3D Transformations

Name	Matrix	# D.O.F.	Preserves:	Icon
translation	$\left[\begin{array}{c|c} I & t \end{array}\right]_{3\times 4}$	3	orientation $+\cdots$	
rigid (Euclidean)	$\left[\begin{array}{c|c} R & t \end{array}\right]_{3\times 4}$	6	lengths $+\cdots$	
similarity	$\left[\begin{array}{c|c} sR & t \end{array}\right]_{3\times 4}$	7	angles $+\cdots$	
affine	$\left[\begin{array}{c} A \end{array}\right]_{3\times 4}$	12	parallelism $+\cdots$	
projective	$\left[\begin{array}{c} \tilde{H} \end{array}\right]_{4\times 4}$	15	straight lines	

Slide from Szeliski, *Computer Vision: Algorithms and Applications*

Projection Models

- Orthographic

$$\mathbf{\Pi} = \begin{bmatrix} i_x & i_y & i_z & t_x \\ j_x & j_y & j_z & t_y \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Weak Perspective

$$\mathbf{\Pi} = f\begin{bmatrix} i_x & i_y & i_z & t_x \\ j_x & j_y & j_z & t_y \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Affine

$$\mathbf{\Pi} = \begin{bmatrix} * & * & * & * \\ * & * & * & * \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Perspective

$$\mathbf{\Pi} = [\mathbf{R} \quad \mathbf{t}]$$

- Projective

$$\mathbf{\Pi} = \begin{bmatrix} * & * & * & * \\ * & * & * & * \\ * & * & * & * \end{bmatrix}$$

Slide from Szeliski, *Computer Vision: Algorithms and Applications*

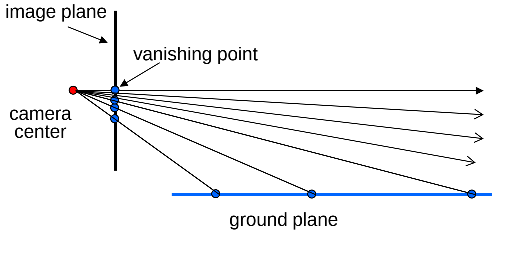
Properties of Projection

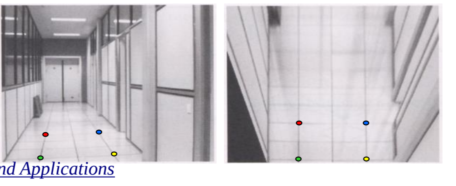
- Preserves
  - Lines and conics
  - Incidence
  - Invariants (cross-ratio)

- Does not preserve
  - Lengths
  - Angles
  - Parallelism

Vanishing Points

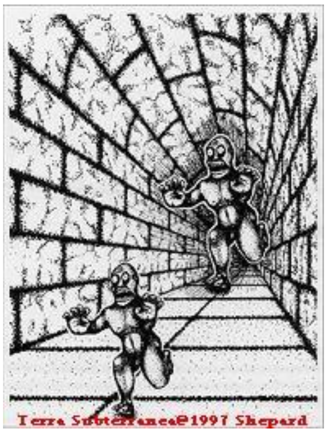
- Vanishing point
  - projection of a point at infinity
  - whiteboard capture, architecture,…

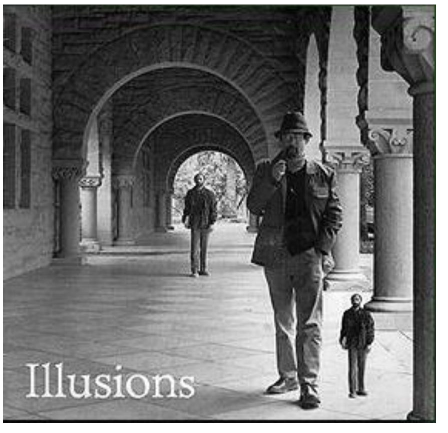




Slide from Szeliski, *Computer Vision: Algorithms and Applications*

Fun With Vanishing Points

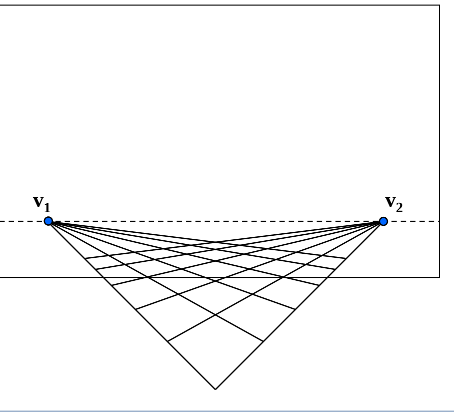




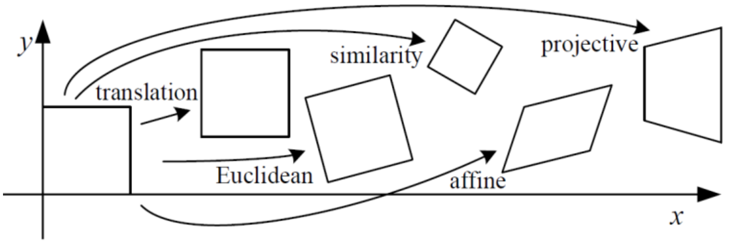
Slide from Szeliski, *Computer Vision: Algorithms and Applications*

Vanishing Lines

- Multiple Vanishing Points
  - Any set of parallel lines on the plane define a vanishing point
  - The union of all of these vanishing points is the horizon line



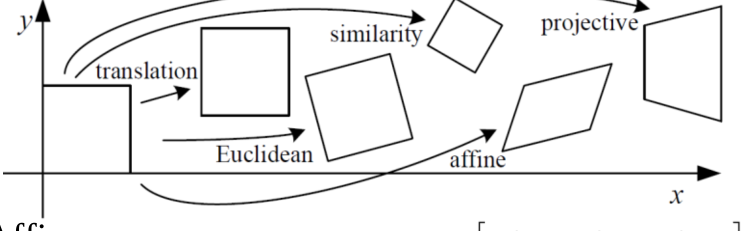
Transformations



- ***x*'**: New Image & **x** : Old Image
- Euclidean: (Distances preserved)
  $$x' = \begin{bmatrix} R & t \end{bmatrix} \underline{x}$$
- Similarity (Scaled Rotation): (Angles preserved)
  $$x' = \begin{bmatrix} sR & t \end{bmatrix} \underline{x}$$

Fig. 2.4 from Szeliski, *Computer Vision: Algorithms and Applications*

Transformations [2]



- Affine : (|| lines remain ||)
  $$x' = \begin{bmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \end{bmatrix} \underline{x}$$
- Projective: (straight lines preserved) H: Homogenous 3x3 Matrix
  $$x' = \mathbf{H}\underline{x}$$
  $$x' = \frac{h_{00}x + h_{01}y + h_{02}}{h_{20}x + h_{21}y + h_{22}}$$
  $$y' = \frac{h_{10}x + h_{11}y + h_{12}}{h_{20}x + h_{21}y + h_{22}}$$

Fig. 2.4 from Szeliski, Computer Vision: Algorithms and Applications

Transformations [3]

- Forward Warp

procedure *forwardWarp*$(f, h,$ **out** $g)$:

For every pixel x in $f(x)$

1. Compute the destination location $x' = h(x)$.
2. Copy the pixel $f(x)$ to $g(x')$.

- **<u>Inverse Warp</u>**

procedure *inverseWarp*$(f, h,$ **out** $g)$:

For every pixel x' in $g(x')$

1. Compute the source location $x = \hat{h}(x')$
2. Resample $f(x)$ at location x and copy to $g(x')$

Sec. 3.6 from Szeliski, *Computer Vision: Algorithms and Applications*

Calibration

See: *Camera Calibration Toolbox for Matlab*
(http://www.vision.caltech.edu/bouguetj/calib_doc/)

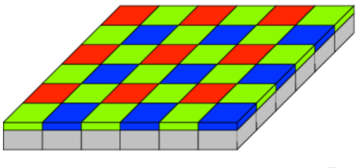
- Intrinsic: Internal Parameters
  - **Focal length:** The focal length in pixels.
  - **Principal point:** The principal point
  - **Skew coefficient:**
    The skew coefficient defining the angle between the x and y pixel axes.
  - **Distortions:** The image distortion coefficients (radial and tangential distortions) (typically two quadratic functions)

- Extrinsics: Where the Camera (image plane) is placed:
  - **Rotations:** A set of 3x3 rotation matrices for each image
  - **Translations:** A set of 3x1 translation vectors for each image

Features

- Colour
- Corners
- Edges
- Lines
- Statistics on Edges: SIFT, SURF

Features -- Colour Features



Bayer Patterns

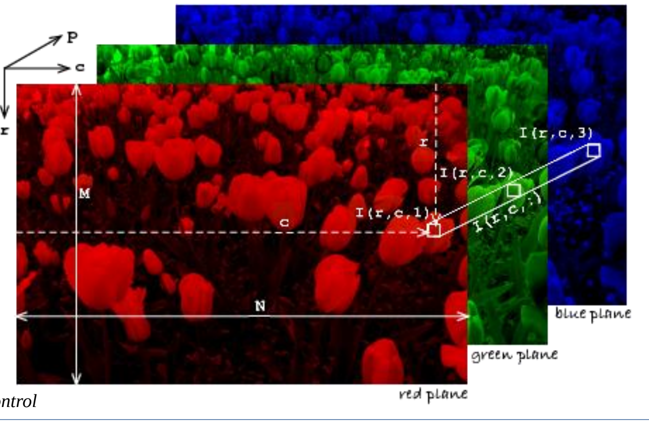


Fig: Ch. 10, *Robotics Vision and Control*

Edge Detection

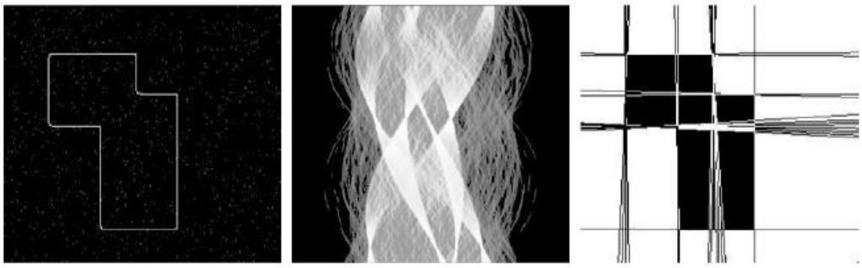
- Canny edge detector:

Fig: Ch. 10, *Robotics Vision and Control*

Edge Detection

- Canny edge detector:

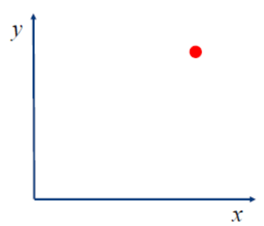
Hough Transform

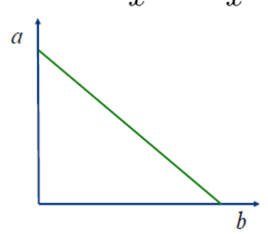


- Uses a voting mechanism
- Can be used for other lines and shapes (not just straight lines)

Hough Transform: Voting Space

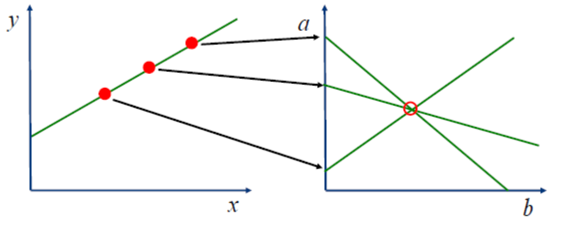
$$y = ax + b \qquad a = -\frac{1}{x}b + \frac{y}{x}$$





- Count the number of lines that can go through a point and move it from the "*x-y*" plane to the "a-b" plane
- There is only a one-"infinite" number (a line!) of solutions (not a two-"infinite" set – a plane)

Hough Transform: Voting Space



- In practice, the polar form is often used

$$a = x\cos a + y\sin b$$

- This avoids problems with lines that are nearly vertical

Hough Transform: Algorithm

1. Quantize the parameter space appropriately.

2. Assume that each cell in the parameter space is an accumulator. Initialize all cells to zero.

3. For each point (x,y) in the (visual & range) image space, increment by 1 each of the accumulators that satisfy the equation.

4. Maxima in the accumulator array correspond to the parameters of model instances.

Cool Robotics Share

D. Wedge, *The Fundamental Matrix Song*