Robust Features and Multiple View Geometry

METR 4202: **Robotics** & Automation

Dr Surya Singh -- Lecture # 8

September 13, 2017

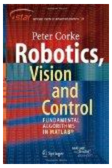
metr4202-staff@itee.uq.edu.au

http://robotics.itee.uq.edu.au/~metr4202/ [http://**metr4202.com**]

Lecture Schedule

Week	Date	Lecture (W: 3:05p-4:50, 7-222)
1	26-Jul	Introduction + Representing Position & Orientation & State
2	2-Aug	Robot Forward Kinematics (Frames, Transformation Matrices & Affine Transformations)
3	9-Aug	Robot Inverse Kinematics & Dynamics (Jacobians)
4	16-Aug	*Ekka Day* (Robot Kinematics & Kinetics Review)
5	23-Aug	Jacobians & Robot Sensing Overview
6	30-Aug	Robot Sensing: Single View Geometry & Lines
7	6-Sep	Robot Sensing: Basic Feature Detection
8	13-Sep	Robot Sensing: Scalable Feature Detection & Multiple View Geometry
9	20-Sep	***Mid-Semester Exam***
	27-Sep	*Study break*
10	4-Oct	Motion Planning
11	11-Oct	Probabilistic Robotics: Localization & SLAM
12	18-Oct	Probabilistic Robotics: Planning & Control (State-Space/Shaping the Dynamic Response/LQR)
13	25-Oct	The Future of Robotics/Automation + Challenges + Course Review

Follow Along Reading:



Robotics, Vision & Control
by Peter Corke

Also online:SpringerLink

UQ Library eBook:
364220144X

Today

➔ **Multiple View Geometry** 🡐

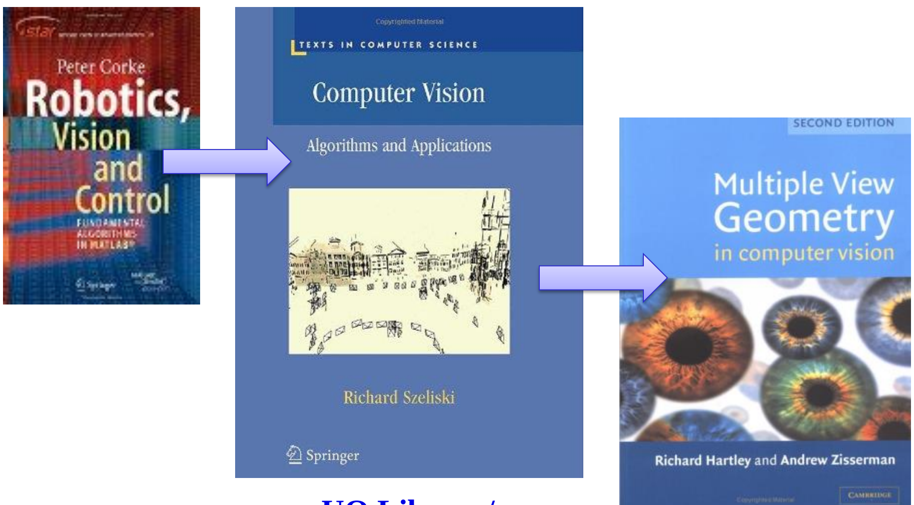
- RVC
  - §14.1-14.4: Multiple Images

(Next week: MidTerm Exam too!)

- Planning
  - pp. 91-103
    (Yup! That's all Peter Corke has to say on that – which explains why there is no planning at ACRV ☺).

Next Time

Reference Material



UQ Library/ SpringerLink

UQ Library (ePDF)

Feature Detection

"A Rose By Any Other Name?

3 8 1 7 6 7 4 7 8 3 5 9 5 3 6 3 7 4 4 6 9 3 8 7 9 0 3 6 3 2 6 6 5 6 0 3 4 2 6 8 3 8 1…
7 6 7 4 7 8 3 5 9 5 3 6 3 7 4 4 6 9 3 8 7 9 0 3 6 3 2 6 6 5 6 0 3 4 2 6 8

– SIFT

How to get Matching Points? **Features**

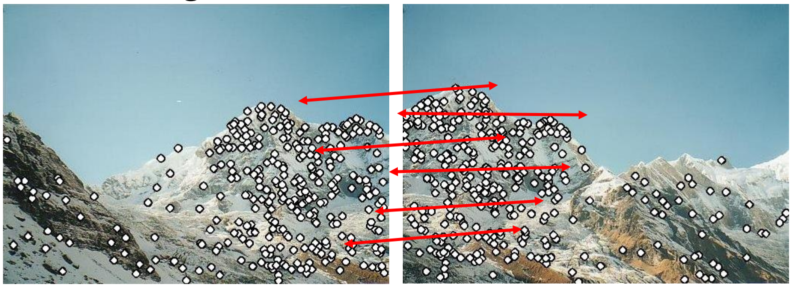
- ~~Colour~~
- Corners
- Edges
- Lines
- Statistics on Edges: SIFT, SURF, ORB…

In OpenCV: The following detector types are supported:

- "FAST" – FastFeatureDetector
- "STAR" – StarFeatureDetector
- "SIFT" – SIFT (nonfree module)
- "SURF" – SURF (nonfree module)
- "ORB" – ORB
- "BRISK" – BRISK
- "MSER" – MSER
- "GFTT" – GoodFeaturesToTrackDetector
- "HARRIS" – GoodFeaturesToTrackDetector with Harris detector enabled
- "Dense" – DenseFeatureDetector
- "SimpleBlob" – SimpleBlobDetector

Why extract features?

- **Object detection**
- Robot Navigation
- Scene Recognition



- Steps:
  - Extract Features
  - Match Features

Adopted drom S. Lazebnik, Gang Hua (CS 558)

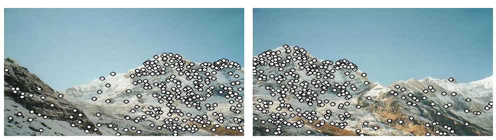
Why extract features? [2]

- Panorama stitching…
  - → Step 3: Align images

Adopted from S. Lazebnik, Gang Hua (CS 558)

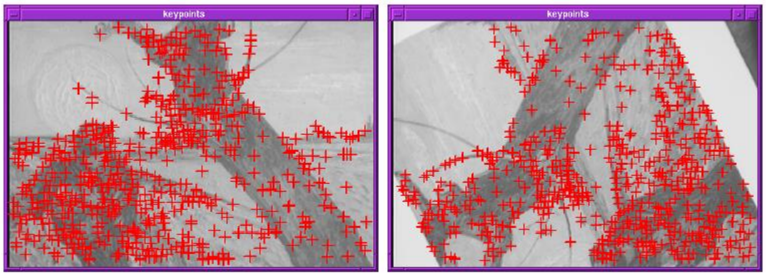
Characteristics of good features

- Repeatability
  - The same feature can be found in several images despite geometric and photometric transformations
- Saliency
  - Each feature is distinctive
- Compactness and efficiency
  - Many fewer features than image pixels
- Locality
  - A feature occupies a relatively small area of the image; robust to clutter and occlusion



Adopted from S. Lazebnik, Gang Hua (CS 558)

Finding Corners



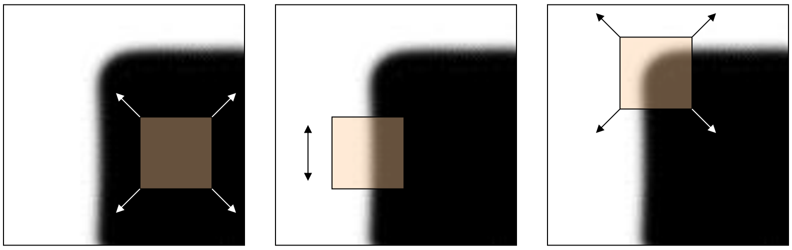
- Key property: in the region around a corner, image gradient has two or more dominant directions
- Corners are repeatable and distinctive

C.Harris and M.Stephens. "A Combined Corner and Edge Detector.“ *Proceedings of the 4th Alvey Vision Conference*: pages 147—151, 1988.

Adopted from S. Lazebnik, Gang Hua (CS 558)

Corner Detection: Basic Idea

- Look through a window
- Shifting a window in any direction should give a large change in intensity



"flat" region: no change in all directions

"edge": no change along the edge direction

"corner": significant change in all directions

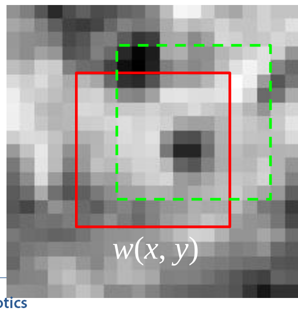
Source: A. Efros

Corner Detection: Mathematics

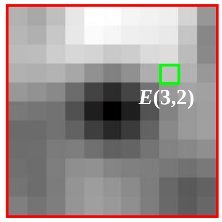
Change in appearance of window $w(x,y)$
for the shift $[u,v]$:

$$E(u,v) = \sum_{x,y} w(x,y)\left[I(x+u, y+v) - I(x,y)\right]^2$$

$I(x, y)$



$E(u, v)$



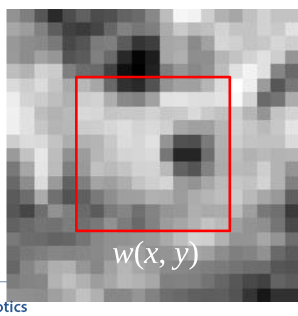
Adopted from
S. Lazebnik,
Gang Hua (CS 558)

Corner Detection: Mathematics

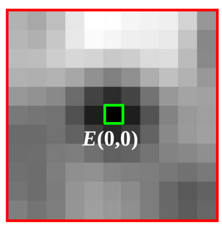
Change in appearance of window $w(x,y)$
for the shift $[u,v]$:

$$E(u,v) = \sum_{x,y} w(x,y)\left[I(x+u, y+v) - I(x,y)\right]^2$$

$I(x, y)$



$E(u, v)$



Adopted from
S. Lazebnik,
Gang Hua (CS 558)

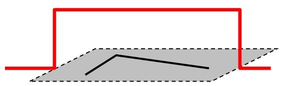
Corner Detection: Mathematics

Change in appearance of window $w(x,y)$
for the shift $[u,v]$:

$$E(u,v) = \sum_{x,y} w(x,y)\left[I(x+u,y+v) - I(x,y)\right]^2$$

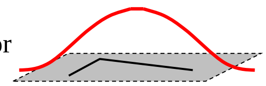
Window function — Shifted intensity — Intensity

Window function $w(x,y)$ =



1 in window, 0 outside

or



Gaussian

Adopted from
S. Lazebnik,
Gang Hua (CS 558)

Source: R. Szeliski

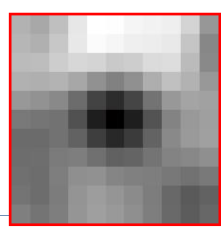
Corner Detection: Mathematics

Change in appearance of window $w(x,y)$
for the shift $[u,v]$:

$$E(u,v) = \sum_{x,y} w(x,y)\left[I(x+u,y+v) - I(x,y)\right]^2$$

We want to find out how this function behaves for small shifts

$E(u, v)$



Adopted from
S. Lazebnik,
Gang Hua (CS 558)

Corner Detection: Mathematics

Change in appearance of window $w(x,y)$
for the shift $[u,v]$:

$$E(u,v) = \sum_{x,y} w(x,y)\left[I(x+u,y+v) - I(x,y)\right]^2$$

We want to find out how this function behaves for small shifts

$$E(u,v) \approx E(0,0) + [u \;\; v]\begin{bmatrix} E_u(0,0) \\ E_v(0,0) \end{bmatrix} + \frac{1}{2}[u \;\; v]\begin{bmatrix} E_{uu}(0,0) & E_{uv}(0,0) \\ E_{uv}(0,0) & E_{vv}(0,0) \end{bmatrix}\begin{bmatrix} u \\ v \end{bmatrix}$$

Local quadratic approximation of $E(u,v)$ in the neighborhood of (0,0) is given by the *second-order Taylor expansion*:

Adopted from
S. Lazebnik,
Gang Hua (CS 558)

Corner Detection: Mathematics

$$E(u,v) = \sum_{x,y} w(x,y)\left[I(x+u,y+v) - I(x,y)\right]^2$$

Second-order Taylor expansion of $E(u,v)$ about (0,0):

$$E(u,v) \approx E(0,0) + [u \;\; v]\begin{bmatrix} E_u(0,0) \\ E_v(0,0) \end{bmatrix} + \frac{1}{2}[u \;\; v]\begin{bmatrix} E_{uu}(0,0) & E_{uv}(0,0) \\ E_{uv}(0,0) & E_{vv}(0,0) \end{bmatrix}\begin{bmatrix} u \\ v \end{bmatrix}$$

$$E_u(u,v) = \sum_{x,y} 2w(x,y)\left[I(x+u,y+v) - I(x,y)\right]I_x(x+u,y+v)$$

$$E_{uu}(u,v) = \sum_{x,y} 2w(x,y)I_x(x+u,y+v)I_x(x+u,y+v)$$
$$+ \sum_{x,y} 2w(x,y)\left[I(x+u,y+v) - I(x,y)\right]I_{xx}(x+u,y+v)$$

$$E_{uv}(u,v) = \sum_{x,y} 2w(x,y)I_y(x+u,y+v)I_x(x+u,y+v)$$
$$+ \sum_{x,y} 2w(x,y)\left[I(x+u,y+v) - I(x,y)\right]I_{xy}(x+u,y+v)$$

Adopted from
S. Lazebnik,
Gang Hua (CS 558)

Corner Detection: Mathematics

$$E(u,v) = \sum_{x,y} w(x,y)\left[I(x+u, y+v) - I(x,y)\right]^2$$

Second-order Taylor expansion of $E(u,v)$ about (0,0):

$$E(u,v) \approx [u \;\; v] \begin{bmatrix} \sum_{x,y} w(x,y) I_x^2(x,y) & \sum_{x,y} w(x,y) I_x(x,y) I_y(x,y) \\ \sum_{x,y} w(x,y) I_x(x,y) I_y(x,y) & \sum_{x,y} w(x,y) I_y^2(x,y) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}$$

$$E(0,0) = 0$$

$$E_u(0,0) = 0$$

$$E_v(0,0) = 0$$

$$E_{uu}(0,0) = \sum_{x,y} 2w(x,y) I_x(x,y) I_x(x,y)$$

$$E_{vv}(0,0) = \sum_{x,y} 2w(x,y) I_y(x,y) I_y(x,y)$$

$$E_{uv}(0,0) = \sum_{x,y} 2w(x,y) I_x(x,y) I_y(x,y)$$

Adopted from
S. Lazebnik,
Gang Hua (CS 558)

Harris detector: Steps

- Compute Gaussian derivatives at each pixel
- Compute second moment matrix M in a Gaussian window around each pixel
- Compute corner response function R
- Threshold R
- Find local maxima of response function (nonmaximum suppression)

C.Harris and M.Stephens. "A Combined Corner and Edge Detector." *Proceedings of the 4th Alvey Vision Conference*: pages 147—151, 1988.

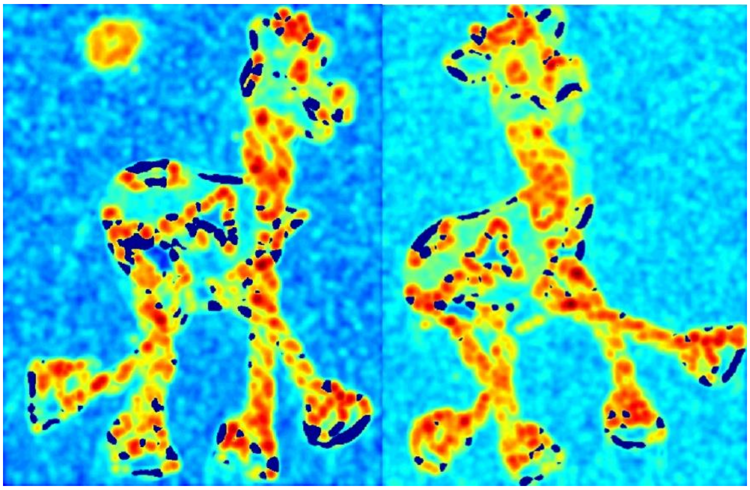
Adopted from
S. Lazebnik,
Gang Hua (CS 558)

Harris Detector: Steps

Adopted from S. Lazebnik, Gang Hua ([CS 558](CS 558))

Harris Detector: Steps

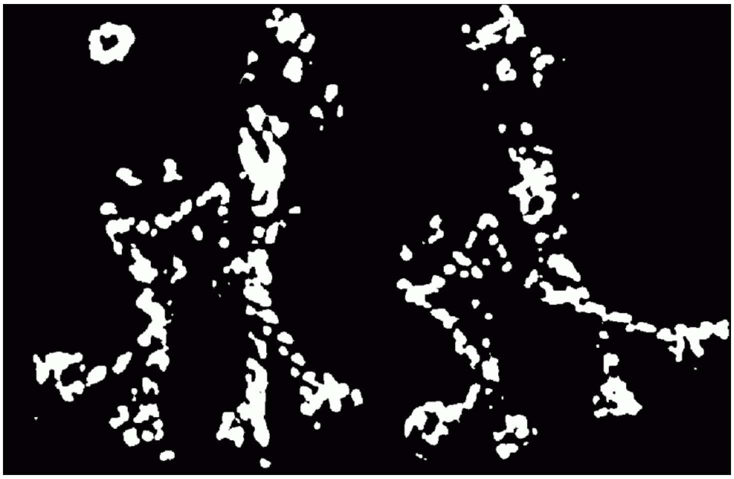
Compute corner response *R*



Adopted from S. Lazebnik, Gang Hua (CS 558)

Harris Detector: Steps

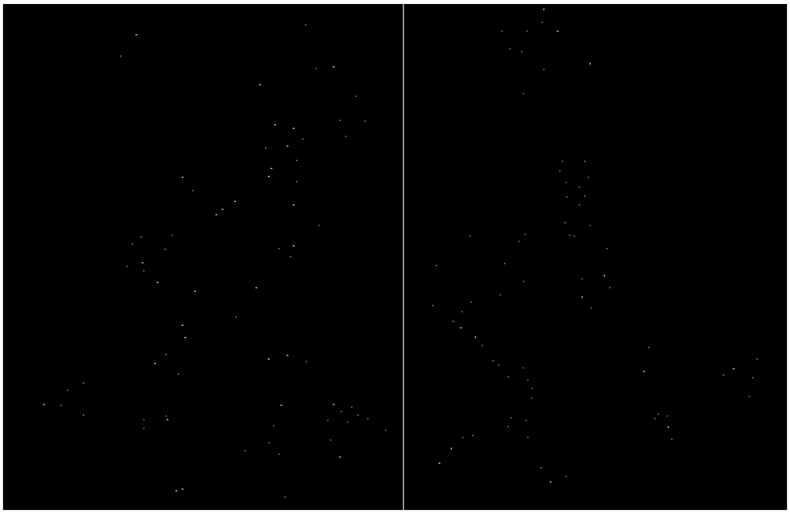
Find points with large corner response: R>threshold



Adopted from S. Lazebnik, Gang Hua (CS 558)

Harris Detector: Steps

Take only the points of local maxima of R



Adopted from S. Lazebnik, Gang Hua (CS 558)

Harris Detector: Steps

Adopted from S. Lazebnik, Gang Hua (CS 558)

Invariance and covariance

- We want corner locations to be invariant to photometric transformations and covariant to geometric transformations
  - Invariance: image is transformed and corner locations do not change
  - Covariance: if we have two transformed versions of the same image, features should be detected in corresponding locations

Adopted from S. Lazebnik, Gang Hua (CS 558)

Feature matching

- Given a feature in I_1, how to find the best match in I_2?
  1. Define distance function that compares two descriptors
  2. Test all the features in I_2, find the one with min distance

From Szeliski, *Computer Vision: Algorithms and Applications*

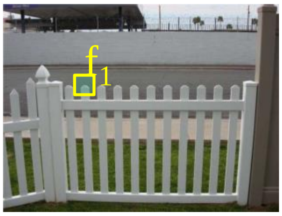
Feature distance

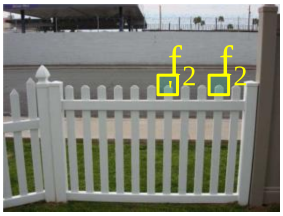
- How to define the difference between two features f_1, f_2?
  - Simple approach is SSD(f_1, f_2)
    - sum of square differences between entries of the two descriptors
    - can give good scores to very ambiguous (bad) matches

From Szeliski, *Computer Vision: Algorithms and Applications*

Feature distance

- How to define the difference between two features f_1, f_2?
  - Better approach: ratio distance = $SSD(f_1, f_2) / SSD(f_1, f_2')$
    - f_2 is best SSD match to f_1 in I_2
    - f_2' is 2^{nd} best SSD match to f_1 in I_2
    - gives small values for ambiguous matches



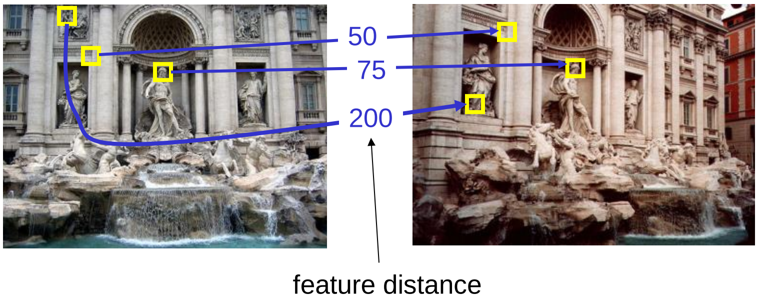


I_1 I_2

From Szeliski, *Computer Vision: Algorithms and Applications*

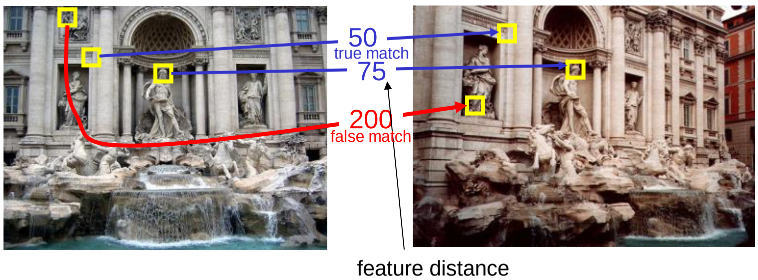
Evaluating the results

- How can we measure the performance of a feature matcher?



From Szeliski, *Computer Vision: Algorithms and Applications*

True/false positives



- The distance threshold affects performance
  - True positives = # of detected matches that are correct
    - Suppose we want to maximize these—how to choose threshold?
  - False positives = # of detected matches that are incorrect
    - Suppose we want to minimize these—how to choose threshold?

From Szeliski, *Computer Vision: Algorithms and Applications*

Levenberg-Marquardt

- Iterative non-linear least squares [Press'92]
  - Linearize measurement equations

$$\hat{u}_i = f(\mathbf{m}, \mathbf{x}_i) + \frac{\partial f}{\partial \mathbf{m}} \Delta \mathbf{m}$$
$$\hat{v}_i = g(\mathbf{m}, \mathbf{x}_i) + \frac{\partial g}{\partial \mathbf{m}} \Delta \mathbf{m}$$

  - Substitute into log-likelihood equation: quadratic cost function in Dm

$$\sum_i \sigma_i^{-2} (\hat{u}_i - u_i + \frac{\partial f}{\partial \mathbf{m}} \Delta \mathbf{m})^2 + \cdots$$

From Szeliski, *Computer Vision: Algorithms and Applications*

Levenberg-Marquardt

- What if it doesn't converge?
  - Multiply diagonal by (1 + l), increase l until it does
  - Halve the step size Dm (my favorite)
  - Use line search
  - Other ideas?
- Uncertainty analysis: covariance S = A-1
- Is maximum likelihood the best idea?
- How to start in vicinity of global minimum?

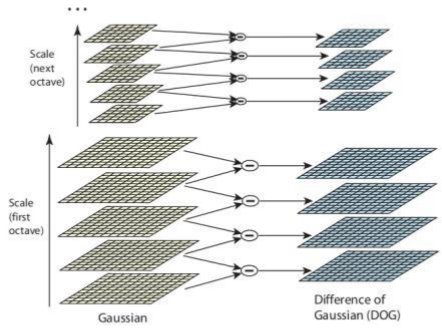
From Szeliski, *Computer Vision: Algorithms and Applications*

Example: SIFT

(Many Others: ORB, MSER, CNN/Deep Learning, etc.)

SIFT: Scale Pyramid

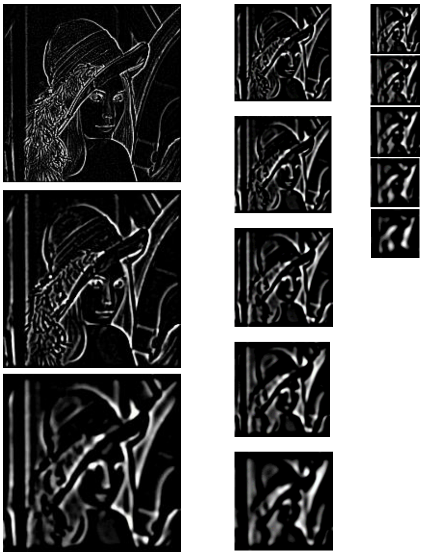
- D images are organised into a pyramid of progressively blurred images.
- Separated into octaves and scale levels per octave.
- Between octaves image is decimated by a factor of 2.

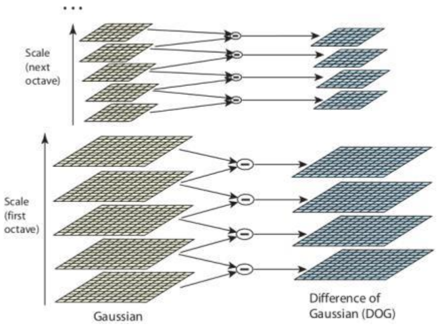


Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, *60*(2), 91-110.

METR 4202: **Robotics** September 13, 2017 -35

SIFT: Scale Pyramid





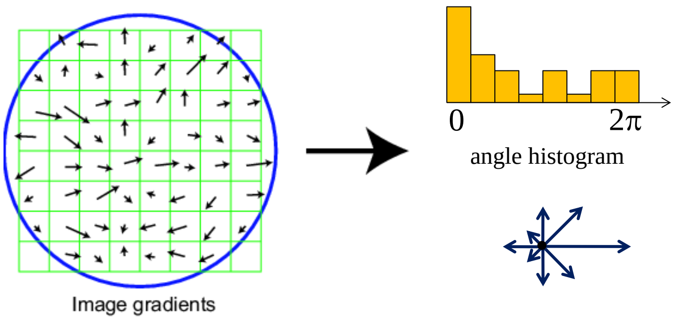
Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, *60*(2), 91-110.

METR 4202: **Robotics** September 13, 2017 -36

Scale Invariant Feature Transform

Basic idea:

- Take 16x16 square window around detected feature
- Compute edge orientation (angle of the gradient - 90°) for each pixel
- Throw out weak edges (threshold gradient magnitude)
- Create histogram of surviving edge orientations

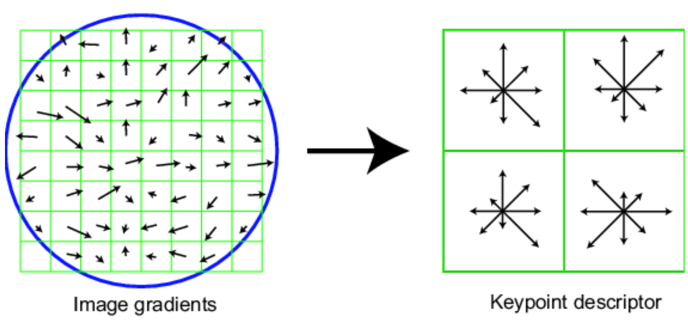


Adapted from slide by David Lowe

SIFT descriptor

Full version

- Divide the 16x16 window into a 4x4 grid of cells (2x2 case shown below)
- Compute an orientation histogram for each cell
- 16 cells * 8 orientations = 128 dimensional descriptor

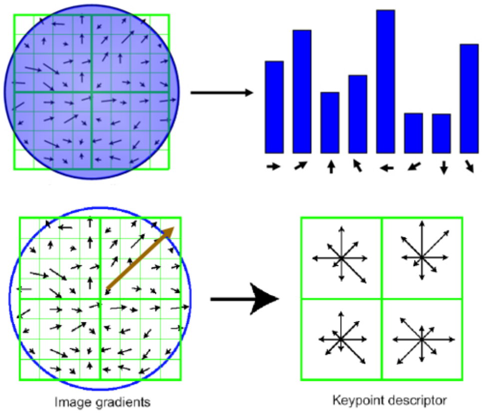


Adapted from slide by David Lowe

SIFT: Feature Description

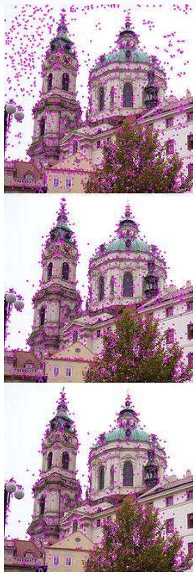
- Features are described using the pixel gradients in a 16x16 square centring on the feature point.
- These gradients are then segmented into 4x4 boxes. An 8 bin orientation histogram is created to define the box.

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, *60*(2), 91-110.



Scale Invariant Feature Transforms

- Goal was to define an algorithm to describe an image with features
- This would enable a number of different applications:
  - Feature Matching
  - Object / Image Matching
  - Orientation / Homography Resolution



Wikipedia: Scale Invariant Feature Transforms (2014)

SIFT: Feature Definition

- SIFT features are defined as the local extrema in a Difference of Gaussian (D) Scale Pyramid.

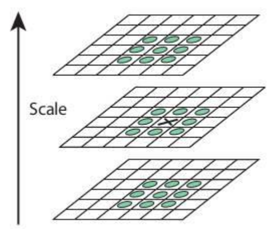
$$D(x, y, \sigma) = L(x, y, k_i\sigma) - L(x, y, k_i\sigma)$$

Where

$$L(x, y, k_i\sigma) = G(x, y, k\sigma) * I(x, y)$$

SIFT: Feature Detection

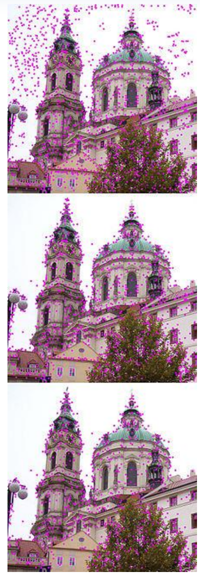
- Each scale level in the image is evaluated for features.
- A feature is defined as a local maximum or minimum.
- For efficiency the 26 surrounding points are evaluated.



Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, *60*(2), 91-110.

SIFT: Feature Reduction

- Initial feature detection over detects features descriptive of the image.
- Initially remove features with low contrast.
- Then evaluate features to remove any edge responses.



Wikipedia: Scale Invariant Feature Transforms (2014)

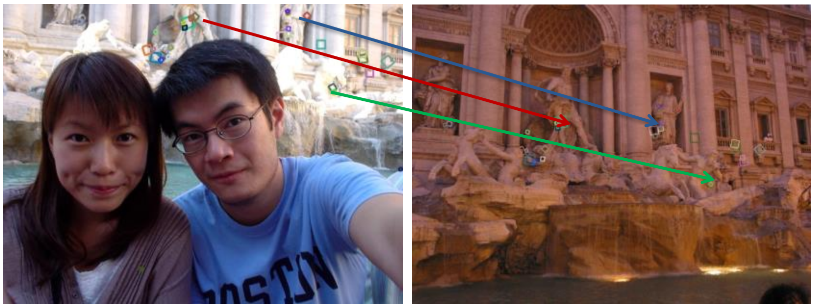
SIFT: Feature Matching

- A match is defined as a pair of features with the closest Euclidian distance to each other.
- Matches above a threshold are culled to improve match.

OpenCV: Feature Matching (2014)

Properties of SIFT

- Extraordinarily robust matching technique
  - Can handle changes in viewpoint
    - Up to about 60 degree out of plane rotation
  - Can handle significant changes in illumination
    - Sometimes even day vs. night (below)
  - Fast and efficient—can run in real time
  - Lots of code available
    - http://people.csail.mit.edu/albert/ladypack/wiki/index.php/Known_implementations_of_SIFT



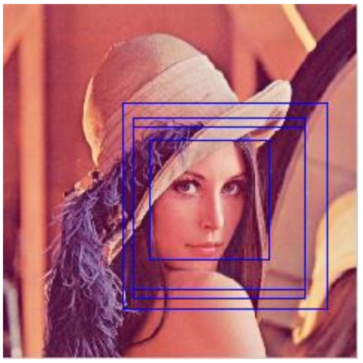
From David Lowe and Szeliski, *Computer Vision: Algorithms and Applications*

"Advanced" Examples:

Some "Learned Approaches"

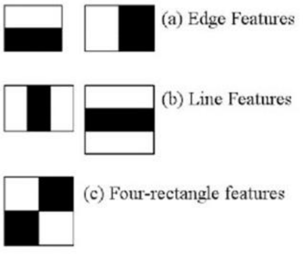
Boosted Cascade Haar-like Weak Classifiers

- Fast object detector designed primarily for use in face detection.
- Uses a cascade of weak classifiers to define object match.



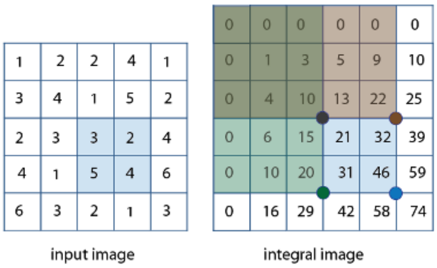
Viola Jones: Feature Definition

- Feature is classified as being the difference between the average intensity of two or more image sections.
- Can be any arithmetic combination of section values.



Viola Jones: Efficient Calculation of Features

- Fast calculation of the feature value is obtained by calculating the integral image.
- This leaves at most 4 sum operations to calculate a feature.



Viola Jones: Boosting

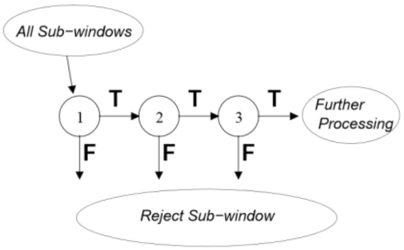
- Iteratively selects best classifier for detection.
- Assigns weights to each classifier to indicate likelihood of classifier indicating positive detection
- If the sum of the weights of positive classifier responses is above a threshold then there is a positive detection.

- Given example images $(x_1, y_1), \ldots, (x_n, y_n)$ where $y_i = 0, 1$ for negative and positive examples respectively.
- Initialize weights $w_{1,i} = \frac{1}{2m}, \frac{1}{2l}$ for $y_i = 0, 1$ respectively, where m and l are the number of negatives and positives respectively.
- For $t = 1, \ldots, T$:
  1. Normalize the weights,
  $$w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}}$$
  so that w_t is a probability distribution.
  2. For each feature, j, train a classifier h_j which is restricted to using a single feature. The error is evaluated with respect to w_t, $\epsilon_j = \sum_i w_i \left| h_j(x_i) - y_i \right|$.
  3. Choose the classifier , h_t, with the lowest error ϵ_t.
  4. Update the weights:
  $$w_{t+1,i} = w_{t,i} \beta_t^{1-e_i}$$
  where $e_i = 0$ if example x_i is classified correctly, $e_i = 1$ otherwise, and $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$.
- The final strong classifier is:
$$h(x) = \begin{cases} 1 & \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0 & \text{otherwise} \end{cases}$$
where $\alpha_t = \log \frac{1}{\beta_t}$

Viola, P., & Jones, M. (2001). Rapid object detection using a boosted cascade of simple features

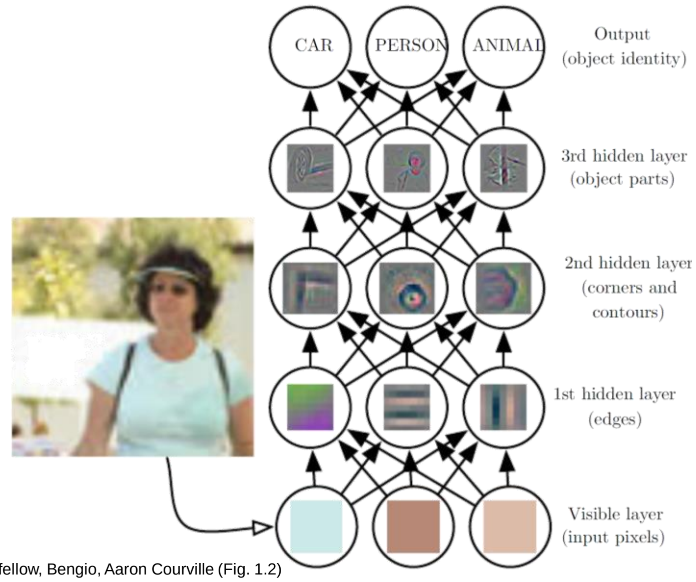
Viola Jones: Boosted Cascades

- Effective boosted classifiers require a high number of weak classifiers.
- However, simple low count classifiers offer high rejection rate.
- Solution is to use cascaded classifiers.



Viola, P., & Jones, M. (2001). Rapid object detection using a boosted cascade of simple features

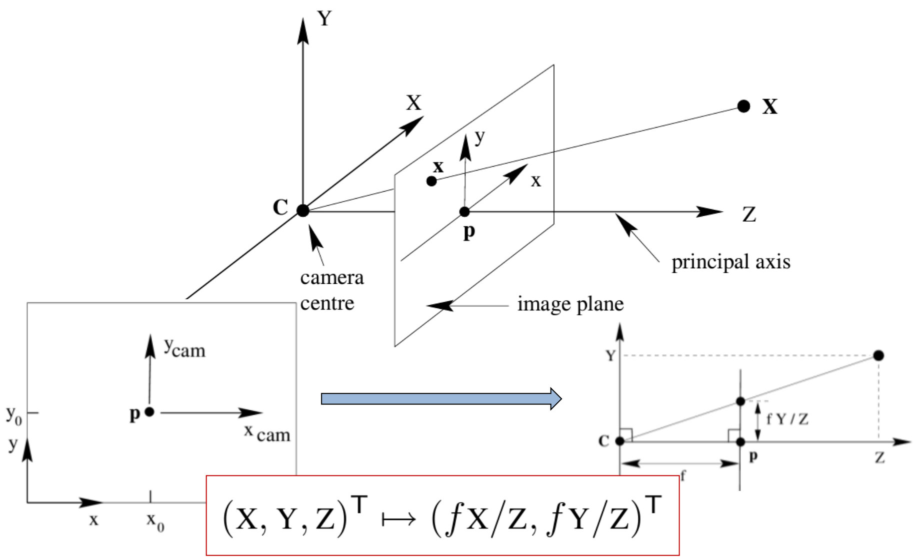
More Generally: A Transform is "just" a Different Representation



Source: Goodfellow, Bengio, Aaron Courville (Fig. 1.2)

Multiple View Geometry ("Notorious MVG")

Image Formation – Single View Geometry [I]



$$(X, Y, Z)^\mathsf{T} \mapsto (fX/Z, fY/Z)^\mathsf{T}$$

Hartly & Zisserman, Ch. 6

Image Formation – Single View Geometry [II]

➔ Camera Projection Matrix

$$\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \mapsto \begin{pmatrix} fX + Zp_x \\ fY + Zp_y \\ Z \end{pmatrix} = \begin{bmatrix} f & & p_x & 0 \\ & f & p_y & 0 \\ & & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

world — camera — Intrinsics

- x = Image point
- $\mathbf{X}$ = World point
- K = Camera Calibration Matrix

$$K = \begin{bmatrix} f & & p_x \\ & f & p_y \\ & & 1 \end{bmatrix}$$

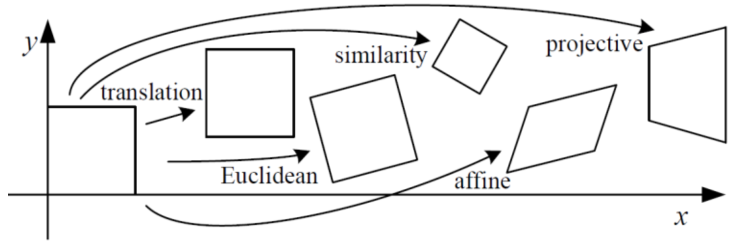
$$\mathbf{x} = K[I \mid \mathbf{0}]\mathbf{X}_{\text{cam}}.$$

➔Perspective Camera as:

$$P = K[R \mid \mathbf{t}]$$

where: P is **3**×4 and of **rank 3**

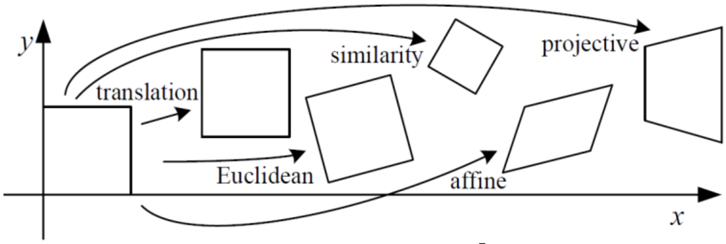
Transformations



- **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 ||)
- Projective:
  (straight lines preserved)
  H: Homogenous 3x3 Matrix

$$x' = \begin{bmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \end{bmatrix} \underline{x}$$

$$x' = \mathbf{H}\underline{x}$$

$$x' = \frac{h_{00}x + h_{01}y + h_{02}}{h_{20}x + h_{21}y + h_{22}}$$

$$y' = \frac{h_{10}x + h_{11}y + h_{12}}{h_{20}x + h_{21}y + h_{22}}$$

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

2-D Transformations

➔ x’ = point in the **new** (or 2nd) image

➔ x = point in the old image

- 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	

3D Transformations

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

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

Projection Models

- Orthographic
- Weak Perspective
- Affine
- Perspective
- Projective

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

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

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

$$\mathbf{\Pi} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \end{bmatrix}$$

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

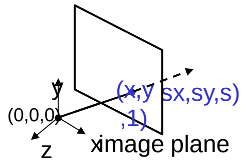
Slide from Szeliski, *Computer Vision: Algorithms and Applications*

Properties of Projection

- Preserves
  - Lines and conics
  - Incidence
  - Invariants (cross-ratio)
- Does not preserve
  - Lengths
  - Angles
  - Parallelism

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 Formation – Two-View Geometry [Stereopsis] ➔ Fundamental Matrix

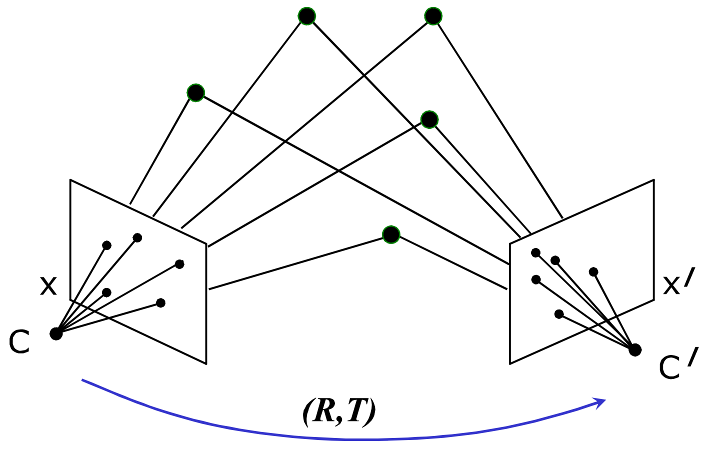
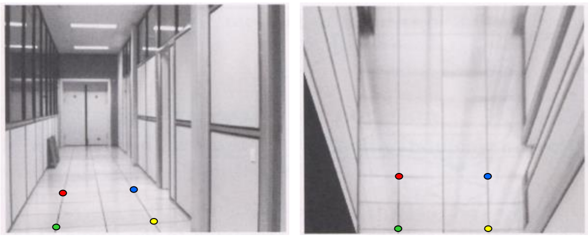


Image Rectification



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

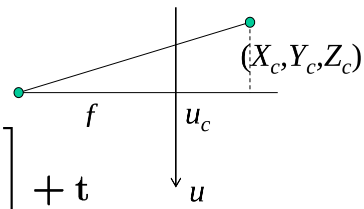
Slide from Szeliski, *Computer Vision: Algorithms and Applications*

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 (Image Formation Equations)



$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = \begin{bmatrix} \mathbf{R} \end{bmatrix}_{3\times3} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \mathbf{t}$$

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

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

3D → 2D Perspective Projection

- Matrix Projection (camera matrix):

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

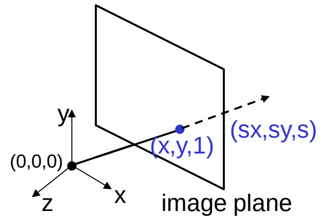
It's useful to decompose Π into $\mathbf{T} \rightarrow \mathbf{R} \rightarrow$ project $\rightarrow \mathbf{A}$

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

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

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

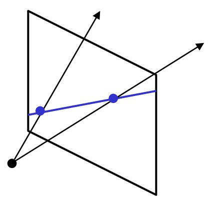


- Each *point* (x,y) on the plane is represented by a *ray* (sx,sy,s)
  - all points on the ray are equivalent: (x, y, 1) ≅ (sx, sy, s)

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

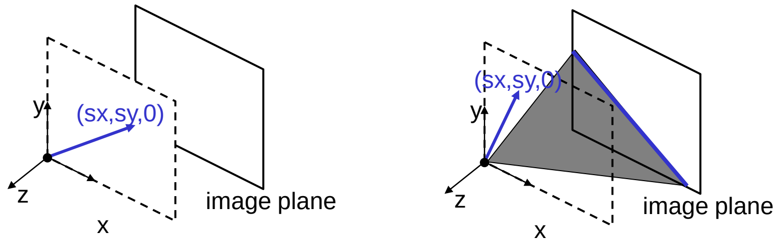
in vector notation: $$0 = \underbrace{\begin{bmatrix} a & b & c \end{bmatrix}}_{\mathbf{l}^T} \underbrace{\begin{bmatrix} x \\ y \\ z \end{bmatrix}}_{\mathbf{p}}$$

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

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

Ideal points and lines

- Ideal point ("point at infinity")
  - $p \cong (x, y, 0)$ – parallel to image plane
  - It has infinite image coordinates

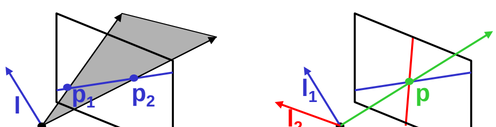


Line at infinity

- $l_\infty \cong (0, 0, 1)$ – parallel to image plane
- Contains all ideal points

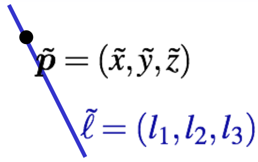
Point and Line Duality

- A line l is a homogeneous 3-vector (a ray)
- It is ⊥ to every point (ray) p on the line: lT p=0



- What is the line **l** spanned by rays $\mathbf{p_1}$ and $\mathbf{p_2}$?
  - **l** is ⊥ to $\mathbf{p_1}$ and $\mathbf{p_2}$ ⇒ $\mathbf{l} = \mathbf{p_1} \times \mathbf{p_2}$ (**l** is the plane normal)
- What is the intersection of two lines $\mathbf{l_1}$ and $\mathbf{l_2}$?
  - **p** is ⊥ to $\mathbf{l_1}$ and $\mathbf{l_2}$ ⇒ $\mathbf{p} = \mathbf{l_1} \times \mathbf{l_2}$
- Points and lines are *dual* in projective space
  - every property of points also applies to lines

Point and Line Duality [II]



Homogeneous ⇔ Cartesian

- Point:

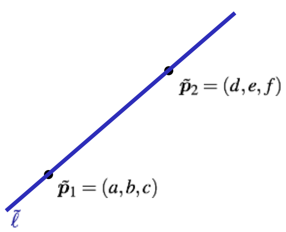
  $\tilde{P} = (\tilde{x}, \tilde{y}, \tilde{z})$ | $P = (x, y)$ $x = \frac{\tilde{x}}{\tilde{z}}, y = \frac{\tilde{y}}{\tilde{z}}$

- Line:
  - Is such that $\tilde{l}^T \tilde{p} = 0$
  - Point Eq of a line is: $y = mx + b$

Image/Notation from: Corke, Ch. 11

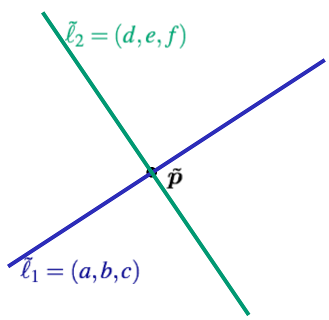
Point and Line Duality [III]

- 2 Points Make a Line



$$\tilde{\ell} = \tilde{p}_1 \times \tilde{p}_2$$

- 2 Lines Make Point!

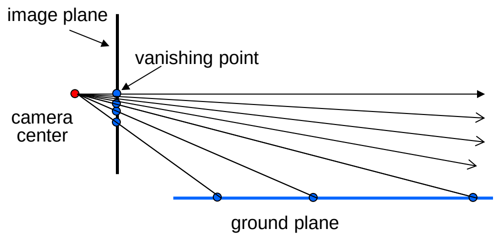


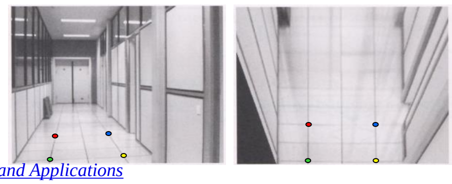
$$\tilde{p} = \tilde{\ell}_1 \times \tilde{\ell}_2$$

Image/Notation from: Corke, Ch. 11

Vanishing Points

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





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

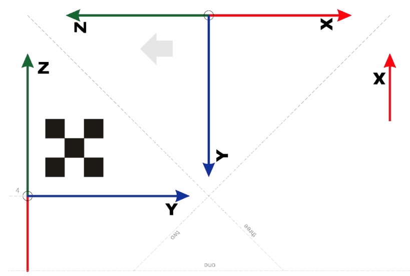
Extra ☺

Tips & Tricks!

SIFT / Corners for the {Frame} finder

To find the Frame, Consider:

- Structure
  - Corners
  - SIFT
  - ???
- Calibration Sequence
- Thought Experiment: How do you make this traceable back to the {camera frame}



UQ Robotics

Camera matrix calibration

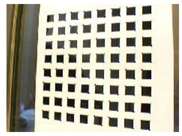
- Advantages:
  - very simple to formulate and solve
  - can recover K [R | t] from M using QR decomposition [Golub & VanLoan 96]
- Disadvantages:
  - doesn't compute internal parameters
  - more unknowns than true degrees of freedom
  - need a separate camera matrix for each new view

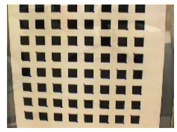
From Szeliski, *Computer Vision: Algorithms and Applications*

Multi-plane calibration

- Use several images of planar target held at unknown orientations [Zhang 99]
  - Compute plane homographies

$$\begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} \sim \mathbf{K} \begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{t} \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} \sim \mathbf{HX}$$





  - Solve for K-TK-1 from Hk's
    - 1plane if only f unknown
    - 2 planes if (f,uc,vc) unknown
    - 3+ planes for full K
  - Code available from Zhang and OpenCV

From Szeliski, *Computer Vision: Algorithms and Applications*