Roadmap: Basics of Digital Image Processing

- Images
- Point operators (ch. 3.1)
- Filtering: (ch. 3.2, ch 3.3, ch. 3.4) – main focus
  - Linear filtering
  - Non-linear filtering
- Fourier Transformation (ch. 3.4)
- Multi-scale image representation (ch. 3.5)
- Edges detection and linking (ch. 4.2)
- Line detection (ch. 4.3)
- Interest Point detection (ch. 4.1.1)
- Using multiple Images: Define Challenges
Reminder: Convolution

- Replace each pixel by a linear combination of its neighbours and itself.

- 2D convolution (discrete)

\[ g = f \ast h \]

\[
\begin{array}{cccccccc}
45 & 60 & 98 & 127 & 132 & 133 & 137 & 133 \\
46 & 65 & 98 & 123 & 126 & 128 & 131 & 133 \\
47 & 65 & 96 & 115 & 119 & 123 & 135 & 137 \\
47 & 63 & 91 & 107 & 113 & 122 & 138 & 134 \\
50 & 59 & 89 & 97 & 110 & 123 & 133 & 134 \\
49 & 53 & 68 & 83 & 97 & 113 & 128 & 133 \\
50 & 50 & 58 & 70 & 84 & 102 & 116 & 126 \\
50 & 50 & 52 & 58 & 69 & 86 & 101 & 120 \\
\end{array}
\]

\[
\begin{array}{cccc}
0.1 & 0.1 & 0.1 & \\
0.1 & 0.2 & 0.1 & \\
0.1 & 0.1 & 0.1 & \\
\end{array}
\]

\[
\begin{array}{cccccccc}
69 & 95 & 116 & 125 & 129 & 132 \\
68 & 92 & 110 & 120 & 126 & 132 \\
66 & 86 & 104 & 114 & 124 & 132 \\
62 & 78 & 94 & 108 & 120 & 129 \\
57 & 69 & 83 & 98 & 112 & 124 \\
53 & 60 & 71 & 85 & 100 & 114 \\
\end{array}
\]

\[ g(x, y) = \sum_{k,l} f(x - k, y - l)h(k, l) = \sum_{k,l} f(k, l)h(x - k, y - l) \]
Reminder – Linear Filters

\[
\frac{1}{K^2} \begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & 1 & \cdots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \cdots & 1 \end{bmatrix}
\quad \frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}
\quad \frac{1}{256} \begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}
\quad \frac{1}{8} \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}
\quad \frac{1}{4} \begin{bmatrix} 1 & -2 & 1 \\ -2 & 4 & -2 \\ 1 & -2 & 1 \end{bmatrix}
\]

\[
\frac{1}{K} \begin{bmatrix} 1 & 1 & \cdots & 1 \\ \frac{1}{4} & 1 & 2 & 1 \\ \frac{1}{16} & 1 & 4 & 6 & 4 & 1 \end{bmatrix}
\quad \frac{1}{2} \begin{bmatrix} -1 & 0 & 1 \\ \frac{1}{2} & 1 & -2 & 1 \end{bmatrix}
\]

(a) box, \( K = 5 \)  
(b) bilinear  
(c) “Gaussian”  
(d) Sobel  
(e) corner
Roadmap: Basics of Digital Image Processing

- Images
- Point operators (ch. 3.1)
- Filtering: (ch. 3.2, ch. 3.3, ch. 3.4) – main focus
  - Linear filtering
  - Non-linear filtering
- Fourier Transformation (ch. 3.4)
- Multi-scale image representation (ch. 3.5)
- Edges detection and linking (ch. 4.2)
- Line detection (ch. 4.3)
- Interest Point detection (ch. 4.1.1)
- Using multiple Images: Define Challenges
Gaussian Image Pyramid

- Represent Image at multiple resolution

High resolution

Low resolution
A naive approach

Take every second pixel – bad!

[From book: Computer Vision A modern Approach, Ponce and Forsyth]
Problem: Aliasing Effect

Problem: High frequencies (sharp transitions) are lost
Solution: Smooth before downsampling

\[ G_\rho^* \]

downsampling

\[ G_\rho^* \]

Gaussian smoothing
Application: template search

Search template: [Diagram]

[from Irani, Basri]
Application: Large Image Segmentation

Small image (100x100)

Small segmentation result

Large image, e.g. 10 MPixel

Trimap: created from small image

Segmentation large image

Banded Segmentation
Application: Large Label Space

Approach:
1. solve problem on small image \((x5\ downscaled)\) with \(40 \times 40\) label space \((\text{coarse motion})\) (1600 labels)
2. Do on full resolution only in \(5\times5\) neighbourhood around each solution \((\text{add fine motion})\) (25 labels)

(color coding)

2 images
(overlaid)

Motion
200 \(x\) 200 possible
discrete movements
(40,000 labels)

Color coding

(200 \(x\) 200 possible discrete movements)

(40,000 labels)

(problem small objects can get lost)
Roadmap: Basics of Digital Image Processing

- Images
- Point operators (ch. 3.1)
- Filtering: (ch. 3.2, ch 3.3, ch. 3.4) – main focus
  - Linear filtering
  - Non-linear filtering
- Fourier Transformation (ch. 3.4)
- Multi-scale image representation (ch. 3.5)
- Edges detection and linking (ch. 4.2)
- Line detection (ch. 4.3)
- Interest Point detection (ch. 4.1.1)
- Using multiple Images: Define Challenges
What do we want:

• **Good detection**: we want to find edges not noise
• **Good localization**: find true edge
• **Single response**: one per edge (independent of edge sharpness)
• **Long edge-chains**
Idealized edge types

We focus on this
What are edges?

- correspond to fast changes in the image
- The magnitude of the derivative is large

Image of 2 step edges

Image of 2 ramp edges

Slice through the image

Slice through the image
What are fast changes in the image?

Image

Scanline 250

Scanline 250 smoothed with Gaussian

Texture or many edges?

Edges defined after smoothing
Edges and Derivatives

We will look at this first.
Edge filters in 1D

\[
\frac{d}{dx} f(x) = \lim_{h \to 0} \frac{f(x + h) - f(x)}{h} \approx f(x + 1) - f(x)
\]

We can implement this as a linear filter:

Forward differences:
\[
\frac{1}{2} \begin{bmatrix} -1 & 1 \end{bmatrix}
\]

Central differences:
\[
\frac{1}{2} \begin{bmatrix} -1 & 0 & 1 \end{bmatrix}
\]
Reminder: Separable Filters

\[ \frac{1}{K^2} \]
\[ \begin{array}{ccc}
1 & 1 & \cdots & 1 \\
1 & 1 & \cdots & 1 \\
\vdots & \vdots & \ddots & \vdots \\
1 & 1 & \cdots & 1 \\
\end{array} \]
\[ \frac{1}{16} \]
\[ \begin{array}{ccc}
1 & 2 & 1 \\
2 & 4 & 2 \\
\vdots & \vdots & \vdots \\
1 & 2 & 1 \\
\end{array} \]
\[ \frac{1}{256} \]
\[ \begin{array}{cccc}
1 & 4 & 6 & 4 \\
4 & 16 & 24 & 16 \\
6 & 24 & 36 & 24 \\
4 & 16 & 24 & 16 \\
1 & 4 & 6 & 4 \\
\end{array} \]
\[ \frac{1}{8} \]
\[ \begin{array}{ccc}
-1 & 0 & 1 \\
-2 & 0 & 2 \\
-1 & 0 & 1 \\
\end{array} \]
\[ \frac{1}{4} \]
\[ \begin{array}{ccc}
1 & -2 & 1 \\
-2 & 4 & -2 \\
1 & -2 & 1 \\
\end{array} \]

**This is the centralized differences operator**

(a) box, \( K = 5 \)

(b) bilinear

(c) “Gaussian”

(d) Sobel

(e) corner
Edge Filter in 1D: Example

Based on 1st derivative

- **Smooth** with Gaussian – to filter out noise
- Calculate derivative
- Find its optima

\[ f \]

\[ g \]

\[ g \ast f \]

\[ \frac{d}{dx} (g \ast f) \]
Simplification:
(saves one operation)

\[
\frac{d}{dx} (g \ast f) = \left( \frac{d}{dx} g \right) \ast f
\]

Derivative of Gaussian
Edge Filtering in 2D

- Derivative in x-direction: \[ D_x \ast (G \ast I) = (D_x \ast G) \ast I \]

  - in 1D:

  - in 2D:
Edge Filter in 2D: Example

Is this $I_x$ or $I_y$?
Is the sign right?
Edge Filter in 2D

x-derivatives with different Gaussian smoothing
What is a gradient

\[
\left( \frac{\partial I}{\partial x}, \frac{\partial I}{\partial y} \right) = (k, 0)
\]

no change

change
What is a gradient

\[
\left( \frac{\partial I}{\partial x}, \frac{\partial I}{\partial y} \right) = (0, k)
\]
What is a gradient

\[
\left( \frac{\partial I}{\partial x}, \frac{\partial I}{\partial y} \right) = (k_x, k_y)
\]

- gradient direction is perpendicular to edge
- gradient magnitude measures edge strength
What is a Gradient

- the gradient is:

\[ \nabla I = (I_x, I_y) = \left( \frac{\partial I}{\partial x}, \frac{\partial I}{\partial y} \right) \]

- the magnitude of the gradient is:

\[ ||\nabla I|| = \sqrt{I_x^2 + I_y^2} \]

- the direction of the gradient is:

\[ \theta = \text{atan}(I_y, I_x) \]
Our goal was to get thin edge chains?
How to get edge chains

Rough Outline of a good edge detector (such as Canny)

1. Compute robust Gradient Image: \((D_x \ast G) \ast I, (D_y \ast G) \ast I\)
2. Find edge-points ("edgels"): non-maximum suppression
3. Link-up edge-points to get chains
4. Do hysteresis to clean-up chains
Non-maximum surppression

Wich pixel is an edge point? (non-max surpression)

1. Check if pixel is local maximum at gradient orientation (interpolate values for p,r)
2. Accept edge-point if above a threshold
Link up edge-points to get chains

1. Link-up neighbouring pixels if both are edge-points.
Clean up chains with Hysteresis

Keep a chain:

High start threshold

Low threshold along the chain
Final Result

Image

Not much smoothing (fine scale)

much smoothing (coarse scale)
small threshold

much smoothing (coarse scale)
large threshold
Alterative: Edge detection with Laplacian Filter

- The Laplacian:
  \[ \nabla^2 I = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2} \]

- just another linear filter:
  \[ \nabla^2 (G \ast I) = (\nabla^2 G) \ast I \]

  Called Laplacian of Gaussian (LoG)

- in 1D:
- in 2D (mexican hat):
Laplacian example in 1D

- using Laplacian

$$\left( \frac{d^2}{dx^2} g \right) * f$$

LoG - Laplacian of Gaussian operator

Find zero-crossing
Approximate LoG with Difference of Gaussian (DoG)

Solid Line: Difference of Gaussian (DoG)

Dashed Line: Laplacian of Gaussian
Laplacian example in 2D

sigma = 4

sigma = 2
Future Lecture: Segmentation

- So far we looked at “jumps” in gray-scale images?
- Humans perceive edges very differently (edges depend on semantic)

“average human drawing”

Hard for computer vision method which operates only locally!

[from Martin, Fowlkes and Mail 2004]
Try to learn semantically meaningful image edges

• Features: brightness gradient; color gradient; texture gradient; weighted combination, etc.
Roadmap: Basics of Digital Image Processing

• Images
• Point operators (ch. 3.1)
• Filtering: (ch. 3.2, ch 3.3, ch. 3.4) – main focus
  • Linear filtering
  • Non-linear filtering
• Fourier Transformation (ch. 3.4)
• Multi-scale image representation (ch. 3.5)
• Edges detection and linking (ch. 4.2)
  • Line detection (ch. 4.3)
• Interest Point detection (ch. 4.1.1)
• Using multiple Images: Define Challenges
Hough voting with edgels (edge-points)

Algorithm:

1. Empty cells in Hough Space
2. Put for each Edgel($\theta, r$) into a cell of the Hough Space
3. Find Peaks in Hough Space (use non-max suppression)
4. Re-fit all edgels to a single line
Hough Voting: original

Goal: find all lines

Image with points

Goal: find all points with many “votes” in accumulator space

Hough transform

All lines that go through the 3 points

Image with just 3 points

This idea of transformation to a voting space can be used for many scenarios
Hough transform: original

[From Wikipedia]
Example: Orthogonal Vanishing point detection

846 line segments found

Algorithm: RANSAC (explained later)
Application: Camera Calibration (see later)

Found 3 orthogonal vanishing points

[Rother, Thesis ‘03]
Roadmap: Basics of Digital Image Processing

• Images
• Point operators (ch. 3.1)
• Filtering: (ch. 3.2, ch 3.3, ch. 3.4) – main focus
  • Linear filtering
  • Non-linear filtering
• Fourier Transformation (ch. 3.4)
• Multi-scale image representation (ch. 3.5)
• Edges detection and linking (ch. 4.2)
• Line detection (ch. 4.3)
• Interest Point detection (ch. 4.1.1)
• Using multiple Images: Define Challenges
What region should we try to match?

We want to find a few regions where this image pair matches: Applications later

Look for a region that is unique, i.e. not ambiguous
Goal: Interest Point Detection

• Goal: predict a few “interest points” in order to remove redundant data efficiently

• Should be invariant against:

  a. Geometric transformation – scaling, rotation, translation, affine transformation, projective transformation etc.

  b. Color transformation – additive (lightning change), multiplicative (contrast), linear (both), monotone etc.;

  c. Discretization (e.g. spatial resolution, focus);
Points versus Lines

„Aperture problem“

Lines are not as good as points
Local measure of **feature uniqueness:**
Shifting the window in any direction: how does it change

- **“flat” region:** no change in all directions
- **“edge”:** no change along the edge direction
- **“corner”:** significant change in all directions

[Shift left, Shift top, left]

[Szeliski and Seitz]
Harris Detector

How similar is the image \( I(x, y) \) to itself?

**Autocorrelation function:**

\[
c(x, y, \Delta x, \Delta y) = \sum_{(u,v) \in W(x,y)} w(u,v) \left( I(u, v) - I(u+\Delta x, v+\Delta y) \right)^2
\]

\( W(x, y) \) is a small vicinity (window) around \((x, y)\)

\( w(u, v) \) is a convolution kernel, used to decrease the influence of pixels far from \((x, y)\), e.g. the Gaussian \( \exp \left[ -\frac{(u-x)^2+(v-y)^2}{2\sigma^2} \right] \)

For simplicity we use \( w(u, v) = 1 \)
Harris Detector

\[ c(x, y, \Delta x, \Delta y) = \sum_{(u,v) \in W(x,y)} \left( I(u, v) - I(u+\Delta x, v+\Delta y) \right)^2 \]

One is interested in **properties** of \( c(x, y, \Delta x, \Delta y) \) at each position \((x, y)\)

Let us look at a linear approximation of Taylor expansion around \((u, v)\)

\[
I(u+\Delta x, v+\Delta y) = I(u, v) + \frac{\partial I(u, v)}{\partial x} \Delta x + \frac{\partial I(u, v)}{\partial y} \Delta y + \epsilon(\Delta x, \Delta y)
\]

\[
\cong I(u, v) + [I_x(u, v), I_y(u, v)] \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix}
\]

Gradient at \((u, v)\)
Harris Detector

Put it together:

\[
c(x, y, \Delta x, \Delta y) = \sum_{(u,v) \in W(x,y)} \left( I(u, v) - I(u+\Delta x, v+\Delta y) \right)^2
\]

\[
\approx \sum_{(u,v) \in W(x,y)} (\begin{bmatrix} I_x(u, v) & I_y(u, v) \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix})^2
\]

\[
= [\Delta x, \Delta y] Q(x, y) \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix}
\]

**Q: Structure Tensor**

with

\[
Q(x, y) = \begin{bmatrix}
\sum_W I_x(u, v)^2 & \sum_W I_x(u, v)I_y(u, v) \\
\sum_W I_x(u, v)I_y(u, v) & \sum_W I_y(u, v)^2
\end{bmatrix} = \begin{bmatrix} A & B \\ B & C \end{bmatrix}
\]

We compute this at any image location \((x, y)\)
The autocorrelation function

\[ c(x, y, \Delta x, \Delta y) = [\Delta x, \Delta y] Q(x, y) \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} \]

Function \( c \) is (after approximation) a **quadratic** function in \( \Delta x \) and \( \Delta y \)

- Isolines are ellipses (\( Q(x, y) \) is symmetric and positive definite);
- Eigenvector \( x_1 \) with (larger) Eigenvalue \( \lambda_1 \) is the direction of fastest change in function \( c \);
- Eigenvector \( x_2 \) with (smaller) Eigenvalue \( \lambda_2 \) is direction of slowest change in function \( c \).

Note \( c = 0 \) for \( \Delta x = \Delta y = 0 \)
Some examples – isolines for $c(x, y, \Delta x, \Delta y) = 1$:

(a) Flat

(b) Edges

(c) Corners

a. Homogenous regions: both $\lambda$-s are small

b. Edges: one $\lambda$ is small the other one is large

c. Corners: both $\lambda$-s are large (this is what we are looking for!)
Harris Detector

Image | \( \lambda_1 \) larger eigenvalue | \( \lambda_2 \) smaller eigenvalue |

Zoom in
Harris detector

“Cornerness” is a characteristic of $Q(x, y)$

$$\lambda_1 \lambda_2 = \text{det} \ Q(x, y) = AC - B^2, \quad \lambda_1 + \lambda_2 = \text{trace} \ Q(x, y) = A + C$$

Proposition by Harris: $H = \lambda_1 \lambda_2 - 0.04(\lambda_1 + \lambda_2)^2$

Downweights edges where $\lambda_1 \gg \lambda_2$
Harris Corners - example
h-score (red- high, blue - low)
Threshold (H-score > value)
Non-maximum suppression
Harris corners in red
Other examples
Maximally stable extremal regions

- Invariant to affine transformation of gray-values
- Both small and large structures are detected
There is a large body of literature on detectors and descriptors (later lecture)

A comparison paper
(e.g. what is the most robust corner detectors):

Roadmap: Basics of Digital Image Processing

• Images
• Point operators (ch. 3.1)
• Filtering: (ch. 3.2, ch 3.3, ch. 3.4) – main focus
  • Linear filtering
  • Non-linear filtering
• Fourier Transformation (ch. 3.4)
• Multi-scale image representation (ch. 3.5)
• Edges detection and linking (ch. 4.2)
• Line detection (ch. 4.3)
• Interest Point detection (ch. 4.1.1)
• Using multiple Images: Define Challenges
Difference between Appearance and Geometry

Appearance based matching:

Geometry based matching:
1) Assume sensible camera model. We will see that: Given 4 matching points on a surface defines how other points on the surface will match
The 3D case

Illustration of the general 3D case

**Appearance** based matching:

**Geometry** based matching:

1) Assume sensible camera model. We will see that: Given 7 matching 3D points defines how other 3D points match.
Sparse versus Dense Matching: Tasks and Applications

Tasks:
• Find places where we could match features (points, lines, regions, etc)
• Extract appearance - features descriptors
• Find all possible (putative) appearance matches between images
• Verify with geometry

For what applications is sparse matching enough:
• Sparse 3D reconstruction of a rigid scene
• Panoramic stitching of a rotating / translating camera
Building Rome on a cloudless day

The old city of Dubrovnik

[Frahm et al. ECCV ’10]
Sparse versus **Dense** Matching

Kinect RGB and Depth data input

Dense flow:
frame 1 $\rightarrow$ 2

Dense flow:
frame 2 $\rightarrow$ 1

3D view interpolation

Flow encoding
Tasks (all in one):
• Find for each pixel the 2D/3D/6D displacement (using both appearance and geometry)
• Find points which are occluded

For what applications is dense matching needed:
• Dense reconstruction of rigid scene
• 3D reconstruction of a non-rigid scene
A road map for the next five lectures

• L4: Geometry of a Single Camera and Image Formation Process

• L5: Sparse Matching two images: Appearance

• L6: Sparse Matching two images: Geometry

• L7: Sparse Reconstructing the world (Geometry of n-views)

• L8: Dense Geometry estimation (stereo, flow and scene flow, registration)
Outlook – matching 2 Images (appearance & geometry)

• Find interest points (including different scales)
• Find orientated patches around interest points to capture appearance
• Encode patch in a descriptor
• Find matching patches according to appearance (similar descriptors)
• Verify matching patches according to geometry
Reading for next class

This lecture:
- Chapter 3.5: multi-scale representation
- Chapter 4.2 and 4.3 - Edge and Line detection
- Chapter 4.1.1 Interest Point Detection

Next lecture:
- Chapter 2 (in particular: 2.1, 2.2) – Image formation process
- And a bit of Hartley and Zisserman – chapter 2