Spring 2022 Principles of Safe Autonomy
Lecture 4:
Basic Perception -> Edge Detection

Sayan Mitra
slides adapted from Svetlana Lazebnik
Autonomy pipeline

Sensing
Physics-based models of camera, LIDAR, RADAR, GPS, etc.

Perception
Programs for object detection, lane tracking, scene understanding, etc.

Decisions and planning
Programs and multi-agent models of pedestrians, cars, etc.

Control
Dynamical models of engine, powertrain, steering, tires, etc.

GEM platform

Control
Dynamical models of engine, powertrain, steering, tires, etc.

Decisions and planning
Programs and multi-agent models of pedestrians, cars, etc.

Perception
Programs for object detection, lane tracking, scene understanding, etc.

Sensing
Physics-based models of camera, LIDAR, RADAR, GPS, etc.
Perception

Programs for object detection, lane tracking, scene understanding, etc.

Perception subsystem converts signals from the environment into meaningful bits
Perception: EM to objects

Problem: Process electromagnetic radiation from the environment to construct a model of the world, so that the constructed model is close to the real world.

Challenging for computers: millions of years of evolution.

Ill-defined problem: impossibility of defining meaning “car”, “bicycle”, etc.
A practical perception pipeline in an AV has many pieces

This architecture from a slide from M. James of Toyota Research Institute, North America
Outline

- Linear filtering
- Edge detection
Motivation: Image denoising

• How can we reduce noise in a photograph?
Image representation

Images are represented as 2D arrays of pixels. Each pixel is represented by (array of) value(s) representing its color.

```python
# read an image
img = cv2.imread('images/noguchi02.jpg')

# show image format (basically a 3-d array of pixel color info, in BGR format)
print(img)
```

Where [72 99 143] is the blue, green, and red values of that pixel.

We will work with grayscale images

Denote by img[i,j] (or f[i,j]) the value of the i,j-th pixel
What is filtering?

Modify the pixels in an image based on some function of a local neighborhood of the pixels.

**Bright**($img, k$): for all $i, j$

$$img'[i][j] = k * img[i][j]$$

**Shifting right by** $s$ **Shift**($img, s$):

$$img'[k] = img[k-s]; \text{img}'[0]...img'[s-1] \text{ is undefined}$$

**Simplest: Linear filtering**

replace each pixel by a linear combination of neighbors
Moving average

• Let’s replace each pixel with a *weighted* average of its neighborhood
• The weights are called the *filter kernel*
• What are the weights for the average of a 3x3 neighborhood?

```
1 1 1
1 1 1
1 1 1
```

"box filter"

Source: D. Lowe
Convolution

Convolution mask $g[,]$

Output or convolved image

$f = g \ast \text{img}$

$$f[i,j] = g[1,1] \text{img}[i-1,j-1] + g[1,2] \text{img}[i-1,j] + g[1,3] \text{img}[i-1,j+1] + g[2,1] \text{img}[i,j-1] + g[2,2] \text{img}[i,j] + g[2,3] \text{img}[i,j+1] + g[3,1] \text{img}[i+1,j-1] + g[3,2] \text{img}[i+1,j] + g[3,3] \text{img}[i+1,j+1]$$
Defining convolution

• Let $f$ be the image and $g$ be the kernel. The output of convolving $f$ with $g$ is denoted $f \ast g$.

$$(f \ast g)[m, n] = \sum_{k,l} f[m-k, n-l] g[k, l]$$

Convention: kernel is “flipped”

Source: F. Durand
For analysis we will work with 1D images

• Let $f$ be the image and $g$ be the kernel. The output of convolving $f$ with $g$ is denoted $f \ast g$.

\[(f \ast g)[m] = \sum_k f[m - k]g[k]\]
Key properties: Prove the first two

• **Shift invariance:** same behavior regardless of pixel location:
\[ \text{filter(shift}(f) = \text{shift(filter}(f)) \]

• **Linearity:**
\[ \text{filter}(f_1 + f_2) = \text{filter}(f_1) + \text{filter}(f_2) \]

• **Theoretical result:** any linear shift-invariant operator can be represented as a convolution
Properties in more detail

- **Commutative:** \( a * b = b * a \)
  - Conceptually no difference between filter and signal

- **Associative:** \( a * (b * c) = (a * b) * c \)
  - Often apply several filters one after another: \(((a * b_1) * b_2) * b_3)\)
  - This is equivalent to applying one filter: \(a * (b_1 * b_2 * b_3)\)

- **Distributes over addition:** \( a * (b + c) = (a * b) + (a * c) \)

- **Scalars factor out:** \( ka * b = a * kb = k (a * b) \)

- **Identity:** unit impulse \( e = [\ldots, 0, 0, 1, 0, 0, \ldots] \), \( a * e = a \)
openCV: **filter2D**

Output image same size as input

Multi-channel: each channel is processed independently

Extrapolation of border

Examples
Practice with linear filters

Original

\[
\begin{bmatrix}
0 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 0 \\
\end{bmatrix}
\]
Practice with linear filters

Original

Filtered (no change)

Source: D. Lowe
Practice with linear filters

Original

![Original Image]

Source: D. Lowe
Practice with linear filters

Original

Shifted left
By 1 pixel

Source: D. Lowe
Practice with linear filters

Original

\[
\begin{array}{ccc}
1 & 1 & 1 \\
1 & 1 & 1 \\
1 & 1 & 1 \\
\end{array}
\]

1/9
Practice with linear filters

Source: D. Lowe
Practice with linear filters

Original

(Note that filter sums to 1)

Source: D. Lowe
Practice with linear filters

Original

Sharpening filter - Accentuates differences with local average

Source: D. Lowe
Sharpening

before

after

Source: D. Lowe
Sharpening

What does blurring take away?

Let’s add it back:
Smoothing with box filter revisited

• What’s wrong with this picture?
• What’s the solution?
Smoothing with box filter revisited

- What’s wrong with this picture?
- What’s the solution?
  - To eliminate edge effects, weight contribution of neighborhood pixels according to their closeness to the center

“fuzzy blob”
Gaussian Kernel

\[ G_\sigma = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}} \]

Constant factor at front makes volume sum to 1 (can be ignored when computing the filter values, as we should renormalize weights to sum to 1 in any case)

Source: C. Rasmussen
Gaussian Kernel

\[
G_\sigma = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}
\]

Standard deviation \( \sigma \): determines extent of smoothing

Source: K. Grauman
Choosing kernel width

The Gaussian function has infinite support, but discrete filters use finite kernels.
Choosing kernel width

Rule of thumb: set filter half-width to about $3\sigma$
Gaussian vs. box filtering
Gaussian filters

- Remove high-frequency components from the image (*low-pass filter*)
- Convolution with self is another Gaussian
  - So can smooth with small-\(\sigma\) kernel, repeat, and get same result as larger-\(\sigma\) kernel would have
  - Convolving two times with Gaussian kernel with std. dev. \(\sigma\) is same as convolving once with kernel with std. dev. \(\sigma\sqrt{2}\)
- *Separable* kernel
  - Factors into product of two 1D Gaussians
  - Discrete example:
    \[
    \begin{bmatrix}
    1 & 2 & 1 \\
    2 & 4 & 2 \\
    1 & 2 & 1 \\
    \end{bmatrix}
    =
    \begin{bmatrix}
    1 \\
    2 \\
    1 \\
    \end{bmatrix}
    \begin{bmatrix}
    1 & 2 & 1 \\
    \end{bmatrix}
    \]
Separability of the Gaussian filter

\[
G_\sigma(x, y) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right)
\]

\[
= \left(\frac{1}{\sqrt{2\pi\sigma}} \exp\left(-\frac{x^2}{2\sigma^2}\right)\right) \left(\frac{1}{\sqrt{2\pi\sigma}} \exp\left(-\frac{y^2}{2\sigma^2}\right)\right)
\]

The 2D Gaussian can be expressed as the product of two functions, one a function of \(x\) and the other a function of \(y\).

In this case, the two functions are the (identical) 1D Gaussian.
Why is separability useful?

• Separability means that a 2D convolution can be reduced to two 1D convolutions (one along rows and one along columns)
• What is the complexity of filtering an $n \times n$ image with an $m \times m$ kernel?
  • $O(n^2 m^2)$
• What if the kernel is separable?
  • $O(n^2 m)$
Noise

- **Salt and pepper noise:** contains random occurrences of black and white pixels
- **Impulse noise:** contains random occurrences of white pixels
- **Gaussian noise:** variations in intensity drawn from a Gaussian normal distribution

Source: S. Seitz
Reducing salt-and-pepper noise

What’s wrong with the results?

3x3  5x5  7x7
Alternative idea: Median filtering

- A **median filter** operates over a window by selecting the median intensity in the window.

\[\begin{array}{ccc}
10 & 15 & 20 \\
23 & 90 & 27 \\
33 & 31 & 30 \\
\end{array}\]

- Is median filtering linear?
Median filter

Salt-and-pepper noise  Median filtered

open cv: cv2.medianBlur (input, output,ksize)
Median filter

- Is median filtering linear?
- Let’s try filtering

\[
\begin{array}{ccc}
1 & 1 & 1 & 0 & 0 & 0 \\
1 & 1 & 2 & + & 0 & 1 & 0 \\
2 & 2 & 2 & & 0 & 0 & 0 \\
\end{array}
\]
Edge detection

Winter in Kraków photographed by Marcin Ryczek
Edge detection

- **Goal:** Identify sudden changes (discontinuities) in an image
- Intuitively, edges carry most of the semantic and shape information from the image
  - E.g., Lanes, traffic signs, cars

Sources: D. Lowe and S. Seitz
Median filter

• What advantage does median filtering have over Gaussian filtering?
  - Robustness to outliers

  filters have width 5:

<table>
<thead>
<tr>
<th></th>
<th>INPUT</th>
<th>MEDIAN</th>
<th>MEAN</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><img src="image" alt="" /></td>
<td><img src="image" alt="" /></td>
<td><img src="image" alt="" /></td>
</tr>
</tbody>
</table>
Gaussian vs. median filtering

Gaussian

3x3

5x5

7x7

Median
Review: Image filtering

- Convolution
- Box vs. Gaussian filter
- Separability
- Median filter
Outline

• Filtering
  • Convolution: Linearity, shift invariance, associativity, commutativity, …
  • Kernels: Gaussian, box, …
  • Separability of Gaussian
  • Median filter

• Today: Edge detection
• Object recognition: Classification
Edge detection

An edge is a place of rapid change in the image intensity function.

- **Image**: A visual representation of an image.
- **Intensity function (along horizontal scanline)**: A line plot showing the intensity values along a horizontal line in the image. Edges correspond to extrema of the derivative.
- **First derivative**: A line plot showing the first derivative of the intensity function. Peaks and valleys indicate edges.

Edges correspond to extrema of the derivative.
Derivatives with convolution

For 2D function \( f(x,y) \), the partial derivative w.r.t \( x \) is:

\[
\frac{\partial f(x, y)}{\partial x} = \lim_{\varepsilon \to 0} \frac{f(x + \varepsilon, y) - f(x, y)}{\varepsilon}
\]

For discrete data, we can approximate using finite differences:

\[
\frac{\partial f(x, y)}{\partial x} \approx \frac{f(x + 1, y) - f(x, y)}{1}
\]

To implement the above as convolution, what would be the associated filter?
Convolution

convolution
mask $g[,]$

Output or convolved image
$f = g \ast \text{img}$

$f[i,j] = -1.\text{img}[i,j-1] + 1.\text{img}[i,j]$
Partial derivatives of an image

Which shows changes with respect to $x$?
Finite difference filters

Other approximations of derivative filters exist:

Prewitt: \[ M_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}; \quad M_y = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix} \]

Sobel: \[ M_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}; \quad M_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \]

Roberts: \[ M_x = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}; \quad M_y = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \]
Image gradient

The gradient of an image: \[ \nabla f = \left[ \frac{\partial f}{\partial x}, \frac{\partial f}{\partial y} \right] \]

The gradient points in the direction of most rapid increase in intensity

• How does this direction relate to the direction of the edge?

The gradient direction is given by \[ \theta = \tan^{-1} \left( \frac{\partial f}{\partial y} / \frac{\partial f}{\partial x} \right) \]

The edge strength is given by the gradient magnitude (norm)

\[ \|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2} \]
Effects of noise

Consider a single row or column of the image

\[ f(x) \]

\[ \frac{d}{dx} f(x)_0 \]

Where is the edge?

Source: S. Seitz
Solution: smooth first

- To find edges, look for peaks in $\frac{d}{dx}(f * g)$
Derivative theorem of convolution

- Differentiation is convolution, and convolution is associative:
  \[
  \frac{d}{dx} (f * g) = f * \frac{d}{dx} g
  \]

- This saves us one operation:

Source: S. Seitz
Derivative of Gaussian filters

Which one finds horizontal/vertical edges?
Derivative of Gaussian filters

Are these filters separable?

\( x\)-direction

\( y\)-direction
Recall: Separability of the Gaussian filter

\[ G_\sigma(x, y) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right) \]

\[ = \left( \frac{1}{\sqrt{2\pi\sigma}} \exp\left(-\frac{x^2}{2\sigma^2}\right) \right) \left( \frac{1}{\sqrt{2\pi\sigma}} \exp\left(-\frac{y^2}{2\sigma^2}\right) \right) \]

The 2D Gaussian can be expressed as the product of two functions, one a function of \( x \) and the other a function of \( y \).

In this case, the two functions are the (identical) 1D Gaussian.
Smoothed derivative removes noise, but blurs edge
Also finds edges at different “scales”

Source: D. Forsyth
Review: Smoothing vs. derivative filters

**Smoothing filters**
- Gaussian: remove “high-frequency” components; “low-pass” filter
- Can the values of a smoothing filter be negative?
- What should the values sum to?
  - **One**: constant regions are not affected by the filter

**Derivative filters**
- Derivatives of Gaussian
- Can the values of a derivative filter be negative?
- What should the values sum to?
  - **Zero**: no response in constant regions
Building an edge detector

Original Image

Edge Image

original image

Grad output

norm of the gradient

\[ \| \nabla f \| = \sqrt{\left( \frac{\partial f}{\partial x} \right)^2 + \left( \frac{\partial f}{\partial y} \right)^2} \]
Building an edge detector

Thresholded norm of the gradient

How to turn these thick regions of the gradient into curves?
Non-maximum suppression

For each location \( q \) above threshold, check that the gradient magnitude is higher than at neighbors \( p \) and \( r \) along the direction of the gradient.

- May need to interpolate to get the magnitudes at \( p \) and \( r \).
Non-maximum suppression

Another problem: pixels along this edge didn’t survive thresholding
Hysteresis thresholding

Use a high threshold to start edge curves, and a low threshold to continue them.

Source: Steve Seitz
Hysteresis thresholding

original image

high threshold (strong edges)

low threshold (weak edges)

hysteresis threshold

Source: L. Fei-Fei
Recap: Canny edge detector

1. Compute x and y gradient images
2. Find magnitude and orientation of gradient
3. **Non-maximum suppression:**
   - Thin wide “ridges” down to single pixel width
4. **Linking and thresholding (hysteresis):**
   - Define two thresholds: low and high
   - Use the high threshold to start edge curves and the low threshold to continue them

opencv: `canny(image, th1, th2)`

Summary

- Convolution as translation invariant linear operations on signals and images
- Definition of convolution and its properties (associativity, commutativity, etc.)
- Artifacts of hard-edge kernels
- Gaussian kernel, its definition and properties (separability)
- Median filter, sharpening
- Derivatives as convolution (Sobel, etc.)
Sharpening

What does blurring take away?

Let's add it back:
Unsharp mask filter

\[ f + \alpha(f - f * g) = (1 + \alpha)f - \alpha f * g = f * ((1 + \alpha)e - g) \]