
University of Idaho
Department of Computer Science
Coeur d'Alene
Lecture #12
Edge Detection
Garrett Wells
revised September 15, 2024


Lecture #12
Edge Detection
Garrett Wells
revised September 15, 2024

Table of Contents I

Table of Contents I

Table of Contents

Review

Introduction to Edge Detection
Sobel Edge Detection
Canny Edge Detection
Types of Edges
Canny In Detail
Contours

Review

1. Grayscale/Color Conversions/Color Space Mapping

Review
1. Grayscale/Color Conversions/Color Space Mapping

1. Building a “stack” of techniques that build on each other to support more complicated algorithms.
2. Limiting/filtering image information to present a simplified set of features.
3. The simplified set of features hopefully allow us to pull information from complicated environments.
4. Simpler → faster, more memory efficient, cheaper **BUT** room for error

Review

1. Grayscale/Color Conversions/Color Space Mapping
2. Blurring/Filters

Review
1. Grayscale/Color Conversions/Color Space Mapping
2. Blurring/Filters

1. Building a “stack” of techniques that build on each other to support more complicated algorithms.
2. Limiting/filtering image information to present a simplified set of features.
3. The simplified set of features hopefully allow us to pull information from complicated environments.
4. Simpler → faster, more memory efficient, cheaper **BUT** room for error

Review

1. Grayscale/Color Conversions/Color Space Mapping
2. Blurring/Filters
   - Gradient calculation

Review
1. Grayscale/Color Conversions/Color Space Mapping
2. Blurring/Filters
   - Gradient calculation

1. Building a “stack” of techniques that build on each other to support more complicated algorithms.
2. Limiting/filtering image information to present a simplified set of features.
3. The simplified set of features hopefully allow us to pull information from complicated environments.
4. Simpler → faster, more memory efficient, cheaper **BUT** room for error

Review

1. Grayscale/Color Conversions/Color Space Mapping
2. Blurring/Filters
   - Gradient calculation
   - X & Y components

1. Building a “stack” of techniques that build on each other to support more complicated algorithms.
2. Limiting/filtering image information to present a simplified set of features.
3. The simplified set of features hopefully allow us to pull information from complicated environments.
4. Simpler → faster, more memory efficient, cheaper **BUT** room for error

Review

1. Grayscale/Color Conversions/Color Space Mapping
2. Blurring/Filters
   - Gradient calculation
   - X & Y components

Review

1. Grayscale/Color Conversions/Color Space Mapping
2. Blurring/Filters
   - Gradient calculation
   - X & Y components
   - Some line/edge detection

Review
1. Grayscale/Color Conversions/Color Space Mapping
2. Blurring/Filters
   - Gradient calculation
   - X & Y components
   - Some line/edge detection

1. Building a “stack” of techniques that build on each other to support more complicated algorithms.
2. Limiting/filtering image information to present a simplified set of features.
3. The simplified set of features hopefully allow us to pull information from complicated environments.
4. Simpler → faster, more memory efficient, cheaper **BUT** room for error

Review

1. Grayscale/Color Conversions/Color Space Mapping
2. Blurring/Filters

3. Histograms

1. Building a “stack” of techniques that build on each other to support more complicated algorithms.
2. Limiting/filtering image information to present a simplified set of features.
3. The simplified set of features hopefully allow us to pull information from complicated environments.
4. Simpler → faster, more memory efficient, cheaper **BUT** room for error

Review

1. Grayscale/Color Conversions/Color Space Mapping
2. Blurring/Filters

3. Histograms

Review

1. Grayscale/Color Conversions/Color Space Mapping
2. Blurring/Filters

3. Histograms
4. Thresholding

1. Building a “stack” of techniques that build on each other to support more complicated algorithms.
2. Limiting/filtering image information to present a simplified set of features.
3. The simplified set of features hopefully allow us to pull information from complicated environments.
4. Simpler → faster, more memory efficient, cheaper **BUT** room for error

Review

1. Grayscale/Color Conversions/Color Space Mapping
2. Blurring/Filters

3. Histograms
4. Thresholding

Review

1. Grayscale/Color Conversions/Color Space Mapping
2. Blurring/Filters

3. Histograms
4. Thresholding
5. Morphological Operations

Review

1. Grayscale/Color Conversions/Color Space Mapping
2. Blurring/Filters

3. Histograms
4. Thresholding
5. Morphological Operations

1. Building a “stack” of techniques that build on each other to support more complicated algorithms.
2. Limiting/filtering image information to present a simplified set of features.
3. The simplified set of features hopefully allow us to pull information from complicated environments.
4. Simpler → faster, more memory efficient, cheaper **BUT** room for error

Review

1. Grayscale/Color Conversions/Color Space Mapping
2. Blurring/Filters

3. Histograms
4. Thresholding
5. Morphological Operations
6. Now…Edge Detection

Review

1. Grayscale/Color Conversions/Color Space Mapping
2. Blurring/Filters

3. Histograms
4. Thresholding
5. Morphological Operations
6. Now…Edge Detection

1. Building a “stack” of techniques that build on each other to support more complicated algorithms.
2. Limiting/filtering image information to present a simplified set of features.
3. The simplified set of features hopefully allow us to pull information from complicated environments.
4. Simpler → faster, more memory efficient, cheaper **BUT** room for error

Types of Edge Detection

- Sobel
  - remember Sobel kernel
  - simple gradients
- Canny
  - algorithm
  - preprocessing step
  - post-processing step

Types of Edge Detection

- Sobel
  - remember Sobel kernel
  - simple gradients
- Canny
  - algorithm
  - preprocessing step
  - post-processing step

Sobel Edge Detection

Review...

- Sobel gradient calculation
- Using X & Y direction at the same time
- From [1]

Sobel Edge Detection
Review...
- Sobel gradient calculation
- Using X & Y direction at the same time
- From [1]

Sobel Edge Detection

Review…

Partial Gradients

$$\partial G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \partial G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \tag{1}$$

Gradient

$$G = \sqrt{G_x^2 + G_y^2} \tag{2}$$

or sometimes…

$$G = |G_x| + |G_y| \tag{3}$$

Sobel Edge Detection
Review…
Partial Gradients
$\partial G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \partial G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$ (1)
Gradient
$G = \sqrt{G_x^2 + G_y^2}$ (2)
or sometimes…
$G = |G_x| + |G_y|$ (3)

Sobel Edge Detection

Useful Preprocessing…

1. Grayscale
   - Limited to one channel
   - Gradients at pixel from two channels…not consistently meaningful
2. Gaussian Blur
   - Reduce noise induced gradients
   - Focus on shapes

Sobel Edge Detection
Useful Preprocessing…
1. Grayscale
   - Limited to one channel
   - Gradients at pixel from two channels…not consistently meaningful
2. Gaussian Blur
   - Reduce noise induced gradients
   - Focus on shapes

Sobel Edge Detection: Code Example

```python
img = cv.imread('sample_images/coins2.jpg')
# convert to grayscale
img = cv.cvtColor(img, cv.COLOR_BGR2GRAY)
# Gaussian Blur
img = cv.GaussianBlur(img, (5,5), 0)
# Sobel derivative
img = cv.Sobel(img, -1, 1, 1, ksize=5)
```


Sobel Edge Detection: Code Example

```python
img = cv.imread('sample_images/coins2.jpg')
# convert to grayscale
img = cv.cvtColor(img, cv.COLOR_BGR2GRAY)
# Gaussian Blur
img = cv.GaussianBlur(img, (5,5), 0)
# Sobel derivative
img = cv.Sobel(img, -1, 1, 1, ksize=5)
```

Canny Edge Detection

Algorithm Introduction...

- Sobel edge detection → noisy!
  - inner edges
  - outer edges
- How about one edge between objects?

Canny Edge Detection
Algorithm Introduction...
- Sobel edge detection → noisy!
  - inner edges
  - outer edges
- How about one edge between objects?

Canny Edge Detection

Algorithm Outline…

Canny Edge Detection [2]

1. Noise Reduction
2. Finding Intensity Gradient of Image
3. Non-Maximum Suppression
4. Hysteresis Thresholding

Canny Edge Detection
Algorithm Outline…
Canny Edge Detection [2]
1. Noise Reduction
2. Finding Intensity Gradient of Image
3. Non-Maximum Suppression
4. Hysteresis Thresholding

1. Edge detection susceptible to noise → filter noise out
2. Pixel intensity gradient(magnitude + direction) is calculated with Sobel kernel
3. Check if pixels are maximums in their neighborhood, if yes, keep. If no, suppress(set to 0).
4. Hysteresis is used to decide which pixels are edges and which are not. Set minimum acceptable gradient and maximum acceptable gradient. Edges with gradient above max are assumed to be edges. Edges below minimum are discarded. Anything between min and max are labeled as edges based on connectivity.

Types of Edges

Edge [3]

Discontinuities in pixel intensity, a sharp difference and change in pixel values.

1. Step
2. Ramp
3. Ridge
4. Roof

Types of Edges
Edge [3]
Discontinuities in pixel intensity, a sharp difference and change in pixel values.
1. Step
2. Ramp
3. Ridge
4. Roof

1. Edges are characterized by graphing pixel intensity

Types of Edges

Step…

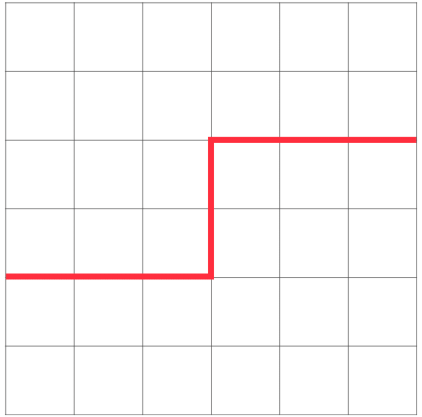


Figure: Step Edge

1. Step edges represent an instantaneous change in intensity.
2. Very clear edge in image, high contrast

Types of Edges
Step…
Figure: Step Edge

Types of Edges

Ramp...

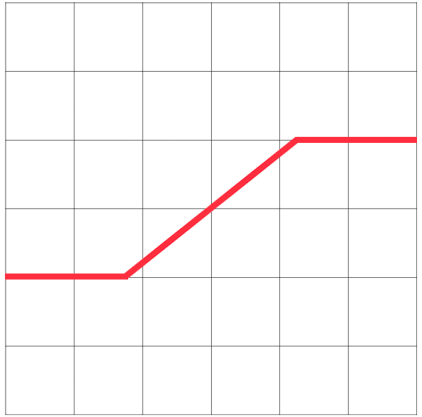


Figure: Ramp Edge

Types of Edges
Ramp...
Figure: Ramp Edge

1. Smooth, continuous change in intensity

Types of Edges

Ridge…

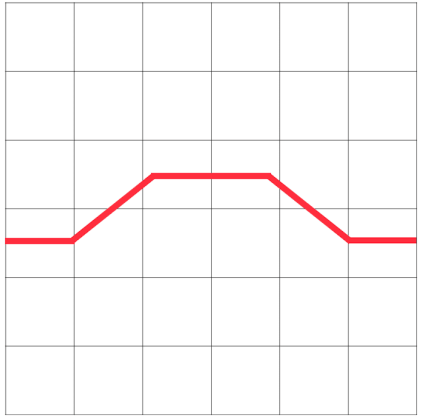


Figure: Ridge Edge

Types of Edges
Ridge…
Figure: Ridge Edge

1. Smooth continuous change on ramps, but with plateau in middle
2. Could represent as two edges, one per ramp. But may not want this behavior.
3. Gradient on plateau is nonexistent, but we may not want *two* lines in output

Types of Edges

Roof...

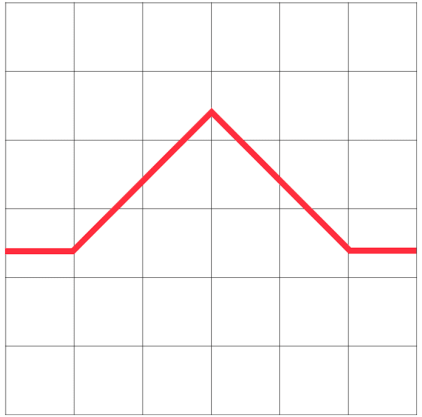


Figure: Roof Edge

Types of Edges
Roof...
Figure: Roof Edge

Canny Edge Detection

Blurring and Gradient Calculation…

1. Gaussian Blur
   - reduce detail, preserve lines
   - remove noise
2. Gradient Calculation
   2.1 convert to grayscale
   2.2 apply Sobel kernel

Canny Edge Detection
Blurring and Gradient Calculation…
1. Gaussian Blur
   - reduce detail, preserve lines
   - remove noise
2. Gradient Calculation
   2.1 convert to grayscale
   2.2 apply Sobel kernel

Canny Edge Detection

Non-Maxima Suppression…

- Sobel gradients produce a lot of potential lines
- Narrow down to best candidates
- Remove inner and outer edges
- Essentially “edge thinning” [3]

Canny Edge Detection
Non-Maxima Suppression…
- Sobel gradients produce a lot of potential lines
- Narrow down to best candidates
- Remove inner and outer edges
- Essentially “edge thinning” [3]

Canny Edge Detection

Non-Maxima Suppression...

- Look at 3x3 grid of neighboring pixels

Canny Edge Detection
Non-Maxima Suppression...
- Look at 3x3 grid of neighboring pixels

Canny Edge Detection

Non-Maxima Suppression…

- Look at 3x3 grid of neighboring pixels
- Use *direction* to compare two pixels
  1. If direction is north-south, check pixels north and south

Canny Edge Detection
Non-Maxima Suppression…
- Look at 3x3 grid of neighboring pixels
- Use *direction* to compare two pixels
  1. If direction is north-south, check pixels north and south

Canny Edge Detection

Non-Maxima Suppression…

- Look at 3x3 grid of neighboring pixels
- Use *direction* to compare two pixels
  1. If direction is north-south, check pixels north and south
  2. If direction is east-west, check pixels east and west

Canny Edge Detection
Non-Maxima Suppression…
- Look at 3x3 grid of neighboring pixels
- Use *direction* to compare two pixels
  1. If direction is north-south, check pixels north and south
  2. If direction is east-west, check pixels east and west

Canny Edge Detection

Non-Maxima Suppression…

- Look at 3x3 grid of neighboring pixels
- Use *direction* to compare two pixels
  1. If direction is north-south, check pixels north and south
  2. If direction is east-west, check pixels east and west
  3. If center pixel magnitude is largest, keep, otherwise suppress

Canny Edge Detection
Non-Maxima Suppression…
- Look at 3x3 grid of neighboring pixels
- Use *direction* to compare two pixels
  1. If direction is north-south, check pixels north and south
  2. If direction is east-west, check pixels east and west
  3. If center pixel magnitude is largest, keep, otherwise suppress

Canny Edge Detection

Non-Maxima Suppression…

- Look at 3x3 grid of neighboring pixels
- Use *direction* to compare two pixels
  1. If direction is north-south, check pixels north and south
  2. If direction is east-west, check pixels east and west
  3. If center pixel magnitude is largest, keep, otherwise suppress
- What do we do if direction is 135°?

Canny Edge Detection
Non-Maxima Suppression…
- Look at 3x3 grid of neighboring pixels
- Use *direction* to compare two pixels
  1. If direction is north-south, check pixels north and south
  2. If direction is east-west, check pixels east and west
  3. If center pixel magnitude is largest, keep, otherwise suppress
- What do we do if direction is 135°?

Canny Edge Detection

Non-Maxima Suppression…

- Look at 3x3 grid of neighboring pixels
- Use *direction* to compare two pixels
  1. If direction is north-south, check pixels north and south
  2. If direction is east-west, check pixels east and west
  3. If center pixel magnitude is largest, keep, otherwise suppress
- What do we do if direction is 135°?
  - Look at diagonal pixels

Canny Edge Detection
Non-Maxima Suppression…
- Look at 3x3 grid of neighboring pixels
- Use direction to compare two pixels
  1. If direction is north-south, check pixels north and south
  2. If direction is east-west, check pixels east and west
  3. If center pixel magnitude is largest, keep, otherwise suppress
- What do we do if direction is 135°?
  - Look at diagonal pixels

Canny Edge Detection

Hysteresis Thresholding…

1. Set hysteresis bounds

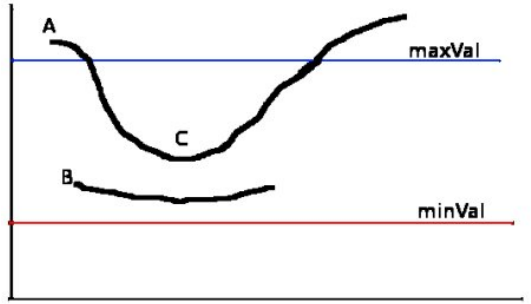


Figure: Hysteresis [2]

Canny Edge Detection
Hysteresis Thresholding…
1. Set hysteresis bounds
Figure: Hysteresis [2]

Canny Edge Detection

Hysteresis Thresholding…

1. Set hysteresis bounds
   1.1 Set minimum gradient magnitude (`minVal`)

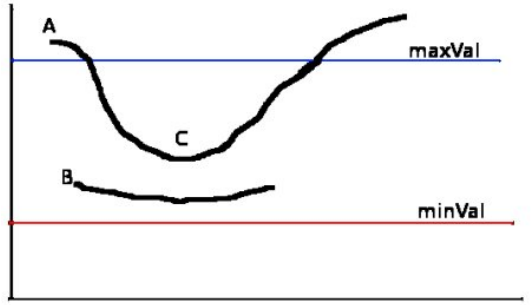
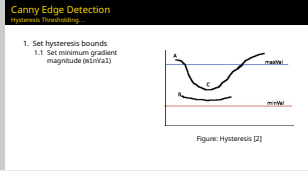


Figure: Hysteresis [2]



Canny Edge Detection

Hysteresis Thresholding…

1. Set hysteresis bounds
   - 1.1 Set minimum gradient magnitude (`minVal`)
   - 1.2 Set maximum gradient magnitude (`maxVal`)

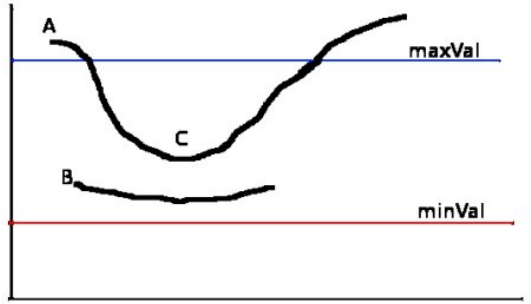


Figure: Hysteresis [2]

Canny Edge Detection
Hysteresis Thresholding…
1. Set hysteresis bounds
1.1 Set minimum gradient magnitude (minVal)
1.2 Set maximum gradient magnitude (maxVal)
Figure: Hysteresis [2]

Canny Edge Detection

Hysteresis Thresholding…

1. Set hysteresis bounds

2. Compare pixel gradient to hysteresis bounds
   - if `magnitude > maxVal` → **it's an edge**

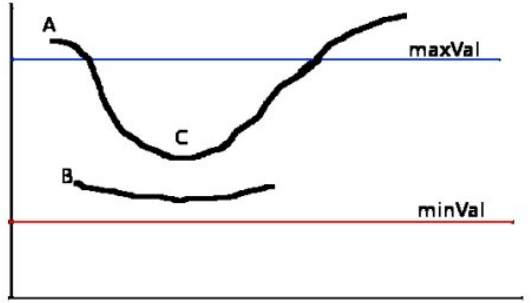
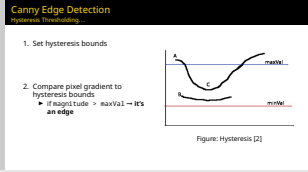


Figure: Hysteresis [2]



Canny Edge Detection

Hysteresis Thresholding…

1. Set hysteresis bounds

2. Compare pixel gradient to hysteresis bounds
   - if `magnitude > maxVal` → it's an edge
   - if `magnitude < minVal` → **discard (pixel = 0)**

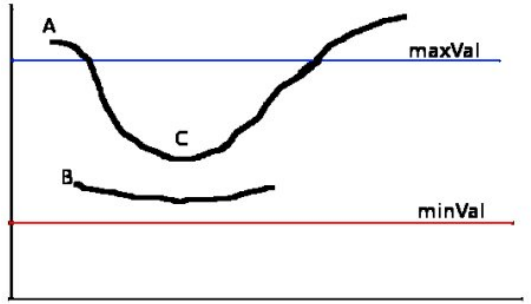
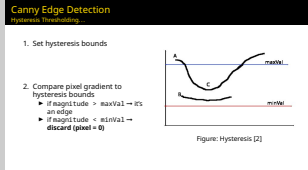


Figure: Hysteresis [2]



Canny Edge Detection

Hysteresis Thresholding…

1. Set hysteresis bounds

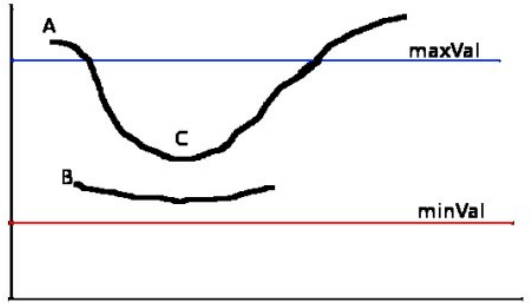
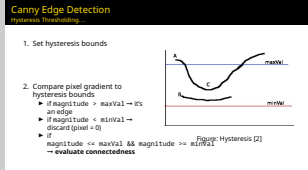


Figure: Hysteresis [2]

2. Compare pixel gradient to hysteresis bounds
   - if `magnitude > maxVal` → it's an edge
   - if `magnitude < minVal` → discard (pixel = 0)
   - if `magnitude <= maxVal && magnitude >= minVal` → **evaluate connectedness**



Canny Edge Detection

Hysteresis Thresholding…

A edge
B not an edge
C edge

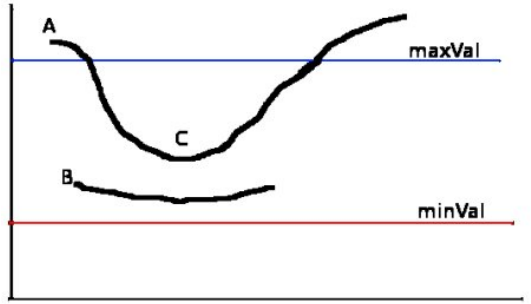


Figure: Hysteresis [2]

Canny Edge Detection
Hysteresis Thresholding…
A edge
B not an edge
C edge
Figure: Hysteresis [2]

Canny Edge Detection

Hysteresis Thresholding…

A **edge**
B not an edge
C edge

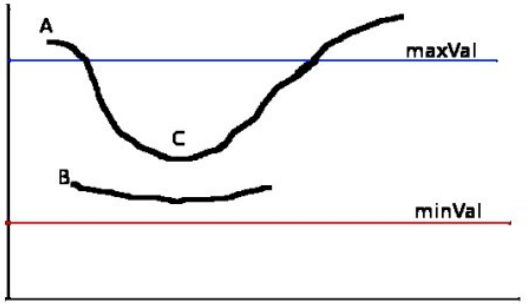


Figure: Hysteresis [2]

Canny Edge Detection
Hysteresis Thresholding…
A edge
B not an edge
C edge
Figure: Hysteresis [2]

Canny Edge Detection

Hysteresis Thresholding…

A edge
B **not an edge**
C edge

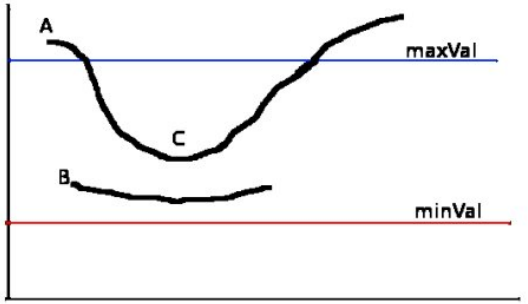


Figure: Hysteresis [2]

Canny Edge Detection
Hysteresis Thresholding…
A edge
B **not an edge**
C edge
Figure: Hysteresis [2]

Canny Edge Detection

Hysteresis Thresholding…

A edge
B not an edge
C **edge**

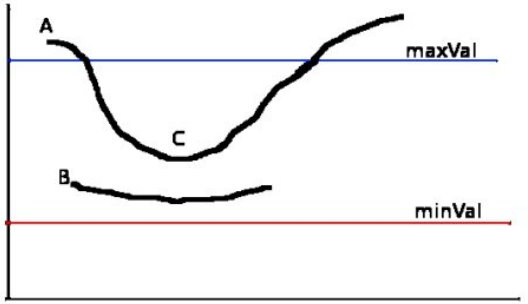


Figure: Hysteresis [2]

Canny Edge Detection
Hysteresis Thresholding…
A edge
B not an edge
C **edge**
Figure: Hysteresis [2]

Canny Edge Detection

Hysteresis Thresholding…

A edge
B not an edge
C edge → **connected to A**

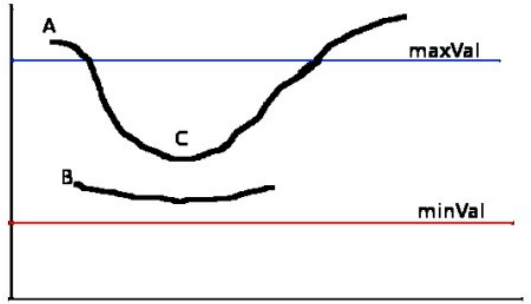


Figure: Hysteresis [2]

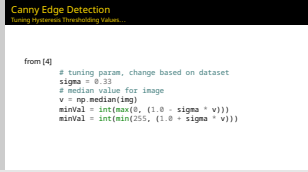
Canny Edge Detection
Hysteresis Thresholding…
A edge
B not an edge
C edge → **connected to A**
Figure: Hysteresis [2]

Canny Edge Detection

Tuning Hysteresis Thresholding Values...

from [4]

```
        # tuning param, change based on dataset
        sigma = 0.33
        # median value for image
        v = np.median(img)
        minVal = int(max(0, (1.0 - sigma * v)))
        minVal = int(min(255, (1.0 + sigma * v)))
```

Canny Edge Detection: Code Example

```python
# tuning param, change based on dataset
sigma = 0.33
# median value for image
v = np.median(img)
minVal = int(max(0, (1.0 - sigma * v)))
minVal = int(min(255, (1.0 + sigma * v)))
img = cv.Canny(img, minVal, maxVal)
```


Canny Edge Detection: Code Example

# tuning param, change based on dataset
sigma = 0.33
# median value for image
v = np.median(img)
minVal = int(max(0, (1.0 - sigma * v)))
minVal = int(min(255, (1.0 + sigma * v)))
img = cv.Canny(img, minVal, maxVal)

Canny Edge Detection: Conclusions

- Works for finding edges
- Can be noisy
- Output contains disconnected lines

Canny Edge Detection: Conclusions
- Works for finding edges
- Can be noisy
- Output contains disconnected lines

Contours

Contours

Connect objects with same intensity into one curve, similar to isophotes.

- Doesn’t require binary images…
- but transforms any input to binary

Contours
Contours
Connect objects with same intensity into one curve, similar to isophotes.
- Doesn't require binary images…
- but transforms any input to binary

Contours: Code Example

```
contours, hierarchy = cv.findContours(
                      # source image, [1, 256] -> 1
                      image,
                      # contour retrieval mode, cv.RetrievalModes
                      mode,
                      # contour approx. method
                      #   cv.ContourApproximationModes
                      method,
                      # shift contour points if desired
                      offset=Point()
                      )
# 2D list of detected contours as point vectors
contours
# hierarchy or contours where
#   hierarchy[i][0] gives parent contour
hierarchy
```


Contours: Code Example
contours, hierarchy = cv.findContours(
source image, [1, 256] -> 1
image,
contour retrieval mode, cv.RetrievalModes
mode,
contour approx. method
cv.ContourApproximationModes
method,
# shift contour points if desired
offset=Point()
)
2D list of detected contours as point vectors
contours
hierarchy or contours where
hierarchy[i][0] gives parent contour
hierarchy

Contour Retrieval Modes

```
# get only outer contours
cv.RETR_EXTERNAL
# get all contours, no hierarchy
cv.RETR_LIST
# all contours in 2 level hierarchy
cv.RETR_CCOMP
# full nested hierarchy
cv.RETR_TREE
# potentially based on pixel connectedness
cv.RETR_FLOODFILL
```


Contour Retrieval Modes
get only outer contours
cv.RETR_EXTERNAL
get all contours, no hierarchy
cv.RETR_LIST
all contours in 2 level hierarchy
cv.RETR_CCOMP
full nested hierarchy
cv.RETR_TREE
potentially based on pixel connectedness
cv.RETR_FLOODFILL

Contour Approximation Methods

```
# get all contour points
cv.CHAIN_APPROX_NONE
# only endpoints
cv.CHAIN_APPROX_SIMPLE
# flavors of Teh-Chin chain approx algorithm
cv.CHAIN_APPROX_TC89_L1
cv.CHAIN_APPROX_TC89_KCOS
```


Contour Approximation Methods

get all contour points
cv.CHAIN_APPROX_NONE
only endpoints
cv.CHAIN_APPROX_SIMPLE
flavors of Teh-Chin chain approx algorithm
cv.CHAIN_APPROX_TC89_L1
cv.CHAIN_APPROX_TC89_KCOS

Drawing Contours

```
cv.drawContours(
        image,      # destination image
        contours,   # input contours
        contourIdx, # indicates which contour to draw, -1 is all
        color,      # color to draw in
        thickness)  # how thick to make lines
```


Drawing Contours

cv.drawContours(
image, # destination image
contours, # input contours
contourIdx, # indicates which contour to draw, -1 is all
color, # color to draw in
thickness) # how thick to make lines

Bibliography I

[1] "OpenCV: Sobel Derivatives," (), [Online]. Available: https://docs.opencv.org/4.x/d2/d2c/tutorial_sobel_derivatives.html (visited on 09/15/2024).

[2] "OpenCV: Canny Edge Detection," (), [Online]. Available: https://docs.opencv.org/4.x/da/d22/tutorial_py_canny.html (visited on 09/15/2024).

[3] A. Rosebrock, "OpenCV Edge Detection (cv2.Canny)," PyImageSearch. (May 12, 2021), [Online]. Available: https://pyimagesearch.com/2021/05/12/opencv-edge-detection-cv2-canny/ (visited on 09/15/2024).

Bibliography I

[1] "OpenCV: Sobel Derivatives," (), [Online]. Available: https://docs.opencv.org/4.x/d2/d2c/tutorial_sobel_derivatives.html (visited on 09/15/2024).

[2] "OpenCV: Canny Edge Detection," (), [Online]. Available: https://docs.opencv.org/4.x/da/d22/tutorial_py_canny.html (visited on 09/15/2024).

[3] A. Rosebrock, "OpenCV Edge Detection (cv2.Canny)," PyImageSearch. (May 12, 2021), [Online]. Available: https://pyimagesearch.com/2021/05/12/opencv-edge-detection-cv2-canny/ (visited on 09/15/2024).

Bibliography II

[4] A. Rosebrock, “Zero-parameter, automatic Canny edge detection with Python and OpenCV,” PyImageSearch. (Apr. 6, 2015), [Online]. Available: https://pyimagesearch.com/2015/04/06/zero-parameter-automatic-canny-edge-detection-with-python-and-opencv/ (visited on 09/15/2024).

Bibliography II

[4] A. Rosebrock, “Zero-parameter, automatic Canny edge detection with Python and OpenCV,” PyImageSearch. (Apr. 6, 2015), [Online]. Available: https://pyimagesearch.com/2015/04/06/zero-parameter-automatic-canny-edge-detection-with-python-and-opencv/ (visited on 09/15/2024).