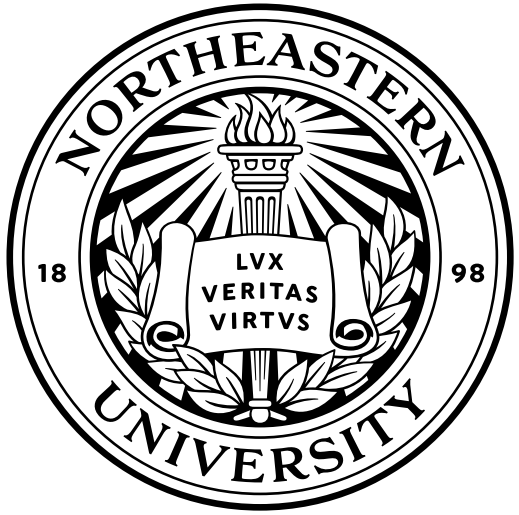


CS 5330: Pattern Recognition and Computer Vision

Northeastern University

Lab 9: Contours

** Contributed by Fall 2024 TAs: Byunghyun Ko, Yihan Wang and Taiwei Cui*

Contours

1. Introduction to Contours
2. Contour Retrieval & Approximation
3. Drawing Contours
4. Code Example
5. Summary

Introduction to Contours

- Contours: closed curves that follow the continuous boundary points of an object within an image. Captures the shape and outline of objects, making them useful for object segmentation and shape analysis.
- Application of edge detection
  - Object detection
  - Object segmentation
  - Object recognition
  - Shape analysis

Contour Detection on single blue channel image | Contour Detection on single green channel image | Contour Detection on single red channel image

https://learnopencv.com/contour-detection-using-opencv-python-c/

Introduction to Contours

- Usual steps in detecting contours
  - Convert image to a binary image by converting it to grayscale and applying thresholding
    - Why? Because this will simplify the distinction between the object and the background, making it easier to detect boundaries
  - Retrieve & approximate contours
    - There are various ”modes” when it comes to retrieving or approximating contours, which we will delve more in the following slides
  - Draw the approximated contours

Introduction to Contours

- In some cases, you may be able to skip binary conversion
  - If your grayscale or color image has strong, well-defined edges and minimal noise, contour detection may still work effectively without binary conversion.
  - If you are detecting contours based on a specialized criteria (e.g., detecting contours based on color boundaries), you will likely not need to implement binary conversion on your image

Contour Retrieval & Approximation

- Different modes for contour retrieval include:
  - **RETR_EXTERNAL**: Retrieves only the outermost contours, ignoring any nested contours. Useful for detecting the overall shape of objects.
  - **RETR_LIST**: Retrieves all contours without establishing a hierarchical relationship. This mode returns all contours as a flat list.
  - **RETR_TREE**: Retrieves all contours and organizes them in a hierarchical structure, showing the relationships between nested contours (e.g., an object within another object)

Contour Retrieval & Approximation

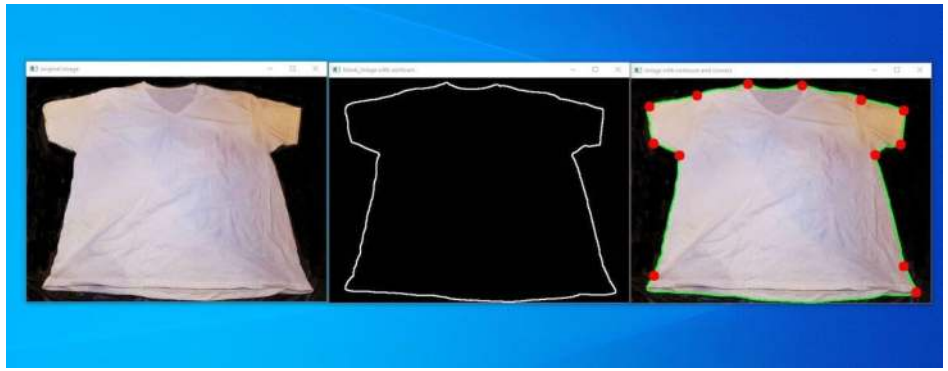
- Different modes for contour approximation include:
  - **CHAIN_APPROX_NONE**: Stores all contour points. This mode provides precise contour information but may consume more memory.
  - **CHAIN_APPROX_SIMPLE**: Reduces the number of points by keeping only the endpoints of lines. This is faster and uses less memory while still capturing the shape

Contour Retrieval & Approximation

- contours, hierarchy = cv2.findContours(image, retrieval mode, approximation mode)
  - **image**: Binary source image (created from thresholding or edge detection).
  - **Retrieval mode**: Contour retrieval mode (e.g., RETR_EXTERNAL or RETR_TREE), which determines which contours are extracted.
  - **Approximation mode**: Contour approximation method (e.g., CHAIN_APPROX_SIMPLE), which controls how much contour detail is saved.
- Returns:
  - Contours: A list of all detected contours. Each contour is represented as an array of points.
  - Hierarchy: An array describing the hierarchy of the contours

Drawing Contours

- cv2.drawContours(image, contours, contourIdx, color, thickness)
  - **image**: The image where contours will be drawn.
  - **contours**: List of contours obtained from findContours.
  - **contourIdx**: Index of the contour to draw. Use -1 to draw all contours.
  - **color**: Color of the contour lines (e.g., (0, 255, 0) for green).
  - **thickness**: Thickness of the contour lines.
- This function will draw contours on an image



https://automaticaddison.com/how-to-draw-contours-around-objects-using-opencv/

Code Example

Code

```
# Load and preprocess the image
image = cv2.imread('input_image.jpg')
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
_, binary = cv2.threshold(gray, 127, 255, cv2.THRESH_BINARY)

# Find contours
contours, hierarchy = cv2.findContours(binary, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

# Draw contours on the original image
cv2.drawContours(image, contours, -1, (0, 255, 0), 2)

# Display result
cv2.imshow("Contours", image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

contours, hierarchy = cv2.findContours(image, retrieval mode, approximation mode)

- **image**: Binary source image (created from thresholding or edge detection).
- **Retrieval mode**: Contour retrieval mode (e.g., RETR_EXTERNAL or RETR_TREE), which determines which contours are extracted.
- **Approximation mode**: Contour approximation method (e.g., CHAIN_APPROX_SIMPLE), which controls how much contour detail is saved.

cv2.drawContours(image, contours, contourIdx, color, thickness)

- **image**: The image where contours will be drawn.
- **contours**: List of contours obtained from findContours.
- **contourIdx**: Index of the contour to draw. Use -1 to draw all contours.
- **color**: Color of the contour lines (e.g., (0, 255, 0) for green).
- **thickness**: Thickness of the contour lines.

Summary

- **Contours**: Useful for isolating and analyzing object shapes in an image.
- **Retrieval Modes**: Different retrieval modes (RETR_EXTERNAL, RETR_TREE, etc.) provide options for extracting only the outer contours or building a hierarchical structure.
- **Approximation Modes**: CHAIN_APPROX_SIMPLE helps save memory by reducing contour points, while CHAIN_APPROX_NONE retains all points for more precise contours.
- **Practical Usage**: Contours are widely used in object detection, shape analysis, and segmentation tasks in computer vision.