University of Idaho
Department of Computer Science
Coeur d'Alene

# Lecture #14

*Foreground Detection and Segmentation*

Garrett Wells

*revised September 19, 2024*

# Table of Contents I

# Review

- Image Moments

# Review

- Image Moments
  - count pixels...

# Review

- Image Moments
  - count pixels...
  - weight by location...

# Review

- Image Moments
  - count pixels…
  - weight by location…
  - gives shape information

# Review

- Image Moments
  - count pixels...
  - weight by location...
  - gives shape information
- Hough Transforms

# Review

- Image Moments
  - count pixels...
  - weight by location...
  - gives shape information
- Hough Transforms
  - detect lines by checking all possible lines

# Review

- Image Moments
  - count pixels...
  - weight by location...
  - gives shape information
- Hough Transforms
  - detect lines by checking all possible lines
  - computationally expensive...

# Review

- Image Moments
  - count pixels...
  - weight by location...
  - gives shape information
- Hough Transforms
  - detect lines by checking all possible lines
  - computationally expensive...
  - but also not gradient based

# Image Classification, Object Detection, and Image Segmentation

| Image Classification |
| --- |
| Applies a class label to an entire image. |

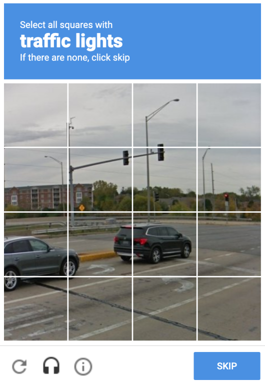
# Image Classification, Object Detection, and Image Segmentation

## Image Classification

Applies a class label to an entire image.

## Example

Select all images that contain traffic lights.



# Image Classification, Object Detection, and Image Segmentation

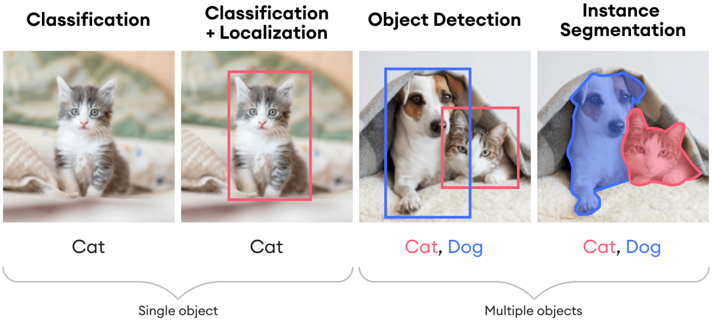
## Object Detection

Generates bounding boxes, showing where objects are located in an image. Can detect multiple objects in an image. Struggles to show precise boundaries between objects.

# Image Classification, Object Detection, and Image Segmentation

## Object Detection

Generates bounding boxes, showing where objects are located in an image. Can detect multiple objects in an image. Struggles to show precise boundaries between objects.



# Image Classification, Object Detection, and Image Segmentation

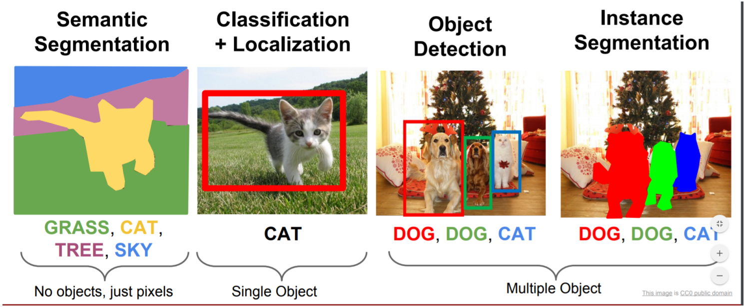
## Image Segmentation

Assign individual pixels classes. Individual pixels may belong to a class, or have class *and* instance.

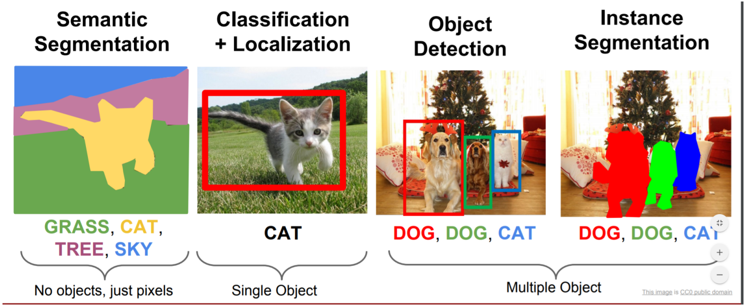
# Image Classification, Object Detection, and Image Segmentation

## Image Segmentation

Assign individual pixels classes. Individual pixels may belong to a class, or have class *and* instance.

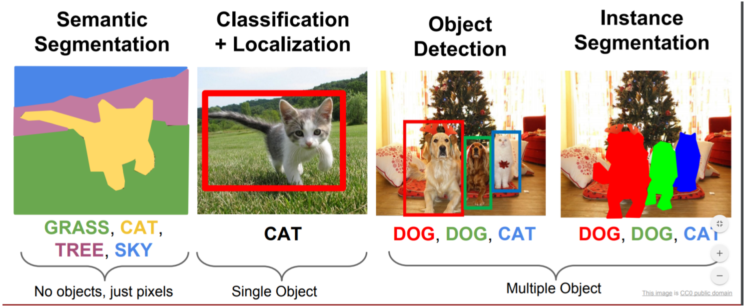


# Image Classification, Object Detection, and Image Segmentation

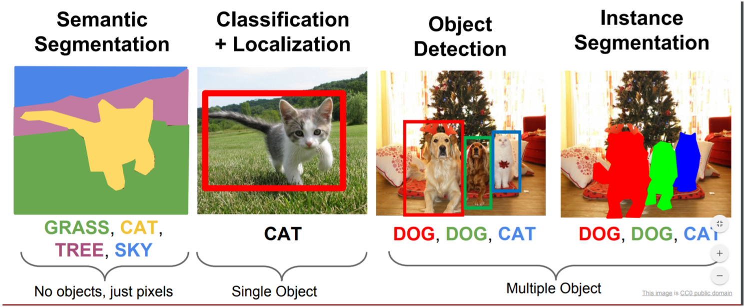



- ► ***Semantic Segmentation*** **–** class for each pixel, pixels from two instances cannot be differentiated
- ► ***Instance Segmentation*** **–** class *and* instance, can now differentiate

# Image Segmentation



# Image Segmentation




- Labeling pixels with classes
- Used for:
  - motion detection
  - object detection/identification

# Foreground Detection

- "Foreground" contains objects of interest
- Many algorithms
  - Connected Components
  - Grab Cut
  - Background Subtractors
  - Contours and Masking
  - Watershed
  - *etc*

# Connected Components

- Uses graph theory [1]
- Constructs graph from input data
- Algorithm traverses the graph labeling vertices based on connectivity and relative values

# Connected Components in OpenCV

```
# outputs labels mask, stats for labels, centroids
labels, stats, centroids = cv.connectedComponentsWithStats(
    image,          # input image, 8bit, single channel
    connectivity,   # 8 or 4 way connectivity
    Itype,          # output image label type
    ccItype,        # connected components algorithm type
)
```

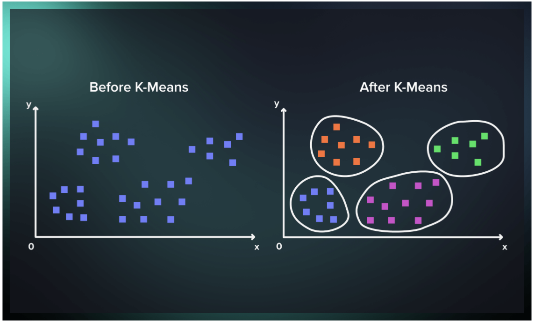
# Clustering Algorithms

- Useful for grouping/choosing foreground and background groups
- K means clustering
- Gaussian Mixture Method/Mixture of Gaussians Method

# Clustering Algorithms

## K-Means Clustering

- Calculates “k” center points from the data
- Attempts to assign each data point to one of the k center cluster’s based on distance
- Assumes distributions are circle shaped
- Sources [2], [3], [4]



# Clustering Algorithms

## Gaussian Mixture Method/Mixture of Gaussians

- Attempts to assign each point in the data to a Gaussian Distribution [5]
- Probabilistic Model
- Expectation-Maximization for fitting
- Can perform better than K-Means, especially over unusually clustered data
- Both are used for unsupervised learning

# GrabCut

- Popular algorithm, particularly before more common CNN methods [6]
- Several Steps [7]
  1. Specify bounding box or image mask over “foreground”
  2. Gaussian Mixture Method used to estimate foreground/background
  3. Markov random field over labels
  4. Optimization

# GrabCut in OpenCV

```
cv.grabCut(
    img,      # input image, 3 channel
    mask,     # input/output single channel mask
    rect,     # estimated bounding box of foreground object
    bdgModel, # temp array for background model
    fgdModel, # temp array for foreground model
    iterCount, # number of iterations for algorithm
    mode,     # one of the cv.GrabCutModes
    )
```

# GrabCut Code Example

```
# basic mask, same shape as image
mask = np.zeros(img.shape[:2], np.uint8)
# two empty arrays for the GrabCut to use
backgroundModel = np.zeros((1,65),np.float64)
foregroundModel = np.zeros((1,65),np.float64)
# estimate a bounding rectangle for the foreground object
rect = (133, 185, 394, 489)
cv.grabCut(
    img,
    mask,
    rect,
    backgroundModel,
    foregroundModel,
    3,  # iterations
    cv.GC_INIT_WITH_RECT) # grabcut method
```

# GrabCut in OpenCV

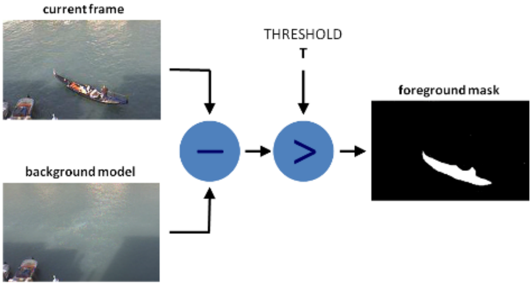
```
# To start, make the mask the same shape as the image
mask = np.zeros(img.shape[:2], np.uint8)
#Common values for foreground and background temp models
backgroundModel = np.zeros((1,65),np.float64)
foregroundModel = np.zeros((1,65),np.float64)
#Rectangle
#rect = (0, 0, img.shape[0], img.shape[1])
#Better rectangle for segmenting
rect = (133, 185, 394, 489)
cv.grabCut(img, mask, rect, backgroundModel, foregroundModel, 3,
```

# Background Subtractors

- Useful “out of the box”
- OpenCV example uses:
  - counting people
  - tracking vehicles
  - moving objects, *etc.*

# Background Subtractors

- Background Subtractors:
  1. only work on video streams
  2. initializes a model of the background
  3. compares to the current frame (subtracts current from model)
  4. applies thresholding to create foreground mask
  5. static pixels are background (0)
- **Assumption:** An image with **no movement** has **no foreground subject**



# Background Subtractors in OpenCV

```python
# don't take arguments
cv.createBackgroundSubtractorMOG2()
cv.createBackgroundSubtractorKNN()

# instead, use with:
# from cv tutorial
backsub = cv.createBackgroundSubtractorMOG2()
while True:
  still = webcam.read()
  fgMask = backsub.apply(still[1])
```

# Contours and Masking

- Calculate image contours
- Area of contours → generate image mask

# Contours and Masking in OpenCV

```
# calculate contours
contours, hierarchy = cv.findContours(img, cv.RETR_EXTERNAL, cv.C
# sort
contours = sorted(contours, key=cv.contourArea)
# generate empty mask
mask = np.zeros_like(img)
cv.drawContours(
    mask, # pass in mask
    [contours[-1]],  # last contour in list
    -1, # draw all
    255, # color
    cv.FILLED, # fill in contour
    1)
```

# Watershed

- Use image topology

# Watershed

- Use image topology
  1. imagine image with pixel values forming slopes and ridges

# Watershed

- Use image topology
  1. imagine image with pixel values forming slopes and ridges
  2. pour water over image

# Watershed

- Use image topology
  1. imagine image with pixel values forming slopes and ridges
  2. pour water over image
  3. water settles in lowest points first, forming small puddles which we label

# Watershed

- Use image topology
    1. imagine image with pixel values forming slopes and ridges
    2. pour water over image
    3. water settles in lowest points first, forming small puddles which we label
    4. continue to pour water

# Watershed

- Use image topology
    1. imagine image with pixel values forming slopes and ridges
    2. pour water over image
    3. water settles in lowest points first, forming small puddles which we label
    4. continue to pour water
    5. whenever water level rises enough that two puddles merge, draw a “barrier”

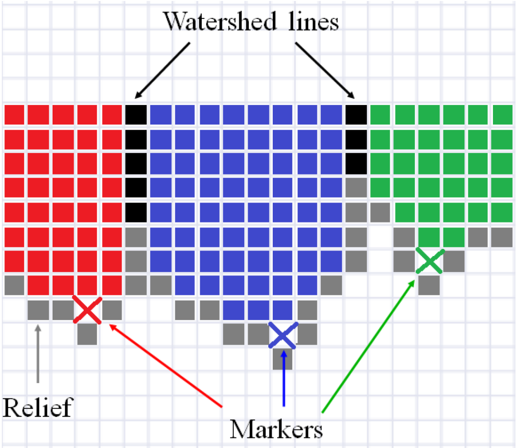
# Watershed

- Use image topology
    1. imagine image with pixel values forming slopes and ridges
    2. pour water over image
    3. water settles in lowest points first, forming small puddles which we label
    4. continue to pour water
    5. whenever water level rises enough that two puddles merge, draw a “barrier”
    6. these barriers reveal regions of the image containing features

# Watershed

- Use image topology
  1. imagine image with pixel values forming slopes and ridges
  2. pour water over image
  3. water settles in lowest points first, forming small puddles which we label
  4. continue to pour water
  5. whenever water level rises enough that two puddles merge, draw a “barrier”
  6. these barriers reveal regions of the image containing features
- *Label minima of image, then draw barriers along boundaries/ridges between minima to segment image into feature areas.*

# Watershed



# Practical Watershed Steps

A workflow from Doug Park:

1. remove noise
2. find background (threshold perhaps)
3. find foreground (distance transform perhaps)
4. find unkown region (subtract background from foreground)
5. use connected components to label foreground
6. outline unknown areas
7. input outline into watershed with original image
8. post-process as needed

# Distance Transform

- The farther a pixel is from a location, the greater the pixel intensity
- Similar to the premise of Watershed
- Provides information helpful for finding the background

# Watershed Example Code

```
# too long to include here, see code example on GitHub
```

# Flood Fill

- Method for filling in bounded area with color [8]
- Paint/bucket fill tool in paint programs
- Uses “seed points” to fill in region
- Components connected to start seed have value changed
  - 4 direction flood
  - 8 direction flood

# Bibliography I


[1] *Connected-component labeling*, in *Wikipedia*, Dec. 27, 2023. [Online]. Available: `https://en.wikipedia.org/w/index.php?title=Connected-component_labeling&oldid=1192036140` (visited on 09/19/2024).

[2] "OpenCV: K-Means Clustering in OpenCV," (), [Online]. Available: `https://docs.opencv.org/4.x/d1/d5c/tutorial_py_kmeans_opencv.html` (visited on 09/19/2024).

[3] E. Ecosystem (LEDU), "Understanding K-means Clustering in Machine Learning," Medium. (Sep. 12, 2018), [Online]. Available: `https://towardsdatascience.com/understanding-k-means-clustering-in-machine-learning-6a6e67336aa1` (visited on 09/19/2024).

# Bibliography II


[4] *K-means clustering*, in *Wikipedia*, Aug. 30, 2024. [Online]. Available: https://en.wikipedia.org/w/index.php?title=K-means_clustering&oldid=1243054475 (visited on 09/19/2024).

[5] "2.1. Gaussian mixture models," scikit-learn. (), [Online]. Available: https://scikit-learn/stable/modules/mixture.html (visited on 09/19/2024).

[6] A. Rosebrock, "OpenCV GrabCut: Foreground Segmentation and Extraction," PyImageSearch. (Jul. 27, 2020), [Online]. Available: https://pyimagesearch.com/2020/07/27/opencv-grabcut-foreground-segmentation-and-extraction/ (visited on 09/20/2024).

[7] *GrabCut*, in *Wikipedia*, Mar. 27, 2021. [Online]. Available: https://en.wikipedia.org/w/index.php?title=GrabCut&oldid=1014501536 (visited on 09/19/2024).

# Bibliography III


[8] *Flood fill*, in *Wikipedia*, Jun. 4, 2024. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Flood_fill&oldid=1227176300 (visited on 09/20/2024).

**Temporary page!**

LaTeX was unable to guess the total number of pages correctly. As there was some unprocessed data that should have been added to the final page this extra page has been added to receive it.
If you rerun the document (without altering it) this surplus page will go away, because LaTeX now knows how many pages to expect for this document.