Computer Vision using OpenCV

CMU Build18 Tutorial, 2016-17

Esha Uboweja
euboweja@[cmu.edu | gmail.com]

Who am I?

Installing OpenCV for Python

- Installing OpenCV 3.1:
  - http://www.pyimagesearch.com/2015/06/22/install-opencv-3-0-and-python-2-7-on-ubuntu/ *(There are updated instructions for v3.1)*
  - OpenCV 3.x v/s OpenCV 2.x
    - More algorithms in OpenCV 3.x
    - OpenCV 3.x compatible with Python 3
    - Choose version best fit for your project. If more resources exist for OpenCV 2.x on RaspberryPi or some other hardware, use it.
  - For different operating systems, installing can be as simples as one command line instruction v/s installing from scratch with the source code from Itseez (https://github.com/opencv/opencv)
    - For this tutorial, any version will work. I use OpenCV 3.1 with Python 2.7 on a Mac OS, and I installed it from scratch using the CMake build system (https://cmake.org). If you run into issues while executing sample code, look for alternative function signatures in the OpenCV version you installed.
    - On MacOS, a command line install with your package manager will be sufficient : `brew install opencv`
    - For Windows, consider installing OpenCV from scratch using CMake GUI (lots of tutorials for help are available on the web).
    - Consider installing from scratch if you are going to use C++ with OpenCV or Python

Read, Display an Image

birds.jpg

In Color

```
# Read and display an image using OpenCV
# Import Libraries
import cv2
# Read image in color format
birdsImg = cv2.imread("./images/birds.jpg")
# Display image in new window
cv2.imshow("Birds", birdsImg)
# Wait for key press on external window
cv2.waitKey(0)
# After key press, destroy all external windows
cv2.destroyAllWindows()
```

In Grayscale

```
# Read and display an image using OpenCV
# Import Libraries
import cv2
# Read image in color format
birdsImgGray = cv2.imread(“./images/birds.jpg”, 0)
# Display image in new window
cv2.imshow(“Birds Gray", birdsImgGray)
# Wait for key press on external window
cv2.waitKey(0)
# After key press, destroy all external windows
cv2.destroyAllWindows()
```

Grayscale

Color Spaces

Storing color information

- Different types of color spaces:
  - RGB : Red, Green, Blue
  - YCrCb
  - YUV
  - Lab
  - HSV - Hue, Saturation, Value
  - HSL - Hue, Saturation, Lightness
  - Many more…
- Images are usually stored as 3D matrices of shape (W * H * 3)
  W = image width, H = image height
- Warning: OpenCV uses BGR format (`matplotlib` uses RGB)

OpenCV BGR v/s RGB

```
# Import Libraries
import cv2
# Read image in color format
birdsImg = cv2.imread("./images/birds.jpg")
# Get R, G, B channels
b,g,r = cv2.split(birdsImg)
# Combine in reverse format
birdsImgSwitched = cv2.merge([r, g, b])
# Display image in new window
cv2.imshow("Birds Switched Channels", birdsImgSwitched)
# Wait for key press on external window
cv2.waitKey(0)
# After key press, destroy all external windows
cv2.destroyAllWindows()
```

OpenCV BGR v/s RGB

```
# Import Libraries
import cv2
# Read image in color format
birdsImg = cv2.imread("./images/birds.jpg")
# Get R, G, B channels
b,g,r = cv2.split(birdsImg)
# Combine in reverse format
birdsImgSwitched = cv2.merge([r, g, b])
# Display image in new window
cv2.imshow("Birds Switched Channels", birdsImgSwitched)
# Wait for key press on external window
cv2.waitKey(0)
# After key press, destroy all external windows
cv2.destroyAllWindows()
```

RGB

Why use a different color space?

- HSV - (Hue, Saturation, Value):
  - Separates intensity information (luma) from color information (chroma)
    - For example, there can be light intensity variations of an object moving in a video and using a different color space can be robust to such variations when tracking the object.
- https://en.wikipedia.org/wiki/HSL_and_HSV

Converting to HSV color space

```
# Import Libraries
import cv2
# Read image in color format
birdsImg = cv2.imread("./images/birds.jpg")
# Convert to HSV space
birdsHSV = cv2.cvtColor(birdsImg, cv2.COLOR_BGR2HSV)
# Display image in new window
# -> shows something weird
cv2.imshow("Birds", birdsHSV)
# Wait for key press on external window
cv2.waitKey(0)
# After key press, destroy all external windows
cv2.destroyAllWindows()
```

Converting to HSV color space

```
# Import Libraries
import cv2
# Read image in color format
birdsImg = cv2.imread("./images/birds.jpg")
# Convert to HSV space
birdsHSV = cv2.cvtColor(birdsImg, cv2.COLOR_BGR2HSV)
# Display image in new window
# -> shows something weird
cv2.imshow("Birds", birdsHSV)
# Wait for key press on external window
cv2.waitKey(0)
# After key press, destroy all external windows
cv2.destroyAllWindows()
```

HSV

Don’t worry! You haven’t lost any color information. The “`imshow(…)`” function expects a BGR array, but we provide it an HSV array, so the image looks different.

Image Filtering

Image Filtering basics

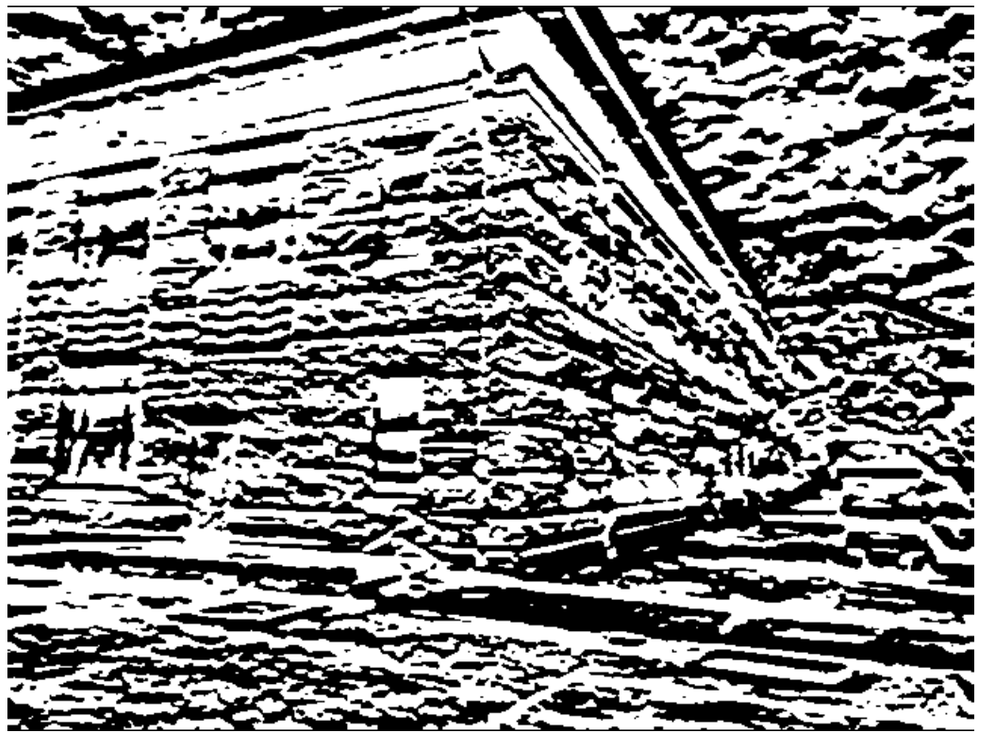
- Box Filter (averaging)
- Gaussian Filter (smoothing) - https://github.com/eknight7/cv_build18/blob/master/filtering/gaussian_filter.py
- Sobel Filter (edge detection) - https://github.com/eknight7/cv_build18/blob/master/filtering/sobel_filter.py

Sobel Filtering for edge detection

Smith Hall @ CMU

Sobel Filtering for edge detection

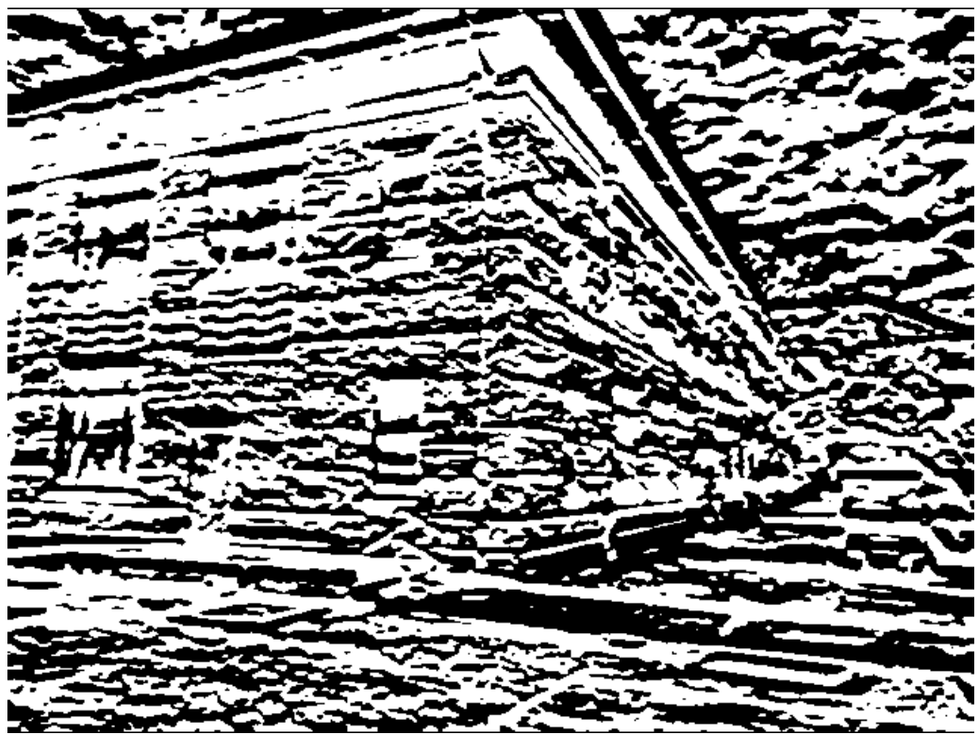
??



??

Sobel Filtering for edge detection

X



Y

Other types of filters

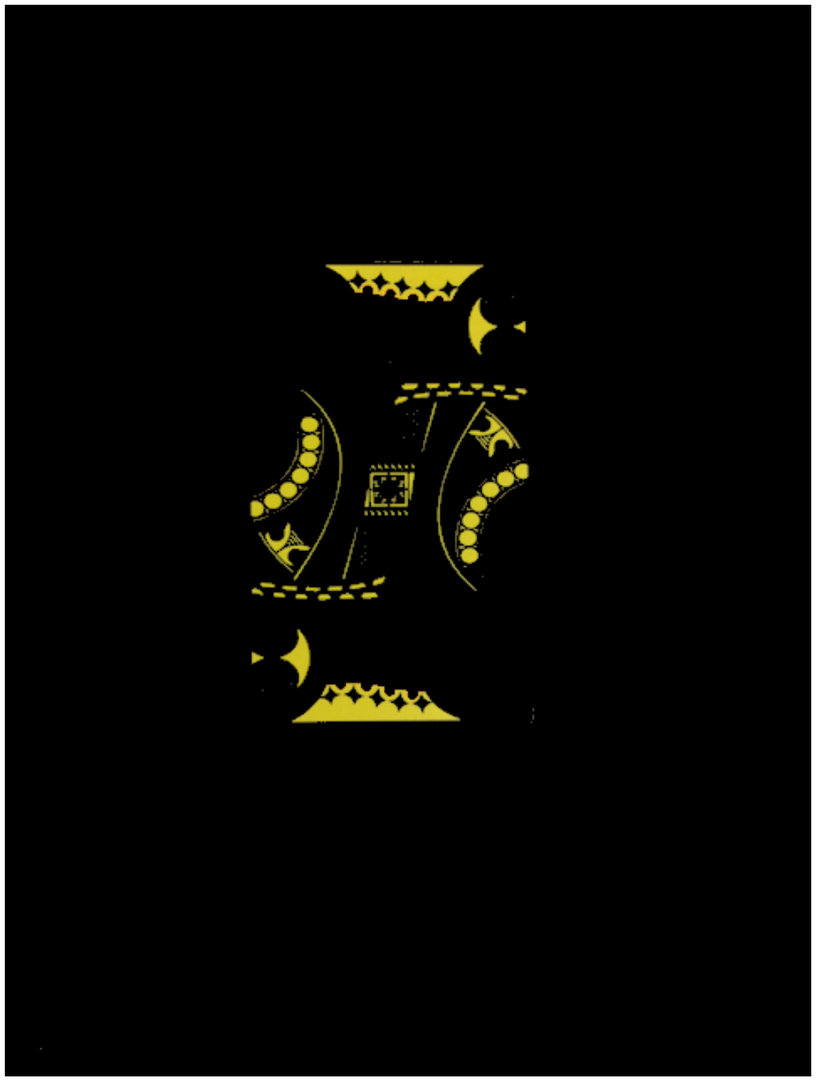
- Canny Edge Detector
- Laplacian Filtering for edge detection
- Median filter
- Bilateral filter (smoothes noise and preserves edges!)

Object Detection

Color based segmentation

- Use HSV color space to select the color range of the object you want to recognize or track
  - This returns color segmentation bounds such as `low_hsv` and `high_hsv` entsf
- Use the color bounds to make a bit mask and segment the region containing the colored object
- Play with an app to use HSV to segment regions based on color here: https://github.com/eknight7/cv_build18/blob/master/segmentation/color_segment.py

Color based segmentation example results



Object Detection

- Why not do pixel based matching?
- Feature descriptors and feature points
- Corners
- SIFT, SURF, BRIEF, HoG, etc.
- Face detection using Haar Cascades
- Machine learning for object recognition

Tracking objects

Capture video from a camera

```
# Import Libraries
import cv2
# Open the "default" camera
vc = cv2.VideoCapture(0)
# Check if we succeeded in opening camera feed
if vc.isOpened():
    rval, inputFrame = vc.read()
else:
    rval = False
# Display captured frames in a new window
cv2.namedWindow("Camera Video Feed")
while rval:
    cv2.imshow("Camera Video Feed", inputFrame)
    rval, inputFrame = vc.read()
    # Wait for ESC key press
    key = cv2.waitKey(20)
    if key == 27: # User pressed ESC key
        break
# Destroy window
cv2.destroyWindow("Camera Video Feed")
# Close VideoCapture feed
vc.release()
```

Demo: Tracking objects based on color in HSV space

(https://github.com/eknight7/cv_build18/blob/master/tracking/color_track.py)

Do’s and Don’ts for computer vision projects at hackathons

Lab Exercises (Choose only 1!)

M&M candy counter

Image Source: Wikipedia

https://upload.wikimedia.org/wikipedia/commons/b/ba/M%26M%27s_Plain.jpg

Multi-object tracker

- Extend the color based tracker to work on multiple objects of different colors or multiple objects of the same color
- If color tracking doesn't work, what extra information can help? Shape? Texture? Size?
- If tracking many objects is hard, try tracking one object, but keep track of its position at each time step. One example would be to draw the path the object takes in each frame.

Card games (A pack of 52 cards)

- Color segmentation can be a first step in many applications. How can you use this to determine what playing card is in front of the camera?
- Can you make an app to solve the game of 24? In this game, there are 4 cards and one has to come up with a set of arithmetic operations (+,-, *, /) to combine the 4 numbers and get the number 24. Can you think of any other interesting game you want to play with cards and computer vision?

Computer vision resources

- http://www.pyimagesearch.com
- http://docs.opencv.org/3.0-beta/doc/py_tutorials/py_tutorials.html
- Book http://programmingcomputervision.com (officially available free as a pdf online, try out all code samples!)
- http://stackoverflow.com/questions/tagged/opencv
- Courses:
  - 16-385 Computer Vision (undergraduate course)
  - 16-720 Computer Vision (graduate course)
  - 15-463/663 Computational Photography (both versions)