

Guided Project Report

Adaptive Thresholding for Edge Detection on Images

Name: Shruti Verma
Course: AI and ML
(Batch 4)
Duration: 10 months

Problem Statement: Use OpenCV, Implementing Edge detection on an image

Prerequisites

What things you need to install the software and how to install them:

Python 3.8 or higher versions This setup requires that your machine has latest version of python. The following url <https://www.python.org/downloads/> can be referred to download python. Once you have python downloaded and installed, you will need to setup PATH variables (if you want to run python program directly, detail instructions are below in how to run software section). To do that check this: <https://www.pythoncentral.io/add-python-to-path-python-is-not-recognized-as-an-internal-or-external-command/>. Setting up PATH variable is optional as you can also run program without it and more instruction are given below on this topic.

Second and easier option is to download anaconda and use its anaconda prompt to run the commands. To install anaconda check this url <https://www.anaconda.com/download/> You will also need to download and install below 3 packages after you install either python or anaconda from the steps above Sklearn (scikit-learn) numpy scipy if you have chosen to install python 3.8 then run below commands in command prompt/terminal to install these packages `pip install -U scikit-learn` `pip install numpy` `pip install scipy` if you have chosen to install anaconda then run below commands in anaconda prompt to install these packages `conda install -c scikit-learn` `conda install -c anaconda numpy` `conda install -c anaconda scipy` . Install opencv.

Dataset used

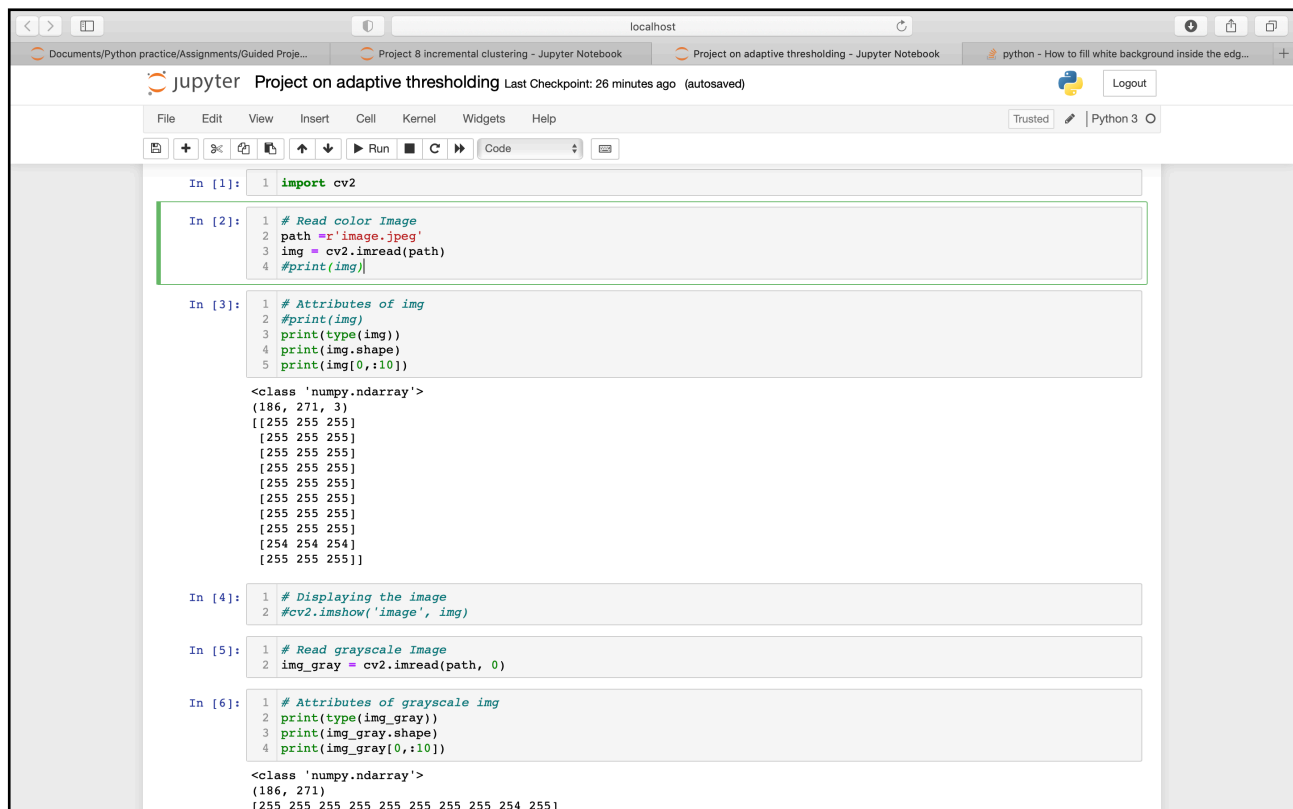
The data is a sample image taken from internet of a fish in jpeg format.

Method used for detection

Canny Edge Detection

Image reading -> Converting to Grayscale image -> applying canny edge detection > applying adaptive threshold > mask

Importing the libraries and reading the image as grayscale:



The screenshot shows a Jupyter Notebook titled "Project on adaptive thresholding" running on a localhost. The notebook contains six code cells. The first cell imports the cv2 library. The second cell reads a color image from a file named 'image.jpeg'. The third cell prints the attributes of the loaded image, showing it is a 186x271x3 numpy.ndarray. The fourth cell displays the image using cv2.imshow. The fifth cell reads the same image as a grayscale image. The sixth cell prints the attributes of the grayscale image, showing it is a 186x271x1 numpy.ndarray.

```
In [1]: 1 import cv2

In [2]: 1 # Read color Image
2 path = 'image.jpeg'
3 img = cv2.imread(path)
4 #print(img)

In [3]: 1 # Attributes of img
2 #print(img)
3 print(type(img))
4 print(img.shape)
5 print(img[0,:10])

<class 'numpy.ndarray'>
(186, 271, 3)
[[255 255 255]
 [255 255 255]
 [255 255 255]
 [255 255 255]
 [255 255 255]
 [255 255 255]
 [255 255 255]
 [255 255 255]
 [255 255 255]
 [254 254 254]
 [255 255 255]]

In [4]: 1 # Displaying the image
2 #cv2.imshow('image', img)

In [5]: 1 # Read grayscale Image
2 img_gray = cv2.imread(path, 0)

In [6]: 1 # Attributes of grayscale img
2 print(type(img_gray))
3 print(img_gray.shape)
4 print(img_gray[0,:10])

<class 'numpy.ndarray'>
(186, 271)
[255 255 255 255 255 255 255 255 254 255]
```

Canny Edge Detection

```
2 #cv2.imshow('image', img)

In [5]: 1 # Read grayscale Image
2 img_gray = cv2.imread(path, 0)

In [6]: 1 # Attributes of grayscale img
2 print(type(img_gray))
3 print(img_gray.shape)
4 print(img_gray[0,:10])

<class 'numpy.ndarray'>
(186, 271)
[255 255 255 255 255 255 255 255 254 255]

In [7]: 1 # Canny Edge Detection
2
3 #Canny Edge Detection is a popular edge detection algorithm.
4 # edge = cv2.Canny(image, minVal, maxVal)
5 resize_img = img
6 edge= cv2.Canny(resize_img,0,255)
7 def edge_change(val):
8     minv = cv2.getTrackbarPos('min:', "Edge")
9     maxv = cv2.getTrackbarPos('max:', "Edge")
10    edge= cv2.Canny(resize_img,minv,maxv)
11    cv2.imshow("Edge",edge)
12    cv2.createTrackbar('min:', "Edge", 0, 255, edge_change)
13    cv2.createTrackbar('max:', "Edge", 0, 255, edge_change)

In [8]: 1 # Applying simple thresholding
2
3 # ret, thres = cv2.threshold(source, thresholdValue, maxVal, thresholdingTechnique)
4 gray = img_gray
5 ret, thres = cv2.threshold(gray, 225, 255, cv2.THRESH_BINARY_INV)
6 def thres_own(val):
7     thres = cv2.getTrackbarPos('thres:', "Thres")
8     thr=np.where(gray>thres,0,255).astype("uint8")
9     cv2.imshow("Thres",thr)
10    cv2.createTrackbar('thres:', "Thres", 0, 255, thres_own)
11    #cv2.THRESH_BINARY: If pixel intensity is greater than the set threshold, value set to 255, else set to 0 (black).
12    #cv2.THRESH_BINARY_INV: Inverted or Opposite case of cv2.THRESH_BINARY.
```

Simple and adaptive thresholding

```
localhost
Documents/Python practice/Assignments/Guided Proje... Project 8 incremental clustering - Jupyter Notebook Project on adaptive thresholding - Jupyter Notebook python - How to fill white background inside the edg...
jupyter Project on adaptive thresholding Last Checkpoint: 27 minutes ago (autosaved)
File Edit View Insert Cell Kernel Widgets Help Trusted Python 3
In [7]: 1 # Canny Edge Detection
2
3 #Canny Edge Detection is a popular edge detection algorithm.
4 # edge = cv2.Canny(image, minVal, maxVal)
5 resize_img = img
6 edge = cv2.Canny(resize_img,0,255)
7 def edge_change(val):
8     minv = cv2.getTrackbarPos('min:', "Edge")
9     maxv = cv2.getTrackbarPos('max:', "Edge")
10    edge = cv2.Canny(resize_img,minv,maxv)
11    cv2.imshow("Edge",edge)
12    cv2.createTrackbar('min:', "Edge", 0, 255, edge_change)
13    cv2.createTrackbar('max:', "Edge", 0, 255, edge_change)
In [8]: 1 # Applying simple thresholding
2
3 # ret, thresh = cv2.threshold(source, thresholdValue, maxVal, thresholdingTechnique)
4 gray = img_gray
5 ret, thresh = cv2.threshold(gray, 225, 255, cv2.THRESH_BINARY_INV)
6 def thresh_own(val):
7     thresh = cv2.getTrackbarPos('thresh:', "Thres")
8     thr = np.where(gray>thresh,0,255).astype("uint8")
9     cv2.imshow("Thres",thr)
10    cv2.createTrackbar('thresh:', "Thres", 0, 255, thresh_own)
11    #cv2.THRESH_BINARY: If pixel intensity is greater than the set threshold, value set to 255, else set to 0 (black).
12    #cv2.THRESH_BINARY_INV: Inverted or Opposite case of cv2.THRESH_BINARY.
In [9]: 1 # Applying adaptive thresholding
2
3 # a_thresh = cv2.adaptiveThreshold(source, maxVal, adaptiveMethod, thresholdType, blockSize, constant)
4 # adaptive methods: cv2.ADAPTIVE_THRESH_MEAN_C, cv2.ADAPTIVE_THRESH_GAUSSIAN_C
5 a_thresh = cv2.adaptiveThreshold(gray,255,cv2.ADAPTIVE_THRESH_GAUSSIAN_C, cv2.THRESH_BINARY,9,1)
6 cv2.imshow("Adaptive",a_thresh)
In [7]: 1 import cv2
2 import numpy as np
3 path = 'image.jpeg'
4 # load image and get dimensions
5 img = cv2.imread(path)
6
```

Mask to give colour to the edges

```
localhost
Documents/Python practice/Assignments/Guided Proje... Project 8 incremental clustering - Jupyter Notebook Project on adaptive thresholding - Jupyter Notebook python - How to fill white background inside the edg...
jupyter Project on adaptive thresholding Last Checkpoint: 27 minutes ago (autosaved)
File Edit View Insert Cell Kernel Widgets Help Trusted Python 3
In [7]: 1 import cv2
2 import numpy as np
3 path = 'image.jpeg'
4 # load image and get dimensions
5 img = cv2.imread(path)
6
7 # convert to hsv
8 hsv = cv2.cvtColor(img,cv2.COLOR_BGR2HSV)
9
10 # threshold using inRange
11 range1 = (50,0,50)
12 range2 = (120,120,170)
13 mask = cv2.inRange(hsv,range1,range2)
14
15 # invert mask
16 mask = 255 - mask
17
18 # apply morphology closing and opening to mask
19 kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (2,2))
20 mask = cv2.morphologyEx(mask, cv2.MORPH_CLOSE, kernel)
21 mask = cv2.morphologyEx(mask, cv2.MORPH_OPEN, kernel)
22
23 result = img.copy()
24 result[mask==0] = (255,255,255)
25
26 # write result to disk
27 cv2.imwrite("fish_mask.png", mask)
28 cv2.imwrite("fish_with_white_background.jpg", result)
29
30 # display it
31 cv2.imshow("mask", mask)
32 cv2.imshow("result", result)
33 cv2.waitKey(1)
34 cv2.destroyAllWindows()
In [ ]: 1
```

