**A Synopsis on**

**IMAGE SEGMENTATION SYSTEM**

**Under the Guidance (Mentor) of**

**Aishwariya Saxena**

**By – Jatan Choudhary**

**Machine Learninng – Python**

**College – LNCT Bhopal**

**Winter Interhip – November 2020 – January 2021**

|  |  |  |
| --- | --- | --- |
| **SR.**  **NO.** | **TOPIC** | **PAGE NO.** |
| **1** | **INTRODUCTION** | **3** |
| **2** | **OBJECTIVE** | **3** |
| **3** | **BACKGROUND** | **4** |
| **4** | **HARDWARE AND SOFTWARE REQUIREMENTS** | **5** |
| **5** | **CODING** | **6** |
| **6** | **OUTPUT SCREENSHOTS** |  |
| **7** | **FUTURE SCOPE** |  |
| **6** | **CONCLUSION** | **6** |
| **7** | **BIBILOGRAPHY AND REFERENCES** | **7** |

**INDEX**

**IMAGE SEGMENTATION**

**Introduction**

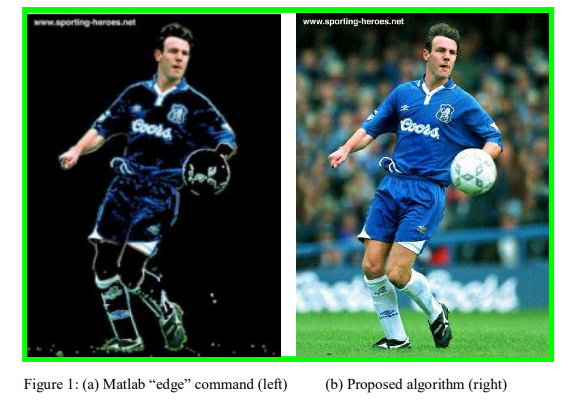
The purpose of image segmentation is to partition an image into meaningful regions with respect to a particular application. The segmentation is based on measurements taken from the image and might be grey level, colour, texture, depth or motion. Usually image segmentation is an initial and vital step in a series of processes aimed at overall image understanding. Image segmentation is the foundation of object recognition and computer vision. In general, image noise should be eliminated through image pre processing. And there is some specifically-given work (such as region extraction and image marking) to do after the main operation of image segmentation for the sake of getting better visual effect.

**Objective**

* In image preprocessing, firstly, various color spaces should be transformed into specifically-given color space.
* Then some techniques such as Gaussian filter are used to smooth image to diminish the influence of noise.
* As the main body of image segmentation system, image segmentation algorithm determines the result of image segmentation.
* After that, region merging and region extraction are used to combine unreasonably discontinuous regions. All the efforts above can ensure a satisfying image segmentation result.

**Background**

There are many algorithms used for image segmentation, and some of them segmented an image based on the object while some can segment automatically. Nowadays, no one can point out which the optimal solution is due to different constraints. In , a similarity close measure was used to classify the belonging of the pixels, and then used region growing to get the object. Unfortunately, it required a set of markers, and if there is an unknown image, it is hard to differentiate which part should be segmented. Linking the area information and the color histogram were considered for building video databases based on objects. However, the color information has to be given first, and it is not useful for the life application. A genetic algorithm adapted the segmentation process to changes in image characteristics caused by variable environmental conditions, but it took time learning. In , a two-step approach to image segmentation is reported. It was a fully automated model-based image segmentation, and improved active shape models, line-lanes and live-wires, intelligent scissors, core-atoms, active appearance models. However, there were still two problems left. It is strong dependency on a close-to-target initialization, and necessary for manual redesign of segmentation criteria whenever new segmentation problem is encountered. The authors in, proposed a graph-based method, the cut ratio is defined following the idea of NP-hard as the ratio of the corresponding sums of two different weights of edges along the cut boundary and models the mean affinity between the segments separated by the boundary per unit boundary length. It allows efficient iterated region-based segmentation as well as pixel-based segmentation. Moreover, in order to understand an image and recognize the represented objects, it is necessary to locate in the image where the objects are.



**HARDWARE AND SOFTWARE REQUIREMENTS**

**Hardware Reuirement:**

|  |  |
| --- | --- |
| **Hardware Tools** | **Minimum Requirements** |
| **Processor** | **I3 or above** |
| **HDD** | **10 GB** |
| **RAM** | **8 GB** |
| **Monitor** | **17 coloured** |
| **Mouse** | **Optical** |
| **Keyboard** | **122 Keys** |

**Software Requirement:**

|  |  |
| --- | --- |
| **Software Tools** | **Minimum Requirements** |
| **Plateform** | **Windows, Linux, Mac** |
| **OS** | **Windows, Linux, Mac** |
| **Technology** | **Machine Learning – Python** |
| **Programming Language** | **Python** |
| **IDE** | **Pycharm or Vscode** |

**Coding**

'''

===============================================================================

Interactive Image Segmentation using GrabCut algorithm.

`

This sample shows intera`ctive image segmentation using grabcut algorithm.

USAGE:

    python grabcut.py <filename>

README FIRST:

    Two windows will show up, one for input and one for output.

    At first, in input window, draw a rectangle around the object using

mouse right button. Then press 'n' to segment the object (once or a few times)

For any finer touch-ups, you can press any of the keys below and draw lines on

the areas you want. Then again press 'n' for updating the output.

Key '0' - To select areas of sure background

Key '1' - To select areas of sure foreground

Key '2' - To select areas of probable background

Key '3' - To select areas of probable foreground

Key 'n' - To update the segmentation

Key 'r' - To reset the setup

Key 's' - To save the results

===============================================================================

'''

# Python 2/3 compatibility

from \_\_future\_\_ import print\_function

import numpy as np

import cv2

import sys

BLUE = [255,0,0]        # rectangle color

RED = [0,0,255]         # PR BG

GREEN = [0,255,0]       # PR FG

BLACK = [0,0,0]         # sure BG

WHITE = [255,255,255]   # sure FG

DRAW\_BG = {'color' : BLACK, 'val' : 0}

DRAW\_FG = {'color' : WHITE, 'val' : 1}

DRAW\_PR\_FG = {'color' : GREEN, 'val' : 3}

DRAW\_PR\_BG = {'color' : RED, 'val' : 2}

# setting up flags

rect = (0,0,1,1)

drawing = False         # flag for drawing curves

rectangle = False       # flag for drawing rect

rect\_over = False       # flag to check if rect drawn

rect\_or\_mask = 100      # flag for selecting rect or mask mode

value = DRAW\_FG         # drawing initialized to FG

thickness = 3           # brush thickness

def onmouse(event,x,y,flags,param):

    global img,img2,drawing,value,mask,rectangle,rect,rect\_or\_mask,ix,iy,rect\_over

    # Draw Rectangle

    if event == cv2.EVENT\_RBUTTONDOWN:

        rectangle = True

        ix,iy = x,y

    elif event == cv2.EVENT\_MOUSEMOVE:

        if rectangle == True:

            img = img2.copy()

            cv2.rectangle(img,(ix,iy),(x,y),BLUE,2)

            rect = (min(ix,x),min(iy,y),abs(ix-x),abs(iy-y))

            rect\_or\_mask = 0

    elif event == cv2.EVENT\_RBUTTONUP:

        rectangle = False

        rect\_over = True

        cv2.rectangle(img,(ix,iy),(x,y),BLUE,2)

        rect = (min(ix,x),min(iy,y),abs(ix-x),abs(iy-y))

        rect\_or\_mask = 0

        print(" Now press the key 'n' a few times until no further change \n")

    # draw touchup curves

    if event == cv2.EVENT\_LBUTTONDOWN:

        if rect\_over == False:

            print("first draw rectangle \n")

        else:

            drawing = True

            cv2.circle(img,(x,y),thickness,value['color'],-1)

            cv2.circle(mask,(x,y),thickness,value['val'],-1)

    elif event == cv2.EVENT\_MOUSEMOVE:

        if drawing == True:

            cv2.circle(img,(x,y),thickness,value['color'],-1)

            cv2.circle(mask,(x,y),thickness,value['val'],-1)

    elif event == cv2.EVENT\_LBUTTONUP:

        if drawing == True:

            drawing = False

            cv2.circle(img,(x,y),thickness,value['color'],-1)

            cv2.circle(mask,(x,y),thickness,value['val'],-1)

if \_\_name\_\_ == '\_\_main\_\_':

    # print documentation

    print(\_\_doc\_\_)

    # Loading images

    if len(sys.argv) == 2:

        filename = sys.argv[1] # for drawing purposes

    else:

        print("No input image given, so loading default image, C:\\Users\\hp\\Desktop\\Jatan.jpg \n")

        print("Correct Usage: python grabcut.py <filename> \n")

        filename = 'plane.jpg'

    img = cv2.imread(filename)

    img2 = img.copy()                               # a copy of original image

    mask = np.zeros(img.shape[:2],dtype = np.uint8) # mask initialized to PR\_BG

    output = np.zeros(img.shape,np.uint8)           # output image to be shown

    # input and output windows

    cv2.namedWindow('output')

    cv2.namedWindow('input')

    cv2.setMouseCallback('input',onmouse)

    cv2.moveWindow('input',img.shape[1]+10,90)

    print(" Instructions: \n")

    print(" Draw a rectangle around the object using right mouse button \n")

    while(1):

        cv2.imshow('output',output)

        cv2.imshow('input',img)

        k = cv2.waitKey(1)

        # key bindings

        if k == 27:         # esc to exit

            break

        elif k == ord('0'): # BG drawing

            print(" mark background regions with left mouse button \n")

            value = DRAW\_BG

        elif k == ord('1'): # FG drawing

            print(" mark foreground regions with left mouse button \n")

            value = DRAW\_FG

        elif k == ord('2'): # PR\_BG drawing

            value = DRAW\_PR\_BG

        elif k == ord('3'): # PR\_FG drawing

            value = DRAW\_PR\_FG

        elif k == ord('s'): # save image

            bar = np.zeros((img.shape[0],5,3),np.uint8)

            res = np.hstack((img2,bar,img,bar,output))

            cv2.imwrite('grabcut\_output.png',res)

            print(" Result saved as image \n")

        elif k == ord('r'): # reset everything

            print("resetting \n")

            rect = (0,0,1,1)

            drawing = False

            rectangle = False

            rect\_or\_mask = 100

            rect\_over = False

            value = DRAW\_FG

            img = img2.copy()

            mask = np.zeros(img.shape[:2],dtype = np.uint8) # mask initialized to PR\_BG

            output = np.zeros(img.shape,np.uint8)           # output image to be shown

        elif k == ord('n'): # segment the image

            print(""" For finer touchups, mark foreground and background after pressing keys 0-3

            and again press 'n' \n""")

            if (rect\_or\_mask == 0):         # grabcut with rect

                bgdmodel = np.zeros((1,65),np.float64)

                fgdmodel = np.zeros((1,65),np.float64)

                cv2.grabCut(img2,mask,rect,bgdmodel,fgdmodel,1,cv2.GC\_INIT\_WITH\_RECT)

                rect\_or\_mask = 1

            elif rect\_or\_mask == 1:         # grabcut with mask

                bgdmodel = np.zeros((1,65),np.float64)

                fgdmodel = np.zeros((1,65),np.float64)

                cv2.grabCut(img2,mask,rect,bgdmodel,fgdmodel,1,cv2.GC\_INIT\_WITH\_MASK)

        mask2 = np.where((mask==1) + (mask==3),255,0).astype('uint8')

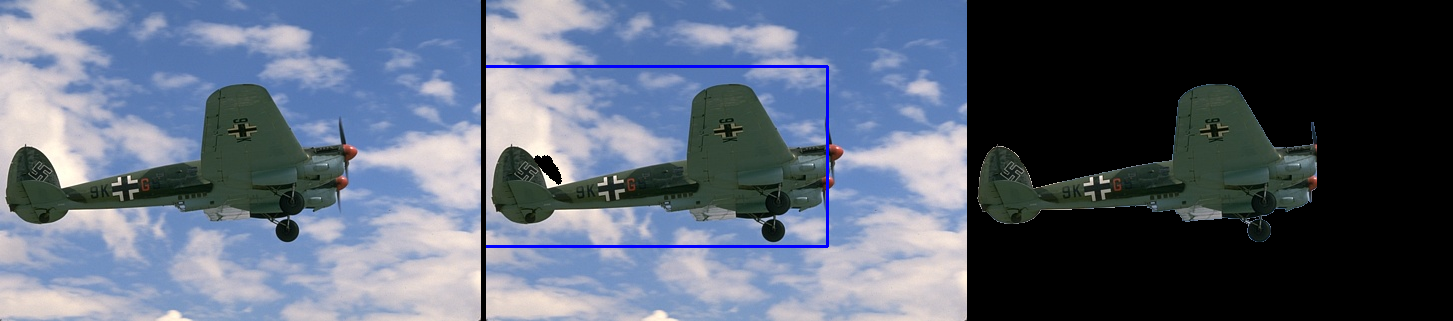
        output = cv2.bitwise\_and(img2,img2,mask=mask2)

    cv2.destroyAllWindows()

**Output Screenshots**



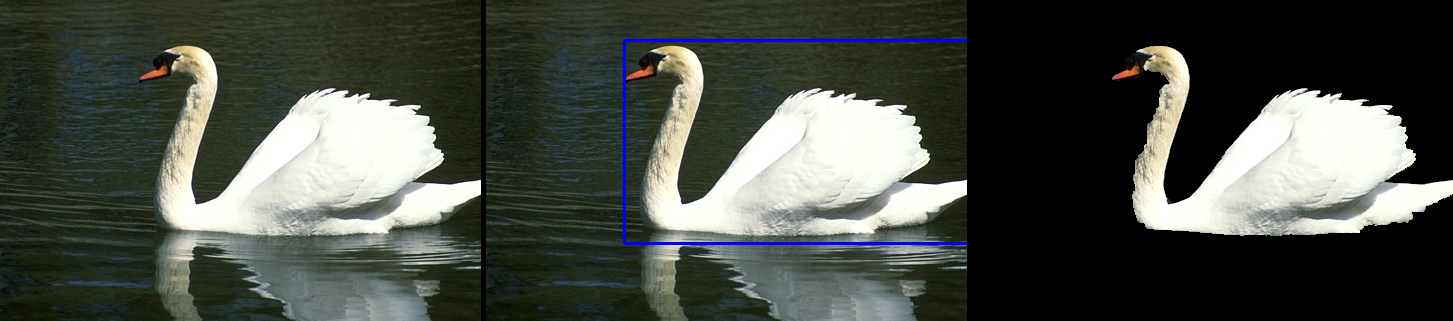
Input 1



Output 1



Input 2



Output 2[Error! Hyperlink reference not valid.]( )

**Future Scope**

The future of image processing will involve scanning the heavens for other intelligent life out in space. Also new intelligent, digital species created entirely by research scientists in various nations of the world will include advances in image processing applications. Due to advances in image processing and related technologies there will be millions and millions of robots in the world in a few decades time, transforming the way the world is managed. Advances in image processing and artificial intelligence6 will involve spoken commands, anticipating the information requirements of governments, translating languages, recognizing and tracking people and things, diagnosing medical conditions, performing surgery, reprogramming defects in human DNA, and automatic driving all forms of transport. With increasing power and sophistication of modern computing, the concept of computation can go beyond the present limits and in future, image processing technology will advance and the visual system of man can be replicated. The future trend in remote sensing will be towards improved sensors that record the same scene in many spectral channels. Graphics data is becoming increasingly important in image processing app1ications. The future image processing applications of satellite based imaging ranges from planetary exploration to surveillance applications.

**Conclusion**

A major challenge for automatic image analysis is that the sheer complexity of the visual task which has been mostly ignored by the current approaches. New technological breakthrough in the areas of digital computation and telecommunication has relevance for future applications of image processing1. The satellite imaging and remote sensing applications programs of the future will feature a variety of sensors orbiting the earth. This technology is required for military and other types of surveillance, statistical data collection in the fields of forestry, agriculture, disaster prediction, weather prediction. In order to extract scientifically useful information, it will be necessary to develop techniques to register real-time data recorded by a variety of sensors for various applications3.

**Bibilography and Refrences:**

[1] Datasets (<https://www.kaggle.com/>)

[2] Sklearn Libraries (<https://scikit-learn.org/stable/>)

[3] “Digital Image Processing” by Rafael C Gonzalez and Richard E Woods

[4] “Digital Image Processing” by William K Pratt

[5] “Digital Image Processing” by S Sridhar

[6] “DIGITAL IMAGE PROCESSING & ANALYSIS” by Bhabatosh Chanda