Project-ObjectErase
IMPR

DEFINITION

ObjectErase is a lightweight Python package for removing small objects from live image. It uses image processing techniques to detect and erase objects below a certain size threshold.

Features:

- **Detects and removes small objects from live image.**

Dependencies:

- **NumPy**
- **OpenCV**

Uses:

- **It is mainly used in movies and Tv shows.**

  (Example:- harry potter invisibility cloak)

Code:-

```
import cv2

import numpy




def example(x):

   print("hello")




cap = cv2.VideoCapture(0)

bars = cv2.namedWindow("bars")



# Medium-Dark blue color hsv value

cv2.createTrackbar("upper_hue", "bars", 130, 180, example)

cv2.createTrackbar("upper_saturation", "bars", 255, 255, example)

cv2.createTrackbar("upper_value", "bars", 255, 255, example)

cv2.createTrackbar("lower_hue", "bars", 110, 180, example)

cv2.createTrackbar("lower_saturation", "bars", 50, 255, example)

cv2.createTrackbar("lower_value", "bars", 50, 255, example)




# Capturing the initial frame for creation of background

while (True):

   cv2.waitKey(1000)

   ret, init_frame = cap.read()
```

```
    # check if the frame is returned then brake
    if (ret):
        break


# Start capturing the frames for actual magic!!
while (True):
    ret, frame = cap.read()
    inspect = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)


    # getting the HSV values for masking the cloak
    upper_hue = cv2.getTrackbarPos("upper_hue", "bars")
    upper_saturation = cv2.getTrackbarPos("upper_saturation", "bars")
    upper_value = cv2.getTrackbarPos("upper_value", "bars")
    lower_value = cv2.getTrackbarPos("lower_value", "bars")
    lower_hue = cv2.getTrackbarPos("lower_hue", "bars")
    lower_saturation = cv2.getTrackbarPos("lower_saturation", "bars")


    # Kernel to be used for dilation
    kernel = numpy.ones((3, 3), numpy.uint8)


    upper_hsv = numpy.array([upper_hue, upper_saturation, upper_value])
    lower_hsv = numpy.array([lower_hue, lower_saturation, lower_value])


    mask = cv2.inRange(inspect, lower_hsv, upper_hsv)
```

```
# For minor noise remove

mask = cv2.medianBlur(mask, 3)

mask_inv = 255-mask

mask = cv2.dilate(mask, kernel, 5)



b = frame[:, :, 0]

g = frame[:, :, 1]

r = frame[:, :, 2]

b = cv2.bitwise_and(mask_inv, b)

g = cv2.bitwise_and(mask_inv, g)

r = cv2.bitwise_and(mask_inv, r)

object_area = cv2.merge((b, g, r))



b = init_frame[:, :, 0]

g = init_frame[:, :, 1]

r = init_frame[:, :, 2]

b = cv2.bitwise_and(b, mask)

g = cv2.bitwise_and(g, mask)

r = cv2.bitwise_and(r, mask)

Invisible_area = cv2.merge((b, g, r))



final = cv2.bitwise_or(object_area, Invisible_area)



cv2.imshow("Original image", frame)

cv2.imshow("invisible pen", final)
```

```
    if (cv2.waitKey(3) == ord('q')):
        break

cv2.destroyAllWindows()
cap.release()
```

Output:-

Original image

Invisible implementation