harkesket model

21-10-25

Image and video processing

- → openev python
- → pil Python

→ open CV → open source computer vision library.
  → 2500 + algorithm

application oven cp
  → face recognition

need:
- → anaconda
- → python
- → openev

Agenda:- (today) inside openev

1. (I) Reading Images
2. (II) Reading videos
3. (III) Reading webcamara
4. (IV) Basic function
5. (V) cropping and resizing Images.
6. (VI) shape and texts
7. (VII) wrap perspective
8. (VIII) Joining Images

Day2 file
↳ reading images

cd ~~Day2~~/ Day 2/ (terminal)

```
import cv2
img = cv2.imread("path")
print(img)
print(img.shape)
cv2.imshow("windowname" img)
cv2.waitKey(0)
```

→ reading video

```
cap = cv2.videoCapture("path")
print(cap)
while true:
    ~~cap.read()~~
    success, frame = cap.read()
    print(frame.shape)
    cv2.imshow("output", img)

    if cv2.waitKey(1) & 0XFF == ord('q'):
        break.
```

→ reading webcam:

```
cap = cv2. videocapture (0)
cap. set (3, 640) #width
cap. set ( 4, 480) # Height

while true:
    success, img = cap. read()
    cv2. imshow ("output", img)
    if cv2. waitkey (1) & 0XFF = ord ('q'):
        break.
```

→ for 1 camara

→ basic function:-

2. basic function.py

```
import cv2
# convert color image to greyscall
img = cv2. imread (" path")
cv2. imshow (img)
     ~~waitkey~~ cv2. imshow ("Grayscale", img_gray)
    cv2. waitkey (0)

img_grey = cv2. cvtcolor (img, cv2.
                          color_BGR-to gray)
```

convert to blur (gray scale এ convert করতে হবে।)

→ ~~same~~ code image কে blur করবে

এক image এর edge select করে shape তৈরী করতে হয় যার জন্য color জানার দরকার

নাই, তাই channel কমিয়ে আনতেছি।

same code

```
img_blur = cv2.GaussianBlur(img_gray,
                            (7,7),0)
```

convert to canny image (canny endge detection)

same code —

```
img_canny = cv2.canny(img_blur, t_lower, t_upper)
                                100      100
```

→ cropping and resizing:

```
import cv2
img = cv2.imread("path")
~~print = cv2.imshow(~~
print (img.imshow)
resize_img = cv2.resize(img, 300,200)
```

cropping —

```
crop_img = img[0:200, 200:500]
```

→ shape and ~~font~~ text :

```
    import cv2
    import ~~n~~numpy as np
    img = np.zeros ((512, 512, 3), np.unit8)
    print =(img.shape)
    .img [:] = 255, 0, 0 # BGR
    cv2. imshow ("Imge", img)
     cv2. waitkey (0)

# create line
   cv2. line (img, (0,0), (300,400), (0,255,0), 3)
   cv2. imshow ("Imag line", img)
   cv2. waitkey (0)
# rectangle
   cv2. rectangle (img, (20,50), (350, 250), (0,0,255), 5)
     cv2. imshow ("rectangle", img)
     cv2. waitkey (0)

# circle
  cv2.circle (img, (400,50), 50, (0,255, 0), 10)
# put text.
cv2. puttext (img, "tanim", (200, 440),
              cv2. Font_Harshey_complex, 1, (0,255,0), 5)
```

→ wrap prespective:

```
import cv2
import numpy as np

width, heigh = 250, 150
img = cv2.imread ("path")

pts1 = np.float32([[752, 118],
        [1120, 265], [540, 668], [871, 838]])

pts2 = np.float32([[0, 0], [width, 0],
                   [height, 0], [width, height]])

metrix = cv2.getperpective Transform(pts1, pts2)
img_out = cv2.warpperspective(img, metrix,
                              (witdth, height))

cv2.imshow('cards', img)
cv2.imshow('cards_warp', img_out)
cv2.imshow('cards', img)
cv2.waitkey(0)
```

→ Joining img:-

```
import cv2
import numpy as np
img = cv2.imread ( "path")
img_hor = np. hstack ((img, img))
img_var = np. vstack((img, img))
cv2. imshow ( "Horizontal", img_hor)
cv2. imshow ( "vertical" , img_var)
cv2. waitkey (0)
```