**linkuri care s-ar putea sa ne ajute:**

[**https://www.pyimagesearch.com/2015/03/30/accessing-the-raspberry-pi-camera-with-opencv-and-python/**](https://www.pyimagesearch.com/2015/03/30/accessing-the-raspberry-pi-camera-with-opencv-and-python/)

[**https://pysource.com/2018/09/25/simple-shape-detection-opencv-with-python-3/**](https://pysource.com/2018/09/25/simple-shape-detection-opencv-with-python-3/)

[**https://pysource.com/2018/12/29/real-time-shape-detection-opencv-with-python-3/**](https://pysource.com/2018/12/29/real-time-shape-detection-opencv-with-python-3/)

**Captura de imagine cu OpenCV**

# import pentru pachetele necesare

**from picamera.array import PiRGBArray**

**from picamera import PiCamera**

**import time**

**import cv2**

# initializam camera, schimbam rezolutia si capturam o referinta a camerei

**camera = PiCamera()**

**camera.resolution = (640, 480)**

**rawCapture = PiRGBArray(camera,size=(640, 480))**

# lăsăm 2 secunde ca sa isi faca camera inițializarea

**time.sleep(2)**

# reusim sa prinde o imagine de la camera

**camera.capture(rawCapture, format="bgr")**

**image = rawCapture.array**

# afisam imaginea

**cv2.imshow("Image", image)**

**cv2.waitKey(0)**

**Real time cu OpenCV**

# import pentru pachetele necesare

**from picamera.array import PiRGBArray**

**from picamera import PiCamera**

**import time**

**import cv2**

# initializam camera, schimbam rezolutia si capturam o referinta a camerei

**camera = PiCamera()**

**camera.resolution = (640, 480)**

**camera.framerate = 32**

**rawCapture = PiRGBArray(camera, size=(640, 480))**

# lăsăm 0.1 secunde ca sa isi faca camera inițializarea

**time.sleep(0.1)**

# capturam încontinuu cadre de la camere

**for frame in camera.capture\_continuous(rawCapture, format="bgr",use\_video\_port=True):**

#imaginea primește cadrele capturate

**image = frame.array**

# afișează cadrele

**cv2.imshow("Frame", image)**

**key = cv2.waitKey(1) & 0xFF**

# ștergem ce avem pentru pregătirea următorului cadrul

**rawCapture.truncate(0)**

#dacă apasam tasta q ieșim din for

**if key == ord("q"):**

**break**

**Procesare de imagine**

1. **Transforma o imagine obișnuită într-o imagine alb-negru(grayscale)**

* folosim funcția **cv2.cvtColor(image,cv2.COLOR\_BGR2GRAY)**

# import pentru pachetele necesare

**from picamera.array import PiRGBArray**

**from picamera import PiCamera**

**import time**

**import cv2**

# initializam camera, schimbam rezolutia si capturam o referinta a camerei

**camera = PiCamera()**

**camera.resolution = (640, 480)**

**rawCapture = PiRGBArray(camera,size=(640, 480))**

# lăsăm 2 secunde ca sa isi faca camera inițializarea

**time.sleep(2)**

# reusim sa prinde o imagine de la camera

**camera.capture(rawCapture, format="bgr")**

**image = rawCapture.array**

# transforma o imagine obișnuită într-o imagine alb-negru

**gray = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)**

# afisam imaginile

**cv2.imshow("Image", image)**

**cv2.imshow("Image1", gray)**

**cv2.waitKey(0)**

1. **Bluram imaginea alb-negru**

* folosim funcția **blurred = cv2.GaussianBlur(gray, (5, 5), 0)**

# import pentru pachetele necesare

**from picamera.array import PiRGBArray**

**from picamera import PiCamera**

**import time**

**import cv2**

# initializam camera, schimbam rezolutia si capturam o referinta a camerei

**camera = PiCamera()**

**camera.resolution = (640, 480)**

**rawCapture = PiRGBArray(camera,size=(640, 480))**

# lăsăm 2 secunde ca sa isi faca camera inițializarea

**time.sleep(2)**

# reusim sa prinde o imagine de la camera

**camera.capture(rawCapture, format="bgr")**

**image = rawCapture.array**

# transforma o imagine obișnuită într-o imagine alb-negru

**gray = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)**

# bluram imaginea

**blurred = cv2.GaussianBlur(gray, (5, 5), 0)**

# afisam imaginile

**cv2.imshow("Image", image)**

**cv2.imshow("Image1", gray)**

**cv2.imshow("Image2", blurred)**

**cv2.waitKey(0)**

1. **Threshold(binarizare)**

* folosim functia **\_, thresh = cv2.threshold(blurred, 80, 255, cv2.THRESH\_BINARY)**

# import pentru pachetele necesare

**from picamera.array import PiRGBArray**

**from picamera import PiCamera**

**import time**

**import cv2**

# initializam camera, schimbam rezolutia si capturam o referinta a camerei

**camera = PiCamera()**

**camera.resolution = (640, 480)**

**rawCapture = PiRGBArray(camera,size=(640, 480))**

# lăsăm 2 secunde ca sa isi faca camera inițializarea

**time.sleep(2)**

# reusim sa prinde o imagine de la camera

**camera.capture(rawCapture, format="bgr")**

**image = rawCapture.array**

# transforma o imagine obișnuită într-o imagine alb-negru

**gray = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)**

# bluram imaginea

**blurred = cv2.GaussianBlur(gray, (5, 5), 0)**

# tot ce are culoare inchisa face negru ,restul il face alb

**\_, thresh = cv2.threshold(blurred, 80, 255, cv2.THRESH\_BINARY)**

# afisam imaginile

**cv2.imshow("Image", image)**

**cv2.imshow("Image2", blurred)**

**cv2.imshow("Image3", blurred)**

**cv2.waitKey(0)**

**Detecția de forme- prima data pe o imagine cu forme**



import cv2

import numpy as np

font = cv2.FONT\_HERSHEY\_COMPLEX\_SMALL

#luăm imaginea

img = cv2.imread("shapes.jpg")

gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY**)**

blurred = cv2.GaussianBlur(gray, (5, 5), 0)

\_, threshold = cv2.threshold(blurred, 80, 255, cv2.THRESH\_BINARY)

\_, contours, \_ = cv2.findContours(threshold, cv2.RETR\_TREE,cv2.CHAIN\_APPROX\_SIMPLE)

for cnt in contours:

approx = cv2.approxPolyDP(cnt, 0.01\*cv2.arcLength(cnt, True), True)

cv2.drawContours(img, [approx], 0, (255), 5)

x = approx.ravel()[0]

y = approx.ravel()[1]

if len(approx) == 3:

cv2.putText(blurred, "Triangle", (x, y), font, 1, (255))

elif len(approx) == 4:

cv2.putText(blurred, "Rectangle", (x, y), font, 1, (255))

elif len(approx) == 5:

cv2.putText(blurred, "Pentagon", (x, y), font, 1, (255))

elif 6 < len(approx) < 15:

cv2.putText(blurred, "Ellipse", (x, y), font, 1, (255))

else:

cv2.putText(blurred, "Circle", (x, y), font, 1, (255))

cv2.imshow("shapes", blurred)

cv2.imshow("Threshold", threshold)

cv2.waitKey(0)

cv2.destroyAllWindows()

**Detecția de forme- de la o imagine de la camera**

**Ar trebui sa avem o testare pe un hol alb altfel imi ca obiecte si alte lucruri care au culori inchise.**

from picamera.array import PiRGBArray

from picamera import PiCamera

import time

import cv2

import numpy as np

camera = PiCamera()

camera.resolution=(640,480)

rawCapture = PiRGBArray(camera,size=(640,480))

time.sleep(1)

camera.capture(rawCapture, format="bgr")

img = rawCapture.array

font = cv2.FONT\_HERSHEY\_COMPLEX\_SMALL

gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

blurred = cv2.GaussianBlur(gray, (5, 5), 0)

\_, threshold = cv2.threshold(blurred,30, 255, cv2.THRESH\_BINARY)

\_, contours, \_ = cv2.findContours(threshold, cv2.RETR\_TREE, cv2.CHAIN\_APPROX\_SIMPLE)

for cnt in contours:

approx = cv2.approxPolyDP(cnt, 0.01\*cv2.arcLength(cnt, True), True)

cv2.drawContours(img,[approx],0,(255),5)

x=approx.ravel()[0]

y=approx.ravel()[1]

if len(approx) == 3:

cv2.putText(blurred, "Triangle", (x, y), font, 1, (255))

elif len(approx) == 4:

cv2.putText(blurred, "Rectangle", (x, y), font, 1, (255))

elif len(approx) == 5:

cv2.putText(blurred, "Pentagon", (x, y), font, 1, (255))

else:

cv2.putText(blurred, "Circle", (x, y), font, 1, (255))

cv2.imshow("imagine",img)

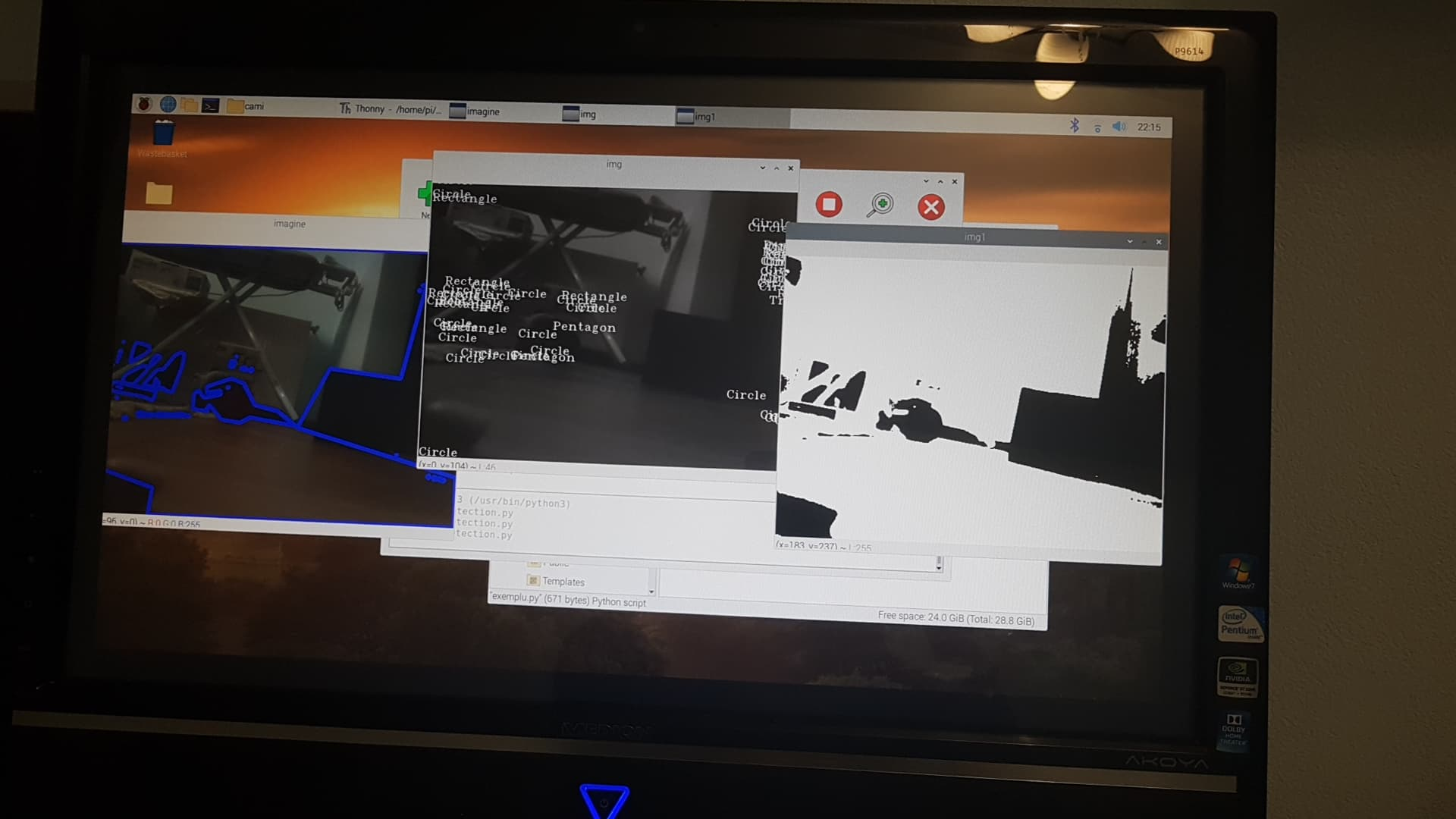
cv2.imshow("img", blurred)

cv2.imshow ("img1", threshold)

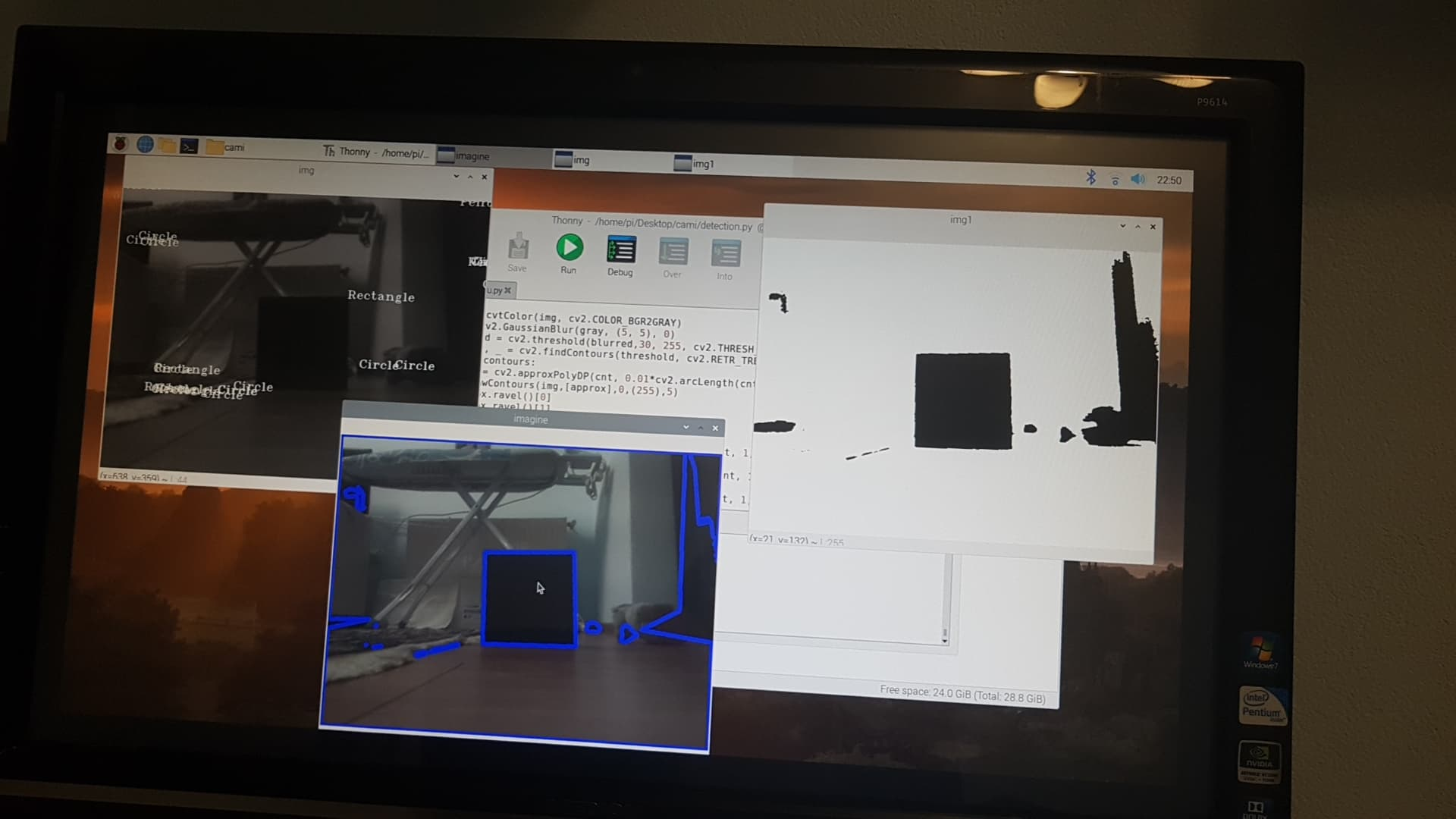
cv2.waitKey(0)

cv2.destroyAllWindows()

**Dacă nu avem culori deschise - e cam un eșec**

****

**Dacă avem culori deschise atunci detecția merge**

****

**Detecția de forme- real time**

**Algoritmul acesta merge doar pe un hol alb și sa nu avem umbre și sa nu avem schimbări de lumina altfel detecteaza aiurea(se poate vedea chestia asta in filmarile facute de noi) ->nu prea este eficient algoritmul ,căutăm altceva mai bun.**

from picamera.array import PiRGBArray

from picamera import PiCamera

import time

import cv2

import numpy as np

font = cv2.FONT\_HERSHEY\_COMPLEX\_SMALL

camera = PiCamera()

camera.resolution = (640, 480)

camera.framerate = 32

rawCapture = PiRGBArray(camera, size=(640, 480))

time.sleep(0.1)

for frame in camera.capture\_continuous(rawCapture, format="bgr",use\_video\_port=True):

image = frame.array

gray = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)

blurred = cv2.GaussianBlur(gray, (5, 5), 0)

\_, threshold = cv2.threshold(blurred,50, 255, cv2.THRESH\_BINARY)

\_, contours, \_ = cv2.findContours(threshold, cv2.RETR\_TREE, cv2.CHAIN\_APPROX\_SIMPLE)

for cnt in contours:

approx = cv2.approxPolyDP(cnt, 0.01\*cv2.arcLength(cnt, True), True)

cv2.drawContours(image,[approx],0,(255),5)

x=approx.ravel()[0]

y=approx.ravel()[1]

if len(approx) == 3:

cv2.putText(blurred, "Triangle", (x, y), font, 1, (255))

elif len(approx) == 4:

cv2.putText(blurred, "Rectangle", (x, y), font, 1, (255))

elif len(approx) == 5:

cv2.putText(blurred, "Pentagon", (x, y), font, 1, (255))

else:

cv2.putText(blurred, "Circle", (x, y), font, 1, (255))

cv2.imshow("Frame", image)

cv2.imshow("img", blurred)

cv2.imshow ("img1", threshold)

key = cv2.waitKey(1) & 0xFF

rawCapture.truncate(0)

if key == ord("q"):

break