

Projekt - PT

KinArt - Raport nr 1

...

Hanyż Małgorzata
Waligóra Cezary
Szkudlarek Damian

27.03.2019

GOSIA

tk

—

□

×

PEN

eraser






```
# 4 COLORS
self.color_button1 = Button(self.root, bg='green', height=3, width=7, command=self.green_color)
self.color_button1.grid(row=0, column=2)
```




```
# PAINTING
self.painting = Canvas(self.root, bg='white', width=600, height=500)
self.painting.grid(row=1, columnspan=6)
```

```
def green_color(self):  
    self.eraser_on = False  
    self.color = 'green'  
    self.activate_button_color(self.color_button1)
```



```
def activate_button_color(self, some_button, eraser_mode=False):  
    self.active_button_color.config(relief=RAISED) # border decoration  
    some_button.config(relief=SUNKEN)  
    self.active_button_color = some_button  
    self.eraser_on = eraser_mode  
    self.activate_button(self.pen_button)
```

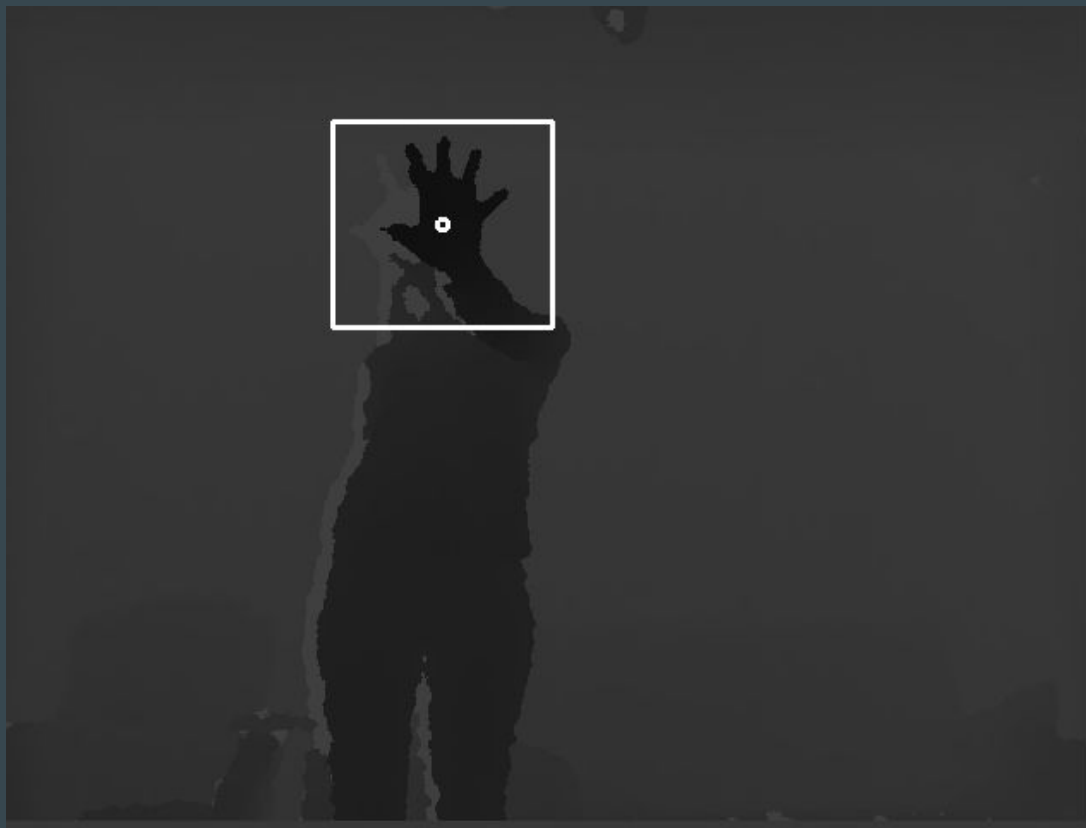
```
def setup(self):
    self.old_x = None
    self.old_y = None
    self.line_width = 20
    self.color = self.DEFAULT_COLOR
    self.eraser_on = False
    self.active_button = self.pen_button
    self.activate_button(self.active_button)
    self.active_button_color = self.color_button4
    self.activate_button_color(self.active_button_color)
    self.painting.bind('<B1-Motion>', self.paint)
    self.painting.bind('<ButtonRelease-1>', self.reset)
```



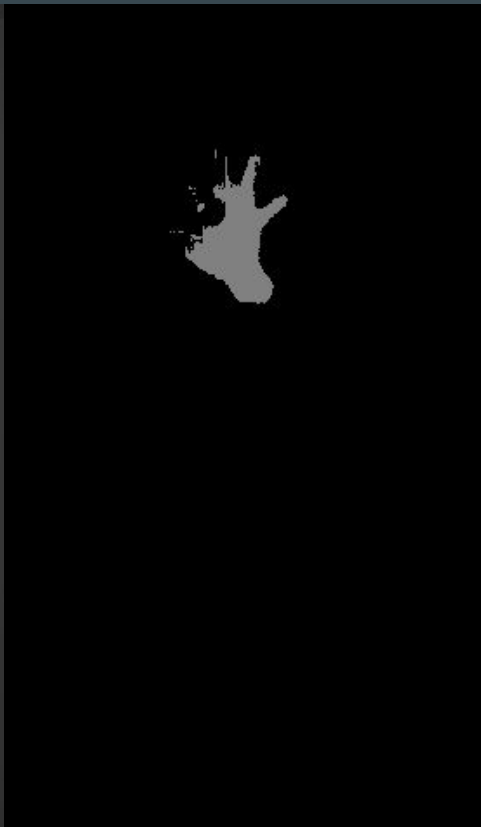
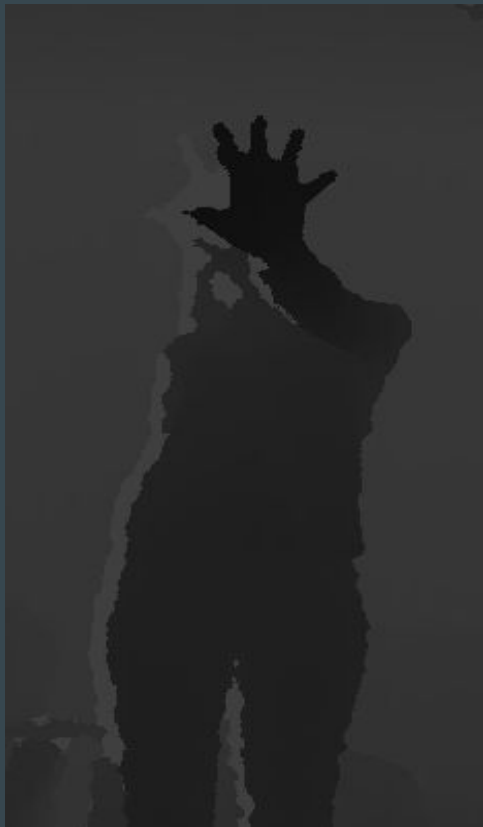
```
def paint(self, event):
    self.line_width = 20.0 if self.eraser_on else 5.0
    paint_color = 'white' if self.eraser_on else self.color
    if self.old_x and self.old_y:
        self.painting.create_line(self.old_x, self.old_y, event.x, event.y,
                                   width=self.line_width, fill=paint_color,
                                   capstyle=ROUND, smooth=TRUE, splinesteps=36)
        self.old_x = event.x
        self.old_y = event.y

def reset(self, event):
    self.old_x, self.old_y = None, None
```

DAMIAN



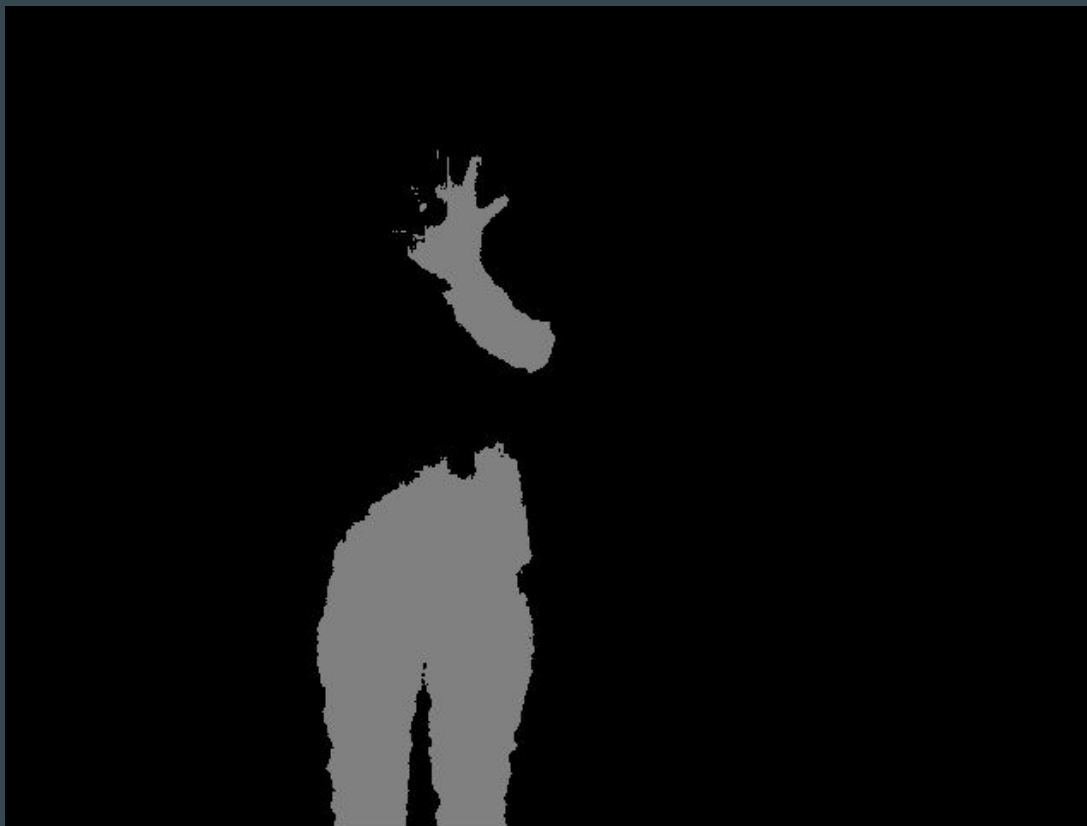
Wskazanie obszaru, w którym znaleźć ma się dłoń



```
1 med_val = np.median(center)
2 med_condition = 5
3 print(med_val)
4 cv2_imshow(gray_copy)
5
6 gray_copy = np.where((abs(gray_copy - int(med_val)) <= med_condition), 128, 0)
7 gray_copy = gray_copy.astype('uint8')
8
9 cv2_imshow(gray_copy)
```



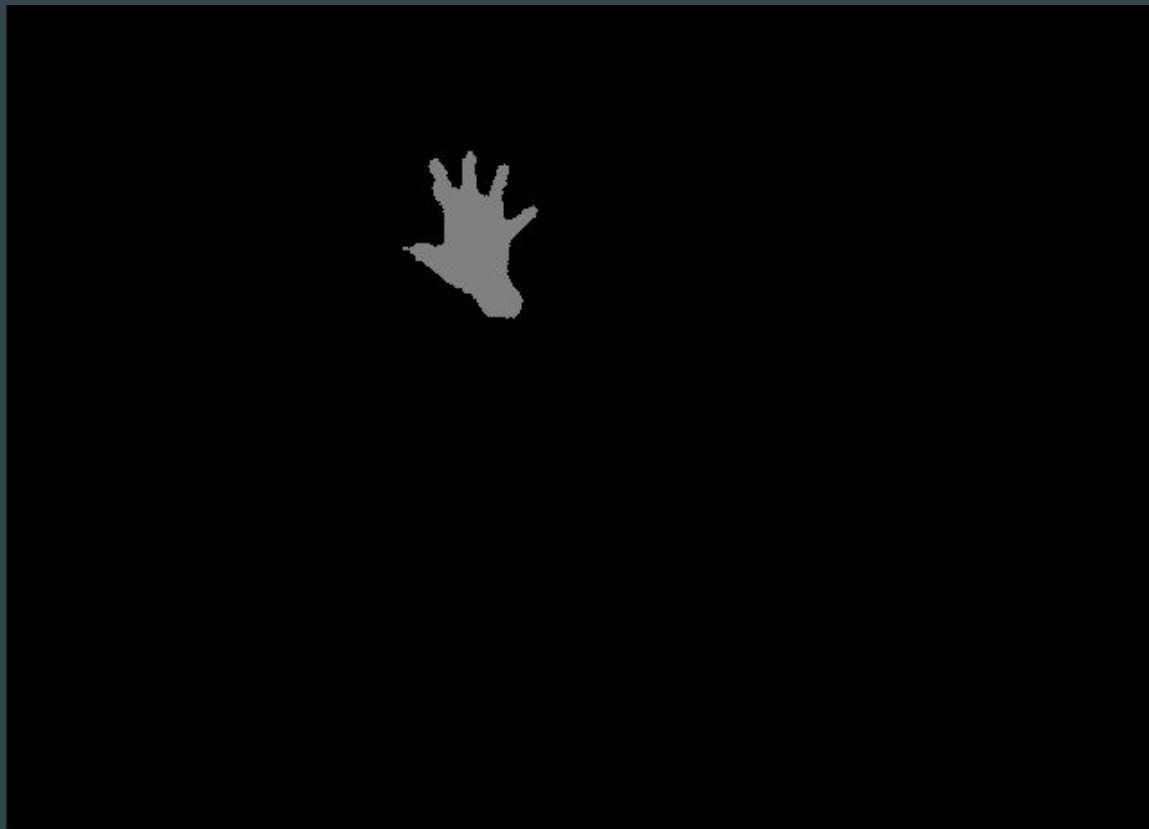
Próba filtracji zdjęcia na podstawie wartości głębi dłoni



Powiększanie granicy nie pomaga

$$16 - 17 = 255^*$$

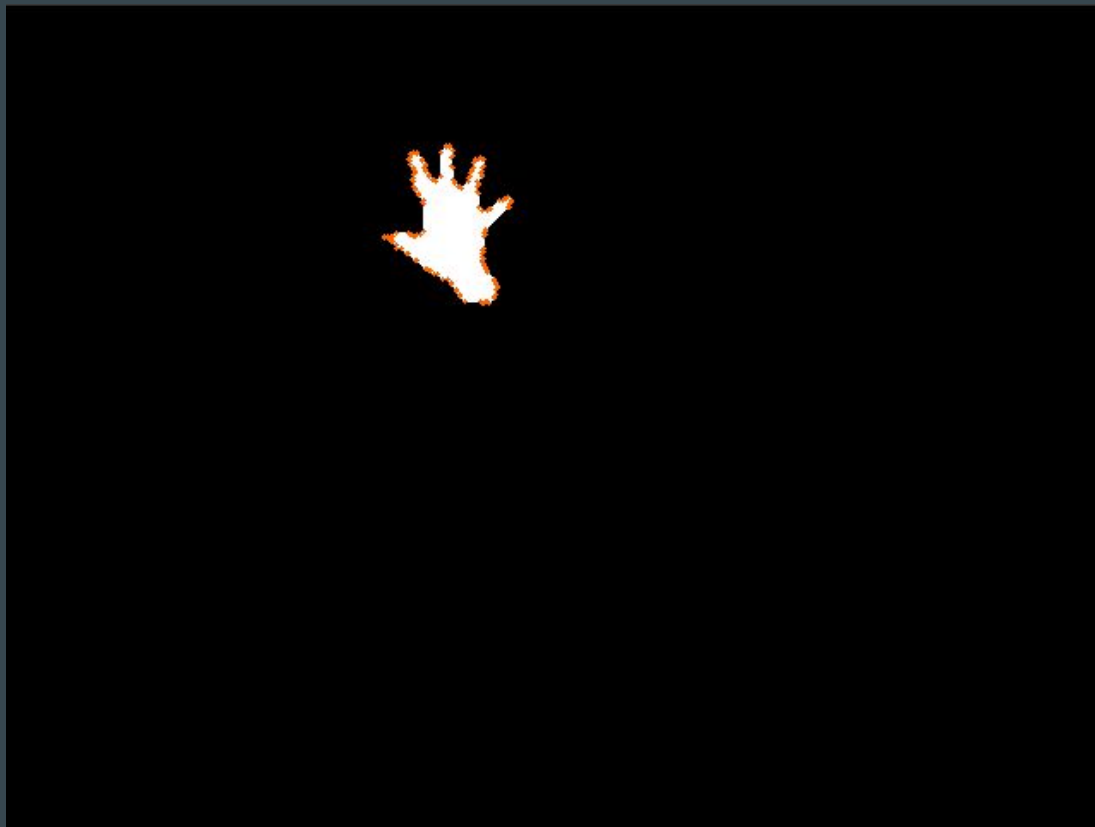
*przedstawione liczby są liczbami 8-bitowymi, całkowitymi, bez znaku



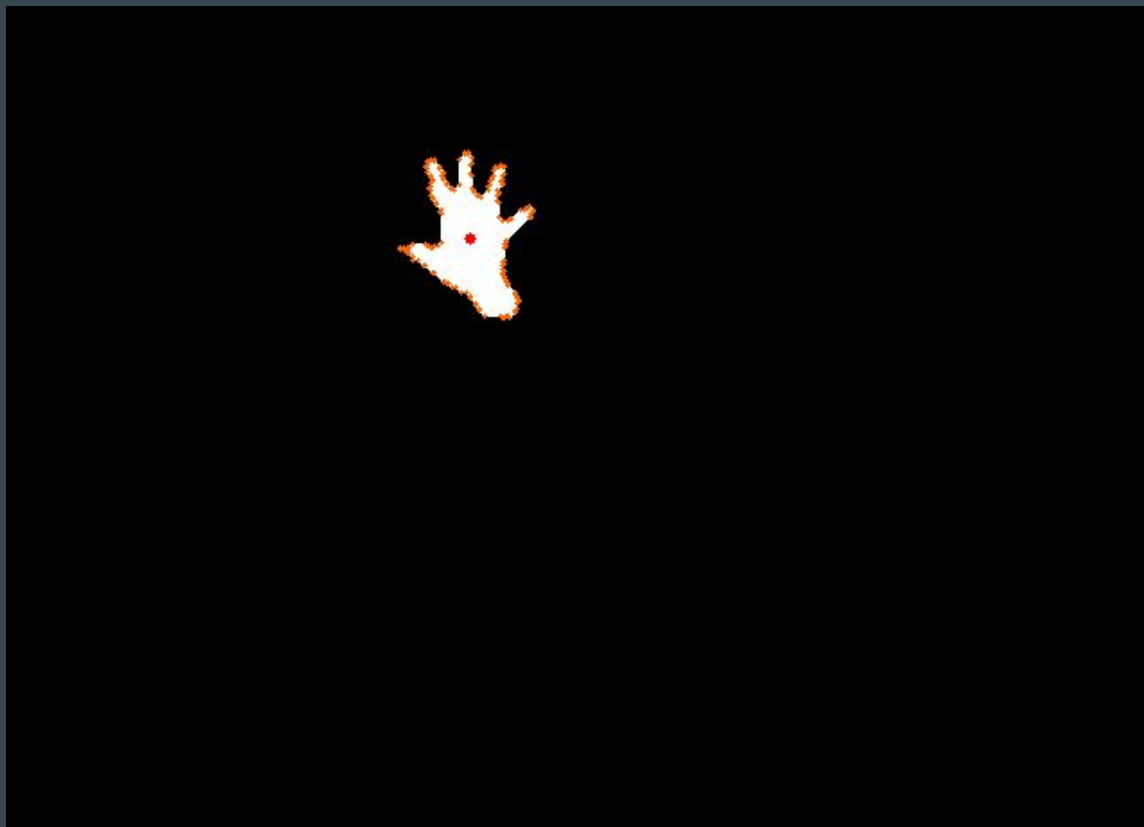
Morfologiczne domknięcie wykonane na obrazie



Flood fill



Obrysowanie konturu



Obliczenie momentu konturu i zaznaczenie go

CEZARY

Klasyfikator kaskadowy cech Haar-podobnych

```
cv2.CascadeClassifier  
detectMultiScale()
```



Podsumowanie

- Udało się spełnić założenia harmonogramu
- Następny etap to rysowanie na podstawie położenia dłoni i poprawa dokładności jej lokalizacji

Uwagi

- Przetestować inne algorytmy lokalizowania dłoni - być może na kamerze FullHD, a nie Kinect