Air Canvas

Membrii echipei:
Cincu Andrada Maria Alexandra
Chira Andreea-Marina

Cuprins

Capitolul 1

Descriere

Acest proiect, presupune desenarea in aer prin accesarea camerei web a laptopului, pe care o vom folosi pentru a desena in aer, prin utilizarea diferitor obiecte(ex: un pix). Rezultatul desenat apare si pe camera web, dar si intr-un canvas cu fundal alb. Iar pentru desenare ne putem alege diferite culori si putem sa stergem si ce am realizat.

Capitolul 2

Prezentare Generala

In principal ne folosim de functiile oferite de opencv, numpy, collection-deque.
In primul rand, detectam culorile cu care vrem sa scriem, iar apoi sa pregatim canvas-ul unde apare ce a fost scris/desenat si sa pregatim si canvas-ul care detecteaza obiectul cu care scriem.se pastreaza coordonatele unui contur pe care le retinem intr-un array pentru a putea desena si pe frame, dar si pe canvas.
In mare parte, un schelet al implementarii urmeaza urmatoarea structura: detectam culorile cu care vrem sa scriem, le afisam pe frame pentru a putea fi accesate, apoi ne ocupam de detectarea obiectului cu care vrem sa scriem si pregătim canvas-ul în care apare ce s-a scris/desenat pe frame-ul initial.

Algoritm:

- Incepe citirea cadrelor si convertirea cadrelor capturate in spatiul de culoare HSV (lumină de detectare a culorilor).
- Se pregateste canvas-ul si se pun butoanele cu care se va alege culoare pentru a putea scrie.
- Se regleaza valorile din track bar pentru a detecta obiectul de scriere.
- Se prelucreaza canvas-ul de detectare a obiectului
- Se detecteaza conturul obiectlui, se gasesc coordonatele centrale ale acestuia si se pastreaza intr-o matrice pentru a oferii cadre succesive (matrice pentru desenarea punctelor pe canvas).
- In cele din urma, punctele stocate intr-o matrice sunt desenate pe cadrul principal si pe canvas.

Capitolul 3

Contributii Personale

Modul de lucru al acestui proiect consta in patru puncte majore:

- Intelegerea spatiului de culoare HSV (nuanta, saturatie, valoare) pentru urmarirea culorilor si urmarirea obiectului colorat.
- Detectarea pozitiei obiectului colorat si formarea unui cerc deasupra acestuia. Aceasta este detectarea conturului.
- Urmarirea obiectului si desenarea de puncte in fiecare pozitie pentru efectul de panza de aer. Aceasta este procesarea cadrelor.
- Corectarea detaliilor minore ale codului pentru ca programul sa functioneze fara probleme. Optimizarea algoritmica.

HSV (pentru nuanta, saturatie, valoare; pentru nuanta, saturatie, luminozitate) sunt reprezentari alternative ale modelului de culoare RGB. In aceste modele, culorile din fiecare nuanta sunt dispuse intr-o felie radiala, in jurul unei axe centrale de culori neutre care variaza de la negru in partea de jos pana la alb in partea de sus.
Acest trackbars este utilizat pentru a ajusta canalele de nuanta, saturatie si valoare ale imaginii. Reglam barele de urmarire pana cand numai obiectul tinta este vizibil, iar restul este negru.

3.1 Test Trackbar

Figura 3.1: Trackbar

Capitolul 4

Rezultate

Odata cu rularea aplicatiei Air Canvas, aceasta va afisa un tracking bar si 3 canvas-uri:

- Web Cam
- White Canvas
- Balck Canvas

Web Cam reprezinta canvas-ul principal care captureaza obiectul si culoarea pentru afisarea desenelor.

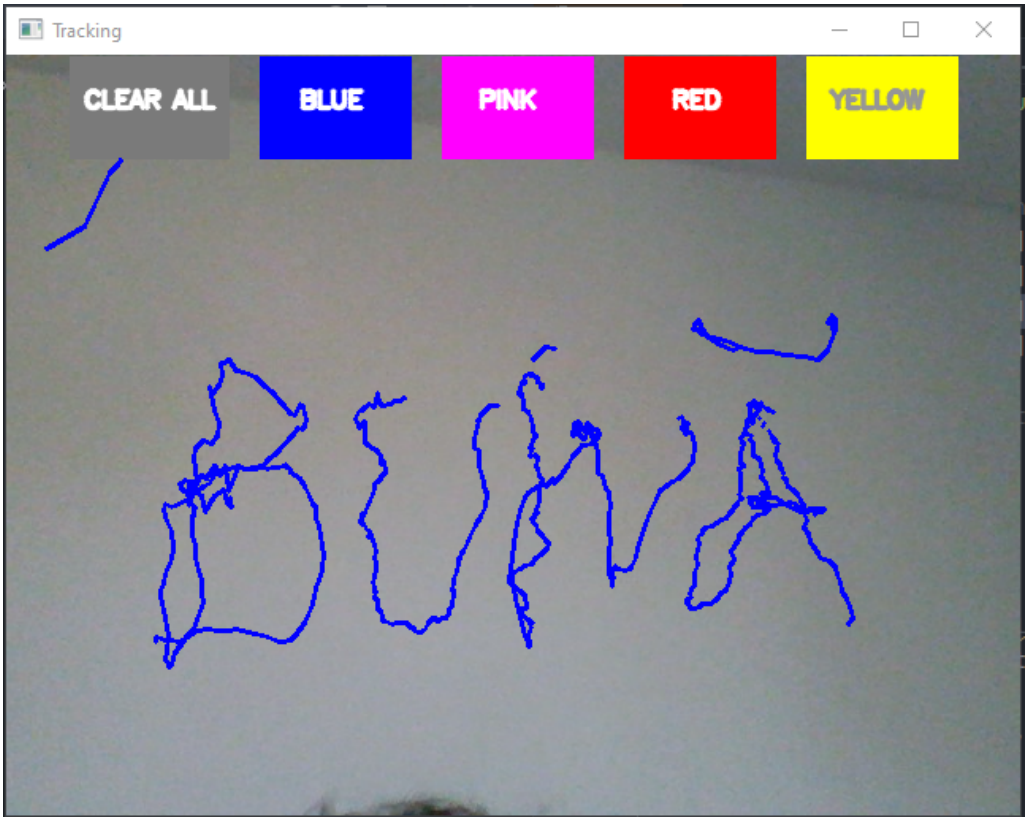

Figura 4.1: Web Cam

Canvas-ul principal contine un buton de stergere si alte 4 butoane pentru diferite culori. Pentru adaugarea butoanelor in canvas-ul principal ne-am folosit de functiile oferite de opencv, determinand forma si pozitia acestora.

```
# Adding the colour buttons to the live frame for colour access
frame = cv2.rectangle(frame, (40,1), (140,65), (122,122,122), -1)
frame = cv2.rectangle(frame, (160,1), (255,65), colors[0], -1)
frame = cv2.rectangle(frame, (275,1), (370,65), colors[1], -1)
frame = cv2.rectangle(frame, (390,1), (485,65), colors[2], -1)
frame = cv2.rectangle(frame, (505,1), (600,65), colors[3], -1)
cv2.putText(frame, "CLEAR ALL", (49, 33), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 255, 255), 2, cv2.LINE_AA)
cv2.putText(frame, "BLUE", (185, 33), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 255, 255), 2, cv2.LINE_AA)
cv2.putText(frame, "PINK", (298, 33), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 255, 255), 2, cv2.LINE_AA)
cv2.putText(frame, "RED", (420, 33), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 255, 255), 2, cv2.LINE_AA)
cv2.putText(frame, "YELLOW", (520, 33), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (150,150,150), 2, cv2.LINE_AA)
```

Figura 4.2: Butoane

White Canvas afiseaza desenele din canvas-ul principal.

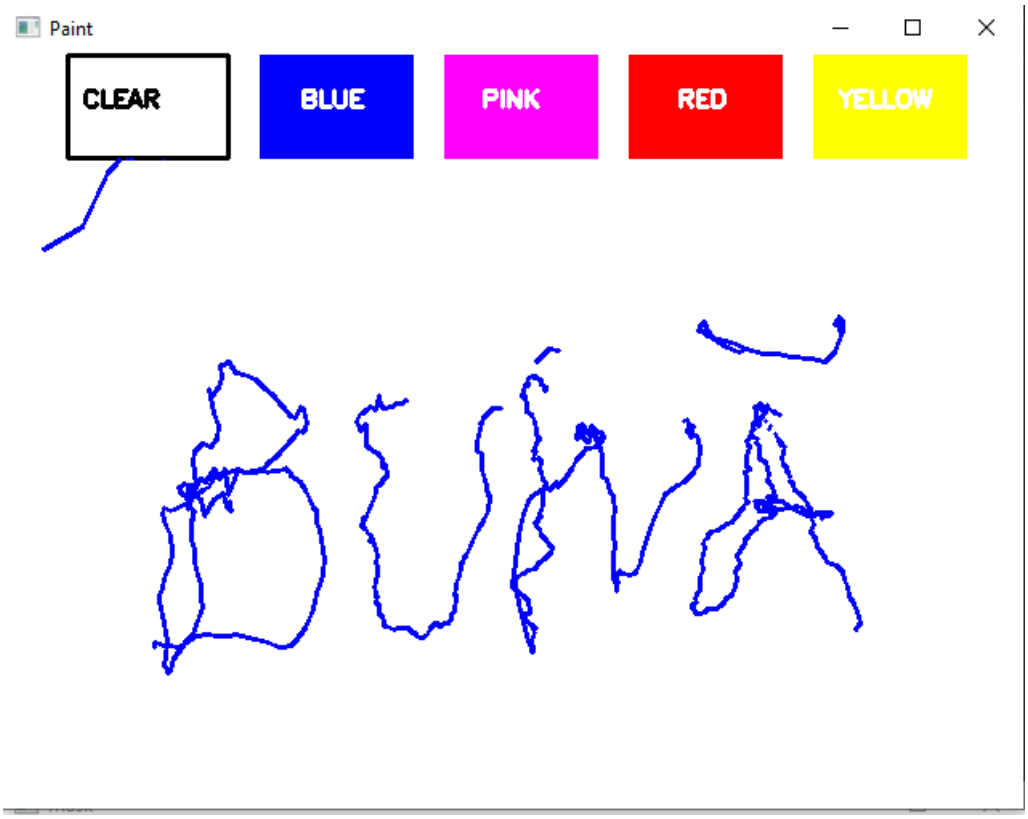

Figura 4.3: White Canvas

La fel ca in canvas-ul principal, si acesta contine un buton de stergere si alte 4 butoane pentru diferite culori. Pentru adaugarea butoanelor in canvas-ul principal ne-am folosit de functiile oferite de opencv, determinand forma si pozitia acestora.

```
# Here is code for Canvas setup
paintWindow = np.zeros((471,636,3)) + 255
paintWindow = cv2.rectangle(paintWindow, (40,1), (140,65), (0,0,0), 2)
paintWindow = cv2.rectangle(paintWindow, (160,1), (255,65), colors[0], -1)
paintWindow = cv2.rectangle(paintWindow, (275,1), (370,65), colors[1], -1)
paintWindow = cv2.rectangle(paintWindow, (390,1), (485,65), colors[2], -1)
paintWindow = cv2.rectangle(paintWindow, (505,1), (600,65), colors[3], -1)

cv2.putText(paintWindow, "CLEAR", (49, 33), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 0), 2, cv2.LINE_AA)
cv2.putText(paintWindow, "BLUE", (185, 33), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 255, 255), 2, cv2.LINE_AA)
cv2.putText(paintWindow, "PINK", (298, 33), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 255, 255), 2, cv2.LINE_AA)
cv2.putText(paintWindow, "RED", (420, 33), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 255, 255), 2, cv2.LINE_AA)
cv2.putText(paintWindow, "YELLOW", (520, 33), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (150,150,150), 2, cv2.LINE_AA)
cv2.namedWindow('Paint', cv2.WINDOW_AUTOSIZE)
```

Figura 4.4: Butoane

Black Canvas detecteaza obiectul si miscarea acestuia.

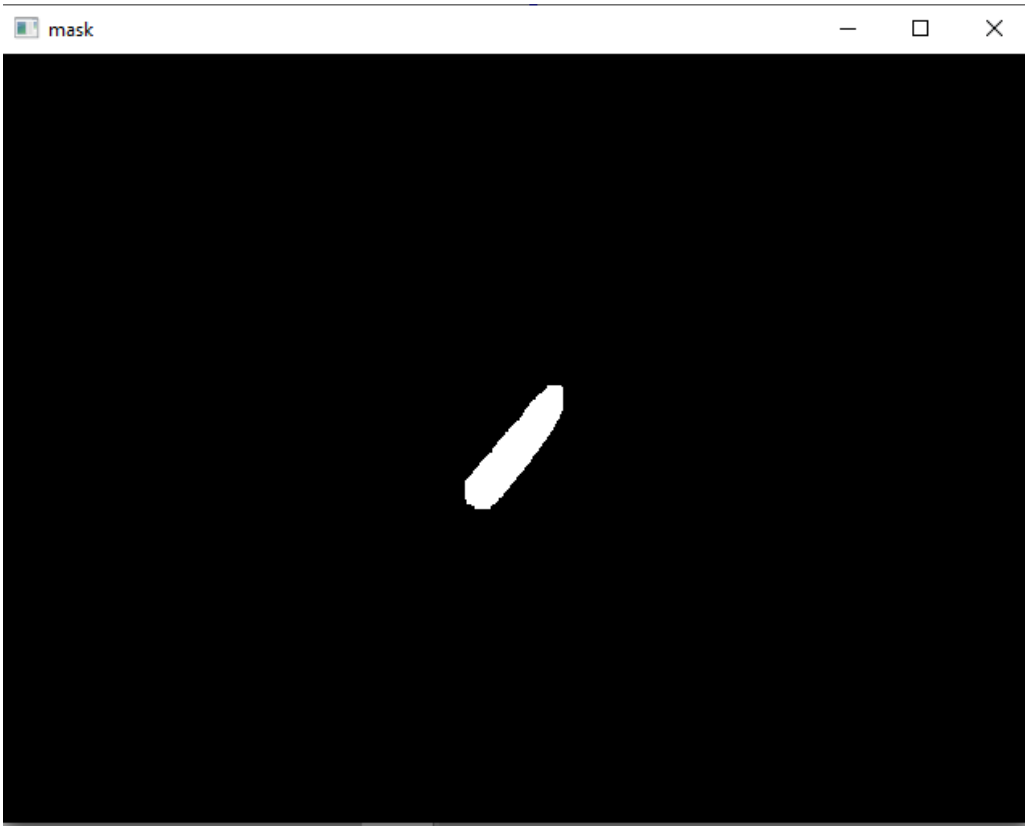

Figura 4.5: Black Canvas

Pentru identificarea conturului obiectului folosit pentru desenare ne folosim de functiile oferite de opencv pentru transformarea morfologica.

```
# Identifying the pointer by making its mask
Mask = cv2.inRange(hsv, Lower_hsv, Upper_hsv)
Mask = cv2.erode(Mask, kernel, iterations=1)
Mask = cv2.morphologyEx(Mask, cv2.MORPH_OPEN, kernel)
Mask = cv2.dilate(Mask, kernel, iterations=1)

# Find contours for the pointer after idetifying it
cnts,_ = cv2.findContours(Mask.copy(), cv2.RETR_EXTERNAL,
    cv2.CHAIN_APPROX_SIMPLE)
center = None
```

Figura 4.6: Black Canvas

Capitolul 5

Referinte Bibliografice

1. https://www.irjet.net/archives/V8/i8/IRJET-V8I8258.pdf