*Facultate: Facultatea de Automatica și Calculatoare*

*Materie: Procesarea imaginilor*

Drawer

*Proiect realizat de: Bonțea Carmen-Diana*

1. Descrierea temei

Proiectul are ca scop implementarea unui paint foarte minimalist și simplu. Folosindu-ne de camera web vrem să afișăm o fereastra pe care să desenăm. Algoritmul detectează marker-ele pe care le vom arâta în fața camerei și va desena folosind culoarea marker-ului acele puncte (vor fi mai multe deorece noi vrem să desenăm traseul rezultat în urma mișcării marker-ului în fața ferestrei) care se vor afla în mijlocul liniei care definește muchia superioară a conturului.

1. Implementare.

Pentru realizarea proiectului am avut nevoie de 2 algoritmi, unul de detecție a culorii și unul pentru desenare, fiecare find implementat individual în fișiere diferite.

*Color detector*

O imagine este carecterizată de 3 proprietăți : nuanța (hue)- desemnează numele culorii (exemplu: roșu, portocaliu sau albastru) , fiecărei culori i se asociază o locație și un număr pe diagrama circulară a culorilor standard; saturația (saturation) – desemnează puritatea sau intensitatea culorii (de exemplu, dacă se amestecă roșu cu gri, se va obține tot roșu, dar mai puțin intens) și valoarea (value) culorii- se referă la luminozitatea sau întunericul acesteia. Aceste caracteristici ne vor ajuta în detectarea culorilor obiectelor.

În ceea ce privește implementarea algoritmului, primul pas făcut a fost să-mi definesc lățimea și lungimea ferestrei cu care o să afișez imaginea în timp real. Pentru aceasta a fost nevoie să lucrez cu camera web , lucru permis de funcția VideoCapture din Opencv. În continuare m-am folosit de funcția namedWindow() pentru a crea o fereastră cu un nume și o dimensiune potrivita pentru a afișa imagini (una singură mai exact) pe ecran. Imaginea este afișată în mod implicit în dimensiunea inițială, așa că a fost nevoie să o redimensionez pentru ca aceasta să nu îmi acopere întreg ecranul. OpenCV citește imaginea în spațiu de culoare BGR, dar pentru a detecta orice culoare, mai întâi, trebuie să convertim cadrul în spațiul de culoare HSV folosind funcția cv2.cvtColor. HSV este un acronim pentru hue-culoarea , saturation - puritatea sau intensitatea culorii , value - luminozitatea culorii, acestea fiind de fapt proprietățile pe care le-am discutat mai sus. Totuși, întrebarea este cum putem afla aceste valori pentru o culoare pe care am vrea să o detectam? Folosind trackbars. Trackbar este un element GUI care permite utilizatorului să selecteze o anumită valoare într-un interval de valori, utlizând un cursor. Acesta limitează utilizatorul să selecteze o anumită valoare cu limitele minime și maxime definite. Deci pentru fiecare caratersitică a imaginii (Hue, Saturation, Value) definim câte un trackbar cu limitele corespunzătoare și în funcție de poziția cursorului putem să identificăm valorile necesare pentru a surprinde doar culoarea dorită. Pentru a crea un trackbar în OpenCV, biblioteca OpenCV oferă funcția cv2.createTrackbar(), iar pentru a citi poziția curentă a pointer-ului trackbar-ului am utilizat cv2.getTrackbarPos(). Într-un loop infinit citesc imaginea și o convertesc din RGB in HSV, citesc valorile trackbar-urilor și le rețin ăn 2 vectori(unul pentru valorile minime și altul pentru cele maxime). Mai departe creăm o masca, care este o imagine binară ce conține valori doar de 255 și 0, unde pixelii albi (255) vor reprezenta acei pixeli din imaginea inițiala care sunt în intervalul definit de cei 2 vectori, iar restul pixelii negri. Astfel ne-am indentificat culoarea dorită, fiind reprezentată de pixelii albi din mască. Cu toate acestea nu este suficient. Noi vrem să avem culoare, nu pixeli albi. Prin urmare aplicăm masca pe imaginea inițială și vom observa că toată imaginea este neagră cu excepția pixelilor care sunt în intervalul definit de noi.

*Drawer*

La fel ca în cazul algorimului de detectare a culorii primul pas făcut a fost să-mi definesc lățimea și înălțimea ferestrei de desenare, după care m-am folosit de funcția VideoCapture care returnează o imagine care va fi de fapt ferestra de desenare. Setez înălțimea, lățimea și luminozitatea ferestrei și definesc culorile pe care vreau să le identific (acestea fiind în spectrul de colori HSV), respectiv culorile(spectrul de culori BGR) cu care vreau să desenez. Pentru realizarea drawer-lui am avut nevoie de 3 funcții: find\_color, getContours, draw\_on\_canvas.

* GetContours:

Această funcție primeste ca parametru imaginea rezultată în urma aplicării măștii pe imaginea inițială și va returna punctul pe care îl vom desena mai târziu. Inițializez cu 0 coordonatele punctului și lățimea și înălțimea dreptunghiului rezultat în urma aplicării funcției boundingRect(). Ne interesează să aflăm conturul determinat de culoarea detectată de mască (cu alte cuvinte marker-ul). Pentru aceasta folosim funcția findCountours din OpenCv care are 2 parametrii importanți: mode- este modul de găsire a contururilor și method: - metoda de aproximare pentru detecție. Pentru acești doi parametrii am folosit RETR\_EXTERNAL- retrunează numai contururile exterioare , respectiv CHAIN\_APPROX\_NONE- reține toate puctele conturului. Funcția findCountours returnează un vector cu toate conturile găsite. S-ar putea ca pixelii mai multor obiecte aflate în imagine să facă parte din intervalul măștii, dar noi dorim să păstrăm doar conturul marker-ului. Pentru fiecare contur din contours (vectorul de contururi) calculez aria acestuia și folosindu-mă de un prag filtrez conturile astfel încât să aplic următoarele operații doar pe conturul marker-ului. Calculez perimetrul conturului, îl aproximez apoi aplic funcția boundingRect pentru a desena un dreptunghi în jurul acestuia. BoundingReact retunează coordonatele punctului de început al dreptunghiului, respectiv înălțimea și lățimea acestuia. Calculez punctul pe care vreau să-l retunez.

* FindColor

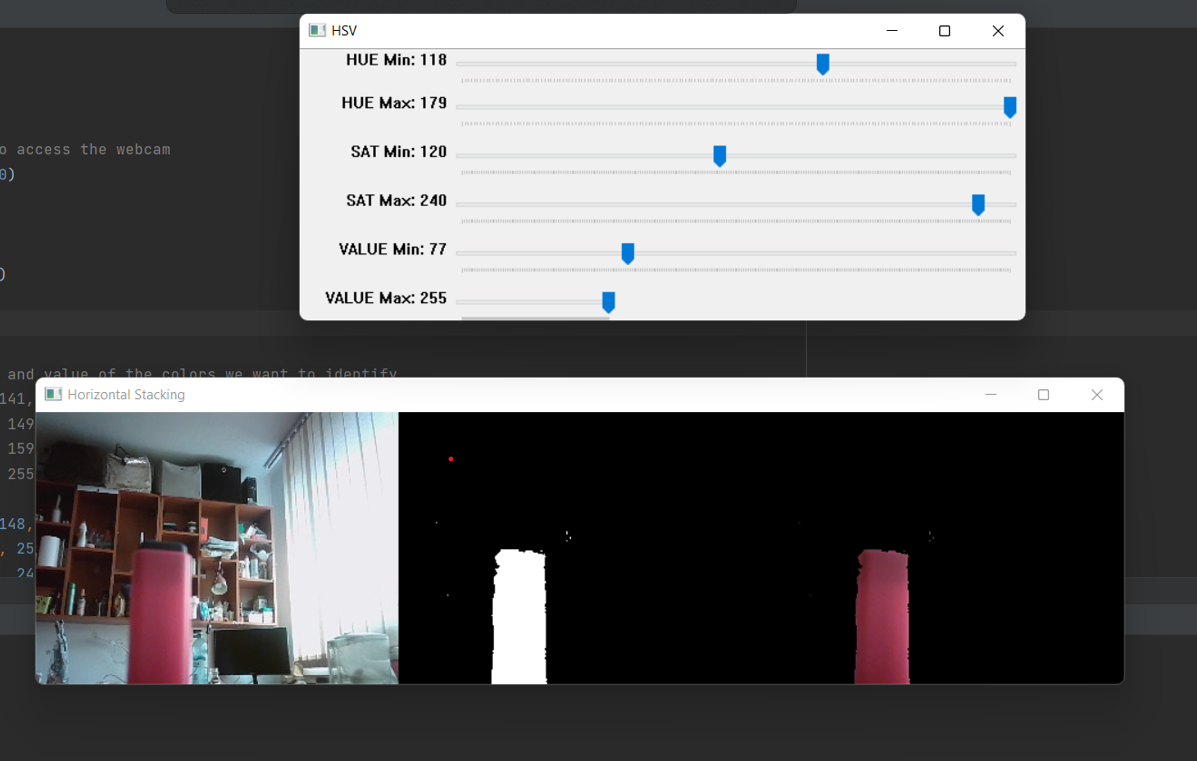
Funcția primește ca parametrii imaginea sursă, culorile detectate și culorile cu care vom desena și returnează un vector de puncte. Convertesc imaginea din spațiul de culori BGR în HSV. Pentru fiecare culoare din vectorul de culori definit rețin în vectorul lower valorile minime ale proprietăților : nuanța, saturație și valoare , respectiv în vectorul upper valorile maxime. Calculez masca și o aplic pe imagine, după care apelez funcția getCountours. Verific dacă coordonatele punctului returnat sunt diferite de 0, dacă da înseamnă că în imagine apare culoarea curentă din parcurgerea vectorului de culori și îl introduc în vectorul de puncte, altfel nu fac nimic.

* Draw\_on\_canvas

Primește ca parametrii un vector de puncte, și vectorul de culori cu care desenăm și va desena în imagine punctele găsite.

1. Rezultate

*Color detector*



*Drawer*

O imagine care conține text, interior, ecran, captură de ecran

Descriere generată automat