SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Diplomski studij

PREPOZNAVANJE RUKOM POKAZANIH GESTI

Projektni zadatak

Obrada slike i računalni vid

Matej Šarčević

Osijek, 2021.

SADRŽAJ

1. UVOD

Tema ovog projektnog zadatka bila je implementacija metode za prepoznavanje rukom pokazanih gesti. Geste koje će se promatrati u sklopu ovog projektnog zadatka bit će rukom pokazani brojevi, od broja 0 do broja 5. Neke od tehnologija koje su korištene prilikom izrade programskog rješenja su programski jezik Python, OpenCV library, te MediaPipe platforma koje će ukratko biti opisane u nastavku seminarskog rada. Metoda za prepoznavanje gesta na slici implementirana je pomoću klasičnih metoda obrade slike i računalnog vida, ali se rješenje moglo dobiti i korištenjem dubokih neuronskih mreža, gdje bi se model istrenirao na setu slika rukom pokazanih brojeva i na osnovu toga radio predikciju pokazane geste. Prilikom pokretanja programa korisniku se nudi mogućnost odabira hoće li prepoznavanje izvršiti na već postojećoj slici ili će se koristiti kamera, te će se u tom slučaju prepoznavanje odvijati kada se kamera uključi.

2. PREGLED PODRUČJA I PROBLEMATIKE

Tijek rješavanja problema:

1. Učitavanje slike
2. Pretvorba iz RGB u HSV model boja
3. Segmentacija slike
4. Traženje kontura
5. Konveksni trup
6. Konveksni defekti
7. Primjena kosinusovog poučka

2.1. Općenito o području i korištenim tehnologijama

Računalni vid je područje umjetne inteligencije koje se bavi prepoznavanjem dvodimenzionalnih te trodimenzionalnih predmeta. Bez računalnog vida robot se ne može snalaziti u prostoru. Računalni vid daje računalu mogućnost da „razumije“ što se nalazi na fotografiji ili videu, on nam omogućuje razvoj novih aplikacija koje koristimo svakodnevno. Primarna upotreba odnosno svrha računalnog vida jest zaštita ljudi, podataka, vrijednih stvari te smanjenje opasnosti od terorističkih napada i sl. Cilj računalnog vida je prepoznavanje te praćenje objekata, rekonstrukcija slike itd. Računalni vid primjenjiv je u mnogim disciplinama kao primjerice u proizvodnoj industriji za pozicioniranje robotskih ruku, uočavanje neispravnosti u proizvodnji.

Digitalna obrada slika obuhvaća transformaciju slike u digitalni format i njezino procesuiranje, odnosno njezinu obradu digitalnim računalima. I input i output digitalne slike jesu digitalne slike. U medicini se digitalna obrada slike koristi najčešće zbog poboljšanja kontrasta slika, pseudokoloriranje slika za bolju vidljivost, rekonstrukciju slike iz projekcija, ultrazvuk, mamografiju i sl. Nadalje, u području geologije obrada digitalne slike koristi se za nalazišta nafte i minerala, u meteorologiji za oblake i atmosferu, a može se također koristiti i u vojne i policijske svrhe.

Python je programski jezik kojega je stvorio Guido van Rossum 1991. godine. S razvojem Pythona Guido van Rossum počeo je 1989. godine te ga je objavio 1991. Python podržava proceduralni, objektno-orijentirani i funkcionalni stil programiranja, koji programerima daje slobodu da sami odaberu pristup koji odgovara njihovom problemu. Baš ta fleksibilnost čini Python sve popularnijim. Najviše se koristi na Linuxu, ali postoje verzije i za druge operacijske sustave.

Python karakterizira jasna i jednostavna sintaksa te je iz tog razloga on savršen za početnike u programiranju. Međutim, zbog svoje jednostavnosti i prilagodljivosti ga koriste i velike kompanije poput Googlea, Yahooa, NASA-e i sl. Programski jezik Python je jezik visoke razine što bi značilo da je bliži govornom nego strojnom jeziku. Programi koji su pisani na Python programskom jeziku su jednostavniji i čitljiviji te se stoga povećava produktivnost programera. Za konstantni razvoj jezika zadužena je neprofitna organizacija Python Software Foundation (PSF) koja se bavi normiranjem koda te prikupljanjem donacija. Python je besplatan te dostupan svima. Koristi se za doista mnogo namjena, koriste ga znanstvenici za analizu gena, u matematičkim znanostima koristi se za simulacije, za umjetnu inteligenciju i sl.

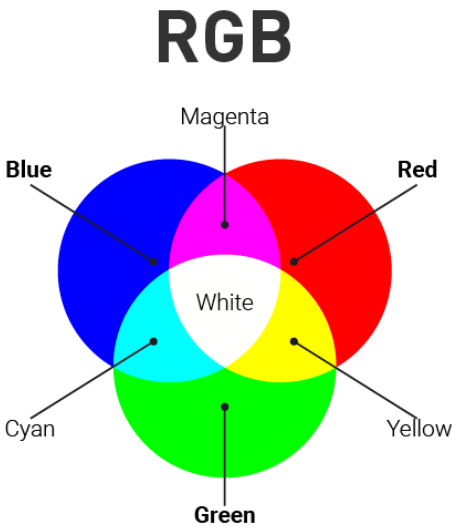
OpenCV iliti Open Source Computer Vision Library je biblioteka otvorenog koda s BSD licencom što znači da je besplatna za akademske i komercijalne svrhe. Sadrži sučelja za rad u programskim okruženjima C, C++, Python, Java i MATLAB te podržava Windows, OS X, Linux, iOS, Android i BlackBerry operacijske sustave. Dizajnirana je i napisana u C/C++ okruženju te uz pomoć OpenCL-a omogućuje rad na heterogenim sustavima. OpenCV biblioteka sadrži optimizirane implementacije 2500 algoritama namijenjenih računalnom vidu.

Projekt OpenCV razvijen je u Intelovom centru u Rusiji. Prva verzija objavljena je 2000. godine, a do 2005. izlazi pet beta verzija. Aktualna verzija izdana je 2015. OpenCV nudi inuitivan način čitanja i pisanja slika te videozapisa. Podržava mnogo formata (BMP, JPEG, PNG itd.).

2.2. Modeli boja

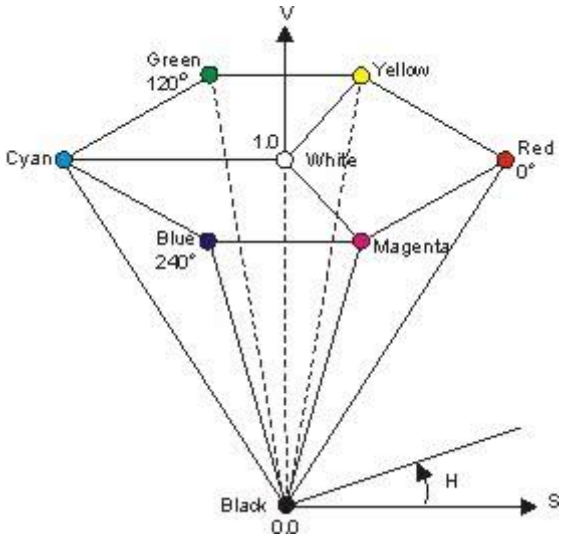
Prva stvar koju je potrebno napraviti nakon učitavanja slike je promjena modela boje koji se promatra na njoj. U ovom slučaju najprikladnije je bilo sliku iz RBG prostora boja pretvoriti u HSV prostor boja, kako bi bilo lakše izvući bitne značajke za slike, u ovom slučaju dlan, na koji se onda dalje primjenjuju ostali postupci kako bi se došlo do rezultata.

Kod RGB (Red, Green, Blue) modela se svaka boja može prikazati kao mješavina triju vrijednosti primarnih svjetla crvene, zelene i plave boje. RGB model svoju upotrebu ima u elektroničkim sustavima kao što su pametni telefoni, kamera, monitori, skeneri i slično. RGB model može prikazati većinu boja koristeći kombinaciju crvene, zelene i plave svjetlosti, ali ipak na taj način nije moguće prikazati sve boje. Svaka se boja u RGB modelu može zapisati u uređenoj trojci tipa (R,G,B) koju redom čine određene vrijednosti crvene, zelene i plave svjetlosti. Ova se uređena trojka može prikazati na više načina. Na primjer, možemo neku boju prikazati u postocima, gdje bi 0 predstavljala nedostatak te boje, a 100 potpunu zasićenost te boje. Na taj bi način (0,0,100) predstavljalo potpuno zasićenu plavu boju, (100,100,0) dalo bi nam žutu boju, a (50,50,50) sivu boju. Pošto bijelu boju čini kombinacija zasićene svjetlosti svih primarnih boja, njezina RGB vrijednost iznosi (100,100,100), dok za crnu boju vrijedi suprotno, želimo najmanju količinu svjetlosti pa će RGB vrijednost za crnu boju iznositi (0,0,0).



Slika 2.1. RGB model boja

Kod HSV (Hue, Saturation, Value) modela boje su prikazane u podskupu prostora kojeg omeđuje izvrnuta šesterostrana piramida. Boja je definirana u bazi piramide kutom od 0 do 360 stupnjeva, gdje je crvena boja ishodište kuta. Kut se mjeri od crvene boje smjerom suprotnim od kazaljke na satu. Žuta boja je na 60°, zelena na 120°, cijan 180°, plava 240° i magenta 300°. Sjajnost boje određena je vertikalnom osi V. Takva piramida u svom vrhu sadržava crnu boju te poprima sjajnost boje poprima vrijednost 0,0 a u sredini baze piramide vrijednost je 1,0 i ona odgovara bijeloj boji. Zasićenost određena je radijalnom udaljenošću od osi V.



2.2. HSV model boja

2.3. Segmentacija slike

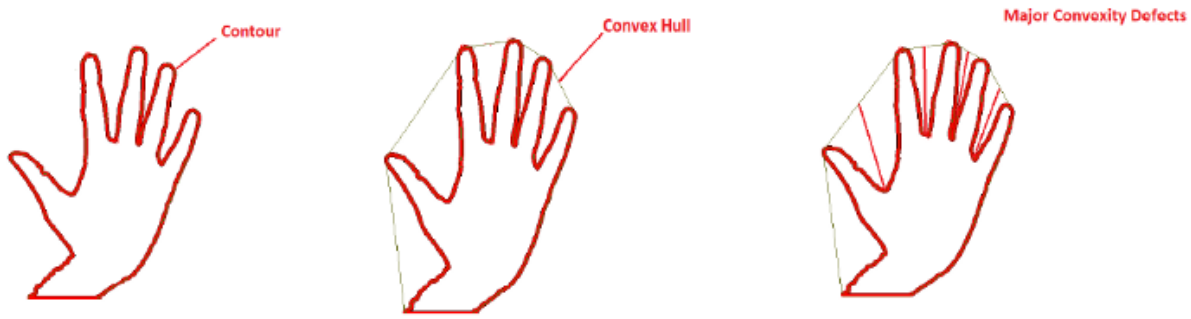
Sljedeći korak je segmentacija slike, u ovome slučaju svođenje boja na crnu ili bijelu, ali prije nego što se na primjeni threshold funkcija iz OpenCV biblioteke, na sliku je potrebno primijeniti funkciju blur iz iste biblioteke. Spomenuta blur funkcija koristi se za zaglađivanje slike i uklanjanje detalja i šuma. Thresholding je binarnizacija slike. Općenito, grayscale sliku pretvaramo u binarnu sliku, gdje su pikseli 0 ili 255. Grayscale slika je ona u kojoj je vrijednost svakog piksela jedan uzorak koji predstavlja samo količinu svjetlosti, odnosno nosi samo informacije o intenzitetu. Jednostavan primjer thresholdinga bio bi odabir vrijednosti praga T, a zatim postavljanje svih intenziteta piksela manjih od T na 0, a svih vrijednosti piksela većih od T na 255. Na taj način možemo stvoriti binarni prikaz slike.

2.4. Konture

Konture se definiraju kao linija koja spaja sve točke duž granice slike koje imaju jednak intenzitet. Konture su korisne u analizi oblika, pronalaženju veličine predmeta koji nas zanima i otkrivanju predmeta. OpenCV biblioteka sadrži funkciju findContour koja se koristi za izdvajanje kontura sa slike. Funkcija najbolje djeluje na binarnim slikama, pa je najbolje prije korištenja findContour funkcije primijeniti thresholding na slici.

Najjednostavnije rečeno, konveksni trup objekta je minimalna granica koja objekt (ili njegovu konturu) može u potpunosti zatvoriti odnosno omotati. OpenCV sadrži funkciju convexHull pomoću koje se lako može pronaći konveksni trup, a kao parametar joj se predaje slika na kojoj su pronađene konture.

Konveksnim defektom se smatra svako odstupanje konture od konveksnog trupa, isto tako, u OpenCV biblioteci postoji funkcija convexityDefects koja pronalazi sve defekte na slici. Slika 2.3. na jednostavnom primjeru prikazuje razlike između kontura, konveksnog trupa i konveksnog defekta.



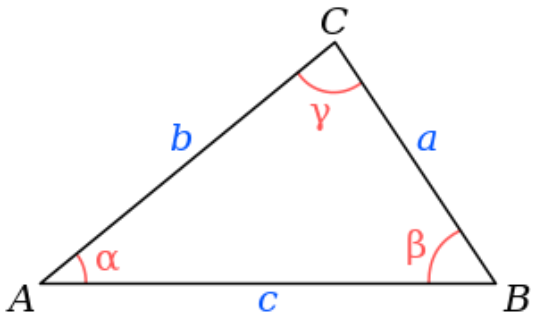
Slika 2.3. Konture, Convex Hull i Convexity Defects

2.5. Kosinusov poučak

Kosinusov poučak je matematički izraz jedne od relacija među elementima trokuta:

$$c = \sqrt{a^2 + b^2 - 2\,a\,b\cos\gamma}$$

gdje su a, b i c stranice trokuta, a γ kut među stranicama a i b. Rabi se za rješavanje računskih problema u trokutu pomoću pojedinih trigonometrijskih funkcija. Poučak glasi da je kvadrat bilo koje stranice trokuta jednak zbroju kvadrata drugih dviju stranica umanjen za dvostruki umnožak tih dviju stranica i kosinusa kuta između njih. Kosinusni se poučak rabi kada su poznate dvije stranice trokuta te kut među njima.



Slika 2.4. Prikaz trokuta sa označenim stranicama i kutovima

U sklopu ovog projekta, poučak se koristi na način da se izračuna kut gama, koji predstavlja kut između dva prsta, a računa se po forumuli:

$$\gamma = \cos^{-1}\left(\frac{a^2 + b^2 - c^2}{2\,a\,b}\right)$$

Kut gama računa se za svaki razmak na slici i ako je kut manji od 90°, tada se to područje na slici smatra kao jedan prst.

3. OBJAŠNJENJE PROGRAMSKOG RJEŠENJA

Na samom početku, potrebno je uvesti sve biblioteke koje se koriste u sklopu programa.

```
import numpy as np
import cv2 as cv
```

U sljedećoj funkciji učitana slika pretvara se iz RGB modela boja u HSV model. Osim toga u ovoj funkciji se radi i izdvajanje dlana od okoline na osnovu HSV vrijednosti boja u slici. Nakon toga, poziva se blur funkcija koja zaglađuje rubove i smanjuje šum. Zadnja stvar u funkciji je thresholding slike, te funkcija vraća takvu sliku.

```
def SkinMask(img):
    HSVIm = cv.cvtColor(img, cv.COLOR_BGR2HSV)
    lower = np.array([0, 48, 80], dtype = "uint8")
    upper = np.array([20, 255, 255], dtype = "uint8")
    skinRegionHSV = cv.inRange(HSVIm, lower, upper)
    blurred = cv.blur(skinRegionHSV, (2,2))
    ret, thresh =
cv.threshold(blurred,0,255,cv.THRESH_BINARY)
    return thresh
```

Druga funkcija služi pronalazak kontura na predanoj slici na kojoj je prethodno obavljen thresholding, pomoću funckije findContours() iz OpenCV biblioteke.

Osim kontura, u ovoj funkciji se radi pronalazak konveksnog trupa na slici. Za to također postoji funkcija iz OpenCV biblioteke, convexHull().

```
def GetContourAndHull(mask_img):
    contours, hierarchy = cv.findContours(mask_img,
cv.RETR_TREE, cv.CHAIN_APPROX_SIMPLE)
    contours = max(contours, key=lambda x: cv.contourArea(x))
    hull = cv.convexHull(contours)
    return contours, hull
```

Sljedeća funkcija služi za traženje konveksnih defekata, a oni se traže pomoću funkcije convexityDefects().

```
def GetDefects(contours):
    hull = cv.convexHull(contours, returnPoints=False)
    defects = cv.convexityDefects(contours, hull)
    return defects
```

Učitavanje slike, odnosno video sekvence ili kamere. Ukoliko se želi učitati slika ili video, potrebno je umjesto nule u funkciji VideoCapture() navesti putanju do slike, odnosno videa.

```
capture = cv.VideoCapture(0) # '0' for camera, or path to your
image/video
```

Glavni dio programskog rješenja u kojem se pozivaju gore navedene funkcije i obavlja prepoznavanje rukom pokazane geste.

```
while capture.isOpened():
    _, image = capture.read()

    try:
        mask_img = SkinMask(image)
        contours, hull = GetContourAndHull(mask_img)
        cv.drawContours(image, [contours], -1, (255,0,0), 2)
        cv.drawContours(image, [hull], -1, (0, 255, 0), 2)
        defects = GetDefects(contours)

        if defects is not None:
            count = 0
            for i in range(defects.shape[0]):
                s, e, f, d = defects[i][0]
                start = tuple(contours[s][0])
                end = tuple(contours[e][0])
                far = tuple(contours[f][0])
                a = np.sqrt((end[0] - start[0]) ** 2 +
(end[1] - start[1]) ** 2)
                b = np.sqrt((far[0] - start[0]) ** 2 +
(far[1] - start[1]) ** 2)
                c = np.sqrt((end[0] - far[0]) ** 2 + (end[1]
- far[1]) ** 2)
                angle = np.arccos((b ** 2 + c ** 2 - a ** 2)
/ (2 * b * c))
                if angle <= np.pi / 2:
                    count += 1
                    cv.circle(image, far, 4, [0, 0, 255], -1)
            if count > 0:
                count = count+1
            cv.putText(image, str(count), (30, 150),
cv.FONT_HERSHEY_SIMPLEX,3, (255, 0, 0) , 4, cv.LINE_AA)
        cv.imshow("Video", image)

    except:
        pass
    if cv.waitKey(1) & 0xFF == ord('q'):
        break
capture.release()
cv.destroyAllWindows()
```

4. ZAKLJUČAK

U sklopu ovog projektnog zadatka izrađena je metoda koja na slici ili video sekvenci radi prepoznavanje rukom pokazanih gesti. Geste koje su se promatrale su rukom pokazani brojevi. Prilikom testiranja performansi metode, pokazalo se da metoda radi dobro u slučaju jednobojne pozadine i sa dobrim osvjetljenjem. Također, pokazalo se da su puno precizniji rezultati dobiveni učitavanjem slike u odnosu na učitavanje video sekvence. Razlog tome je što je za statički sadržaj lakše odrediti konture i ostale potrebne parametre za prepoznavanje geste. Kod video sekvence može doći do nesigurnosti prilikom promjene rukom pokazanog broja, ali kada se ruka stabilizira, rezultat u većini slučajeva bude točan. Na to, i ostale probleme u radu metode može se utjecati tako da se podese parametri funkcija za thresholding, blur, traženje kontura, konveksnog trupa i konveksnih defekata.

LITERATURA

- Franić, D. (2016). *OpenCV biblioteka i primjene* (Diplomski rad). Preuzeto s https://urn.nsk.hr/urn:nbn:hr:217:310289
- Habrun, M. (2018). *Usporedba modela boja i primjena u računalnoj grafici* (Završni rad). Preuzeto s https://repozitorij.foi.unizg.hr/islandora/object/foi%3A4020/datastream/PDF/view
- Hrga, M. (2018). *RAČUNALNI VID. Zbornik radova Veleučilišta u Šibeniku,* (1-2/2018), 207-216. Preuzeto s https://hrcak.srce.hr/198597
- Jayaram, M., Fleyeh, H. (2016) *Convex Hulls in Image Processing: A Scoping Review*. American Journal of Intelligent Systems, 6(2): 48-58 Preuzeto s https://www.diva-portal.org/smash/get/diva2:931027/FULLTEXT02.pdf
- Pitas, I. (2000). *Digital image processing algorithms and applications*. Preuzeto s https://books.google.hr/books?hl=hr&lr=&id=VQs_Ly4DYDMC&oi=fnd&pg=PA1&dq=digital+image+processing&ots=jJnejIrhTt&sig=g9Bgc9AU79FDaO9IxvmudaxVeIc&redir_esc=y#v=onepage&q=digital%20image%20processing&f=false
- Radotić, J. (2013). *Korišćenje programskog jezika Python u izradi školskih administrativnih aplikacija* (Master rad). Preuzeto s http://elibrary.matf.bg.ac.rs/bitstream/handle/123456789/4924/Master%20rad%20-%20Jovana%20Radotic.pdf?sequence=1

Internetski izvori:

https://stackoverflow.com/questions/19479024/what-is-contours-in-computer-vision [25.7.2021]

https://en.wikipedia.org/wiki/Law_of_cosines [24.7.2021]

https://learnopencv.com/convex-hull-using-opencv-in-python-and-c/ [24.7.2021]

https://theailearner.com/2020/11/09/convexity-defects-opencv/ [25.7.2021]

https://docs.opencv.org/4.5.2/d8/d1c/tutorial_js_contours_more_functions.html [24.7.2021]