UNIVERZITET U TUZLI
FAKULTET ELEKTROTEHNIKE


# SEMINARSKI RAD

## OBARADA I ANALIZA MEDICINSKIH SLIKA

## Tema:
## Prepoznavanje kvadrata na slici

Ismar Osmanović / Lejla Zahirović

# Sadržaj

# 1 Ideja projekta

Segmentacija slike predstavlja jedan od osnovnih problema u obradi slike. Zadatak našeg projekta je kreirati algoritam koji će segmentirati, a zatim i prebrojati crvene i zelene kvadate na slici.

Slika 1: Početna slika

# 2 Realizacija projekta

```
1    img = imread('img/studenti.jpg');
```

`Imread` funkcija učitava sliku iz file-a koji je specificiran pomoću njegovog imena, utvrđujući format na osnovu sadržaja istog. Ako je ime file-a file sa više slika, funkcija imread učitava prvu sliku u tom file-u. Dakle, 'img/test1.jpg' je putanja do slike koja se učitava. img je varijabla u kojoj se čuva učitana slika. Nakon što je slika učitana, sadržaj slike (u formi matrice piksela) biće sačuvan u varijabli "img". Ova funkcija je dio biblioteke koju koristi MatLab.

## 2.1 Određivanje pragova poje

```
1    hsvImg = rgb2hsv(img);
```

Koristeći funkciju `rgb2hsv` slika se konvertuje iz RBG color prostora u HSV (Hue, Saturatiom, Value) color prostor. Dok RGB prostor koristi tri komponente (crvenu, zelenu i plavu) za reprezentaciju boja, HSV prostor koristi tri komponente:

- Hue (boja slika kao crvena, plava),
- Saturation (intenzitet boje),
- Value(osvijetljenost boje).

Dakle, img je ulazna slika u RGB formatu, koja je pritome učitana pomoću `imread` funkcije, a "hsvImg" je promjenjljiva u kojoj će biti sačuvana slika u HSV formatu.

Zašto je primjenjena ova konverzija, iz jednog color prostora u drugi, na ovom primjeru?

U opštem slučaju se u ovaj format koristi zbog svojih prednosti u obradi same slike. Naime, ovaj format pruža lakšu manipulaciju bojama. Naprimjer, H komponenta je nezavisna od svjetlosti i zasićenosti, pa je koristeći navedeni format lakše izolavati određene boje na slici, sto će kasnije biti primjenjeno, nego isključivo koristeći RGB format. U implementaciji koda je upravo bio zadatak pronaći piksele crvene i zelene boje na datim slikama. Međutim, u prvobvitnom prostoru, taj zadatak bi bio znatno otežan jer bi bilo potrebno razmatrati sve tri komponente, a u ovom slučaju smo posmatrali samo H komponentu za filtriranje crvenih tonova, kao i zelenih. Preostale dvije mogu biti korištene za kontrolu jačine boje i osvjetljenosti.

```
1    lowerRed1 = [0, 0.2, 0.2]; % Lower threshold for red
2    upperRed1 = [7/360, 1, 1];  % Upper threshold for red
3    lowerRed2 = [353/360, 0.2, 0.2]; % Lower threshold for red
4    upperRed2 = [1, 1, 1];      % Upper threshold for red
```

Ove linije koda definišu pragove za filtriranje crvene boje u HSV color prostoru. Konkretno, koristi se Hue (H), Saturation (S) i Value (V) komponente za postavljanje opsega boje crvene(a kasnije i zelene), tako da se mogu prepoznati pikseli koji odgovaraju toj boji u slici.

Zašto se koristi ovaj pristup? Ovaj pristup koristi dva opsega vrednosti za Hue (H), jer crvena boja u HSV prostoru nije predstavljena sa samo jednim opsegom. U HSV prostoru, crvena boja se prostire kroz početni i krajnji dio opsega Hue vrednosti:

- Prvi opseg (`lowerRed1` i `upperRed1`) pokriva crvenu boju koja je bliža početku opsega (oko 0, što je početak crvene).

- Drugi opseg (`lowerRed2` i `upperRed2`) pokriva crvenu boju koja je bliža kraju opsega (oko 1, što je kraj crvene boje).

Ovaj pristup omogućava preciznije prepoznavanje crvene boje u cijeloj paleti, jer se crvena boja širi kroz ceo opseg Hue vrednosti u HSV prostoru. Sličan pristup je korišten i za zelenu boju ali samo sa jednim opsegom pošto se zelena boja ne nalzi na krajevima HSV prostora.

## 2.2 Maska

Generalno, maska se koristi za segmentaciju ili izdvavanje određenih dijelova date slike pomoću nekih kriterijuma. Koristeći ovaj koncept željeni dijelovi slike se izdvoje, dok ostali bivaju zanemareni.

Maska je zapravo binarna slika, što dalje znači da se sastoji isključivo od 1 i 0 te ona označava koji dijelovi slike trebaju biti podvrgnuti željenoj analzi, obradi ili promjeni u opštem slučaju. Naime, vrijednost 1 označava da je odgovarajući piksel od interesa, dok se onaj sa 0 zanemaruje. Ovaj koncept je implementiran na datom kodu za pronalaženje samo crvenih i/ili zelenih piksela na slikama.

Dalje se na njima mogu primjeniti filteri ili operacije. Dakle, maska predstavlja jedan od ključnih alata u obradi i analizi slike te se koristi upravo za: izolaciju određenih dijelova slike, filtriranje, segmentaciju objekta, obrada na temelju boje ili intenziteta, praćenje objekta, kao i pobošljanje preformansi (u raddu sa velikim slikama ili skupovima podataka).

```
1   % Create masks for red and green colors
2   mask1 = (hsvImg(:,:,1) >= lowerRed1(1) & hsvImg(:,:,1) <= upperRed1(1) &
    ...
3             hsvImg(:,:,2) >= lowerRed1(2) & hsvImg(:,:,2) <= upperRed1(2) &
     ...
4             hsvImg(:,:,3) >= lowerRed1(3) & hsvImg(:,:,3) <= upperRed1(3));
5
6   mask2 = (hsvImg(:,:,1) >= lowerRed2(1) & hsvImg(:,:,1) <= upperRed2(1) &
    ...
7             hsvImg(:,:,2) >= lowerRed2(2) & hsvImg(:,:,2) <= upperRed2(2) &
     ...
8             hsvImg(:,:,3) >= lowerRed2(3) & hsvImg(:,:,3) <= upperRed2(3));
9
10  mask3 = (hsvImg(:,:,1) >= lowerGreen(1) & hsvImg(:,:,1) <= upperGreen(1)
    & ...
11            hsvImg(:,:,2) >= lowerGreen(2) & hsvImg(:,:,2) <= upperGreen(2)
     & ...
12            hsvImg(:,:,3) >= lowerGreen(3) & hsvImg(:,:,3) <= upperGreen(3)
    );
13
14  redMask = mask1 | mask2;
15  greenMask = mask3;
```

`hsvImg(:,:,1) >= lowerRed1(1) & hsvImg(:,:,1) <= upperRed1(1)` provjerava da li vrijednost Hue svakog piksela leži između minimalne (lowerRed1(1)) i maksimalne (upperRed1(1)) granice za Hue. Ovaj uslov osigurava da boja piksela bude u opsegu za crvenu boju.

`hsvImg(:,:,2) >= lowerRed1(2) & hsvImg(:,:,2) <= upperRed1(2)` provjerava Saturation, tj. zasićenost boje, da li je unutar određenog opsega.

`hsvImg(:,:,3) >= lowerRed1(3) & hsvImg(:,:,3) <= upperRed1(3)` provjerava Value, tj. svjetlost boje, da li je unutar željenog opsega.

Ovaj uslov znači da će maska "mask1" biti true (1) za piksele koji imaju crvenu boju u određenom opsegu, prema vrijednostima iz lowerRed1 i upperRed1.

`Mask2` ima isti oblik, ali koristi druge granice (`lowerRed2 i upperRed2`), što omogućava da se prepoznaju i druge nijanse crvene boje (na primjer, svjetlija crvena).

```
redMask = mask1 | mask2;
```

Ova linija koristi logički operator | (ili) kako bi spojila dvije maske. Znači da će `redMask` biti true (1) na mjestima gdje je barem jedan od uslova zadovoljen, tj. ili piksela u maski 1 ili piksela u maski 2.

Analogan pristup se koristi i za `greenMask`.

Nakon ovih linija maske imaju sljedeći izgled:

Slika 2: a) redMask b) greenMask

Vidimo da su sa početne slike izdvojena svi regioni gdje se pojavljuju naši kvadrati.

## 2.3 Morfološke operacije

```
1    redMask = imerode(redMask, strel('disk', 3));
2    redMask = imopen(redMask, strel('disk', 10));
3    redMask = imdilate(redMask, strel('disk', 10));
```

U obradi slike, matematička morfologija je korištena kao sredstvo za identificiranje i ekstrakciju značajnih deskriptora slike na osnovu osobina oblika unutar slike.

Ključno područje primjene morfologije predstavlja segmentacija slike, zajedno sa automatskim brojenjem i inspekcijom. Morfološke operacije su također osnovni alati te se kao takvi primarno koriste u binarnim slikama, ali se mogu primjeniti i na sive slike. Ove operacije direktno koriste matematičke metode za analizu i promjenu strukture objekta.

Ključni operatori su *dilatacija i erozija*.

Moguće je sve složenije procedure svesti na dvije pomenute. Osvrćući se na sam pojam binarne slike, on je obrađen u prethodnom tekstu. Međutim, važno je pomenuti da se binarna slika sastoji od bilo koje grupe povezanih piksela. Tu dolazi do razlike između dvije vrste povezanosti i to 4-povezanost i 8-povezanost. 4-povezanost se temelji na povezanosti piksela koji su u neporednoj horizontalnoj ili vertikalnoj blizini. To znači da su povezani samo oni pikseli koji se nalaze uzduž stranica kvadrata, a ne dijagonalno. Pikseli **(i, j)** su povezani ako postoji neki piksel u njegovom neposrednom okruženju, koji je u istoj boji (npr. bijeli piksel u binarnoj slici), a taj piksel se nalazi u jednoj od sledećih pozicija:

- (i-1, j) — gornji susjed
- (i+1, j) — donji susjed
- (i, j-1) — lijevi susjed
- (i, j+1) — desni susjed

Ovaj tip povezanosti primjenjuje se kada je potrebno razmatrati samo vertikalne i horizontalne susjede za povezivanje piksela.

Za 4-povezanost, N4, koristi se city-block ili Manhattan distanca. 8-povezanost se temelji na širem okruženju, gdje su povezani pikseli koji se nalaze u neposrednom susjedstvu, uključujući vertikalne, horizontalne, i dijagonalne susjede. Dakle, osim vertikalnih i horizontalnih piksela, uključeni su i dijagonalni susjedi.

Pikseli *(i, j)* su povezani ako postoji neki piksel u njegovom neposrednom okruženju, koji je u istoj boji, a taj piksel se nalazi u jednoj od sledećih pozicija:

- (i-1, j) — gornji susjed
- (i+1, j) — donji susjed
- (i, j-1) — lijevi susjed
- (i, j+1) — desni susjed
- (i-1, j-1) — gornji lijevi dijagonalni susjed
- (i-1, j+1) — gornji desni dijagonalni susjed
- (i+1, j-1) — donji lijevi dijagonalni susjed
- (i+1, j+1) — donji desni dijagonalni susjed

Primenjuje se kada je potrebno razmatrati sve susjede, uključujući dijagonalne, za povezivanje piksela. Za 8-povezanost, N8, koristi se Chebysheva ili šahovska distanca.
Dakle, korišćenje 8-povezanosti može biti korisno kada želimo da prepoznamo objekte koji su "nešto razdvojeni", ali su dijagonalno povezani, dok 4-povezanost može biti korisna kada želimo da analiziramo striktne vertikalne ili horizontalne veze između objekata.

Sljedeći bitan pojam, koji služi objašnjavanju gornjih operacije, jeste strukturni element.

**Strukturni elementi (SE)** su osnovni alati u morfološkoj obradi slike. Oni su matrice sa fiksnim oblikom koje se koriste za primjenu različitih morfoloških operacija na binarnim ili grayscale slikama. Strukturni element definiše oblik i veličinu regiona u okolini svakog piksela slike, koji se koristi za manipulaciju tim pikselima tokom morfoloških operacija. SE mogu imati

različite oblike kao što su kružni, kvadratni, pravougaoni, dok njihova veličina zavisi direktno od njihove aplikacije u samom kodu. Navedeni pojmovi su ključni pri pristupu eroziji i dilataciji.

Najprije, **erozija** se koristi za smanjenje objekata na slici. Ova operacija omogućava uklanjanje manjih struktura i detalja u slici, kao i smanjenje veličine objekata u binarnoj slici. Erozija je vrlo korisna u segmentaciji, uklanjanju šuma i prepoznavanju oblika. Erozija funkcioniše tako što, za svaki piksel u slici, gleda sve njegove susjedne piksele. Ako svi susjedni pikseli koji čine oblik strukturnog elementa imaju vrijednost 1 (odnosno pripadaju objektu), onda centralni piksel ostaje 1. Ako neki od susjednih piksela nije 1, centralni piksel se postavlja na 0, što znači da se objekat smanjuje.

S druge strane, **dilatacija** se koristi za proširivanje objekata na slici. Ova operacija omogućava povećanje veličine objekata u binarnoj slici i povezivanje objekata koji su blizu jedan drugog. Dilatacija je suprotnost eroziji, jer proširuje piksele objekta umjesto da ih smanjuje. Dilatacija koristi strukturni element (SE) koji se "širi"oko svakog piksela objekta u slici. Za svaki piksel u slici, dilatacija postavlja vrijednost 1 u centralni piksel (ako je bar jedan piksel iz strukturnog elementa 1). Ako je bar jedan od susjednih piksela 1, centralni piksel postaje 1. Dakle, objekat se "proširuje"tako da susjedni pikseli postaju dio objekta.
Dilatacija i erozija se često koriste zajedno. Na primjer, primjena dilatacije može učiniti objekat većim, dok erozija može ukloniti nepotrebne dijelove objekta. Ove operacije zajedno čine osnovu mnogih "otvoreno"i žatvoreno"morfoloških operacija.

**Otvaranje** je definisano kao morfološka operacija erozije praćena sa dilatacijom sa istim strukturnim elementom. Erozija uklanja neke piksela objekta sa ivica regiona objekta, dok dilatacija dodajepiksele objekta. Tako veličina objekta ostaje ista, ali su njegove konture zaglađenije.

**Zatvaranje** je morfološka operacija koja se sastoji od dilatacije praćene erozijom sa istim strukturnim elementom. Zatvaranje zaglađuje konture objekata, ali i popunjava uske prekide ili praznine i eliminiše male šupljine, mijenjajući te male regione pozadine u regione objekta.

Gore navedeni kod koristi nekoliko morfoloških operacija za obradu slike u kojoj je detektovana crvena boja, pri čemu je isti pristup urađen i za zelenu boju. Morfološke operacije koriste strukturne elemente za manipulaciju oblikom objekata u slici. U ovom slučaju, strukturni element je disk, što znači da se koristi diskasti oblik za primjenu operacija.

```
redMask = imerode(redMask, strel('disk', 3));
```

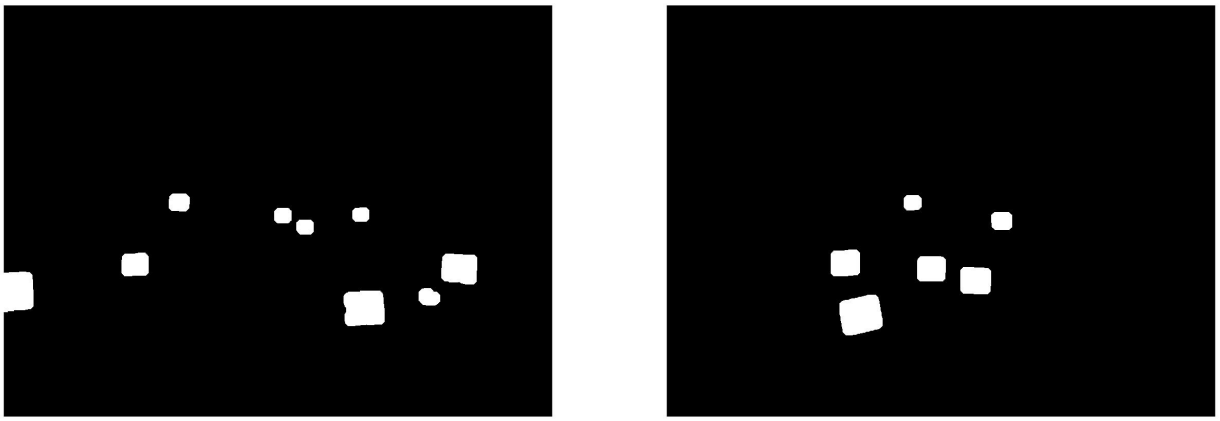
Ovdje se primenjuje erozija na `redMask` koristeći **diskasti strukturni element** sa poluprečnikom od 3 piksela. Ovaj diskasti strukturni element je matrica koja sadrži vrijednosti 1 u obliku diska (poluprečnik 3) i 0 izvan tog diska. Erozija će ukloniti piksele sa ivica objekta. Zbog ove operacije, objekti u redMask postaju manji, a sitni objekti mogu biti uklonjeni. Ova operacija se radi dva puta uzastopno.

```
redMask = imopen(redMask, strel('disk', 10));
```

U ovoj liniji je primjenjena otvorena operacija. Prvo se smanjuju objekti pomoću erozije, a zatim se proširuju pomoću dilatacije. Ovaj postupak služi za uklanjanje sitnih objekata i šupljina u većim objektima, dok istovremeno zadržava osnovnu strukturu objekata. Zbog erozije, manji objekti nestaju, dok dilatacija omogućava da preostali objekti ponovo postanu "veći"i sačuvaju osnovni oblik.

```
redMask = imdilate(redMask, strel('disk', 10));
```

Ovom linijom se proširuju preostali objekti u slici koristeći diskasti strukturni element. Ovo može povezati objekte koji su se prethodno razdvojili i može proširiti ivice objekata, čineći ih "većim". Ovaj niz operacija omogućava precizno filtriranje i obradu binarnih slika, sa ciljem da se izdvoje važni objekti (u ovom slučaju crvena boja) i da se poboljša njihova vidljivost i konture na slici.



Slika 3: redMask i greeenMask nakon morfoloških operacija

## 2.4 Boundary

```
    [Br, Lr] = bwboundaries(redMask, 'noholes');
    [Bg, Lg] = bwboundaries(greenMask, 'noholes');
```

## 2.5 Funkcija brojanje

Poziv funkcije:

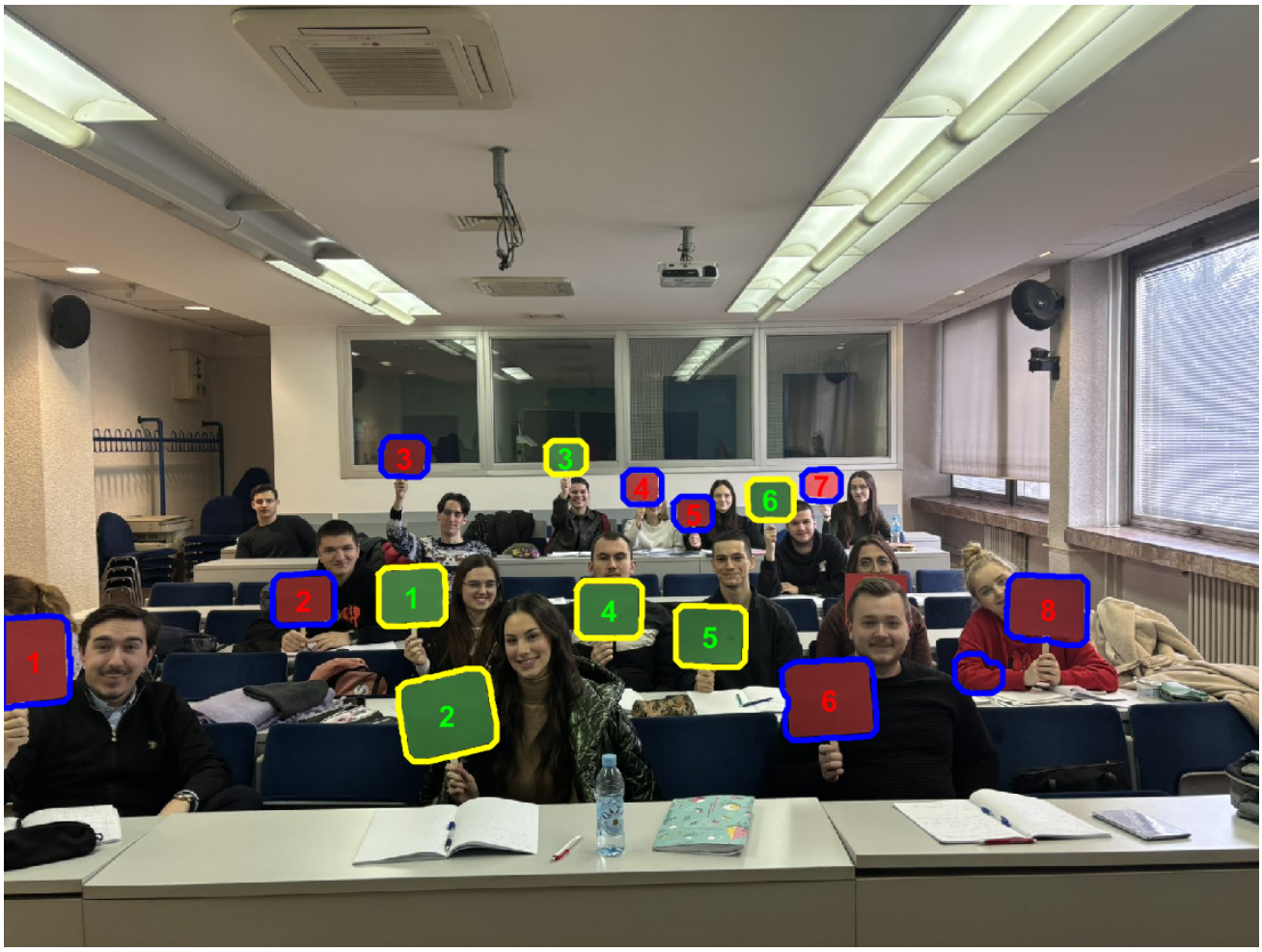
```
    RsquareCount = Funkcija_brojanje(Br,minAreaThreshold,'red',lineHeight);
    GsquareCount = Funkcija_brojanje(Bg,minAreaThreshold,'green',lineHeight);
```

Funkcija:

```
function output = Funkcija_brojanje( B, minArea, color, Yheight )
squareCount = 0;
    for k = 1:length(B)
        boundary = B{k};

        % Calculate area and perimeter
        area = polyarea(boundary(:,2), boundary(:,1));
        perimeter = sum(sqrt(sum(diff(boundary).^2, 2)));

        % Calculate the bounding box
        minX = min(boundary(:,2));
        maxX = max(boundary(:,2));
        minY = min(boundary(:,1));
        maxY = max(boundary(:,1));



        width = maxX - minX;
```

```
        height = maxY - minY;

        % Calculate aspect ratio
        tempArea = minArea;
        if nargin == 4  % Check if the 'mode' argument is missing
          if mean(boundary(:,1))< Yheight
            tempArea  = minArea / 3;
          end;
        end

        aspectRatio = height / width;  % or width / height depending on your preference
        if perimeter>0
        % Check for minimum area and aspect ratio
            if area >= tempArea && aspectRatio >= 0.5 && aspectRatio <= 2
                   roundness = (4 * pi * area) / (perimeter^2);
                if roundness < 1% Adjust this threshold as necessary
                    squareCount = squareCount + 1;
                    centroidX = mean(boundary(:,2));
                    centroidY = mean(boundary(:,1));

                    % Place the label near the boundary
                    label = sprintf('%d', squareCount);
                    text(centroidX, centroidY, label, 'Color', color, 'FontSize', 20, 'FontWeight', 'bold', 'HorizontalAlignment', 'center');
                    continue;
                end
            end
        end
    end

output = squareCount;

end
```

Rezultat:




Slika 4: 8 crvenih 6 zelenih

# 3 Zaključak:

# 4 Izvori: