VEŽBA 4 – Prostori boja

Potrebno predznanje

- Poznavanje programskog jezika C
- Računske operacije nad 2d matricama

Šta će biti naučeno tokom izrade vežbe

Tokom izrade ove vežbe upoznaćete se sa osnovama 2D signala. Naučićete na koji način su 2D slike predstavljene u računarima. Naučićete koji su to prostori boja i formati zapisa najčešći u upotreb. Nakon urađene vežbe znaćete da:

- Prikažete sliku zapisanu u nekom od standardnih formata
- Izvršite konverziju slike iz RGB prostora boja u YUV i obratno
- Vršite prostu obradu nad slikama u predstavljenim u različitim prostorima boja
- Izvršite decimaciju hromatskih koponenti u okviru YUV zapisa boja sa ciljem kompresije slike

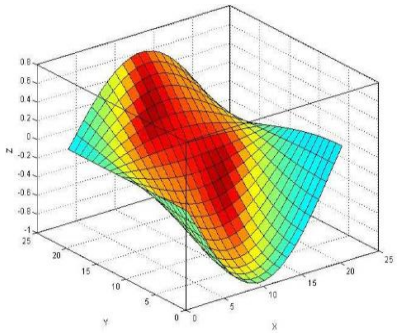
Motivacija

Uopšteno gledano da bi se bilo koji signal obradio u okviru nekog diskretnog sistema on mora biti predstavljen u diskretnom obliku. Dvodimenzionalni signali se predstavljaju kao matrice konačnih vrednosti. Poput jednodimenzionalnih i dvodimenzionalni signali se odabiraju, i svakoj tački se pridružuje jedna ili više vrednosti koje je opisuju. Kada se govori o digitalnoj obradi slika, vrednosti pridružene tačkama opisuju boju koju ta tačka sadrži. Ono šta te vrednosti konkretno predstavljaju definisano je prostorom boja u kom je predstavljena data slika. Kroz ovu vežbu upoznaćete se sa osnovama prostora boja i predstavljanja dvodimenzionalnih slika u okviru diskretnih sistema.

Teorijske osnove

2D signali

Dvodimenzioni ili 2D signal predstavlja funkciju dve nezavisne promenljive (koordinate) *S(x,y)*. Kod ovakvih signala svaka tačka u 2D koordinatnom prostoru može biti opisana sa jednom ili više vrednosti. Dvodimenzioni signali mogu biti promenljivi u vremenu. U tom slučaju svaka vrednost signala *S(x,y,t)* određena koordinatama *(x,y)* predstavlja jedan jednodimenzioni signal zavistan od vremena *S'(t)*. Slika 1 prikazuje 2D signal prikazan u 3D prostoru.



Slika 1 - 2D signal u 3D prostoru

Tipični primeri upotrebe 2D signala su:

- Fotografije (2D signal nepromenljiv u vremenu)
- Video slike (2D signal promenljiv u vremenu, 2D + T = 3D)
- Medicinski signali (MRI, CT, OCT)

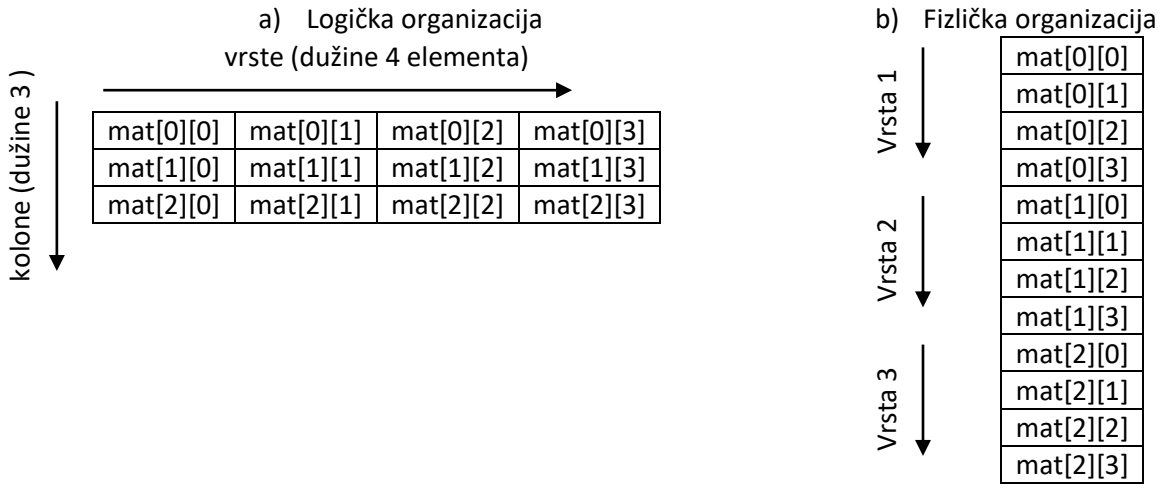
Predstava 2D signala u programskom jeziku C

Dvodimenzionalni signali se u programskom jeziku C predstavljaju kao dvodimenzionalni nizovi ili matrice. Primer definicije dvodimenzionalnog niza:

```
Int16 niz2d[32][16]
```

U programskom jeziku C, višedimenzionalni nizovi su zapravo nizovi nizova, preciznije niz čiji su elementi nizovi. U datom primeru niz2d predstavlja niz dužine 32 elementa, gde je svaki od elemenata niz od 16 celobrojnih vrednosti.

Ovako definisani niz nizova logićki predstavlja matricu brojeva. Međutim, s obzirom da memorija računara predstavlja jednodimenzioni niz lokacija, fizička realizacija date matrice svodi se na jednodimenzioni niz. Na slici 2 prikazane su logička i fizička organizacija dvodimenzione matrice definisane u programskom jeziku C sa *TIP mat[3][4]*. Sa slike se može primetiti da je fizička organizacija ovakve matrice ekvivalentna nizu definisanom sa *TIP mat[12]*.



Slika 2 – Logi;ka i fizička organizacija dvodimenzionog niza

Svaka logička vrsta matrice predstavlja podniz koji počinje na odstojanju (*j*dužina vrste)*, gde *j* predstavlja redni broj vrste. Odatle sledi da se elementu matrice, sa koordinatama *j, i* može pristupiti na jedan od dva načina:

a) mat[j][i]
*b) mat[j *dužina_vrste + i]*

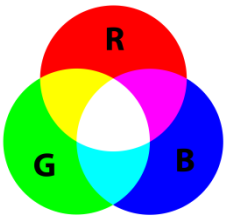
Da bi se mogao koristiti prvi način neophodno je da niz bude definisan kao matrica. Ukoliko dvodimenzioni niz predstavlja parametar funkcije, da bi se omogućio prvi način indeksiranja, neophodno je prilikom definicije funkcije navesti dužinu vrste.

```
int func(int mat[][4]);
```

Ovakav pristup uvodi ograničenje da funkcija može kao parametar primiti isključivo matricu sa odgovarajućom dužinom vrste, nije moguće proslediti matricu proizvoljne veličine. Zbog toga, preporučeno je da se za indeksiranje elemenata matrice koristi pristup naveden pod *b)*.

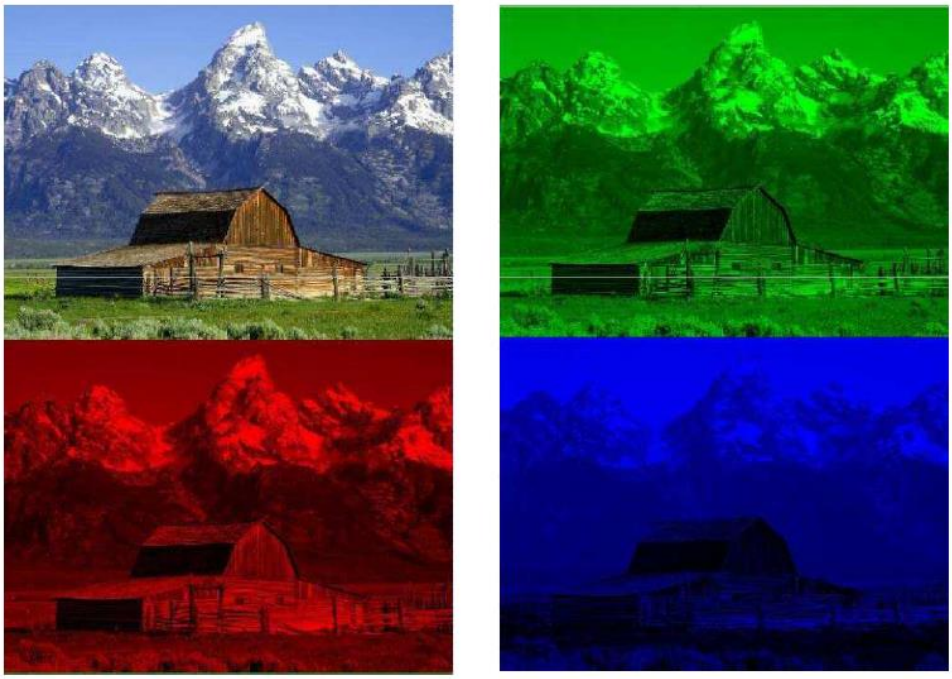
Prostori boja

2D slika predstavlja 2D signal koji se sastoji iz od tačaka. Broj tačaka po horizontali i vertikali određuju dimenzije slike. Svaka tačka se opisuje se kombinacijom vrednosti. Ove vrednosti definisane su formatom koji nazivamo prostor boja. Prostori boja definišu broj vrednosti koji se pridružuje svakoj tački, veličinu tih vrednosti u bitima i šta te vrednosti određuju. Najčešće korišćeni prostori boja u računarstvu su RGB i YUV prostori boja.



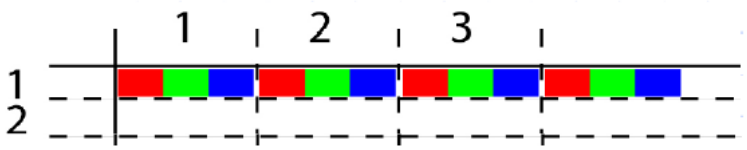
Slika 3 – Osnovne komponente RGB prostora boja

RGB predstavlja aditivan prostor boja što znači da se bilo koja boja predstavlja kao zbir osnovnih komponenti. Osnovne komponente sa kojima se sve boje mogu predstaviti jesu crvena, zelena i plava svetlost. Na slici 4 dat je primer slike i prikaza vrednosti njenih osnovnih komponenti. Sabiranjem ovih komponenti dobija se originalna slika.



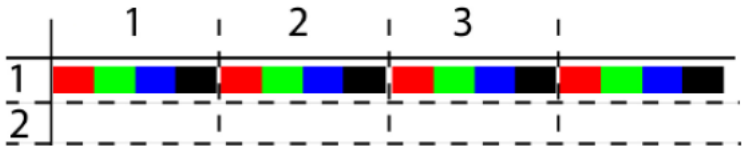
Slika 4 - Primer slike razložene na osnovne RGB kompoenente

Svaka tačka u RGB prostoru opisana je sa tri vrednosti gde svaka vrednost predstavlja intenzitet jedne od tri komponente koji se mora emitovati kako bi se proizvela odgovarajuća boja. RGB takođe opisuje sa kojim brojen bita je predstavljena svaka komponenta. Na primer format RGB888 (ili RGB24), podrazumeva da je svaka komponenta predstavljena sa 8 bita, dok RGB565 sadrži petobitne vrednosti za crvenu i plavu boju dok je zelena komponenta predstavljena sa 6 bita. Na slici 4 prikazan je zapis slike u memoriji koristeći RGB24 format.



Slika 5 - Zapis RGB24 slike u memoriji

Jednu modifikaciju RGB prostora boja predstavlja RGBA. RGBA pored 3 osnovne komponente sadrži i četvrtu alfa (*alpha*) komponentu koja diktira prozirnost slike. Ukoliko je alfa jednako nula slika je potpuno prozirna, dok maksimalna vrednost alfa odgovara potpuno neprozirnoj slici.

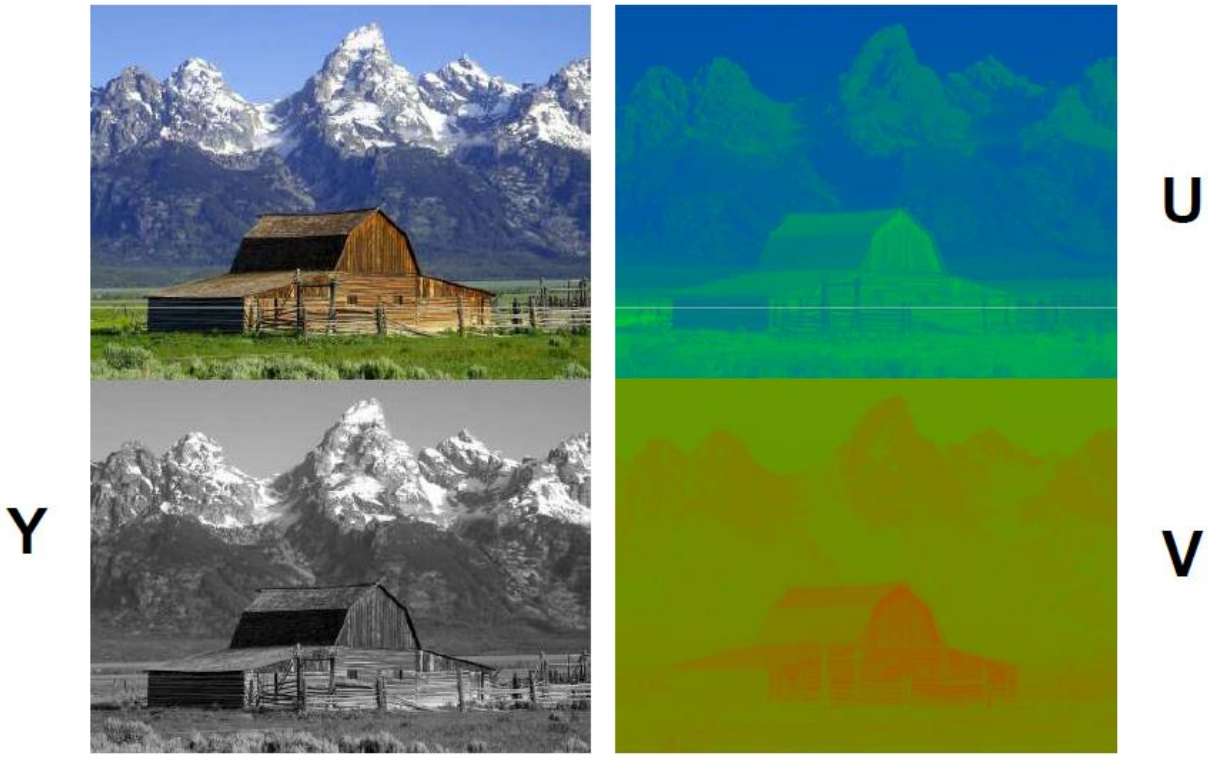


Slika 6 - Zapis slike u RGBA formatu

YUV prostor boja takođe podrazumeva predstavljanje tačaka u slici sa tri komponente.. U YUV prostoru boja svaka tačka se predstavlja sa luminantnom i dve hrominantne komponente. Luminantna komponenta (Y) odgovara opaženoj jačini osvetljaja reflektovanog svetla (osvetljaj u slici) dok hrominantne komponente (U i V) opisuju opažene komponente boja (odgovaraju količini plave, odnosno crvene boje u slici). YUV prostor je bliži ljudskoj percepciji boja. Poznato je da je ljudsko oko osetljivije na detalje i konture u slici a manje na same boje, stoga se količina informacija koja se nalazi u U i V komponentama može značajno redukovati a da se pri tome ne izgubi na kvalitetu slike. Ova činjenica je rezultovala u nastanku nekoliko različitih YUV prostora boja.

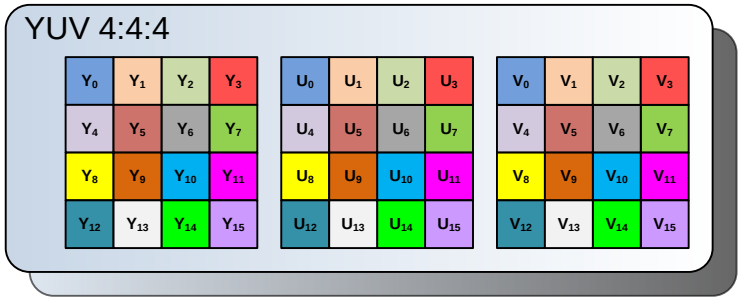
YUV prostori boja

Najpoznatiji i najčešće korišćeni YUV prostori boja su YUV 4:4:4, YUV 4:2:2 i YUV 4:2:0. Cifre koje prate YUV naziv predstavljaju šemu decimacije frekvencije odabiranja (engl. *subsampling*). Prva cifra predstavlja horizontalnu frekvenciju odabiranja luminantnih (Y) komponenti (najčešće predstavlja umnožak od 3.579 MHz u NTSC televizijskom sistemu). Sledeće dve cifre predstavljaju horizontalnu frekvenciju odabiranja hrominantnih komponenti (U pa V) relativno u odnosu na prvu cifru. U slučaju da je zadnja cifra 0, to znači da su hrominantne komponente još i vertikalno odabirane sa učestanošću 1:2 u odnosu na Y. Sledeće slike prikazuju primere ovakvih YUV formata.



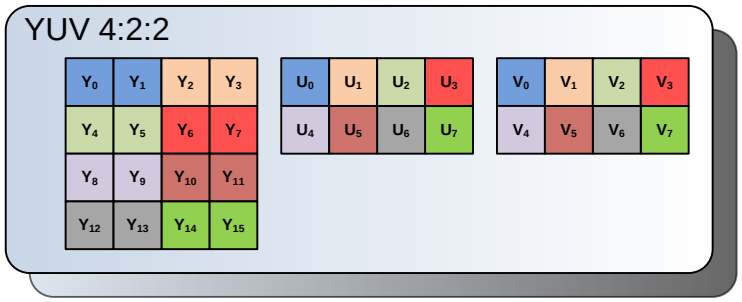
Slika 7 - Primer slike razložene na osnovne Y, U iV kompoenente

YUV 4:4:4 – Svaki Y odbirak, poseduje svoj odgovarajući U i V odbirak (Slika 7). Ovaj YUV prostor poseduje istu količinu informacija kao i RGB prostor boja. Sve komponente su odabirane sa istom frekvencijom odabiranja.



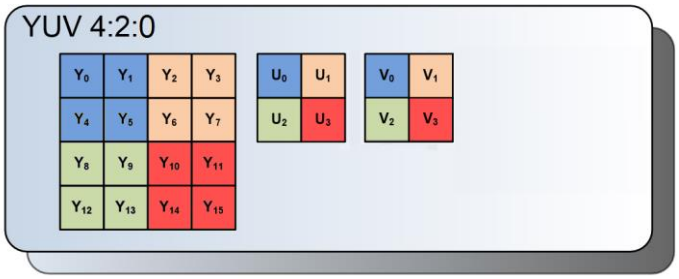
Slika 8 - Količina informacija potrebnih u YUV 4:4:4 prostoru boja

YUV 4:2:2 – Dva susedna Y odbirka, poseduju jedan odgovarajući U i V par odbiraka (Slika 8). Ovaj YUV prostor poseduje trećinu manje informacija u odnosu na RGB prostor boja. U i V koponente su odabirane sa dvostruko manjom horizontalnom frekvencijom odabiranja.



Slika 9 - Količina informacija potrebnih u YUV 4:2:2 prostoru boja

YUV420 – Blok od 2x2 susedna Y odbirka, poseduju jedan odgovarajući U i V par odbiraka (Slika 9). Ovaj YUV prostor poseduje dva puta manje informacija u odnosu na RGB prostor boja. U i V koponente su odabirane sa dvostruko manjom horizontalnom i vertikalnom frekvencijom odabiranja.



Slika 10 - Količina informacija potrebnih u YUV 4:2:0 prostoru boja

Postoje dve vrste zapisa vrednosti tačaka YUV slike: planarno i isprepleteno. Planarni zapis podrazumeva da je jedan okvir slike zapisan tako što prvo idu sve Y vrednosti zatim U pa V. Isprepleteni zapis podrazumeva da vrednosti komponenti budu isprepletene. Za YUV444 redosled je Y_0, U_0 ,V_0, Y_1, U_1, V_1… Kod YUV422 redosled je Y_0, U_0, Y_1, V_0, Y_2, V_1…

Y0	Y1	Y2	…	U0	U1	…	V0	V1	…

Y0	U0	Y1	V0	Y2	U1	Y3	V1	Y4	…

Slika 11 - Planarni i isprepleteni zapis YUV422 formata

Konverzija između RGB i YUV prostora boja

Konvezija iz RGB prostora boja u YUV data je sledećom jednačinom:

$$\begin{bmatrix} Y' \\ U \\ V \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.14713 & -0.28886 & 0.436 \\ 0.615 & -0.51499 & -0.10001 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

Konverzija iz YUV u RGB vrši se množenjem sa inverznom matricom u odnosu na matricu iz prethodne jednačine:

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1.13983 \\ 1 & -0.39465 & -0.58060 \\ 1 & 2.03211 & 0 \end{bmatrix} \begin{bmatrix} Y' \\ U \\ V \end{bmatrix}$$

U nastavku je dat primer realizacije koverzije slike iz RGB u YUV444 prostor boja i iz YUV444 u RGB. Prilikom konverzije iz YUV u RGB prostor boja neophodno je voditi računa o mogućem prekoračenju. Parametri u obe funkcije su:

- `rgbImg[]` - slika u RGB formatu
- `x` – širina slike
- `y` – visina slike
- `Y_buff[]` – matrica koja sadrži Y komponentu slike u YUV formatu
- `U_buff[]` – matrica koja sadrži U komponentu slike u YUV formatu
- `V_buff[]` – matrica koja sadrži V komponentu slike u YUV formatu

```
void RGBtoYUV444(const uchar rgbImg[], int x, int y, uchar Y_buff[], char
U_buff[], char V_buff[])
{
    uchar R, G, B;
    for(int i = 0; i<x; i++)
    {
        for(int j = 0; j<y; j+=1)
        {
            R = rgbImg[j*3*x+i*3];
            G = rgbImg[j*3*x+i*3 + 1];
            B = rgbImg[j*3*x+i*3 + 2];

            Y_buff[j*x+i] =  0.299*R + 0.587*G + 0.114*B;
            U_buff[j*x+i] =  - 0.14713*R - 0.28886*G + 0.436*B;
            V_buff[j*x+i] =  R*0.615 - 0.51499*G - 0.10001*B;
        }
    }
}

void YUV444toRGB(const uchar Y_buff[], const char U_buff[], const char V_buff[],
int x, int y, uchar rgbImg[])
{
    double R,G,B;
    double Y, U, V;
    for(int i = 0; i<x; i++)
    {
        for(int j = 0; j<y; j+=1)
        {
            Y = Y_buff[j*x+i];
            U = U_buff[j*x+i];
            V = V_buff[j*x+i];

            R = Y + 1.13983*V;
            G = Y - 0.39465*U - 0.58060*V;
            B = Y + 2.03211*U;

            if (R < 0)
                R = 0;
```

```
            else if (R > 255)
                R = 255;
            if (G< 0)
                G = 0;
            else if (G> 255)
                G = 255;
            if (B < 0)
                B = 0;
            else if (B > 255)
                B = 255;

            rgbImg[j*3*x+i*3] =  R;
            rgbImg[j*3*x+i*3 + 1] = G;
            rgbImg[j*3*x+i*3 + 2] =  B;
        }
    }
}
```

Zadaci

Zadatak 1

Implementirati funkciju:

- `void RGBtoYUV444(const uchar rgbImg[], int x, int y, uchar Y_buff[], char U_buff[], char V_buff[])`

koja vrši konverziju iz RGB prostora boja u YUV444. Parametri funkcije su:

- `rgbImg` – ulazna slika u RGB formatu
- `x` – hotizontalna dimenzija slike u pikselima
- `y` – vertikalna dimenzija slike u pikselima
- `Y_buff`, `U_buff` i `V_buff` – Izlazni nizovi u koje je potrebno smestiti Y; U i V komponentu svakog piksela. Svi nizovi su dimenzije `x*y`

Napomena: slike su dvodimenzionalni nizovi, ali su u C jeziku predstavljeni kao jednodimenzionalni. Matrica predstavljena kao jednodimenzionalni niz se može indeksirati na sledeći način:

buffer[j][i] ⇔ buffer[j*X_SIZE + i]

Implementirati funkciju:

- `void procesing_YUV444(uchar Y_buff[], char U_buff[], char V_buff[], int x, int y, double Y, double U, double V)`

u okviru koje se vrši skalarno množenje svake vrednosti matrice Y_buff sa Y konstantom, U_buff sa U konstantom i V_buff sa V konstantom. Y, U i V su povezne na grafičke kontrole koje se nalaze ispod izlazne slike.

Pokrenuti program i koristeći grafičke kontrole prikazati jednu po jednu YUV komponentu slike (smanjivanjem druge dve).

Zadatak 2

Implementirati funkciju:

- `void RGBtoYUV422(const uchar rgbImg[], int x, int y, uchar Y_buff[], char U_buff[], char V_buff[])`

koja vrši konverziju iz RGB prostora boja u YUV422. Parametri funkcije su isti kao u prethodnom zadatku, jedina razlika je što se na izlazu dobijaju U i V komponente horizontalne dimenzije y/2.

U i V komponentu računati kao aritmetičku sredinu U i V vrednosti za dve susedna piksela po horizontalnoj osi, s obzirom da jedna U i V komponenta oslikava dva susedna piksela.

Implementirati funkciju:

- `void procesing_YUV422(uchar Y_buff[], char U_buff[], char V_buff[], int x, int y, double Y, double U, double V)`

Funkcija vrši istu obradu kao i prethodnom zadatku, promena se odnosi na dimenzije U i V memorijskog niza.

Zadatak 3

Implementirati funkciju:

- `void RGBtoYUV420(const uchar rgbImg[], int x, int y, uchar Y_buff[], char U_buff[], char V_buff[])`

koja vrši konverziju iz RGB prostora boja u YUV420. Parametri funkcije su isti kao u prethodnom zadatku, jedina razlika je što se na izlazu dobijaju U i V komponente dimenzija i/2 * y/2.

Za svaki blok od po 2x2 piksela računati jednu U i V komponentu kao aritmetičku sredinu odgovarajućeg bloka vrednosti U i V.

Implementirati funkciju:

- `void procesing_YUV420(uchar Y_buff[], char U_buff[], char V_buff[], int x, int y, double Y, double U, double V)`

Funkcija vrši istu obradu kao i prethodnom zadatku, promena se odnosi na dimenzije U i V memorijskog niza.

Zadatak 4

Implementirati funkciju:

- `void decimate_Y(uchar Y_buff[], int x, int y)`

koja vrši decimaciju Y komponente na isti način na koji je to vršeno kod `RGBtoYUV420`. Za svaki blok od 2x2 piksela vrednost prvog dodeliti ostalima. Pozvati ovu funkciju nad Y komponentom slike umesto `procesing_YUV444`.

Pokrenuti program i uporediti izlaznu sliku kada je izvršena decimacija nad Y komponentom sa slikom kada su decimirane U i V komponente.