

## VEŽBA 8

### Pronalaženje obeležja slike

#### 1.1 Zadatak 1

Realizovati funkciju

- `void SIFTDetect(uchar input[], int xSize, int ySize)`

Koja vrši pronalaženje obeležja slike koristeći SIFT algoritam. Parametri funkcije su:

- `input` – ulazna slika u RGB formatu
- `xSize` – horizontalna dimenzija ulazne slike u pikselima
- `ySize` – vertikalna dimenzija ulazne slike u pikselima

Pre same obrade neophodno je izvršiti konverziju iz RGB u YUV prostor boja. Pretraga obeležja se vrši na osnovu Y komponente slike. Za računanje SIFT transformacije iskoristiti funkciju:

- `SiftKeypointList calculateSIFT(const uchar Y_buff[], int width, int height);`

Povratna vrednost funkcije je lista ključnih tačaka predstavljenih u formi strukture *SiftKeypoint*.

Nakon izračunavanja neophodno je prikazati u ulazanoj (RGB) slici sva detektovana obeležja. Za iscrtavanje obeležja iskoristiti funkciju:

- `void markSIFTKeypointsRGB(uchar imgRGB[], int width, int height, SiftKeypointList kpList, uchar R, uchar G, uchar B);`

#### 1.2 Zadatak 2

Realizovati funkciju

- `double l2Distance(SiftKeypoint kp1, SiftKeypoint kp2)`

Koja vrši računanje euklidske razdaljine između dva deskriptora koji odgovaraju tačkama `kp1` i `kp2`. Dužina deskriptora je definisana sa konstantom `DEGREE_OF_DESCRIPTOR`.

Realizovati funkciju

- `void matchFeatures(SiftKeypointList leftImageKP, SiftKeypointList rightImageKP, list<pair<QPoint, QPoint>>& matchPairs, double threshold)`

Parametri funkcije su:

- `leftImageKP` – Lista obeležja pronađenih u levoj slici

- `rightImageKP` – Lista obeležja pronađena u desnoj slici
- `matchPairs` – Lista uređenih parova tačaka koje za koje je utvrđeno da odgovaraju jedna drugoj
- `threshold` – Maksimalna euklidska udaljenost dve tačke za koje važi da odgovaraju jedna drugoj

Unutar funkcije potrebno je proći kroz listu tačaka pronađenih u jednoj slici, i za svaku tačku proći kroz listu tačaka pronađenih u drugoj slici i računati euklidsku udaljenost između deskriptora te dve tačke. Ukoliko je euklidska udaljenost manja od zadate granice potrebno je dodati u listu *matchPairs* uređen par dve tačke, gde koordinate tih tačaka odgovaraju koordinatama obeležja u prvoj oktavi (polja *r* i *c*).

Realizovati funkciju

- `void SIFTDetectPlusMatch(uchar input[], int xSize, int ySize, double threshold)`

Koja vrši računanje SIFT obeležja za ulaznu sliku (slično kao u prethodnom zadatku). Nakon završetka računanja potrebno je podeliti sve tačke u dva skupa, one koje odgovaraju levoj polovini slike (levoj slici) i one koje odgovaraju desnoj polovini slike (desnoj slici) koristeći funkciju *splitFeatures*. Nakon toga potrebno je pozvati funkciju za traženje podudarajućih obeležja (*matchFeatures*). Za svaki pronađen uređen par između dva obeležja potrebno je u ulaznoj slici (RGB) iscrtati liniju koja ih povezuje. To se može izvršiti pozivom funkcije *drawMatchedFeaturesLinesRGB*.

### 1.3 Zadatak 3

Realizovati funkciju

- `void HarrisCornerDetection(uchar input[], int xSize, int ySize, double threshold);`

Koja vrši detekciju uglova u slici upotrebom Harrisovog algoritma.

Parametri funkcije su:

- `input` – ulazna slika u RGB formatu
- `xSize` – horizontalna dimenzija ulazne slike u pikselima
- `ySize` – vertikalna dimenzija ulazne slike u pikselima
- `threshold` – granica za parametar *R* koja se koristi za utvrđivanje da li data tačka odgovara uglu

Prvi korak u algoritmu jeste konverzija iz RGB u YUV prostor boja. Detekcija se radi nad *Y* komponentom. Nakon toga neophodno je napraviti kopiju *Y* komponente za potrebe filtriranja horizontalnim i vertikalnim Sobel operatorom.

Sobel operator je dat sa:

$$h_v(v, h) = \frac{1}{4} \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad h_v(v, h) = \frac{1}{4} \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

Filtrirati jednu ulaznu sliku koristeći horizontalni operator, a drugu koristeći vertikalni kao što je to urađeno u vežbi 2.

Nakon toga, **za svaki piksel** u slici potrebno je posmatrati okruženje oko piksela, blok dimenzija  $N_w \times N_w$  i izračunati težinsku matricu  $M$ . Gde  $I_x$  odgovara rezultatu filtriranja horizontalnim Sobel operatorom, a  $I_y$  vertikalnim.

$$M = \begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix}$$

U ovom slučaju koristite okruženje piksela 3x3. Obzirom da je matrica  $M$  simetrična, potrebno je izračunati 3 vrednosti:  $mSumI_x$ ,  $mSumI_y$  i  $mSumI_{xy}$ .

Nakon računanja vrednosti  $M$  potrebno je izračunati vrednost faktora  $R$  koristeći jednačinu:

$$R = \det(M) - \alpha \text{trace}(M)^2$$

Gde  $\det$  označava determinantu matrice, a  $\text{trace}$  zbir elemenata na glavnoj dijagonali. Koeficijent  $\alpha$  predstavlja empirijski određenu konstantu i ima vrednost 0.04.

Determinantu matrice računate kao:

$$\det(M) = mSumI_x * mSumI_y - mSumI_{xy} * mSumI_{xy}$$

Nakon što su izračunate vrednosti  $R$  za svaki piksel u slici, potrebno je skalirati vrednosti matrice  $R$  tak da su u intervalu  $[0, 255]$ . Pronaći minimalni i maksimalni element matrice, potom svaku vrednost iz matrice transformisati koristeći jednačinu:

$$R' = (R - \min(R)) * 255 / (\max(R) - \min(R))$$

Proći kroz normalizovanu matricu  $R$ , i za svaku vrednost  $R(i, j)$  koja prelazi granicu *threshold* proveriti da li predstavlja lokalni maksimum za blok okolnih piksela dimenzija 3x3. Ukoliko predstavlja, tu tačku smatramo uglomn.

Za svaki detektovani ugao i njegove koordinate  $i, j$  na ulaznoj slici u RGB formatu ucrtati zeleni krug sa centrom u  $i, j$  prečnika 3 piksela. Za crtanje kruga koristiti funkciju `drawCircleRGB`.