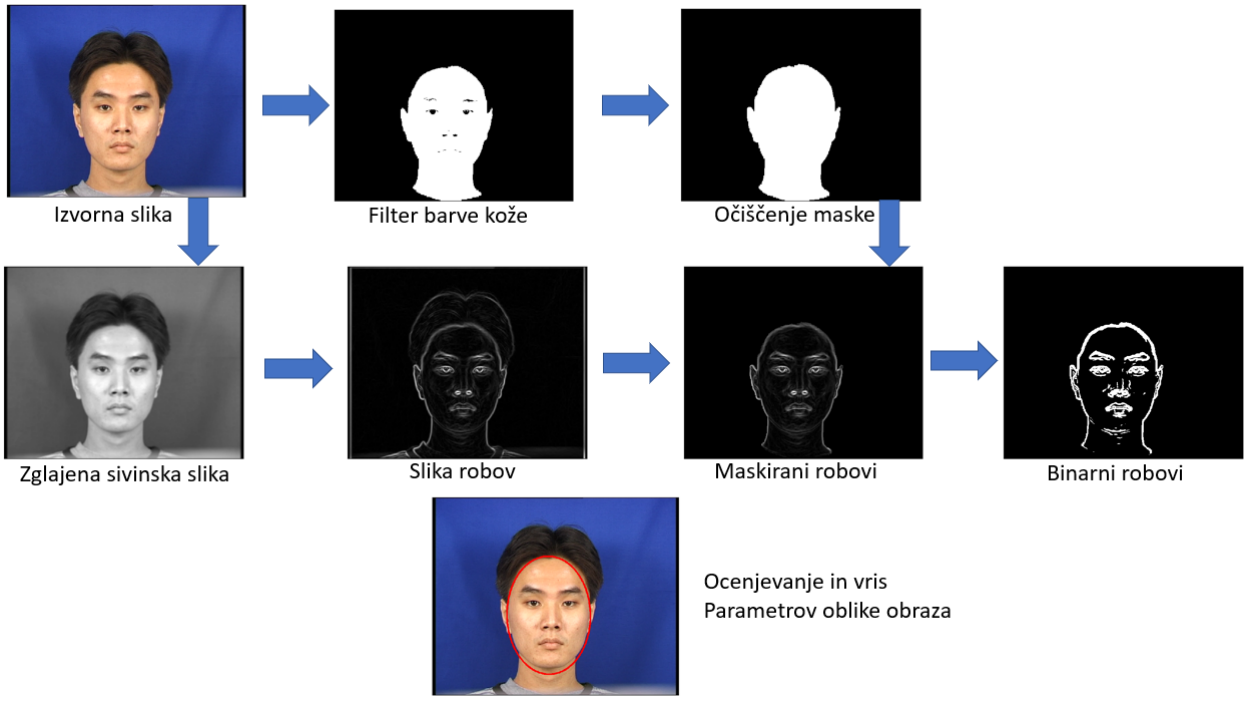
LABORATORIJSKA VAJA 5 – OBDELAVA SLIK ZA DETEKCIJO OBRAZOV

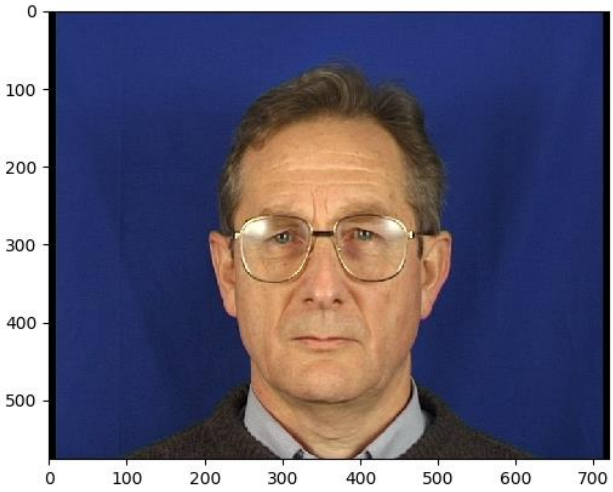
IDEJA VAJE

Namen vaje je seznaniti študente z različnimi postopki obdelave slik ter osnovami postopka optimizacije v smislu najmanjših kvadratov na primeru aplikacije za zaznavo obrazov na slikah. V ta namen bomo tekom vaje naredili pregled celotnega postopka obdelave slike od zajema slike do izračuna lokacije in oblike obraza na sliki. Vse postopke, ki jih potrebujete zaizvedbo vaje ste poznali tudi na predavanjih in so natančneje predstavljeni v prosojnicah predavanj. Shema celotnega sistema je predstavljena na sledeči sliki:

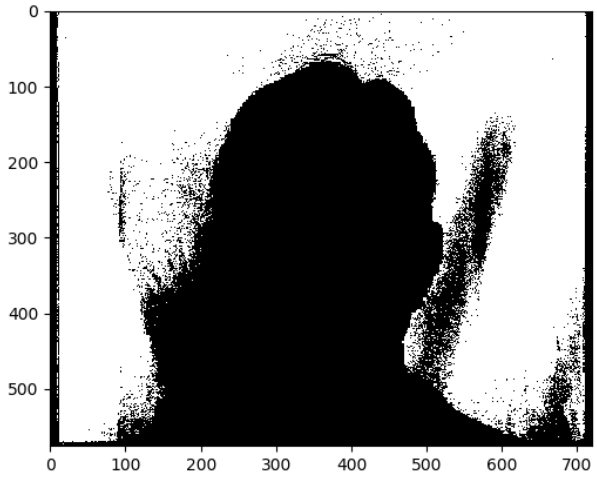


TEORETIČNO OZADJE

Barvno filtriranje: Za določitev lokacije obraza na sliki potrebujemo množico kandidatov za lokacije, kjer bi se obraz lahko nahajal. Kot tako množico lahko vzamemo tiste piksle slike, ki imajo barvo, ki ustreza barvi kože. Množico takih barv lahko identificiramo preko vrednosti R, G, B kanalov pikslov, in na podlagi omejitev teh vrednosti ustrezno omejimo področja slike, na katerih bi se obraz lahko nahajal. Če kot primer vzemimo sledečo sliko:



Lahko na njej prikažemo primer barvnega filtra (R<60, G<60, B>90), ki nam poda piksle slike, kjer imata zelen in rdeči kanal vrednost, manjšo od 60, modri kanal pa ima vrednost, večjo od 90. Piksli, kjer ta pogoj drži, so na sledeči sliki (tj., sliki po filtriranju) prikazani z belo barvo, piksli, kjer pogoj ni izpolnjen, pa s črno barvo:



Vidimo, da smo z opisanim barvnim filtrom skoraj v celoti pokrili območje modrega ozadja slike. Z več pogoji lahko na podoben način sestavimo barvni filter, ki se odziva na barvo kože, kar lahko služi kot prvi korak v postopku detekcije lokacije in oblike obraza na sliki.

V pythonu je bil opisan barvni filter implementiran s sledečo kodo:

```
import numpy as np
import matplotlib.pyplot as plt
import cv2
from functools import reduce

img = cv2.imread("slike/slika2.png")[:, :, ::-1]
plt.imshow(img)
plt.show()

r, g, b = [img[:, :, c] for c in range(3)]

mask = reduce(np.logical_and, (r<60, g<60, b>90))

plt.imshow(mask, cmap="gray")
plt.show()
```

Opomba: funkcija functools.reduce nam omogoča lažje veriženje logične operacije „in“ („and“) nad numpy arrayi. Ekvivalentni zapis izračuna maske brez njene uporabe je

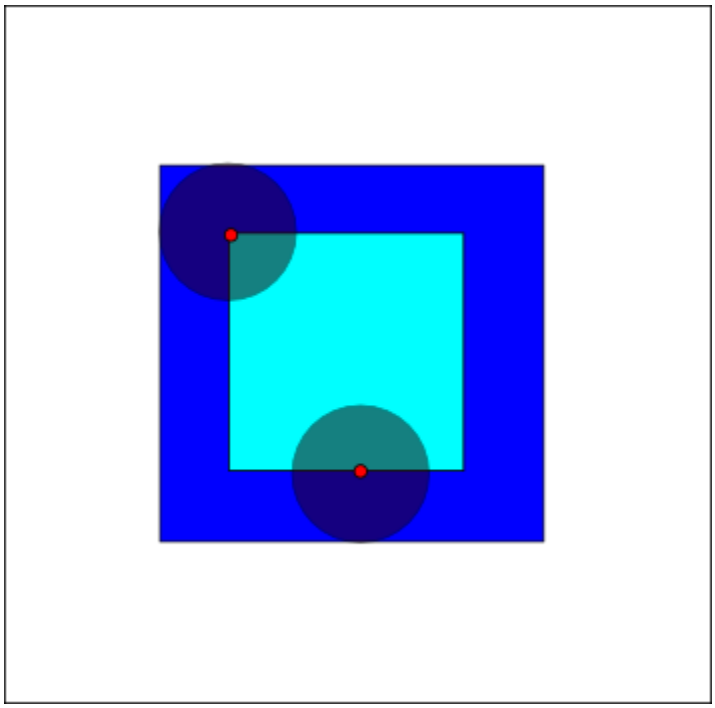
```
mask = np.logical_and(r<60, np.logical_and(g<60, b>90))
```

Morfološke operacije: Po prvotni določitvi področja na sliki, ki ustreza barvi kože, želimo to oceno postopoma izboljšati. Prvi korak v to smer je, da zapolnimo luknje v področju ki lahko nastanejo npr. zaradi oči, in zgladimo robove področja. To lahko dosežemo z t.i. morfološkimi operacijami. Morfološke operacije so lokalne operacije nad binarnimi slikami (slikami, kjer imajo piksli vrednosti 1 ali 0), pri katerih z uporabo strukturnega elementa sistematično spreminjamo oblike objektov na sliki. Osnovni morfološki operaciji sta:

Erozija binarne slike A s strukturnim elementom B, ki je definirana kot

$$A \ominus B = \bigcap_{b \in B} A_{-b},$$

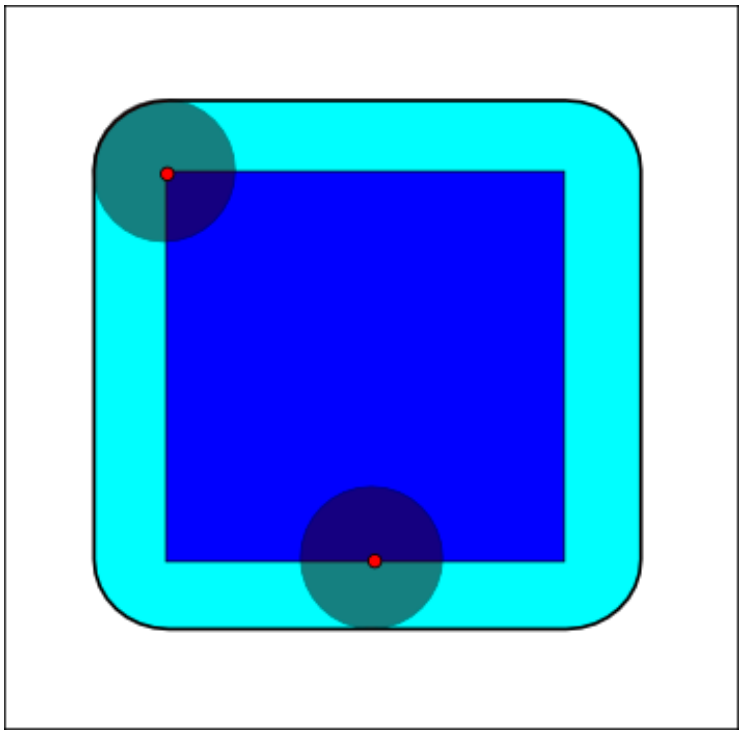
kjer A_{-b} pomeni translacijo (premik) matrike A za negativne koordinate vnosa strukturnega elementa b. Na spodnji sliki je prikazan primer erozije objekta temno-modre barve z okroglim strukturnim elementom črne barve. Rezultat je objekt svetlo modre barve.



Druga osnovna morfološka operacija je *dilatacija* binarne slike A s strukturnim elementom B. Operacija dilatacije je definirana kot

$$A \oplus B = \bigcup_{b \in B} A_b,$$

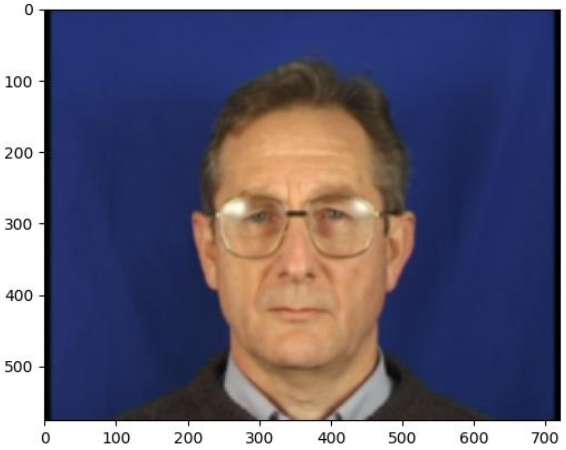
kjer A_b pomeni translacijo (premik) matrike A za koordinate vnosa strukturnega elementa b. Na spodnji sliki je prikazan primer dilatacije objekta temno-modre barve z okroglim strukturnim elementom črne barve. Rezultat je objekt svetlo modre barve.

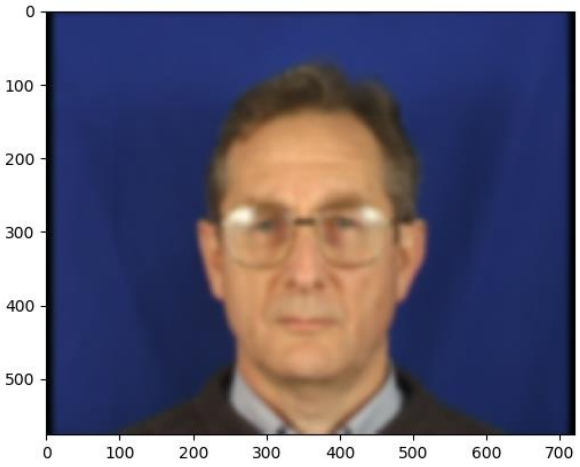


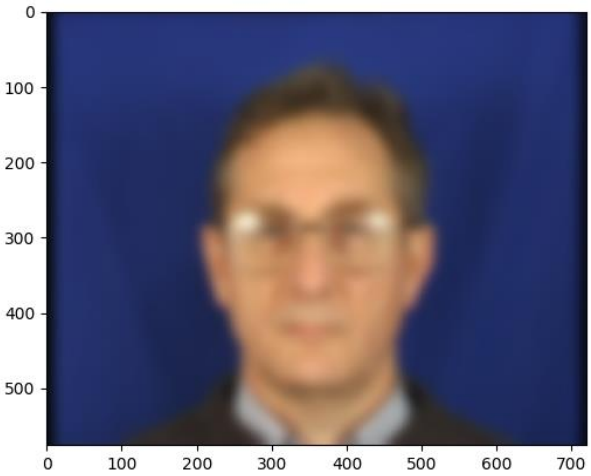
Lokalna obdelava slik: Lokalni postopki obdelave slik obsegajo družino operacij obdelave slik, izvedljivih preko operacije konvolucije. S to operacijo lahko implementiramo več pogosto uporabljenih postopkov obdelave slik, kot so glajenje, ostrenje, iskanje robov in odstranitev šuma. V okviru te vaje bomo za predobdelavo slike za detekcijo obrazov sliko najprej zgladili, nato pa na zglajeni sliki izvedli detekcijo robov. Glajenje je potrebno, ker zahteva postopek iskanja robov izračun aproksimacije gradienta slike, operacija odvajanja pa je občutljiva na potencialni šum v sliki. Z glajenjem vpliv šuma omilimo. Za glajenje slike uporabljamo konvolucijo z Gaussovim jedrom, ki je v koordinatah (x, y) določeno kot

$$\frac{1}{2\pi\sigma_x\sigma_y} e^{-\frac{x^2+y^2}{2\sigma_x\sigma_y}},$$

kjer σ_x in σ_y predstavljata standardni odklon Gaussovega jedra po x in y koordinatah. Za namen glajenja slik tipično uporabljamo enake vrednosti, $\sigma_x = \sigma_y = \sigma$. Na spodnji sliki je prikazana konvolucija slike z gaussovimi jedri velikosti 11, 21 in 41 pikslov, z vrednostmi σ 2, 4 oz. 8.


0
100
200
300
400
500
0
100
200
300
400
500
600
700


0
100
200
300
400
500
0
100
200
300
400
500
600
700


0
100
200
300
400
500
0
100
200
300
400
500
600
700

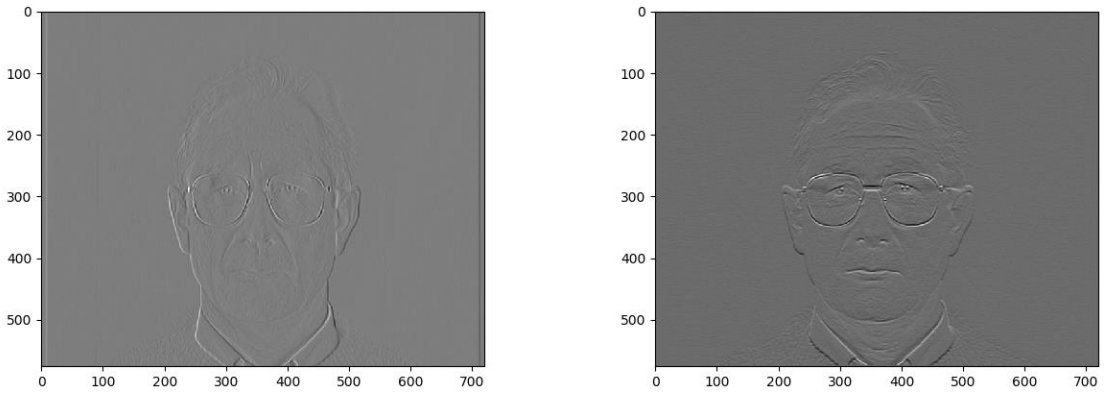
Učinek konvolucije slike z Gaussovim jedrom v frekvenčnem prostoru je zadušitev visofrekvenčnih komponent v sliki. Z obratno operacijo, ki zaduši nizkofrekvenčne komponente, lahko dosežemo izolacijo področij na sliki z visokofrekvenčnimi spremembami, kar lahko uporabimo za zaznavo robov področij. Primer konvolucijskega jedra, ki nam to omogoča, je Sobelov operator, ki je za zaznavanje robov v horizontalni smeri enak

$$\boldsymbol{G}_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}.$$

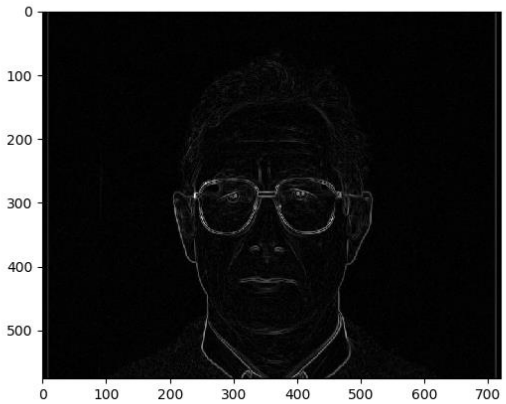
Za zaznavanje robov v vertikalni smeri je operator enak $\boldsymbol{G}_y = \boldsymbol{G}_x^T$. Odziva obeh operatorjev lahko združimo v približek amplitude gradienta slike $\boldsymbol{X}$ z operacijo

$$\nabla \boldsymbol{X} = \sqrt{(\boldsymbol{X} * \boldsymbol{G}_x)^2 + \left(\boldsymbol{X} * \boldsymbol{G}_y\right)^2}$$

Spodnja slika prikazuje odziva Sobelovih operatorjev v horizontalni in vertikalni smeri. Ti sliki ustrezata približkom odvodov slike v horizontalni in vertikalni smeri:



Sledeča slika pa prikazuje oceno amplitude gradienta slike glede na zgornja odziva:



Izračun oblik: Na podlagi zaznanih točk robov obraza bi radi izračunali lokacijo in obliko obraza na sliki. Pri tej vaji bomo za ta namen lokacijo in obliko obraza predstavili v obliki elipse. Elipsa je množica točk na koordinatah (x, y), ki izpolnjujejo enačbo

$$ax^2 + bxy + cy^2 + dx + ey + f = 0$$

Kjer vrednosti a, b, c, d, e, f določajo položaj, obliko in orientacijo elipse. Elipso, ki vsebuje obraz na sliki določimo dako, da najprej poiščemo koordinate vseh točk, ki sestavljajo rob obraza, torej točke $(x_1, y_1), \dots, (x_N, y_N)$. Te dobimo z binarizacijo slike amplitud gradienta, maskirane z odzivom filtra barve kože. Nato določimo središče elipse kot povprečno koordinato,

$$(\mu_x, \mu_y) = \left(\frac{1}{N}\sum_{i=1}^{N} x_i, \frac{1}{N}\sum_{i=1}^{N} y_i\right)$$

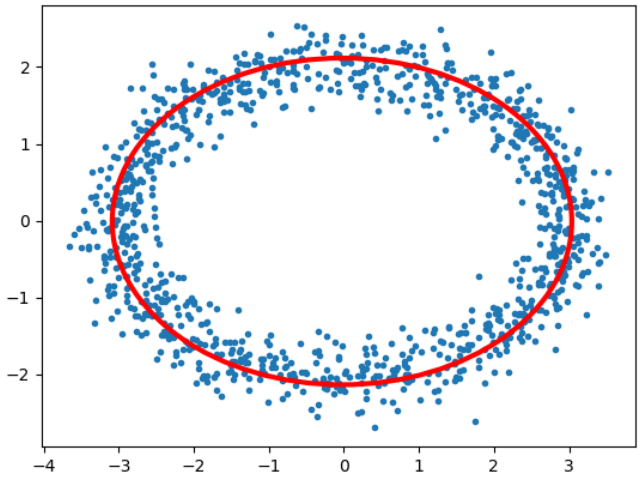
Na podlagi točk roba obraza nato sestavimo matriko $\boldsymbol{X} \in \mathbb{R}^{N\times 5}$, katere stolpci sestojijo iz vektorjev

$$\left[(x_i - \mu_x)^2, (x_i - \mu_x)(y_i - \mu_y), (y_i - \mu_y)^2, (x_i - \mu_x), (y_i - \mu_y)\right].$$

Vektor parametrov elipse $\theta = [a, b, c, d, e]$ nato določimo kot rešitev matrične enačbe $\boldsymbol{X}\theta = \mathbf{1}$ v smislu najmanjših kvadratov, kjer $\mathbf{1}$ predstavlja vektor razsežnosti N, katerega elementi imajo vsi vrednost enako 1. Najmanjšo kvadratno napako tega izraza zagotovi približek

$$\theta \approx (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\mathbf{1}$$

Na sledeči sliki rdeča elipsa prikazuje rešitev oblike oblaka modrih točk v smislu najmanjših kvadratov:



IZVEDBA

Naloga 1 (1 točki)

Dopolnite funkcijo skin_color_filter, tako, da implementira barvni filter za barvo kože. Barva kože je glede na vrednosti kanalov slike R, G, B določena s pogojem

$$R > 95, G > 40, B > 20, \max\{R, G, B\} - \min\{R, G, B\} > 15, |R - G| > 15, R > G, R > B.$$

Barvni filter je funkcija barvne slike, katere rezultat je matrika z vrednostmi 1 na pikslih, ki pogoju ustrezajo in z vrednostmi 0 na pikslih, ki pogoju ne ustrezajo.

Nato dopolnite funkcijo clean_skin_mask, tako, da masko kože, dobljeno z barvnim filtrom, popravite z zaporednima operacijama zapolnjevanja lukenj in morfološke dilatacije z uporabo kvadratnega strukturnega elementa velikosti 5×5 pikslov. Za zapolnjevanje lukenj uporabite v predlogi pripravljeno funkcijo fill(mask), za dilatacijo z ustreznim strukturnim elementom pa funkcijo cv2.dilate.

Naloga 2 (2 točke)

Dopolnite funkcijo rgb2gray, tako, da RGB barvno sliko pretvori v sivinsko. Vrednosti pikslov sivinske slike določite kot povprečja vrednosti pikslov po kanalih R, G in B.

Nato dopolnite funkcijo smooth_gray_image, tako, da dobljeno sivinsko sliko zgladi z uporabo Gaussovega filtra velikosti 7 pikslov s $\sigma = 2$. Za filtriranje uporabite funkcijo cv2.GaussianBlur.

Po glajenju sive slike določite njene robove. Funkcijo gradient_approximation dopolnite tako, da preko sobelovega operatorja določite njene gradiente po horizontalni in vertikalni smeri in jih ustrezno združite v sliko robov. Za filtriranje sivinske slike s sobelovimi operatorji uporabite funkcijo cv2.filter2D.

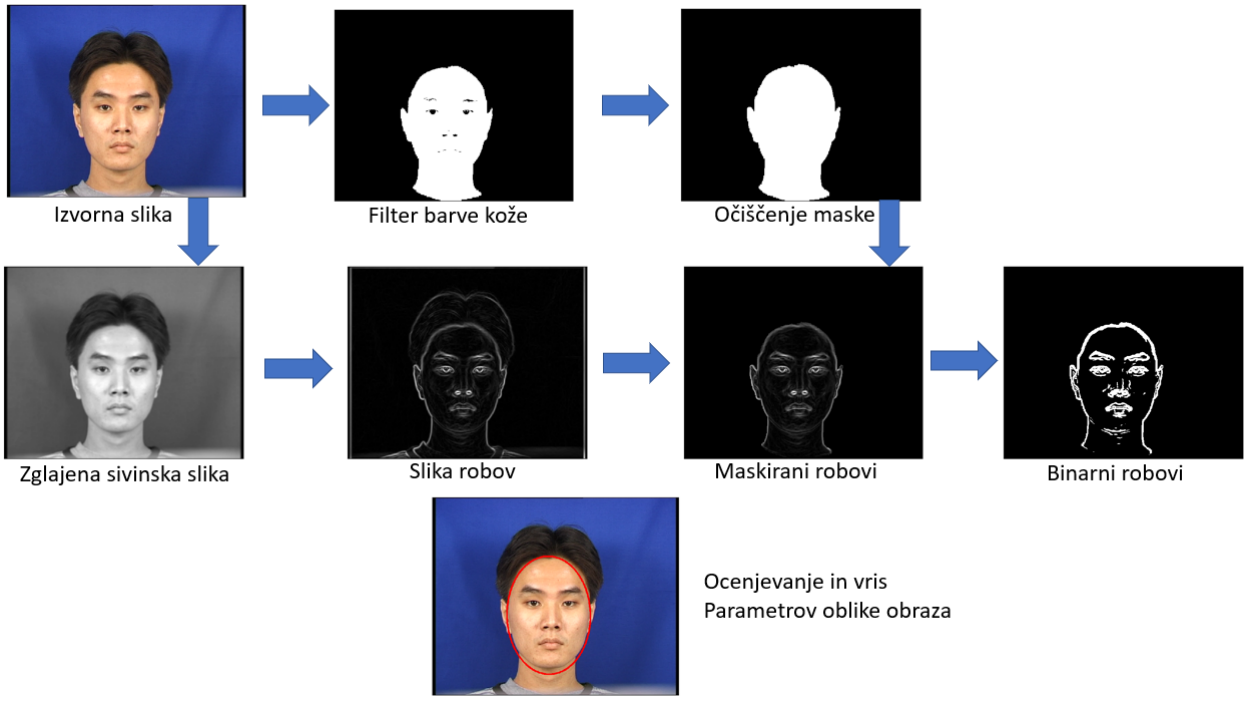
Pridobljen približek amplitude gradienta slike nato obdelajte tako, da se zavržejo robovi, ki ne pripadajo obrazu. To storite tako, da funkcijo mask_edges dopolnite tako, da gradiente slike pomnoži z masko barve kože.

Dobljen približek amplitude gradientov obraza, nato pretvorite v binarno sliko s postopkom upragovljanja. Funkcijo threshold_edges dopolnite tako, da glede na dano sliko gradientov vrne binarno matriko, ki ima vrednost 1, kjer je amplituda gradientov večja od 40, in 0 sicer.

Naloga 3 (1 točke)

S pomočjo implementiranih funkcij ter danih funkcij fit_ellipse in draw_ellipse implementirajte skripto, ki na dani vhodni sliki izvede zaznavo obraza ter izriše rezultat. Funkcija fit_ellipse kot vhod prejme binarno sliko robov obraza ter vrne parametre elipse, dobljene preko rešitve v smislu najmanjših kvadratov. Funkcija draw_ellipse kot vhoda prejme barvno sliko ter parametre elipse, in vrne sliko z vrisano elipso.

Celotni postopek detekcije obrazov je prikazan na spodnji shemi:



Naloga 4 (1 točka)

Implementirani sistem za zaznavanje obrazov preizkusite na desetih priloženih slikah in vsaj eni lastni sliki. Pri uporabi lastnih slik upoštevajte, da bo sistem deloval najbolje, če se lastnosti slik ujemajo priloženim – podobna resolucija in ustrezno osvetljen obraz na ozadju brez izrazitih podrobnosti. Preizkusite še delovanje sistema za zaznavanje obrazov, če slike rotirate oz. zrcalite preko horizontalne osi, kot je prikazano na spodnjem zgledu.