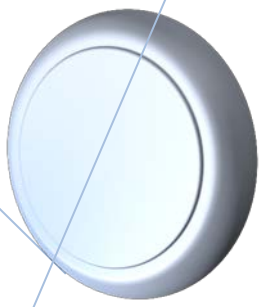


Laboratorijske vaje pri predmetu GST

Zbirka nalog, PRVA IZDAJA

Zbirka nalog predstavlja dopolnilno študijsko gradivo pri predmetu Govorne in slikovne tehnologije (GST) . Zbirka obsega navodila za izvedbo šestih laboratorijskih vaj, ki predstavljajo del obveznosti predmeta.

Simon Dobrišek in Vitomir Štruc
Ljubljana, oktober 2012

U N I V E R Z A V L J U B L J A N I

Fakulteta za elektrotehniko

Simon Dobrišek in Vitomir Štruc

LABORATORISJKE VAJE PRI PREDMETU

GOVORNE IN SLIKOVNE TEHNOLOGIJE

ZBIRKA NALOG

NA UNIVERZITETNEM ŠTUDIJSKEM PROGRAMU I. STOPNJE
ELEKTROTEHNIKA - ELEKTORNIKA

PRVA IZDAJA

Ljubljana, 2012

Predgovor

Pričujoča zbirka nalog predstavlja dopolnilno študijsko gradivo pri predmetu *Govorne in slikovne tehnologije* na Univerzitetnem študijskem programu I. stopnje, Elektrotehnika – Elektronika. Nastala je iz navodil za izvedbo laboratorijskih vaj pri tem predmetu v študijskem letu 2012/2013.

Namen gradiva je seznaniti študente z navodili laboratorijskih vaj in podati smernice za njihovo izvedbo. Zbirka nalog obsega sedem poglavij, od katerih prva tri namenjena vajam iz področja govornih tehnologij, druga tri vajam iz področja slikovnih tehnologij in zadnje, sedmo poglavje pa predstavlja vzorec poročila, ki ga morajo študentje v okviru vsake vaje izpolniti in oddati asistentu.

Avtorja se zahvaljujeta vsem sodelavcem Laboratorija za umetno zaznavanje, sisteme in kibernetiko na Fakulteti za elektrotehniko, ki so kakorkoli pripomogli k nastanku te zbirke.

Ljubljana, oktober 2012 Simon Dobrišek in Vitomir Štruc

Kazalo vsebine

1 Prva vaja: Predstavitev in analiza govornega signala

1.1 Kratka predstavitev vaje

Namen vaje je seznaniti študente z osnovnimi računskimi postopki analize in predstavitve govornih signalov z uporabo programskega okolja Matlab. Za uspešno izvedbo vaj se od študentov pričakuje osnovno predznanje iz teorije signalov in programskega okolja Matlab.

Za izvedbo vaje pridejo v poštev Matlab funkcije, ki so del originalne Matlab zbirke programskih orodij "*Signal Processing Toolbox*" ter funkcije, ki so del odprto-kodne zbirke "*Speech Processing Toolbox*" ter nekaj dodatnih funkcij, ki so jih pripravili sodelavci laboratorija LUKS. Za snemanje, osnovno analizo in predstavitev zvočnih govornih posnetkov v operacijskem sistemu Linux, pride v poštev še samostojni program "*Audacity*" ter program "*WaveSurfer*". Vsa navedena programska orodja so nameščena na računalnikih v učilnici LUKS.

Študentom je na razpolago podrobnejši opis posameznih nalog laboratorijske vaje. Dodatna pojasnila so podana na samih vajah.

Z vsemi izvedenimi nalogami študenti na samih vajah zberejo do **sedem točk.** Preostale **tri točke** zberejo s poročilom o izvedbi nalog, ki so ga dolžni predati asistentu najkasneje v roku enega tedna po izvedbi vaje (osebno ali preko e-pošte).

Ocenjevanje izvedbe posameznih nalog vaje se izvaja v sami učilnici ob koncu termina vaj. Ocenjevanje poročila se izvede naknadno v roku enega tedna po oddaji poročila.

1.2 Podrobna navodila

Namen vaje je seznaniti študente z osnovnimi računskimi postopki analize in predstavitve govornih signalov z uporabo programskega okolja Matlab. Za uspešno izvedbo vaj se od študentov pričakuje osnovno predznanje iz teorije signalov, klasičnimi programskimi orodji za obdelavo avdio signalov ter poznavanje programskega okolja Matlab.

Za izvedbo vaje pridejo v poštev Matlab funkcije, ki so del originalne Matlab zbirke programskih orodij »*Signal Processing Toolbox*« ter funkcije, ki so del odprto-kodne zbirke »Speech Processing Toolbox«

http://www.ee.ic.ac.uk/hp/staff/dmb/voicebox/voicebox.html

ter nekaj dodatnih funkcij, ki so jih pripravili sodelavci laboratorija LUKS. Za snemanje, osnovno analizo in predstavitev zvočnih govornih posnetkov v operacijskem sistemu Linux, pride v poštev še samostojni program Audacity

http://audacity.sourceforge.net/

ter program WaveSurfer

http://sourceforge.net/projects/wavesurfer/.

Vsa navedena programska orodja so nameščena na računalnikih v učilnici LUKS.

Med pripravami na izvedbo vaj pri tem predmetu so se študenti v učilnici seznanili z omenjenima samostojnima programoma in osnovnimi Matlab funkcijami iz omenjenih zbirk, kot so 'wavread', 'buffer', 'fft', 'spectrogram', 'plot', ipd. Za uporabo dodatnih Matlab funkcij je potrebno pred pričetkom dela v Matlab okolju dodati poti do ustreznih direktorijev, kar se izvede s spodaj podanimi ukazi.

```
addpath('/opt/voicebox');
addpath('/opt/luks/gst');
```

Naloga 1 (1 točka)

S pomočjo programa Audacity posnemite zvočni posnetek glasno in razločno izgovorjenega stavka: »*Nikóli in nikdár ne pústi pêti níz pôsla nekjé v vŕsti!*«. Posnetek naj vsebuje približno 500 ms začetnega in končnega premora. Za odstranitev morebitnega predolgega začetnega in končnega premor in šumov pos-

netek obrežite (funkcija »Trim« v programu Audacity). Posnetek shranite v zvočno datoteko 'govor.wav' v običajnem Microsoft RIFF formatu, pri čemer uporabite frekvenco vzorčenja 16 KHz in 16-bitno linearno kvantizacijo amplitude. V primeru, da ne uspete pridobiti primernega govornega posnetka, za nadaljnje naloge uporabite posnetek '/storage/vaje/databases/gst/govor.wav'. Raziščite možnosti različnih prikazov in obdelave zvočnega posnetka s programom Audacity.

Signal nato prikažite še s programom wavesurfer, kjer pri odpiranju posnetka izberite konfiguracijo »Speech analysis«.

Opomba: V poročilu komentirajte svojo izkušnjo z obema programom in na kratko opišite prikaze in funkcije, ki ste jih raziskali.

Naloga 2 (1 točka)

Spišite programski skript v okolju Matlab, ki pridobljeni posnetek v datoteki ‘govor.wav’ naloži v programsko okolje in v enem oknu grafično prikaže enega pod drugim njegov časovni potek ter kratko-časovni frekvenčni spektrogram. Za ta namen uporabite Matlab funkcije 'wavread', 'subplot', 'plot' in 'spectrogram'. Zgled programskega skripta, ki izvede zahtevano je podan spodaj. Opazujte vpliv parametrov funkcije 'spectrogram' na prikaz spektrograma in poskusite prikazati enkrat ozko- in drugič široko-pasovni spektrogram.

```
[s,fs]=wavread('govor.wav');
t=0:1/fs:(length(s)-1)/fs;

figure;
subplot(211);
plot(t,s); axis tight;
xlabel('t (sek)');ylabel('x(t)');
title('Govorni signal');

subplot(212);
spectrogram(s,160,120,512,fs,'yaxis');
xlabel('t (sek)'); ylabel('f (Hz)');
title('Govorni spekter');
```

Opomba: V poročilu podajte pridobljene grafične prikaze ter komentirajte opažanja, ki jih ponuja vpogled v spektrograme analiziranega govornega signala.

Naloga 3 (do 2 točki)

Spišite programski skript v okolju Matlab, ki pridobljeni posnetek ‘govor.wav’ naloži v programsko okolje in ga s pomočjo funkcije 'buffer' razdeli na prekrivajoče se kratke izseke N odtipkov in jih uredi po stolpcih v matriko (denimo, izseke dolžine 20 ms, t.j. 320 odtipkov, s polovičnim zaporednim prekrivanjem).

Za vsak izsek (stolpec omenjene matrike) s pričetkom pri *m*-tem odtipku govornega signala in z dolžino *N* odtipkov, izračunajte njegovo kratko-časovno glasnost po izrazu

$$L(m) = 10 \log_{10} \sum_{n=m}^{m+N} s(n)^2$$

ter število prehodov skozi ničlo po izrazu

$$Z(m) = \sum_{n=m+1}^{m+N} \frac{\left|\text{sign}\big(s(n)\big) - \text{sign}\big(s(n-1)\big)\right|}{2} \ .$$

V enem oknu grafično prikažite enega pod drugim poleg časovnega poteka signala in njegovega spektrograma še potek kratko-časovnih glasnosti in števila prehodov skozi ničlo.

Pri izvedbi vaje si pomagajte s spodnjim zgledom dela programske skripte, ki izvede zahtevane izračune.

```
…
% Rezanje signala na prekrivajoče izseke

N=320;P=160;
sx=buffer(s,N,P,'nodelay');

% Računanje glasnosti
Lx=10*log10(sum(sx.^2));

% Računanje prehodov skozi ničlo
Zx=sum(abs(diff(sign(sx))/2));

%Določanje časovnih trenutkov začetkov vsakega izseka
t=0:P/fs:P*(length(Lx)-1)/fs;
…
```

Na pravilen izračun števila prehodov skozi ničlo pri tihih delih govornega signala ima precej vpliva morebitna enosmerna komponenta v signalu. Te se znebimo z odštevanjem povprečne vrednosti odtipkov vsakega izseka signala, kot je ponazorjeno v spodnjem programskem skriptu.

```
% Odštevanje enosmerne komponente
so=sum(sx)/N;
for j=1:length(so);
    sx(:,j)=sx(:,j)-so(j);
end
…
```

Opomba: V poročilu podajte in komentirajte pridobljene grafične prikaze, denimo, pri katerih delih govornega signala je število prehodov skozi ničlo visoko in pri katerih nizko.

Naloga 4 (do 3 točke)

V prvi nalogi omenjeni izgovorjeni stavek »*Nikóli in nikdár ne pústi pêti níz pôsla nekjé v vŕsti!*« vsebuje vseh osem dolgih naglašenih samoglasnikov slovenskega govorjenega jezika, ki jih v računalniški različici mednarodne fonetične abecede IPA zapišemo (po vrsti pojavljanja v stavku na podčrtanih mestih) s simboli o:, a:, u:, E:, i:, O:, e:, in @:.

S pomočjo grafičnih predstavitev s programom ‘wavesurfer’ (tako, kot je omenjeno v prvi nalogi), ugotovite in si zabeležite začetne časovne trenutke v govornem signalu, kjer je izvedena akustična uresničitev omenjeni osem samoglasnikov. Zabeležene časovne trenutke pretvorite v začetne indekse odtipkov izsekov govornega signala, kjer so omenjeni glasovi uresničeni.

Za vsak tak izsek odtipkov govornega signala dolžine 20-50 ms (320-800 odtipkov) izvedite frekvenčno analizo in grafično predstavitev prevajalne funkcije LPC modela, iz katere lahko vizualno približno ocenite vrednost prve in druge formantne frekvence (F1 in F2) za vsak glas posebej. Grafično predstavitev pridobite s pomočjo posebej pripravljene Matlab funkcije ‘lpcformanti', kot je prikazano v naslednjem programskem skriptu.

```
[s,fs]=wavread('govor.wav');
% Indeks prvega odtipka izseka obravnavanega glasu
% (primer za zabeleženi začetni časovni trenutek pri 0.7s)
ti=0.7*fs;

% Število odtipkov izseka, število točk FFT in število LPC koefi-
cientov
wsize=640;
nfft=1024;
p=15;
lpcformanti(s(ti+1:ti+wsize),fs,wsize,nfft,p);
```

Ugotovljene vrednosti za vseh osem samoglasnikov nato vrišite v graf vrednosti formantnih frekvenc F1/F2, kjer vrednosti za F1 raztezajo od zgoraj navzdol od 0 do 1000 Hz in vrednosti za F2 od leve proti desni od 0 do 3000 Hz. Grafični prikaz takšnega grafa lahko pridobimo z Matlab programskim skriptom, ki je podan spodaj.

```
% Osem izmišljenih parov vrednosti formantnih frekvenc F1 in F2
ff=[304 734;282 625;617 852;726 1133;500 1063;554 1695;367
2094;265 2242];
% Osem obravnavanih samoglasnikov
ph=['u:';'o:';'O:';'a:';'@:';'E:';'e:';'i:'];

figure;
plot(ff(:,2),ff(:,1),'d','Color','red');
axis([0 3000 0 1000]);grid on;
set(gca,'YDir','reverse');
xlabel('F2 (Hz)'); ylabel('F1 (Hz)');

% Izpis simbolov samoglasnikov na pripadajočij F1/F2 točkah v
grafu
for i=1:length(ph)
    text(ff(i,2)+20,ff(i,1)-40,ph(i),'Color','blue');
end
```

Opomba: V poročilu podajte in komentirajte pridobljene grafične prikaze, denimo, v kakšni obliki se porazdeljujejo F1/F2 točke za obravnavane samoglasnike.

2 Druga vaja: Razpoznavanje ločeno izgovorjenih ukazov

2.1 Kratka predstavitev vaje

Namen vaje je seznaniti študente z računskimi postopki, ki omogočajo izvedbo osnovnega samodejnega razpoznavanja ločeno izgovorjenih govornih ukazov. Študenti se najprej seznanijo z postopkom detekcije govora v zvočnem signalu in postopkom pretvorbe govornega signala v nize vektorjev značilk, ki opisujejo njegove kratke časovne izseke. Nato se seznanijo še s postopkom razpoznavanja ločeno izgovorjenih govornih ukazov, ki temelji na prileganju dveh govornih signalov s dinamičnim ukrivljanjem časovne osi (angl. Dynamic Time Warping - DTW).

Za izvedbo vaje pridejo v poštev Matlab funkcije, ki so del originalne Matlab zbirke programskih orodij "*Signal Processing Toolbox*" ter funkcije, ki so del odprto-kodne zbirke "*Speech Processing Toolbox*" ter nekaj dodatnih funkcij, ki so jih pripravili sodelavci laboratorija LUKS.

Študentom je na razpolago podrobnejši opis posameznih nalog laboratorijske vaje. Dodatna pojasnila so podana na samih vajah.

Z vsemi izvedenimi nalogami študenti na samih vajah zberejo do **sedem točk.** Preostale **tri točke** zberejo s poročilom o izvedbi nalog, ki so ga dolžni predati asistentu najkasneje v roku enega tedna po izvedbi vaje (osebno ali preko e-pošte).

Ocenjevanje izvedbe posameznih nalog vaje se izvaja v sami učilnici ob koncu termina vaj. Ocenjevanje poročila se izvede naknadno v roku enega tedna po oddaji poročila.

2.2 Podrobna navodila

Namen vaje je seznaniti študente z računskimi postopki, ki omogočajo izvedbo osnovnega samodejnega razpoznavanja ločeno izgovorjenih govornih ukazov. Študenti se najprej seznanijo s postopkom detekcije govora v zvočnem signalu in postopkom pretvorbe govornega signala v nize vektorjev značilk, ki opisujejo njegove kratke časovne izseke. Nato se seznanijo še s postopkom razpoznavanja ločeno izgovorjenih govornih ukazov, ki temelji na prileganju dveh govornih signalov s dinamičnim ukrivljanjem časovne osi (angl. Dynamic Time Warping - DTW).

Za izvedbo vaje pridejo v poštev Matlab funkcije, ki so del originalne Matlab zbirke programskih orodij »*Signal Processing Toolbox*« ter funkcije, ki so del odprto-kodne zbirke »Speech Processing Toolbox«

http://www.ee.ic.ac.uk/hp/staff/dmb/voicebox/voicebox.html

ter nekaj dodatnih funkcij in programov, ki so jih pripravili sodelavci laboratorija LUKS.

Za uporabo dodatnih Matlab funkcij je potrebno pred pričetkom dela v Matlab okolju dodati poti do ustreznih direktorijev, kar se izvede s spodaj podanimi ukazi.

```
addpath('/opt/voicebox');
addpath('/opt/luks/gst');
```

Med pripravami na izvedbo te vaje so se študenti v učilnici seznanili s posebnim programom 'micivad', ki izvaja detekcijo govora v zvočnem signalu ter osnovnimi Matlab funkcijami iz omenjenih zbirk in dodatnimi funkcijami, kot so 'melcepst', 'dtwarp', 'dtwplot' ipd.

Naloga 1 (1 točka)

Spišite Matlab funkcijo 'getspeech', ki kliče zunanji program 'micivad' za neposredno pridobivanje odtipkov govornega signala v programsko okolje Matlab. Pri tem za klic zunanjega programa v Matlab okolju uporabi funkcijo 'eval' (ali podobno) po spodnjem zgledu.

```
...
filename = ['/tmp/' getenv('USER') '_micivad.wav'];
...
eval(['!micivad ' filename]);
...
[s,fs]=wavread(filename);
```

Opomba: V poročilu podajte in komentirajte celotno programsko kodo spisane funkcije getspeech in komentirajte vaše izkušnje s programom'micivad'.

Naloga 2 (1 točka)

Z uporabo Matlab funkcije 'melcepst' izvedite pretvorbo govornega signala v nize vektorjev značilk, ki so določeni kot koeficienti melodičnega kepstra. Grafično prikažite in vizualno analizirajte niz vektorjev, ki ste jih pridobili za poskusne kratke govorne posnetke. Preizkusite učinek različnih parametrov na določitev koeficientov, predvsem parametra ‘w’.

Opomba: V poročilu podajte grafični prikaz primera niza vektorjev značilk in komentirajte vaša opažanja v zvezi z dinamiko spreminjanja vrednosti posameznih koeficientov.

Naloga 3 (2 točki)

Preizkusite, preštudirajte in komentirajte pripravljeni Matlab funkciji 'dtwarp' in 'dtwplot', ki omogočata prileganje dveh govornih signalov z dinamičnim ukrivljanjem časovne osi. Ugotovite, kakšne prehode med stanji dovoljuje postopek, ki je implementiran v funkciji 'dtwarp'. Izvorni programski datoteki obeh funkcij se nahajata v direktoriju '/opt/luks/gst/'. Preizkusite učinek različnih parametrov na določitev koeficientov, predvsem parametra ‘w’ funkcije Matlab funkcije 'melcepst'.

Opomba: V poročilu podajte grafični prikaz prileganja dveh primerov govornih signalov z dinamičnim ukrivljanjem časovne osi, ki ste ga pridobili s funkcijo 'dtwplot'. Podajte še programsko kodo funkcije 'dtwarp', ki je po vrsticah opremljena z vašimi kratkimi komentarji. Grafično skicirajte, kakšne prehode med stanji dovoljuje postopek, ki je implementiran v funkciji 'dtwarp'.

Naloga 4 (2 točki)

Spišite program v okolju Matlab, ki z uporabo funkcije 'getspeech' in 'dtwarp' izvaja razpoznavanje ločeno izgovorjenih kratkih govornih ukazov, tako da posnetek neznanega govornega ukaza, pridobljenega s funkcijo 'getspeech', prilega danemu številu znanih prototipnih posnetkov izbranih ukazov, ki so bili na začetku delovanja programa naloženi v pomnilnik okolja Matlab.

Izberite pet primernih govornih ukazov (denimo, vendar ne nujno, »gor«, »dol«, »levo« , »desno« in »stop«) in za vsak ukaz posnemite tri učne primere/prototipe.

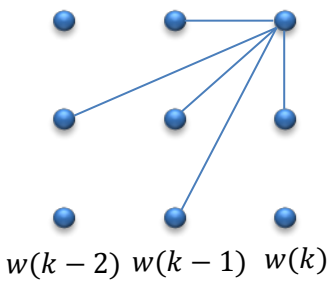
Demonstrirajte delovanje programa, ki najprej naloži vse učne primere v pomnilnik in nato izvaja omenjeno razpoznavanje ločeno izgovorjenih kratkih govornih ukazov. Razpoznane ukaze naj program izpiše kot besedilo ali se za vsak razpoznan ukaz drugače smiselno odzove.

Posnemite še po pet testnih posnetkov za vsak izbran ločeno izgovorjen ukaz. Nato preizkusite delovanja programa z razpoznavanje testnih posnetkov in podajte rezultat poskusa kot odstotek pravilno razpoznanih testnih posnetkov.

Opomba: V poročilu podajte komentirano spisano programsko kodo, opišite izbrane govorne ukaze in vašo izkušnjo z demonstracijo delovanjem vašega programa. Podajte še rezultat poskusa s testnimi posnetki.

Naloga 5 (1 točka)

Programsko kodo Matlab funkcije '/opt/luks/gst/dtwarp.m' kopirajte v svoj delovni direktorij v datoteko 'dtwarp1.m' in preimenujte funkcijo v 'dtwarp1'. Funkcijo 'dtwarp1' nato spremenite tako, da bo dovoljevala prehode med stanji, ki jih ponazarja spodnja slika.



Slika 1: Ponazoritev dovoljenih prehodov pri vaji 2

Na podoben način pridobite in spremenite funkcijo 'dtwplot1' tako, da bo uporabljala vašo novo funkcijo 'dtwarp1'. Primerjajte grafični prikaz prileganja dveh posnetkov, kjer enkrat uporabite originalno funkcijo 'dtwplot' in drugič vašo spremenjeno funkcijo 'dtwplot1'.

Opomba: V poročilu podajte komentirano spisano programsko kodo funkcije 'dtwarp1' in podajte grafična prikaza obeh prileganj dveh posnetkov ter komentirajte morebitne razlike.

3 Tretja vaja: Tvorjenje umetnega govornega signala

3.1 Kratka predstavitev vaje

Namen vaje je seznaniti študente z računskimi postopki, ki omogočajo tvorjenje umetnih zvočnih govornih signalov.

Za izvedbo vaje pridejo v poštev Matlab funkcije, ki so del originalne Matlab zbirke programskih orodij "Signal Processing Toolbox" ter zbirke Matlab funkcij "Sinewave Speech Analysis/ Synthesis" in nekaj dodatnih funkcij in programov, ki so jih pripravili sodelavci laboratorija LUKS. Med pripravami na izvedbo te vaje so se študenti v učilnici seznanili z dodatnimi MatLab funkcijami, kot so, 'swsmodel', 'synthtrax', 'lpcfit', 'lpcsynth', 'tdpsola' ipd.

Študentom je na razpolago podrobnejši opis posameznih nalog laboratorijske vaje. Dodatna pojasnila so podana na samih vajah.

Z vsemi izvedenimi nalogami študenti na samih vajah zberejo do **sedem točk.** Preostale **tri točke** zberejo s poročilom o izvedbi nalog, ki so ga dolžni predati asistentu najkasneje v roku enega tedna po izvedbi vaje (osebno ali preko e-pošte).

Ocenjevanje izvedbe posameznih nalog vaje se izvaja v sami učilnici ob koncu termina vaj. Ocenjevanje poročila se izvede naknadno v roku enega tedna po oddaji poročila.

3.2 Podrobna navodila

Namen vaje je seznaniti študente z računskimi postopki, ki omogočajo sintezo zvočnih govornih signalov. Vaja vsebuje **štiri naloge**, ki so podane v nadaljevanju.

Za izvedbo vaje pridejo v poštev Matlab funkcije, ki so del originalne Matlab zbirke programskih orodij »*Signal Processing Toolbox*« ter zbirke Matlab funkcij »Sinewave Speech Analysis/ Synthesis«

http://labrosa.ee.columbia.edu/matlab/sws

in nekaj dodatnih funkcij in programov, ki so jih pripravili sodelavci laboratorija LUKS.

Za uporabo dodatnih Matlab funkcij je potrebno pred pričetkom dela v Matlab okolju dodati poti do ustreznih direktorijev na enak način, kot je bilo izvedeno pri prejšnjih vajah.

Med pripravami na izvedbo te vaje so se študenti v učilnici seznanili z dodatnimi MatLab funkcijami, kot so, 'swsmodel', 'synthtrax', 'lpcfit', 'lpcsynth', 'tdpsola' ipd.

Naloga 1 (2 točki)

Spišite Matlab funkcijo 'sinsum', ki tvori signal kot uteženo vsoto sinusnih signalov izbranih frekvenc in amplitud po spodnjem izrazu

$$s(n) = \sum_{k=1}^{M} A_k \sin(2\pi n \frac{f_k}{f_0}) \ , \ \ n = 0, \dots, N-1 \ ,$$

pri čemer f_0 označuje izbrano frekvenco vzorčenja v Hertzih, A_k izbrano amplitudo posamezne sinusne komponente in f_k njeno izbrano frekvenco v Hertzih. Funkcijo najprej spišite z uporabo 'for' zank. Delovanje spisane funkcije nato primerjajte s spodnjim zgledom, ki izračun izvede z izrabo matričnih operacij.

```
function [s,fs] = sinsum(f,a,T,fs)

t = 0:1/fs:T-1/fs;
s = sum(sin(2*pi*f'*t).*repmat(a',1,length(t)));

return
```

V zgornjem zgledu sta 'f' in 'a' vektorja, ki vsebujeta frekvence f_k oziroma amplitude A_k (v linearnem merilu), 'T' predstavlja želeni čas tvorjenega signala (kjer je $T = N/f_o$) in 'fs' predstavlja frekvenco vzorčenja f_0.

Opomba: V poročilu podajte programsko kodo svoje spisane Matlab funkcije 'sinsum' in komentirajte primerjavo z zgornjo funkcijo. Tvorite primere signalov, ki vsebujejo do štiri sinusne komponente različnih amplitud, in v poročilu podajte grafični izris časovnega poteka in frekvenčnega spektra tvorjenega signala.

Naloga 2 (2 točki)

S pomočjo funkcije 'sinsum' iz prejšnje naloge tvorite signal, sestavljen iz dveh do štirih frekvenčnih komponent, katerih amplitude in frekvence se spremenijo vsake pol sekunde. Amplitude in frekvence sinusnih komponent nastavite na vrednosti prvih dveh do štirih formantnih frekvenc pet samoglasnikov ('a', 'E', 'i', 'O', 'u') slovenskega govora. Frekvence in amplitude izmerite (uporabite) s pomočjo LPC analize tako, kot ste to storili **pri četrti nalogi prve vaje**.

Glede na to, da so pri LPC analizi amplitude frekvenčnih komponent podane v decibelih, je potrebo v funkciji 'sinsum' amplitude preslikati iz logaritemskega v linearno merilo tako, kot je ponazorjeno spodaj, pri čemer se upošteva, da ji bilo pri LPC analizi uporabljeno okno širine 640 odtipkov.

```
function [s,fs] = sinsum(f,a,T,fs)

a = exp(a./20)*2/640;
a = a./sum(a);

t = 0:1/fs:T-1/fs;
s = sum(sin(2*pi*f'*t).*repmat(a',1,length(t)));

return
```

Zgled dela Matlab funkcije, ki tvori zahtevani signal je podan spodaj, pri čemer sta upoštevani samo prvi dve formantni frekvenci izmerjeni na laboratorijskem poskusnem posnetku.

```
fs = 16000;

f = [610 1211 0 0]; a = [12.7 4.8 -5.5 -9.4];
sa = sinsum(f,a,0.5, fs);

f = [485 1695 0 0]; a = [12.7 4.8 -5.5 -9.4];
se = sinsum(f,a,0.5, fs);

f = [229 2148 0 0]; a = [12.7 4.8 -5.5 -9.4];
si = sinsum(f,a,0.5, fs);

f = [429 859 0 0]; a = [12.7 4.8 -5.5 -9.4];
so = sinsum(f,a,0.5, fs);

f = [289 640 0 0]; a = [12.7 4.8 -5.5 -9.4];
su = sinsum(f,a,0.5, fs);

s=[sa se si so su];
t = 0:1/fs:(length(s)-1)/fs;

...
```

Opomba: V poročilu podajte spisano programsko kodo, ki izvede zahtevano in podajte grafični izris časovnega poteka in frekvenčnega spektra tvorjenega signala. Komentirajte svoje ugotovitve.

Naloga 3 (1 točka)

Najprej pridobite poskusni govorni posnetek v trajanju do nekaj sekund. Pri tem uporabite frekvenco vzorčenja 16 kHz.

Z uporabo funkcij 'swsmodel' in 'synthtrax' iz zbirke Matlab funkcij »Sinewave Speech Analysis/ Synthesis« nato tvorite sinusni model s štirimi sinusnimi komponentami za dani posnetek govora in nato tvorite umetni govorni signal na podlagi pridobljenega modela. Grafično prikažite časovni potek in frekvenčni spekter izvirnega in umetnega govornega signala. Način uporabe obeh funkcij razberite iz njunega opisa, ki ju pridobite s klicem 'help swsmodel' in 'help synthtrax'.

Na podoben način uporabite še funkciji 'lpcfit' in 'lpcsynth' iz iste zbirke Matlab funkcij. Tudi v tem primeru izvedite najprej LPC analizo danega posnetka govora

in nato še LPC sintezo umetnega govora iz pridobljenega LPC modela ter grafično prikažite časovni potek in frekvenčni spekter izvirnega in umetnega govornega signala. Način uporabe obeh funkcij razberite iz njunega opisa, ki ju pridobite s klicem 'help lpcfit ' in 'help lpcsynth'.

Osnovni zgled uporabe omenjenih funkcij je podan spodaj. V zgledu se predpostavlja privzete vrednosti neuporabljenih vhodnih spremenljivk funkcij. Te vrednosti lahko študenti tudi spremenijo.

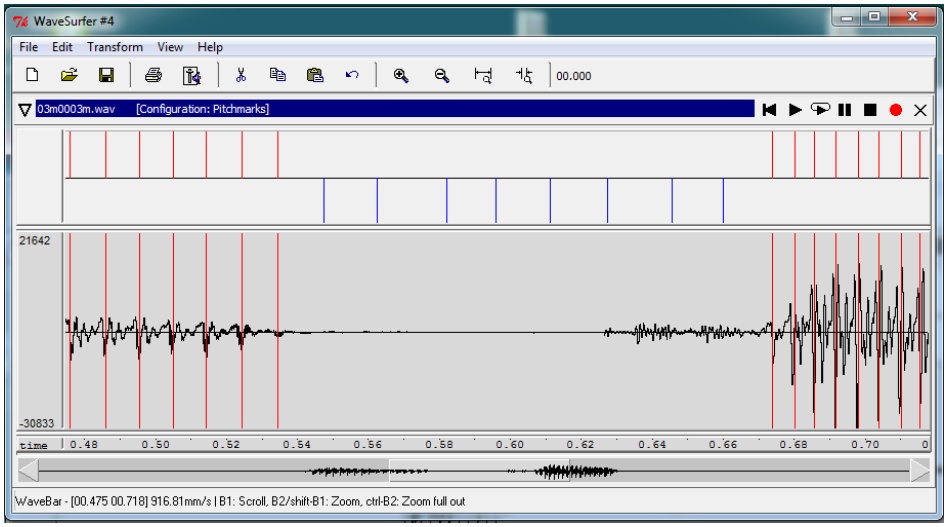
```
[s,fs]=wavread(posnetek.wav');

[F,M] = swsmodel(s,fs);
sr = synthtrax(F,M,fs);

[a,g] = lpcfit(s);
sl = lpcsynth(a,g);

...
```

Opomba: V poročilu podajte spisano programsko kodo in podajte grafični izris časovnega poteka in frekvenčnega spektra izvirnega in obeh tvorjenih govornih signalov. Komentirajte svoje ugotovitve

Naloga 4 (2 točki)

Najprej posnemite krajšo poskusno razločno izgovorjeno govorno izjavo (daljša beseda ali dve krajši besedi). Tudi tokrat uporabite frekvenco vzorčenja 16 kHz.

S pomočjo programa 'wavesurfer' natančno označite posamezne periode danega govornega signala na način, ki ga predvideva postopek spreminjanja govornega signala TD-PSOLA. Po zagonu programa 'wavesurfer' in odpiranju poskusnega posnetka izberite osnovno privzeto konfiguracijo 'Waveform'. Nato z desnim klikom in menijsko izbiro 'Create Pane -> Pitchmarks' dodajte panelno ploščo, ki omogoča natančno označevanje posameznih period. Oznake period dodajamo s postavitvijo kurzorja v zgornjo ali spodnjo polovico dodanega panela, opazovanjem položaja rdeče črte v panelu s časovnim potekom signala ter dvojnim klikom srednje tipke miške, ko je črta na mestu, ki označuje vrh poteka amplitude dane periode.



Slika 2: Primer pravilno označenega posnetka govora za vajo 3.

Oznake v rdeči barvi v zgornji polovici tega panela označujejo zveneče dela govora z očitnimi periodičnimi ponovitvami, oznake v modri barvi v spodnji polovici pa za nezveneče dele. Pri nezvenečih delih oznake postavljamo v razmikih, ki približno ustrezajo povprečnim razmikom v zvenečih delih. Zgled pravilno označenega govornega signala je podan na zgornji sliki.

Po shranjevanju posnetka se oznake period izvozijo v tekstovno datoteko z istim imenom, kot je ime datoteke označenega posnetka in s končnico '.pm'. Prva vrstica te datoteke je komentar, ki podaja časovno enoto v nadaljevanju podanih časovnih trenutkov oznak period (privzeto je to 1 sekunda). Sledijo po vrsticah podani časovni trenutki oznak period, kjer negativni predznak predstavlja (modro) oznako za nezveneči in pozitivna vrednost (rdečo) oznaka za zveneči del govornega signala. Primer vsebine te datoteke je podan na naslednji strani.

```
# 1.00000000
-0.017390
-0.035402
-0.055483
-0.073495
0.092334
0.111588
0.133947
0.154650
...
```

Pridobljeno datoteko preglejte v urejevalniku besedil in odstranite morebitno ponovitev oznak v istem časovnem trenutku. To se lahko zgodi zaradi nepazlji-

vega označevanja period in lahko povzroči težave pri nadaljnji uporabi teh oznak v funkciji 'tdpsola'.

V okolju Matlab nato spišite program, ki naloži obdelani signal in prebere oznake period iz omenjene tekstovne datoteke. Datoteko z oznako period lahko preberete z naslednjimi ukazi.

```
...
[s,fs]=wavread('posnetek.wav');
pm=textread('posnetek.pm','%f','commentstyle','shell');

vuv=sign(abs(pm)+pm);
pm=double([int32(abs(pm*fs));length(s)])';

%pm = find_pmarks(s,fs); %samodejno določi pm
%vuv = detect_vuv(s,fs,pm); %samodejno določi vuv

figure;  %Grafični prikaz signala z oznakami period
subplot(211); plot_pmarks(s,fs,pm,vuv);
subplot(212); spectrogram(s,160,120,512,fs,'yaxis');
xlabel('t (sek)'); ylabel('f (Hz)');
...
```

V zgornjem zgledu vektor 'pm' (iz angl. pitch marks) po prebiranju oznak iz tekstovne datoteke vsebuje indekse odtipkov signala, kjer so oznake period, in vektor 'vuv' (iz angl. voiced-unvoicede) dvojiško vrednost, kjer '1' označuje zveneči in '0' nezveneči odsek. Zgled ponuja tudi možnost samodejne določitve obeh vektorjev iz signala s funkcijama 'find_pmarks' in 'detect_vuv' ter pokaže, kako s pomočjo funkcije 'plot_pmarks' prikažemo oznake period.

S pomočjo funkcije 'tdpsola' nato spremite trajanje in/ali osnovni ton govora v izvirnem govornem signalu, pri čemer uporabite skaliranje nekje med 0.5 in 1.5. Način uporabe funkcije razberite iz njenega opisa, ki ga pridobite s klicem 'help tdpsola'. Poslušajte spremenjeni posnetek in grafično prikažite časovni potek in frekvenčni spekter izvirnega in umetno spremenjenega govornega signala.

Opomba: V poročilu podajte spisano programsko kodo in podajte grafični izris časovnega poteka in frekvenčnega spektra izvirnega in spremenjenega govornega signala. Komentirajte svoje ugotovitve.

4 Četrta vaja: Obdelava in analiza slik

4.1 Kratka predstavitev vaje

Namen vaje je seznaniti študente z različnimi postopki obdelave digitalnih slik. Študentje udejanjijo različne algoritme, ki se uporabljajo na področju slikovnih tehnologij, in jih preizkusijo na problemu samodejnega detektiranja obrazov.

Za izvedbo vaje pridejo v poštev Matlab funkcije, ki so del originalne Matlab zbirke programskih orodij "Image Processing Toolbox" ter nekaj dodatnih funkcij in programov, ki so jih pripravili sodelavci laboratorija LUKS. Med pripravami na izvedbo vaje se študenti v učilnici seznanijo s problematiko, ki jo poskušajo rešiti ter programskimi orodji, ki jih pri tem uporabljajo.

Študentom je na razpolago podrobnejši opis posameznih nalog laboratorijske vaje. Dodatna pojasnila so podana na samih vajah.

Z vsemi izvedenimi nalogami študenti na samih vajah zberejo do **sedem točk.** Preostale **tri točke** zberejo s poročilom o izvedbi nalog, ki so ga dolžni predati asistentu najkasneje v roku enega tedna po izvedbi vaje (osebno ali preko e-pošte).

Ocenjevanje izvedbe posameznih nalog vaje se izvaja v sami učilnici ob koncu termina vaj. Ocenjevanje poročila se izvede naknadno v roku enega tedna po oddaji poročila.

4.2 Podrobna navodila

Namen vaje je seznaniti študente z različnimi postopki obdelave digitalnih slik, ki jih nato aplicirajo na problem detekcije obrazov. Študenti se najprej seznanijo s predvidenim postopkom detekcije (glej prilogo), ki ga je potrebno udejanjiti, ter algoritmi, postopki ter orodji, ki jih pri tem uporabljajo. Študenti se v okviru prve vaje seznanijo z lastnostmi, karakteristikami in načinom uporabe:

- barvnih filtrov,
- morfoloških operacij (dilatacije, erozije, zapiranja, …),
- konvolucijskih filtrov (glajenja, …),
- postopkov iskanja robov z gradientnimi operatorji (Sobel, …),
- upragovljanja,
- logičnih operacij med slikami,
- geometrijskih operacij (skaliranje, …), in
- posplošene Hough-ove transformacije.

Poleg zgoraj naštetih postopkov se študenti seznanijo še s frekvenčno predstavitvijo slike ter histogrami slik.

Za izvedbo vaje pridejo v poštev Matlab funkcije, ki so del originalne Matlab zbirke programskih orodij »*Image Processing Toolbox*« ter nekaj dodatnih funkcij, skript in programov, ki so jih pripravili sodelavci laboratorija LUKS in so dosegljivi preko spletnih strani predmeta; arhiv na:

http://luks.fe.uni-lj.si/sl/studij/GST/vaje/vaja4b.html).

Za uporabo dodatnih Matlab funkcij je potrebno pred pričetkom dela v Matlab okolju dodati poti do direktorijev, kamor je bil prenesen in razširjen arhiv *Vaja4_predloge.rar*. To se izvede bodisi z uporabo ukaza

```
addpath('ustrezna_pot');
```

bodisi s spremembo delovnega direktorija Matlab-a na direktorij, kjer se nahajajo dodatne funkcije.

Osnovo za delo študentov predstavlja skripta `vaja4_predloga.m`, znotraj katere se nahaja ogrodje programske kode za izvedbo postopka detekcije obraza. Skripta `vaja4_predloga.m` izvaja klice funkcij, ki pa jih morajo študenti udejanjiti sami. Vsaka takšna funkcija je znotraj skripte jasno označena s

komentarjem %IMPLEMETIRAJ. Primer takšnega funkcijskega klica je predstavljen spodaj:

```
%% Morfološka obdelava binarne maske koze
Skin_corrected = clean_skin_mask(Skin);     % IMPLEMENTIRAJ
```

Pri izvedbi funkcij si študenti pomagajo z ogrodjem, ki je vključeno v arhivu *Vaja4_predloge.rar* in določa zahtevani vmesnik (angl. Application Programming Interface - API). Za funkcijo iz prejšnjega primera je API zapisan v glavi datoteke `clean_skin_mask.m` in ga je mogoče prebrati direktno iz Matlaba z ukazom

```
help clean_skin_mask
```

ki vrne:

```
Funkcija popravi binarno masko podrocja koze tako, da zapolni luknje in
podrocje nekoliko razsiri

  Vhod:
       Skin ... binarna slika s prvo oceno maske podrocja koze (velikosti
axb)
  Izhod:
      Skin_corrected ... popravjena binarna maska podrocja koze (velikosti
                          axb) brez lukenj; slikovni elementi, ki
                         ustrezajo
                          barvi koze imajo vrednost 1, ostali vred-
                         nost 0

  Avtor: Vitomir Štruc
  Datum: 25.4.2012
  Copyright (c) 2012 FE UL
```

Kot testne podatke študenti uporabljajo izbrane slike z zbirke XM2VTS, ki jih lahko prenesejo z naslednje povezave:

http://luks.fe.uni-lj.si/sl/studij/GST/pub/data/

Vsak par študentov pri preizkušanju svojih rešitev uporabi eno, od asistenta določeno sliko obraza.

Naloga 1 (1 točka)

Spišite Matlab funkcijo `skin_color_filter_RGB`, ki s pomočjo posebnega »barvnega« filtra iz slike izloči vse slikovne elemente, ki ne ustrezajo barvi kože. Filter naj kot vhod vzame barvno sliko zapisano v RGB barvnem prostoru (v obliki 3D matrike velikosti *a* x *b* x *3*, kjer sta *a* in *b* višina in širina slike, in je tretja dimenzija barvni prostor) in na izhodu vrne binarno sliko, na kateri beli slikovni elementi z vrednostjo 1 ustrezajo slikovnim elementom z barvo kože in naj črni slikovni elementi z vrednostjo 0 ustrezajo vsem ostalim slikovnim elementom. Pri pisanju funkcije uporabite filter, ki upošteva naslednje pogoje:

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

Pri implementaciji si pomagajte z naslednjim zgledom, ki udejanja preprostejši filter z zgolj dvema pogojema:

```
%% Inicializacija izhoda – POZOR: nepreizkušena funkcija
Skin = [];

%% Zacetne operacije
[a,b,c] = size(X);
Skin = zeros(a,b);

%% Filter
R=X(:,:,1);
G=X(:,:,2);
B=X(:,:,3);

for i=1:a
    for j=1:b
       if(R(i,j)>95 & G(i,j)>40)
         Skin(i,j)=1;
       else
         Skin(i,j)=0;
       end
    end
end

%Zgornji filter lahko udejanjimo v eni vrstici
%Skin = R>95 & G>40
```

Opomba: V poročilu podajte in komentirajte celotno programsko kodo spisane funkcije* `skin_color_filter_RGB`*. Prikažite vhodno in izhodno sliko funkcije (ukaz: imshow), podajte in komentirajte frekvenčni spekter in histogram binarne slike, ki predstavlja izhod udejanjene funkcije ter ga primerjajte s spektrom in histogramom (sive) vhodne slike. Prikažite tudi posamezne barvne komponente vhodne slike.

POZOR: Funkciji* `show_histogram` *in* `show_log_mag_spectrum` *prikažeta histogram in spekter sive različice vhodne slike.

Naloga 2 (1 točka)

Spišite Matlab funkcijo `clean_skin_mask`, ki kot vhod vzame binarno sliko področja kože, ki predstavlja izhod funkcije `skin_color_filter_RGB`, to binarno področje obdela z zaporedjem morfoloških operacij:

- polnjenja lukenj ter
- dilatacije

in na izhodu vrne očiščeno področje kože brez lukenj. Pri dilataciji uporabite kvadratni (angl. square) strukturni element velikost 5 slikovnih elementov.

Pri udejanjanju funkcije si pomagajte z Matlab-ovimi funkcijami `imfill`, `strel`, in `imdilate`. Delovanje navedenih funkcij preverite s pomočjo ukaza `help`:

```
help imfill
help strel
help imdilate
```

Opomba: V poročilu prikažite vhodno in izhodno sliko ter komentirajte učinek morfološke obdelave. Podajte in komentirajte še celotno programsko kodo spisane funkcije.

Naloga 3 (1 točka)

Spišite Matlab funkcijo `rgbimage2greyimage`, ki kot vhodni podatek vzame izvirno vhodno RGB sliko obraza in na izhodu vrne sivo sliko. Funkcijo udejanjite s pomočjo Matlab funkcije `mean`, pri čemer upoštevajte, da sivi nivoji sive slike predstavljajo povprečno svetilnost vseh treh RGB barvnih komponent. Pri pisanju funkcije ne pozabite, da mora rezultat funkcije (t.j., izhodna siva slika) predstavljati sliko z elementi tipa `uint8`. Rezultat povprečenja je torej potrebno pretvoriti na naslednji način:

```
uint8(mean(…))
```

Opomba: V poročilu prikažite vhodno in izhodno sliko funkcije in podajte kodo spisane funkcije.

Naloga 4 (1 točka)

Spišite Matlab funkcijo `smooth_grey_image`, ki kot vhodni podatek vzame sivo sliko obraza in na njej izvede glajenje z Gaussovim filtrom velikosti 7x7 slikovnih elementov in standardno deviacijo σ=2. Funkcija naj na izhodu vrne glajeno različico vhodne sive slike.

Pri udejanjanju vaše funkcije uporabite Matlab funkcijo `fspecial` za izgradnjo Gaussovega filtra ter ukaz `imfilter` za izvedbo filtriranja. Problem glajenja robnih elementov slike rešite s pomočjo podvajanja (angl. *replication*) robnih slikovnih elementov.

Način uporabe potrebnih funkcij lahko preverite z naslednjimi ukazi

```
help fspecial
help imfilter
```

Opomba: V poročilu podajte in komentirajte spisano programsko kodo. Prikažite vhodno in izhodno sliko ter njuna frekvenčna spektra. Podajte vaša opažanja in morebitne komentarje v zvezi z razlikami v frekvenčnih spektrih obeh slik.

Naloga 5 (1 točka)

Spišite Matlab funkcijo `gradient_approximation`, ki kot vhodni podatek vzame glajeno različico sive slike obraza, s pomočjo Sobelovega operatorja izvede aproksimacijo odvoda slike v horizontalni in vertikalni smeri ter na izhodu vrne amplitudo odvoda. Pri udejanjanju vaše funkcije uporabite Matlab funkcijo `fspecial` za izgradnjo vertikalne Sobelove maske ter funkcijo `imfilter` za izvedbo filtriranja. Problem filtriranja robnih elementov slike rešite s pomočjo podvajanja (angl. *replication*) robnih slikovnih elementov.

Upoštevajte, da lahko Sobelov filter (oz. masko) za aproksimacijo odvoda v horizontalni smeri, pridelate s transponiranjem vertikalnega filtra. Pri izvedbi si pomagajte z naslednjimi kodnimi izseki:

```
%% Izgradnja filtra in filtriranje
hv = fspecial( . . . );
% generiraj vertikalno masko

hh = hv';
% generiraj horizontalno masko

Gx = (imfilter(double(SmoothGrey),hv,'replicate'));
% odvod v x smeri

Gy = . . .
% odvod v y smeri

Edges = sqrt(Gx.^2+Gy.^2);          % rezultat
```

Opomba: V poročilu podajte komentirano spisano programsko kodo funkcije in vhodno ter izhodno sliko funkcije. Prikažite frekvenčna spektra obeh slik in podajte vaša opažanja. Enako storite še za histograme slik.

Naloga 6 (1 točka)

Spišite Matlab funkcijo `threshold_image`, ki kot vhodni podatek vzame sliko robov dobljenih s Sobelovim operatorjem, jo upragovi in na izhodu vrne upragovljeno binarno sliko. Funkcija torej vsem slikovnim elementom, ki so nad vrednostjo praga, priredi vrednost ena in vsem slikovnim elementom, katerih sivi nivo je pod vrednostjo praga priredi vrednost nič.

Opomba: V poročilu podajte komentirano programsko kodo funkcije in vhodno ter izhodno sliko funkcije. Prikažite histograma obeh slik in podajte vaša opažanja.

Naloga 7 (1 točka)

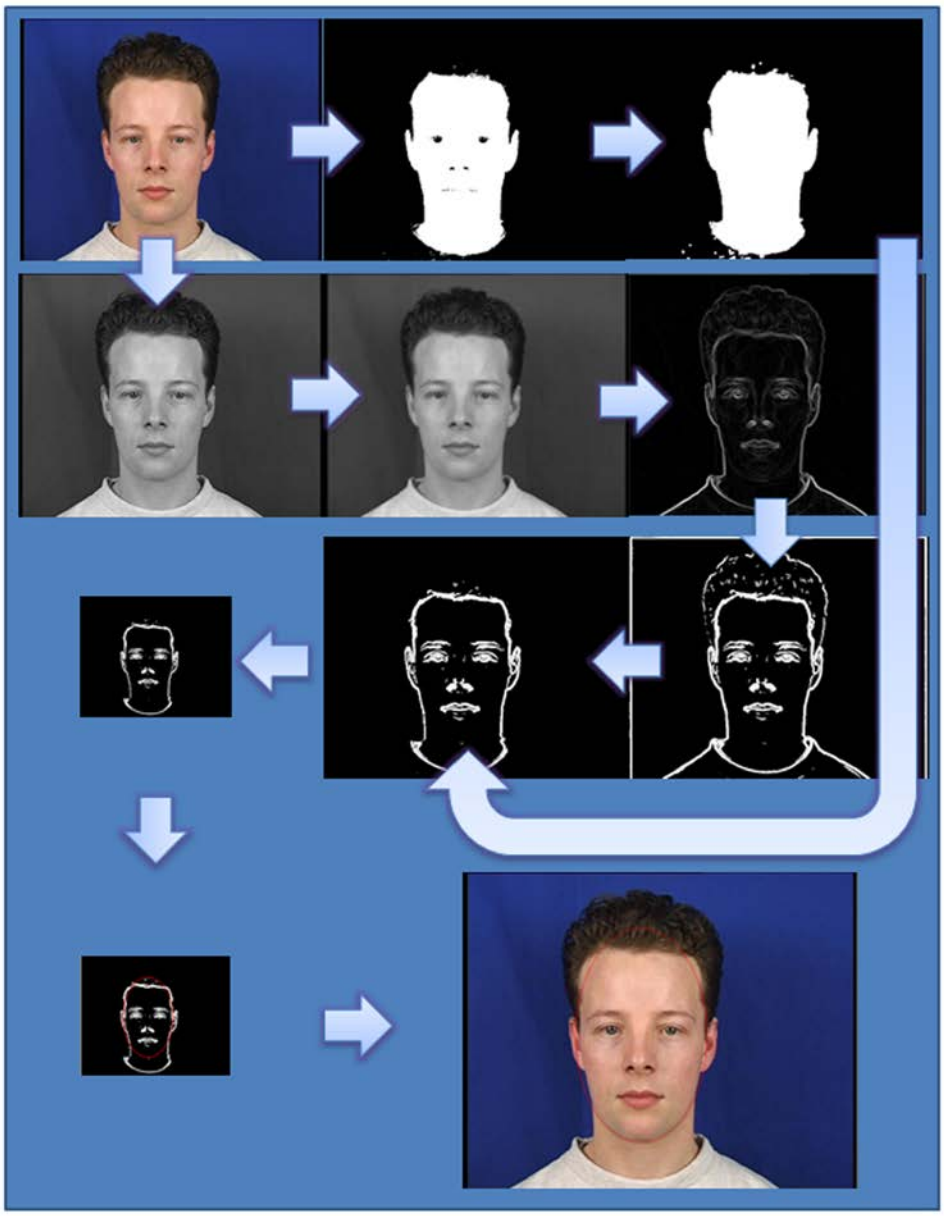
Spišite Matlab funkciji `mask_edges` in `im_resize`, kjer prva funkcija izvede množenje elementov očiščene binarne maske področja kože in upragovljenje slike robov in na izhodu vrne produkt obeh slik. Druga funkcija kot vhodni podatek vzame produkt slik in ga zmanjša na 0.3 njegove velikosti. Prvo funkcijo izvedite s pomočjo matričnega operatorja `'.*'`, ki pomnoži istoležne elemente slike in drugo s pomočjo Matlab funkcije `imresize` s privzeto metodo interpolacije.

Na koncu zaženite vse programske komponente skupaj s eliptično Hough-ovo transformacijo in si oglejte rezultate.

Opomba: V poročilu podajte komentirano programsko kodo funkcij in vhodne ter izhodne slike. Prikažite tudi končni rezultat celotne verige procesiranja, ki ga opravi skripta `vaja4_predloga` s pomočjo vaših funkcij.

4.3 Priloga

Na spodnji sliki je prikazan shematski potek postopka detekcije obraza, ki ga je v okviru vaje potrebno udejanjiti.



Slika 3: Shematski prikaz postopka vaje 4

5 Peta vaja: Šivanje slik in SURF deskriptorji

5.1 Kratka predstavitev vaje

Namen vaje je seznaniti študente z enim od možnih področij uporabe slikovnih tehnologij. V okviru vaje si bomo ogledali področje šivanja slik, ki ga v angleški literaturi označujemo z izrazom "image stitching". Študentje udejanjijo različne postopke, ki so potrebni za izvedbo šivanja slik in pri tem spoznajo še značilnosti in enega od namenov lokalnih deskriptorjev slik.

Za izvedbo vaje pridejo v poštev Matlab funkcije, ki so del originalne Matlab zbirke programskih orodij "Image Processing Toolbox", funkcije z javno dostopnih zbirk orodij ImageMosaicUsingSIFT ter OpenSURF in nekaj dodatnih funkcij in programov, ki so jih pripravili sodelavci laboratorija LUKS. Študenti vsa potrebna orodja prenesejo z naslednjega repozitorija.

Med krajšimi pripravami na izvedbo vaje se študenti v učilnici seznanijo s problematiko, ki jo poskušajo rešiti ter programskimi orodji, ki jih pri tem uporabljajo.

Študentom je na razpolago podrobnejši opis posameznih nalog laboratorijske vaje. Dodatna pojasnila so podana na samih vajah.

Z vsemi izvedenimi nalogami študenti na samih vajah zberejo do **sedem točk.** Preostale **tri točke** zberejo s poročilom o izvedbi nalog, ki so ga dolžni predati asistentu najkasneje v roku enega tedna po izvedbi vaje (osebno ali preko e-pošte).

Ocenjevanje izvedbe posameznih nalog vaje se izvaja v sami učilnici ob koncu termina vaj. Ocenjevanje poročila se izvede naknadno v roku enega tedna po oddaji poročila.

5.2 Podrobni opis

Namen vaje je seznaniti študente z enim od področij uporabe slikovnih tehnologij. V okviru te vaje se študenti srečajo s postopki šivanja slik, s katerimi je mogoče sestavljati večje mozaike iz večjega števila slik iste scene. Študenti se okviru vaje seznanijo s karakteristikami postopkov šivanja slik in primer takšnega postopka poskusijo tudi udejanjiti. Pri tem študenti podrobneje spoznajo:

- lokalne deskriptorje SIFT oz. SURF,
- postopke za iskanje ujemanja lokalnih deskriptorjev,
- postopek RANSAC,
- perspektivne preslikave ter pristope k njihovi določitvi,
- geometrijske transformacije slik, in
- celosten postopek šivanja slik.

Za izvedbo vaje pridejo v poštev Matlab funkcije, ki so del originalne Matlab zbirke programskih orodij »*Image Processing Toolbox*«, funkcije z javno dostopnih zbirk orodij kot sta »*Image-MosaicU-sing-SIFT*« ter »*OpenSURF*« in nekaj dodatnih funkcij in programov, ki so jih pripravili sodelavci laboratorija LUKS. Vsa potrebna orodja so dosegljiva preko spletnih strani predmeta; arhiv »`Vaja5_slike.rar`« na:

http://luks.fe.uni-lj.si/sl/studij/GST/vaje/vaja5b.html.

Za uporabo dodatnih Matlab funkcij je potrebno pred pričetkom dela v Matlab okolju dodati poti do direktorijev, kamor je bil prenesen in razširjen arhiv *Vaja5_slike.rar*. To se najhitreje izvede preko Matlabove menijske izbire

»`File –> Set Path -> Add with Subfolders`«.

Osnovo za delo študentov predstavlja skripta `vaja5_predloga.m`, znotraj katere se nahaja ogrodje programske kode za izvedbo postopka šivanja slik. Skripta `vaja5_predloga.m` izvaja klice funkcij, ki pa jih morajo študenti udejanjiti sami. Vsaka takšna funkcija je znotraj skripte jasno označena s komentarjem `%IMPLEMETIRAJ`. Primer takšnega funkcijskega klica je predstavljen spodaj:

```
%% Izracun SURF deskriptorjev
[des1, loc1] = izracunaj_surfe(X1,Options);    % IMPLEMENTIRAJ
```

Pri izvedbi funkcij si študenti pomagajo z ogrodjem, ki je vključeno v arhivu *Vaja5_slike.rar* in določa zahtevani vmesnik (angl. Application Programming Interface - API). Za funkcijo iz prejšnjega primera je API zapisan v glavi datoteke `izracunaj_surfe.m` in ga je mogoče prebrati direktno iz Matlaba z ukazom

```
help izracunaj_surfe
```

Kot testne podatke študenti uporabljajo slike, ki jih lahko prenesejo z naslednje povezave:

http://luks.fe.uni-lj.si/sl/studij/GST/pub/data/

Vsak par študentov pri preizkušanju svojih rešitev uporabi en par slik, ki ga določi asistent.

Naloga 1 (1 točka)

S pomočjo programa GIMP odprite obe sliki para, ju poizkusite poravnati ter sestaviti panoramsko sliko prikazane scene (mozaik), sestavljene iz obeh slik vašega para. Ko zaženete GIMP najprej ustvarite novo (prazno) sliko zadostnih dimenzij in vanj kot novi plasti (angl. *open as layers*) uvozite par slik, ki ju morate poravnati. Z GIMP-ovimi orodji nato poravnajte sliki tako, da se istoležni objekti obeh slik med sabo poravnajo. Rezultat poravnave shranite kot novo sliko.

Opomba: V poročilu prikažite rezultat vašega dela (sestavljeno sliko) in poročajte o morebitnih težavah pri poravnavanju slik.

Naloga 2 (1 točka)

Spišite Matlab funkcijo `izracunaj_surfe`, ki kot vhod vzame digitalno sliko in na izhodu vrne vrednosti in lokacije deskriptorjev slike. Pri udejanjanju funkcije uporabite funkcijo iz zbirke orodij OpenSurf z imenom `OpenSurf`. Delovanje navede funkcije preverite s pomočjo ukaza `help`:

```
help OpenSurf
```

Opomba: V poročilu podajte spisano programsko kodo in prikažite vhodni sliki ter obdelani sliki, na katerih so vrisane lokacije izračunanih SURF deskriptorjev.

Naloga 3 (1 točka)

Spišite Matlab funkcijo `prilegaj_surfe`, ki kot vhodni podatek vzame lokacije in vrednosti deskriptorjev izračunanih na paru slik in na izhodu vrne seznama lokacij ujemajočih SURF deskriptorjev. Pri udejanjanju funkcije uporabite funkcijo `siftMatch`, katere delovanje lahko preverite z ukazom:

```
help siftMatch
```

Opomba: V poročilu podajte spisano kodo in prikažite obe sliki para slik ter vanje vrišite ujemajoče se pare SURF deskriptorjev.

Naloga 4 (1 točka)

Spišite Matlab funkcijo homo_RANSAC, ki kot vhodni podatek vzame seznama izračunanih ujemajočih se parov deskriptorjev in na izhodu vrne projektivno transformacijsko matriko, s katero lahko ujemajoče se točke med seboj poravnamo. Transformacijsko matriko pri tem določite s postopkom RANSAC. Za udejanjenje funkcije uporabite funkcijo `findHomography`, delovanje katere lahko preverite z ukazom

```
help findHomography
```

Opomba: V poročilu podajte in komentirajte spisano programsko kodo. Izpišite transformacijsko matriko ter prikažite rezultat preslikave slike z izračunano transformacijsko matriko. Na koncu podajte še rezultat šivanja slik, ki ste ga dobili s pomočjo izračunane transformacijske matrike. Primerjajte sestavljeno sliko s sliko, ki ste jo sestavili s pomočjo GIMP-a in komentirajte opažanja.

Naloga 5 (1 točka)

V krovni skripti vaje `vaja5_predloga` zamenjajte imena slik tako, da bo na prvem mestu desna slika in na drugem leva slika. Še enkrat zaženite celotni algoritem ter opazujte spremembe.

```
ime_slikce1 = 'slike/par1_left.jpg';      % ime prve slikce, ki
jo bomo obdelali
ime_slikce2 = 'slike/par1_right.jpg';     % ime druge slikce, ki
jo bomo obdelali
```

Opomba: V poročilu podajte rezultat postopka ter komentirajte razlike v primerjavi s postopkom iz prejšnjih točk.

Naloga 6 (2 točki)

Popravite funkcijo `izracunaj_surfe` tako, da bo poleg slike na vhode sprejela še strukturo nastavitev `Options` , ki jo lahko posredujete funkciji `OpenSurf`. V strukturi spreminjajte polja:

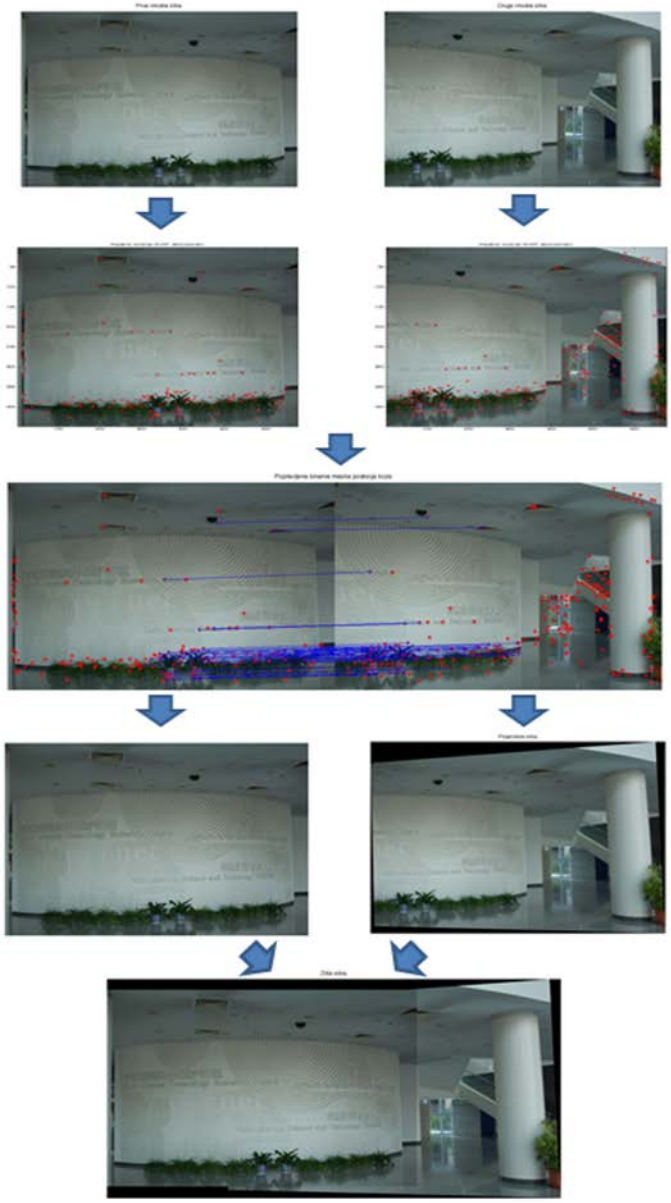
```
Options.tresh    (privzeta vrednost: 0.0002) - povečujte to vrednost
Options.octaves  (privzeta vrednost: 5)      - zmanjšujte to vrednost
```

in opazujte vpliv parametrov na postopek detekcije SURF deskriptorjev in posledično na uspešnost/hitrost šivanja slik.

Opomba: V poročilu podajte spisano kodo in komentirajte vaša opažanja glede vpliva parametrov **`Options.tresh`** ***in*** **`Options.octaves`** ***na učinkovitost/hitrost postopka šivanja slik.***

5.3 Priloga

Na spodnji sliki je prikazan shematski potek postopka šivanja slik, ki ga je v okviru vaje potrebno udejanjiti.



Slika 4: Shematski prikaz postopka vaje 5

6 Šesta vaja: Razpoznavanje objektov z analizo glavnih komponent

6.1 Kratka predstavitev vaje

Namen vaje je seznaniti študente z enim od možnih področij uporabe slikovnih tehnologij. V okviru vaje si bomo ogledali področje razpoznavanja objektov s postopkom analize glavnih komponent, ki ga v angleški literaturi označimo z izrazom "Principal Component Analysis - PCA ". Študentje udejanjijo različne postopke, ki so potrebni za izvedbo razpoznavanja objektov s postopkom PCA in pri tem spoznajo še značilnosti in karakteristike analize glavnih komponent.

Za izvedbo vaje pridejo v poštev Matlab funkcije, ki so del originalne Matlab zbirke programskih orodij "Image Processing Toolbox", funkcije z javno dostopne zbirke orodij PhD Toolbox in nekaj dodatnih funkcij, skript in programov, ki so jih pripravili sodelavci laboratorija LUKS. Kot podlago za razpoznavanje študenti uporabijo izbrane slike z javno dostopne zbirke COIL-20. Študenti vsa potrebna orodja in podatke prenesejo z naslednjega repozitorija.

Med krajšimi pripravami na izvedbo vaje se študenti v učilnici seznanijo s problematiko, ki jo poskušajo rešiti ter programskimi orodji, ki jih pri tem uporabljajo.

Študentom je na razpolago podrobnejši opis posameznih nalog laboratorijske vaje. Dodatna pojasnila so podana na samih vajah.

Z vsemi izvedenimi nalogami študenti na samih vajah zberejo do **sedem točk.** Preostale **tri točke** zberejo s poročilom o izvedbi nalog, ki so ga dolžni predati asistentu najkasneje v roku enega tedna po izvedbi vaje (osebno ali preko e-pošte).

Ocenjevanje izvedbe posameznih nalog vaje se izvaja v sami učilnici ob koncu termina vaj. Ocenjevanje poročila se izvede naknadno v roku enega tedna po oddaji poročila.

6.2 Podrobni opis

Namen vaje je seznaniti študente z enim od možnih področij uporabe slikovnih tehnologij. V okviru vaje si bomo ogledali področje razpoznavanje objektov s postopkom analize glavnih komponent (angl. Principal Component Analysis - PCA). Študenti se v okviru vaje seznanijo s/z:

- postopkom učenja projekcijske baze PCA,
- postopki preslikave slik v pod-prostor,
- rekonstrukcijskimi lastnostmi postopka PCA,
- razvrščanjem slik na podlagi postopka najbližjega soseda (1NN),
- postopki merjenja podobnosti z Evklidsko razdaljo, in
- postopki merjenja podobnosti z razdaljo City-block.

Za izvedbo vaje pridejo v poštev Matlab funkcije, ki so del originalne Matlab zbirke programskih orodij »*Image Processing Toolbox*«, funkcije z javno dostopne zbirke »PhD toolbox« ter nekaj dodatnih funkcij, skript in programov, ki so jih pripravili sodelavci laboratorija LUKS in so dosegljivi preko spletnih strani predmeta. Študenti vsa potrebna orodja in izbrane slike s podatkovne zbirke objektov (COIL-20) prenesejo s spletne strani vaje (arhiv: »*Vaja6_slike.rar*«). URL strani vaje je:

http://luks.fe.uni-lj.si/sl/studij/GST/vaje/vaja6b.html

Za uporabo dodatnih Matlab funkcij je potrebno pred pričetkom dela v Matlab okolju dodati poti do direktorijev, kamor je bil prenesen in razširjen arhiv *Vaja6_slike.rar*. To se najhitreje izvede preko Matlab-ove menijske izbire

»`File –> Set Path -> Add with Subfolders`«.

Osnovo za delo študentov predstavlja skripta `vaja6_predloga.m`, znotraj katere se nahaja ogrodje programske kode za izvedbo vaje. Skripta `vaja6_predloga.m` izvaja klice funkcij, ki jih morajo študenti dopolniti/udejanjiti ali prirediti, na označenih mestih pa je potrebno kodo tudi izpopolniti. Deli kode, ki zahtevajo pozornost študentov, so v skripti jasno označeni s komentarji, napisanimi z velikimi tiskanimi črkami, npr.: `%IMPLEMETIRAJ, % IZVEDI PRIKAZ, ….` Primer takšnega dela kode znotraj skripte `vaja6_predloga.m` je predstavljen spodaj:

```
figure('units','normalized','outerposition',[0 0 1 1]);
%prikazi rekonstruirano sliko - IZVEDI PRIKAZ
```

Naloga 1 (1 točka)

Preden lahko slike objektov razvrščate v različne razrede na podlagi njihovih koeficientov v pod-prostoru PCA, je potrebno pod-prostor določiti. To v praksi izvedemo na podlagi množice učnih slik, ki vsebuje primere slik objektov, ki jih želimo razpoznavati. Učni podatki, ki jih boste uporabili pri izračunu projekcijske baze postopka PCA, se nahajajo v direktoriju »ucne«.

Ustvarite seznam (tekstovno datoteko) z imeni vseh učnih slik v direktoriju »ucne« in ga shranite z izbranim imenom. Seznam lahko ustvarite direktno iz Linux terminala z naslednjim ukazom:

```
ls > seznam_ucnih_slikic.txt
```

Odprite ustvarjeno tekstovno datoteko in z nje izbrišite vse morebitne vrstice, ki se ne nanašajo na učne slike v direktoriju.

V skripti `vaja6_predloga.m` zamenjajte vrednost spremenljivke `ime_na_slikic` na ustrezno ime datoteke, ki ste jo pravkar ustvarili in pot do direktorija »ucne« tako, da bo ustrezala vaši poti.

```
%% Inicializacija - UREDI !!!!
clear all
verbose      = 1; % 1 - vklopi/ 0 - izklopi osnovno tekstovno porocanje
verbose_im   = 1; % 1 - vklopi/ 0 - izklopi prikazovanje rezultatov (slik)

% seznam datotek, ki jo bomo uporabili za ucenje PCA
imena_slikic = 'seznam_slikic.txt';       % ZAMENJAJ NA SVOJE IME

%pot do direktorija, kjer je zgornja datoteka in ucne slikce
pot_slikic = 'C:\Users\VitoS\Documents\GST\ucne\';    % ZAMENJAJ NA SVOJO
POT
```

Opomba: V poročilu kot prilogo priložite generirani seznam učnih slik in dodajte urejeno kodo.

Naloga 2 (1 točka)

Na označenem mestu (nekje v okolici vrstice 67) izvedite funkcijski klic funkcije `perform_pca_PhD`, ki iz učne množice slik izračuna projekcijsko bazo postopka PCA. Pri funkcijskem klicu izvedite učenje tako, da bo izračunani podprostor sestavljen iz 10 baznih vektorjev. Delovanje potrebne funkcije preverite s pomočjo ukaza `help`:

```
help perform_pca_PhD
```

Posebej si oglejte izhodno strukturo funkcije in v obliki slik prikažite izračunano povprečno sliko ter prvih pet baznih slik postopka PCA. Upoštevajte, da so tako povprečna slika kot tudi bazne slike (stolpci transformacijske matrike `model.W`) po navadi podani v obliki vektorjev in jih je potrebno za prikaz pretvoriti v slikovno obliko (glej ukaz `reshape`).

Opomba: Prikažite in komentirajte spisano kodo. V poročilu predstavite še povprečno sliko ter prvih pet lastnih vektorjev.

Naloga 3 (1 točka)

Spišite Matlab funkcijo `rekonstruiraj_sliko`, ki kot vhodni podatek vzame vektor značilk (t.j., vektor projekcijskih koeficientov postopka PCA), iz katerega želimo zopet rekonstruirati sliko, ter model, ki ga vrne funkcija `perform_pca_PhD` in na izhodu vrne rekonstruirano sliko. Rekonstrukcijo slike pri tem izračunamo v skladu z naslednjo enačbo:

$$\widetilde{X} = Wy + P,$$

kjer je W transformacijska matrika postopka PCA, y vektor značilk in P povprečni vektor učne množice slik. Prikažite rekonstruirano sliko in pri tem upoštevajte, da je rekonstruirana slika $\widetilde{X}$ zopet predstavljena v obliki vektorja.

Opomba: V poročilu podajte in komentirajte spisano kodo ter prikažite originalno sliko ter njeno rekonstrukcijo. Podajte opažanja glede razpoznavnosti slike, ipd.

Naloga 4 (1 točka)

Na označenem mestu skripte `vaja6_predloga.m` (nekje v okolici vrstice 164) izvedite funkcijski klic funkcije `linear_subspace_projection_PhD`, ki vhodno testno sliko preslika v izračunani PCA pod-prostor, in izračunane značilke vrne v vektor z imenom `testni_vektor`. Delovanje funkcije preverite s pomočjo ukaza `help`:

```
help linear_subspace_projection_PhD
```

Ne pozabite, da je potrebno pred izvedbo ukaza spremeniti še pot, ki določa, kje v vašem sistemu se nahajajo testne slike.

Opomba: V poročilu podajte in komentirajte spisano programsko kodo. Prikažite še primer testne slike, ki ste jo preslikali v pod-prostor.

Naloga 5 (1 točka)

Spišite Matlab funkcijo `ravrsti_testni_vzorec`, ki kot vhodni podatek vzame testni vektor značilk, vektorje značilk učnih slik (bazo sistema) ter oznake učnih slik in na izhodu vrne oznako testnega vektorja ter indeks učne slike, katere vektor značilk je bil pri izračunu razdalje testnemu vektorju najbližji. Funkcijo implementirajte z izračunavanjem Evklidske razdalje med vzorci, ki je med vektorjema $z = [z_1, z_2, \dots, z_d]^T$ in $w = [w_1, w_2, \dots, w_d]^T$ definirana kot:

$$d^2 = (z_1 - w_1)^2 + (z_2 - w_2)^2 + \cdots + (z_d - w_d)^2$$

Najpomembnejši del funkcije lahko izvedete v eni vrstici s pomočjo naslednjega ukaza:

```
%% Izračun ustreznega indeksa
[dummy,ind] = min(sum(((repmat(testni_vektor,b,1) -
ucne_znacilke).^2),2));
```

Funkcijo preizkusite in jo nato poizkusite udejanjiti še v obliki `for` zanke, tako da bo izvajala naslednji psevdo-algoritem:

```
Za vsako ucni vektor znacilk
   Izracunaj Evklidsko razdaljo med testnim vektorjem in trenutnim ucnim vektorjem
   Ce je razdalja najnizja med do zdaj izracunanimi razdaljami
        Si zapomni indeks ucnega vektorja
   Konec ce
Konec za vsak
```

Opomba: V poročilu podajte komentirano spisano programsko kodo funkcije ter rezultat razpoznavanja za izbrano testno sliko.

Naloga 6 (1 točka)

Preizkusite razpoznavanje za vseh 10 testnih slik iz direktorija »`testne`« in izračunajte učinkovitost razpoznavanja (odstotek pravilno razpoznanih slik).

Opomba: Podajte odstotek pravilno razpoznanih slik in komentirajte zakaj po vašem mnenju razpoznavanje ni bilo 100%.

Naloga 7 (1 točka)

Spremenite udejanjeno funkcijo `ravrsti_testni_vzorec` tako, da jo bo moč klicati z dodatnim argumentom, ki govori o razdalji, uporabljeni za razvrščanje. V primeru, da je četrti argument funkcije enak '*euc*', naj funkcija izvede enako funkcionalnost kot do sedaj. Ko je je četrti argument funkcije enak '*ctb*', naj funkcija vektorje razvršča na podlagi razdalje City-block, ki se med vektorjema $z = [z_1, z_2, \dots, z_d]^T$ in $w = [w_1, w_2, \dots, w_d]^T$ izračuna kot:

$$d = abs(z_1 - w_1) \quad + abs(z_2 - w_2) \quad + \cdots + abs(z_d - w_d)$$

Del, ki se nanaša na razdaljo City-block izvedite na podlagi naslednjega psevdoalgoritma:

```
Za vsako ucni vektor znacilk
  Izracunaj city-block razdaljo med testnim vektorjem in trenutnim ucnim vektorjem
  Ce je razdalja najnizja med do zdaj izracunanimi razdaljami
        Si zapomni indeks ucnega vektorja
  Konec ce
Konec za vsak
```

Preizkusite učinkovitost razpoznavanja z razdaljo City-block z vsemi testnimi slikami iz direktorija »test« in določite odstotek pravilno razpoznanih slik.

Opomba: V poročilu podajte komentirano programsko kodo funkcij. Primerjajte rezultat razpoznavanja z rezultatom razpoznavanja z Evklidsko razdaljo.

7 Zgled obrazca za pisanje poročila

POROČILO O LABORATORIJSKI VAJI XX

Podatki o študentih:

Vpisna številka	Ime in priimek	Datum

Namen vaje je bil seznaniti študente z izbranim področjem uporabe slikovnih tehnologij, natančneje področjem razpoznavanja objektov z analizo glavnih komponent. Pri vaji smo … in potem z …

Spodaj so podani komentarji, grafični prikazi in rezultati, ki se nanašajo na posamezne naloge te vaje.

Naloga 1

Nalogo smo izvedli tako, da smo najprej …

Naloga 2

Nalogo smo izvedli tako, da smo najprej …

Naloga 3

Nalogo smo izvedli tako, da smo najprej …

Naloga 4

Nalogo smo izvedli tako, da smo najprej …

Naloga 5

Nalogo smo izvedli tako, da smo najprej …

…