VIZUALIZACIJA BIOELEKTROMAGNETNEGA POLJA ČLOVEKA

Peter Peer

Raziskovalna naloga
iz
računalništva

predložena
Fakulteti za računalništvo in informatiko
Univerze v Ljubljani

September 1999

Mentorja: prof. dr. Igor Kononenko, prof. dr. Franc Solina

Naloga je bila izdelana pod mentorstvom prof. dr. Igorja Kononenka in prof. dr. Franca Soline in je last Fakultete za računalništvo in informatiko v Ljubljani. Za objavljanje in uporabo rezultatov dela je potrebno soglasje zgoraj omenjene ustanove.

Tekst je oblikovan z LaTeX urejevalnikom besedil.

Kazalo

[1] Članek, ki je izšel v zborniku ERK'99 (Portorož, Slovenija)!

Slike

Tabele

Poglavje 1

Vizualizacija bioelektromagnetnega polja človeka[1]

[1]Članek, ki je izšel v zborniku ERK'99 (Portorož, Slovenija)!

Vizualizacija bioelektromagnetnega polja človeka

Peter Peer, Bor Prihavec, Igor Kononenko, Franc Solina
Fakulteta za računalništvo in informatiko
Univerza v Ljubljani
Tržaška 25, 1000 Ljubljana, Slovenija
Peter.Peer@fri.uni-lj.si

Povzetek

Z razvojem tehnologije je postalo možno znanstveno raziskovanje nekaterih vidikov pojava avre (bioelektromagnetnega polja). Za pridobivanje podob aver prstov preiskovane osebe uporabljamo Kirlianov pojav, znan tudi pod imenom tehnika Vizualizacije Izločenih Plinov (angl. Gas Discharge Visualization - GDV). Ker razvijamo ekspertni sistem za postavljanje diagnoze iz podob GDV z uporabo tehnik strojnega učenja, ki se je izkazalo za uspešno tudi v klasični medicini, moramo iz pridobljenih podob izvleči kvantitativne informacije. Pričujoči članek zajema opis metod računalniškega vida, uporabljenih na podobah GDV z namenom dobiti avro celotnega telesa, kar predstavlja prvi korak k omenjenemu ekspertnemu sistemu.

1.1 Uvod

Nenavadne pojave proučujejo mnogi raziskovalci širom sveta, največkrat pod krinko parapsihologije; raziskovalci z največ tovrstnimi izkušnjami so Rusi. Najbolj proučujejo telepatijo, telekinezo in izvenčutno zaznavanje. Z napredkom tehnologije je postalo možno znanstveno proučevanje nekaterih vidikov pojava avre [4].

Zgodovina tako imenovanega Kirlianovega pojava, imenovanega tudi tehnika Vizualizacije Izločenih Plinov (angl. Gas Discharge Visualization - GDV) (širši pojem, ki vključuje tudi nekatere druge tehnike, je bioelektrografija) sega v leto 1777. Takrat je G. C. Lihtenberg v Nemčiji prvi zabeležil elektrografe drsečega izločanja v prahu, ki so ga povzročile statična elektrika in električne iskre. Kasneje so k razvoju tehnike prispevali mnogi raziskovalci [4]: Nikola Tesla v ZDA, J. J. Narkiewich-Jodko v Rusiji, Pratt in Schlemer v Pragi, vse dokler nista ruski tehnik Seymon D. Kirlian in njegova soproga Valentina opazila, da so se zaradi interakcije električnega toka s fotografskimi ploščami na filmu razvili odtisi živih organizmov. V letu 1970 je Kirlianove posnetke izdelovalo že na stotine navdušencev, toda raziskovanje je bilo do leta 1995 omejeno zgolj na uporabo fotografskega papirja.

Slika 1.1: Naprava Crown-TV za zajemanje bioelektromagnetnega polja

1.2 Kirlianova kamera

V letu 1995 je Korotkov [3, 4] s svojo skupino iz St. Petersburga v Rusiji razvil nov pristop, osnovan na video napravah CCD in računalniški obdelavi z njimi dobljenih podatkov. Njihova naprava Crown-TV je preprosta za uporabo in zato odpira praktične možnosti za proučevanje učinkov GDV.

Osnova za pristop GDV je elektromagnetno polje, ki ga ustvarimo z visokonapetostnim in visokofrekvenčnim generatorjem. Ko napetost preseže določeno mejno vrednost, plini, ki obdajajo preiskovani predmet, ionizirajo, stranski učinek te ionizacije pa je oddajanje kvantov svetlobe - fotonov.

Izločanje lahko optično zabeležimo s fotografskim aparatom, foto-senzorjem ali video kamero. Na postopek ionizacije vplivajo različni parametri [4]; tako je Kirlianov pojav rezultat mehanskih, kemičnih in elektromagnetnih procesov in vplivov polja. Izločanje plinov nam služi za poudarjanje in vizualizacijo izredno šibkih procesov.

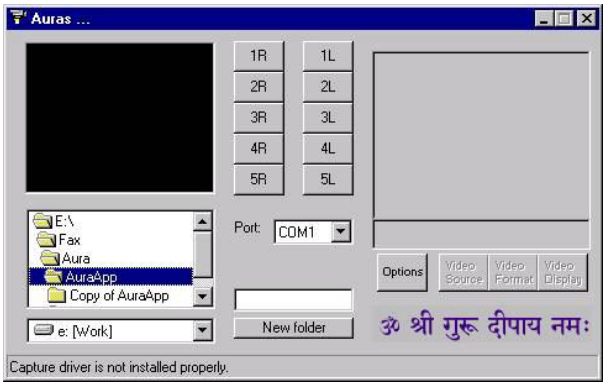
Parametrov, ki vplivajo na Kirlianov pojav, je razmeroma veliko. Izredno težko ali celo nemogoče je nadzirati vse, zato je v postopku izločanja vedno prisoten dejavnik nedoločenosti in stohastičnosti. To je tudi vzrok za to, da se tehnika ni bolj uveljavila v praksi, saj so imeli dobljeni rezultati nizko stopnjo ponovljivosti. Vse razlage Kirlianovega pojava pojmujejo fluorescenco kot izžarevanje živega bitja. Zaradi majhne ponovljivosti se je v akademskih krogih razširilo mnenje, da so vsi opazovani pojavi nič drugega kot nestalnost koronskega izločanja brez vsakršne povezave s proučevanim predmetom. Z moderno tehnologijo je postala ponovljivost dovolj visoka za zadostitev resnih znanstvenih preiskav.

Pri človeku najpogosteje snemamo avre prstov [5, 4] in zapise GDV krvnih izvlečkov [9]. V svetu trenutno potekajo številna proučevanja, katerih izsledki potrjujejo, da je človekova avra nedvomno tesno povezana z njegovim psihofizičnim stanjem, in kot taka uporabna pri diagnosticiranju, napovedovanju bolezni, izbiranju

terapije in nadziranju posledic le-te.

S pomočjo ruske naprave Crown-TV načrtujemo razvoj ekspertnega sistema za postavljanje diagnoze s pomočjo podob avre in uporabo tehnik strojnega učenja [1, 6]. Tehnike strojnega učenja so se v klasični medicini že potrdile [2].

1.3 Razvoj aplikacije



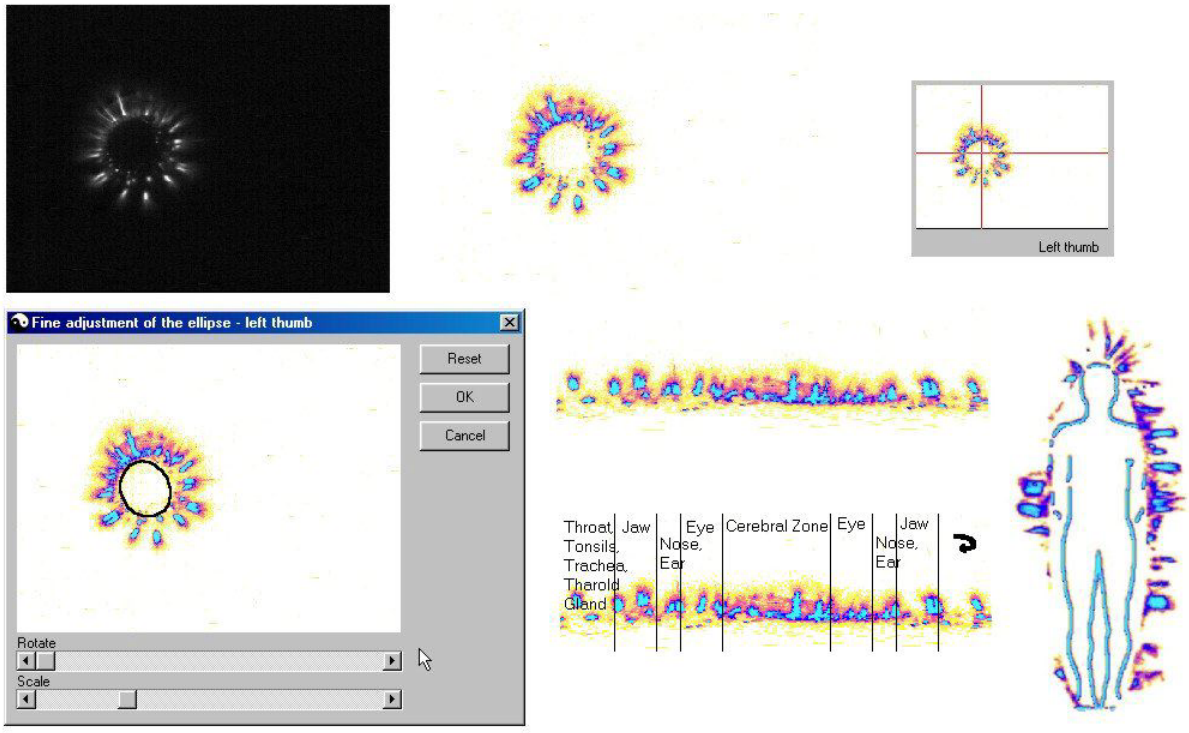
Slika 1.2: Uporabniški vmesnik aplikacije za zajemanje bioelektromagnetnega polja

Poleg možnosti direktnega diagnosticiranja iz avre celega telesa, je glavni namen aplikacije za vizualizacijo človeškega bioelektromagnetnega polja postaviti temelje za iskanje atributov, ki jih kot vhod potrebujejo algoritmi strojnega učenja.

Ker postopek vizualizacije izhaja iz sivinskih slik vseh desetih prstov, smo morali najprej razviti aplikacijo za zajemanje takšnih slik s pomočjo omenjene naprave Crown-TV. Uporabniški vmesnik te aplikacije je razviden iz slike 1.2.

Ko imamo na razpolago bioelektromagnetno polje vseh desetih prstov, lahko začnemo z vizualizacijo bioelektromagnetnega polja celotnega telesa. Osnovni koraki so naslednji:

- najprej se izvrši dinamična barvna transformacija zajetih sivinskih slik (transformacija temelji na izkušnjah bioenergetikov):
  - izvrši se posvetlitev vhodne sivinske slike,
  - izračuna se prag,
  - nad izračunan prag se izvrši preslikava barvne palete, ki je bila določena s pomočjo bioenergetikov,
- izračuna se središče vsakega prsta (uporabnik lahko središče interaktivno popravi),
- izvrši se postopek prileganja elips (tudi tukaj lahko uporabnik interaktivno popravi njeno lokacijo, velikost in zasuk),



Slika 1.3: Postopek vizualizacije: vhodna sivinska slika, izvršena barvna transformacija, interaktivno iskanje središča, interaktivno prileganje elips, generiranje panoramske slike, razrez panoramske slike in transformacija na telo

- na podlagi znane konture (elipse) prsta naredimo transformacijo barvne slike v normalizirano panoramsko sliko:
  - sliko razrežemo od središča proti desni strani,
  - v negativni matematični smeri naredimo za vsako stopinjo ustrezno preslikavo iz koordinatnega sistema xy v koordinatni sistem φd, pri čemer je φ kot pogleda in d razdalja od središča,
  - normalizacijo nam omogoča najdena elipsa; v panoramski sliki se elipsa preslika v premico,
- panoramske slike sedaj razrežemo na dele, ki naj bi že po kitajski tradicionalni medicini odražali stanje posameznih delov telesa [4], in končno
- ustrezno preslikamo dele panoramskih slik na obris človeškega telesa.

Opisani postopek vizualizacije je razviden tudi iz slike 1.3.

Prilagajanje elips temelji na algoritmu M. Piluja in sodelavcev [8]. Algoritem na vhodu zahteva množico točk, za katere predpostavlja, da pripadajo elipsi. To množico točk moramo zato zagotoviti v predprocesiranju:

- izvršimo binarizacijo vhodne sivinske slike avre prsta,
- izračunamo središče iz binarizirane slike,
- radialno poiščemo rob in
- sestavimo množico točk, ki naj bi pripadale konturi (elipsi) prsta.

Izhod algoritma je množica točk, ki pripadajo najbolje prilegajoči se elipsi.

Osnovne atribute, potrebne za strojno učenje, bomo dobili iz generiranih panoramskih slik. Če se bo pokazala potreba po dodatnih informacijah, bomo seveda uporabili tudi alternativne vire, na primer informacije iz narejene transformacije na celo telo ipd.

1.4 Zaključek

V članku je predstavljen pristop k obdelavi človeškega bioelektromagnetnega polja. Za namen diagnosticiranja uporabljamo napravo Crown-TV, tehniko GDV, metode računalniškega vida in strojno učenje. Prispevek daje težo predvsem metodam računalniškega vida, ki omogočajo vizualizacijo človeškega bioelektromagnetnega polja. Predstavljena je aplikacija, ki iz zajetih sivinskih slik avre prstov zgradi avro celega telesa.

Cilj projekta je ugotoviti zmožnost diagnosticiranja iz informacij, ki nam jih daje avra, s pomočjo metod strojnega učenja [1, 6], ki so se že izkazale v domenah zahodne klasične medicine[2].

V dosedanjih raziskavah so potekala snemanja koron jabolčnih olupkov, snemanja ljudi pod vplivom živobarvne majčke in snemanja žensk v različnih fazah ovulacijskega ciklusa. Rezultati kažejo, da:

- korone jabolčnih olupkov nosijo koristno informacijo za razločevanje vrste in starosti jabolk (Korone niso rezultat zgolj naključnih procesov!),
- bioelektromagnetno polje ljudi se pod vplivom živopisane majčke močno poveča,
- faza ovulacijskega ciklusa je povezana z deli koron prstov, ki po kitajski medicini ustrezajo urogenitalnemu sistemu in hipofizi, kar se sklada z znanjem zahodne medicine na področju ginekologije.

V prihodnosti bomo preučevali vplive različnih pripomočkov na človeško bioelektromagnetno polje:

- piramide, kristali, radiestezijsko kodirani predmeti,
- ergonomsko oblikovani delovni stoli,

- akupunktura in bioenergetska terapija,
- naravno zdravilno sevanje v Tunjicah pri Kamniku.

Poskusili bomo tudi verificirati zemljevid organov na koronah prstov v sodelovanju z zdravniki in bioenergetiki.

Poglavje 2

Tehnično poročilo

2.1 Opis naloge

Programski paket GDVAura (*Gas Discharge Visualisation Technique*, *Kirlian Aura*) podjetja Kirlionics Technologies International omogoča kreiranje avre človeškega telesa. Vhod v program so slike GDV vseh prstov na rokah, ki so zajete z napravo Crown-TV. Procesiranje temelji na diagnostični mapi, ki podaja korelacijo med posameznimi sektorji prstov in posameznimi organi in sistemi človeškega telesa!

Slike avre vsebujejo nekaj komponent človeškega bioelektomagnetnega polja [4]. Slika je povezana s psiho-fizičnim stanjem osebe: ponovljiva v primerih stabilnega stanja osebe, spreminja pa se med terapijami in drugimi pomembnimi vplivi [4].

Moja naloga je rekonstruirati omenjen programski paket. Prvo vprašanje, ki se na tem mestu poraja, se glasi: Zakaj je rekonstrukcija sploh potrebna? Zaradi preverjanja delovanja potrebujemo kodo programskega paketa, saj sicer ne moremo dodajati novega znanja! Ker v podjetju, kjer programski paket izdelujejo, niso bili zainteresirani za sodelovanje, nam ni preostalo drugega, kot da se lotimo pisanja programa sami! Moja naloga je narediti prototip s povdarkom na naslednjih točkah:

- barvna transformacija sivinskih slik koron (avre) prstov
- prileganje elips na slikah koron prstov
- iskanje ustrezne transformacije iz koron prstov na korono celega telesa
- uporabniški vmesnik.

Cilj naloge je torej postaviti temelje za nadaljne raziskave.

2.1.1 Osnovni problem rekonstrukcije

Na žalost postopki, ki jih uporablja paket GDVAura, niso nikjer opisani; vsaj javno niso dostopni! To pomeni, da so vsi algoritmi bili razviti na podlagi testiranja t.i. črne škatle. Testiranje črne škatle lahko na kratko opišemo z naslednjimi točkami:

1. pripravi takšne vhodne podatke, iz katerih boš lahko sklepal o delovanju algoritma
2. tako pripravljene podatke posreduj na vhod programu, ki velja za črno škatlo
3. na podlagi pričakovanj in dejanskih rezultatov sklepaj o delovanju algoritma
4. sklepe preverjaj tudi na drugih množicah vhodnih podatkov.

2.2 Opis postopkov

Po analogiji z imenom GDVAura sem moj program poimenoval GDVAuraSlo (Slo - slovenska različica), zato bom to ime uporabljal pri nadaljnem opisovanju.

2.2.1 Barvna transformacija

Program na vhodu zahteva 10 slik:

- **1L.bmp** - levi palec
- **2L.bmp** - levi kazalec
- **3L.bmp** - levi sredinec
- **4L.bmp** - levi prstanec
- **5L.bmp** - levi mezinec
- **1R.bmp** - desni palec
- **2R.bmp** - desni kazalec
- **3R.bmp** - desni sredinec
- **4R.bmp** - desni prstanec
- **5R.bmp** - desni mezinec.

Vse slike so sivinske. Njihova velikost mora biti 320×240 slikovnih elementov[1]!

Če želimo narediti barvno transformacijo sivinske slike, moramo najprej določiti paleto, ki bo služila kot osnova transformacije. Ob testiranju programa GDVAura in GDVPrint[2] sem prišel do sklepa, da programa ne uporabljata iste palete; paleti vsebujeta iste barve, do razlik pa prihaja med intervali, ki definirajo razpon posamezne barve.

Tabela 2.1 podaja definicijo palete, ki jo uporablja program GDVAuraSlo. Paleta je približek tiste, ki jo uporablja program GDVAura. Da uporabim to paleto sem se odločil iz dveh razlogov:

1. uporablja jo program, ki ga moram rekonstruirati in
2. velikosti intervalov so si podobne bolj kot pri paleti programa GDVPrint; razlog AD-HOC narave!

In zakaj sem napisal približek? Program GDVAura namreč ne shranjuje (vmesnih) rezultatov na disk, zato sem lahko velikosti intervalov le ocenil s pomočjo zajemanja slike aktivnega okna!

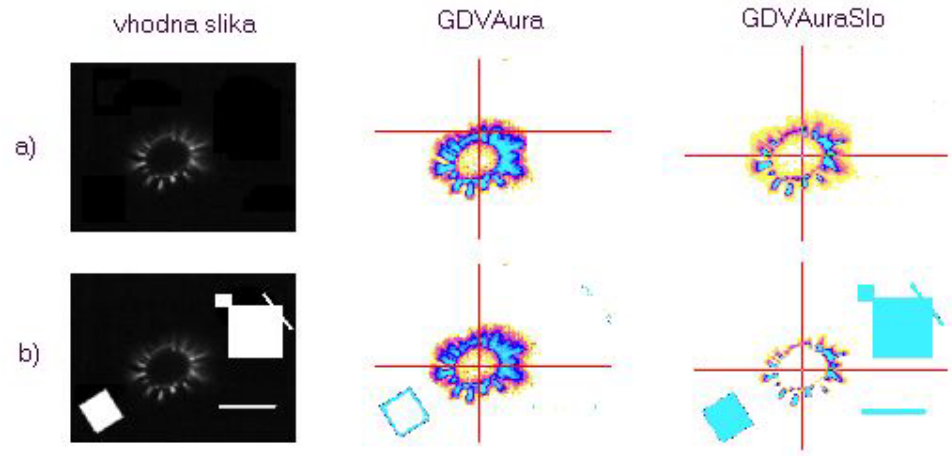
Na tem mestu bi opozoril tudi na to, da se je program GDVAura obnašal čudno ob nekaterih sintetičnih slikah!

Zapišimo sedaj algoritem barvne transformacije[3]:

[1] Ta velikost slike je izbrana zato, ker jo v praksi največkrat uporabljamo pri zajemu potrebnih informacij.

[2] Zelo podoben program, namenjen procesiranju posameznih slik GDV.

[3] V poročilu so opisani le končni algoritmi, torej tisti, ki so dali rezultate najbolj podobne rezultatom programa GDVAura! Razvitih in testiranih je bilo seveda več podobnih algoritmov.



Slika 2.1: Primer čudnega obnašanja programa GDVAura: a) originalna slika, b) originalna slika z dodatki

1. Vhodno sliko najprej posvetlimo za faktor 3.

2. Po formuli
$$prag = \text{zaokroži}(povprečna\_sivina\_slike * 2) \tag{2.1}$$
izračunamo prag.

3. Vse slikovne elemente, katerih vrednost (sivina) je v intervalu [0,70%∗*prag*][4], preslikamo v belo barvo. Za tiste, katerih vrednost (sivina) je v intervalu (70%∗*prag*,255], pa najprej izvršimo preslikavo palete v ta interval, nato pa preslikavo vrednosti posameznega slikovnega elementa v ustrezno barvo palete.

Slika 2.2 podaja primerjavo med rezultati barvne transformacije programa GDVAura in programa GDVAuraSlo.

2.2.2 Iskanje središča

Za potrebe iskanja središča moramo najprej binarizirati sliko. Zapišimo algoritem za binarizacijo vhodne slike:

1. V vhodni sliki primerjamo vsak slikovni element i z vrednostjo spremenljivke *prag*; glej enačbo (2.1). V primeru, da velja $i \geq prag$, posvetlimo vse točke okoli točke i, sicer pa zmanjšamo intenziteto točke i. Rezultate zapisujemo v novo sliko! Tako dobimo bolj kompaktno obliko korone.

2. Izvršimo čisto binarizacijo; črni elementi predstavljajo ozadje.

[4]70% uporabljam zaradi večje podobnosti rezultatov z rezultati programa GDVAura, 100% pa uporabljam pri binarizaciji vhodne slike, kar potrebujemo pri računanju središča!

SIVINA	R	G	B
0..5	255	255	255
6..30	249	249	88
31..55	249	214	135
56..82	243	145	129
83..105	250	88	144
106..132	189	88	208
133..157	135	0	213
158..182	6	11	255
183..207	34	132	255
208..232	62	198	255
233..255	62	241	255

Preglednica 2.1: Definicijo palete, ki jo uporablja program GDVAuraSlo

3. Z upoštevanjem okolice belih elementov binarizirane slike izločimo šumne točke in povdarimo močne bele regije; dodatno povdarjanje kompaktnosti korone.

Sedaj poiščemo središče; pravzaprav bi bilo pravilneje, če bi rekli težišče: povprečimo koordinati (x, y) vseh belih elementov!

Postopek iskanja središča je je ilustriran na sliki 2.3.

2.2.3 Prileganje elips

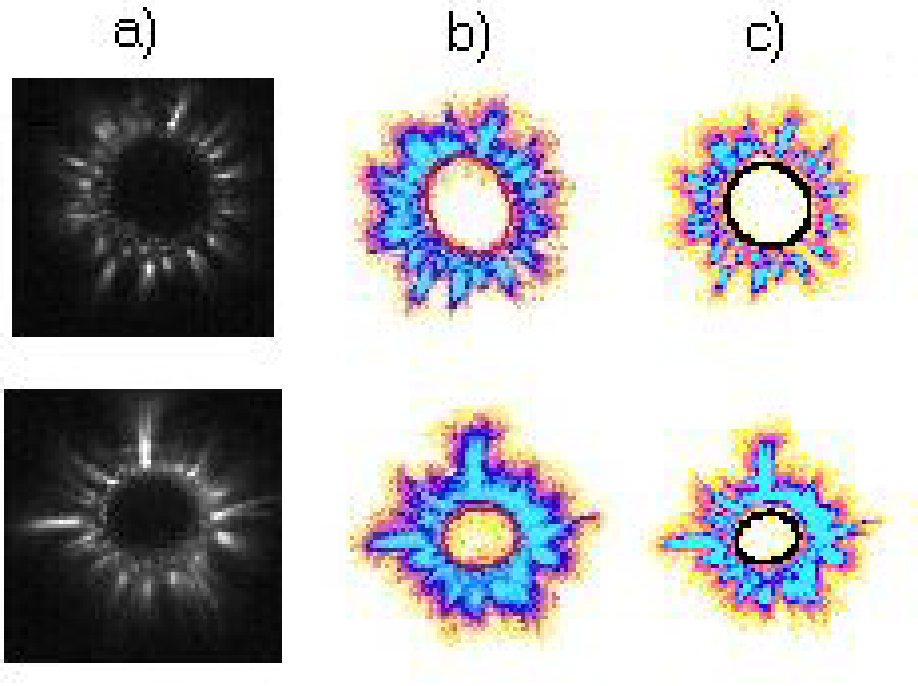
Prilagajanje elips temelji na algoritmu M. Piluja in sodelavcev [8]. Algoritem na vhodu zahteva množico točk, za katere predpostavlja, da pripadajo elipsi. To množico točk moramo zato zagotoviti v predprocesiranju:

1. izvršimo binarizacijo vhodne sivinske slike avre prsta,
2. izračunamo središče iz binarizirane slike,
3. radialno poiščemo rob in
4. sestavimo množico točk, ki naj bi pripadale konturi (elipsi) prsta.

Izhod algoritma je množica točk, ki pripadajo najbolje prilegajoči se elipsi.

Za prvi dve točki smo algoritem že podali, zato sedaj zapišimo le še preostala algoritma. Algoritem za radialno iskanje robov:

1. Izračunamo kot ϕ med središčem in trenutnim slikovnim elementom i.



Slika 2.2: Primerjava med rezultati barvne transformacije programa GDVAura in programa GDVAuraSlo: a) vhodna slika, b) GDVAura, c) GDVAuraSlo

2. Izračunamo vrednost sovpadajočega slikovnega elementa v sliki z robom; glede na kvadrant in nato še glede na razmerje med dx in dy. Primer: 1. kvadrant, $dy > dx \Rightarrow$

$$rob[i] \;=\; zaokroži(\;\; -bin\_slika[i] + \sin^2\phi \cdot bin\_slika[i - širina\_slike] + \\ + \cos^2\phi \cdot bin\_slika[i - širina\_slike + 1] \;\;). \tag{2.2}$$

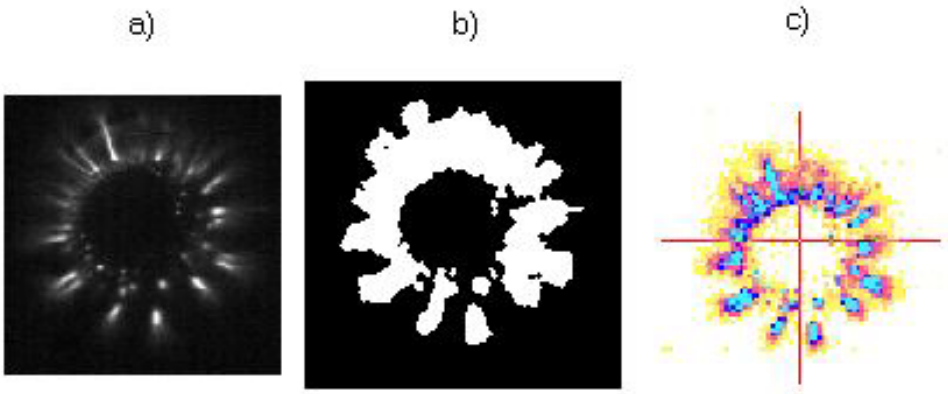
3. Obdržimo le slikovne elemente s sivino nad določenim pragom.

4. Postopek ponavljamo za vse slikovne elemente!

In algoritem za iskanje točk:

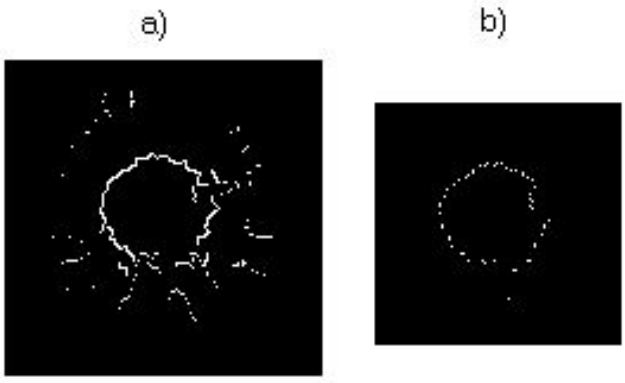
1. Za vsako robno točko si zapomnemo:
   - koordinati x in y
   - kot med središčem in robno točko
   - razdaljo med središčem in robno točko.

2. Sortiramo po kotu.



Slika 2.3: Postopek iskanja središča: a) vhodna slika, b) binarizirana slika, c) izračunano središče

3. Sliko razdelimo na segmente iz središča pod kotom ϕ (na primer $\phi = 6^o$).
4. Znotraj vsakega segmenta najdi točko z minimalno razdaljo od središča.
5. Najdeno točko dodamo v množico.



Slika 2.4: Postopek iskanja množice točk, ki naj bi pripadale konturi (elipsi) prsta: a) najdeni robovi, b) množica točk (slika 2.4 je logično nadaljevanje slike 2.3)

Slika 2.4 ilustrira postopek iskanja množice točk.

Preden zapišemo algoritem za prileganje elips pa naj opozorim še na dejstvo, da program GDVAuraSlo popravi središče glede na dobljeno elipso!

2.2.3.1 Algoritem za prileganje elips

Najprej namenimo nekaj besed problemu prileganje modelov (ang. *model fitting*). Predvidevajmo, da vemo, da nekateri slikovni elementi ležijo na premici. Zaradi

predstavitve slike s slikovnimi elementi in napak, ki so prisotne zaradi načina zajema slike in postopka iskanja robov, ne obstaja premica, ki gre skozi vse točke. Torej moramo poiskati najboljši kompromis med danimi točkami in idealno premico. Zapišimo enačbo premice $\mathbf{a}^{\mathrm{T}}\mathbf{x}$=ax+by+c=0 in poiščimo vektor parametrov $\mathbf{a}_0$, ki opisuje premico, ki poteka kar se da blizu vsaki točki. Vektor $\mathbf{a}_0$ izračunamo s pomočjo *funkcije razdalje* D med premico in množico slikovnih elementov (robnih točk) tako, da poiščemo minimalno vrednost D pri vseh možnih vrednostih vektorja $\mathbf{a}$. Največkrat je D kvadrirana razdalja, iskanje vektorja $\mathbf{a}_0$ pa tako predstavlja iskanje rešitve problema najmanjših kvadratov.

Veliko objektov je v dvorazsežnem prostoru zgrajenih iz krožnih delov, ki so na intenzitetnih (sivinskih) slikah skoraj vedno predstavljeni z elipsami. Prav zaradi tega so algoritmi za iskanje elips zelo uporabno orodje v računalniškem vidu. Problem, ki ga želimo rešiti, lahko v splošnem podamo kot:

Prileganje elips
Naj bodo $\mathbf{p}_1$,...,$\mathbf{p}_N$ *množica robnih elemetov (*$\mathbf{p}_i = [x_i, y_i]^{\mathrm{T}}$*),* $\mathbf{x}$= $[x^2, 2xy, y^2, 2x, 2y, 1]^{\mathrm{T}}$, $\mathbf{p}$= $[x, y]^{\mathrm{T}}$ *in*

$$f(\mathbf{p}, \mathbf{a}) = \mathbf{x}^{\mathrm{T}}\mathbf{a} = ax^2 + 2bxy + cy^2 + 2dx + 2ey + f = 0$$

implicitno zapisana enačba elipse[5]*, določena z vektorjem parametrov* $\mathbf{a}$= $[a, b, c, d, e, f]^{\mathrm{T}}$.

Poišči takšen vektor parametrov $\mathbf{a}_0$*, ki bo najboljše opisoval elipso skozi vse podane robne točke v smislu najmanjših kvadratov:*

$$\min_{\mathbf{a}} \sum_{i=1}^{N} [D(\mathbf{p}_i, \mathbf{a})]^2, \tag{2.3}$$

kjer je $D(\mathbf{p}_i, \mathbf{a})$ *primerna funkcija razdalje.*

In kakšna je primerna funkcija razdalje? Na to vprašanje lahko v osnovi podamo dva odgovora: Evklidova razdalja in algebraična razdalja [7, 10]!

M. Pilu s sodelavci uporablja algebraično razdaljo, saj Evklidova razdalja zahteva aproksimaciji in numerično optimizacijo, algebraična razdalja pa predstavlja približno razdaljo, ki nas pripelje do rešitve brez zahtev po nadaljnih aproksimacijah.

Definicija algebraične razdalje
Algebraična razdalja točke $\mathbf{p}$ *od krivulje* $f(\mathbf{p}, \mathbf{a}) = 0$ *je enostavno kar* $|f(\mathbf{p}, \mathbf{a})|$.

Algebraična razdalja je drugačna od prave geometrijske razdalje med krivuljo in točko. Torej smo na tem mestu vpeljali aproksimacijo, ki pa je edina na katero bomo naleteli, saj z uvedbo algebraične razdalje spremenimo problem (2.3) v linearni problem, ki ga lahko rešimo brez nadaljnih aproksimacij.

[5]Pravzaprav enačba za $f(\mathbf{p}, \mathbf{a})$ v resnici predstavlja splošno enačbo krivulj drugega reda (stožernic).

Spremenjena enačba problema (2.3) ima sedaj naslednjo obliko:

$$\min_{\mathbf{a}} \sum_{i=1}^{N} |f(\mathbf{p}_i, \mathbf{a})|^2 \quad \text{oziroma} \quad \min_{\mathbf{a}} \sum_{i=1}^{N} |\mathbf{x}_i^{\mathrm{T}}\mathbf{a}|^2. \tag{2.4}$$

Da se izognemo trivialni rešitvi $\mathbf{a}$=0, moramo $\mathbf{a}$ omejiti. Izberemo takšno omejitev, ki prisili rešitev, da opisuje elipso:

$$D = b^2 - 4ac = \mathbf{a}^{\mathrm{T}}\mathbf{C}\mathbf{a} = -1, \tag{2.5}$$

$$\mathbf{C} = \begin{bmatrix} 0 & 0 & -2 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ -2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}. \tag{2.6}$$

Opazimo lahko, da lahko na zgornjo omejitev gledamo kot na normalizirano različico splošne omejitve $D = b^2 - 4ac < 0$ (diskriminanta mora biti negativna), ki zagotavlja elipso.

Sedaj zapišimo enačbo (2.4) kot

$$\min_{\mathbf{a}} \|\mathbf{a}^{\mathrm{T}}\mathbf{X}^{\mathrm{T}}\mathbf{X}\mathbf{a}\| \quad \text{oziroma} \quad \min_{\mathbf{a}} \|\mathbf{a}^{\mathrm{T}}\mathbf{S}\mathbf{a}\|, \tag{2.7}$$

$$\mathbf{X} = \begin{bmatrix} x_1^2 & 2x_1y_1 & y_1^2 & 2x_1 & 2y_1 & 1 \\ x_2^2 & 2x_2y_2 & y_2^2 & 2x_2 & 2y_2 & 1 \\ & & \cdots & & & \\ x_N^2 & 2x_Ny_N & y_N^2 & 2x_N & 2y_N & 1 \end{bmatrix}. \tag{2.8}$$

Po terminologiji metode najmanjših kvadratov z omejitvami, $\mathbf{X}$ poimenujmo matrika načrta, $\mathbf{S}$=$\mathbf{X}^{\mathrm{T}}\mathbf{X}$ razširjena matrika in $\mathbf{C}$ matrika omejitve. Uporabimo Lagrangeov obrazec [7, 10], ki nam problem (2.7) prevede na

$$\mathbf{S}\mathbf{a} = \lambda\mathbf{C}\mathbf{a}. \tag{2.9}$$

Prišli smo do t.i. posplošenega problema lastne vrednosti. Dokažemo lahko, da je rešitev $\mathbf{a}_0$ lastni vektor, ki ustreza edini negativni lastni vrednosti:

Dokaz
Če par[6] $(\lambda_i, \mathbf{u}_i)$ predstavljata rešitev enačbe (2.9), potem jo predstavljata tudi par $(\lambda_i, \mu\mathbf{u}_i)$ za vsak μ, s pomočjo enačbe (2.5) pa lahko najdemo vrednost μ_i iz $\mu_i^2\mathbf{u}_i^{\mathrm{T}}\mathbf{C}\mathbf{u}_i = -1$:

$$\mu_i = \sqrt{\frac{-1}{\mathbf{u}_i^{\mathrm{T}}\mathbf{C}\mathbf{u}_i}} = \sqrt{\frac{-\lambda_i}{\mathbf{u}_i^{\mathrm{T}}\mathbf{S}\mathbf{u}_i}}. \tag{2.10}$$

[6] λ_i je lastna vrednost, $\mathbf{u}_i$ pa lastni vektor.

Ob upoštevanju omejitve podane v enačbi (2.5), $\mathbf{a}_0 = \mu_i \mathbf{u}_i$ predstavlja rešitev enačbe (2.9). V splošnem lahko imamo do šest realnih rešitev (šest parov $(\lambda_i, \mathbf{u}_i)$). Vsaka rešitev predstavlja lokalni minimum, če je ulomek pod korenom enačbe (2.10) pozitiven. V splošnem je $\mathbf{S}$ pozitivno definitna, torej je $\mathbf{u}_i^{\mathrm{T}}\mathbf{C}\mathbf{u}_i$ vedno pozitivna vrednost za vsak $\mathbf{u}_i$. Kvadratni koren tako obstaja le, če $\lambda_i < 0$, kar pomeni, da mora imeti rešitev enačbe (2.9) negativno lastno vrednost. Matrika omejitve (2.6) ima le eno negativno lastno vrednost, kar pomeni, da imamo le eno samo rešitev.

□

Zapišimo algoritem:

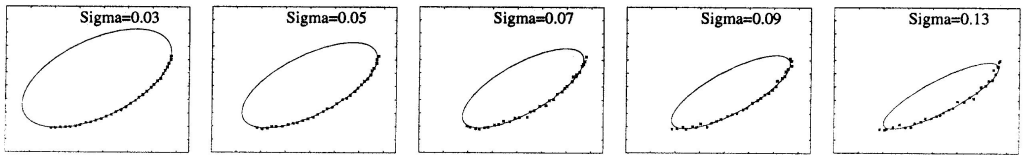
Algoritem za iskanje elips s pomočjo algebraične razdalje

Vhod naj bo množica N slikovnih točk $\mathbf{p}_1$,...,$\mathbf{p}_N$. Uporabimo označbe podane v opisu problema iskanja elips.

1. Sestavi matriko načrta $\mathbf{X}$, kot je to zapisano v (2.8).
2. Sestavi razširjeno matriko $\mathbf{S}$=$\mathbf{X}^{\mathrm{T}}\mathbf{X}$.
3. Sestavi matriko omejitve $\mathbf{C}$, kot je to zapisano v (2.6).
4. Z numeričnim postopkom izračunaj lastne vrednosti posplošenega problema lastne vrednosti. Edino negativno lastno vrednost označimo z λ_n .

Izhod je vektor $\mathbf{a}_0$, ki opisuje najbolje prilegajočo se elipso, dobljeno iz lastnega vektorja povezanega z lastno vrednostjo λ_n.

Slika 2.5 prikazuje rešitve algoritma za iskanje elips s pomočjo algebraične razdalje ob naraščajoči količini Gaussovega šuma.

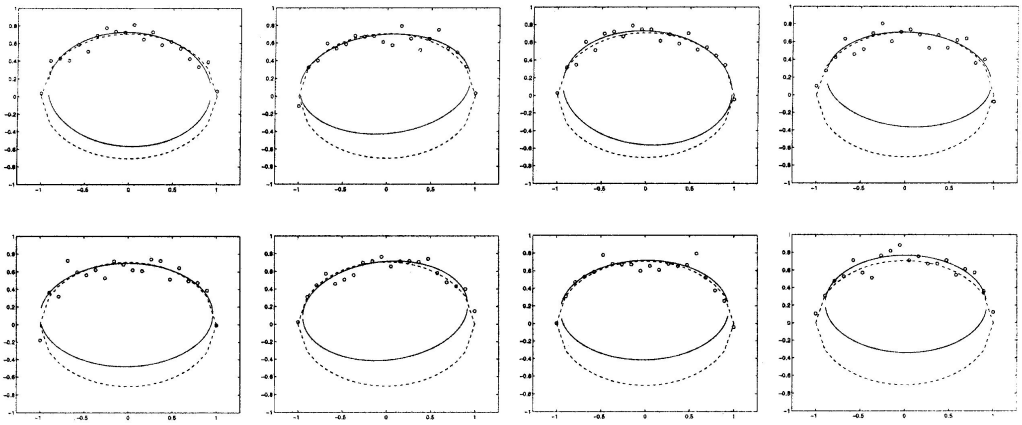


Slika 2.5: Rešitve algoritma za iskanje elips s pomočjo algebraične razdalje ob naraščajoči količini Gaussovega šuma; slike od leve proti desni vsebujejo od 3% do 20% šuma v podatkih (Sigma (σ) predstavlja standardno odstopanje

Zgornji algoritem kaže nagnjenost k rešitvam majhne ekscentričnosti[7]. To last-

[7]Kaj je ekscentričnost objekta? Objekt nima srednje točke v svojem središču; izsredinjenost. Z geometrijskega stališča s pojmom linearna ekscentričnost elipse podajamo oddaljenost gorišča od središča.

nost imajo vse metode, ki uporabljajo algebraično razdaljo. Laično povedano to pomeni, da ima algoritem raje debele elipse kot suhe elipse, kar je lepo razvidno iz slike 2.6. Ponazorimo to trditev z geometrijsko interpretacijo algebraične razdalje:



Slika 2.6: Ilustracija nagnjenosti algoritma za iskanje elips s pomočjo algebraične razdalje k rešitvam majhne ekscentričnosti. Algoritem je na vhodu dobil 20 točk, ki so opisovale polovico elipse in so bile uniformno oddaljene med seboj glede na os x, hkrati pa so vsebovale različno stopnjo Gaussovega šuma ob konstantnem standardnem odstopanju (σ=0.08, kar je približno 10% velikosti manjše polosi). Opazimo lahko, da je najdena rešitev (polna krivulja) vedno "debelejša" od prave rešitve (črtkana krivulja)

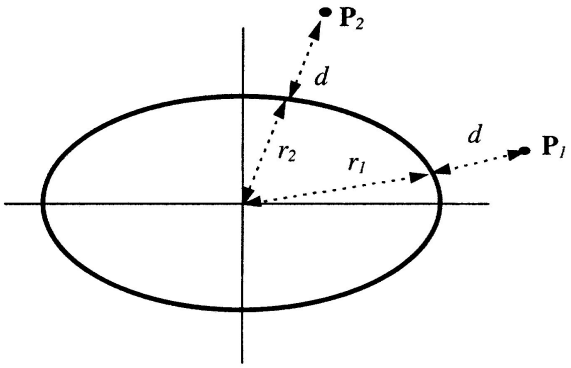
Geometrijska interpretacija algebraične razdalje
Recimo, da točka $\mathbf{p}_i$ *ne leži na elipsi* $f(\mathbf{p}, \mathbf{a}) = 0$. *Algebraična razdalja* $f(\mathbf{p}_i, \mathbf{a})$ *je proporcialna vrednosti*

$$Q = 1 - \left[\frac{r^2}{(r+d)^2}\right],$$

kjer je r *razdalja od središča elipse do elipse po premici, ki gre skozi točko* $\mathbf{p}_i$, d *pa je razdalja od elipse do točke* $\mathbf{p}_i$ *po isti premici (slika 2.7).*[8]

Za konstanten, opredeljen d je Q maksimalen v presečišču elipse s svojo manjšo osjo (podobno velja za točko $\mathbf{P}_2$ na sliki 2.7) in minimalen v presečišču elipse s svojo večjo osjo (podobno velja za točko $\mathbf{P}_1$ na sliki 2.7). Torej algebraična razdalja poda maksimalne vrednosti opazovanim točkam okoli položnih delov elipse, minimalne vrednosti pa opazovanim točkam okoli najbolj ukrivljenega dela elipse. Kot posledica, algoritem, ki temelji na geometrijski interpretaciji algebraične razdalje Q, verjame, da je večina podanih točk zgoščenih okoli položnega dela elipse. To se odraža v rešitvi tako, da le-ta opisuje "debelejšo" elipso.

[8]Interpretacija velja za vse krivulje drugega reda (stožernice). Za hiperbolo je središče presečišče asimptot, za parabolo pa je središče v neskončnosti.



Slika 2.7: Ilustracija razdalj r in d pri geometrijski interpretaciji algebraične razdalje Q. Pri enakosti razdalj d, je vrednost Q večja za točko $\mathbf{P}_2$ kot za točko $\mathbf{P}_1$

2.2.3.2 Interaktivno popravljanje lokacije, velikosti in zasuka elipse

Te akcije dosežemo po formuli Afine transformacije:

$$\begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = \mathbf{A} \cdot \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} + \mathbf{B},$$

kjer je par (x_1, y_1) koordinata točke v stari sliki, par (x_2, y_2) koordinata točke v novi sliki, $\mathbf{A}$ in $\mathbf{B}$ pa matriki, ki določata akcijo:

- Zasuk $\Rightarrow$

  $$\mathbf{A} = \begin{bmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{bmatrix}, \ \mathbf{B} = \begin{bmatrix} 0 \\ 0 \end{bmatrix},$$

  kjer je ϕ kot zasuka.

- Sprememba lokacije $\Rightarrow$

  $$\mathbf{A} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \ \mathbf{B} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix},$$

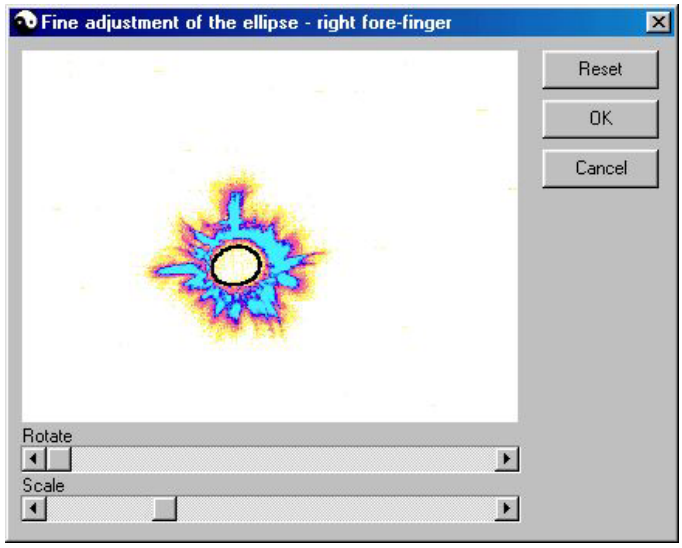
  kjer je b_1 sprememba v smeri osi x in b_2 sprememba v smeri osi y.

- Sprememba velikosti $\Rightarrow$

  $$\mathbf{A} = \begin{bmatrix} A_{11} & 0 \\ 0 & A_{22} \end{bmatrix}, \ \mathbf{B} = \begin{bmatrix} 0 \\ 0 \end{bmatrix},$$

  kjer sta A_{11} in A_{22} faktorja spremembe velikosti v smeri x in y.

Na sliki 2.8 je prikazan uporabniški vmesnik, ki omogoča izvrševanje akcij zasuka in spremembe velikosti. Uporabniški vmesnik, ki omogoča izvršitev akcije spremembe lokacije si lahko ogledate na sliki 2.3 (primer c).



Slika 2.8: Uporabniški vmesnik, ki omogoča izvrševanje akcij zasuka in spremembe velikosti

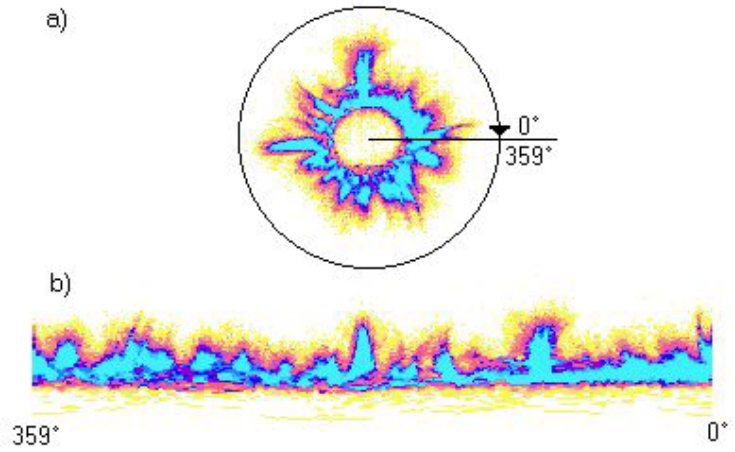
2.2.4 Transformacija na telo

In zakaj sploh potrebujemo elipso? Če hočemo narediti preslikavo sektorja na telo, moramo najprej vedeti, kje se začne oziroma se naj bi začela korona prsta, saj ni nujno, da korona obstaja na vseh delih prstnega obsega! S pomočjo najdene elipse tako generiramo normalizirano panoramsko sliko korone prsta. Normalizirano pomeni, da naredimo takšno preslikavo, da se najdena elipsa preslika v premico v panoramski sliki! Zapišimo sedaj algoritem generiranja panoramske slike:

1. Izhodišče je središče elipse in sama elipsa.
2. Sliko korone razrežemo od središča proti desni strani.
3. V negativni matematični smeri naredimo za vsako stopinjo ustrezno preslikavo slikovnih elementov iz koordinatnega sistema xy v koordinatni sistem φd, pri čemer je φ kot pogleda in d razdalja od središča.

Slika 2.9 podaja postopek generiranja panoramske slike.

In sedaj smo pripravljeni na zadnji korak, preslikavo posameznih sektorjev na telo! Informacija o razrezu na sektorje je spravljena v datoteki KOROTKOV.INI (dodatek A). Za to transformacijo je bilo potrebno razviti dva različna algoritma: enega za preslikavo na desno in levo stran telesa in drugega za preslikavo na zgornji del glave! Razvita nista splošna algoritma, bistvena razloga za to pa sta posebnosti preslikave: prehodi skozi 0^o (panoramska slika) in računanje unij med, lahko le delno, prekrivajočimi se sektorji; tudi oboje hkrati!



Slika 2.9: Postopek generiranja panoramske slike: a) vhodna slika po izvršeni barvni transformaciji z nakazanim principom generiranja panoramske slike, b) panoramska slika

Rezultat testiranja transformacije na telo programskega paketa GDVAura je razviden iz slike 2.10[9]!

Zapišimo najprej primer psevdo kode algoritma za preslikavo na desno in levo stran telesa; za primer vzemimo sektor 3R/6 (šesti sektor sredinca desne roke):

```
//Shrinking the sector!
//sectorwidth - width of the sector in the panorama image
//numberofcoordinates - width of the sector in the body (result) image
//a, b, c, y - shrinking parameters
//v - sector start position in the panorama image
a=int (sectorwidth/numberofcoordinates);
b=sectorwidth-a*numberofcoordinates;
if (b==0) c=numberofcoordinates+1;
else c=int (numberofcoordinates/b+0.5);
v=sectorendvector;
y=1;
for (i=1;i<=numberofcoordinates;i++)
{
    //Take 'a' columns (panorama image) and calculate the average row (result), but
    //for every 'c'th row take 'a+1' columns!
    for (w=0;w<panoramaimageheight;w++)
    {
       if (y<c)
       {
            initialize(); //Set the value of the coordinate in result image to zero!
            //At the end we have to take all the remaining columns!
            if ((i==numberofcoordinates) && (w==0))
                newa();
            for (e=0;e<a;e++)
                add(); //Add the value of the coordinate in panorama image to
```

[9]Tukaj bi rad opozoril na dvom, ki se mi je porajal ob opazovanju slike. Kot primer vzemimo sektor 4R/4, ki je povezan z vranico. Vranica se namreč nahaja na zgornji polovici leve strani trupa, sektor pa se preslika na sredino desnega stegna! Ker nimam znanja s tega področja, bom odločitev o pravilnosti oziroma nepravilnosti prepustil strokovnjakom, osebno pa se mi preslikava trenutno ne zdi najbolj razumljiva.

```
                    //the value of the coordinate in result image!
               average(); //Calculate the average value of the coordinate in result image!
               CorrespondingColor(); //Find the nearest corresponding color in the palette!
          }
          else
          {
               initialize(); //Set the value of the coordinate in result image to zero!
               //At the end we have to take all the remaining columns!
               if ((i==numberofcoordinates) && (w==0))
                   newa();
               for (e=0;e<a+1;e++)
                   add(); //Add the value of the coordinate in panorama image to
                          //the value of the coordinate in result image!
               average(); //Calculate the average value of the coordinate in result image!
               CorrespondingColor(); //Find the nearest corresponding color in the palette!
          }
     }
     if (y==c)
     {
         y=1;
         v=v+a+1;
     }
     else
     {
         y++;
         v=v+a;
     }
}
```

Oglejmo si še konkretne podatke za preslikavo tega sektorja:

- $sectorwidth = 60$
- $numberofcoordinates = 23$
- $a = 2$, $b = 14$, $c = 2$
- $v_0 = 0$.

Zapišimo interpretacijo delovanja algoritma:

- Sektor širine 60 slikovnih elementov (60^o) v panoramski sliki stisnemo na sektor širine 23 slikovnih elemetov, kot je za sektor predvideno ob človeškem telesu (slika 2.10).
- Stiskanje poteka po naslednjem postopku: združimo dva (a) stolpca panoramske slike, nato tri, dva, tri, dva, tri, dva,..., na koncu pa združimo preostale stolpce; (2+3)*11+5=60!
- Delamo preslikavo posameznih slikovnih elemetov.

Slika 2.11 prikazuje postopek preslikave sektorja 3R/6.

Sedaj zapišimo še primer psevdo kode algoritma za preslikavo na zgornji del glave; za primer vzemimo sektor 3R/8 (osmi sektor sredinca desne roke):

```
//sectorwidth - width of the sector in the panorama image
//numberofcoordinates - number of coordinates in the body (result) image;
//                      each coordinate covers the space defined by beta
//startangle - angle defining the start position of the sector
//             according to 'the head center'
//endangle - angle defining the end position of the sector
//           according to 'the head center'
//a, alpha, beta, gamma1, gamma2 - transformation parameters
//v - sector start position in the panorama image
a=int (sectorwidth/numberofcoordinates+0.5);
alpha=endangle-startangle;
beta=alpha/numberofcoordinates;
gamma1=startangle;
v=359-sectorendvector;
for (i=1;i<=numberofcoordinates;i++)
{
    //Take 'a' columns (panorama image) and transform them into space defined
    //by beta; gamma2 (result)!
    for (j=0;j<=a;j++)
    //The range should be [0,a-1] but in this case I got little white regions
    //inside the aura, so I extended it to [0,a] and the problem is solved (almost)!
    {
        gamma2=(beta/a)*j+gamma1;
        for (w=0;w<panoramaimageheight;w++)
        {
            //For each pixel in panorama image calculate corresponding dx and dy;
            //using dx and dy with coordinate covering the space defined by beta
            //we get the corresponding coordinate in result image!
            dxresult=(int)(cos((pi*gamma2)/(180))*w+0.5);
            dyresult=(int)(sin((pi*gamma2)/(180))*w+0.5);
            //If the pixel in the result image is not white we perform an union,
            //otherwise we only assign new value to the pixel!
            if (pixelwhite())
                newvalue();
            else
            {
                union();
                CorrespondingColor(); //Find the nearest corresponding color in the palette!
            }
        }
        v++;
    }
    v--; //Since we extended the range to [0,a]!
    gamma1=gamma2;
}
```

Oglejmo si še konkretne podatke za preslikavo tega sektorja:

- $sectorwidth = 60$
- $numberofcoordinates = 20$
- $startangle = 0$
- $endangle = 45$
- $a = 3$, $alpha = 45$, $beta = 2.25$
- $v_0 = 239$.

Zapišimo interpretacijo delovanja algoritma:

- Sektor širine 60 slikovnih elementov (60^o) v panoramski sliki stisnemo na sektor podan z 20. koordinatami, ki vsaka pokriva kot *beta*, kot je za sektor predvideno na zgornjem delu glave (slika 2.10).
- Preslikava poteka po naslednjem postopku: vzemi tri (a) stolpce panoramske slike in jih preslikaj na območje podanim z ustrezno koordinato in kotom *beta* oziroma *gamma*2! Postopek ponovimo za vsako od 20. koordinat; 3*20=60!
- Delamo preslikavo posameznih slikovnih elemetov.

Slika 2.12 prikazuje postopek preslikave sektorja 3R/8.

In kako se izračunajo unije med prekrivajočimi se sektorji? V [4] (stran 214) naletimo na omembo enostavnega povprečenja, podobnost rezultatov po implementaciji pa je ta postopek tudi potrdila! Ker pa povprečenje lahko botruje barvam, ki niso v paleti, je bilo potrebno sestaviti tudi funkcijo razdalje, ki je opisovala, kako različni sta dve barvi. Na koncu se vsaki povprečeni barvi priredi barva palete, ki ji je najbližja v barvnem prostoru RGB:

$$\min_{\substack{barve\\palete}}(\sqrt{(R_{povprečena}-R_{paleta})^2+(G_{povprečena}-G_{paleta})^2+(B_{povprečena}-B_{paleta})^2}).$$

2.2.4.1 Primerjava

Slika 2.13 podaja rezultat preslikave obeh programov: GDVAura in GDVAuraSlo. Zapišimo nekaj ugotovitev:

- Rezultata sta si zelo podobna!
- Očitno program GDVAura vsebuje nekaj programskih hroščev, saj si drugače ne znam razlagati odsotnost določenih delov avre telesa, za katere vem, da sektorji, ki se preslikajo na tisti del telesa, niso prazni! Hkrati pa da program GDVAura ob preslikavi le tega sektorja drugačen (pravilen) rezultat! Zanimivo je tudi to, da program GDVAura na takšnih mestih ostro odreže avro, ob telesu pa se ne nahaja niti najmanjši zametek avre, ki bi po mojem mnenju morala biti prisotna!
- Avra obstaja okoli celega telesa, torej tudi pod nogami! Eden izmed naslednjih korakov pri gradnji paketa GDVAuraSlo bo vsekakor tudi preslikava pod nogami, ni pa mi jasno, zakaj tega niso naredili že pri paketu GDVAura! Očitno pa je, da paket GDVAura uporablja nekakšno zaključevanje avre na skrajnem spodnjem delu nog!
- Rezultat programa GDVAura ima mehkejše prehode med barvami palete v avri. V programu GDVAuraSlo bi podoben učinek lahko dosegli z uporabo filtra glajenja (po narejeni barvni transformaciji ali po narejeni transformaciji na telo), vendar osebno za to ne vidim potrebe.

2.3 O kodi

- Programski jezik: C/C++
- Programski paket: Borland C++ Builder Standard
- Opis datotek programa:
  - GDVAuraSlo.mak - projektna (osnovna) datoteka
  - GDVAuraSlo.cpp - datoteka glavnega programa
  - U1.cpp - koda funkcij osnovnega okna uporabniškega vmesnika
  - U2.cpp - koda funkcij okna za izbiranje slik, ki jih bomo procesirali
  - U3.cpp - koda funkcij okna za prikaz slik po barvni transformaciji in interaktivno popravljanje središča
  - U4.cpp - koda funkcij okna za prikaz najdenih elips in interaktivno popravljanje lokacije le-teh
  - U5.cpp - koda funkcij okna za interaktivno popravljanje zasuka in velikosti elipse
  - U6.cpp - koda funkcij okna za prikaz rezultata transformacije na telo
  - Settings.cpp - koda funkcij okna z nastavitvami
  - About.cpp - koda funkcij okna z osnovnimi informacijami o programu
  - Bg.cpp - koda funkcij okna uniformnega (sivega) ozadja
  - image.cpp - funkcije za delo s slikami
  - ellipse.cpp - funkcije za iskanje elips
  - mem.cpp - funkcije za delo s spominom
  - ustrezne *.h datoteke (ang. *header files*); opis razredov,...
  - palette.h - definicija palete
  - ostale datoteke, ki jih generira C++ Builder
- Berljivost kode:
  - razdeljenost na smiselne razrede (ang. *class*) in enote (ang. *unit*)
  - uporabljena dolga imena spremenljivk, zamiki,...
  - komentarji v angleščini (postopki, nasveti,...)
  - ...
- Datoteke potrebne za izvajanje:
  - GDVAuraSlo.exe - zagonska datoteka

  - KOROTKOV.INI, MANDEL.INI - datoteki z informacijo o razrezu na sektorje
  - body.bmp - slika telesa brez avre
  - slike koron prstov

- Vmesni rezultati:
  - barvna transformacija koron prstov; na primer:
    1L.bmp ⇒ 1LC.bmp (C - Color)
  - normalizirane panoramske slike koron prstov; na primer:
    1L.bmp ⇒ p1L.bmp (p - panorama image)
  - panoramske slike razdeljene na sektorje; na primer:
    1L.bmp ⇒ p1Lsectors.bmp

- Končen rezultat: result.bmp - slika avre na telesu

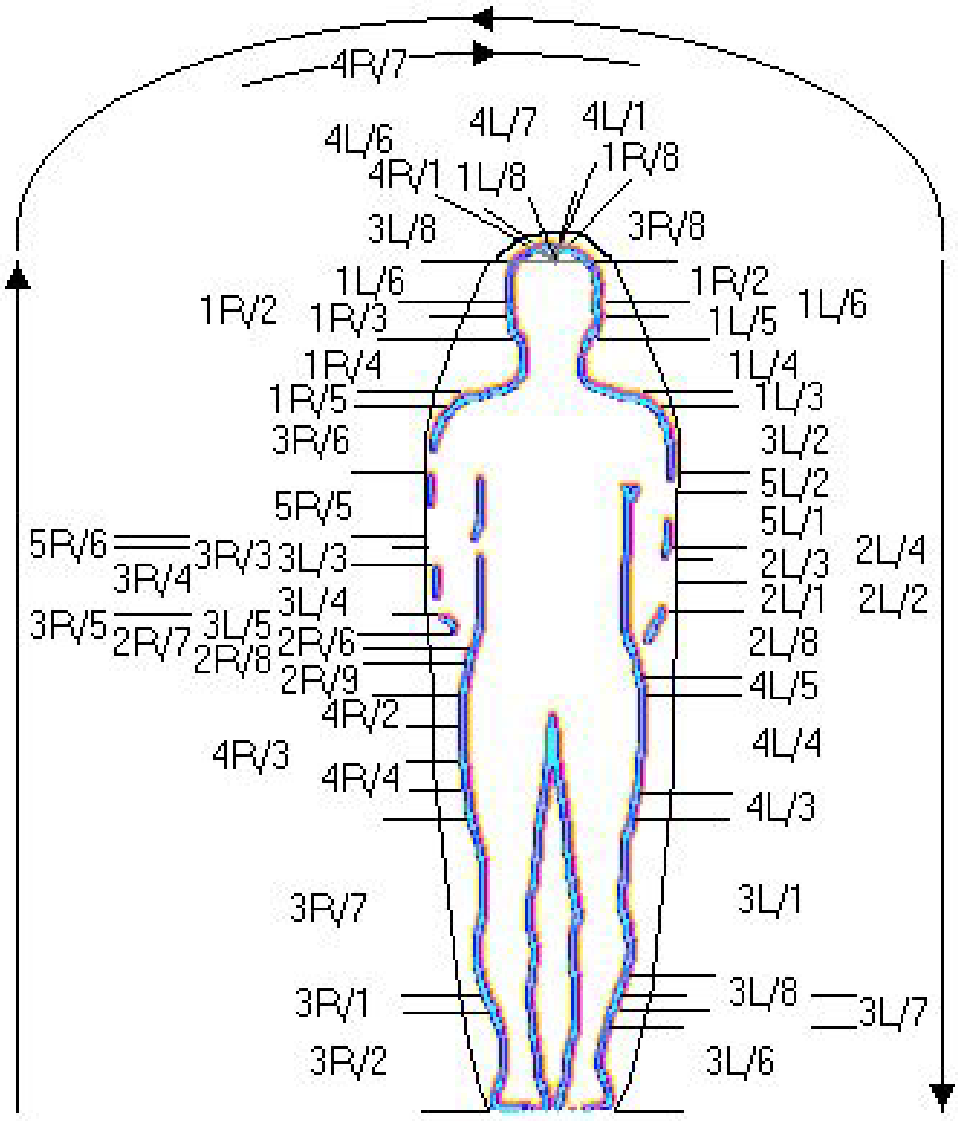
- Pomoč za programerje; obstajata dva gumba na uporabniškem vmesniku, ki sta namenjena izključno programerjem:
  - gumb <Sectorize> (sklop datotek U4) - vhodne slike razdeli na posamezne sektorje
  - gumb <Transform> (sklop datotek U3) - ob že narejenih panoramskih slikah lahko z minimalnim popravkom kode (odkomentiramo prvo for zanko v datoteki U4.cpp, v funkciji Button2Click) skočimo iz narejene barvne transformacije direktno na transformacijo na telo; znatna pospešitev ob testiranju transformacije na telo

- Silhueta, ki opisuje začetek preslikave na telo (začetne koordinate avre v sliki body.bmp; glej sliko 2.10), je opisana v datoteki transformation.dat.

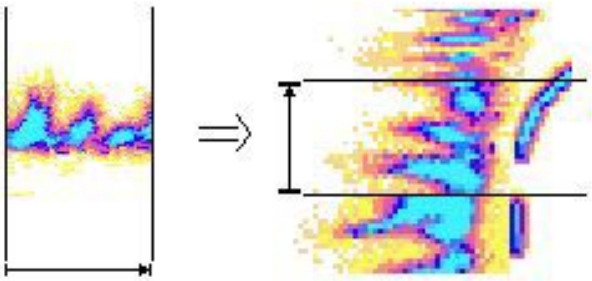
- Omejitve programa:
  - priporočena ločljivost ekrana: 800×600 slikovnih elementov
  - velikost vhodnih slik: 320×240 slikovnih elementov

- Možne izboljšave programa:
  - pospešitev delovanja programa - Program GDVAuraSlo je bil razvit kot prototip, kar pomeni, da je bil bistven cilj zagotoviti ustrezno funkcionalnost programa. Pri razvoju so bili zelo pomembni vmesni rezultati, zato program veliko dela s trdim diskom.
  - optimizacija kode
  - zaradi specifičnosti prileganja elips v programu je najverjetneje možna tudi pospešitev tega dela programa

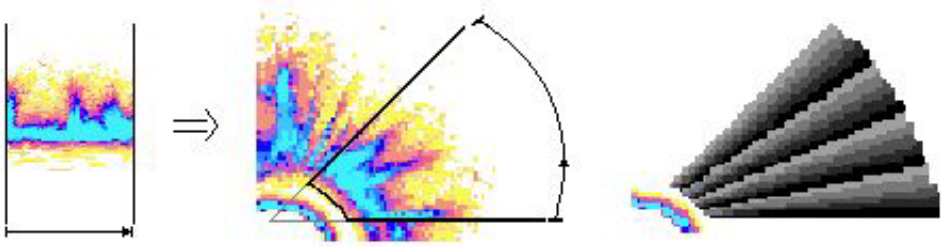
- slabost algoritma za prileganje elips - Trenutno algoritem predpostavlja, da so točke, ki naj bi pripadale elipsi, (približno) enakomerno porazdeljene po njej. Ker vemo, da lahko korona prsta vsebuje prazen prostor (luknje), bi lahko uporabili informacijo o interaktivno določenem središču kot izhodišče pri generiranju dodatnih točk. Tako bi omenjeno slabost odpravili.
- nadgradnja
- ...



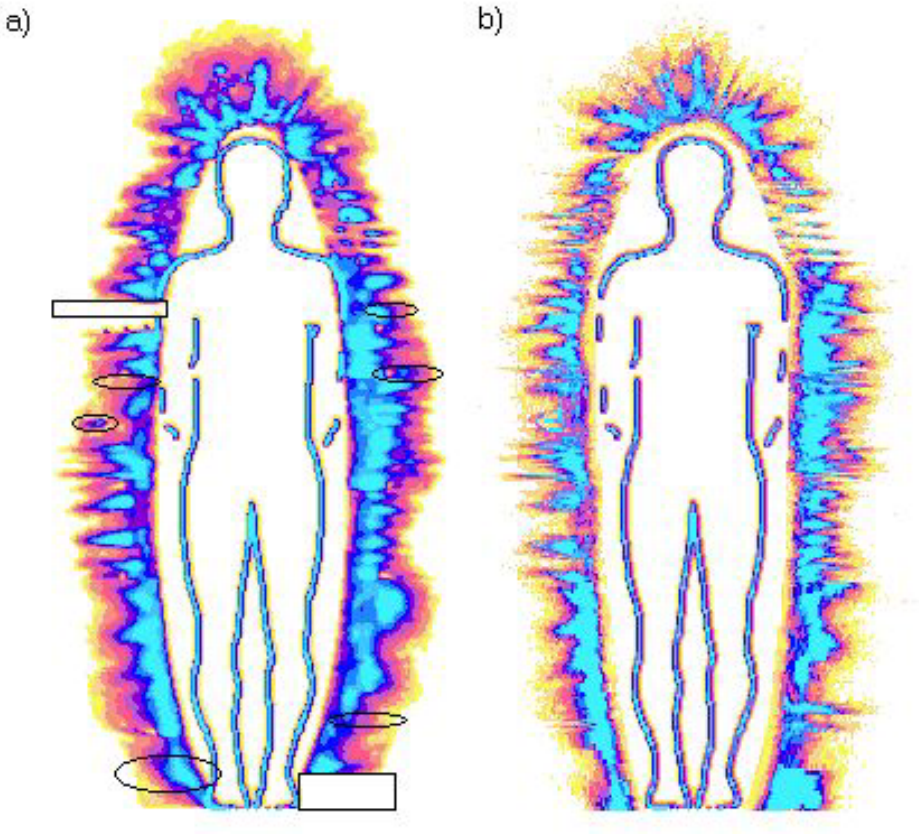
Slika 2.10: Mapa preslikave posameznih sektorjev na telo, ki jo uporablja program GDVAuraSlo. Puščice podajajo smer preslikave sektorja, ki se v panoramski sliki razteza od leve proti desni strani. Imenovanje sektorjev: XY/Z - prst X ($X \in \{1, 2, 3, 4, 5\}$; $X = 1$ ⇒palec,..., $X = 5$ ⇒mezinec), roka Y ($Y \in$ {L - leva,R - desna}) in sektor Z ($Z \in \{1, 2, ..., 9\}$); dodatek A



Slika 2.11: Preslikava sektorja 3R/6 na telo



Slika 2.12: Preslikava sektorja 3R/8 na zgornji del glave; zadnja slika podaja odvisnost med posamezno koordinato, ki opisuje sektor na glavi in kotom *beta* oziroma *gamma*2

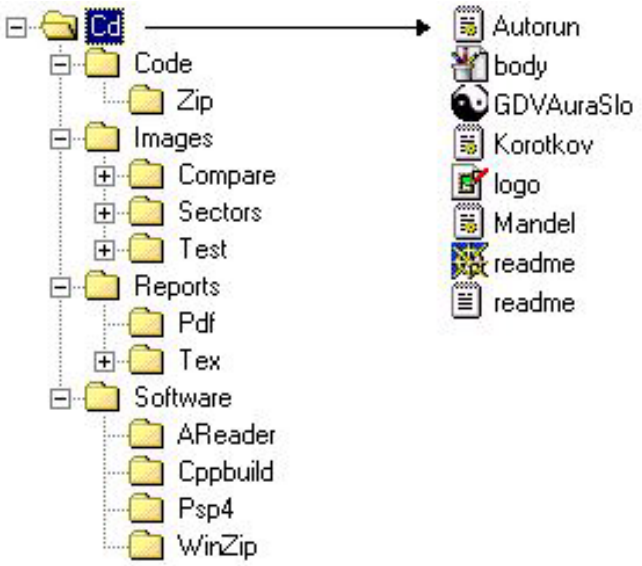


Slika 2.13: Rezultat preslikave obeh programov: a) GDVAura in b) GDVAuraSlo; na sliki a) so s pravokotniki označeni deli avre oziroma telesa, za katere predvidevam, da so posledica programskega hrošča v programu GDVAura, z elipsami pa tisti deli avre, ki se razlikujejo od rezultata programa GDVAuraSlo

Poglavje 3

Vsebina CD plošče

Slika 3.1 podaja drevesno strukturo vsebine CD plošče. Ob vstavitvi CD plošče v CD-ROM pogon, se avtomatsko zažene spletni brskalnik (če ga seveda imate nameščenega) in prikaže datoteko z osnovnimi informacijami. V primeru, da spletnega brskalnika nimate nameščenega, si oglejte datoteko readme.txt, ki se nahaja na korenu CD plošče.



Slika 3.1: Drevesna struktura vsebine CD plošče

Opis vsebine CD plošče:

- datoteke na korenu CD plošče:
  - Autorun.inf - datoteka za avtomatski zagon CD plošče
  - readme.htm - datoteka z osnovnimi informacijami v formatu HTML (zraven spada tudi datoteka logo.gif)
  - readme.txt - tekstovna datoteka z osnovnimi informacijami
  - datoteke potrebne za izvajanje programa GDVAuraSlo:
    - GDVAuraSlo.exe - zagonska datoteka
    - KOROTKOV.INI, MANDEL.INI - datoteki z informacijo o razrezu na sektorje
    - body.bmp - slika telesa brez avre
- direktorij Code:
  - koda programa GDVAuraSlo z vsemi potrebnimi datotekami
  - v direktoriju Zip se nahaja arhiv kode

- direktorij Images:
  - direktorij Test vsebuje primere slik koron prstov; namenjeno prvemu testu programa GDVAuraSlo
  - direktorij Sectors vsebuje primer slik koron prstov razdeljenih na posamezne sektorje
  - direktorij Compare vsebuje rezultate različih testov programa GDVAura in GDVAuraSlo
- direktorij Reports:
  - direktorij Pdf vsebuje poročilo k programu GDVAuraSlo v formatu PDF
  - direktorij Tex vsebuje poročilo k programu GDVAuraSlo v formatu TEX
- direktorij Software vsebuje programske pakete, ki so potrebni za pregled vsebine CD plošče oziroma za prevajanje kode:
  - direktorij AReader vsebuje program Adobe Acrobat Reader, ki omogoča pregled datotek v formatu PDF
  - direktorij Cppbuild vsebuje program Borland C++ Builder Standard, ki je služil kot implementacijsko orodje programa GDVAuraSlo
  - direktorij Psp4 vsebuje program JASC Paint Shop Pro 4, ki omogoča delo s slikami
  - direktorij WinZip vsebuje program za arhiviranje NMC WinZip

Dodatek A

Razrez na sektorje

Informacija o razrezu na sektorje je spravljena v datoteki KOROTKOV.INI:

```
[Vectors Setting]
Left hand
Thumb
vector 1=135
vector 2=170
vector 3=190
vector 4=225
vector 5=315
vector 6=350
vector 7=10
vector 8=45
Fore finger
vector 1=170
vector 2=190
vector 3=230
vector 4=280
vector 5=320
vector 6=350
vector 7=10
vector 8=40
Middle finger
vector 1=120
vector 2=180
vector 3=240
vector 4=270
vector 5=300
vector 6=350
vector 7=10
vector 8=60
Ring finger
vector 1=135
vector 2=170
vector 3=190
vector 4=225
vector 5=315
vector 6=0
vector 7=45
Little finger
vector 1=135
vector 2=180
vector 3=225
vector 4=315
```

```
vector 5=0
vector 6=45
Right hand
Thumb
vector 1=135
vector 2=170
vector 3=190
vector 4=225
vector 5=315
vector 6=350
vector 7=10
vector 8=45
Fore finger
vector 1=135
vector 2=170
vector 3=190
vector 4=230
vector 5=260
vector 6=280
vector 7=320
vector 8=350
vector 9=10
Middle finger
vector 1=120
vector 2=170
vector 3=190
vector 4=240
vector 5=270
vector 6=300
vector 7=0
vector 8=60
Ring finger
vector 1=135
vector 2=170
vector 3=225
vector 4=315
vector 5=350
vector 6=10
vector 7=45
Little finger
vector 1=135
vector 2=180
vector 3=225
```

```
vector 4=315
vector 5=0
vector 6=45
[End Vectors]


[Medical Setting]
Left hand
Thumb
sector 7=Eye
sector 8=Cerebral Zone
sector 1=Eye
sector 2=Nose,Ear
sector 3=Jaw
sector 4=Throat,Tonsils,Trachea,Tharold Gland
sector 5=Jaw
sector 6=Nose,Ear
Fore finger
sector 7=Cervical Spine
sector 8=Colons Transverse
sector 1=Descending
sector 2=Sigmoid
sector 3=Rectum
sector 4=Sacral
sector 5=Lumbar Spine
sector 6=Dorsal Spine
Middle finger
sector 7=Thorax Zone
sector 8=Head Zone
sector 1=Blood Circulation
sector 2=Heart
sector 3=Kidney
sector 4=Liver
sector 5=Abdominal Zone
sector 6=Lymph
Ring finger
sector 6=Pituitary Gland
sector 7=Pineal Gland
sector 1=Hypothalamus
sector 2=
sector 3=Spleen
sector 4=Uro-Genital System
sector 5=Endocrine System
```

```
Little finger
sector 5=Ileum
sector 6=Coronary Vessels
sector 1=Heart
sector 2=Kidney
sector 3=Respiratory System,Mammary Glands
sector 4=
Right hand
Thumb
sector 7=Eye
sector 8=Cerebrum Zone
sector 1=Eye
sector 2=Nose,Ear
sector 3=Jaw
sector 4=Throat,Tonsils,Trachea,Tharold Gland
sector 5=Jaw
sector 6=Nose,Ear
Fore finger
sector 9=Colons Transverse
sector 1=Cervical Spine
sector 2=Dorsal Spine
sector 3=Lumbar Spine
sector 4=Sacral
sector 5=Coccyx
sector 6=Cecum
sector 7=Appendix
sector 8=Ascending
Middle finger
sector 7=Blood Circulation
sector 8=Head zone
sector 1=Thorax Zone
sector 2=Lymph
sector 3=Abdominal Zone
sector 4=Liver
sector 5=Kidney
sector 6=Heart
Ring finger
sector 6=Hypothalamus
sector 7=Pineal Gland
sector 1=Pituitary Gland
sector 2=Endocrine System
sector 3=Uro-Genital System
sector 4=Spleen
```

```
sector 5=
Little finger
sector 5=Heart
sector 6=Coronar Vessels
sector 1=Duodenum
sector 2=Jejunum
sector 3=Respiratory System,Mammary Glands
sector 4=Kidney
[End Medical]
[End]
```

Literatura

[1] I. Kononenko, *Machine Learning* (in Slovene), Faculty of Computer and Information Science Publisher, Ljubljana, 1997.

[2] I. Kononenko, I. Bratko, M. Kukar, Application of machine learning to medical diagnosis, In: R. S. Michalski, I. Bratko, and M. Kubat (eds.): *Machine Learning, Data Mining and Knowledge Discovery: Methods and Applications*, John Wiley & Sons, 1998.

[3] K. Korotkov, *Light after Life: A Scientific Journey Into the Spiritual World*, Fair Lawn, USA, Backbone Publ. Comp., 1998.

[4] K. Korotkov, *Aura and Consciousness: A New Stage of Scientific Understanding*, St. Petersburg, Russia, State Editing & Publishing Unit "Kultura", 1998.

[5] A. Kraweck, *Life's Hidden Forces: A Personal Journey Into Kirlian Photography*, Edmont, Canada, Triune-Being Research Organization Ltd., 1994.

[6] T. Mitchell, *Machine Learning*, McGraw Hill, 1997.

[7] P. Peer, *Houghova transformacija in pomerjanje elips*, seminarska naloga pri predmetu Koncepti za modeliranje vizualnih informacij, Fakulteta za računalništvo in informatiko, Univerza v Ljubljani, 1999.

[8] M. Pilu, A. W. Fitzgibbon, R. B. Fisher, Direct Least Squares Fitting of Ellipses, *13th Int. Conf. on Pattern Recognition*, Volume I, Track A, Computer Vision, pp. 253-257, Vienna, Austria, 1996.

[9] V. L. Voeikov, Living blood outside an organism, *In Proc. Int. Scientific Conf. Kirlionics*, White Nights 98 (Abstracts), Federal Technical University SPIFMO, St. Petersburg, Russia, June 1998.

[10] E. Trucco, A. Verri, *Introductory Techniques for 3-D Computer Vision*, Prentice Hall, 1998.