University of Ljubljana
FRI Faculty of Computer and Information Science

HOUGHOVA TRANSFORMACIJA IN POMERJANJE ELIPS

seminarska naloga pri predmetu
KONCEPTI ZA MODELIRANJE VIZUALNIH INFORMACIJ

Avtor: Peter Peer **Mentor:** prof. dr. Franc Solina

V Velenju, 29.3.1999

Kazalo

1 Uvod

Premice in krivulje so v računalniškem vidu zelo pomembne značilke, saj opisujejo konture (obrise oziroma obrobe) objektov na slikah. V tem delu so predstavljene metode za iskanje premic in splošnih sklenjenih krivulj. Problem, ki ga pri tem želimo rešiti, lahko v splošnem podamo kot:

Iskanje premic in krivulj
Na sliki z najdenimi robovi najdi vse primere iskanih krivulj (npr. premice ali elipse) ali njenih delov (npr. daljic ali lokov elips)!

Seveda lahko iščemo premice tudi po metodi ujemanja vzorcev (ang. *template matching*). Naredimo lahko enostavno konvolucijo med sliko in množico mask (filtrov). Ta metoda pa ima dve neugodni lastnosti:

1. za pravilno lokalizacijo premice potrebujemo (pre)veliko število mask in
2. naletimo na kup standardnih problemov povezanih s samimi filtri.

Pri iskanju krivulj po tej metodi pa naletimo na še večji problem. Zato se držimo raje naslednje ideje: najprej najdemo na sliki robove, nato pa na takšni sliki poiščemo željene krivulje. Ta ideja poraja dva nova pojma:

Grupiranje
Katere (robne) točke na sliki pripadajo določeni krivulji na sliki?

Pomerjanje modelov (ang. *model fitting*)
Za podano množico (robnih) točk, ki predvidoma pripadajo neki ciljni krivulji, najdi najboljšo krivuljo, ki poteka kar se le da blizu vsaki točki iz množice.

Metode opisane v tem delu se ukvarjajo z obema problemoma (grupiranje - glej Houghova transformacija, pomerjanje modelov - glej iskanje elips), primerne pa so le za iskanje premic in enostavnih krivulj.

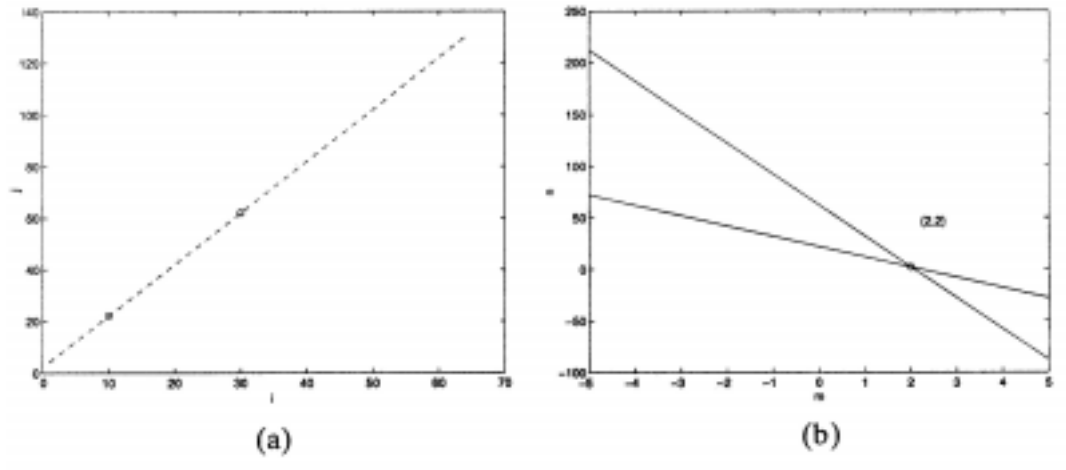
2 Houghova transformacija

Houghova transformacija je bila najprej namenjena iskanju kompleksnih vzorcev v binarnih slikah, hitro pa je postala popularna metoda za iskanje premic in krivulj. Osnovna ideja transformacije se glasi: *"Preslikaj težak problem iskanja vzorcev (iskanje primerov neke krivulje) v enostaven problem iskanja vrhov v prostoru parametrov krivulje."* Metoda kot vhod zahteva sliko z najdenimi robovi.

2.1 Iskanje premic

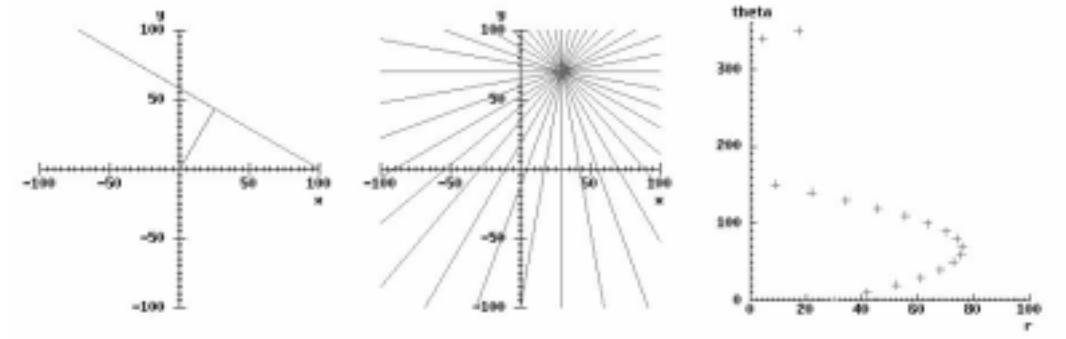
Za lažje razumevanje Houhgove transformacije si najprej oglejmo postopek iskanja premic:

1. *Preslikaj problem iskanja premic v problem iskanja presečišč premic.* Vsaka premica $y = mx + n$ je določena z dvema parametroma, parom (m, n). Torej je premica predstavljena s točko v ravnini mn. To ravnino poimenujemo parametrični prostor oziroma akumulacijsko polje. (Zakaj akumulacijsko polje, bomo videli v nadaljevanju.) Vsaka točka $\mathbf{p}= [x, y]^{\mathrm{T}}$ na sliki tako ustreza premici $n = x(-m) + y$ v parametričnem prostoru. Ob spreminjanju vrednosti spremenljivkam m in n, premica $n = x(-m) + y$ predstavlja vse možne premice, ki gredo skozi točko $\mathbf{p}$ na sliki. Torej premica, ki je definirana z N kolinearnimi slikovnimi točkami $\mathbf{p}_1$,...,$\mathbf{p}_N$, je definirana v parametričnem prostoru kot presečišče premic povezanih s temi točkami (glej sliko 1).



Slika 1: Ilustracija osnovne ideje Houghove transformacije za iskanje premic: robni točki (10,22) in (30,62) (a) se preslikata v premici $n = -10m+22$ in $n = -30m+62$ v parametričnem prostoru (b); koordinata presečišča teh dveh premic podaja parametra (m, n), ki opisujeta premico skozi točki iz (a)

2. *Preslikaj problem iskanja presečišč premic v problem iskanje vrhov (ali iskanja maksimuma).* Predstavljajte si, da razdelimo ravnino m, n v končno število celic, nekakšno mrežo. Velikost celice je odvisna od natančnosti, ki jo potrebujemo. Vsaki celici priredimo števec $c(m, n)$, ki ga ob inicializaciji postavimo na 0. Zaradi enostavnosti naj slika vsebuje le eno premico (m', n'), ki jo sestavljajo točke $\mathbf{p}_1$,...,$\mathbf{p}_N$. Za vsako točko $\mathbf{p}_i$ povečajmo vse števce v celicah, ki imajo točko $\mathbf{p}_i$ na ustrezni premici (m, n) v parametričnem prostoru. Vse premice v parametričnem prostoru, ki smo jih dobili na ta način, se sekajo v točki (m', n'). V tej točki je števec $c(m', n') = N$, v vseh ostalih pa 1. Torej lahko najdemo premico na sliki z enostavnim iskanje vrha oziroma maksimuma $c(m', n')$ v parametričnem prostoru.



Slika 2: Od leve proti desni: parametra (polmer r in kot θ), ki določata premico v normalni obliki, šop premic skozi robni element in predstavitev šopa premic v parametričnem prostoru

Če hočemo zagotoviti pravilno delovanje zgornjega postopka, pa moramo biti pozorni še na naslednje stvari, ki jih lahko poimenujemo tudi bistveni problemi Houghove transformacije:

- **Zagotavljanje končnosti parametričnega prostora.** Oba parametra m in n lahko zavzameta vrednosti iz intervala $(-\infty, \infty)$, kar pomeni, da lahko obravnavamo le omejen del parametričnega prostora, saj bi bilo sicer število celic[1] preveliko za iskanje v doglednem času. Problem bi lahko rešili tako, da bi velikost celic povečali, vendar je posledica tega

[1]Prej omenjena mrežna razdelitev parametričnega prostora.

manjša natančnost postopka, še vedno pa ostaja tudi problem, da nismo preiskali celotnega parametričnega prostora. Še večji problem predstavlja dejstvo, da premic tipa $x = k$ (k je konstanta) v takšnem parametričnem prostoru ne moremo najti, saj so premice $n = x(-m) + y$ med seboj vzporedne; torej nimamo presečišč. Če enačbo zapišemo v *normalni obliki* $r = x\cos\theta + y\sin\theta$ oba problema izgineta[2]! Definicijski območji obeh parametrov r in θ sta namreč končni, predstavimo oziroma najdemo pa lahko tudi poljubno premico. Na tem mestu je potrebno izpostaviti še dejstvo, da je sedaj vsaka slikovna točka (oziroma šop premic skozi njo) predstavljena v parametričnem prostoru kot sinusoida in ne kot premica (glej sliko 2).

- **Hkratno iskanje več premic.** Slike z najdenimi robovi lahko vsebujejo veliko premic. Da najdemo vse, lahko enostavno poiščemo vse lokalne maksimume $c(m, n)$.

- **Učinek nelinearnih kontur.** Slike z najdenimi robovi pogosto vsebujejo robne točke, ki ne pripadajo nobeni premici. Primer takšnih točk so na primer točke, ki pripadajo neki ukrivljeni konturi ali pa šum, ki ga povzroči postopek za iskanje robov. Te točke v parametričnem prostoru povzročijo, da imajo števci dodatno, majhno, naključno vrednost. Posledica tega je množica lokalnih, šumnih vrhov v parametričnem prostoru, ki jih moramo znati ločiti od tistih vrhov, ki resnično predstavljajo premice. To lahko na najenostavnejši način zagotovimo s postavitvijo ustreznega pragu vrednostim števcev $c(m, n)$.

- **Šum.** Zaradi predstavitve slike s slikovnimi elementi in omejene točnosti postopka iskanja robov, se premice v parametričnem prostoru, ki ustrezajo točkam premice na sliki, velikokrat ne sekajo v eni točki. Posledično robne točke ne prispevajo le k povečanju enega samega števca, ampak k povečanju večih števcev v majhni okolici pravega števca. Torej je vrh v parametričnem prostoru, ki opisuje premico na sliki, razpršen v tej majhni okolici. Ta lastnost je še bolj opazna ob prisotnosti šumnih točk. V odvisnosti od ločljivosti parametričnega prostora in zahtevane natančnosti, bi lahko *iskali prave parametre* le z iskanjem lokalnega maksimuma (m', n') ali pa kot obteženo povprečje vrednosti (m, n) v okolici (m', n'), kjer velja $c(m, n) > \tau c(m', n')$ (τ je konstanta, za katero velja $0 < \tau < 1$; smiselna vrednost je na primer τ=0,9), uteži pa določimo skladno glede na vrednosti števcev.

  Slika 3(a) prikazuje sintetično sliko velikosti 64×64 slikovnih elementov, na kateri sta dve premici. Prikazana je le podmnožica točk, ki pripadajo premicama, na naključnih lokacijah pa so prisotne tudi šumne točke. Slika 3(b) prikazuje vrednost števcev v odvisnosti od parametrov m in n v parametričnem prostoru.

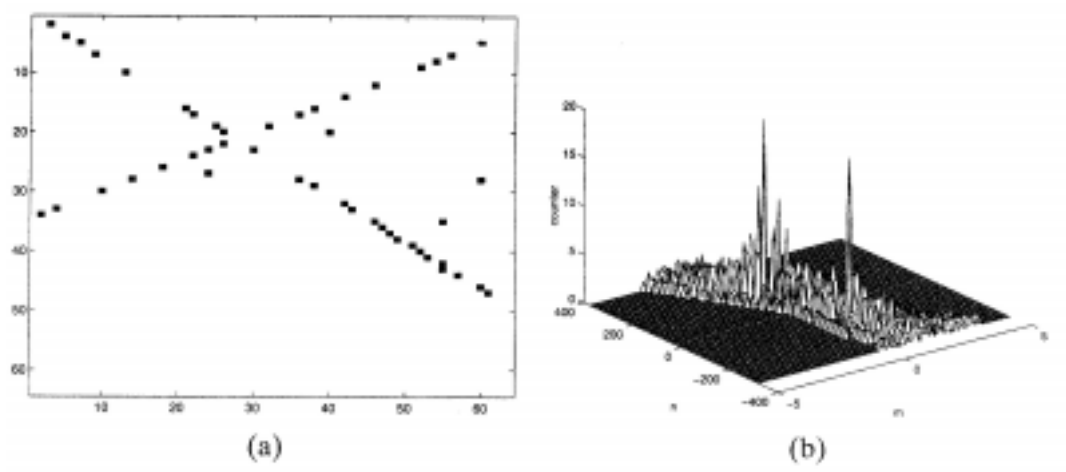
- **Kvantizacija.** Večje je akumulacijsko polje, boljši je približek krivulje, vendar algoritem potrebuje za izvršitev več časa, in obratno, manjše je akumulacijsko polje, slabši je približek krivulje, algoritem pa potrebuje za izvršitev manj časa.

- **Prag.** Kje postaviti mejo, ki določa, katere krivulje vsebujejo dovolj elementov (robnih točk)? Vedno je potrebno najti ustrezen kompromis.

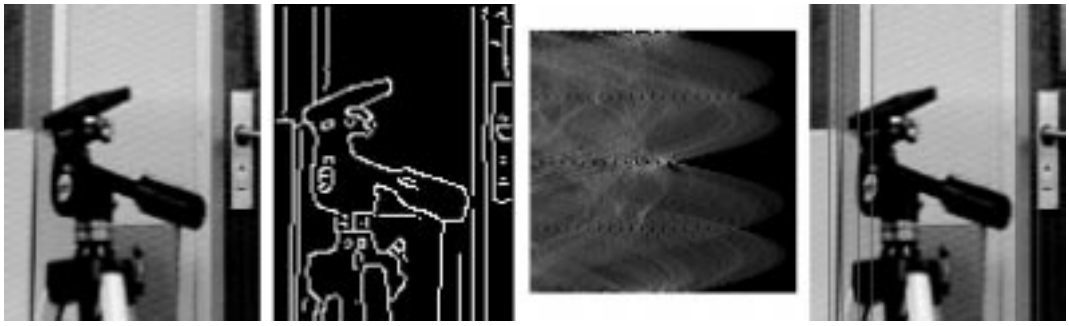
- **Večparametrične krivulje.** Akumulacijsko polje ima več dimenzij, kar občutno poveča časovno kompleksnost postopka.

Ker je v primeru iskanja linearnih segmentov akumulacijsko polje dvorazsežno, ga lahko prikažemo kot sivinsko sliko, s katere so lepo razvidni vrhovi akumulacijskega polja - zanimive premice (glej sliko 4). Vrednost posameznega polja (števca) v akumulacijskem polju je ponazorjena s svetlostjo slikovnega elementa. V sliki bolje vidimo strukturo akumulacijskega polja, če

[2] r je velikost daljice, ki predstavlja najkrajšo razdaljo od izhodišča slike do premice (torej gre za pravokotnico na premico), θ pa kot med abscisno osjo in omenjeno daljico.



Slika 3: (a) sintetična slika velikosti 64×64 slikovnih elementov, na kateri sta dve premici in nekaj naključnih (šumnih) točk, (b) vrednost števcev v odvisnosti od parametrov m in n v parametričnem prostoru (vrednost pove, koliko robnih točk pripomore k posamezni celici (m, n)); opazimo lahko, da sta maksimalna vrha očitna, veliko pa je tudi sekundarnih vrhov



Slika 4: Od leve proti desni: vhodna slika, najdeni robovi, akumulacijsko polje in izbrane premice na vhodni sliki

zmanjšamo skalo za velike vrednosti akumulacijskih elementov. Eden izmed možnih načinov je, da v sliki prikažemo kvadratni koren vrednosti akumulacijskega polja.

Sedaj lahko zapišemo algoritem za iskanje premic:

Algoritem za iskanje premic s pomočjo Houghove transformacije

Vhod naj bo binarna slika E, velikosti $M \times N$, v kateri je vsak slikovni element $E(i, j)$ enak 1, če gre za robni slikovni element in 0 sicer. r_d in θ_d naj bosta tabeli, ki predstavljata diskretizirana intervala parametrov r in θ ($r \in [0, \sqrt{M^2 + N^2}]$, $\theta \in [0, \pi]$), R in T pa naj bosta števili elementov teh tabel.

1. Diskretiziraj parametra r in θ z uporabo koraka Δr oziroma $\Delta\theta$, tako da bo zagotovljena ustrezna ločljivost in obvladljiva velikost tabel r_d in θ_d.

2. Naj bo $A(R, T)$ tabela celoštevilčnih števcev (akumulatorjev). Vse elemente tabele A postavi na 0.

3. Za vsak robni slikovni element $E(i, j) = 1$ in za $h = 1, ..., T$ delaj:

   a) izračunaj $r = i \cos\theta_d(h) + j \sin\theta_d(h)$,

   b) najdi takšen indeks k, ki opisuje najbližji element r_d izračunani vrednosti r,

c) povečaj $A(k,h)$ za ena.

4. Najdi vse lokalne maksimume (k_p, h_p), za katere velja $A(k_p, h_p) > \tau$, kjer je τ prag, ki ga določi uporabnik.

Izhod je množica parov $(r_d(k_p), \theta_d(h_p))$, ki opisujejo premice na sliki E v normalni obliki.

Postopek lahko še izboljšamo, če poznamo oceno $m_g(\mathbf{p})$ smeri roba skozi točko $\mathbf{p}$ in predpostavljamo, da predstavlja $m_g(\mathbf{p})$ tudi smer premice skozi točko $\mathbf{p}$. To pomeni, da lahko namesto povečanja vrednosti množici števcev na celotni premici v parametričnem prostoru, povečamo vrednost le števcu na lokaciji (m_g, n_g). Zaradi nezaupanja v oceno povezano s smerjo roba lahko povečamo vse števce na majhnem segmentu okoli lokacije (m_g, n_g). Velikost segmenta je odvisna od zanesljivosti ocene smeri roba. Na ta način lahko postopek Houghove transformacije zelo pospešimo.

2.2 Iskanje krivulj

Houghovo transformacijo lahko enostavno posplošimo tako, da lahko na sliki najdemo krivulje $y = f(x, \mathbf{a})$, kjer je $\mathbf{a}$=$[a_1, ..., a_P]^{\mathrm{T}}$ vektor P parametrov. Osnovni algoritem je zelo podoben algoritmu za iskanje premic:

Algoritem za iskanje krivulj s pomočjo Houghove transformacije

Vhod naj bo binarna slika E, velikosti $M \times N$, v kateri je vsak slikovni element $E(i,j)$ enak 1, če gre za robni slikovni element in 0 sicer. Naj bo $y = f(x, \mathbf{a})$ izbrana parametrizacija ciljne krivulje.

1. Diskretiziraj parametre $a_1, ..., a_P$ z uporabo ustreznega koraka, tako da bo zagotovljena ustrezna ločljivost in obvladljiva velikost parametričnega prostora. $s_1, ..., s_P$ naj bodo velikosti diskretiziranih intervalov.

2. Naj bo $A(s_1, ..., s_P)$ tabela celoštevilčnih števcev (akumulatorjev). Vse elemente tabele A postavi na 0.

3. Za vsak robni slikovni element $E(i,j) = 1$ povečaj vse števce, ki opisujejo krivuljo definirano z enačbo $y = f(x, \mathbf{a})$.

4. Najdi vse lokalne maksimume $\mathbf{a}_m$, za katere velja $A(\mathbf{a}_m) > \tau$, kjer je τ prag, ki ga določi uporabnik.

Izhod je množica vektorjev $\mathbf{a}_1$,...,$\mathbf{a}_P$, ki opisujejo krivulje na sliki E.

Kot primer si oglejmo postopek iskanja krožnic. Zapišimo enačbo krožnice v kartezičnih koordinatah:

$$(x - x_0)^2 + (y - y_0)^2 = r^2.$$

Iz enačbe je razvidno, da potrebujemo trorazsežno akumulacijsko polje, saj je vsaka krožnica določena s tremi parametri: r, x_0 in y_0. Čeprav pa imamo tukaj opravka s trorazsežnim akumulacijskim poljem, lahko hitro izvajanje zagotovimo tako, da izvršimo transformacijo le nad zanimivimi regijami slike. Podajmo sedaj primer psevdo kode za iskanje krožnih segmentov (tretja točka zgornjega algoritma):

```
za vsak robni element E(i,j) slike/regije delaj
   za x_0 = 1 do širine slike/regije delaj
      za y_0 = 1 do višine slike/regije delaj
         r = sqrt((i - x_0)^2 + (j - y_0)^2)
         zaokroži r tako, da ustreza željeni kvantizaciji
         če r ≤ min{širina slike/regije / 2, višina slike/regije / 2}, potem
            povečaj vrednost elementa (r, x_0, y_0) v akumulacijskem polju
```

Predzadnja vrstica pove, da iščemo le krožnice, katerih polmer je največ polovica širine oziroma višine slike oziroma regije. S tem je zagotovljeno, da je krožnica v večini vsebovana v sliki oziroma regiji, hkrati pa pozitivno vplivamo tudi na vsebino akumulacijskega polja; števci imajo manjšo vrednost in opisi krožnic so bolj zgoščeni v območju zanimivih regij, če smo se na preiskovanje le-teh seveda omejili.

Velikost parametričnega prostora narašča eksponentno s številom parametrov, s tem pa postane postopek časovno prezahteven. Če zaradi enostavnosti predvidevamo, da imajo vsi diskretizirani intervali velikost N, je kompleksnost izčrpnega iskanja krivulje s p parametri proporcialna N^p! Ta problem predstavlja največjo pomanjkljivost Houghove transformacije. Rešitev bi lahko iskali v ideji spremenljive ločljivosti parametričnega prostora, saj bi lahko kompleksnost postopka zmanjšali, če bi parametrični prostor razdelili bolj grobo, daleč od pomembnih maksimumov.

2.3 Ugotovitve

Algoritem Houghove transformacije je *algoritem glasovanja*: vsaka robna točka *glasuje* za vse kombinacije parametrov, ki bi generirali to točko, če bi bila del ciljne krivulje. S tega stališča lahko na akumulacijsko polje oziroma tabelo števcev v parametričnem prostoru gledamo kot na *histogram*. Končno število glasov $c(\mathbf{m})$ (števec na koordinatah $\mathbf{m}$) lahko interpretiramo kot relativno verjetnost hipoteze "*krivulja s parametri* $\mathbf{m}$ *obstaja na sliki*".

Na Houghovo transformacijo pa lahko gledamo tudi kot na algoritem *ujemanja vzorcev* (ang. *pattern matching*): razred krivulj predstavljen v parametričnem prostoru je razred vzorcev. Kot že omenjeno, je Houghova transformacija učinkovitejša od postopka neposrednega primerjanja vzorcev (ang. *template matching*), ki predstavlja primerjavo vseh možnih primerov ciljnega vzorca z vhodno sliko.

Izpostavimo pozitivne lastnosti Houghove transformacije:

- Ker so vse robne točke obdelane neodvisno, se algoritem uspešno spopada z okluzijo (s tem mislimo na lastnost, da se največkrat pravi maksimumi v parametričnem prostoru ne mešajo oziroma zamenjujejo s tistimi, ki jih povzroči šum v sliki). To seveda velja, če šum ne povzroča takšnih maksimumov kot so tisti, ki predstavljajo najmanjše krivulje.
- Postopek je relativno neobčutljiv na šum, ker je malo verjetno, da bi vse šumne robne točke vplivale na le en števec, ali na le neko majhno število števcev.
- V enem prehodu skozi sliko postopek odkrije vse primere iskane krivulje.

Najverjetneje je največja omejitev Houghove transformacije naglo povečevanje iskalnega časa ob večanju števila parametrov ciljne krivulje. Problem predstavljajo tudi neciljne krivulje na sliki, ki lahko povzročijo lažne maksimume (vrhove) v parametričnem prostoru: na primer iskanje premic je lahko neuspešno, če so na sliki prisotne krožnice z majhno ukrivljenostjo.

3 Pomerjanje elips

Najprej namenimo nekaj besed problemu pomerjanja modelov (ang. *model fitting*). Predvidevajmo, da vemo, da nekateri slikovni elementi ležijo na premici. Zaradi predstavitve slike s slikovn-

imi 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. Algoritmi za iskanje elips, omenjeni v tem poglavju, zahtevajo na vhodu sliko z najdenimi robovi, na izhodu pa dobimo enačbo elipse, ki se najbolje prilega vsem robnim točkam. Predvidevajmo torej, da smo našli množico robnih točk, ki najverjetneje pripadajo isti elipsi. Problem, ki ga želimo rešiti, lahko v splošnem podamo kot:

Iskanje 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[3]*, določena z vektorjem parametrov* $\mathbf{a}$= $[a, b, c, d, e, f]^{\mathrm{T}}$.

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

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

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

In kakšna je primerna funkcija razdalje? Na to vprašanje lahko podamo dva odgovora: Evklidova razdalja in algebraična razdalja!

3.1 Evklidova razdalja

Prva ideja predlaga iskanje minimuma Evklidove razdalje med elipso in podanimi točkami. Enačbo (1) lahko torej zapišemo kot

$$\min_{\mathbf{a}} \sum_{i=1}^{N} \|\mathbf{p} - \mathbf{p}_i\|^2, \tag{2}$$

pri čemer $\mathbf{p}$ pripada elipsi $f(\mathbf{p},\mathbf{a})$=0.

Geometrijsko izgleda Evklidova razdalja kot najprimernejša, vendar pa, na žalost, vodi le k približnemu, numeričnemu algoritmu. Kako to? Izhajajmo iz *Lagrangeovega obrazca* za iskanje interpolacijskega polinoma in poskušajmo rešiti problem podan v enačbi (2). Definirajmo funkcijo

$$L = \sum_{i=1}^{N} \Big(\|\mathbf{p} - \mathbf{p}_i\|^2 - 2\lambda f(\mathbf{p}, \mathbf{a}) \Big)$$

in postavimo $\frac{\partial L}{\partial x} = \frac{\partial L}{\partial y} = 0$, kar nas pripelje[4] do

$$\mathbf{p} - \mathbf{p}_i = \lambda \nabla f(\mathbf{p}, \mathbf{a}). \tag{3}$$

Ker ne poznamo vrednosti $\mathbf{p}$, jo poskušamo izraziti kot funkcijo izračunljivih količin. Da lahko to naredimo, moramo vpeljati dve *aproksimaciji*:

[3]Pravzaprav enačba za $f(\mathbf{p}, \mathbf{a})$ v resnici predstavlja splošno enačbo krivulj drugega reda (stožernic). O tem bomo spregovorili nekoliko več kasneje.

[4]Celotna izpeljava je podana v dodatku A.

1. V obzir vzamemo aproksimacijo prvega reda

$$0 = f(\mathbf{p}, \mathbf{a}) \approx f(\mathbf{p}_i, \mathbf{a}) + [\mathbf{p} - \mathbf{p}_i]^{\mathrm{T}} \nabla f(\mathbf{p}_i, \mathbf{a}). \tag{4}$$

2. Predvidevajmo, da je $\mathbf{p}_i$ dovolj blizu krivulji, tako da lahko zapišemo $\nabla f(\mathbf{p}, \mathbf{a}) \approx \nabla f(\mathbf{p}_i, \mathbf{a})$.

Na podlagi druge točke lahko enačbo (3) zapišemo kot

$$\mathbf{p} - \mathbf{p}_i = \lambda \nabla f(\mathbf{p}_i, \mathbf{a}),$$

ki nam vstavljena v enačbo (4) da

$$\lambda = \frac{-f(\mathbf{p}_i, \mathbf{a})}{\|\nabla f(\mathbf{p}_i, \mathbf{a})\|^2}.$$

Če sedaj dobljeno enačbo vstavimo v enačbo (3), dobimo rešitev, ki smo jo iskali:

$$\|\mathbf{p} - \mathbf{p}_i\| = \frac{|f(\mathbf{p}_i, \mathbf{a})|}{\|\nabla f(\mathbf{p}_i, \mathbf{a})\|}.$$

Zgornja enačba nam sedaj omogoča zamenjavo neznane količine $\|\mathbf{p}-\mathbf{p}_i\|$ v enačbi 2 s funkcijo, ki jo znamo izračunati. Zapišimo algoritem:

Algoritem za iskanje elips s pomočjo Evklidove razdalje

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

1. Postavi $\mathbf{a}$ na začetno vrednost[5] $\mathbf{a}_0$.

2. Izhajaj iz začetne vrednosti $\mathbf{a}_0$ in izvrši postopek numerične minimizacije nad

$$\min_{\mathbf{a}} \sum_{i=1}^{N} \frac{|f(\mathbf{p}_i, \mathbf{a})|^2}{\|\nabla f(\mathbf{p}_i, \mathbf{a})\|^2}.$$

Izhod je vektor $\mathbf{a}_m$, ki opisuje najbolje prilegajočo se elipso.

In kako učinkovit je podan algoritem? Zadovolji nas le delno. Izhajali smo iz najboljše možne, *prave* (Evklidske) razdalje, vendar smo bili prisiljeni uporabiti aproksimaciji, ki sta nas pripeljali do nelinearne minimizacije, ki jo lahko rešimo le numerično. Postopek nam celo ne jamči, da bomo kot rešitev dobili krivuljo elipse! Dobimo lahko katerokoli krivuljo drugega reda[6] (stožernico), saj nismo vektorja parametrov $\mathbf{a}$ nikakor omejili! Poleg tega naletimo na standardne probleme postopka numerične optimizacije, med katerimi sta najbolj pereča iskanje začetne vrednosti $\mathbf{a}_0$ in globalnega minimuma[7].

Torej, če *prava* (Evklidova) razdalja zahteva aproksimaciji in numerično optimizacijo, ali lahko mogoče najdemo približno razdaljo, ki bi nas pripeljala do rešitve brez zahtev po nadaljnih aproksimacijah? Odgovor je pritrdilen, postopek pa je podan v naslednjem poglavju.

[5] Razumna začetna vrednost je rešitev algoritma podanega v poglavju 3.2.

[6] Ta lastnost ni nujno le slaba, saj nam podan postopek omogoča iskanje splošnih krivulj drugega reda (stožernic), zavedati pa se moramo, da je lahko rezultat iskanja krivulja, ki je nismo pričakovali oziroma iskali!

[7] Kako zagotoviti, da postopek ne obtiči v lokalnem minimumu?

3.2 Algebraična razdalja

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 (1) v linearni problem, ki ga lahko rešimo brez nadaljnih aproksimacij.

Spremenjena enačba problema (1) ima sedaj naslednjo obliko:

$$\min_{\mathbf{a}} \sum_{i=1}^{N} |f(\mathbf{p}_i,\mathbf{a})|^2 \quad \text{oziroma} \quad \min_{\mathbf{a}} \sum_{i=1}^{N} |\mathbf{x}_i^{\mathrm{T}}\mathbf{a}|^2. \tag{5}$$

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

$$D = b^2 - 4ac = \mathbf{a}^{\mathrm{T}}\mathbf{C}\mathbf{a} = -1, \tag{6}$$

$$\mathbf{C} = \begin{bmatrix} 0 & 0 & -2 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ -2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}. \tag{7}$$

Opazimo lahko, da lahko na zgornjo omejitev[8] gledamo kot na normalizirano različico splošne omejitve $D = b^2 - 4ac < 0$ (diskriminanta mora biti negativna), ki zagotavlja elipso.

Sedaj zapišimo enačbo (5) kot

$$\min_{\mathbf{a}} \|\mathbf{a}^{\mathrm{T}}\mathbf{X}^{\mathrm{T}}\mathbf{X}\mathbf{a}\| \quad \text{oziroma} \quad \min_{\mathbf{a}} \|\mathbf{a}^{\mathrm{T}}\mathbf{S}\mathbf{a}\|, \tag{8}$$

$$\mathbf{X} = \begin{bmatrix} x_1^2 & 2x_1y_1 & y_1^2 & 2x_1 & 2y_1 & 1 \\ x_2^2 & 2x_2y_2 & y_2^2 & 2x_2 & 2y_2 & 1 \\ & & \cdots & & & \\ x_N^2 & 2x_Ny_N & y_N^2 & 2x_N & 2y_N & 1 \end{bmatrix}. \tag{9}$$

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. Ponovno uporabimo Lagrangeov obrazec, ki nam problem (8) prevede na

$$\mathbf{S}\mathbf{a} = \lambda\mathbf{C}\mathbf{a}. \tag{10}$$

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[9]. 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 (9).
2. Sestavi razširjeno matriko $\mathbf{S}$=$\mathbf{X}^{\mathrm{T}}\mathbf{X}$.

[8] Omejitev spada med t.i. kvadratične omejitve.
[9] Glej dodatek B.

3. Sestavi matriko omejitve $\mathbf{C}$, kot je to zapisano v (7).

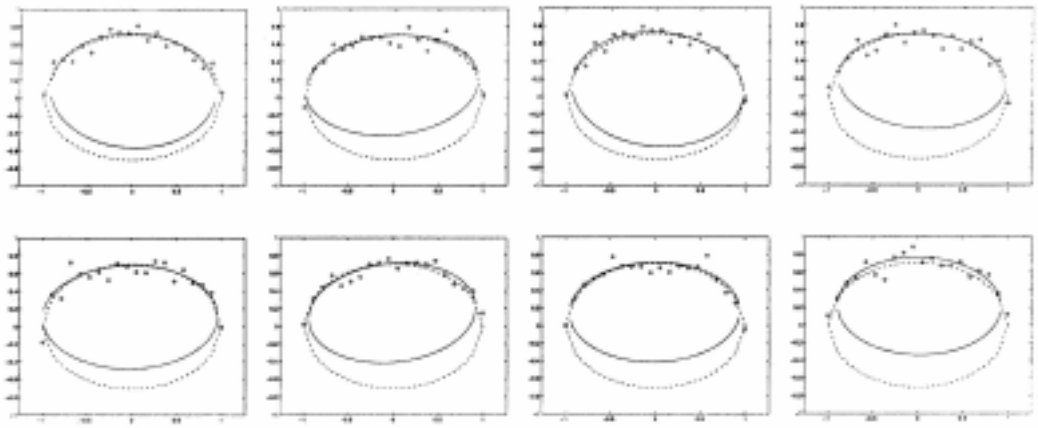
4. Z numeričnim postopkom izračunaj lastne vrednosti posplošenega problema lastne vrednosti. Edino negativno lastno vrednost označimo z λ_n .

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

Slika 5 prikazuje rešitve algoritma za iskanje elips s pomočjo algebraične razdalje ob naraščajoči količini Gaussovega šuma.

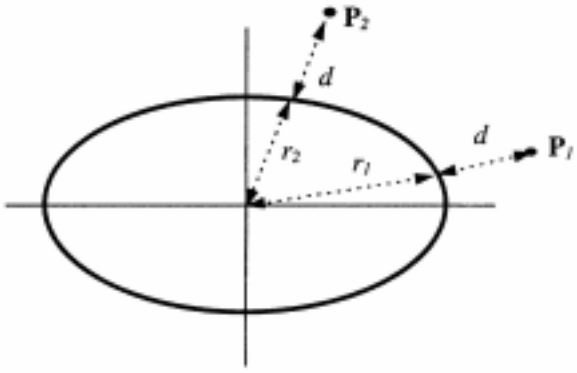
Slika 5: Rešitve algoritma za iskanje elips s pomočjo algebraične razdalje ob naraščajoči količini Gaussovega šuma; slike od leve proti desni vsebujejo od 3% do 20% šuma v podatkih (Sigma (σ) predstavlja standardno odstopanje)

Zgornji algoritem kaže nagnjenost k rešitvam majhne ekscentričnosti[10]. To lastnost 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 6. Ponazorimo to trditev z geometrijsko interpretacijo algebraične razdalje:



Slika 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).

[10]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.



Slika 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 $\mathbf{P}_1$

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 7).*[11]

Za konstanten, opredeljen d je Q maksimalen v presečišču elipse s svojo manjšo osjo (podobno velja za točko $\mathbf{P}_2$ na sliki 7) in minimalen v presečišču elipse s svojo večjo osjo (podobno velja za točko $\mathbf{P}_1$ na sliki 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.

3.3 Učinkovito iskanje elips

Ali ste se že vprašali od kod sploh dobimo množico N slikovnih točk $\mathbf{p}_1$,...,$\mathbf{p}_N$? V realnih aplikacijah, kjer ta informacija ni vnaprej podana, predstavlja iskanje teh točk težek problem. V nekaterih primerih lahko te točke enostavno podamo ročno. Če pa to ni sprejemljiv postopek, smo bolj ali manj odvisni od algoritma, ki nam reši problem grupiranja, z zagotovitvijo urejenega spiska robnih točk, na primer verižne kode.

V obeh primerih obstaja velika verjetnost, da so v množici točk tudi takšne točke, ki ne podpirajo statistične predpostavke cenilca (človeka ali nekega algoritma). V našem primeru je takšna točka na primer robna točka, za katero napačno predvidevamo, da pripada elipsi. Oba do sedaj omenjena algoritma za iskanje elips gledata, v smislu cenilcev po metodi najmanjših kvadratov, na množico vhodnih točk kot na točke, ki *pripadajo* krivulji. Torej lahko že nekaj točk, za katere napačno predvidevamo, da pripadajo elipsi, rezultat algoritma močno poslabša!

S pojmom *učinkoviti cenilci*[12] označujemo metode, ki takšne točke tolerirajo. Učinkovita razdalja, ki največkrat daje dobre rezultate je absolutna vrednost. Uporabimo jo v algoritmu:

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

[12] Kratka (in jedrnata) predstavitev učinkovitih cenilcev je podana v dodatku C.

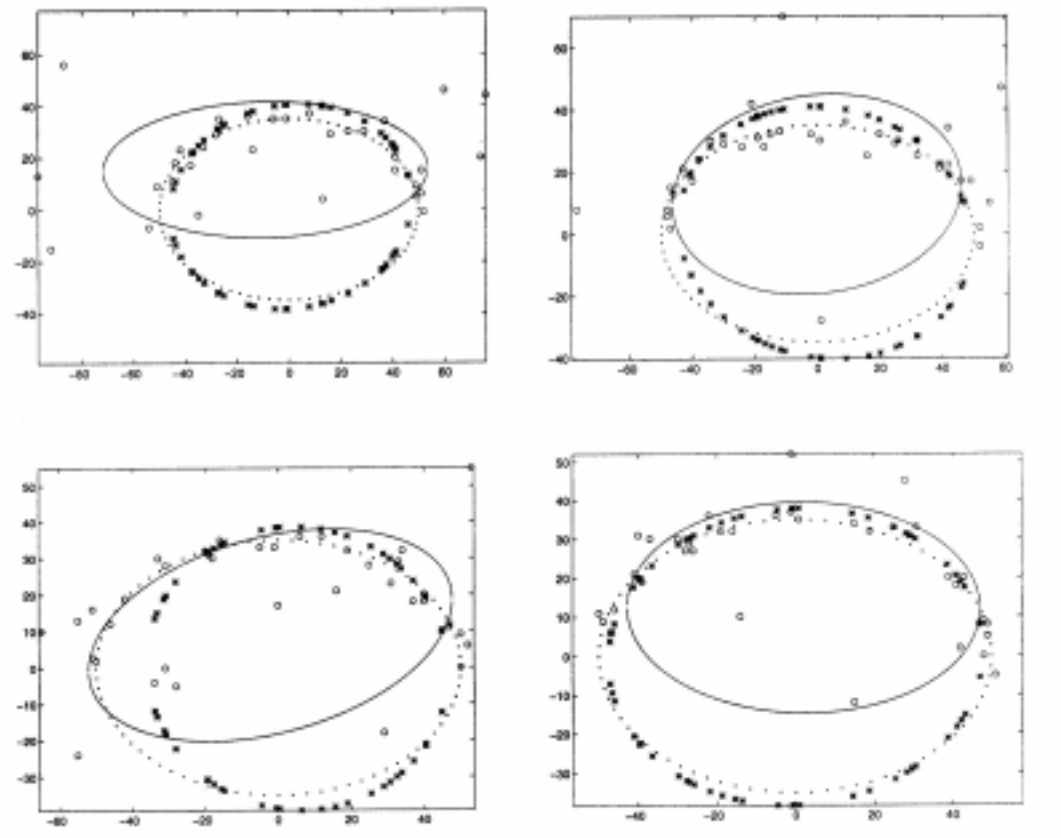
Algoritem za učinkovito iskanje elips

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

1. Rešitev algoritma za iskanje elips s pomočjo algebraične razdalje označi z $\mathbf{a}_0$.
2. Izhajaj iz začetne vrednosti $\mathbf{a}_0$ in izvrši postopek numerične minimizacije nad

$$\min_{\mathbf{a}} \sum_{i=1}^{N} |\mathbf{x}_i^{\mathrm{T}} \mathbf{a}|.$$

Izhod je vektor $\mathbf{a}_m$, ki opisuje najbolje prilegajočo se elipso.



Slika 8: Primerjava rešitev algoritma za iskanje elips s pomočjo algebraične razdalje in algoritma za učinkovito iskanje elips, kjer vhodna množica točk vsebuje okoli 20% takšnih točk, za katere napačno predvidevamo, da pripadajo elipsi; s krogci so označeni robni slikovni elementi, zvezdice prikazujejo rešitev algoritma za učinkovito iskanje elips, polna krivulja prikazuje rešitev algoritma za iskanje elips s pomočjo algebraične razdalje in pike pravo (resnično) krivuljo

Slika 8 ilustrira problem, ki ga algoritmu za iskanje elips s pomočjo algebraične razdalje povzročijo točke, za katere napačno predvidevamo, da pripadajo elipsi in nam omogoča primerjavo rešitev algoritma za iskanje elips s pomočjo algebraične razdalje in algoritma za učinkovito iskanje elips. Robne točke vsebujejo tudi veliko Gaussovega šuma. Algoritma sta na vhodu dobila 40 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=0.05, kar je približno 7% velikosti manjše polosi b). Okoli 20.% točk sta bili predhodno spremenjeni koordinati tako, da se jim je dodalo uniformno odstopanje z intervala $[-b, b]$. S tem smo dobili točke, za katere napačno predvidevamo, da pripadajo elipsi. Opazimo lahko, da algoritmu za iskanje elips s pomočjo algebraične razdalje te točke povzročajo hude probleme, algoritem za učinkovito iskanje elips pa takšne točke uspešno tolerira.

3.4 Ugotovitve

Če vhodna množica točk vsebuje *malo šuma*, predstavlja algoritem za iskanje elips s pomočjo algebraične razdalje dobro izbiro. Če je vhodna množica točk *zelo šumna*, ne smemo podcenjevati ekscentričnosti najboljše rešitve. V primeru zelo šumnih podatkov predstavlja dobro izbiro algoritem za iskanje elips s pomočjo Evklidove razdalje, pri čemer izhajamo iz rešitve $\mathbf{a}_0$ algoritma za iskanje elips s pomočjo algebraične razdalje. Če je v vhodni množici prisotnih veliko točk, za katere *napačno predvidevamo*, da pripadajo elipsi, rešitvi obeh algoritmov (algoritma za iskanje elips s pomočjo Evklidove razdalje in algoritma za iskanje elips s pomočjo algebraične razdalje) ne zadovoljita naših pričakovanj. V takšnem primeru predstavlja dobro izbiro algoritem za učinkovito iskanje elips, pri čemer izhajamo iz rešitve $\mathbf{a}_0$ algoritma za iskanje elips s pomočjo algebraične razdalje. Če predstavlja *hitrost* največji problem, potem moramo uporabiti algoritem za iskanje elips s pomočjo algebraične razdalje. Na tem mestu predvidevamo, da imamo na voljo učinkovit paket (algoritem, knjižnico ali program) za reševanje problema iskanja lastnih vrednosti.

Kvantitativna razlaga pojmov *malo šuma* in *veliko šuma* je odvisna od števila točk, gostote točk na sami elipsi, statistične porazdelitve točk in prisotnosti šuma. Algoritem za iskanje elips s pomočjo algebraične razdalje naj bi dajal dobre rezultate, če ima na voljo več kot 10 točk, ki opisujejo polovico elipse in so le te uniformno oddaljene med seboj glede na os x, hkrati pa lahko vsebujejo Gaussov šum, ki lahko povzroči standardno odstopanje do največ 5% velikosti manjše polosi.

Literatura

[1] J. N. Bronštejn, K. A. Semendjajev, *Matematični priročnik*, Tehniška založba Slovenije, 1988.

[2] A. W. Fitzgibbon, M. Pilu, R. B. Fisher, Direct Least Squares Fitting of Ellipses, *Proceedings of ICPR*, Vienna, 1996, pp. 253-257.

[3] A. Leonardis, Izbrana poglavja iz programske opreme - računalniški vid, *Zapiski predavanj*, 1997.

[4] P. Peer, Avtomatsko iskanje človeških obrazov na slikah, *Diplomska naloga*, Univerza v Ljubljani, Fakulteta za računalništvo in informatiko, 1998.

[5] W. H. Press, W. T. Vetterling, S. A. Teukolsky, B. P. Flannery, *Numerical Recepies in C*, Cambridge University Press, 1992.

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

Dodatki

A Uporaba Lagrangeovega obrazca

V poglavju 3.1 smo uporabili Lagrangeov obrazec za iskanje interpolacijskega polinoma in poskušali rešiti problem podan v enačbi (2). Definirali smo funkcijo

$$L = \sum_{i=1}^{N} \Big(\|\mathbf{p} - \mathbf{p}_i\|^2 - 2\lambda f(\mathbf{p}, \mathbf{a}) \Big)$$

in postavili $\frac{\partial L}{\partial x} = \frac{\partial L}{\partial y} = 0$, kot nam to narekuje postopek iskanja interpolacijskega polinoma. Člen $2\lambda f(\mathbf{p}, \mathbf{a})$ lahko poimenujemo optimizacijski člen, parameter λ pa optimizacijski parameter.

$\mathbf{p}$ je točka na elipsi, ki je najbližja točki $\mathbf{p}_i$, zato lahko točko $\mathbf{p}$ za vsako točko $\mathbf{p}_i$ posebej zapišemo kot $\mathbf{p}{=}[x, y]^{\mathrm{T}}{=}\mathbf{r}_i{=}[z_i, w_i]^{\mathrm{T}}$. Torej sta vrednosti $\mathbf{p}_i$ in $\mathbf{r}_i$ medsebojno odvisni. L lahko tako v splošnem zapišemo kot

$$L = \sum_{i=1}^{N} g(\underbrace{z_i, w_i}_{\mathbf{p}=\mathbf{r}_i}, \underbrace{x_i, y_i}_{\mathbf{p}_i}, \underbrace{a, b, c, d, e, f}_{\mathbf{a}}, \lambda).$$

Sedaj preoblikujmo L:

$$\begin{aligned} L &= \sum_{i=1}^{N} \left(\left(\sqrt{((x - x_i)^2 + (y - y_i)^2)} \right)^2 - 2\lambda(ax^2 + 2bxy + cy^2 + 2dx + 2ey + f) \right) = \\ &= \sum_{i=1}^{N} \Big(((x - x_i)^2 + (y - y_i)^2) - 2\lambda(ax^2 + 2bxy + cy^2 + 2dx + 2ey + f) \Big) \end{aligned}$$

in zapišimo parcialna odvoda:

$$\frac{\partial L}{\partial x} = \frac{\partial L}{\partial z_i} = \frac{\partial g(z_i, w_i, x_i, y_i, a, b, c, d, e, f, \lambda)}{\partial z_i} = 2(z_i - x_i) - 2\lambda(2az_i + 2bw_i + 2d) = 0$$

$$\frac{\partial L}{\partial y} = \frac{\partial L}{\partial w_i} = \frac{\partial g(z_i, w_i, x_i, y_i, a, b, c, d, e, f, \lambda)}{\partial w_i} = 2(w_i - y_i) - 2\lambda(2cw_i + 2bz_i + 2e) = 0.$$

V splošnem torej velja:

$$\begin{aligned} x - x_i &= \lambda(2ax + 2by + 2d) \\ y - y_i &= \lambda(2cy + 2bx + 2e) \end{aligned},$$

kar nas (če ti dve enačbi združimo) pripelje do vektorskega zapisa

$$\begin{bmatrix} x - x_i \\ y - y_i \end{bmatrix} = \lambda \begin{bmatrix} 2ax + 2by + 2d \\ 2cy + 2bx + 2e \end{bmatrix},$$

ki je ekvivalenten zapisu

$$\mathbf{p} - \mathbf{p}_i = \lambda \nabla f(\mathbf{p}, \mathbf{a}),$$

kjer je

$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y} \right]^{\mathrm{T}}.$$

B Rešitev posplošenega problema lastne vrednosti

Če par[13] $(\lambda_i, \mathbf{u}_i)$ predstavlja rešitev enačbe (10), potem jo predstavlja tudi par $(\lambda_i, \mu\mathbf{u}_i)$ za vsak μ, s pomočjo enačbe (6) pa lahko najdemo vrednost μ_i iz $\mu_i^2 {\mathbf{u}_i}^{\mathrm{T}}\mathbf{C}\mathbf{u}_i = -1$:

$$\mu_i = \sqrt{\frac{-1}{{\mathbf{u}_i}^{\mathrm{T}}\mathbf{C}\mathbf{u}_i}} = \sqrt{\frac{-\lambda_i}{{\mathbf{u}_i}^{\mathrm{T}}\mathbf{S}\mathbf{u}_i}}. \tag{11}$$

Ob upoštevanju omejitve podane v enačbi (6), $\mathbf{a}_0 = \mu_i\mathbf{u}_i$ predstavlja rešitev enačbe (10). 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 (11) pozitiven. V splošnem je $\mathbf{S}$ pozitivno definitna, torej je ${\mathbf{u}_i}^{\mathrm{T}}\mathbf{S}\mathbf{u}_i$ vedno pozitivna vrednost za vsak $\mathbf{u}_i$. Kvadratni koren tako obstaja le, če $\lambda_i < 0$, kar pomeni, da mora imeti rešitev enačbe (10) negativno lastno vrednost. Matrika omejitve (7) ima le eno negativno lastno vrednost, kar pomeni, da imamo le eno samo rešitev.

C Učinkoviti cenilci in pomerjanje modelov

V tem poglavju je predstavljenih nekaj osnovnih konceptov, ki stojijo za pojmom *učinkoviti cenilci*, povdarek pa je na t.i. *M-cenilcih*[14]. Omejimo se na razlago naslednjih vprašanj:

- Zakaj lahko s stališča statistike na metodo najmanjših kvadratov gledamo kot na cenilca maksimalne verjetnosti?
- Zakaj metoda najmanjših kvadratov daje toliko slabše rezultate ob prisotnosti točk, za katere napačno predvidevamo, da pripadajo elipsi (lahko poljubni krivulji)?
- Zakaj cenilec, ki temelji na absolutni vrednosti, tolerira točke, za katere napačno predvidevamo, da pripadajo elipsi, bolje kot metoda najmanjših kvadratov? (Glej poglavje 3.3.)

Metoda najmanjših kvadratov kot cenilec maksimalne verjetnosti

Na voljo imamo N robnih slikovnih elementov $\mathbf{p}_i = [x_i, y_i]^{\mathrm{T}}$, $i = 1, ..., N$, in model krivulje $y = f(x, \mathbf{a})$, kjer je $\mathbf{a}$ vektor parametrov, f pa znana funkcija. Predvidevajmo, da vsebujejo točke določeno stopnjo šuma. Dobro poznana ocena vektorja parametrov $\mathbf{a}_0$, s pomočjo katerega funkcija $f(x, \mathbf{a}_0)$ najbolje interpolira vhodne podatke, v smislu najmanjših kvadratov, je

$$\mathbf{a}_0 = \min_{\mathbf{a}} \sum_{i=1}^{N} |y_i - f(x_i, \mathbf{a})|^2. \tag{12}$$

Ob prepostavki o količini šuma v podatkih, *hočemo pokazati, da je* $\mathbf{a}_0$ *vektor parametrov, ki maksimizira verjetnost, da so podatki šumna različica krivulje* $f(x, \mathbf{a})$.

Od kod pa sploh dobimo zgornjo enačbo in na kakšnem principu temelji? Odgovor nas pripelje do pojma cenilec maksimalne verjetnosti. Ob podani množici vrednosti x_i in y_i, nam intuicija pove, da so določeni konkretni vektorji $\mathbf{a}$ malo verjetni (tisti, ki vstavljeni v enačbo krivulje $f(x, \mathbf{a})$ nikakor ne opisujejo robnih točk), medtem ko so drugi lahko zelo verjetni (tisti, ki dobro opisujejo robne točke). Kako bi lahko kvantificirali ta intuitivni občutek? Kako lahko izberemo parametre, ki imajo veliko verjetnost, da so pravi? Vprašanje: "Kakšna je verjetnost, da je nek konkreten vektor $\mathbf{a}$ iskana rešitev?", je povsem neumestno, saj takšne verjetnosti ne znamo oceniti. Razlog za to je, da ne obstaja nek statističen prostor modelov iz katerih bi

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

[14] M označuje t.i. cenilce maksimalne verjetnosti (ang. *maximum likelihood estimators*), za katere je značilno, da imajo modeli, ki jih iščemo, M parametrov. M-cenilci so najpomembnejša skupina učinkovitih cenilcev, obstajata pa še dve drugi skupini: L-cenilci in R-cenilci. Osnovo M-cenilcev podaja enačba (13).

lahko dobili parametre. Na razpolago imamo le en model, tisti pravi, in statističen prostor podatkovnih množic, ki jih dobimo iz tega modela.

S tem v mislih lahko vprašanje postavimo drugače: "Kakšna je verjetnost, da bi dobili vhodno podatkovno množico, če poznamo konkretne parametre?" Če y_i zavzema zvezne vrednosti, bo verjetnost vedno enaka nič, razen če vsaki točki dodamo frazo *plus ali minus* Δy. Vzemimo torej to frazo v zakup. Če je verjetnost, da bi dobili vhodno podatkovno množico zelo majhna, potem lahko zaključimo, da trenutni konkretni parametri niso ustrezni. Intuicija nam pove, da podatkovna množica ne bi smela biti neverjetna za pravilne vrednosti parametrov.

Z drugimi besedami, istovetimo verjetnost pojavitve podatkov ob podanih parametrih (kar znamo izračunati) z verjetnostjo parametrov ob podanih podatkih. In ta istovetnost temelji zgolj na intuiciji; nima matematične osnove.

Ko enkrat sprejmemo to istovetnost, smo le še korak od najboljših možnih parametrov, ki jih dobimo z iskanjem *maksimalne verjetnosti* (podobnosti). Ta način ocenjevanja parametrov imenujemo tudi *ocenjevanje maksimalne verjetnosti*.

Sedaj se lahko vrnemo k enačbi (12). Predvidevajmo, da ima vsaka koordinata y_i neko napako, ki je povsem naključna, neodvisna in porazdeljena po normalni (Gaussovi) porazdelitvi okoli pravega modela $y = f(x, \mathbf{a})$. Predvidevajmo tudi, da so standardna odstopanja σ teh normalnih porazdelitev enaka za vse vhodne podatke. Potem je verjetnost P, da vsi vhodni podatki padejo v Δy območje prave vrednosti, ob podanem parametričnem vektorju, enaka

$$P \propto \prod_{i=1}^{N} \left(e^{-\frac{(y_i - f(x_i, \mathbf{a}))^2}{2\sigma^2}} \Delta y \right). \tag{13}$$

Da dobimo enačbo (12) iz enačbe (13) moramo le še maksimizirati enačbo (13), kar pa je ekvivalentno iskanju maksimuma logaritma te enačbe oziroma iskanju minimuma negativne vrednosti logaritma te enačbe:

$$-\log P = \left[\sum_{i=1}^{N} \frac{(y_i - f(x_i, \mathbf{a}))^2}{2\sigma^2} \right] - N \log \Delta y,$$

kjer lahko v postopku minimizacije konstante σ, N in Δy ignoriramo.

Pokazali smo torej, da je metoda najmanjših kvadratov cenilec maksimalne verjetnosti, če so napake v podatkih neodvisne in normalno porazdeljene ob konstantnem standardnem odstopanju.

Prisotnost točk, za katere napačno predvidevamo, da pripadajo krivulji

Ker smo predvidevali, da je šum v podatkih Gaussov, se verjetnost, da šumna točka leži v območju razdalje d od prave točke, naglo manjša z d. Posledično metoda najmanjših kvadratov verjame, da večina točk leži znotraj intervala nekaj standardnih odstopanj od pravega, vendar neznanega modela. Recimo sedaj, da podatki vsebujejo nek majhen odstotek točk, ki nikakor ne morejo pripadati krivulji[15], kateri pripadajo ostale točke. Recimo tudi, da te točke niso konsistentne z Gaussovo hipotezo. Ob pomankanju podrobnejše informacije metoda najmanjših kvadratov verjame, da so tudi te točke blizu pravega modela. Te točke lahko na koncu povzročijo, da je rešitev zelo različna od pravega modela. To pa je povsem nesprejemljivo.

Uporaba absolutne vrednosti

Bistvo problema metode najmanjših kvadratov je, da se normalna (Gaussova) porazdelitev naglo manjša, ko d postane večji od σ. Rešitev lahko iščemo v porazdelitvi šuma, ki ne izgine tako hitro

[15] Razlogov, zakaj so te točke med podatki, je lahko veliko. Zapišimo le nekaj možnih: mogoče zaradi nihanja napetosti v času meritve, lahko je nekdo premaknil (stresel) aparaturo ali pa je enostavno nekdo zapisal napačen podatek.

kot Gaussova; torej takšno, ki predvideva večjo verjetnost pojavljanja točk, za katere napačno predvidevamo, da pripadajo neki krivulji. Primer takšne porazdelitve je dvostrana eksponentna porazdelitev (ang. *double / two-sided exponential distribution*)[16]

$$Verjetnost\{d\} = Verjetnost\{y_i - f(x_i, \mathbf{a})\} \propto e^{-\left|\frac{y_i - f(x_i, \mathbf{a})}{\sigma}\right|}.$$

Verjetnost P sedaj zapišemo kot

$$P \propto \prod_{i=1}^{N} \left(e^{-\left|\frac{y_i - f(x_i, \mathbf{a})}{\sigma}\right|} \Delta y \right). \tag{14}$$

Iz istega razloga, ki nas je popeljal od enačbe (13) do enačbe (12), pridemo iz enačbe (14) do učinkovitega cenilca maksimalne verjetnosti

$$\min_{\mathbf{a}} \sum_{i=1}^{N} |y_i - f(x_i, \mathbf{a})|.$$

Vidimo torej, da je v tem primeru povprečno absolutno odstopanje (ang. *mean absolute deviation*) cenilec maksimalne verjetnosti. Repa porazdelitve sta tukaj asimptotično veliko večja kot pri normalni (Gaussovi) porazdelitvi.

Cena, ki jo moramo plačati zaradi uporabe absolutne vrednosti je, da moramo za iskanje rešitve največkrat uporabiti numerične metode, ki sicer niso potrebne.

[16] Podoben rezultat dosežemo tudi z Cauchyijevo ali Lorentzovo porazdelitvijo.