Razpoznavanje preprostih kontur z uporabo kompleksne FFT

Mirko Robida, absolvent avtomatike
Rovt 24, 3341 Šmartno ob Dreti
RobidaM@Bigfoot.com

***Recognition of simple contours using complex* FFT.**

Abstract: In this paper we described the algorithm, based on Freeman chain code and complex* FFT *for recognition of simple contours from images. The final algorithm for automatic recognition is accomplished with calculating* SNR. *We demonstrate that this algorithm work well, independent from size, position and orientation of an object.

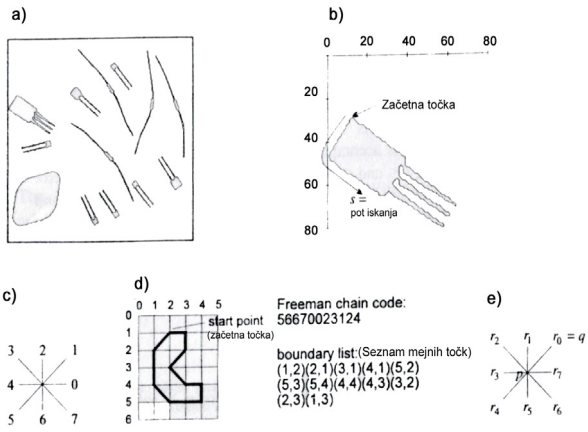
1 Uvod

Obstaja veliko tehnik, kako razpoznati neko konturo (krivuljo, obliko, objekt) na sliki. Disciplina, ki se s temi tehnikami največ ukvarja, je zagotovo disciplina računalniškega vida. Ta skuša oponašati sposobnost človekovega vida. Eden njenih ciljev je iskanje kontur na dvodimenzionalnih slikah. Zaradi vse večje razširjenosti obsežnih slikovnih podatkovnih baz, je inteligentno oz. avtomatsko razpoznavanje in iskanje objektov v njih izredno pomembno. V tem članku smo prikazali uporabo metode Freemanove kode verige (Freeman chain code), ki skupaj s kompleksno hitro Furierjevo transformacijo (FFT) in računanjem razmerja signal/šum (SNR), tvori algoritem za avtomatsko razpoznavanje kontur na sliki.

2 Iskanje konture na sliki

S pomočjo morfoloških operacij je možno pretvoriti sliko nekega področja oz. regije v nabor mejnih točk. Žal pa matematični opis takšnega iskanja ni preprost. Iskanje in opis je enostavnejši, kadar so krivulje zaključene.

Krivuljo v digitalni obliki nekega območja tvori pot po obstoječih mejnih točkah tega območja. Kadar je krivulja podana v digitalni obliki, obstaja veliko metod, kako najti in opisati to pot. Ena izmed teh metod, ki je uporabljena tudi v tem članku, se imenuje Freemanova koda verige (Freeman chain code), slika 1.



Slika 1: Kontura in njena predstavitev

a) mejne točke regij,

b) iskanje konture,

c) definiranje Freeman chain kode,

d) dve predstavitvi konture (s Freeman chain kodo in seznamom mejnih točk),

e) primer maske, uporabljene pri iskanju konture

Sledeči algoritem za določitev liste mejnih točk prične z eno mejno točko (imenovana začetna točka).

Algoritem iskanja konture [1]:

Vhod:

- območje
- začetna točka $p_{\text{init}} = (n_0, m_0)$

1. Kreiramo prazno listo mejnih točk in postavimo začetno točko na vrh te liste.
2. Postavimo $p = p_{\text{init}}$

3. Izmed osmih sosednjih točk p_{init} izberemo točko ki ni točka našega območja in jo označimo s q. Torej je $q \in N_8(p_{\text{init}})$ in $q \notin$ našega območja.
4. Postavimo $q_{\text{init}} = q$.
5. Določimo osem sosednjih točk r_i (kjer je $i = 0,\ldots,7$) točke p tako, da velja:
   - $r_i \in N_8(p_{\text{init}})$.
   - $r_0 = q$.
   - $r_0, r_1, \ldots\ldots, r_7$ je števec zaključene krivulje, orientirane v nasprotni smeri urinega kazalca.
6. Izberemo j tako, da bo:
   - $r_0, \ldots\ldots, r_{j1} \notin$ našega območja
   - $r_j \in$ našega območja.
7. Postavimo $p = r_j$ in $q = r_{j-1}$
8. Dodamo točko p v listo mejnih točk.
9. Če je $(p = p_{\text{init}})$ in $q_{\text{init}} \in \{ r_0,\ldots\ldots, r_{j-1}\}$, končaj.
10. Pojdi na korak 5.

Izhod: urejena lista oz. nabor mejnih točk krivulje.

3 Preslikava v frekvenčni prostor s FFT

Listo mejnih točk lahko zapišemo tudi v parametrični obliki v zveznem področju kot $(x(s), y(s))$. Če je p obseg našega področja, lahko koordinate $(x(s), y(s))$ zapišemo kot dve periodični funkciji s periodo p. Ker je krivulja zaprta oz. zaključena, nima odprtin. Zato sta obe funkciji zvezni.

Krivuljo v ravnini lahko predstavimo tudi kot krivuljo v kompleksni ravnini, z realnim in imaginarnim delom:

$$z(s) = x(s) + jy(s) \qquad (1)$$

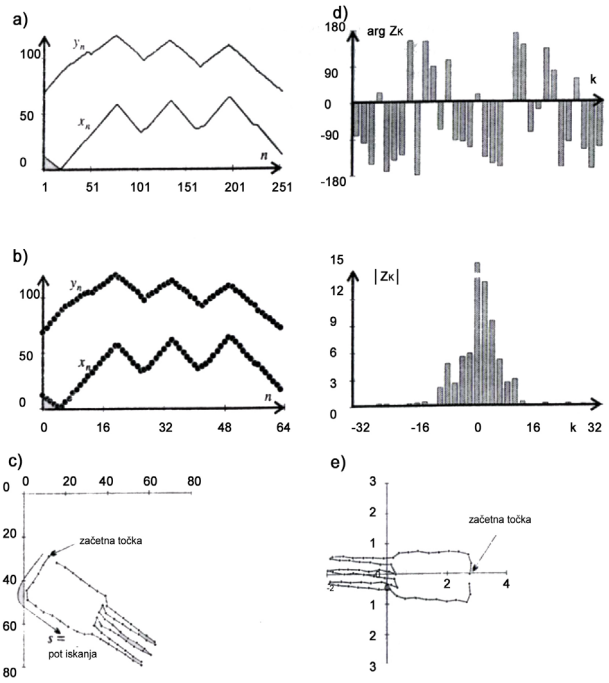
V tem primeru je krivulja kompleksna, periodična in zvezna funkcija. Naša želja je preslikati to krivuljo v frekvenčni prostor (FP). To lahko naredimo s pomočjo kompleksne hitre Furierjeve transformacije (FFT). Najprej moramo krivuljo razdeliti na N enakih intervalov (oziroma jo resemplirati). FFT-algoritem zahteva, da je N potenca števila 2. Dobimo:

$$z_n = z(n) = x(n) + jy(n); \quad n = 0,\ldots,N-1. \qquad (2)$$

Kompleksni spekter računamo s FFT:

$$Z_k = Z(k) = \frac{1}{N} \cdot \mathrm{FFT}(z_n),$$
$$Z_k = \frac{1}{N} \cdot \sum_{n=0}^{N-1} z_n \cdot e^{\frac{-j \cdot k \cdot n \cdot 2 \cdot \pi}{N}}; \quad k = 0,\ldots,N-1 \qquad (3)$$

Spekter običajno rišemo okrog frekvenco 0, od $-N/2-1$ do $N/2$. Zato ga preuredimo.



Slika 2: Fourierjev opis mejnega območja s slike 1b).

a) Lista mejnih točk v parametrični obliki,

b-c) Enakomerno(ekvidistantno) otipana krivulja (N=64),

d) Amplitudni in fazni spekter dobljen iz točk na b),

e) Rekonstruirana krivulja po normalizaciji.

Naslednji korak je normalizacija, kjer izločimo informacije v FFT, ki se nanašajo na velikost, pozicijo, orientacijo in začetno točko krivulje. Normalizacijo pozicije dosežemo s postavitvijo Z_0 na nič. Normalizacijo velikosti, orientacije in začetne točke dosežemo s skaliranjem in vrtenjem krivulje tako, da bo amplituda prvega harmonika Z_1 zavzela vrednost 1, njegova faza pa na nič. Pri normalizaciji začetne točke je nujno, da vključimo tudi ostale harmonike. Normalizacijo

začetne točke zaključimo z njenim pomikom tako, da postane faza harmonika z največjo amplitudo (Z_0 in Z_1 ne upoštevamo), recimo L-tega harmonika, enaka nič. Število rešitev, ki izpoljnjuje fazna pogoja (arg $Z_1 = 0$ in arg $Z_L = 0$), je $|L - 1|$. Samo, če je $L = 2$, je normalizacija orientacije in začetne točke enolična, slika 2.

Sledi algoritem normalizacije, ki sta ga predlagala Wallace in Wintz (1980) [1]:

Vhod: Furierjev spekter Z_k .

1. Postavimo $Z_0 = 0$ (normalizacija pozicije). Z_0 je težišče.
2. Delimo vse vrednosti Z_k z $|Z_1|$ (normalizacija velikosti; Z_1 je merilo velikosti).
3. Najdimo Z_L z največjo amplitudo (Z_0 in Z_1 ne upoštevamo).
4. Izračunamo $\Phi_1 = \arg Z_1$, in $\Phi_2 = \arg Z_L$.
5. Množimo vse Z_k z $e^{j\frac{(k-L)\cdot\phi_1+(1-k)\cdot\phi_2}{L-1}}$.
6. Če je $L = 2$, potem je normalizacija enolična: izhod.
7. Izračunamo funkcijo negotovosti rešitve A, ki je definirana kot: $A = \sum_{k=0}^{N-1} \mathrm{Re}(Z_k)\left|\mathrm{Re}(Z_k)\right|$.
8. Če je bil korak 7 izvršen za vseh $|L-1|$ normalizacij, pojdi na korak 9. Drugače množi vse Z_k z $e^{j\frac{2\pi(k-1)}{L-1}}$; pojdi na korak 7.
9. Izberi normalizacijo, ki da največji A; izhod.

Izhod: normaliziran Furierjev opis Z_k .

4 Razpoznavanje konture na osnovi razmerja signal/šum (SNR)

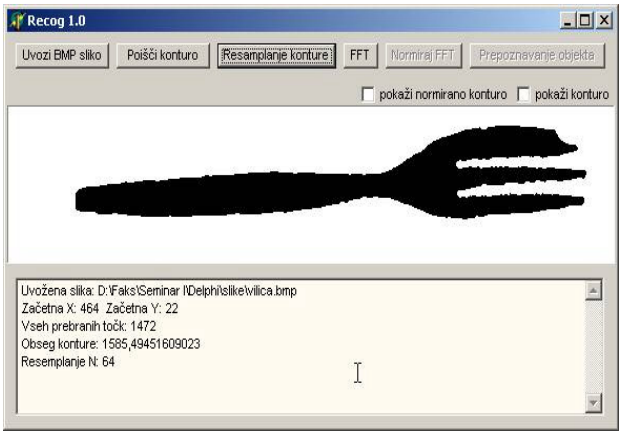
Predstavitev krivulje v frekvenčnem prostoru imamo sedaj normirano. Naslednja naloga je, da razpoznamo obliko oz. kaj krivulja predstavlja. To ugotovimo iz razmerij signal/šum (SNR), izračunanih iz normaliziranih spektrov opazovane krivulje in referenčnih krivulj v bazi. Krivulje, ki jih imamo v bazi, so znane in vsaka predstavlja znano obliko. Izraz za izračun SNR je:

$$SNR = 10 \cdot \log \frac{\sum_{k=0}^{N-1} \left|Zref_k\right|}{\sum_{k=0}^{N-1} \left|Z_k - Zref_k\right|}, \qquad (4)$$

kjer je $Zref_k$ normiran kompleksni spekter referenčne krivulje. Večji je SNR, manjša je razlika med opazovano krivuljo in referenčno krivuljo iz baze. Referenčna krivulja, ki se najbolj ujema z opazovano, bo imela tudi največji SNR. V tem primeru lahko rečemo, da je opazovana krivulja najbližja obliki, ki jo predstavlja referenčna krivulja.

5 Analiza in ugotovitve

S pomočjo programskega paketa Delphi 7.0 smo sestavili aplikacijo Recog 1.0, v kateri smo združili zgoraj omenjene algoritme. Pri izdelavi aplikacije smo poenostavil dve stvari: (i) slike smo zapisali v BMP formatu, (ii) objekte, ki jih prepoznavamo, smo obarvali črno. S tem smo nekoliko poenostavili algoritem za iskanje konture v sliki, slika 3.

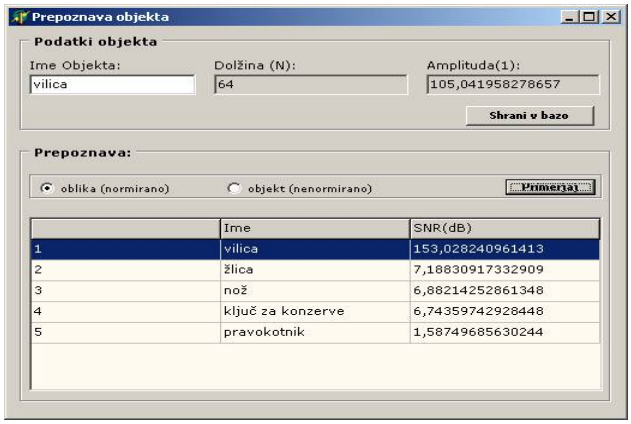


Slika 3: Prikaz vilice v aplikaciji Recog 1.0

Najprej smo morali vnesti v bazo konture, ki predstavljajo referenčne objekte. Na primer, konture jedilnega pribora so na sliki 4.

Slika 4: Objekti, katerih referenčne konture so shranjene v bazi.

Sedaj primerjamo eno od že znanih kontur, npr. vilico, s konturami iz baze. Rezultat vidimo na sliki 5. Največji SNR dobimo pri konturi vilice, kar je tudi razumljivo, saj sta konturi enaki. Vse ostale konture imajo mnogo manjši SNR.

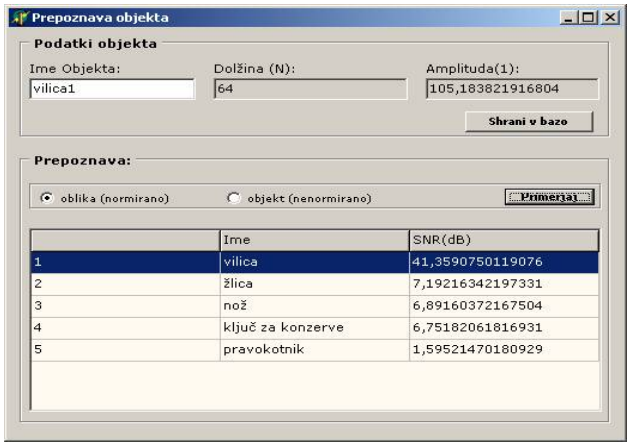


Slika 5: Rezultat primerjave vilice z referenčnimi konturami v bazi.

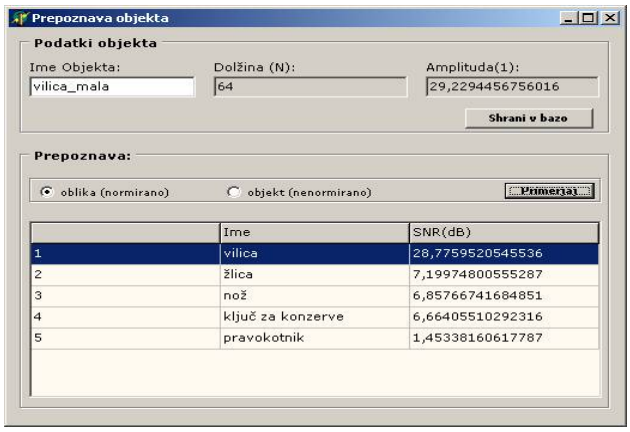
Nato vnesemo sliko vilice, ki smo jo zamaknili in zavrteli za 90°. S tem želimo pokazati, da je algoritem neodvisen od pozicije in orientacije objekta na sliki, slika 6.

Vidimo, da je tudi v tem primeru SNR pri vilici prepričljivo večji kot pri ostalih konturah. V naslednjem preizkusu smo to vilico še pomanjšali na tretjino originalne velikosti. Rezultat razpoznavanja vidimo na sliki 7.


vilica1



Slika 6: Rezultat primerjave zavrtene in premaknjene vilice

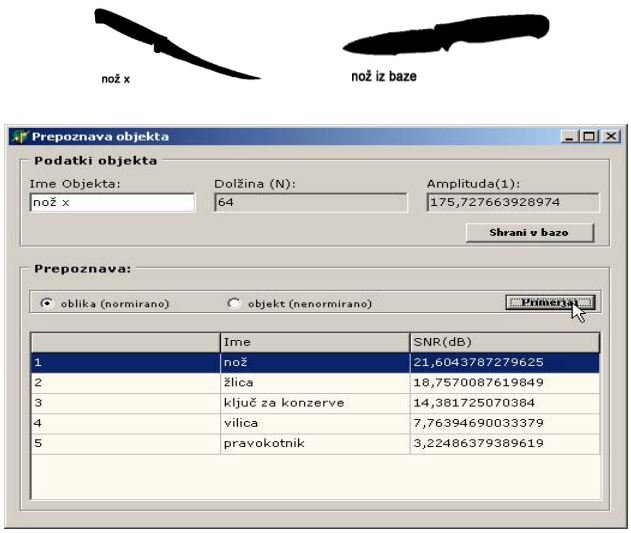


Slika 7: Rezultat primerjave s pomanjšano vilico

Tudi tokrat vidimo, da je SNR pri vilici kar štirikrat večji kot SNR pri žlici, ki je naslednji največji. V zadnjih dveh preizkusih smo pokazali, da je algoritem resnično neodvisen od pozicije, orientacije in velikosti objekta. V vseh treh primerih je bil SNR pri vilici najmanj štirikrat večji kot SNR pri ostalih objektih.

Na koncu naredimo nekoliko zahtevnejši in realnejši test. V praksi se namreč zgodi, da algoritem, ki išče konturo iz slike, te ne najde najbolje oz. je najdena kontura zaradi različnih vplivov motenj »popačena«. Kakšen rezultat

torej dobimo, če uporabim nož, ki je samo podoben tistemu iz baze, glej sliko 8.



Slika 8: Rezultat primerjave z nožem, ki je samo podoben nožu iz baze

Vidimo, da je SNR največji pri nožu, vendar je le za slabe 3dB večji, kot pri naslednji najbolj podobni konturi. Lahko rečemo, da je bil v tem primeru algoritem še uspešen. Pri nekoliko bolj »popačeni« konturi, pa bi lahko dobili že napačne rezultate.

Iz zgornjih testov lahko sklepamo, da je algoritem zelo uspešen, kadar so konture »prave« oblike oz. nepopačene. Če je kontura močno »popačena«, lahko pride do napačnega razpoznavanja, zlasti, ko imamo v bazi več podobnih kontur. Ta problem lahko odpravimo z izboljšanjem algoritma za iskanje konture iz slike ter s povečanjem števila otipkov *N*.

6 Zahvala

Zahvaljujem se dr. Jožetu Mohorku, ki je začel s tem projektom, doc. dr. Petru Planinšiču, in ostalim, ki so omogočili objavo tega članka.

7 Literatura

[1] Ferdinand van der Heijden, *Image Based Measurement Systems*, John Wiley & Sons Ltd., 1994.

[2] Nikola Pavešič, *Razpoznavanje vzorcev. Uvod v analizo in razumevanje vidnih in slušnih signalov*, FE Ljubljana, 2000.

[3] Marco Cantu, *Mastering Delphi 6*, SYBEX, Inc., Almeda, CA, 2001.