

MASARYKOVA UNIVERZITA
FAKULTA INFORMATIKY



Knihovna výpočtů charakteristik 3D objektů

bakalářská práce

Zdeněk Hrnčíř
2004, Brno

Prohlášení

Prohlašuji, že tato práce je mým původním autorským dílem, které jsem vypracoval samostatně. Všechny zdroje, prameny a literaturu, které jsem při vypracování používal nebo z nich čerpal, v práci řádně cituji s uvedením úplného odkazu na příslušný zdroj.

Děkuji svému vedoucímu Mgr. Jiřímu Stiborovi a jeho kolegovi Mgr. Davidu Svobodovi za jejich odbornou pomoc při vytváření této práce.

Shrnutí

Úkolem mé bakalářské práce bylo rozšířit knihovnu funkcí `i3dlib`, která pracuje s diskrétním 3D obrazem a je používána Laboratoří optické mikroskopie. Nově přidané funkce vypočítávají charakteristiky diskrétních 2D a 3D objektů.

Nejprve bylo nutné nastudovat obecnou problematiku digitálního zpracování obrazu — vzorkování, lineární i nelineární filtry, segmentace objektů v obraze, Houghovy transformace a využití Fourierovy transformace. Dále jsem nastudoval problematiku geometrických algoritmů pro práci s vektorovou grafikou. Její principy lze totiž mnohdy uplatnit i v diskrétním obraze.

Vlastní analýza charakteristik objektů je poměrně dobře zpracovaná v mnoha materiálech pro dvojrozměrné objekty. Na poli třírozměrných objektů jsem se však setkal i s výroky smutně konstatujícími, že určitý problém nebyl doposud efektivně vyřešen. Přesto se mi podařilo na použitelné úrovni implementovat všechny podstatné funkce pro popis tělesa - objem, povrch, kulovitost, kvádrovitost, poloměry, směr a velikost hlavních os objektu, protažení, křivost a několik Fourierových deskriptorů.

Klíčová slova:

tvarové charakteristiky, analýza tvaru, diskrétní 3D obraz, voxel, Fourierovy deskriptory.

Obsah

Úvod	1
1 Základní Pojmy	2
Základní pojmy	2
2 Základní charakteristiky objektů	4
Základní charakteristiky ojektů	4
2.1 Obsah/objem	4
2.2 Obvod/povrch	4
2.2.1 2D	4
2.2.2 3D	5
2.3 Kruhovitost/kulovitost	6
2.4 Statistické momenty	6
2.5 Poloměr	7
2.6 Hlavní osy	9
2.7 Protážení	10
2.8 Symetrie	10
2.9 Obdélníkovitost/kvádrovitost	11
2.10 Křivost	12
3 Charakteristiky využívající Fourierovu transformaci	15
Charakteristiky využívající Fourierovu transformaci	15
3.1 2D Fourierovy deskriptory	15
3.2 3D Fourierovy deskriptory	16
4 Implementace	18
Implementace	18
4.1 Objekt Shape	18
4.2 Obsah/objem	19
4.3 Hranice	19
4.4 Obvod/povrch	20
4.5 Kruhovitost/kulovitost	20
4.6 Statistické momenty	20

4.7	Poloměr	21
4.8	Hlavní osy	22
4.9	Protažení	22
4.10	Symetrie	22
4.11	Obdélníkovitost/kvádrovitost	23
4.12	Křivost	23
4.13	Fourierovy deskriptory	24
Závěr		26
Literatura		27
Příloha		A
	Obsah přiloženého CD	A
	Ukázková data	B

Úvod

Laboratoř optické mikroskopie provozuje skener s vysokým rozlišením a přesným zaostřováním, umožňující skenovat pozorované objekty v řezech. Výstupem je sada diskrétních obrazů, reprezentující jednotlivé řezy. Tento skener je používán pro pozorování buněk a celých tkání. Snahou je plně zautomatizovat nejen pořizování digitálních obrazových dat, ale i jejich zpracování a následné vyhodnocení. Podobné systémy se již v medicínské praxi používají a slouží především pro zpracování dat pořízených magnetickou rezonancí (MR) nebo počítačovou tomografií (CT). Mohou sloužit například k vyhledávání nádorů v tkáních. Analýza objektů, která byla mým úkolem, je jedním z článků v řetězci zpracování pořízených dat.

Na úvod stručně popíši základní kroky zpracování obrazu:

- Předzpracování obrazu — filtrování pořízených dat za účelem odstranění šumů, deformací a a chyb snímacího zařízení.
- Segmentace obrazu — rozpoznání a separace jednotlivých naskenovaných objektů, přesné určení hranic objektů.
- Popis objektů — z jednotlivých objektů jsou získány důležité charakteristiky. Ty musí být jednoduché a cílené na dílčí vlastnosti objektů. Dobře navržené charakteristiky umožní objekty přesně klasifikovat.
- Klasifikace objektů — řazení objektů do tříd podle jejich charakteristik. V podstatě se jedná o použití statistických metod klasifikace.
- Porozumění obrazu — vyvození závěrů o pořízených datech, získání cílových dat z celého obrazu.

V dalším textu se budu zabývat metodami pro získávání charakteristik objektů a jejich efektivními implementacemi.

První kapitola je věnována vysvětlení používaných pojmů. Druhá a třetí kapitola se zabývá základními charakteristikami a přístupy založenými na Fourierově transformaci. Poslední kapitola se věnuje detailnějšímu popisu implementace. Většinou budu nejprve popisovat metody pro dvojrozměrné objekty a následně jejich upravené verze pro třírozměrné objekty.

Kapitola 1

Základní Pojmy

Nejprve bych vás rád seznámil se základními pojmy a obecnými předpoklady použitými v dalším textu.

Diskrétní 3D obraz, voxel — diskrétní 3D obraz vznikne poskládáním jednotlivých řezů (diskrétních 2D obrazů) na sebe. Je reprezentován třírozměrnou sítí voxelů, což jsou třírozměrné ekvivalenty pixelů. Voxel může mít stejné atributy jako pixel. Pro reprezentaci objektů se nejčastěji používají binární obrazy. Binární hodnota voxelu určuje, zda se jedná o voxel objektu, nebo voxel pozadí. Pro reprezentaci většího množství různých objektů v jednom obraze se používají šedotónní voxely. Množina voxelů jednoho objektu pak má stejný šedotónní index. Na voxely lze nahlížet dvojím způsobem. Buď jako na body v síti, nebo jako na krychličky, z nichž je obraz poskládán. Oba pohledy mají své uplatnění.

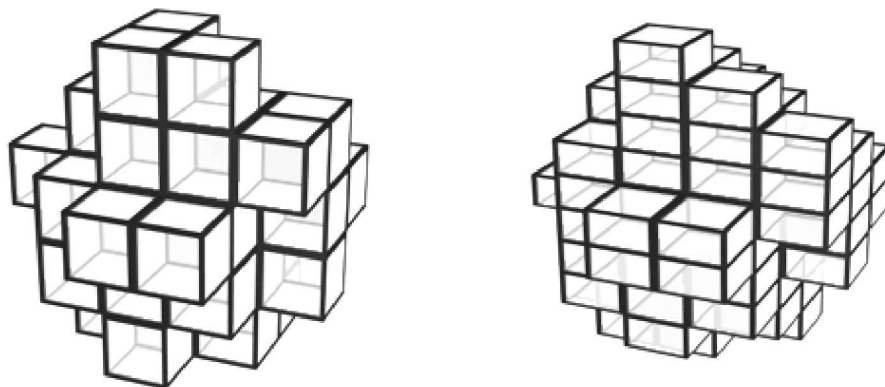
Rozlišení — počet voxelů na mikrometr ve všech třech hlavních osách. Rozlišení určuje rozměry (případně rozteč) voxelů. Rozlišení nemusí být isometrické. Často jsou jednotlivé řezy naskenovány ve větších vzdálenostech, než je rozteč bodů v jednom řezu. Neisometrické rozlišení přidává některým algoritmům na složitosti a je podrobněji diskutováno.

Sousedství, sousedé — podobně jako jsou v diskrétním 2D obraze zavedeny pojmy 4-sousedství a 8-sousedství, v 3D figurují hlavně 6-sousedství, 18-sousedství a 26-sousedství. Obecně je sousedstvím množina vektorů. Tyto vektory určují polohy voxelů - sousedů od daného voxelu. toho

Spojité objekt — množina voxelů, navíc souvislá (spojitá) vzhledem k nějakému sousedství (zpravidla 6-sousedství pro 3D a 4-sousedství pro 2D). Tzn. že pro libovolné dva voxely objektu existuje posloupnost $n_1 \dots n_k$ voxelů patřících objektu a platí, že $n_{[i]}$ je v sousedství $n_{[i+1]}$. Pak hovoříme o např. o 6-spojitém 3D objektu. Model jednoduchého 6-spojitého objektu je na obrázku 1.1.

Jednoduchý objekt — spojitý objekt bez "děr". Tj takový spojitý objekt, kde voxely pozadí tvoří opět spojitý objekt. Za voxely pozadí jsou v tomto případě považovány i voxely mimo obraz.

Hranice — hranice, nebo také hraniční voxely, se vztahují k určitému sousedství, které ji generuje. Hranice je množina voxelů objektu, které mají v daném sousedství alespoň jeden voxel nepatřící objektu. Ve 2D obraze figuruje hlavně pojem "spojitost hranice". Ta je pro jednoduché objekty vyjádřena minimálním sousedstvím, vzhledem k němuž jsou hraniční voxely spojité. Spojitost hranice je tedy závislá na spojitosti objektu a na sousedství, které ji generuje. Budu-li ve 2D uvažovat nejčastěji používané 4-spojité objekty,



Obrázek 1.1: Voxelový model jednoduchého 6-spojitého objektu. Vlevo s isometrickým, vpravo a neisometrickým rozlišením (řezy jsou hustší).

generující sousedství	spojitost hranice
6-sousedství	18-spojité
18-sousedství	6-spojité
27-sousedství	6-spojité

Tabulka 1.1: Vztah mezi spojitostí hranice a okolím, které ji generuje, 6-spojitého 3D objektu

pak 4-sousedství generuje 8-spojitou hranici a naopak. Podobné, nikoli symetrické, vztahy pro 6-spojité 3D objekt shrnuje Tabulka 1.1. Pro jednoznačnost v dalším textu jako typ hranice vždy uvádím okolí, které ji generuje, ačkoli v literatuře zabývající se 2D obrazem bývá zvykem uvádět spojitost.

V dalších kapitolách budu předpokládat, že na vstupu je obraz, který je výsledkem předchozího zpracování obrazu – segmentace:

Binární obraz g o rozměrech $X \times Y \times Z$ reprezentující jediný 6-spojité objekt a $g(x, y, z) = 1$ pro voxely patřící objektu, $g(x, y, z) = 0$ pro voxely nepatřící objektu (voxely pozadí).

Respektive pro 2D případ binární obraz g o rozměrech $X \times Y$ reprezentující 4-spojité objekt a $g(x, y) = 1$ pro pixely patřící objektu, $g(x, y) = 0$ pro pixely pozadí.

Kapitola 2

Základní charakteristiky objektů

2.1 Obsah/objem

Obsah/objem je základní a nejjednodušší charakteristika objektu. Nic však nevypovídá o jeho tvaru. Jeho výpočet je nezbytný pro získání dalších charakteristik. Definice je podle [1]¹ následující:

2D:

$$S = \sum_{x=0}^{X-1} \sum_{y=0}^{Y-1} g(x, y)$$

3D:

$$V = \sum_{x=0}^{X-1} \sum_{y=0}^{Y-1} \sum_{z=0}^{Z-1} g(x, y, z)$$

V podstatě se jedná o prostý součet všech voxelů v objektu.

2.2 Obvod/povrch

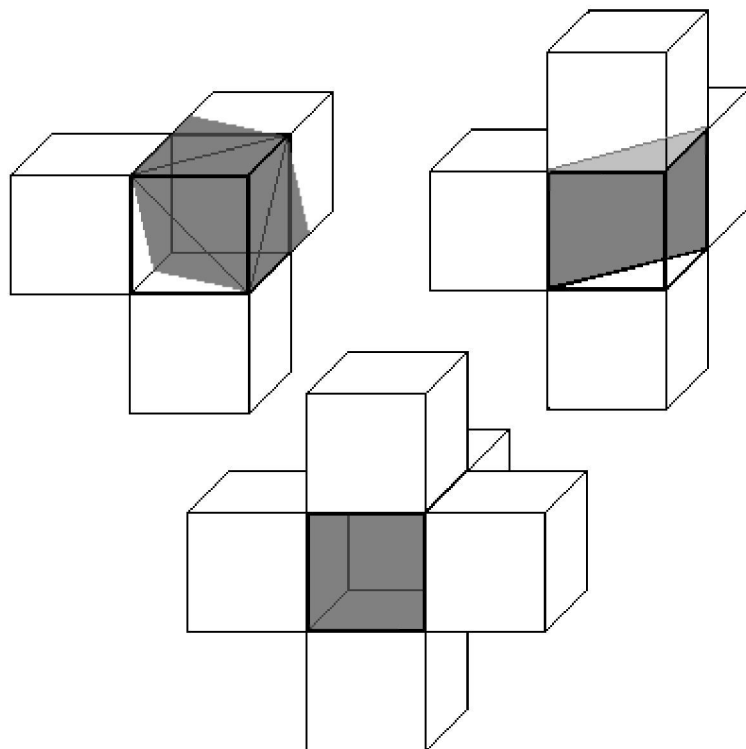
Obvod/povrch je opět základní charakteristikou objektu, sám o sobě nic nevypovídá o jeho tvaru.

2.2.1 2D

Nejjednodušším přístupem k výpočtu je sečtení pixelů na hranici objektu. Problémem je, že pixely ležící diagonálně jsou od sebe více vzdáleny, než pixely ležící ve stejném řádku/sloupci. Možností je použít 8-spojitou hranici převedenou do řetězcové reprezentace. To je seznam, kde za sebou následují sousední pixely a ty navíc "obcházejí" objekt ve směru hodinových ručiček.² Pak při sekvenčním procházení takové hranice určíme směr, ve kterém leží následující sousedé. Obvod je aproximován:

¹téměř všechny definice charakteristik pro dvojrozměrné objekty v této kapitole jsem převzal z knihy *Shape analysis and classification*.

²K algoritmu pro vytvoření takové reprezentace se ještě v následujícím textu vrátím.



Obrázek 2.1: Aproximace povrchu voxelu v závislosti na jeho okolí

$$O = N_r + N_d * \sqrt{2} \quad (2.1)$$

kde N_d je počet pixelů ležících diagonálním směrem a N_r je počet pixelů ležících směrem rovnoběžným s osami obrazu. Nevýhody tohoto přístupu stále zůstávají ve vysoké míře nepřesnosti. Je to dáno tím, že při výpočtu je použito velmi malé okolí zkoumaných bodů hranice.

2.2.2 3D

Analogicky je možné sečíst voxely ležící na hranici, avšak chyba je v tomto případě opravdu značná. Reprezentace hranice pomocí řetězce navíc pro třírozměrné objekty není možná. Řešení nabízejí postupy pro převedení diskrétního (voxelového) objektu do vektorové (polygonální) reprezentace. Tyto postupy nahrazují jednotlivé voxely v závislosti na jejich okolí jedním či více polygony. Je zřejmé, že čím větší okolí je použito, tím větší je šance na přesnost. Naproti tomu je ale nutné ošetřit větší množství variací. Já jsem implementoval aproximaci povrchu na základě 6-sousedství. To skýtá 9 základních konfigurací (ostatní získáme jejich rotacemi), z nichž některé jsou na obrázku 2.1 i s vyobrazením jejich aproximací povrchu.

Pro neisometrické rozlišení je počet základních konfigurací mnohonásobně vyšší. Proto jsem v implementaci ošetřil zvlášť všech 64 možných kombinací okolí.

průměr koule	hranice podle okolí			aproximace podle 3D-6 okolí
	3D-6	3D-18	3D-26	
24	77,9	115,4	130,4	104,0
50	81,1	122,3	141,6	106,9
100	82,0	125,9	145,9	107,8
200	82,5	127,4	148,1	108,4
Průměrná chyba	-19,1%	+22,8%	+41,5%	+6,8%

Tabulka 2.1: Přesnost jednotlivých metod pro výpočet povrchu

Tabulka 2.1 shrnuje přesnost jednotlivých metod. Modelem byly vygenerované koule o průměrech 24, 50, 100 a 200 voxelů. Ve sloupcích jsou povrchy vypočtené jednotlivými metodami v procentuálním poměru vzhledem k očekávaným hodnotám. Je vidět, že aproximace dává výrazně lepší výsledky, než prostý součet povrchových voxelů. Se zvětšujícím se měřítkem mírně narůstá chyba. To je důsledkem použití konstantního sousedství, které při vzrůstajícím měřítku více zachycuje šum a méně celkový tvar objektu. V praxi se ale neobjevují mnohonásobně rozměrnější objekty, u kterých by mohla chyba dále růst, neboť jejich zpracování je až příliš paměťově i časově náročné.

2.3 Kruhovitost/kulovitost

Kruhovitost/kulovitost je první charakteristikou nezávislou na měřítku objektu. Nejvíce kruhovitý/kulovitý objekt je přirozeně kruh/koule a ten se vyznačuje nejnižším poměrem obvodu ku obsahu. (resp. povrchu ku objemu). Proto je také kulovitost definována následovně:

$$2D: c = \frac{o^2}{S} \quad 3D: c = \frac{S^3}{V^2}$$

Nejnižší hodnoty je pro kruh dosaženo 4π , pro kouli 36π .

2.4 Statistické momenty

Z pohledu statistiky je voxelový objekt množina bodů v $\mathbf{N}^3$, na kterou lze aplikovat libovolnou statistickou metodu. Statistické metody použité pro popis objektů tak nejsou omezeny jen na objekty, ale je možné jimi charakterizovat libovolnou množinu voxelů. Základním statistickým měřítkem jsou statistické momenty, které lze velice snadno vypočítat podle následujících vzorců:

2D:

$$m_{r,s} = \sum_{x=0}^{X-1} \sum_{y=0}^{Y-1} x^r y^s g(x, y)$$

3D:

$$m_{r,s,t} = \sum_{x=0}^{X-1} \sum_{y=0}^{Y-1} \sum_{z=0}^{Z-1} x^r y^s z^t g(x, y, z) \quad (2.2)$$

Všimněte si, že výpočet obsahu/objemu je speciálním případem momentu, přesněji $S = m_{0,0}$, $V = m_{0,0,0}$.

Poloha těžiště je další charakteristikou, kterou lze získat ze statistických momentů. Pro 2D je těžiště definováno jako $T = (\bar{x}, \bar{y})$, kde:

$$\bar{x} = \frac{m_{1,0}}{m_{0,0}} \quad \bar{y} = \frac{m_{0,1}}{m_{0,0}}$$

Pro 3D je těžiště definováno analogicky jako $T = (\bar{x}, \bar{y}, \bar{z})$, kde:

$$\bar{x} = \frac{m_{1,0,0}}{m_{0,0,0}} \quad \bar{y} = \frac{m_{0,1,0}}{m_{0,0,0}} \quad \bar{z} = \frac{m_{0,0,1}}{m_{0,0,0}} \quad (2.3)$$

Výpočet hlavních os a symetrií vyžadují, aby byl střed souřadnicové soustavy umístěn v těžišti, proto se zavádějí tzv. centrální momenty:

2D:

$$\mu_{r,s} = \sum_{x=0}^{X-1} \sum_{y=0}^{Y-1} (x - \bar{x})^r (y - \bar{y})^s g(x, y)$$

3D:

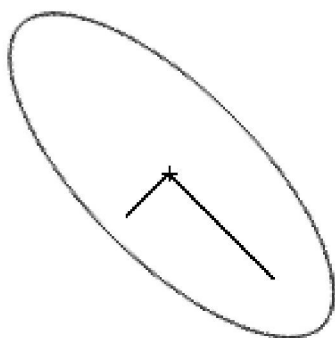
$$\mu_{r,s,t} = \sum_{x=0}^{X-1} \sum_{y=0}^{Y-1} \sum_{z=0}^{Z-1} (x - \bar{x})^r (y - \bar{y})^s (z - \bar{z})^t g(x, y, z) \quad (2.4)$$

2.5 Poloměr

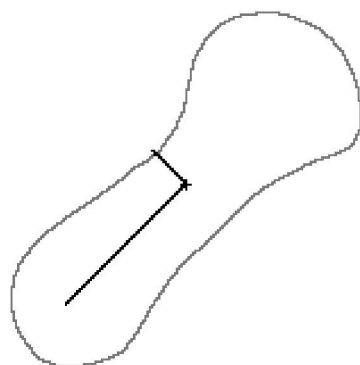
Poloměr je vzdálenost mezi středem (těžištěm) objektu a jeho hranicí. Pro obecný objekt je tato vzdálenost proměnlivá v závislosti na zvoleném hraničním voxelu. Můžeme počítat tyto statistické charakteristiky:

- Maximální poloměr
- Minimální poloměr
- Průměrný poloměr
- Střední hodnota poloměru

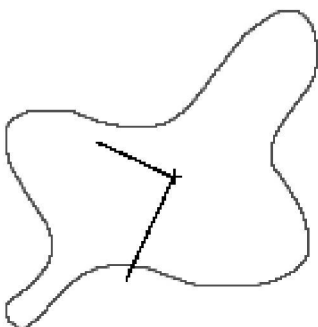
a)



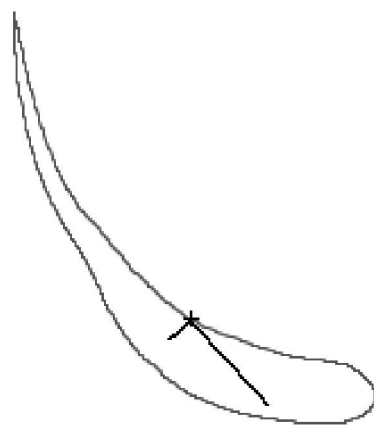
b)



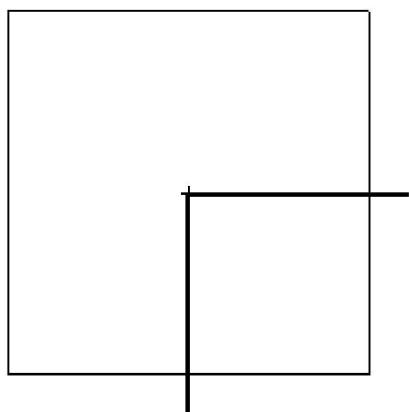
c)



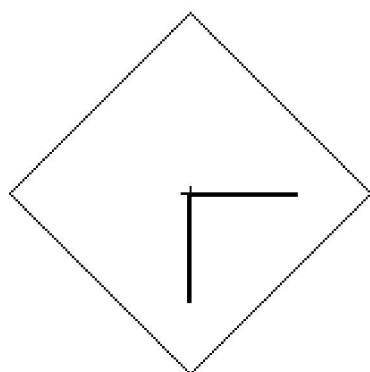
d)



e)



f)



Obrázek 2.2: Hlavní osy, pro přehlednost jsou vyobrazeny pouze hranice objektů a délka os odpovídá odmocninám vlastních hodnot

2.6 Hlavní osy

Geometrický význam hlavních os je zřetelný na obrázcích 2.2 a—d. Hlavní osy jsou navzájem kolmé. Pro třídimenzionální objekty přibude třetí osa, která je kolmá na zbylé dvě. Získání velikosti a směru hlavních os objektu je základním předpokladem pro výpočet dalších charakteristik. Mimo to můžeme objekty transformovat tak, aby střed souřadnicové soustavy ležel v těžišti a hlavní osy objektu byly rovnoběžné s osami souřadnicové soustavy. Takto transformované objekty můžeme přímo porovnávat "voxel proti voxelu", tedy například výpočtem korelace transformovaných objektů. Chceme-li porovnávat tvary a nikoli velikost objektů, stačí je transformovat změnou měřítka, aby měly stejný objem. Toto měřítko podobnosti objektů může být velice přesné, ale vzhledem k časové náročnosti je zcela nepoužitelné pro vzájemné srovnávání většího počtu objektů. Proto jsou na výpočtu hlavních os založeny další jednodušší charakteristiky.

Výpočet hlavních os je úkolem statistické metody PCA (Principal Component Analysis), která je popsána mj. v [3]. Pro výpočet velikosti a směru hlavních os se využívá vlastností kovariance. Centrální momenty druhého řádu (tj. takové, kde $r + s + t = 2$) vyjadřují kovarianci mezi dvěma ze složek x, y, z všech bodů. Tak například μ_{110} je kovariance mezi x a y , μ_{200} je variance x . Kovariance vyjadřuje symetrii a právě to je jeden z účelů hlavních os. Centrální momenty jsou počítány z toho důvodu, že nás zajímá symetrie vzhledem k těžišti objektu nikoli k počátku souřadné soustavy. Potom můžeme sestavit následující kovarianční matice:

2D:

$$\mathbf{C} = \begin{pmatrix} \mu_{20} & \mu_{11} \\ \mu_{11} & \mu_{02} \end{pmatrix}$$

3D:

$$\mathbf{C} = \begin{pmatrix} \mu_{200} & \mu_{110} & \mu_{101} \\ \mu_{110} & \mu_{020} & \mu_{011} \\ \mu_{101} & \mu_{011} & \mu_{002} \end{pmatrix}$$

Velikost a směr hlavních os odpovídá vlastním vektorům a vlastním hodnotám matice $\mathbf{C}$. Vektor přidružený s největší hodnotou je hlavní osou objektu. Vektory s nižšími vlastními hodnotami pak postupně odpovídají vedlejším osám. Matice řádu $n \times n$ má n vlastních hodnot a přidružených vektorů, což odpovídá dimenzi obrazu. Vlastní hodnota λ matice $\mathbf{C}$ a jí odpovídající vlastní vektor $\mathbf{z}$ jsou definovány rovnicí:

$$\mathbf{C}\mathbf{z} = \lambda\mathbf{z} \quad \lambda \neq 0$$

Kuriózní případ můžete vidět na obrázcích 2.2 f, g. Jedná se v obou případech o čtverce a přesto mají směr hlavních os vzhledem ke své poloze v obraze rozdílný. Důvodem je větší počet os symetrie čtverce a charakter výpočtu hlavních os, který ve sporných případech preferuje směr rovnoběžný s osami obrazu. Tento nedostatek se projeví pouze u dokonale symetrických tvarů. V praxi stačí nepatrně změnit poměr stran a výsledný směr bude vždy vzhledem k poloze objektu stejný.

2.7 Protažení

Protažení (elongation), jak název napovídá, popisuje celkové rozložení objektu v prostoru. Po kulovitosti je dalším deskriptorem nezávislým na měřítku. Kulovité objekty mají hodnotu protažení nejnižší, opakem jsou dlouhé (protáhlé) objekty. V dvourozměrném případě je protažení definováno jako poměr velikostí hlavní a vedlejší osy objektu a je tedy vyjádřitelné jedním číslem. Pro třírozměrné objekty jsem protažení definoval následovně:

$$E_1 = \frac{v_1}{v_2}$$

$$E_2 = \frac{v_2}{v_3}$$

kde v_1, v_2, v_3 jsou sestupně uspořádané velikosti hlavních os.

Následující výčet shrnuje interpretace možných výsledků hodnot E_1, E_2 .

$E_1, E_2 \rightarrow 1$	kulovité rozložení v prostoru
$E_1 \gg 1 \quad E_2 \rightarrow 1$	celkově tyčovitý tvar (např. elipsoid o poloosách 2,1,1)
$E_1 \rightarrow 1 \quad E_2 \gg 1$	celkově diskovitý tvar (např. elipsoid o poloosách 2,2,1)
$E_1 \approx E_2 \quad E_1, E_2 \gg 1$	zcela disproporční objekt
$E_1, E_2 \gg 1 \quad E_1 \neq E_2$	obecný disproporční objekt

2.8 Symetrie

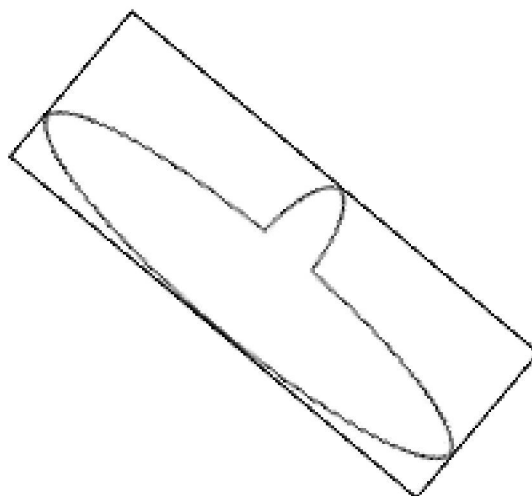
V [1] je popsána bilaterální symetrie. Míra symetrie je vyjádřena pomocí počtů voxelů jednotlivých intenzit ve složeném obraze, který vznikne sečtením původního obrazu g se symetricky transformovaným obrazem g' . Ve výsledném šedotónním obraze se objeví voxely intenzit 0–pozadí, 1–asymetrické části objektu a 2–symetrické části objektu. Pak míra symetrie je vyjádřena jako poměr

$$\frac{N_2}{N}$$

kde N je počet voxelů intenzit 1 nebo 2, N_2 je počet voxelů intenzit 2.

Je-li objekt podle dané osy/roviny plně symetrický (tj. $g = g'$), pak se ve složeném obraze vyskytují pouze voxely intenzit 0 a 2, z toho plyne, že $N = N_2$ a míra symetrie je rovna jedné. Naopak pro zcela nesymetrický objekt (tj. $\forall x, y, z : g(x, y, z) = 0 \vee g'(x, y, z) = 0$) se ve složeném obraze nevyskytují vyšší hodnoty než 1, pak $N_2 = 0$ a míra symetrie je nulová.

Nejdůležitějšími osami symetrie objektu jsou jeho hlavní osy procházející těžištěm. Je to z toho důvodu, že jejich výpočet je postaven na kovarianci. Ta sama o sobě vyjadřuje symetrii. Hlavní osy objektu jsou vlastně i hlavními osami symetrie. A přirozeně hlavní roviny symetrie jsou ty, které jsou tvořeny hlavními osami.



Obrázek 2.3: Minimální obklopující obdélník (MER)

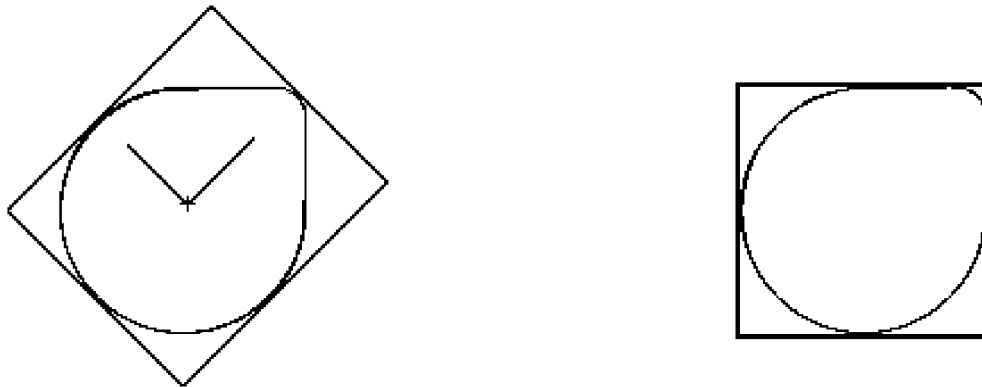
2.9 Obdélníkovitost/kvádrovitost

Tyto charakteristiky jsou analogií ke kruhovitosti/kulovitosti. Obdélníkovitost/kvádrovitost vyjadřuje jak moc se objekt tvarem blíží ideálnímu obdélníku/kvádru. Jejich výpočet je založen na znalosti minimálního obklopujícího obdélníku/kvádru (MER — Minimal Enclosing Rectangle). Ukázka minimálního obklopujícího obdélníku je na obrázku 2.3. Obdélníkovitost/kvádrovitost je definována jako poměr:

$$2\text{D: } \frac{S}{S_o} \quad 3\text{D: } \frac{V}{V_k}$$

kde S, V jsou plocha resp. objem objektu, S_o je plocha minimálního obklopujícího obdélníku a V_k je objem minimálního obklopujícího kvádru. Pro obdélník je obdélníkovitost rovna jedné, pro ostatní objekty se pohybuje v intervalu $(0, 1)$.

Jedním z přístupů, jak přibližně určit MER je nejprve vypočítat hlavní osy. Hledaný MER je minimální obdélník/kvádr, který má hrany rovnoběžné s hlavními osami a obsahuje celý objekt. Ve skutečnosti takto nalezený MER nemusí být minimální, protože hlavní osy jsou vyjádřením symetrie a orientace objektu, nikoli jeho tvaru. Typickým příkladem, kdy tento přístup selhává je čtverec na obrázku 2.2 g. Hlavní osy směřují úhlopříčně, takže takto definovaná obdélníkovitost je rovna jedné polovině. Kosočtverce budou zatíženy podobnou chybou, i když menší. Chyba se může objevit i u kulovitých objektů. Příklad je na obrázku 2.4. Obecně je tento přístup dost nepřesný. Přesnější MER by bylo možné získat otestováním obklopujících obdélníků/kvádrů ve všech polohách a zvolit ten nejmenší. To by ale vedlo k vysoké časové náročnosti, hlavně u třírozměrných objektů.



Obrázek 2.4: MER — vlevo získaný pomocí hlavních os, vpravo skutečný minimální obklopující obdélník

2.10 Křivost

Křivost je jednou z nejproblematictějších částí analýzy diskretních objektů. Vztahuje se ke křivkám a povrchům a vyjadřuje míru jejich zakřivení v určitém bodě. Pro analýzu objektů je proto zajímavá křivost hranice. Voxely s vysokou křivostí odpovídají vrcholům a výběžkům na hranici, voxely s nulovou křivostí jsou součástí rovných úseků hranice. Podle [1] je křivost $k(t)$ parametrické křivky $c(t)=(x(t),y(t))$ definována jako:

$$k(t) = \frac{\dot{x}(t)\ddot{y}(t) - \ddot{x}(t)\dot{y}(t)}{(\dot{x}(t)^2 + \dot{y}(t)^2)^{3/2}} \quad (2.5)$$

kde $\dot{x}$, $\ddot{x}$ značí první a druhou derivaci x . Právě jejich získání je problematické v případě diskretních křivek. Metodám pro jejich výpočet, které jsem vyzkoušel, se budu podrobněji věnovat ve čtvrté části o implementaci.

Na tomto místě ještě zmíním alternativní definici křivosti, která je vhodná pro detekci vrcholů. Je definována pro diskretní křivku $c(n) = (x(n), y(n))$ pomocí vektorů, které v sousedních intervalech aproximují tečnu ke křivce:

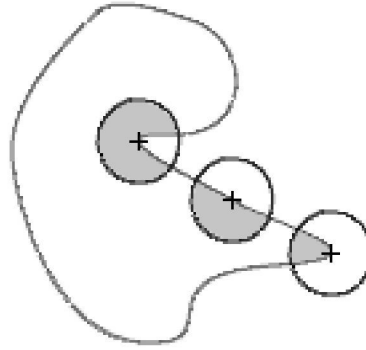
$$v_i(n) = c(n) - c(n - i)$$

$$w_i(n) = c(n) - c(n + i)$$

Vlastní křivost je pak definována jako kosinus úhlu těchto vektorů:

$$r_i(n) = \frac{v_i(n)w_i(n)}{|v_i(n)||w_i(n)|} \quad (2.6)$$

Veličina i reprezentuje měřítko. Jeho volba určuje úroveň detailů, které má křivost zachytit. Automatický odhad měřítka je předmětem dalších metod popsanych v [1]. Výsledek křivosti se pohybuje mezi hodnotami -1 pro rovné úseky a 1 pro extrémní body (vrcholy i prohlubně).



Obrázek 2.5: Alternativní křivost definovaná objemem hmoty objektu v kruhovém/kulovém okolí.

Jedním z deskriptorů, které lze z křivosti odvodit je "ohybová energie":

$$B = \frac{1}{N} \sum_{n=0}^{N-1} k(n)^2 \quad (2.7)$$

kde N je počet hraničních voxelů. Její normalizovaná verze, nezávislá na měřítku je:

$$B_n = \frac{L^2}{N} \sum_{n=0}^{N-1} k(n)^2 = L^2 B \quad (2.8)$$

kde L je obvod objektu.

Třírozměrné objekty a výpočet křivosti s sebou přinášejí potřebu parametrizace povrchu. Tuto otázku řeším v následující kapitole. V knize [4] jsem objevil alternativní přístup spočívající v určování křivosti povrchového voxelu v závislosti na jeho nejbližším okolí. Popisovaná metoda se mi jevila příliš náchylná k "voxelovému šumu" a proto jsem se pokusil navrhnout metodu vlastní, která by byla vhodná k detekci extrémních bodů.

Vycházím z empirického pohledu na objekty, kde vysoce zakřivené ("ostré") konvexní vrcholy lze charakterizovat nízkým objemem hmoty objektu v blízkém okolí a konkávní vrcholy naopak vyšším objemem hmoty v blízkém okolí. Body s nulovou křivostí jsou součástí roviny, která rozděluje okolí na dvě poloviny a proto mají poloviční poměr hmoty objektu vzhledem k objemu okolí. Dvourozměrná verze této myšlenky je zobrazena na obr. 2.5. Na hranici objektu jsou vyznačeny tři body s různou křivostí, jejich kruhové okolí a objem hmoty objektu v tomto okolí. Křivost v bodě p pak definuji jako:

$$k(p, r) = \frac{V(p, r)}{V_o(r)} - 1/2$$

kde $V(p, r)$ je objem objektu v kulovém sousedství o poloměru r se středem v bodě p a $V_o(r)$ je objem kulového sousedství o poloměru r . Jak je vidět, jedná se o víceměřítkovou

charakteristiku závislou na poloměru kulového okolí. Při implementaci se však tato metoda projevila jako příliš časově náročná.

Kapitola 3

Charakteristiky využívající Fourierovu transformaci

Fourierova transformace je mocným a často používaným nástrojem při zpracování digitalizovaných dat. Pro popis objektů se používají především Fourierovy deskriptory, což jsou charakteristiky získané ve frekvenční doméně pomocí Fourierovy transformace. K tomu, aby mohla být transformace na data aplikována, je nutno z nich nejprve extrahovat nějakou funkci.

3.1 2D Fourierovy deskriptory

Pro dvojrozměrné binární obrazy lze typicky použít dvojrozměrnou funkci souřadnic v obraze, jejíž hodnoty jsou intenzity pixelů ($g : (x, y) \rightarrow \{0, 1\}$). Výstupem diskrétní Fourierovy transformace funkce g je opět dvojrozměrná funkce s komplexním oborem hodnot. Tato funkce však pro popis objektu není vhodná. Transformovaná data mají větší objem než vstupní obraz a pro tvarovou analýzu objektu jsou špatně čitelná, protože se vztahují k celému obrazu.

Lepším přístupem je extrahovat nějakou funkci z hranice objektu. Dříve zmíněný seznam hraničních pixelů uspořádaný ve směru hodinových ručiček je ideálním kandidátem. Tento seznam lze vyjádřit jako funkci $u : n \rightarrow x + yi$, kde n je pořadí pixelu v seznamu o N prvcích a x, y jsou souřadnice pixelu. Pro získání invariance vzhledem k posuvu a rotaci je nezbytné souřadnice pixelů přepočítat tak, aby těžiště bylo umístěno v bodě $(0,0)$. Výsledek diskrétní Fourierovy transformace této funkce můžeme přímo prohlásit za jeden z Fourierových deskriptorů, definovaný následovně:

$$FD(s) = \frac{1}{N} \sum_n^N u(n) e^{-j2\pi ns/N}, \quad s = 0, \dots, N-1 \quad (3.1)$$

Kde j je imaginární jednotka a koeficienty nejnižších frekvencí jsou pro $s = 0$ a $s = N-1$. Pro další zpracování je častěji používána posunutá transformace, kde $s = -\lfloor N/2 \rfloor, \dots, 0, \dots, N-1 - \lfloor N/2 \rfloor$ a koeficienty nejnižších frekvencí jsou pro s blízko nuly. V dalším textu budu uvažovat takto posunutou transformaci.

Jednou z výhod frekvenční domény je možnost odfiltrování vysokých frekvencí, které v obraze odpovídají drobným nerovnostem na hranici objektů a v neposlední řadě "pixelovému šumu".¹

Jednodušším Fourierovým deskriptorem, který vychází z předchozího deskriptoru, je "akumulovaná energie". Tento deskriptor počítá pouze s amplitudami spojenými s nízkými frekvencemi — hlavními nositeli informace o tvaru objektu:

$$E(s) = \sum_{i=0}^s |FD(i)|^2 \quad (3.2)$$

s určuje počet uvažovaných frekvencí a s rostoucím s se energie postupně stabilizuje. Všimněte si, že tento deskriptor již pracuje jen s vlnovými amplitudami. Amplitudy v případě jednodušších kulovitých objektů při vyšších frekvencích rychle klesají k nule. Energie se v tomto případě stabilizuje už pro nízká s . Tento deskriptor je závislý na měřítku objektu.

Další deskriptory vycházejí z normalizované Fourierovy transformace. Normalizace má za úkol dále sjednotit deskriptory objektů v různém měřítku a je definována takto:

$$\begin{aligned} NFD(s) &= 0 \quad \text{pro } s = 0 \\ NFD(s) &= FD(s)/|FD(1)| \quad \text{pro } s \neq 0 \end{aligned}$$

Odtud je odvozen jednodušší normalizovaný deskriptor:

$$FF = \frac{\sum_{s=-\lfloor N/2 \rfloor}^{N-\lfloor N/2 \rfloor+1} |NFD(s)|/|s|}{\sum_{s=-\lfloor N/2 \rfloor}^{N-\lfloor N/2 \rfloor+1} |NFD(s)|} \quad (3.3)$$

Jeho hodnota se pohybuje v intervalu $\langle 0, 1 \rangle$. A nejvyšších hodnot dosahuje pro kompaktní, málo členité objekty.

Podobně můžeme definovat i normalizovanou energii:

$$NE = \sum_{s=-\lfloor N/2 \rfloor}^{N-\lfloor N/2 \rfloor+1} |NFD(s)|^2 \quad (3.4)$$

Z hranice objektu můžeme extrahovat i další funkce. Jednou z nich je vzdálenost povrchu objektu od jeho těžiště v závislosti na obvodovém úhlu ($u_d : \varphi \rightarrow d$). Tato funkce však přináší problémy v podobě složitějších objektů, kdy pro konkrétní úhel může paprsek protnout hranici vícekrát a vzdálenostní funkce musí být nějakým způsobem aproximována. Tento přístup tak vlastně zahrnuje množinu funkcí.

3.2 3D Fourierovy deskriptory

Pro třírozměrné objekty jsem narazil na dosud největší problém — extrakce funkce z objektu, jejíž Fourierova transformace by byla smysluplná. Povrch objektu zřejmě nemá smysl zapisovat do jednorozměrné funkce (To by muselo být realizováno po řezech a záleželo by tedy na poloze objektu v obraze). Pokus o reprezentaci povrchu dvourozměrnou rovinou

¹Na tomto principu je založena jedna z metod pro určení křivosti.

funkcí ztroskotává na faktu, že povrchové voxely mají proměnlivý počet sousedních povrchových voxelů. Rozumnou možností je použít polární souřadnice, vést z těžiště objektu pravidelně rozmístěné paprsky do prostoru a sledovat vzdálenost protnutého povrchu od těžiště. Tím získáme sférické funkce ($u_s : (\varphi, \psi) \rightarrow d$). Tyto funkce jsou zobecněním funkcí u_d zmíněných v předchozí části. Fourierovou transformací této funkce by měly vzniknout tzv. "sférické harmonické vlny". Bohužel jsem ve výzkumných zprávách na Internetu nenalezl podrobnější informace k implementaci, přesné definici funkce a její transformace a využití. K samotným sférickým harmonickým vlnám se vztahuje poměrně složitá teorie. Implementace třírozměrných Fourierových deskriptorů proto spadá za hranice této práce.

Kapitola 4

Implementace

4.1 Objekt Shape

Nejen pro pohodlí uživatele knihovny, ale hlavně z důvodů efektivity, jsem zapouzdřil všechny funkce do objektu **Shape**, který je inicializován pro konkrétní obraz. Aplikační programátor volá funkce nad instancí objektu **Shape**, a proto nemusí předávat obraz v parametrech funkcí. Objekt **Shape** si pamatuje důležité charakteristiky a konstrukce, které nad daným obrazem počítal. Jelikož většina charakteristik je základního typu, není potřeba paměti nijak zatěžující. Náročnější nebo často volané funkce si nejprve ověří, zda předtím neprováděly stejný výpočet. Pokud ano, vrátí dříve spočítanou hodnotu uloženou v objektu. Tato konstrukce zajišťuje, že se náročné výpočty budou provádět tehdy a jen tehdy, když je to nezbytné. Programátorovi to umožňuje volat funkce v libovolném pořadí a množství bez ztrát na efektivitě. I v samotné knihovně se jednotlivé funkce často volají navzájem.

Objekt sám nemůže kontrolovat, zda obraz se kterým je referenčně svázán nebyl změněn, to by popíralo snahu o maximální efektivitu. Je na odpovědnosti programátora, aby při případné změně v obraze zavolal funkci **EraseValues**. Neočekávám, že se vyskytne mnoho případů, kdy bude této funkci zapotřebí. Pro opakované použití objektu pro více obrazů je k dispozici funkce **SetImage**, která se zároveň postará o vymazání starých uložených charakteristik.

Většina implementovaných charakteristik je ve dvou verzích — pro dvourozměrná a třírozměrná data. Datová struktura třírozměrného obrazu **Image3d** je navržena tak, aby mohla uchovávat i dvourozměrné obrazy. Pak má obraz rozměry $X \times Y \times 1$ a každý voxel¹ má souřadnici Z nulovou. To umožňuje větší univerzálnost — obě verze charakteristiky obsluhuje stejná funkce.

V případě neisometrického měřítka vyžadují složitější charakteristiky provádění výpočtů nikoli v souřadnicích voxelů, ale v mikronech. Proto jsou základní charakteristiky jako objem, povrch a statistické momenty dále rozděleny na základní verze a "mikronové" verze, které mají v názvu funkce příponu **Um**.

U každé charakteristiky se pozastavím nad asymptotickou časovou složitostí. Zde budu pro přehlednost a s ohledem na praxi uvažovat vstupní data rozměrů $X = Y = Z = n$,

¹V popisu funkcí budu vždy hovořit o voxelech i když se bude jednat o pixely. V implementaci mezi nimi není žádný rozdíl a z kontextu bude zřejmé v jaké dimenzi se pohybujeme.

resp. $X = Y = n, Z = 1$. Lineární průchod trojrozměrnými daty tak bude mít časovou složitost n^3 .

4.2 Obsah/objem

Výpočet obsahu/objemu zajišťují následující funkce.

- **Volume** vrací počet voxelů objektu, což je ekvivalentem objemu. Pro dvourozměrný obraz je počet voxelů ekvivalentem plošného obsahu.
- **VolumeUm** vrací obsah/objem přepočítaný podle rozlišení na $\mu m^2/\mu m^3$.
- **Area** je čistě dvourozměrná verze počtu voxelů v objektu. Ve skutečnosti volá obecnější funkci **Volume**. Zavedl jsem ji pro jasnější pojmenování a přehlednost.
- **AreaUm** Podobně jako předchozí funkce je zavedena pro přehlednost.

Algoritmy jsou velice jednoduché. Jsou realizovány sekvenčním procházením daty. Složitost je n^3 pro 3D a n^2 pro 2D.

4.3 Hranice

Hranice, seznam voxelů na hranici, je získána funkcí **GetBoundary**, která volá specializovanější funkce **GetBoundary2d** a **GetBoundary3d**.

- **GetBoundary3d** prochází sekvenčně obraz a podle daného sousedství vrací voxely, které jsou na hranici. Je optimalizována pro implicitní 6-sousedství, ale umožňuje konstruovat hranici podle libovolného sousedství. Asymptotická složitost je opět n^3 .
- **GetBoundary2d** tato funkce vrací hraniční voxely seřazené po směru hodinových ručiček. Princip algoritmu spočívá v nalezení počátečního hraničního voxelu. Prohledáním jeho sousedství ve směru hodinových ručiček nalezneme další hraniční voxel, jehož sousedství prohledáváme, a tak pokračujeme dokud objekt neobejdeme k počátečnímu voxelu. Důležité je pamatovat si směr, ve kterém přecházíme na sousední hraniční voxel, abychom se nevraceli zpět. Podle prohledávaného sousedství může být hranice 4-spojité (generovaná 8-sousedstvím), nebo implicitní 8-spojité (generovaná 4-sousedstvím). Ve výsledné posloupnosti se mohou některé voxely vyskytnout vícekrát, což není chyba. Dojde k tomu např. v místech, kde jsou objekty široké jeden voxel.
Složitost může dosahovat až n^2 , nenarazíme-li při hledání prvního voxelu na žádný objekt, nebo je-li objekt příliš členitý.

Seznam hraničních voxelů pro implicitní sousedství je taktéž uložen v objektu **Shape**, přestože jeho velikost nelze zanedbat. V obecném případě by mohla asymptotická prostorová složitost odpovídat množství vstupních dat, ve 2D by tomu odpovídala jednovoxelová svislá mřížka spojená vodorovným pruhem. Zde bych chtěl upozornit na očekávaná data,

pro která jsem funkce optimalizoval. Terčem Laboratoře optické mikroskopie a mým vstupem jsou jednotlivé buňky. U nich lze očekávat převážně kulovitý, nebo alespoň kompaktní tvar s minimálním množstvím děr. Takové objekty budou mít asymptoticky velikost hranice n^2 , respektive n ve 2D. Hranice je využita v několika dalších funkcích, které jsou pak časově mnohem efektivnější.

4.4 Obvod/povrch

Základním, ač velice nepřesným, přístupem k získání obvodu/povrchu je počet povrchových voxelů. Funkce **Surface** a **Perimeter** jen volají **GetBoundary** a zjišťují velikost jejího výstupu. Funkce **Surface** s cílem univerzálnosti v případě dvoudimenzionálního vstupu volá **Perimeter**.

Přesnější funkce, které navíc počítají i s neisometrickým měřítkem jsou následující.

- **PerimeterUm** počítá obvod v mikrometrech. Funkce sekvenčně prochází 8-spojité hraniční voxely získané funkcí **GetBoundary** a určuje počty voxelů ležících v diagonálním a ortogonálním směru od svých sousedních předchůdců. Pro isometrické rozlišení lze přímo použít vzorec 2.1 na str. 5. Pro neisometrické rozlišení je třeba přepočítat diagonální a ortogonální vzdálenost mezi sousedy.
- **SurfaceUm3d** počítá povrch v μm^2 . Pro každou kombinaci hodnot sousedů v 6-sousedství (včetně rotací) je v závislosti na měřítku lineárně aproximován povrch libovolného voxelu, kterému konkrétní kombinace sousedů odpovídá. Následně funkce všechny hraniční voxely získané funkcí **GetBoundary** zařadí do příslušné kategorie podle jejich sousedů. Počty voxelů v jednotlivých kategoriích jsou vynásobeny odpovídajícím povrchem a sečteny dohromady.

V obou případech je složitost samotných funkcí bez volání **GetBoundary** lineární vzhledem k velikosti hranice. Pokud byla předtím hranice vygenerována, pak pro celý algoritmus platí časová složitost odpovídající prostorové složitosti hranice v 4.3.

4.5 Kruhovitost/kulovitost

Tato charakteristika je vypočtena funkcí **Roundness** podle vzorců v kapitole 2.3 na straně 6. Pro výpočet jsou použity hodnoty povrchu a objemu v mikrometrech.

4.6 Statistické momenty

Statistické momenty a těžiště jsou počítány podle vzorců v části 2.4 na straně 6. Všechny funkce jsou sestaveny pro třírozměrná data, ale dokáží pracovat i s dvourozměrnými daty, jak jsou popsána v 4.1. Stačí na vstupu funkcí třetí koeficient ponechat roven nule a výsledkem je dvourozměrná varianta vzorců. Funkce jsou následující:

- **StatMoment** — základní statistické momenty podle vzorce 2.2 ve voxelových souřadnicích.

- **StatMomentUm** — pro neisometrické rozlišení je nutné přepočítat souřadnice na mikrony. Statistické momenty se pak vypočítají podle vzorce:

$$\text{StatMomentUm}(\mathbf{r}, \mathbf{s}, \mathbf{t}) = \sum_{x=0}^{X-1} \sum_{y=0}^{Y-1} \sum_{z=0}^{Z-1} (xR_x)^r (yR_y)^s (zR_z)^t g(x, y, z)$$

kde R_x, R_y, R_z jsou rozteče mezi voxely v osách x, y, z . Tento vzorec jsem upravil a vyjádřil pomocí funkce **StatMoment**:

$$\text{StatMomentUm}(\mathbf{r}, \mathbf{s}, \mathbf{t}) = R_x^r R_y^s R_z^t \sum_{x=0}^{X-1} \sum_{y=0}^{Y-1} \sum_{z=0}^{Z-1} x^r y^s z^t g(x, y, z)$$

$$\text{StatMomentUm}(\mathbf{r}, \mathbf{s}, \mathbf{t}) = R_x^r R_y^s R_z^t \text{StatMoment}(\mathbf{r}, \mathbf{s}, \mathbf{t})$$

- **Centroid** vrací souřadnice těžiště, které počítá podle rovnic 2.3. Těžiště ve voxelových souřadnicích je nezávislé na rozlišení. Pokud by to neplatilo, museli bychom použít funkci **StatMomentUm** a výsledek poté přepočítat na voxelové souřadnice, což by pro x -ovou souřadnici těžiště vypadalo následovně:

$$\bar{x} = \frac{\text{StatMomentUm}(1, 0, 0)}{\text{StatMomentUm}(0, 0, 0)} \cdot \frac{1}{R_x}$$

Podle předchozího bodu lze rovnici vyjádřit pomocí funkce **StatMoment**.

$$\begin{aligned} \bar{x} &= \frac{R_x \text{StatMoment}(1, 0, 0)}{R_x \text{StatMoment}(0, 0, 0)} \\ \bar{x} &= \frac{\text{StatMoment}(1, 0, 0)}{\text{StatMoment}(0, 0, 0)} \end{aligned}$$

To odpovídá výpočtu pro isometrické rozlišení. Pro ostatní souřadnice je vyjádření obdobné.

- **CStatMoment** — Centrální statistické momenty podle vzorce 2.4.
- **CStatMomentUm** — Centrální statistické momenty v mikronech. Jejich výpočet lze analogicky jako u funkce **StatMomentUm** vyjádřit pomocí **CStatMoment**.

Vypočtené základní statistické momenty pro $r, s, t < 10$ jsou uchovávány v objektu **Shape**. Jejich hodnoty v mikronech pak funkce **StatMomentUm** a **CStatMomentUm** vypočítají v konstantním čase.

Časová složitost pro malá r, s, t je lineární vzhledem k objemu dat.

4.7 Poloměr

Funkce **Radius** ve svých parametrech vrací maximální, minimální a průměrný poloměr. Všechny hodnoty jsou v mikronech. Využívá funkci **GetBoundary** podobně jako funkce pro výpočet obsahu/obvodu. Její časová složitost tomu odpovídá.

4.8 Hlavní osy

Výpočet hlavních os je podrobně popsán v části 2.6 a je zajišťován funkcí `PrincipalAxes`. Jen k tomu doplním, že v závislosti na rozlišení jsou použity buď standardní statistické momenty, nebo v mikronech. Výstupní vektory, uspořádané sestupně podle velikosti, odpovídají souřadnicím v mikronech. Dvourozměrná verze je speciálním případem třírozměrné, a proto je délka její nejkratší osy rovna nule. Pro výpočet vlastních vektorů a hodnot jsem použil knihovnu LAPACK (Linear Algebra PACKage).

4.9 Protažení

Protažení jak je definováno v části 2.7 zajišťuje funkce `Elongatin`. Opět je dvourozměrná verze spec. případem třírozměrné a jako druhou (vedlejší) hodnotu protažení vrací `infinity`.

4.10 Symetrie

Výpočet bilaterální symetrie popsáný v části 2.8 není nejvhodnější na implementaci, protože vyžaduje vytvoření nového šedotónního obrazu, což je poměťově příliš náročné.

Výpočet lze interpretovat i jinak:

míra symetrie s vzhledem k dané rovině/ose je vyjádřena jako poměr

$$\frac{N_s}{N_s + 2N_a}$$

kde N_s je počet symetrických voxelů objektu, tj. voxelů jejichž symetrický obraz odpovídá voxelu z objektu, a N_a je počet asymetrických voxelů objektu.

Počty symetrických a asymetrických voxelů lze zjistit přímo postupnými transformacemi jednotlivých bodů a ověřením jejich symetrických protějšků v původním obraze.

Symetrický obraz bodu je získán posloupností afinních transformací:

1. změna měřítka (nezbytné pro neisometrické rozlišení)
2. posun osy/roviny do počátku souř. soustavy
3. rotace – srovnání normály osy/roviny s osou x
4. symetrie podle osy x / roviny yz
5. zpětná rotace
6. zpětný posun
7. zpětná změna měřítka

Přesný tvar transformačních matic viz. [2].

Funkce `Symmetry` má tyto vstupní parametry:

1. typ symetrie — osová, nebo rovinná

2. střed symetrie

3. vektor — normálový pro rovinnou symetrii / směrový pro osovou symetrii

Funkce nejprve vypočte transformační matici, s jejíž pomocí následně pro každý voxel objektu najde příslušný symetricky sdružený voxel. Transformační matice je vždy sestavena s ohledem na rozlišení. Nemá smysl optimalizovat výpočet pro voxelové souřadnice, neboť transformační matice v praxi téměř nikdy nevyjde celočíselně. Výpočet pro dvourozměrná data je speciálním případem výpočtu pro třírozměrná, proto je transformační matice sestavena identickým způsobem. Samozřejmě u dvourozměrných dat nemá smysl uvažovat typ symetrie, ta je v konečném důsledku vždy osová.

Doba transformace jednoho bodu je konstantní, celková časová složitost je tak lineární vzhledem k objemu dat.

Funkce `PrincipalSymmetry` počítá hlavní symetrie. Tato funce volá `Symmetry` a jako střed symetrie volí těžiště a normálami rovin symetrie jsou hlavní osy. Výsledkem jsou tři hodnoty symetrie. Pro dvourozměrná data je třetí hodnota, přidružená s nulovou hlavní osou, vždy rovna jedné.

4.11 Obdélníkovitost/kvádrovitost

Je počítána, jak bylo uvedeno v části 2.9, s pomocí hlavních os funkcí `Rectangularity`, která volá funkce `Rectangularity2d` a `Rectangularity3d`. Jádrem problému je hledání maximálních vzdáleností voxelů objektu od těžiště ve směru hlavních os. To pro třírozměrné objekty znamená počítat vzdálenosti bodů od roviny a pro dvojrozměrné objekty počítat vzdálenosti bodů od přímky. Funkce nejprve vypočítají obecné rovnice přímek (rovin), které procházejí těžištěm a jsou kolmé na hlavní osy. Poté pro každý voxel objektu určí jeho vzdálenost v obou směrech od všech přímek (rovin). Ze zapamatovaných maximálních vzdáleností je pak vypočítán obsah (objem) přiléhajícího obdélníku (kvádru) a výsledná obdélníkovitost (kvádrovitost). Výpočty jsou prováděny v mikronových souřadnicích. Jak jsem uvedl v teoretické části, může tento přístup vykazovat značné nepřesnosti. Implementovaná funkce by proto měla být používána jako doplňující charakteristika pro podobné objekty, nikoli jako hlavní kritérium klasifikace.

4.12 Křivost

Jako první přístup pro určení křivosti dvojrozměrných objektů jsem ve funkci `Curvature2d` implementoval metodu popsanou rovnicí 2.6 na straně 12. Celočíslným parametrem funkce je měřítko určující míru detailů. Malé měřítko vede k šumu, velké naopak k ignorování malých detailů. Pokud je měřítko větší než čtvrtina délky hranice, nemá výpočet smysl. Funkce vrací seznam křivostí, který koresponduje se seznamem hraničních voxelů, vrácený funkcí `GetBoundary`.

Další dva přístupy uvedené v [1] se snaží aproximovat první a druhou derivaci v rovnici 2.5. První z nich počítá derivace z jednoho sousedního voxelu. Výpočet je příliš omezený a nepřesný. Druhý přístup vychází z aproximace úseků hranice B-spliny. S odkazem na další literaturu jsou v [1] uvedeny tato rovnice:

$$k = 2 \frac{c_1 b_2 - c_2 b_1}{(c_1^2 + c_2^2)^{3/2}} \quad (4.1)$$

$$b_1 = \frac{1}{12}((x_{n-2} + x_{n+2}) + 2(x_{n-1} + x_{n+1}) - 6x_n)$$

$$b_2 = \frac{1}{12}((y_{n-2} + y_{n+2}) + 2(y_{n-1} + y_{n+1}) - 6y_n)$$

$$c_1 = \frac{1}{12}((x_{n+2} - x_{n-2}) + 4(x_{n+1} + x_{n-1}))$$

$$c_2 = \frac{1}{12}((y_{n+2} - y_{n-2}) + 4(y_{n+1} + y_{n-1}))$$

Rovnice 4.1 je analogií ke klasické definici křivosti a počítá křivost v bodě n . Koeficienty b_1 , b_2 , c_1 , c_2 mají odpovídat derivacím $\ddot{x}$, $\ddot{y}$, $\dot{x}$, $\dot{y}$ aproximovaným B-spliny. Souřadnice voxelů jsou vztaženy k těžišti. Tuto křivost jsem implementoval ve funkci **BSCurvature2d** a její výsledky jsou poměrně dobré. Proto jsem ji použil i pro implementaci ohybové energie definované rovnicí 2.7 ve funkci **BendingEnergy2d**. Funkce **NBendingEnergy2d** počítá normalizovanou ohybovou energii. Nikoli však podle rovnice 2.8, protože při praktických testech na kruhových objektech různých poloměrů se výsledky podstatně lišily. Normalizovanou ohybovou energii počítám přímo pomocí funkce **NBSCurvature2d**, která souřadnice voxelů přepočítává vzhledem k aktuálně zkoumanému voxelu a stává se tak přesnější obdobou funkce **Curvature2d**.

Všechny algoritmy pracují v lineárním čase vzhledem k velikosti hranice

4.13 Fourierovy deskriptory

Z důvodů uvedených v teoretické části jsem implementoval pouze charakteristiky pro dvourozměrné objekty. Pro získání 8-spojité hranice jsem použil funkci **GetBoundary**. Fourierovu transformaci provádí knihovna funkcí FFTW (Fastest Fourier Transform on the West), která uvádí časovou složitost vždy $n \log n$ vzhledem k objemu vstupních dat. Diskrétní Fourierova transformace FFTW však není počítána podle vzorce 3.1 na straně 15, ale podle jeho nenormalizované verze:

$$FD(s) = \sum_n^N u(n) e^{-j2\pi ns/N}, \quad s = 0, \dots, N-1$$

Výstup funkce proto normalizuji $1/N$ a provádím posun:
 $s = -\lfloor N/2 \rfloor, \dots, 0, \dots, N-1 - \lfloor N/2 \rfloor$.

Funkce počítající Fourierovy deskriptory jsou následující:

- **LPEnergy2d** — Low Pass Energy (nízkofrekvenční energie) je deskriptor popsáný rovnicí 3.2. Jeho vstupní parametr určuje počet prvních koeficientů Fourierovy transformace, ze kterých je energie počítána. Vzhledem k tomu, že tento deskriptor je

závislý na měřítku, jsou souřadnice hraničních voxelů vždy přepočítány na mikrony. Fourierova transformace, ani výsledný deskriptor nejsou uchovávány v objektu **Shape**. Jednak je deskriptor parametrizován, jednak předpokládám častější využití následujících normalizovaných deskriptorů, jejichž výpočet je nezávislý.

- **NFDs2d** — Normalizované Fourierovy Deskriptory. Tato funkce vrací ve svých parametrech dva normalizované deskriptory popsané rovnicemi 3.3 a 3.4. První z nich souvisí s kompaktností objektu, druhý je zmíněná normalizovaná energie. Vzhledem k normalizaci jsou souřadnice voxelů převáděny na mikrony pouze pro neisometrické obrazy. Oba deskriptory jsou uchovávány v objektu **Shape**.

Jelikož jsem použil rychlou fourierovu transformaci a výpočet deskriptorů z transformované funkce je lineární vzhledem k velikosti hranice, je očekávaná složitost $n \log n$, nejhůře $n^2 \log n$ pro hodně členitou hranici.

Závěr

Hlavním úkolem této bakalářské práce bylo implementovat charakteristiky 2D a 3D objektů reprezentovaných diskretním obrazem.

Vycházel jsem především z knihy *Shape analysis and classification* [1], která mimo jiné popisuje principy výpočtů charakteristik pro dvourozměrné objekty. K rozšíření pro třírozměrné objekty jsem používal převážně Internetové články a zprávy, které se touto problematikou zabývají.

Při samotné implementaci jsem kladl velký důraz na efektivitu, neboť objem zpracovávaných dat je obrovský. To bylo hlavním důvodem, proč jsem některé funkce zaváděl ve více verzích pro různé typy vstupních dat. Časová náročnost mě vedla mimo jiné k zapouzdření funkcí do objektu **Shape** s opakovaným použitím vypočítaných hodnot. Paměťová náročnost vyvolala potřebu zobecnění výpočtů i pro neisometrické obrazy, které při skenování objektů po řezech vznikají.

Nejnáročnější bylo ošetřit funkce tak, aby dokázaly efektivně pracovat i s neisometrickými obrazy a aby bylo možné porovnávat isometrické objekty s neisometrickými.

Vzhledem k rozsahu oblasti analýzy objektů a množství alternativních charakteristik jsem v této práci obsáhl pouze část problému. Přesto by naimplementované charakteristiky měly být dostačující pro úspěšnou klasifikaci objektů, které jsou pořizovány v Laboratoři optické mikroskopie.

Literatura

- [1] Costa, Luciano da Fontoura - Cesar, Roberto Marcondes, *Shape analysis and classification: theory and practice*, Boca Raton : CRC Press, 2001
- [2] Jiří Žára, Bedřich Beneš, Petr Felkel, *Moderní počítačová grafika*, Computer Press, Praha, 1998
- [3] Lindsay I. Smith, *Principal component analysis*, University of Otago, New Zealand,
http://www.cs.otago.ac.nz/cosc453/student_tutorials/principal_components.pdf
- [4] Gilles Bertrand, Atsushi Imiya, Reinhard Klette, *Digital and image geometry: advanced lectures*, Berlin : Springer, 2001.

Příloha

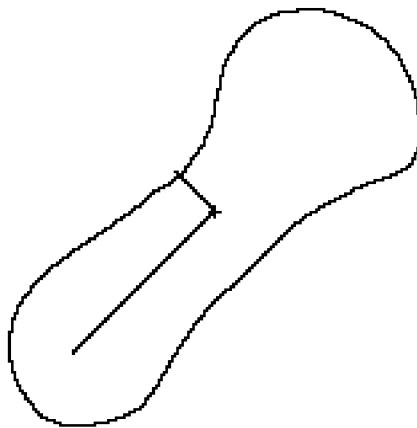
Obsah přiloženého CD

Na přiloženém CD jsou umístěny následující materiály:

- Tento dokument v PDF formátu
- Zdrojové materiály tohoto dokumentu.
- Zdrojové soubory knihovny I3Dlib.
- Ukázková data.

Ukázková data

Následuje ukázka několika dvourozměrných objektů a jejich charakteristik.



rozlišení: $10 \times 10 \times 10$

obsah v μm^2 : 105,1

obvod v μm : 49,6

kruhovitost: 23,4

protažení: 10,2

obdélníkovitost: 0,678

hlavní symetrie: 0,80

vedlejší symetrie: 0,91

ohybová energie: $2,28 \cdot 10^{-5}$

normalizovaná ohybová energie: 290

Fourierův deskriptor FF: 0,82

normalizovaná energie: 1,23



Tento brázek je v ose x zmenšenou verzí předchozího obrázku. Při odpovídající změně rozlišení jsou výsledky přibližně stejné. Rozdíly jsou způsobeny především ztrátou informace při zmenšení obrázku.

rozlišení: $5 \times 10 \times 10$

obsah v μm^2 : 106,7

obvod v μm : 52

kruhovitost: 25

protažení: 9,95

obdélníkovitost: 0,69

hlavní symetrie: 0,81

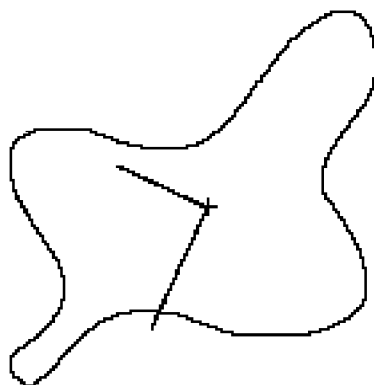
vedlejší symetrie: 0,84

ohybová energie: $9,93 \cdot 10^{-5}$

normalizovaná ohybová energie: 307

Fourierův deskriptor FF: 0,76

normalizovaná energie: 1,13



rozlišení: $10 \times 10 \times 10$

obsah v μm^2 : 98,5

obvod v μm : 53,9

kruhovitost: 29,5

protažení: 2,21

obdélníkovitost: 0,49

hlavní symetrie: 0,53

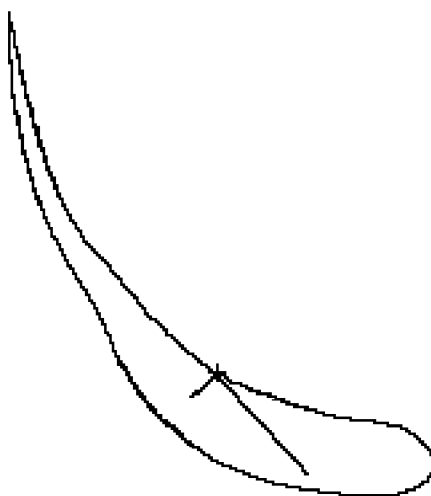
vedlejší symetrie: 0,38

ohybová energie: $2,59 \cdot 10^{-6}$

normalizovaná ohybová energie: 282

Fourierův deskriptor FF: 0,68

normalizovaná energie: 1,12



rozlišení: $10 \times 10 \times 10$
obsah v μm^2 : 48,34
obvod v μm : 56,6
kruhovitost: 66
protažení: 14,3
obdélníkovitost: 0,27
hlavní symetrie: 0,385
vedlejší symetrie: 0,184
ohybová energie: 3,84
normalizovaná ohybová energie: 265
Fourierův deskriptor FF: 0,775
normalizovaná energie: 1,67514