Univerzita Komenského v Bratislave

Fakulta matematiky, fyziky a informatiky

Rozoznávanie orientácie riasiniek

bakalárska práca

2015

Mária Vajdová

Univerzita Komenského v Bratislave

Fakulta matematiky, fyziky a informatiky

Rozoznávanie orientácie riasiniek

bakalárska práca

Študijný program:	Informatika
Študijný odbor:	2508 Informatika
Školiace pracovisko:	Katedra informatiky
Školiteľ:	RNDr. Michal Forišek PhD.

Bratislava, 2015

Mária Vajdová

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Mária Vajdová
Študijný program: informatika (Jednoodborové štúdium, bakalársky I. st., denná forma)
Študijný odbor: 9.2.1. informatika
Typ záverečnej práce: bakalárska
Jazyk záverečnej práce: slovenský
Sekundárny jazyk: anglický

Názov: Rozpoznávanie orientácie riasiniek
Recognizing orientation of cilia

Cieľ: Cieľom práce je spolupracovať s lekármi skúmajúcimi využitie riasiniek pri diagnostike chorôb a vyvinúť softvér, ktorý im umožní zefektívniť a spresniť túto diagnostiku. Softvér bude spracúvať a vyhodnocovať mikroskopické zábery riasiniek. Tie je potrebné v obrázku zdetekovať, zistiť ich orientáciu a získané merania vhodne agregovať.

Vedúci: RNDr. Michal Forišek, PhD.
Katedra: FMFI.KI - Katedra informatiky
Vedúci katedry: doc. RNDr. Daniel Olejár, PhD.

Dátum zadania: 21.10.2014

Dátum schválenia: 28.10.2014

doc. RNDr. Daniel Olejár, PhD.
garant študijného programu

.....
študent

.....
vedúci práce

Poďakovanie: Chcela by som sa poďakovať Mišofovi za vedenie práce a cenné rady k bakalárke aj do života. Magde za uvedenie do sveta riasiniek a pomoc s biologickou časťou práce.

Sysľovi za rôzne rady k implementácii aplikácie, mamine za gramatické korekcie, posledným dvom menovaným, zvyšku mojej rodiny a Kike za podporu a rozptyľovanie od písania :).

Abstrakt

Hlavným cieľom bakalárskej práce je vytvorenie aplikácie, ktorá je schopná rozoznávať orientácie riasiniek z fotografií zostrojených mikroskopom. Riasinky sú bunkové organely, nachádzajúce sa na povrchu eukaryotických buniek. Aplikácia vznikla na podnet lekárov z oddelenia pľúcnych chorôb Pediatrickej kliniky fakultnej nemocnice Motol v Prahe, ktorí sa zaoberajú skúmaním riasiniek a ich orientácie. Súčasťou aplikácie je aj grafické užívateľské rozhranie, ktoré sprevádza používateľa rozoznávaním a vyhodnocovaním fotografií riasiniek. V práci sme sa ďalej zaoberali aj motiváciou skúmania riasiniek, existujúcimi metódami spracovania obrazu alebo štatistickými metódami vyhodnocovania výsledkov z meraní orientácií. Nakoniec v práci testujeme existujúcu aplikáciu pri bežnom používaní.

Kľúčové slová: riasinka, detekcia orientácie, spracovanie obrazu

Abstract

We develop an application for distinguishing cilial orientation in microscope images. The cilia are surface organelles of eukaryotic cells. We introduce the motivation for studying cilia, cover relevant statistical methods for orientation measurement and survey current approaches to image processing. The application's user friendly GUI for distinguishing and evaluating cillial images will benefit medical professionals studying cilial orientations of cells involved in pulmonary disease at the department of pulmonary diseases, Clinic of Pediatry, Motol University Hospital in Prague. We conclude with an evaluation of the application's performance on genuine cilial images.

Keywords: cilia, detection of orientation, image processing

Obsah

Zoznam obrázkov

Úvod

Bakalárska práca vznikla na podnet lekárov z oddelenia pľúcnych chorôb Pediatrickej kliniky Fakultnej nemocnice Motol v Prahe, ktorí sa zaoberajú štúdiom riasiniek v pľúcach. Riasinky sú bunkové organely nachádzajúce sa na povrchu eukaryotických buniek, deformácie ich orientácie môžu byť príznakom viacerých chorôb. Lekári v rámci svojho výskumu spracovávajú veľké množstvo fotografií riasiniek. Doteraz toto spracovávanie prebiehalo manuálne, preto vznikla potreba aplikácie, ktorá by dokázala sama automaticky detekovať orientácie riasiniek na fotografiách a vhodne vyhodnocovať výsledky jednotlivých pacientov, čím by značne urýchlila a zjednodušila spracovávanie týchto fotografii.

Hlavná časť práce sa zaoberá návrhom algoritmu, ktorý je schopný rozoznávať orientáciu riasiniek na fotografiách. Ďalším aspektom bol vývin aplikácie s grafickým rozhraním, ktorá by fotografie riasiniek spracovávala a vyhodnocovala výsledky orientácie riasiniek pre jednotlivých pacientov.

Prvá kapitola práce sa zaoberá popisom riasiniek, motivácie k ich štúdiu a základných biologických pojmov, ktoré sa neskôr používajú v práci. Druhá kapitola je venovaná existujúcim metódam spracovania obrazu a popisu Qt knižnice použitej pri grafickom rozhraní aplikácie. V tretej kapitole sú definované štatistické metódy, ktoré sú použité v algoritme rozoznávajúcom riasinky, ako aj pri spracovaní výsledkov. Štvrtá kapitola sa zaoberá návrhom algoritmu rozoznávajúceho orientáciu riasiniek z fotografií. V piatej kapitole sú popísane požiadavky na grafické rozhranie aplikácie a jeho spracovanie. Posledná kapitola ukazuje testovanie aplikácie pri bežnom používaní.

Kapitola 1

Riasinky

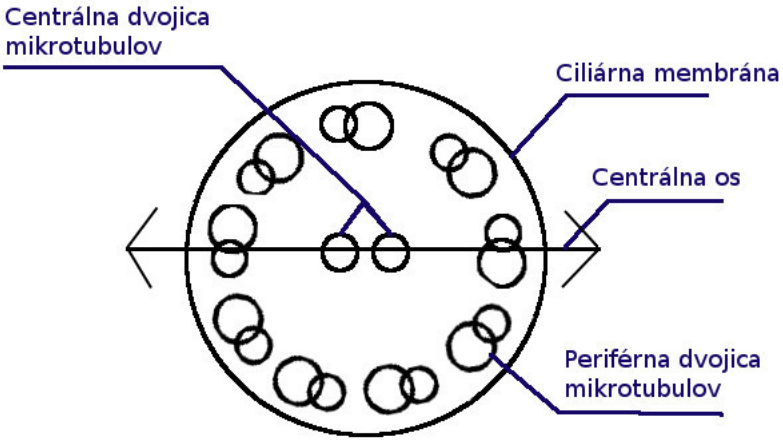
V tejto kapitole popíšeme základné biologické pojmy týkajúce sa riasiniek a motiváciu k štúdiu ich orientácie.

1.1 Základné pojmy

Riasinky (cilia angl.) sú bunkové organely, vystupujúce z povrchu eukaryotických buniek. Delia sa na dve základné kategórie:

- pohyblivé - nachádzajú sa na povrchu niektorých buniek vo veľkých počtoch, ich primárna funkcia je transport látok určeným, jednotným smerom, napr. odvádzanie hlienu z pľúc alebo pohyb mozgovo-miešneho moku. Nachádzajú sa tiež vo vajcovodoch, kde posúvajú vajíčko smerom k maternici. Dôležitú úlohu majú v embryonálnom vývoji, pri ich nesprávnom pohybe sa môžu vyskytnúť vrodené vývojové vady srdca, jeho uloženie v pravej časti hrudníka, príp. neobvyklé umiestnenie iných orgánov (pľúc, pečene, žalúdka, sleziny). V tejto práci sa ďalej budeme zaoberať len pohyblivými riasinkami.

- nepohyblivé (primárne) - slúžia ako zmyslové a signalizačné organely buniek, majú napríklad za úlohu detekciu svetla na sietnici oka alebo vnímanie pachov na nosnej sliznici, nachádzajú sa i v obličkách, kde je



Obr. 1.1: Priečny rez riasinky

ich úloha najmä signalizačná [1]. Ďalej sa nimi v tejto práci nebudeme zaoberať.

Riasinky majú úzky vlasovitý tvar, dĺžku 5-10 mikrometrov, priemer asi 0,2 mikrometra a vykonávajú **kmitavý pohyb**. Tento pohyb sa dá rozdeliť do dvoch častí:

- účinný úder - v smere, v ktorom riasinka prenáša látky
- obnovovací pohyb v opačnom smere, ktorý nemá slúžiť na prenos látok, ale len na návrat riasinky do pôvodnej polohy

Pohyb riasiniek je koordinovaný tak, aby vytváral postupujúcu pohybovú vlnu, ktorá efektívne posúva obsah na povrchu riasiniek. Ak by sa všetky pohybovali naraz, prenos látok by nebol taký efektívny. Smer, ktorým sa riasinka pohybuje, je možné určiť z priečneho rezu riasinky, ktorý je na obrázku 1.1. Šípky vyznačené na centrálnej osi riasinky naznačujú smer jej pohybu.

Mikrotubuly sú rúrkovité útvary, ktoré sa skladajú z makromolekúl tubulínu. Podľa polohy centrálnych mikrotubúl sa dá určiť smer pohybu (kmitania) riasinky. Ten je na obrázku naznačený **centrálnou osou** riasinky [2].

Riasinky sú v súčasnosti veľmi intenzívne skúmané. V medicíne je známa celá skupina chorôb označovaných súhrnne ako ciliopatie. Do tejto skupiny patrí aj geneticky podmienené ochorenie primárna ciliárna dyskinéza. Zatiaľ je popísaných 27 kauzatívnych génov. Veľká genotypová variabilita spôsobuje aj veľkú variabilitu fenotypov, ako napr. úplná nepohyblivosť riasiniek, porucha ich orientácie, porucha vzorca alebo frekvencie ich pohybu.

1.2 Skúmanie orientácie riasiniek

V zdravom organizme sú riasinky nachádzajúce sa v jednej oblasti (napr. na povrchu priedušiek, priedušnice, nosnej sliznice) koordinované a ich orientácia je približne rovnaká. Hodnotenie orientácie riasiniek má význam experimenálny, ale aj diagnostický. Pri podozrení na primárnu ciliárnu dyskinézu je súčasťou vyšetrenia pacienta aj morfologické hodnotenie vzorky pľúcneho tkaniva, kedy sa hodnotia štrukturálne poruchy riasiniek a ich orientácia. Jeden typ primárnej ciliárnej dyskinézy má typicky úplnú a v čase nemennú dezorganizáciu riasiniek. Porucha koordinácie nie je daná iba geneticky. Zápalový proces, infekčné ochorenie alebo fajčenie poškodzujú pľúca a dochádza i k poruche jednotnej orientácie riasiniek. Známkou uzdravenia je okrem iného aj obnova orientácie a transportnej funkcie riasiniek. [3].

Kapitola 2

Metódy spracovania obrazu

V tejto kapitole sa budeme zaoberať už existujúcimi nástrojmi a algoritmami pre spracovanie obrazu. Budeme stručne písať o existujúcich metódach a knižniciach, ktoré sú použiteľné v tejto práci.

2.1 Algoritmy spracovania obrazu

2.1.1 Detekcia hrán na obrázku

Je to jeden zo základných algoritmov používaných pri spracovaní obrazu. Často sa používa na predspracovanie obrazu pre iné, zložitejšie algoritmy.

Motivácia k hľadaniu hrán na obrázku

Nájdenie hrán na obrázku môže výrazne zmenšiť množstvo dát, ktoré sa neskôr budú spracovávať, a tým urobiť neskoršie spracovanie obrazu výpočtovo menej náročné. Predpokladá sa, že po detekcii hrán zostanú dôležité informácie na obrázku zachované. Nájdené hrany na obrázku môžu detekovať zmenu hĺbky obrázku, zmenu povrchu, reflexnosti alebo osvetlenia.

Definícia hrany na obrázku

Je ťažké presne povedať definíciu hrany na obrázku, tento pojem je skôr teoretický, zvyčajne sa hrana definuje ako ostrá zmena v jase obrázku. Nájdenie hrán na obrázku sa považuje za netriviálny problém, pretože dáta, ktoré opisujú skutočné obrázky sú diskrétne a je ťažké definovať, čo vlastne znamená nespojitosť pri diskrétnej funkcii [4].

Konkrétne metódy

Pred použitím konkrétneho filtra pre hľadanie hrán na obrázku sa zvyčajne používa niektorá z metód pre redukciu šumu, resp. vyhladenie obrázku (napr. Gaussovo vyhladenie). Následne sa použije niektorá z metód pre detekciu hrán. Existuje viacero metód, väčšina sa delí do dvoch hlavných kategórií. Prvá používa prvú deriváciu jasu a v nej hľadá lokálne maximá, druhá používa druhú deriváciu jasu na hľadanie prechodov nulou, ktoré korešpondujú s miestami v strede prechodu medzi úrovňami šedi (napr. Robertsov operátor, Sobelov operátor alebo Prewittovej operátor). Nakoniec sa zvyčajne použije niektorá z metód pre označovanie hrán a vyčistenie výsledkov. [5]

2.1.2 Detekcia kruhov

Detekcia kruhov je špecifickejší problém počítačovej grafiky. V algoritmoch na hľadanie kruhov v obrazoch sa veľmi často ako preprocessingový krok používa detekcia hrán. Jeden z najzákladnejších algoritmov pre hľadanie kruhov v obraze je Houghova transformácia. V implementácii algoritmu na hľadanie stredov riasiniek sme ju skúsili použiť, ale ukázala sa ako nedostatočne presná.

Houghova transformácia

Houghova transformácia je založená na analytickom vyjadrení kruhu. Funguje tak, že pre každý bod, ktorý je kandidátom na okraj kruhu (týchto kandidátov môžeme získať napríklad použitím niektorého z algoritmov pre

detekciu hrán), si skúsime zaznačiť všetky potenciálne stredy tohto kruhu. Pri Houghovej transformácii potrebujeme dopredu vedieť približný polomer hľadaných kruhov. [6]

Algoritmus popísaný presnejšie:

1. Inicializujeme si dvojrozmerné pole reprezentujúce každé políčko obrázku na nulu.

2. Pre každý bod (x, y), ktorý považujeme za možného kandidáta na stred kruhu, nájdeme všetky body a, b vyhovujúce rovnici

   $$(x - a)^2 + (y - b)^2 = r^2,$$

   kde r je predpokladaný polomer hľadaného kruhu.

3. Vo vzniknutom dvojrozmernom poli nájdeme lokálne maximá, ktoré by mali reprezentovať stredy kruhov.

Medzi nevýhody Houghovej transformácie patrí potreba definovať dopredu polomer hľadaných kruhov. Ďalším problémom je chybná detekcia prelínajúcich sa kruhov.

Na metóde Houghovej transformácie kruhov sú založené ďalšie zložitejšie metódy, napr. Všeobecná Houghova transformácia alebo Pravdepodobnostná Houghova transformácia [7].

Ak chceme použiť Houghovu transformáciu na hľadanie kruhov v diskrétnych obrazoch, musíme použiť niektorú z metód pre hľadanie rasterizácie kruhov v obraze, napr. Bresenhamov algoritmus.

Bresenhamov algoritmus

Bresenhamov algoritmus slúži na rasterizáciu kružnice s daným polomerom a daným stredom. Stačí nám vyrátať body na jednej osmine kružnice a zvyšné dokážeme z neho odvodiť, pretože ak bod (x, y) leží na kružnici, potom aj

body $(y, x), (y, -x), (x, -y), (-x, y), (-x, -y), (-y, x), (-y, -x)$ musia ležať na kružnici so stredom v strede súradnicovej sústavy.

Cieľom algoritmu je nájsť pixely, ktoré sa čo najviac približujú riešeniu rovnice $x^2 + y^2 = r^2$, kde x, y sú súradnice hľadaného bodu a r je polomer hľadaného kruhu. Podľa toho, ktorý oktant berieme ako základný, sa môžeme rozhodnúť, ktorá z našich premenných bude riadiaca. Pre každý oktant platí, že ak postupne hľadáme body kružnice, jedna z premenných x, y sa v každom kroku inkrementuje (prípadne dekrementuje), teda každý z nájdených pixelov bude mať rôznu minimálne jednu z premenných x, y. V najjednoduchšej verzii Bresenhamovho algoritmu vieme potom zvyšný bod dorátať úpravou rovnice $x^2 + y^2 = r^2$. Keďže ale platí, že nájdené body kružnice spolu susedia, dá sa tento algoritmus ďalej zjednodušiť. [8]

Algoritmy popisované v tejto časti práce som skúšala použiť na rozoznávanie stredov riasiniek, ale ukázali sa ako príliš nepresné a ako predprípravný krok algoritmu príliš pomalé.

2.2 Knižnice pre spracovanie obrazu

V jazyku C++ je implementovaných veľa knižníc zaoberajúcich sa počítačovým videním a spracovaním obrazu. Medzi najznámejšie a najobsiahlejšie z nich patrí OpenCV, VLFeat alebo VXL.

2.2.1 OpenCV

Knižnica OpenCV je open-source knižnica poskytujúca komplexné nástroje na prácu s obrazom. Delí sa na niekoľko základných modulov, z ktorých sa každý zaoberá inou súčasťou počítačového videnia. V OpenCV je možno vytvoriť celú aplikáciu vrátane jednoduchého grafického užívateľského rozhrania, pričom toto grafické rozhranie je dobre prispôsobené na prácu s obrazom. Medzi základné moduly tejto knižnice patria napríklad:

- **core** - modul zabezpečujúci základnú funkcionalitu, dátové štruktúry a funkcie, ktoré poskytuje ostatným modulom
- **imgproc** - tento modul obsahuje funkcie pre spracovanie obrazu, transformácie, filtre, prácu s farbami
- **highgui** - modul poskytuje užívateľské grafické rozhranie, základné ovládacie prvky ako tlačidlá a posuvník
- **gpu** - optimalizácia ostatných modulov pre grafickú kartu

Pre knižnicu OpenCV existuje veľké množstvo voľne dostupných návodov a ukážkových riešení [9].

2.3 Grafické knižnice v C++

Pre jazyk C++ neexistuje štandardné grafické užívateľské rozhranie, ako napr. JavaFX pre Javu, ale existuje množstvo rôznych alternatív, ktoré poskytujú tretie strany. Niektoré z najznámejších grafických knižníc pre C++ sú Qt, GTK+, FLTK alebo wxWidgets.

2.3.1 Qt

Qt je jedna z najpopulárnejších grafických knižníc pre programovací jazyk C++ (rovnako obsahuje táto knižnica podporu aj pre Python). Jej najväčšou výhodou je multiplatformovosť. Momentálne je vývoj Qt rozdelený do dvoch vetiev, Qt Widgets, ktoré sú vhodnejšie na tvorbu aplikácií pre stolové počítače a Qt Quick, ktorý umožňuje rýchly a jednoduchý vývoj mobilných aplikácií s podporou pre JavaScript.

Jedným zo základných konceptov Qt je úplná abstrakcia grafického rozhrania nezávisle na platforme. Qt ponúka základnú množinu grafických prvkov, ako napríklad tlačidlo (QPushButton) alebo kalendár (QCalendarWidget). Tie sa potom v aplikácii zobrazujú v súlade s vizuálnym štýlom ope-

račného systému, pre ktorý bola aplikácia vytvorená. Z tohto dôvodu sa tá istá aplikácia môže pod rôznymi operačnými systémami vizuálne líšiť.

Ďalší dôležitý koncept, na ktorom je postavená knižnica Qt, sú signály a sloty, ktoré zabezpečujú interakciu medzi jednotlivými grafickými prvkami. Signály sú jednotlivé udalosti, ktoré môžu nastať pre konkrétny grafický prvok (napríklad tlačidlo má signál kliknutia) a sloty sú funkcie, ktoré prijímajú informácie zo signálov. Pre grafické prvky sú už základné signály a sloty dostupné, ale je možné si dodefinovať vlastné. Tie sa potom dajú spojiť pomocou funkcie connect, pričom slot automaticky dostáva informácie zo signálu ako parametre pre svoje nasledujúce správanie (napríklad hodnota z posuvníka sa môže automaticky zobraziť užívateľovi v textovom okne, v prípade, že použijeme už existujúce signály a sloty, ktoré len stačí prepojiť) [10].

Qt rovnako ponúka aj abstrakciu pre vykresľovanie grafických objektov v podobe grafickej scény a okna. Grafická scéna ponúka funkcie pre vykresľovanie jednoduchých objektov (elipsa, čiara a pod.) a grafické okno slúži na zobrazenie grafickej scény užívateľovi, pričom ponúka automatické centrovanie scény, prispôsobenie veľkosti scény vzhľadom k veľkosti okna alebo pridávanie posuvníkov.

Spolu s Qt je vyvíjaný aj softvér Qt Creator, ktorý je prispôsobený vytváraniu Qt aplikácií a obsahuje aj vizuálny editor grafického rozhrania [11].

Kapitola 3

Štatistické metódy

V tejto kapitole sú popísané štatistické metódy, na ktoré sa neskôr v práci odvolávame. Pre pochopenie úplných základov štatistiky, napríklad pojmu premennej, odporúčame čitateľovi knihu *Introductory Statistics* [12].

3.1 Výberová štandardná odchýlka

3.1.1 Výberový priemer

Definícia 1. *Priemer pozorovaní pre náhodnú premennú x sa nazýva* ***výberový priemer****, označuje sa $\bar{x}$ a*

$$\bar{x} = \frac{\sum x_i}{n},$$

kde n je veľkosť výberu.

Výberový priemer budeme v tejto práci používať na zistenie priemeru nameraných orientácií riasiniek.

3.1.2 Definícia výberovej štandardnej odchýlky

Definícia 2. *Štandardná odchýlka pozorovaní pre premennú x sa nazýva* ***výberová štandardná odchýlka****, označuje sa s, kde*

$$s = \sqrt{\frac{\sum (x_i - \bar{x})^2}{n-1}},$$

pričom n je veľkosť výberu a $\bar{x}$ výberový priemer premennej x.

Výberová štandardná odchýlka slúži ako indikátor toho, ako veľmi sa namerané hodnoty premennej líšia od priemeru, respektíve do akej miery sú jednotlivé merania podobné.

Konkrétne v tejto práci sa výberová štandardná odchýlka používa na zistenie toho, ako veľmi sa orientácia jednotlivých riasiniek od seba líši.

3.1.3 Kombinovanie štandardných odchýliek

Definícia 3. *Pre dve nezávislé výberové štandardné odchýlky s_1, s_2 môžeme vyrátať* ***spoločnú štandardnú odchýlku****, ako:*

$$s_p = \sqrt{\frac{(n_1 - 1){s_1}^2 + (n_2 - 1){s_2}^2}{n_1 + n_2 - 2}},$$

kde n_1, n_2 sú veľkosti jednotlivých výberov.

Počítanie spoločnej štandardnej odchýlky sa v tejto práci používa na zrátanie spoločnej štandardnej odchýlky z viacerých obrázkov.

3.2 Pearsonov korelačný koeficient

Definícia 4. *Pre množinu n bodov premenných x a y,* ***Pearsonov korelačný koeficient*** *r je definovaný ako:*

$$r = \frac{\frac{1}{n-1} \sum (x_i - \bar{x})(y_i - \bar{y})}{s_x s_y},$$

kde $\bar{x}$ a $\bar{y}$ sú výberové priemery premenných x, y a s_x, s_y sú výberové štandardné odchýlky.

Pearsonov korelačný koeficient sa v štatistike používa ako miera sily lineárneho vzťahu medzi dvomi premennými, respektíve miera toho, ako veľmi sú si hodnoty dvoch premenných podobné.

V tejto práci sa bude používať ako súčasť rozoznávania polohy riasiniek na obrázkoch.

Kapitola 4

Návrh algoritmu

Algoritmus pre rozoznávanie riasiniek funguje v niekoľkých postupných krokoch. V tejto kapitole sú popísané jednotlivé kroky rozoznávania a definícia problému. Algoritmy sú písané v pseudokóde, pretože skutočný zdrojový kód obsahuje príliš veľa detailov nepodstatných pre pochopenie algoritmov.

4.1 Definícia algoritmického problému

4.1.1 Definície pojmov

Pre lepšie porozumenie nasledujúcemu textu tu definujeme niekoľko základných pojmov:

- **stred riasinky** - je kruh opísaný centrálnemu páru mikrotubulov
- **presný stred riasinky** - je miesto, ktoré sa nachádza v strede medzi centrálnym párom mikrotubulov, resp. miesto, z ktorého vzdialenosť k obidvom centrálnym mikrotubulom je najmenšia
- **bod** - jeden pixel na obrázku riasiniek
- **hodnota bodu** - je číselná hodnota bodu v odtieňoch sivej - hodnota z intervalu $[0, 255]$, vysoká hodnota bodu znamená farbu blízku bielej

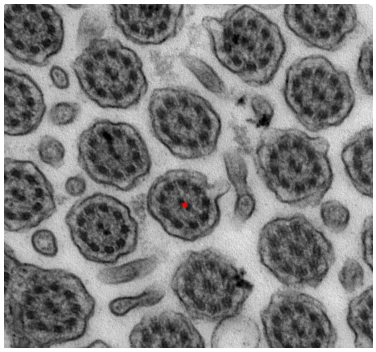
a naopak, nízke hodnoty znamenajú farbu blízku čiernej

- r - polomer riasinky

4.1.2 Zadanie úlohy

Zadanie: Cieľom algoritmu je rozoznávať orientáciu riasiniek z fotografií. Problém sa dá preformulovať na hľadanie centrálneho páru mikrotubulov v každej riasinke a jeho orientácie voči ostatným riasinkám. Hlavným kritériom pre algoritmus je jeho presnosť pri rozoznávaní riasiniek z obrázkov. Algoritmus by mal byť schopný rozoznať čo najviac z orientácií riasiniek, ktoré sú ešte rozoznateľné ľudským okom.
Vstup: Program dostane zadanú fotografiu riasiniek v odtieňoch sivej, polomer riasiniek vzhľadom k obrázku a súradnice presného stredu jednej riasinky. Riasinky na obrázkoch sú všetky približne rovnako veľké.
Výstup: Zoznam orientácií jednotlivých riasiniek.

Fungovanie algoritmu budeme ilustrovať na obrázku 4.1. Ako vstupné hodnoty pre algoritmus sme zadali $r = 80px$ a stredový bod ukážkovej riasinky $x = 454$ a $y = 501$, ktorý je vyznačený na obrázku.



Obr. 4.1: Ilustračný obrázok riasiniek s vyznačeným stredom

4.2 Predspracovanie obrázku

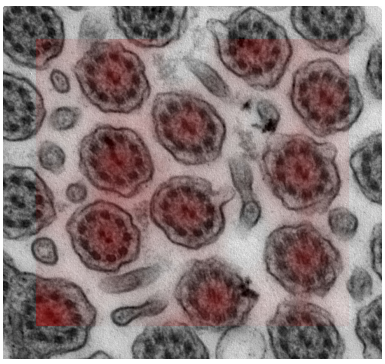
Tento krok algoritmu slúži ako pomocný krok k rozoznaniu stredov riasiniek. Jeho úlohou je iba znížiť výpočtovú náročnosť nasledujúceho kroku. Funguje tak, že vyfiltruje body, ktoré by určite nemali byť stredmi riasiniek. Tieto body budú neskôr ignorované.

Pôvodne ako tento krok mala slúžiť zjednodušená verzia Houghovho algoritmu pre hľadanie kruhov. Nakoniec sa ukázalo, že tento predspracujúci krok sa dá zrýchliť. Pre všetky testované obrázky s riasinkami platí, že riasinka je výrazne tmavšia ako jej okolie. Z toho vyplýva, že istú informáciu o polohe stredu riasinky nesú už hodnoty bodov v okolí presného stredu riasinky. Pre tmavšie body je preto väčšia pravdepodobnosť, že budú súčasťou riasinky. Z týchto pozorovaní sa dá usúdiť, že ak pre každý bod obrázku zrátame hodnoty bodov v jeho okolí, ktoré by ešte malo byť súčasťou riasinky, tak pre stredy riasiniek bude tento výsledok menší (ich okolie obsahuje málo bielych bodov, t.j. bodov s vysokými hodnotami).

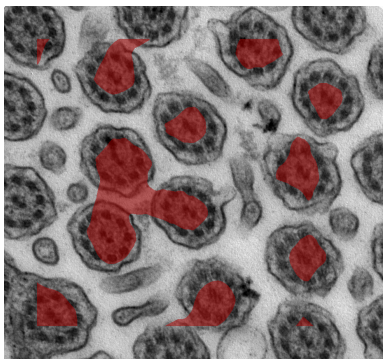
```
1  for i in picture_in.height:
2    for j in picture_in.width:
3      suma[i][j]=suma[i][j-1]+suma[i-1][j]-suma[i-1][j-1]+
   picture_in[i-1][j-1]
4  for i in picture_in.height:
5    for j in picture_in.width:
6      picture_out[i][j]=suma[i+1+rad][j+1+rad]-suma[i+1+rad][
   j+1-rad]-suma[i+1-rad][j+1+rad]+suma[i+1-rad][j+1-rad];
```

Listing 4.1: Rátanie prefixových súm v obrázkoch

Prakticky sme tento výpočet robili tak, že pre každý bod (x, y) sme zrátali sumu hodnôt bodov $(x - r, y - r)$ až $(x + r, y + r)$. Aby tento výpočet bežal v $O(n)$, kde n je počet bodov na obrázku, použili sme dvojrozmerné prefixové sumy. Algoritmus je spracovaný v pseudokóde vo výpise 4.1. Výsledok je zobrazený na obrázku 4.2(a), kde pre každý bod jeho zafarbenie dočervena závisí od sumy okolitých bodov (čím nižšia suma, tým červenší bod).



(a) Použitie prefixových súm pre riasinky



(b) Filter pre vyššie hodnoty súm okolia riasiniek

Obr. 4.2: Predspracovanie obrázku

Do ďalšieho kroku algoritmu sme následne vyfiltrovali iba body, pre ktoré táto suma prekračuje istú hranicu[1], ilustráciu čoho je vidno na obrázku 4.2(b).

4.3 Algoritmus hľadania presného stredu riasinky

Hľadanie presného stredu riasiniek na obrázkoch funguje v niekoľkých krokoch, pričom do úvahy ako možné presné stredy riasiniek budeme brať iba body, ktoré prešli filtrom v predošlom kroku.

4.3.1 Ohodnotenie bodov

Pre každý bod potrebujeme ohodnotenie, udávajúce ako veľmi sa okolie daného bodu podobá na riasinku s presným stredom v ňom. Ako základ pre tento algoritmus použijeme Pearsonov korelačný koeficient, ktorý je vysvet-

[1]Presnejšie tie, pre ktoré platí, že ich hodnota dosahuje aspoň 70% hodnoty maxima na obrázku.

lený v definícii 4. Pre každý bod a jeho okolie veľkosti r budeme rátať jeho korelačný koeficient s ukážkovým presným stredom riasinky a jeho okolím.

```
simple_representation_of_cilia(cilia c):
  for i in 1..c.radius:
    for p in random_points[i]:
        c.simple_representation[i]+=picture[c.center.y+p.y][c
    .center.x+p.x]
        c.simple_represenation[i]/=c.simple_representation[i
    ].length()
    c.mean = sum(simple_represenation)/c.radius
```

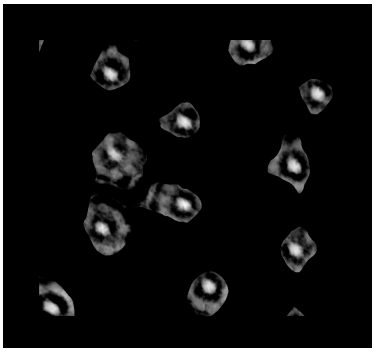
Listing 4.2: Vyjadrenie zjednodušenej reprezentácie riasinky

```
simple_pearson(Cilia a, Cilia b):
  for i in 1..a.rad:
    sum+=(a.simple_represenation[i]-a.mean)
      *(b.simple_represenation[i]-b.mean);
    disperse_b+=(a.simple_represenation[i]-a.mean)
      *(a.simple_represenation[i]-a.mean);
    disperse_b+=(b.simple_represenation[i]-b.mean)
      *(b.simple_represenation[i]-b.mean);
  return sum/(sqrt(disperse_a)*sqrt(disperse_b))*255;
```

Listing 4.3: Zrátanie korelácie dvoch rôznych možných stredov riasiniek

Keďže počet bodov v riasinke je napríklad v ilustračnom prípade $80^2\pi \sim 20000$, budeme brať v každej vzdialenosti od presného stredu riasinky do úvahy iba 10 náhodných bodov, pričom do samotného výpočtu korelácie zahrnieme iba ich súčet, čo postup značne zrýchli, a to bez prílišnej straty presnosti. Nasledujúci kus kódu je implementáciou tohto algoritmu, funkcia `simple_representation_of_cilia` v listingu 4.2 zráta pre každú riasinku sumu vybraných bodov v jednotlivých vzdialenostiach, a zároveň priemer súm vo všetkých vzdialenostiach. Funkcia `simple_pearson` v listingu 4.3 potom zráta pre dva dané body ich koreláciu a vráti číselnú hodnotu nového bodu v rozmedzí $[0, 255]$, teda hodnotu v odtieňoch sivej. Výsledok tohto

algoritmu možno vidieť na obrázku 4.3.



Obr. 4.3: Znázornenie korelácie bodov s daným stredom riasiniek

4.3.2 Nájdenie stredov riasiniek

Ako je vidno z predošlého kroku, body, ktoré boli na obrázku zobrazené ako najbelšie (mali najvyššie hodnoty korelácie), zhruba zodpovedali stredom riasiniek. V tomto kroku program vyhľadá na obrázku 4.3 tieto body. Zároveň berie do úvahy to, či sú tieto body dostatočne ďaleko od seba na to, aby boli vhodnými kandidátmi na presné stredy riasiniek. Pracuje tak, že najskôr si body na obrázku utriedi zostupne podľa ich hodnoty. Následne vezme všetky tie, pre ktoré platí, že ich vzdialenosť od všetkých predtým vybraných stredov je aspoň $2r$. Algoritmus je ilustrovaný vo výpise 4.4, v rámci neho je volaná funkcia `find_exact_centre`, ktorá je bližšie popísaná v nasledujúcej časti. Výsledok tohto kroku algoritmu možno vidieť na obrázku 4.4(a).

```
1  sort(points,descending)
2  for p in points:
3    if(have_distance_from_found_centres(p,found_centres)):
4      if(p>threshold):
5        exact_p=find_exact_centre(p)
6        found_centres.add(exact_p)
```

Listing 4.4: Hľadanie stredov riasiniek

4.3.3 Zlepšenie odhadu stredov riasiniek

V predošlom kroku sa nám síce podarilo nájsť približné presné stredy riasiniek, ale pre nasledujúce kroky programu je vhodné ešte tento odhad vylepšiť.

Z grafického hľadiska môžeme tento problém definovať ako hľadanie presných stredov bielych oválov (resp. predpokladaných bielych stredov) na obrázku 4.3. Z algoritmického pohľadu to zodpovedá hľadaniu susediacich bodov s dostatočnou svetlosťou a následne vybratie toho bodu, ktorého vzdialenosť od všetkých ostatných nájdených bodov je čo najmenšia.

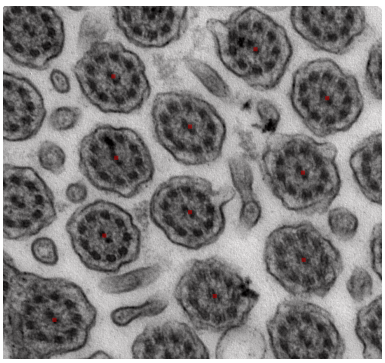
```
1 find_exact_centre(p):
2   for rp in reachable_points:
3     if rp.value>threshold:
4       sum_x+=rp.x
5       sum_y+=rp.y
6       count++
7   return (sum_x/count,sum_y/count)
```

Listing 4.5: Upresnenie nájdených stredov riasiniek

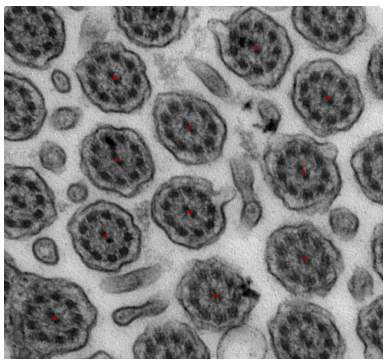
Prakticky sme z bodu, o ktorom sme predpokladali, že je stredom riasinky, spustili prehľadávanie do šírky, ktoré sa zastavilo, ak narazilo na bod, ktorý už nedosahoval dostatočnú hodnotu svetlosti. Počas prehľadávania sme si zrátavali súradnice jednotlivých bodov. Výsledok, ktorý tento výpočet vráti, je priemer všetkých návštívených súradníc bodov. Možno ho vidieť na obrázku 4.4(b). Algoritmus je ilustrovaný v listingu 4.5.

4.4 Algoritmus hľadania orientácie riasinky

Z predošlého kroku algoritmu dostávame k dispozícii pravdepodobné presné stredy riasiniek na obrázku. Zjednodušene sa dá výpočet v tomto kroku popísať ako hľadanie priamky, ktorá prechádza nájdeným presným stredom riasinky, a zároveň aj stredom centrálneho páru mikrotubulov.



(a) Nájdené stredy pred korekciou



(b) Nájdené stredy po korekcii

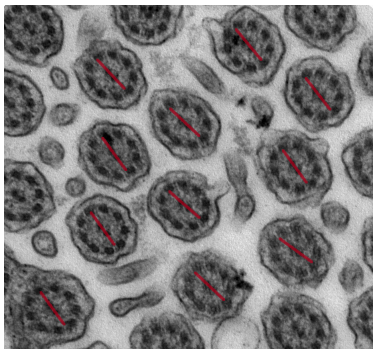
Obr. 4.4: Stredy riasiniek

```
1  for c in centres:
2    for i in 0..angles:
3      sum=0
4      y1=cos(((PI)/angles)*i)
5      x1=sin(((PI)/angles)*i)
6      for p in cilia_radius:
7        y2=p.y-c.y; x2=p.x-c.x;
8        sum+=picture[p.y][p.x]/(abs(-y1*y2+x1*x2)+1);
9      if sum<min:
10       c.min_angle=i
11       min=sum
```

Listing 4.6: Nájdenie orientácií riasiniek

Algoritmus použitý v tejto časti pre každý predtým vyrátaný presný stred riasinky postupne skúša možné orientácie z rozsahu $[0, 180]$ (orientácia $x \in [0, 180]$ je ekvivalentná s orientáciou $x + 180$, čo vyplýva z vlastností riasiniek), pričom pomocou premennej `angles` je možné špecifikovať ako veľa možností orientácii algoritmus vyskúša (pri testovaní na danom obrázku to bolo 180). Pre každú orientáciu algoritmus zráta sumu všetkých bodov patriacich do stredu riasiniek, pričom pre každý bod sa jeho hodnota predelí jeho

vzdialenosťou k priamke možného priemeru. Priamka, ktorá najviac zodpovedá orientácii riasinky, by mala byť zároveň aj priamkou, pre ktorú je táto suma najmenšia. Tento predpoklad vyplýva z toho, že hľadaná priamka by mala prechádzať veľkým počtom čiernych bodov, ktoré v tomto prípade bude brať do úvahy s oveľa väčšou váhou ako vzdialenejšie biele body. Výsledok tohto algoritmu je vidno na obrázku 4.5. Samotný algoritmus je popísaný v listingu 4.6.



Obr. 4.5: Orientácie riasiniek nájdené algoritmom

Kapitola 5

Návrh prostredia aplikácie CiDi

V tejto kapitole sa budeme zaoberať požiadavkami na samotnú aplikáciu, implementáciou užívateľského rozhrania a spracovaním výstupov aplikácie. Aplikáciu sme nazvali CiDi (skratka pre Cilia Direction).

5.1 Užívateľské požiadavky na aplikáciu

Aplikácia má slúžiť ako pomocný nástroj pre lekárov na oddelení pľúcnych chorôb Pediatrickej kliniky fakultnej nemocnice Motol v Prahe. Jej úlohou je zjednodušenie rozoznávania orientácie riasiniek z mikroskopických obrázkov riasiniek a následne zjednodušenie ich spracovania a vyhodnocovania. Spolu s MUDr. Magdalénou Havlišovou, ktorá na spomenutej klinike pracuje, sa nám podarilo sformulovať nasledovné požiadavky na aplikáciu:

- **desktopová aplikácia pre operačný systém Windows 7,8** - neboli žiadne špeciálne požiadavky na operačný systém, aplikácia sa má využívať na bežných PC, neexistuje požiadavka na jej rozšírenie na tablety alebo mobilné telefóny, prípadne online
- **jednoduché používateľské rozhranie** - znova sa vychádza z predpokladu, že používatelia aplikácie nemusia byť veľmi technicky zdatní,

preto sme sa snažili, aby ovládanie aplikácie bolo do veľkej miery jednoduché a intuitívne (napr. konzolová aplikácia neprichádzala do úvahy)

- **spracovávanie výsledkov jednotlivých pacientov** - aplikácia má umožňovať triediť fotografie podľa príslušnosti k pacientovi a pre jednotlivých pacientov vyhodnocovať výsledky z jeho fotografií, pričom výsledok jednej fotografie by mal byť jeden číselný údaj hovoriaci ako veľmi rozdielne sú orientácie riasiniek na danej fotografii. Keďže fotografie jedného pacienta spracováva vždy ten istý doktor, zatiaľ nevznikla požiadavka na centralizovanú databázu pacientov

Na základe týchto požiadaviek sme urobili nasledovné rozhodnutia:

- **jazyk C++** - aplikácia pracuje s rozoznávaním obrazu, čo je vo všeobecnosti pomalá operácia, preto sme sa rozhodli použiť rýchly kompilovaný programovací jazyk C++

- **Qt knižnica** - pre aplikáciu je potrebné grafické používateľské rozhranie, Qt je jedna z najobsiahlejších grafických knižníc pre C++ a dovoľuje prípadné neskoršie rozšírenie na väčšinu iných platforiem

- **spracovanie výsledkov do CSV súborov** - jednotlivé výsledky pacientov sme sa rozhodli zapisovať do súborov formátu CSV (Comma Separated Values, jednotlivé hodnoty sú oddelené čiarkou), čo je formát ľahko spracovateľný počítačom, čitateľný človekom a takisto aj podporovaný mnohými tabuľkovými procesormi (napr. Microsoft Excel)

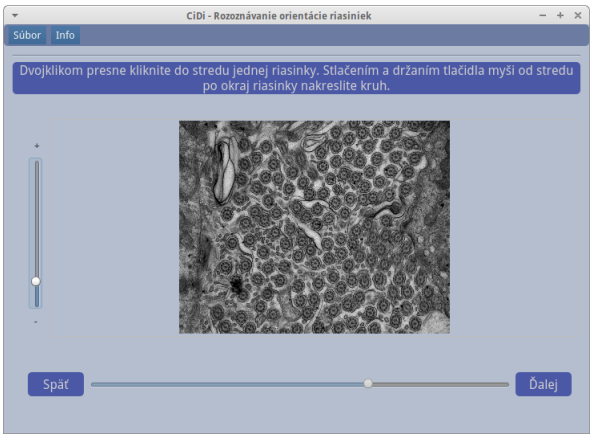
- **interakcia s používateľom** - keďže od aplikácie sa vyžaduje presnosť výsledkov, je potrebné, aby užívateľ bol schopný opraviť jednotlivé výstupy vytvorené algoritmom, cieľom je však čo najviac ho odbremeniť od manuálneho zakresľovania orientácií riasiniek

5.2 Grafické užívateľské prostredie aplikácie

Aplikácia je navrhnutá tak, aby kopírovala kroky algoritmu popísané v predošlej kapitole. Na rozdiel od konzolovej aplikácie však sprevádza užívateľa celým procesom zisťovania orientácie, vizualizuje jednotlivé kroky a umožňuje mu ich upravovať.

5.2.1 Všeobecné prvky aplikácie

Hlavné okno aplikácie je zobrazené na obrázku 5.1. Jej dizajn zodpovedá dizajnu sprievodcu (wizard). V každom kroku používateľovi dovoľuje posunúť sa na ďalší krok tlačidlom `Ďalej` a vrátiť sa k predošlému kroku tlačidlom `Späť`. V hlavnom okne je zobrazený spracovávaný obrázok, ktorý si užívateľ posuvníkom vľavo môže priblížiť alebo oddialiť. Toto je nevyhnutné, pretože fotografie riasiniek bývajú značne veľké (cca $3000 \times 3000px$). Posuvník v dolnej časti aplikácie slúži v niektorých krokoch na úpravu konštánt pre algoritmus. Text v hornej časti obrazovky sprevádza používateľa celým procesom rozoznávania riasinky a v každom kroku mu vysvetľuje, ako má postupovať.



Obr. 5.1: Hlavné okno aplikácie

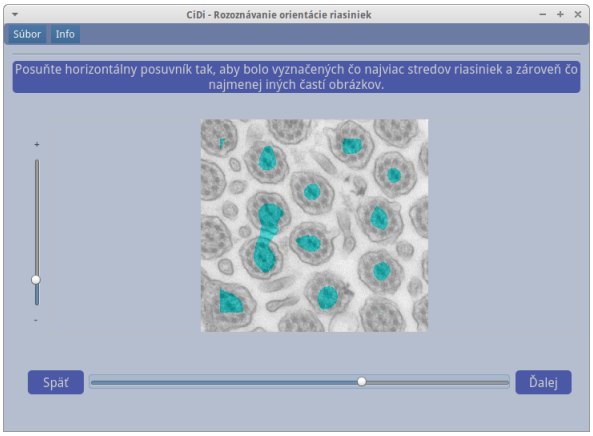
5.2.2 Zadanie počiatočného vstupu

Užívateľ je v prvom kroku vyzvaný k výberu obrázku riasinky určeného na spracovanie. Momentálne aplikácia podporuje bežné formáty ako JPG, PNG alebo PGM. Ďalej musí k danej fotografii priradiť pacienta. V tomto momente je rovnako možné vytvoriť v aplikácii nového pacienta.

Po zobrazení obrázku je ešte potrebné, aby užívateľ klikol na presný stred jednej riasinky a nakreslil kruh zodpovedajúci veľkosti jednej riasinky. Ďalšie kroky programu už teoreticky môžu prebiehať nezávisle od používateľa, pokiaľ ich nebude nutné korigovať.

5.2.3 Výber hranice pre hľadanie stredov

V tomto kroku sa zvýraznené zobrazia tie body, ktoré bude algoritmus brať do úvahy pre ďalšie spracovanie. Užívateľ môže množstvo týchto bodov upraviť. Pre správne nastavenie by malo platiť, že zvýraznené sú všetky stredy riasiniek, a zároveň čo najmenej inej plochy. To značne urýchli ďalší krok algoritmu. Správne nastavenie hranice je vidno na obrázku 5.2.



Obr. 5.2: Nastavenie hranice pre algoritmus

5.2.4 Zobrazenie stredov a orientácie riasiniek

Ďalšie dva kroky slúžia už na zobrazenie výsledkov algoritmu. V prvom z nich sa užívateľovi zobrazia nájdené stredy riasiniek. Tieto stredy môže posúvať, mazať alebo pridávať. Horizontálny posuvník mu tiež dovoľuje zmeniť celkové množstvo zobrazených stredov nastavením inej hranice pre algoritmus.

V druhom z týchto krokov pre každý nájdený stred algoritmus zistí zodpovedajúcu orientáciu riasinky. Aj v tomto kroku je ešte možné jednotlivé zle nájdené orientácie manuálne upraviť.

5.2.5 Ukončenie spracovania obrázku

Program po zobrazení orientácie riasiniek užívateľovi zráta odchýlku orientácií riasiniek na obrázku a ponúkne mu možnosť uložiť tento výsledok spolu s ostatnými dátami pacienta.

5.3 Spracovanie výstupu aplikácie

Dôležitým aspektom práce bolo vytvorenie jednoduchého systému vyhodnocovania výsledkov obrázkov jednotlivých pacientov.

5.3.1 Vyhodnocovanie orientácie riasiniek

Výstupom algoritmu zisťovania orientácie riasiniek popísaného v predošlej kapitole sú orientácie jednotlivých riasiniek v stupňoch vzhľadom k horizontálnej osi. Požiadavka na výstup aplikácie je jeden číselný údaj hovoriaci ako veľmi sú riasinky na obrázku rozdielne. Ako vhodnú číselnú reprezentáciu tohto údaja sme zvolili výberovú štandardnú odchýlku, ktorá je bližšie vysvetlená v definícii 2.

Kedže program vracia orientácie iba z intervalu $[0, 180]$, vznikali nezrovnalosti pri rátaní ich priemeru (orientácie 1 a 180 by mali byť skoro identické). Vyriešili sme to tak, že sme postupne iterovali cez všetky možné priemery. Pre každý z nich sme pre jednotlivé riasinky brali do úvahy menšiu

zo vzdialeností od priemeru: $min(abs(moznypriemer - orientacia), 180 - abs(moznypriemer - orientacia))$. Nakoniec sme vybrali ten z možných priemerov, ktorý bol celkovo najmenší. Podobný postup sme použili aj pri následnom rátaní odchýlky.

Ďalšou požiadavkou bolo, aby program vedel spájať výsledky pacienta z jednotlivých obrázkov, t. j. vedieť vhodne spájať jednotlivé štandardné odchýlky. Na to sme využili kombinovanie štandardných odchýliek popísané v definícii 3.

5.3.2 Ukladanie dát pacientov

Ako je spomenuté vyššie, dáta pacientov sme sa rozhodli ukladať do CSV súborov, ktoré sú ľahko použiteľné aj na ďalšie spracovanie. Pre každého pacienta ukladáme jeden súbor, v ktorom sú údaje o jednotlivých výsledkoch fotografií. V budúcnosti plánujeme pracovať aj na rôznych vizualizáciách vzniknutých dát.

Kapitola 6

Testovanie aplikácie

V tejto kapitole sa budeme zaoberať testovaním aplikácie pri bežnom používaní.

6.1 Metódy testovania aplikácie

Keďže aplikácia sa bude používať s grafickým rozhraním, kde bude možné veľa vecí pre konkrétny obrázok nastaviť, čisto testovanie algoritmickej časti aplikácie nemá pre praktické účely zmysel. Testovanie bude preto prebiehať reálnym používaním aplikácie s grafickým rozhraním, pričom budeme sledovať faktory ako čas spracovávania obrázku, presnosť algoritmu a množstvo korekcií, ktoré je nutné pre daný obrázok urobiť. Toto testovanie môže byť do istej miery subjektívne a záleží aj od schopnosti užívateľa pracovať s aplikáciou. Slúži iba na rýchly prehľad, ako asi bude fungovať aplikácia pri reálnom používaní. Takisto je niekedy diskutabilné, či sa dá pre niektorú riasinku na obrázku ešte určiť orientácia, pokiaľ je napríklad rozmazaná alebo sa časť z nej na obrázku nevyskytuje.

Testované obrázky sú kvôli svojej veľkosti k dispozícii iba v elektronickej prílohe B.

6.2 Sledované dáta a spôsob ich merania

Pri testovaní aplikácie budeme sledovať nasledovné aspekty, pričom pre každý uvedieme v zátvorke jeho označenie v tabuľke meraní:

- **čas spracovávania obrázku** - čas meraný od kliknutia užívateľa na tlačidlo `Súbor` až po kliknutie na tlačidlo `Save`
- **počet zle nájdených stredov** - počet označených stredov, ktoré buď nepatrili riasinke alebo ich polohu bolo treba dodatočne korigovať
- **počet nenájdených stredov** - počet stredov riasiniek, ktoré algoritmus nebol schopný nájsť
- **počet riasiniek** - celkový počet riasiniek na obrázku, ktoré sa dajú na danom obrázku rozoznať
- **počet zle označených orientácií riasiniek** - označuje počet orientácií, ktoré bolo nutné dodatočne korigovať

Testovanie prebiehalo na notebooku Acer Aspire s3 MS2346, s pripojeným externým monitorom, kedže počítač má malý 13"displej a myšou, kvôli väčšej presnosti.

6.3 Namerané dáta

A	B	C	D	E	F	G	H	I
1.png	109	7	6,42%	12	11,01%	11	10,09%	3:40
2.png	146	10	6,85%	12	8,22%	10	6,85%	3:10
3.png	129	5	3,88%	0	0,00%	2	1,55%	2:05
4.png	123	4	3,25%	9	7,32%	6	4,88%	2:20
5.png	98	3	3,06%	0	0,00%	7	7,14%	2:00
6.png	59	0	0,00%	4	6,78%	8	13,56%	1:55
7.png	133	6	4,51%	21	15,79%	7	5,26%	2:58
8.png	97	0	0,00%	3	3,09%	6	6,19%	2:15
9.png	119	8	6,72%	12	10,08%	3	2,52%	2:07
10.png	90	1	1,11%	2	2,22%	5	5,56%	1:50
11.png	109	3	2,75%	0	0,00%	2	1,83%	1:43
12.png	154	4	2,60%	1	0,65%	5	3,25%	2:03
priemer	113,8	4,25	3,43%	6,33	5,43%	6,00	5,72%	02:20
štandardná výberová odchýlka	26,13	3,14	2,39%	6,76	5,23%	2,86	3,49%	00:36

Legenda k tabuľke

- A - názov testovaného obrázku
- B - počet riasiniek
- C - počet nenájdených stredov
- D - percentuálny podiel C a B
- E - počet zle nájdených stredov
- F - percentuálny podiel E a B
- G - počet zle označených orientácií riasiniek

- H - percentuálny podiel G a B
- I - čas spracovávania obrázku (mm:ss)

6.4 Vyhodnotenie

Z nameraných dát sa dá usúdiť, že spracovávanie jedného obrázka programom by malo trvať menej ako 3 minúty. Chybovosť meraní každého sledovaného aspektu sa pohybuje okolo 5%, veľakrát však bolo diskutabilné, či je ešte pre danú riasinku orientácia určiteľná. Ako je spomenuté vyššie, výsledky testovania sú orientačné a slúžia iba na hrubý odhad schopností aplikácie.

Záver

Algoritmus na rozoznávanie riasiniek sa nám podarilo implementovať a začleniť do funkčnej aplikácie s grafickým rozhraním. Momentálne prebieha testovanie aplikácie s pomocou MUDr. Magdalény Havlišovej a pracuje sa na jej zavedení do praxe. Aplikácia by mala značne zrýchliť a uľahčiť proces rozoznávania orientácie riasiniek a ich vyhodnocovania, čo by malo následne pomôcť ďalšiemu výskumu riasiniek.

V budúcnosti plánujeme na aplikácii ďalej pracovať a vylepšovať ju podľa vzniknutých požiadaviek. Možné zlepšenia aplikácie, ktoré prichádzajú do úvahy, sú napríklad vizualizácie výstupov alebo ďalšie zrýchlenie rozoznávania riasiniek.

Literatúra

[1] SATIR P., CHRISTENSEN S. T., Structure and function of mammalian cilia, Histochem Cell Biol., 129(6): 687–693, 2008

[2] den TOONDER J. M. J., ONCK P. R., O'BRIEN P., 2013, Artificial Cilia, The Royal Society of Chemistry, 265 s.

[3] OROSOVÁ J., 2000, Primárna ciliárna dyskinézia (syndróm imotílnych cílií, Kartagenerov syndróm), Pediatria pre prax, 12(1):7-11, 2011

[4] ZIOU D., TABBONE S., Edge Detection Techniques - An Overview, International Journal of pattern recognition and image analysis, 8: 537-559, 1998

[5] HAFFNER O., Detekcia hrán v obraze, [online] Dostupné na internete: http://www.posterus.sk/?p=11234

[6] PEDERSEN S. J. K.,Circular Hough Transform, Aalborg University, Vision, Graphics, and Interactive Systems, 2007

[7] BALLARD D. H., Generalizing the hough transform to detect arbitrary shapes, Pattern recognition, 13(2):111–122, 1981

[8] RUŽICKÝ E., FERKO A., Počítačová grafika a spracovanie obrazu, Bratislava Sapientia, 1995

[9] Dokumentácia OpenCV, [online] Dostupné na internete: http://docs.opencv.org/

[10] MOLKENTIN, D., The book of Qt 4: the art of building Qt applications, Open Source Press San Francisco, 2007

[11] Dokumentácia Qt knižnice, [online] Dostupné na internete: http://doc.qt.io/

[12] WEISS, N. A., Introductory Statistics, Pearson Education, 2012

Appendix A

Príloha A obsahuje zdrojový kód a spustiteľnú aplikáciu. Nájdete ju na priloženom CD v priečinku `CiDi`.

Appendix B

Príloha B obsahuje fotografie riasiniek použité pri testovaní aplikácie. Kvôli veľkému rozlíšeniu fotografii sú uložené na priloženom CD v priečinku `fotografie`.