Univerzita Hradec Králové
Fakulta informatiky a managementu
Katedra informatiky a kvantitativních metod

DIPLOMOVÁ PRÁCE

Návrh a realizace software pro analýzu biomedicínských dat

Autor: Lukáš Sulík

Studijní obor: AI2

Vedoucí práce: doc. Ing. Ondřej Krejcar, Ph.D.

Hradec Králové 2015

Prohlášení

Prohlašuji, že jsem diplomovou práci zpracoval samostatně a s použitím uvedené literatury.

V Hradci Králové dne 13. listopadu 2015 Podpis:

Poděkování

Rád bych poděkoval vedoucímu diplomové práce doc. Ing. Ondřeji Krejcarovi, Ph.D. za ochotu a úsilí vést tuto práci. Dr. Orcanovi Alparovi za konzultace ohledně vyvíjených algoritmů. A v neposlední řadě své rodině a přátelům, kteří mě při studiu podporovali.

Anotace

Práce v teoretické části představuje kapslovou endoskopii a její využití ve zdravotnictví a zároveň základy počítačového vidění, jeho úskalí a přednosti. A dále představuje i několik základních algoritmů, které se v práci používají a průzkum trhu, jenž analyzuje aktuální stav technologií. Praktická část je založena na vývoji nového programu pro kapslovou endoskopii a základních algoritmů pro detekci krvácení v tenkém střevě. Výsledkem práce je software na platformě Java, využívající knihovnu OpenCV, který dokáže detekovat krvácení v zažívacím traktu.

Annotation

Theoretical part of the thesis presents capsule endoscopy and its use in healthcare and basics of computer vision, its pitfalls and advantages. Along with that, it is presented several fundamental algorithms that are used in the thesis and market research, which analyzes the current state of the art. Practical part is based on the development of a new program for capsule endoscopy and basic algorithms for detecting bleeding in the small intestine. The result is software based on the Java platform, using the OpenCV library and it can detect bleeding in the digestive tract.

Obsah

1. Úvod

S nárůstem moderních technologií se rozvijí také lékařské odvětví. Dnes je již téměř vše řízené alespoň částečně pomocí počítačů a je snaha co nejvíce úkonů zautomatizovat, aby se lékaři mohli soustředit pouze na věci, které vyžadují jejich odborné znalosti a zkušenosti.

Kapslová endoskopie se zabývá analýzou tenkého střeva. To je nejobtížnější část střeva k prozkoumání, z důvodu vzdálenosti od úst a řitního otvoru a jeho komplikovaného tvaru, který může obsahovat různé smyčky. Konvenční endoskopické techniky např. enteroskopie nebo kolonoskopie jsou limitovány délkou tenkého střeva (3.35–7.85 m)[1], a proto je potřeba získávat data způsobem, který přinesla právě kapslová endoskopie.

Získaná data je nutné vyhodnotit a to může provést osoba s potřebnými znalosti k tomu určená, což přináší hned několik úskalí. Vzhledem k množství vyšetření a délky záznamu spotřebovává spoustu času projetí všech záznamů a určení diagnózy. To má za následek větší časovou náročnost na odborníky a také, pokud má pacient nějaký akutní problém, chvíli potrvá, než se podaří odhalit ze záznamů jeho příčinu.

Z těchto důvodů je často lékařské odvětví propojováno s rozhodovacími a vyhodnocovacími systémy, které zpracovávají nepřeberné množství dat a usnadňují tím lékařům práci. Specializované biomedicínské programy pak dokáží rychle identifikovat zdroj choroby a zkrátí tím čas, který by byl potřebný k projetí všech dat. V případě, že program nalezne nějakou anomálii, lékař hned objeví zdroj problému a může stanovit diagnózu podstatně rychleji. Samozřejmě, že tyto systémy nemohou plně nahradit znalosti a zkušenosti odborníků, z právního hlediska tak činit ani nesmějí, mohou však minimálně usnadnit jejich práci.

1.1. Cíle práce

Cílem této práce je navrhnout systém, který by dokázal zpracovávat data z kapslové endoskopie. Měl by být konkurenceschopný mezi aktuálními systémy na trhu.

Konkrétním úkolem systému pro tuto diplomovou práci je z dodaných dat analyzovat, zdali se v nich nachází stopy po krvácení. Kromě detekce krve systém musí být připraven i na další možné rozšíření v podobě nových detekčních algoritmů i pro jiné nemoci, které by se mohly implementovat později, pokud se systém ověří. Algoritmy pro detekci krve je nutné zaměřit na co největší rozsah detekce, aby byla téměř 100% jistota, že bude krev nalezena.

Systém bude ověřen na vzorcích dat dodaných Fakultní nemocnicí Hradec Králové, s kterou bude posuzována i funkčnost a využitelnost daného řešení.

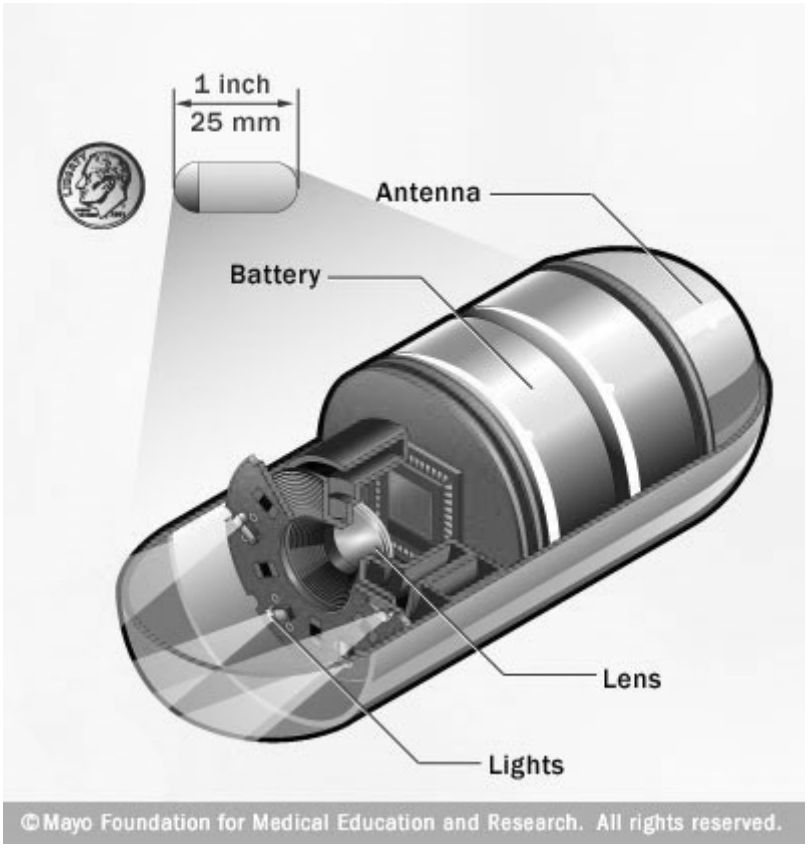
Aktuální návrh systému počítá s implementací GUI v podobě desktopové aplikace, jež bude spustitelná nezávisle na zvolené platformě, prioritně na Microsoft Windows. Systém však musí být schopný snadné rozšiřitelnosti v podobě implementace jiného GUI, např. webové aplikace.

1.2. Kapslová endoskopie

Kapslová endoskopie spočívá v malé kapsli o velikosti větší pilulky, jež má vlastní zdroj světla, kameru, anténu a zdroj energie viz obrázek 1.1. Kamera snímá obraz ze střeva a posílá ho bezdrátově do přijímače umístěného mimo pacientovo tělo. V současné době existuje několik výrobců, jejichž specifikace kapsle se může mírně lišit. Jsou jimi např.:

- Olympus EC-S10 – 11x26 mm, výdrž až 12 hodin.[2]
- IntroMedic MicroCam – 11x24mm, výdrž 11+ hodin.[3]
- GivenImaging PillCam SB3 – 11x26mm, výdrž 8+ hodin.[4]

Ve většině případů je zapotřebí okolo osmi hodin [6] pro průchod celým zažívacím traktem. Po dokončení celého procesu je kamera vyloučena z pacientova těla přirozenou cestou a již se nedá znovu použít. Vyšetření se provádí za účelem zjištění následujících nemocí (volně přeloženo z [5]):



Obrázek 1.1.: Kapsule [5]

- Krvácení v zažívacím traktu - kapslová endoskopie může pomoci při určení příčiny krvácení.
- Zánětlivé onemocnění střev - kapslová endoskopie může odhalit místa zánětu v tenkém střevě a tak pomoci diagnostikovat Crohnovu nemoc a další zánětlivá onemocnění.
- Rakovina - kapslová endoskopie může identifikovat nádory v tenkém střevě, které by jinak byly těžko identifikovatelné. Kapslová endoskopie je občas prováděna se spojením s magnetickou rezonancí, protože magnetická rezonance může odhalit nádory ve stěně tenkého střeva.
- Celiakie - některé malé studie tvrdí, že kapslová endoskopie může detekovat střevní změny spojené s celiakií - alergickou reakci po požití lepku - a může detekovat komplikace.

- Polypy - lidé, kteří mají vrozenou polypózu, která může způsobovat polypy v tenkém střevě např. Peutz-Jeghersův syndrom, tak mohou využívat kapslovou endoskopii pro zobrazení polypů.

2. Počítačové vidění v endoskopii

2.1. Úvod do počítačového vidění

Počítačové vidění (dále jen CV) je obor výpočetní techniky jehož počátky sahají do 60. let minulého století. Tomuto oboru se v posledních letech dostalo velkého rozmachu. Na trh se dostává stále více zařízení schopných snímat obrazová data,vzrůstá jejich kvalita a cena klesá. Ač CV není vyloženě určeno nějakou definicí,dalo by se říci, že CV je strojové porozumění obrazu, který se získává ze snímacích zařízení (např. různé fotoaparáty či kamery). V podstatě by se dalo srovnat s viděním biologickým s tím rozdílem, že u většiny vícebuněčných organismů jde o naprosto přirozenou a běžnou věc.[7][8] CV najde zastoupení v celé řadě odvětví např.

- Automatizovaný průmysl - čidla pro kontrolu vadností výrobku, ovládání robotů, řízení logistických procesů, ...
- Medicína - zpracování obrazových dat z mikroskopů, ultrazvuku, endoskopii, RTG, ... CV zkvalitňuje diagnostické nástroje a je již standardní součástí diagnostických procesů.
- Armáda - automaticky naváděné rakety, detekce pohybu vojáků, bezpilotní stroje (letadla, ponorky, ...)
- Zábavní průmysl - interakce mezi strojem a člověkem. Např. ovládání počítačové hry gesty.
- Rozšířená realita - v kombinaci s počítačovou grafikou je základním prvkem v rozšířené realitě. Např. Microsoft hololens.

- A mnoho dalších.

Oproti biologickému vidění se jedná z pohledu zpracování o velice komplikovanou věc. Počítač přijímá pouze číselné hodnoty jako obrazová data. Proto je pro lidský druh snadné identifikovat např. auto na obrázku 2.1, ale pro stroj je nutné definovat auto, jako složitý matematický model. CV také naráží na problémy, které vznikají při sběru dat. Jsou jimi: (volně přeloženo z [8]).

- 3D > 2D - ztráta prostorové informace. Obraz z reálného světa je uložen jako 2D. Díky modelu dírkové komory[1] jsou malé objekty blízké kameře stejné jako velké vzdálené objekty. Lidé jsou schopni tyto objekty rozeznat, pokud je v obraze nějaký objekt, u kterého znají jeho velikost, např. krabička od sirek.

- Interpretace - porozumění obsahu scény. Lidský mozek vyhodnocuje obraz pomocí svých znalostí a zkušeností. A proto je schopný vyhodnocovat nové problémy, které předtím neznal. Zatímco takováto schopnost strojů je velice omezená a její zdokonalení je klíčem k inteligentním systémům. Počítač je schopen analyzovat obraz pomocí matematické logiky, sémantiky a teorií formálních jazyků. Pokud je znám algoritmus pro řešení určitého problému, je možno analyzovat obraz, např. řeka v satelitním obrazu.

- Šum - je přítomný v každém měření v reálném světě.

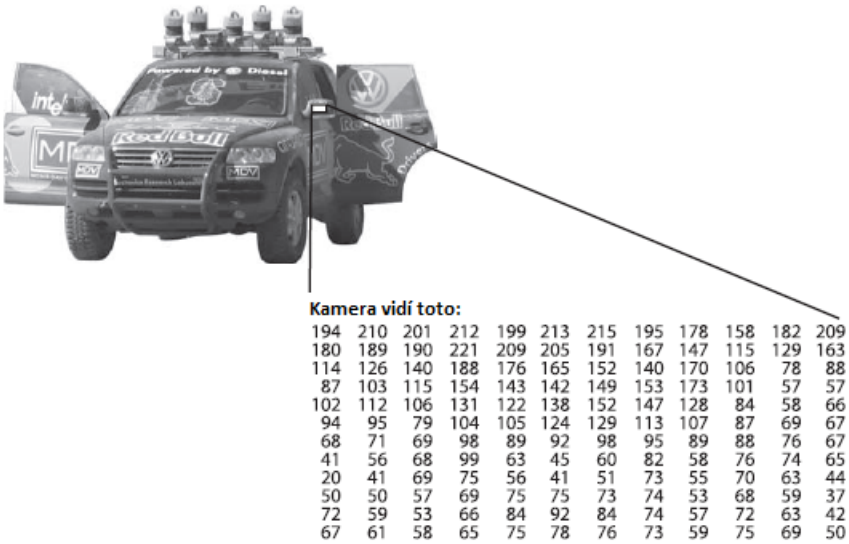
- Příliš mnoho dat - Obrázky a videa jsou obrovské. Stránka A4, 300 dpi, 8 bit per pixel = 8,5 MB. Neprokládané video 512 x 768, RGB (24 bit) = 225 MB/s. Je proto tak těžké dosáhnout složité analýzy v reálném čase.

- Měřený jas - je dán složitým fyzikálním postupem vytváření obrazu. Zář závisí na ozáření (typ světelných zdrojů, jejich poloha a intenzita), poloze pozorovatele, lokální geometrii povrchu a odrazivosti povrchu. Obrácená úloha je špatně podmíněna.

- Lokální okno versus potřeba globálního pohledu - počítač vidí svět pouze z pohledu klíčové dírky bez celkového kontextu. Je proto těžké určit kontext obrazu, pokud je

[1] z anglického pinhole model

dostupná jen jeho část. Lidský mozek je schopný si domyslet potřebné informace pro určení, pokud je zná.



Obrázek 2.1.: Auto přeloženo z[7]

Zpracování obrazu počítačem se dá rozdělit na několik úrovní a výsledkem je řetězec, kde se postupuje od nejnižší po nejvyšší úroveň. Na nejnižší úrovni se zpracovávají surová data, která jdou přímo ze snímacích senzorů. Poté, co je obraz z nejnižší úrovně zpracován, používají se metody pro odstranění šumu, detekce hran, barevná filtrace či různá komprese obrazu. Těmito úpravami vznikne obraz, ze kterého lze již přejít k vyšší úrovni zpracování a tím je porozumění daného obrazu. Na této úrovni se dají aplikovat poznatky z umělé inteligence (např. neuronové sítě) nebo matematické definování objektů pro určení výsledného významu obrazu. Vyšší úroveň zpracování je nejtěžší věc na CV, často není ani možné dosáhnout požadovaného výsledku a vše se musí zjednodušovat, aby se dosáhlo alespoň výsledku částečného. [7][8]

2.2. Algoritmy počítačového vidění

V biomedicíně se používá většina známých algoritmů z počítačového vidění. Jelikož je jich velké množství, jsou pro charakter této práce popsány pouze ty, které se pak používají v

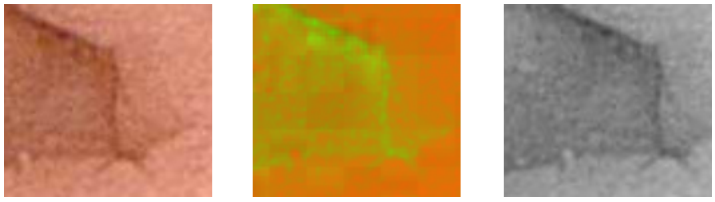
praktickém řešení konkrétního problému této práce. Všechny tyto základní známé algoritmy jsou implementovány v knihovnách či programech pro počítačové vidění, např. OpenCV nebo Matlab.

2.2.1. Konverze barevných formátů

Obrazová data, s nimiž se v počítačovém vidění pracuje, jsou reprezentována různými grafickými schématy. V této práci se vyskytuje převážně schéma RGB, ve kterém jsou uložena data. Dále se pracuje s HSV a GrayScale.

- RGB formát je aditivní míchání červené, zelené a modré barvy. Každý bod na obrázku je udáván jako vektor tří elementů, tedy intenzitou tří základních barev.[8]
- HSV formát se skládá ze tří složek: barevný tón(hue), sytost(saturation) a hodnota jasu neboli množství bílého světla (value). Je velice blízký lidskému vnímání barvy a to zejména tónu a sytosti. Tento formát se hojně používá při algoritmech pracujících s obrazem.[8]
- Grayscale vyjadřuje data pouze v odstínech šedi. Každá hodnota je definována na stupnici šedi a výsledný obraz je tak černo-bílý.

Převody mezi těmito formáty nejsou triviální záležitostí a jsou pro ně definovány složité vzorce. Na obrázku 2.2 lze vidět ukázku barevných schémat.

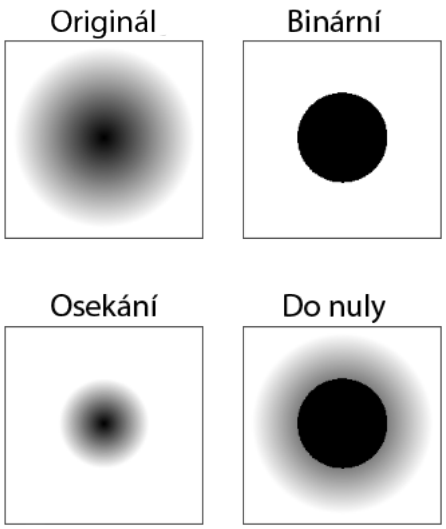


Obrázek 2.2.: Barevná schémata zleva: RGB, HSV a GrayScale

2.2.2. Práhování

Práhování[2] je nejjednodušší z metod segmentace obrazu. Metoda spočívá v definování hranice hodnot, která bude sloužit pro rozhodnutí, bude-li daný bod v obraze akceptován či nikoliv. To, co se bude dít s akceptovaným a neakceptovaným bodem, záleží na konkrétní metodě. Mezi základní patří:[9]

- Binární - Akceptovaný bod je nastaven na 1, ostatní na 0.
- Osekání - Akceptovaný bod je ponechán. Ostatní jsou zahozeny.
- Do nuly - Pokud je bod menší než daná hranice, tak je nastaven na 0. Ostatní jsou ponechány.



Obrázek 2.3.: Metody práhování převzato z [9]

Mezi složitější metody práhování patří adaptivní práhování, kde práh není stejný po celou dobu běhu algoritmu, ale je určován dynamicky. Dá se tím docílit lepších výsledků při různých světelných podmínkách. Hodnota práhu může být vypočítána průměrem okolních hodnot nebo pomocí váženého součtu okolních bodů, kde váhy jsou Gaussovo okno.[9]

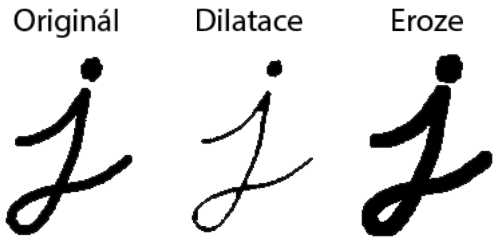
[2] Z anglického threshold

2.2.3. Morfologické operace

Morfologické operace představují zpracování snímků založených na tvarech. Ve struktuře se generuje vstupní obraz na výstupní obraz. Uplatnění nachází především v oblastech spojování nesourodých prvků v obraze, hledání intenzity hrbolů nebo dírek v obraze či změny tvaru a velikosti geometrických objektů. Mezi základní morfologické operace patří eroze a dilatace.[10]

Dilatace se skládá z obrazu (A), který v sobě skrývá jádro (B) určitého tvaru, nejčastěji čtverce nebo kruhu. Jádro má definován tzv. kotevní bod, který představuje střed jádra. Toto jádro je skenováno přes obraz. Počítáním maximální hodnoty pixelu překrytého jádra B a obrazového pixelu v bodě poloze kotvy způsobí světlé oblasti v obraze. Dilataci můžeme vyjádřit jako sjednocení posunutých bodových množin. Uplatní se především v případech nutnosti zaplnění malých děr, úzkých zálivů, zvětšení objektů či použití jako stavební kámen složitějších operací. [10]

Eroze je považována za sestru dilatace. Umožňuje spočítání lokálního minima přes oblast jádra. Jádro B je skenováno přes obraz a výpočtem minimální hodnoty vrátí obrazový pixel kotevního bodu.[10]



Obrázek 2.4.: Morfologické operace převzato z [10]

2.2.4. Vyhlazování obrazu

Vyhlazování obrazu, mnohdy také nazývané jako rozmazávání, je často používaná a jednoduchá operace při zpracování obrazu. Díky této operaci dosáhneme vyhlazení obrazu, například za účelem snížení šumu. Pro vyhlazování jsou používány nízkoprůchodové, nejčastěji lineární filtry, jako konvoluční matice. Tím se odstraní vysokofrekvenční obsah,

např. hrany nebo šum.[11] Nejpoužívanější filtry jsou:

- Normalizovaný filtr - pro každý pixel je spočítán průměr z jeho samého a okolí. Velikost okolí závisí na zvolené velikosti konvoluční matice. Filtr se používá pro odstranění vysokých frekvencí v obraze.[11]
- Gaussův filtr - konvoluční matice je vypočtena pomocí 2D Gaussovy rovnice. Jelikož by byl výsledek nekonečný, musí se specifikovat velikost matice a do ní se pak spočtou váhy Gaussova vrcholu. Filtr se používá pro odstranění šumu a vyhlazení obrazu.[11]
- Medián filtr - algoritmus vezme všechny pixely z konvoluční matice a určí jejich medián. Jím je pak nahrazen zpracovávaný pixel. Filtr se využívá pro odstranění "sůl a pepř"šumu.[11]
- Bilaterální filtr - je podobný Gaussovu filtru, ten bere v potaz všechny body, které se nachází v konvoluční matici, ale již neřeší jejich intenzitu. Bilaterální filtr proto používá ještě další funkci a zahrnuje pouze ty body v matici, jejichž intensita je podobná, jako právě zpracovávaného. To má za následek, že filtr vyhlazuje obraz, ale zachovává hrany. [11]

2.2.5. Detektory hran

Detektory hran jsou velmi důležité pro algoritmy počítačového vidění. Neurofyziologický a psychofyzický výzkum ukazuje, že pro zrakové vnímání vyšších organismů jsou důležitá místa v obraze, kde se náhle mění hodnota jasu. Místa jež odpovídají významným hranám, nesou více informací než jiná místa v obraze. Je zde proto možné značně zredukovat množství dat, aniž by došlo k významné redukci informací o obraze.[8]

Mezi nejpoužívanější hranové detektory patří Cannyho hranový detektor, který také završil hledání po ideálním detektoru hran. Byl vymyšlen Johnem F. Cannym v roce 1986 a míří k uspokojení tří kriterií:

- Minimální počet chyb - algoritmus musí detekovat všechny důležité hrany a neměly by být detekovány žádné falešné výsledky.

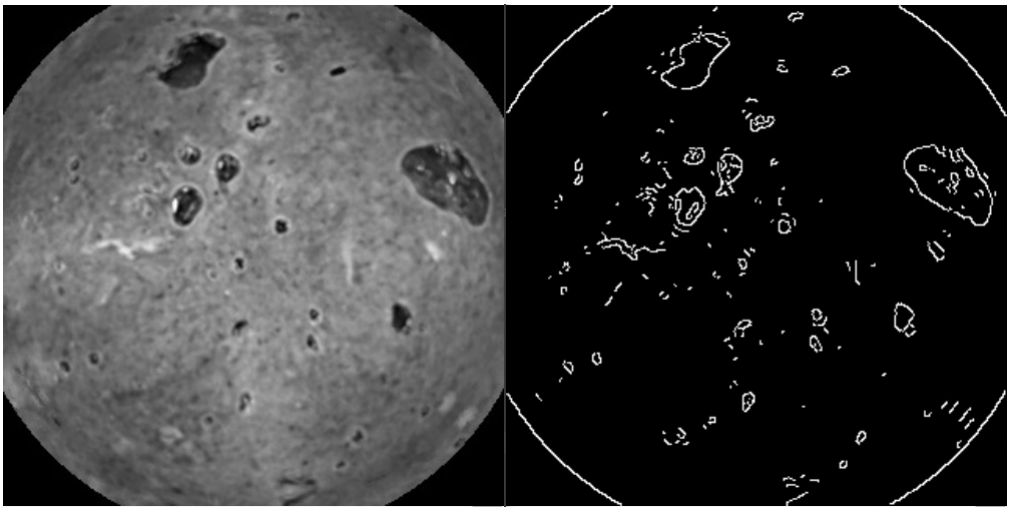
Obrázek 2.5.: Ukázka filtrů z průběhu práce

- Přesnost - rozdíl polohy skutečné a detekované hrany musí být minimální.
- Jednoznačnost - pouze jeden výsledek pro jednu hranu. Toto je částečně pokryto bodem jedna - pro jednu hranu nelze detekovat více hran, ostatní by měly být falešné.

Cannyho algoritmus je se skládá z následujících kroků:

1. Filtrování šumu obrazu pomocí Gaussovského filtru.
2. Zjištění intensity gradientu pomocí Sobelova operátoru - aplikace konvolučních matic
3. Potlačení bodů, které nejsou lokální maxima vzhledem ke směru gradientu. Tímto krokem se odeberou body, které nejsou považované za hranu. Pouze slabé linie zůstanou.
4. Práhování s hysterezí zajistí výběr pouze významných hran. Určí se horní a dolní hranice práhování a hrany jsou pak vybrány na základě následujících pravidel:
   - Pokud je hodnota gradientu větší než horní hraníce, pak je pixel přijat jako hrana.
   - Pokud je hodnota gradientu menší než horní hraníce, pak je pixel odmítnut.

- Pokud je hodnota gradientu mezi hranicemi, je pixel přijat jen tehdy, je-li spojen s jiným pixelem, jehož hodnota gradientu je větší než-li horní hranice.

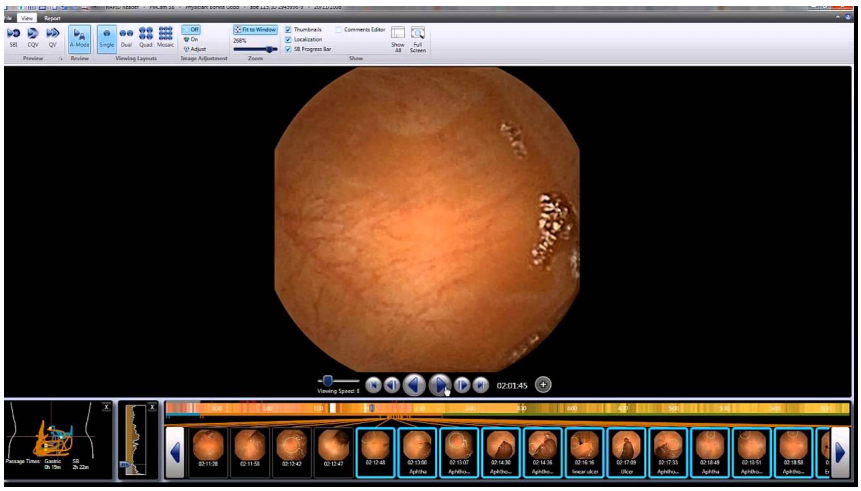


Obrázek 2.6.: Ukázka Cannyho detektoru z průběhu práce

2.3. Aktuální stav trhu

2.3.1. Rapid Software v8

Rapid Software v8 je vyvinut společností Given Imaging, která mimo jíné vyrábí i kamery PillCam SB3. Jejich software je proto dodáván s těmito kamerami. Software umí spoustu užitečných věcí, ale pro charakter této práce je pouze podstatné, že nabízí detekci krve ve střevech. Škoda, že výrobce již neudává s jakou přesností. Na obrázku 2.7 je vidět ukázka rozhraní.[4]



Obrázek 2.7.: Ukázka Rapid v8[12]

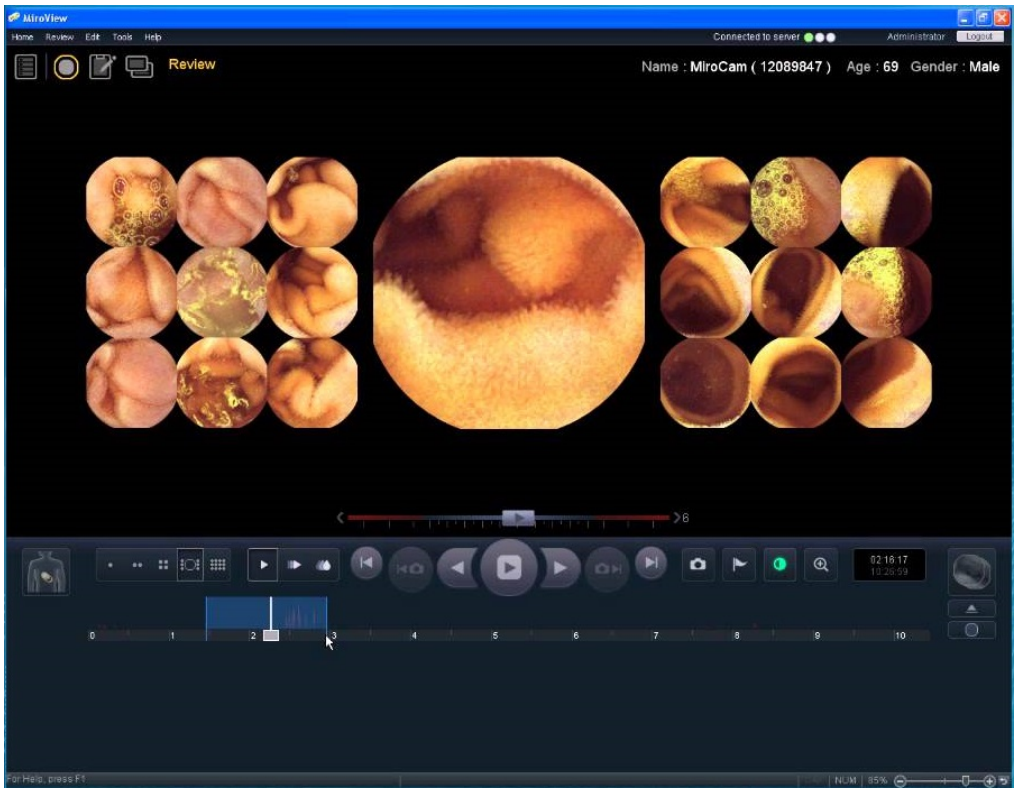
2.3.2. Endocapsule 10 System

Endocapsule 10 System je software vyvinutý společností Olympus, který ho také dodává ke svým kamerám. Software mimo základních věcí má algoritmy pro detekci červené barvy, bublin, nečistot a stejných obrázků. Díky těmto algoritmům je možné odfiltrovat nezajímavé či nehodnotné obrázky a nemusí se tak procházet celý vzorek. Ukázka rozhraní je vidět na obrázku 2.8.[2]

Obrázek 2.8.: Ukázka Endocapsule 10 System [13]

2.3.3. MiroView 2.5

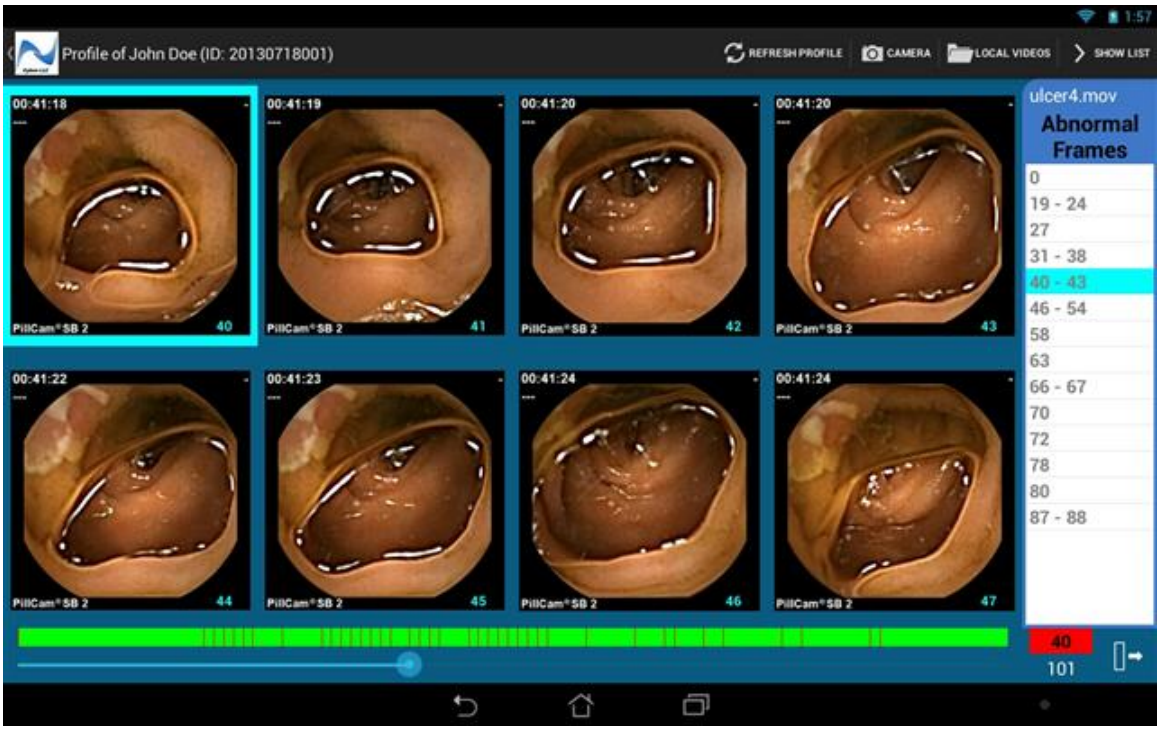
MiroView 2.5 je software vyvinutý společností IntroMedic, která taktéž dodává řešení se svými kamerami. Poskytuje základní funkcionalitu práce s videem a nabízí také automatickou detekci krve. Na obrázku 2.9 je vidět ukázka rozhraní.[3]



Obrázek 2.9.: Ukázka MiroView 2.5[14]

2.3.4. GISentinel

GISentinel je software vyvinutý společností Xyken ve spolupráci s Mayo Klinikou[3]. Jejich software umí detekovat krev, polypy a vředy. Je k dispozici také na platformu Android, což umožňuje značnou mobilitu. Ukázka rozhraní je vidět na obrázku2.10.[16]



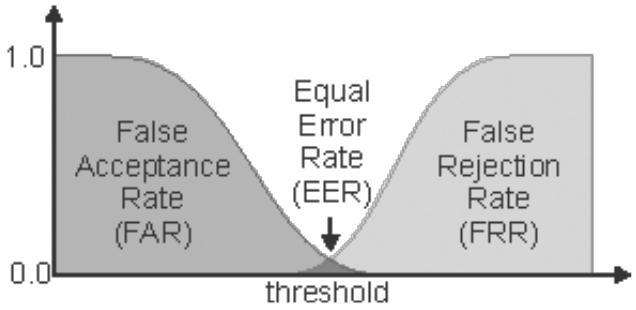
Obrázek 2.10.: Ukázka GISentinel[16]

[3]Jedna z nejlepších klinik v USA, mimo jiné se také soustředí na výzkum. Spolupracuje i s ČR.[15]

3. Definice problému

Hlavním problémem při zpracování dat z kapslové endoskopie je jejich nepřeberné množství. Lékaři musí projít obrázek po obrázku, což pro ně znamená velkou časovou náročnost. Ta by šla využít při řešení konkrétního problému jinak než jeho hledáním. Je proto potřeba nalézt takové řešení, které by tuto práci maximálně zjednodušilo a usnadnilo.

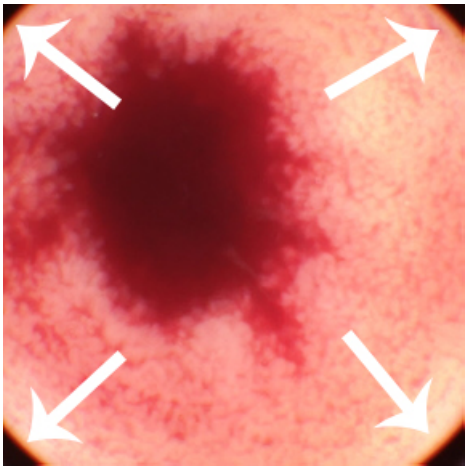
Vyvinutá aplikace by měla být schopná vyhodnotit záznam a detekovat možné krvácení ve střevech. Data z kamery jsou extrahována do proprietárního formátu každého výrobce, ze kterého pak lze extrahovat sekvenci nafocených obrázku buď do formátu AVI, ale s velkou kompresí, a tudíž ztrátou informací, anebo do sekvence po sobě jdoucích JPG obrázku beze ztráty kvality. Aplikace přijme tato data jako vstup. A na výstupu pak bude informace, zda se v daných datech nachází krvácení.



Obrázek 3.1.: EER graf [17]

Je nutné popsat nebo definovat hledaný objekt tak, aby se za pomocí zpracování obrazu dal objekt nalézt s co největší přesností a nejmenším počtem chyb. Bylo by tedy potřeba nalézt EER mezi FAR a FRR viz obrázek, 3.1. Jelikož je v biomedicíně nutné nalézt

všechny pozitivní objekty a ideální stav je 0% FRR, tak se dá akceptovat větší míra FAR. Jinými slovy - je lépe označit více objektů, i když některý z nich může být chybou, než-li minout snímek, v němž se vyskytuje problém.



Obrázek 3.2.: Anomálie při sběru dat

Největším problémem je definování hledaného objektu, neboť kvalita dat není kvůli nízkému rozlišení kamer příliš velká. A navíc, každý výrobce má specifické barevné schéma, rozlišení a artefakty, které vznikají při sběru dat. Ukázkou může být obrázek 3.2, jenž znázorňuje, že tato kamera má u okrajů viditelný pruh barev, který pak může způsobovat problémy. V datech se také vyskytuje spousta anomálií, jež komplikují rozpoznávání objektu (různé odlesky, zbytky potravy, podobné barevné rozpoložení, …). Bude tedy potřeba zkombinovat více metod pro dosažení co nejlepšího výsledku.

V následujících kapitolách jsou uvedeny související vědecké práce, které se zabývají definováním stejného problému.

3.1. Detection of bleeding in wireless capsule endoscopy using range ratio color [18]

V článku „Detection of bleeding in wireless capsule endoscopy images using range ratio color“ výsledky byly klasifikovány na možnosti krvácení či neoznačení výskytu krve. Byla zde použita platforma C#.

Analýza obrazu probíhala vždy ve dvou krocích. První krok efektivně rozdělí vstupní videa na ty, které obsahují výskyt krvácení a naopak. V druhém kroku následuje samotná detekce krvavých míst a klasifikace aktivních vzorků krvácení. První testovanou funkcí byla barva, která přinesla pouze selhání. Použitý algoritmus nepřinesl výsledky, které se očekávaly. Druhou testovací funkcí bylo použití parametrů poměr – barva. Tato funkce přinesla lepší výsledky než první. Oba dva algoritmy probíhaly téměř stejně - testovaly se všechny pixely na obrázku o proti podmínce, která se lišila v závislosti na použitém algoritmu. Podmínky je možné vidět na obrázku 3.3.

1. R == 255 && G == 0 && B == 0
2. R >= 75 && R < 125 && G <= 25 && G >= 14 && B <= 15 && B >= 0

Obrázek 3.3.: Použité algoritmy.[18]

Bylo testováno 100 WCE snímků pořízených na různých částech trávícího traktu. Při použití stanovených podmínek byl výsledek první funkce s přesností stanoven na pouhých 48%. Avšak druhá funkce umožnila povýšit přesnost až na 98%. Přehled dosažených výsledků u obou metod můžeme shledat v tabulce 3.1.

Klasifikace	Detekovaných snímků s krví	Detekovaných snímku bez krve	Celková správná hodnota	Přesnost
Pouze červená barva	2	98	48	45%
Rozsah barev	52	48	98	98%

Tabulka 3.1.: Výsledku algoritmů. Přeloženo z [18]

Závěrem můžeme konstatovat, že tato metoda je jednoduchá a přesná.

3.2. „A technique for blood detection in wireless capsule endoscopy images [19]

Článek „A technique for blood detection in wireless capsule endoscopy images“ se zabývá způsobem rozlišování mezi běžnou sliznicí a regionů krvácejících tkání. Opět pomocí WCE snímků. Tohoto způsobu bylo dosaženo identifikaci lišících se oblastí. Lišit se mohou

prostorové nebo spektrální charakteristiky od svého okolí. Oblasti pokryté krví jsou považovány za izolované cíle na pozadí. Celý tento proces je označen za detekce anomálií.

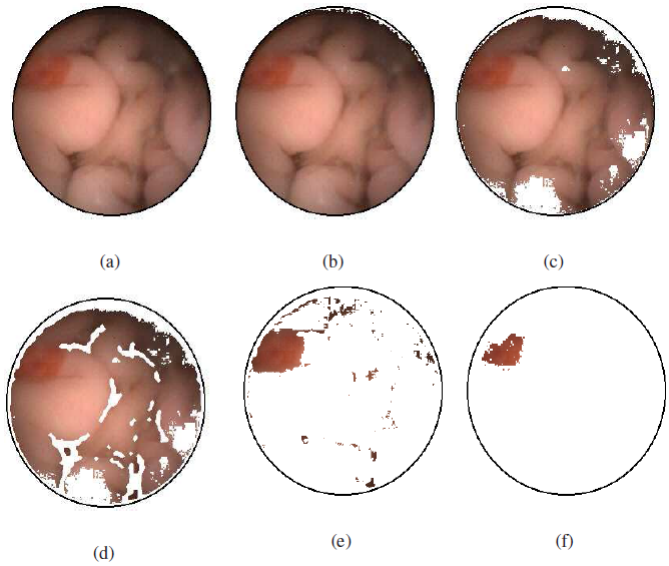
Detekce anomálií se provádí prostřednictvím testování hypotéz daného problému. V případě detekce krve se používá pixel (anomální pixely) při reprezentaci barvy a stanovené hodnoty kovarianční matice. Vše se provádí za předpokladu stacionárního procesu v pozadí. Tento předpoklad se potlačuje především při výskytu vzduchových bublin či organických zbytků. Z tohoto důvodu je aplikován předzpracující stupeň – tedy příprava dat pro detekci anomálií a následně po dokončení spuštění RX algoritmu. Postup vícestupňové detekce krve je následující:

1. Odstranění tmavých míst a detekce červených pixelů - V první fázi se vyloučí regiony, které nejsou detekovány jako užitečné (střevní šťávy, tmavé oblasti) a vyberou se pouze regiony s výskytem krve. Druhá fáze je rozdělena na další pod části. Nejprve se odstraní pixely tmavé velmi barvy, poté jsou červené pixely detekovány jako potencionální krev. Střevní obrazy dokáží poukázat na vysoce červený odstín. Výjimkou jsou snímky pořízené z oblasti tlustého střeva, kde jsou dominantní barvy oranžové až žlutozelené vzhledem k výskytu fekálních ostatků.

2. Hrana maskování - Abychom snížili výskyt anomálií, aplikujeme okrajové maskování. Hrany jsou získány za pomocí Mumford-Stashova algoritmu jako výsledky funkční minimalizace problému. Obraz je dále předán dalšímu zpracování.

3. Detekce krve - V tomto okamžiku jsou shromážděny výsledky z již předchozích fází. Kombinací těchto dvou způsobů třídění dat dosáhneme lepší spolehlivosti konečného výsledku. Kombinací detekcí červených barev, detekcí anomálií a maskováním získáme nekrvácející oblasti.

4. Morfologické operace - Krvácející oblasti se nejeví jako malé a izolované oblasti. Na základě tohoto faktu je užitečné odebrat izolované anomálie pixelů. Na obrázku 3.4 můžeme shledat konečnou podobu detekce krve.



Obrázek 3.4.: Postup algoritmu a.originální obraz;b.odstranění černých pixelů;c.detekce červené barvy;d.maskování hran;e.detekce krve;f.výsledek RX algoritmu [19]

Simulace byly prováděny na 11 sekvencích (8 patologických případů, 3 zdraví jedinci). Každá sekvence se skládá ze 101 rámců o velikosti 256 x 256 pixelů. Přesnost výsledků byla hodnocena za pomocí standardních kriterií v biomedicíně (upřesněno v kapitole 6.1). Na obrázku 3.5 můžeme shledat výkonnost navrhovaného algoritmu.

Výsledky ukázaly vyšší citlivost a specifičnost experimentálních navrhovaných způsobů. Tím pádem se dosahuje uspokojivých výsledků. Nesmí se však opomenout fakt, že použité sekvence jsou komprimovány. V důsledku komprimace dochází k negativnímu ovlivnění zaváděné komprese. Budoucí vývoj bude aplikovat konkrétní algoritmus na ověření nekomprimovaných dat a vykořisťování korelace mezi sousedními rámy a to vše za účelem zlepšení výkonu detekce.

	SE	SP	FAR	MDR
bleeding video1	0.76	0.73	0.13	0.14
bleeding video2	1	0.92	0.08	0
bleeding video3	0.67	0.89	0.099	0.02
bleeding video4	0.93	0.93	0.03	0 .04
bleeding video5	0.94	0.82	0.02	0.05
bleeding video6	0.82	0.97	0.03	0.02
bleeding video7	1	0.57	0.09	0
bleeding video8	0.95	0.70	0.18	0.02
normal video1	-	-	0.11	0
normal video2	-	-	0.11	0
normal video3	-	-	0.06	0
average	0.92	0.88	0.08	0.03

Obrázek 3.5.: Výsledky RX algoritmu [19]

3.3. Problémy detekce artefaktů v reálném prostředí

Z vědeckých článků je zřejmé, že na vybrané databázi testovaných vzorků dosáhli celkem dobré úspěšnosti. Nicméně, nabízená řešení jsou příliš orientována na detekci dle barvy. V prvním případě se bude v reálných datech vyskytovat mnoho FP, protože na každém snímku se objeví alespoň jeden pixel, který bude zapadat do daného rozsahu.

V druhém článku je přidáno odstranění nechtěných anomálií po detekci barev. To může mít za následek zpřesnění výsledků, neexistuje zde však další mechanismus nezávisle na barvě, např. detekování samotného tvaru.

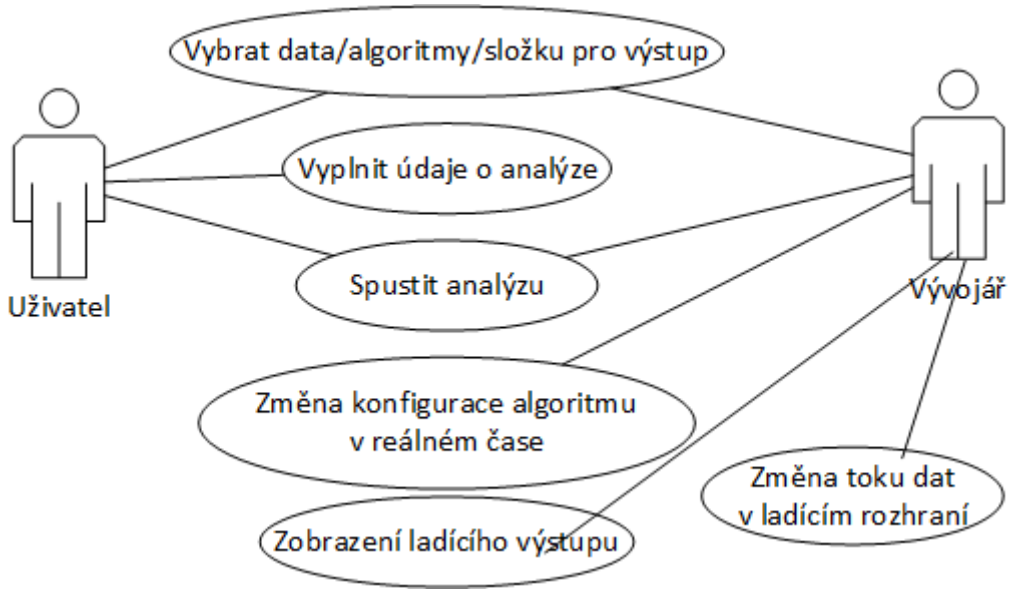
Z článku lze vycházet, jelikož přece jen detekce barvy je stěžejní pro určení krve. Pro přesnější výsledek je však nutné hledaný objekt definovat za pomocí detekce tvaru. Detekováním tvaru bude možné popsat rozdílnost od pozadí (lidský mozek zpracovává informace podobně, je schopný detekovat tvar objektu, aniž by záleželo na barvě).

4. Návrh softwarového řešení

Pro softwarové řešení byla provedena základní analýza požadavků na systém, ze které se specifikovaly funkční a nefunkční požadavky. Na základě těchto požadavků bylo rozhodnuto o zvolených technologiích a architektuře softwaru.

4.1. Analýza požadavků na systém

Diagram užití, který je ztvárněn na obrázku, 4.1 je pro uživatele velice jednoduchý, protože aplikace má být intuitivní a slouží pouze k jednomu účelu a to zadání nové analýzy dat. Pro vývojáře bude nabízet možností více a to zejména ladění algoritmů.



Obrázek 4.1.: Diagram užití

4.1.1. Funkční požadavky

Funkční požadavek	Popis
Vybrání dat/algoritmů/ složky pro výstup	Aplikace zobrazí dialog pro výběr dat, algoritmů a složky pro výstup.
Vyplnění údajů o analýze	Aplikace nabídne formulář pro vyplnění názvu a popisu analýzy.
Spuštění analýzy	Aplikace spustí analýzu dat.
Změna konfigurace algoritmu v reálném čase	Aplikace umožní změnu konfigurace algoritmů v reálném čase pro lepší ladění algoritmů.
Zobrazení ladícího výstupu	Aplikace zobrazí ladící výstup aktuálně zpracovávaného snímku, jako nové okno s obrázkem snímku.
Změna toku dat v ladícím rozhraní	Aplikace umožní vývojářům přehrávat již analyzované video, posouvat rychlost přehrávání či vybrat konkrétní snímek.

Tabulka 4.1.: Funkční požadavky na aplikaci

4.1.2. Nefunkční požadavky

Nefunkční požadavek	Popis
Intuitivní ovládání	Aplikace bude jednoduchá k použití i bez manuálu.
Optimalizace	Aplikace musí zvládat zpracovávat velké množství dat a být co nejefektivnější.
Generování výstupních souborů	Aplikace bude umět generovat výstup do předem zvolených formátů.
Multiplatformní použití	Aplikace by měla být nezávislá na konkrétní platformě.

Možnost implementace více GUI	Aplikace by měla být navržena tak, aby bylo možné vystavět na ní různá rozhraní např. desktop nebo web.
Oddělené prostředí	Vývojové prostředí bude oddělené od produkčního, aby zbytečně nedocházelo k nepředvídatelným chybám. Bude pouze využívat část společné funkcionality.

Tabulka 4.2.: Nefunkční požadavky na aplikaci

4.2. Použité technologie a knihovny

Tato kapitola se zabývá použitými technologiemi a knihovnami. Technologie byly zvoleny na základě autorových zkušeností s nimi a také vhodností pro charakter práce. Následující výčet ukazuje hlavní technologie použité v práci, mimo ně jsou ještě použity různé podpůrné knihovny od Apache, které již však nejsou klíčové pro funkčnost a spíše usnadňují práci.

- Java - objektově orientovaný programovací jazyk vyvinutý společností Sun Microsystems a později odkoupen společností Oracle. Java je multiplatformní jazyk, tudíž není problém spustit jeden kód na více operačních systémech či zařízení. V práci je použita Java 1.8_0.45 a je tak minimální podporovanou verzí pro spuštění software. V práci jsou využity nejmodernější přístupy, které Java 1.8 přináší.

- JavaFX8 - technologie pro tvorbu bohatých klientských aplikací. Je jednou z možných implementací GUI aplikace (více o GUI v kapitole **??**). Od verze Javy 1.8 je již standardem pro tvorbu GUI a nahrazuje starý swing.

- OpenCv - knihovna pro počítačové vidění pod licencí BSD. Knihovna nabízí mnoho nástrojů pro práci s obrazovými daty a je vysoce optimalizovaná pro produkční aplikace. Využívá nativní kód přímo pro určité platformy a hardwarovou akceleraci pomocí OpenCL, pokud je možná. Je hlavním nástrojem pro výpočty a manipulaci s obrazem, v Jave jsou napsány pouze minoritní výpočetní operace[7]

- Apache Maven - nástroj pro sestavování programu, řešení automatických závislostí na knihovny a správy modulů projektu.[20]

- Apachae POI - knihovna pro manipulaci různých formátů založených na Office Open XML standars a Microsoft OLE 2 Compound Document format. Lze tak vytvářet a pracovat se soubory XLS a XLSX, případně číst další soubory vytvořené z Microsoft Office.[21]

- Apache PDFBox - knihovna, která umožňuje vytvářet PDF dokumenty, manipulovat s existujícími a získávat obsah dokumentů.[22]

- ProGuard - nástroj, který minimalizuje, optimalizuje, obfuskuje a před ověřuje Java třídy. Umí detekovat a odstranit nepoužívané třídy, pole, metody a atributy pro zmenšení výsledného kódu. Optimalizuje bytecode a odstraňuje nepoužívané instrukce. Přejmenovává zbývající třídy, pole a metody použitím krátkých nevýznamných jmen, aby nebylo jednoduché program dekompilovat. Přeznačuje a před ověřuje existující třídy pro Java 6 a vyšší, aby tak zrychlil jejich načítání.[23] V práci je použit jako plugin do Mavenu při sestavování programu.

- Launch4J - nástroj pro obalování spustitelného Java programu nativním kódem pro Windows a vytváření tak .exe souborů. Nástroj také umožňuje definování minimálního JRE a případě přesměrování na stránky pro stažení případně je možné vložit celé JRE, jako knihovnu, kterou nástroj bude načítat pří spuštění programu.[24] V práci je použit jako plugin do Mavenu při sestavování programu.

4.3. Vývoj algoritmů

K zjištění přítomnosti krve v trávícím traktu nebylo možné použít stejné principy pro všechny možné případy výskytů krvavých oblastí. Proto byly vyvinuty dva algoritmy pro detekci velkých a malých skvrn, které používají víceméně stejné metody, ale odlišné postupy. Jediná stejná část pro oba dva algoritmy je, že se na začátku zpracování ořízne obraz kvůli vznikajícím barevným anomáliím na okrajích snímací části kamery. Popis algoritmů je zde obecný bez použití konkrétních nakonfigurovaných hodnot, v příloze A a B jsou vidět konkrétní hodnoty a jejich vysvětlení.

4.3.1. Detekce malých skvrn

Algoritmus je rozdělen na dvě nezávislé částí, které se pak na konci porovnají, a v případě schody se snímek vyhodnotí pozitivně.

První část je zaměřena na filtrování obrazu dle určitého barevného rozmezí, která je zpřesněna dalšími technikami.

1. Převod barevného formátu z BGR do HSV. S HSV formátem je možné přesněji vyjádřit barevné rozmezí, které je potřeba zachovat, viz obr. 4.2.E.

2. Filtrace matice dle definovaného barevného rozmezí. Výsledkem operace je nová matice, která obsahuje pouze vybrané barvy a jejich hodnoty jsou reprezentovány jako binární data, dochází tedy ke ztrátě barevné informace, viz obr. 4.2.F.

3. Pro všechny spojité oblasti, které vznikly z předchozího kroku, se vypočte jejich oblast a ponechají se pouze ty, které jsou větší než definovaná hodnota. Tím dojde k odstranění nevýznamných miniaturních oblastí, které jsou pravděpodobně vzniklé šumem, případě jinými anomáliemi. A také se vyplní nevybarvené oblasti uvnitř spojitých oblastí, viz obr. 4.2.G.

4. Na všechny body s hodnotou 1 ve vyfiltrované matici se zkopíruje barevná informace z původní HSV matice, tím se vrátí barevná informace k dalšímu zpracování, viz obr. 4.2.H.

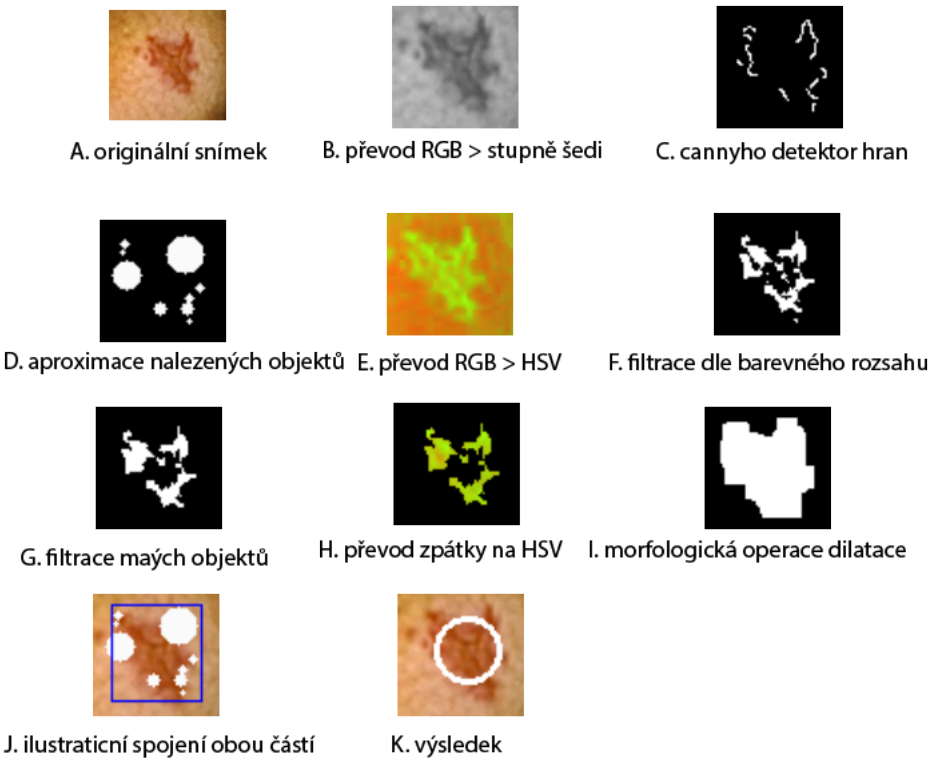
5. Provede se morfologická operace dilatace dle zadané konfigurace. Tím dojde ke spojení všech blízkých bodů v matici a vzniknou tak větší spojité oblasti, kde by potencionálně mohla být krev, viz obr. 4.2.I.

6. Pro všechny spojité oblasti, které vzniknou po dilataci, se vypočte jejich ohraničující čtverec, který se vyřízne z matice.

7. V každém takto vzniklém čtverci se provede znovu filtrování dle barvy, ale s jinou "přisnější"konfigurací. V nově vzniklé matici už se nefiltrují malé artefakty a provede se rovnou dilatace. To má za následek, že se v podstatě nic nezmění, nebo se čtverec

rozpadne na více oblastí, anebo dojde k vyřazení čtverce z potencionálních detekovaných oblastí.

8. V každém takto upraveném čtverci se najdou všechny spojité oblasti, ohraničí se čtvercem a uloží se do pole, které v sobě drží ostatní ROI, viz obr. 4.2.J.



Obrázek 4.2.: Ukázka průchodu algoritmu detekce malých skvrn.

V druhé části algoritmu se detekují hrany a určí se tak potencionální objekty, které jsou odlišné od pozadí. Postup je následující:

1. Převod barevného formátu z GRB na odstíny šedi, jelikož je k detekci hran vhodnější, viz. obr. 4.2.B.

2. Gaussovo rozmazání dle definované konfigurace.

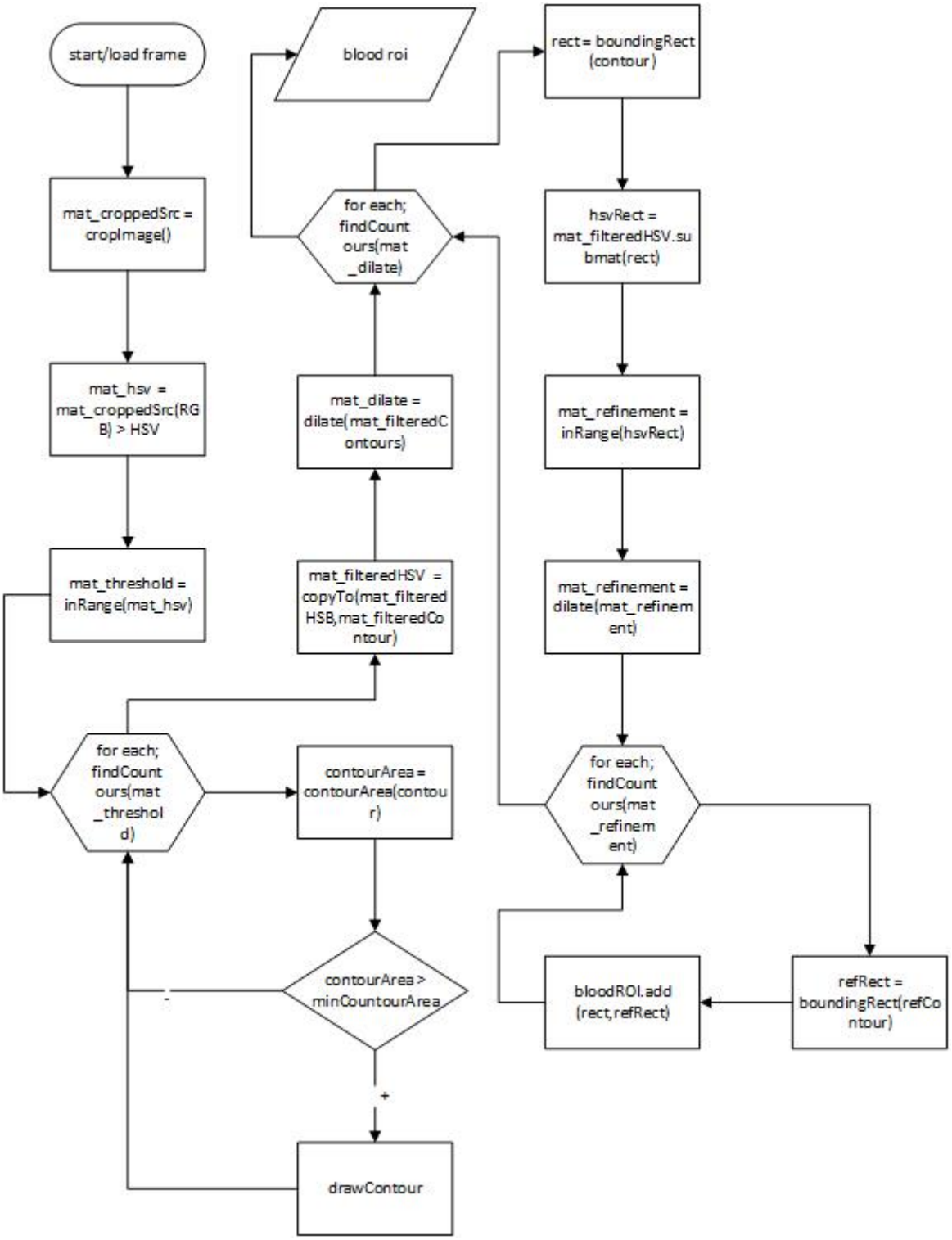
3. Cannyho detektor hran dle definované konfigurace, viz. obr. 4.2.C.

4. Pro všechny spojité nalezené hrany se vypočte minimální obklopující kružnice. Ta slouží, jako jednoduchá aproximace hledaného objektu.

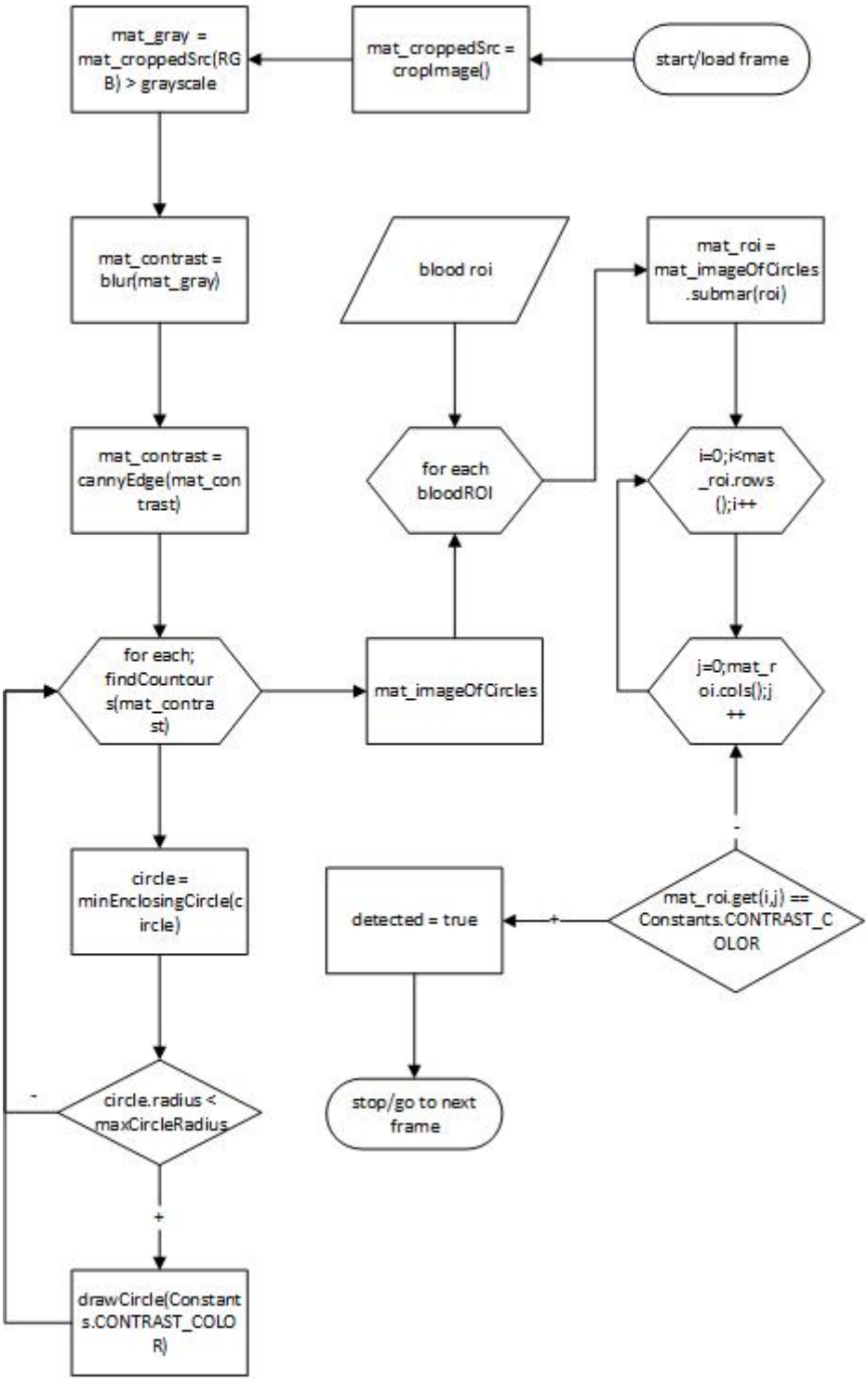
5. Jelikož vznikají anomálie při detekci hran a zejména okrajové části kamery jsou často zahrnuty ve výsledku, tak je nutné vybrat pouze ty kružnice, které nepřesahují maximálně definovaný průměr. Je v podstatě nemožné, aby se tímto odstranila hrana, která se nemá odstranit, protože tento algoritmus detekuje pouze menší skvrny a anomálie, které vznikají jsou velmi velké.

6. Zanesení zbylých kružnic do pomocné matice, viz. obr. 4.2.D.

Jakmile jsou k dispozici obě části algoritmu, tedy matice, která obsahuje pomocné kružnice, a list všech ROI. Tak dojde k vyhodnocení, zda-li je matice pozitivní. To je provedeno iterací přes všechny čtverce v listu a pokud se v oblasti konkrétního čtverce nachází alespoň jeden bod z kružnice, tak je místo pozitivní, viz obr. 4.2.K.



Obrázek 4.3.: Vývojový diagram první části algoritmu detekce malých skvrn.

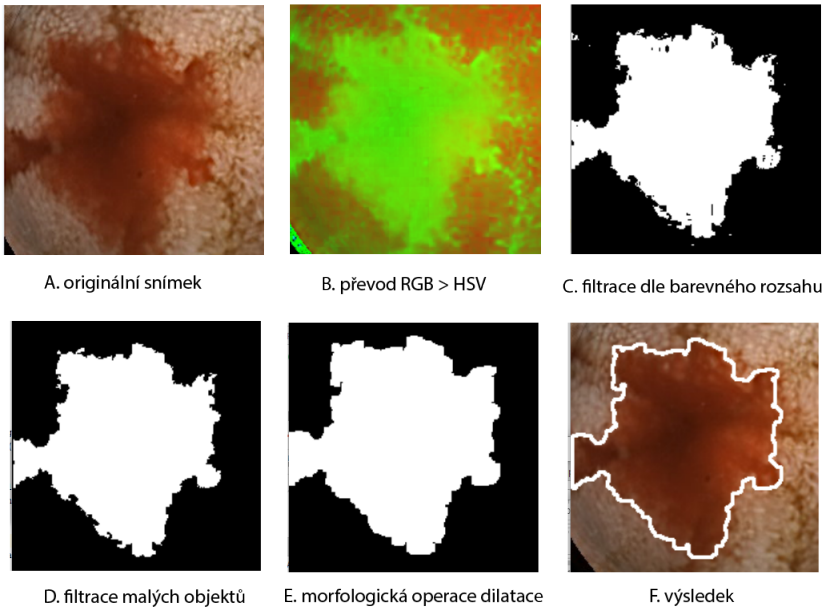


Obrázek 4.4.: Vývojový diagram druhé části algoritmu detekce malých skvrn a spojení obou částí.

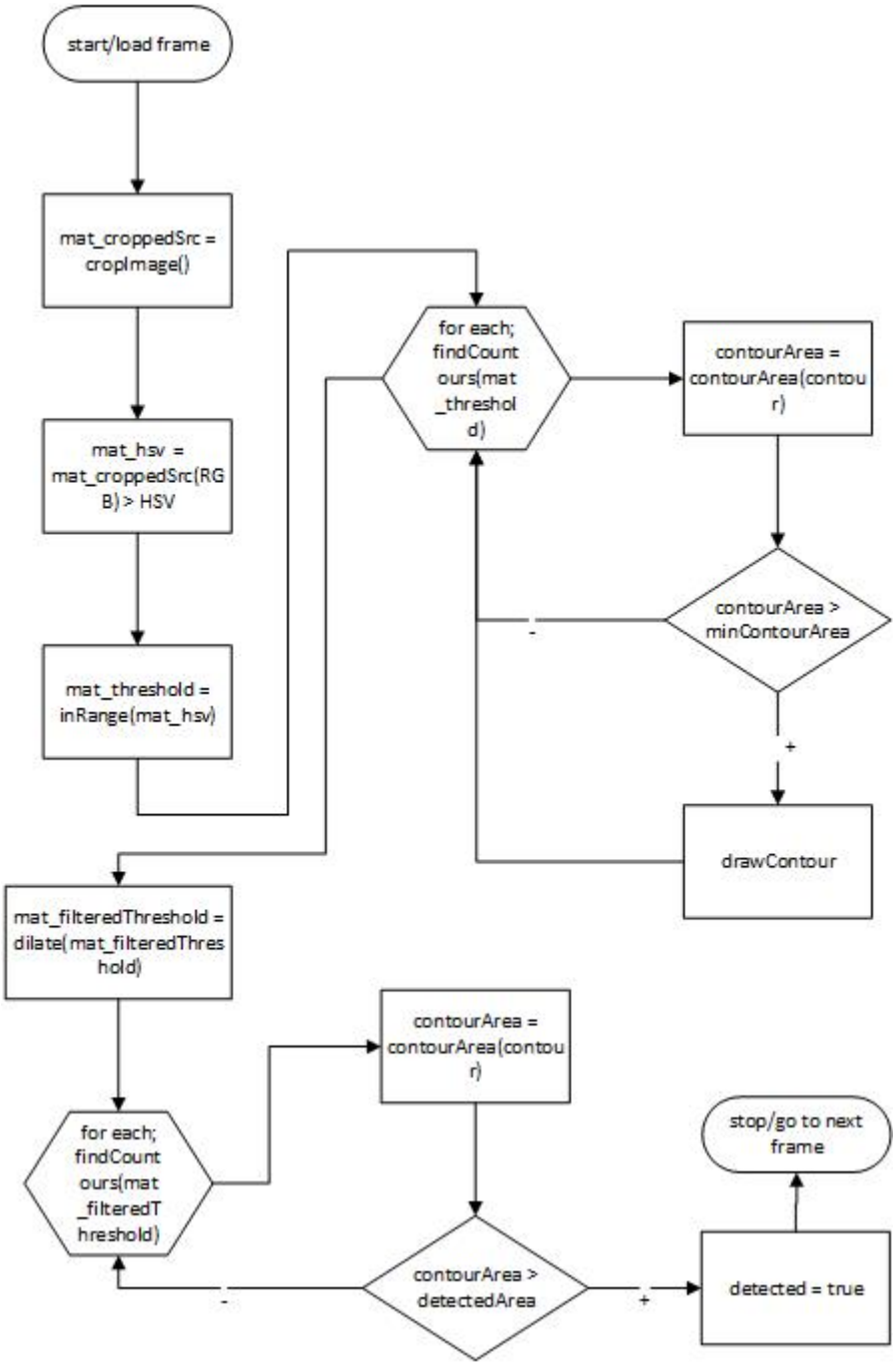
4.3.2. Detekce velkých skvrn

Konstrukce druhého algoritmu je značně jednodušší a přímočařejší než-li detekce menších skvrn. Jedná se o nalezení velké oblasti se stejným barevným spektrem. Postup je následující:

1. Převod barevného formátu z BGR do HSV, viz obr. 4.5.B.
2. Filtrace matice dle definovaného barevného rozmezí, viz obr. 4.5.C.
3. Filtrování miniaturních částí, viz obr. 4.5.D.
4. Morfologická operace dilatace, viz obr. 4.5.E.
5. Pro všechny spojité oblasti v matici se zjistí jejich plocha. Ta se porovná proti definované procentuální velikosti celkové oblasti a pokud je větší, tak se rámec vyhodnotí pozitivně. Např. pokud je spojitá oblast větší než-li 15% celkové oblasti, viz obr. 4.5.F.



Obrázek 4.5.: Ukázka průchodu algoritmu detekce velkých skvrn.



Obrázek 4.6.: Vývojový diagram algoritmu velkých skvrn.

5. Implementace software

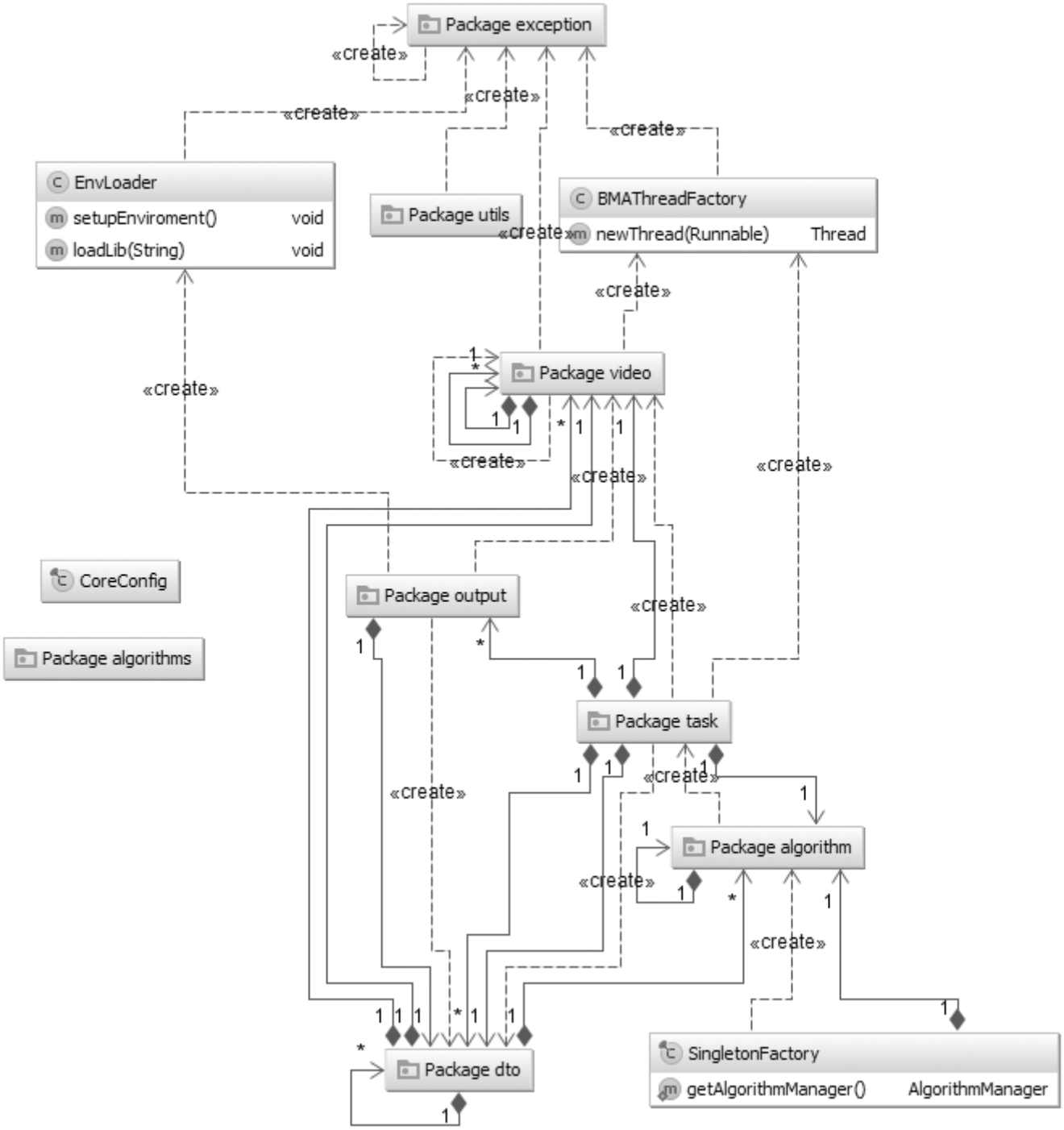
Software je rozdělen na čtyři moduly. Každý modul v sobě uzavírá specifickou funkcionalitu. Hlavním modulem je modul core, kde je veškerá logika programu. Ostatní moduly už jsou jen pro uživatelské rozhraní, které je případně možné implementovat i jinak např. je možné udělat webové rozhraní.

5.1. Modul Core

Modul core je jádro celého programu, uzavírá v sobě funkcionalitu pro načítání algoritmů, práci s obrazem, generování výstupních souboru a práci s více vlákny. Na obrázku 5.1 je vidět základní schéma modulu. Modul je rozdělen do několika balíčku, z nichž stěžejní jsou algorithm, video, task a output. Ostatní balíčky obsahuje pomocné třídy, jako jsou různé utility třídy, DTO, vyjímky, ...

V modulu core je ještě několik tříd, které nejsou zařazené do žádného balíčku. Třída EnvLoader má na starost správné načtení konkrétních knihoven pro danou platformu, jelikož knihovna OpenCV využívá nativní knihovny zkompilované pro každou platformu zvlášť a v Jave jsou tyto knihovny volány přes JNA, tak je nutné načíst nativní knihovny do systému. Momentálně je v programu podporována platforma Windows x86 a x64.

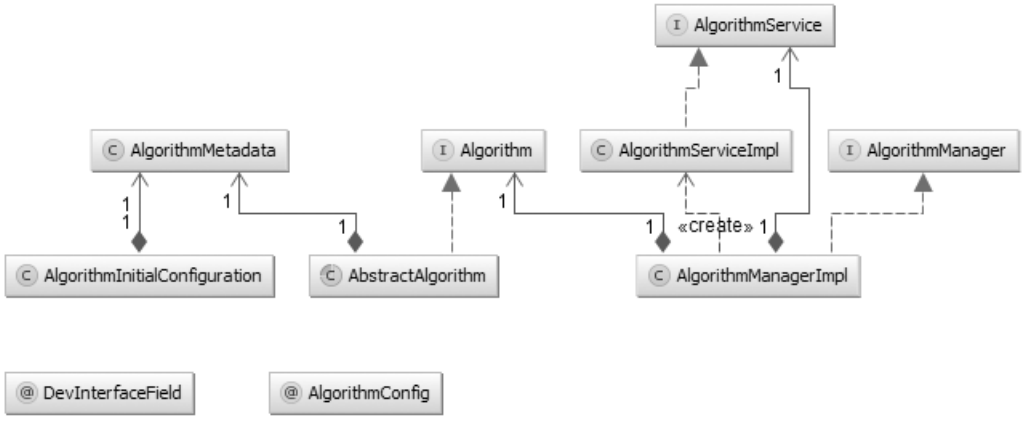
Další třídou je BMAThreadFactory, která implementuje rozhraní ThreadFactory. Třída slouží, jako factory pro executor servise. Vlákna, která se zde vytvoří, jsou řazena pod jednu skupinu a ta může mít svůj volitelný název. Všechna vlákna jsou pak vytvořena s prioritou normal a neběží v režimu "démon". Veškerá práce s vlákny v programu pak probíhá pomocí ExecutorService, kde je využito BMAThreadFactory pro tvorbu nových vláken.



Obrázek 5.1.: Modul core

5.1.1. Balíček Algorithm

V balíčku algorithm, jehož schéma je vidět na obrázku 5.2, je funkcionalita pro načítání počáteční konfigurace algoritmu a práci s načteními algoritmy. Veškerou funkcionalitu balíčku v sobě uzavírá třída AlgorithmManager, přes kterou se volá AlgorithmService a také v sobě drží aktuálně načtené algoritmy. V programu se vytváří pouze jedna singleton instance manageru a ta se pak získává přes SingletonFactory.



Obrázek 5.2.: Balíček algorithm

Pro vytvoření nového algoritmu je nutné, aby nová třída implementovala rozhraní Algorithm nebo dědila již od základní implementace AbstractAlgorithm. Dále může volitelně obsahovat třídu, která slouží jako konfigurace algoritmu. Tato třída musí mít anotaci AlgorithmConfig. V konfiguraci je pak možné anotovat jednotlivé atributy anotací DevInterfaceField, ty pak budou na základě konfigurace v anotaci vidět ve vývojovém rozhraní. Konkrétní nastavení algoritmu je uloženo v externím XML souboru, který je mapován na třídu AlgorithmInitialConfiguration. Díky tomu je možné mít více algoritmů s různou konfigurací. Ukázka nového algoritmu je zobrazena na obrázku 5.3. Načtení algoritmu pak probíhá následovně:

1. Procházení všech XML souborů.
2. Soubor je rozmaršálován na třídu AlgorithmInitialConfiguration pomocí JAXB.

3. Pomocí reflexe je nalezena a vytvořena nová instance třídy, která je uvedena v className, viz. obrázek 5.3.

4. Nastavení metadat algoritmu.

5. Vytvoření nové konfigurace algoritmu a nastavení atributů na základě hodnot v XML. Vše je pak dosazeno do instance třídy pomocí reflexe.

6. Vložení inicializovaného algoritmu do mapy k ostatním algoritmům.

```
public class BigBlobAlgorithm extends AbstractAlgorithm {

    private BBConf conf;

    @Override
    public AlgorithmProcessResult process(VideoFrame frame) {
```

```
@AlgorithmConfig
public class BBConf {

    @DevInterfaceField(fieldType = DevInterfaceField.Type.SLIDER, max = 2, min = 0, inc
    private Double crop;
    @DevInterfaceField(fieldType = DevInterfaceField.Type.SLIDER, max = 179, min = 0)
    private Double lowH;
```

```
<algorithmInitialConfiguration>
    <algorithmName>BloodDetection</algorithmName>
    <className>cz.ls.bma.core.algorithms.coloredBlob.ColoredBlobAlgorithm</className>
    <algorithmMetadata>
        <author>Lukas Sulik</author>
        <date>20.5.2015</date>
        <version>1.0</version>
    </algorithmMetadata>
    <properties>
        <entry>
            <key>highH</key>
            <value>10</value>
        </entry>
        <entry>
            <key>lowH</key>
            <value>8</value>
        </entry>
```

Obrázek 5.3.: Ukázka nového algoritmu

5.1.2. Balíček Video

V balíčku video jsou pouze dvě třídy: Buffer a VideoFrame. VideoFrame je třída, která slouží k uchovávání informace o jednom obrazovém rámci. Má v sobě matici, která reprezentuje obrázek, číslo rámce a zámek. Zámek je zde proto, že se s rámcem pracuje ve více vláknech a je nutné poznat, zdali už byl rámec zpracován všemi algoritmy. Zámek je vyjádřen třídou AtomicInteger a je to číselná hodnota, která se nastavuje vždy při vytváření rámce (popsáno níže), a jedná se o vyjádření počtu algoritmů, které s ním budou pracovat. Po zpracování algoritmem se hodnota zámku sníží o jedna. VideoFrame také obsahuje pomocnou funkcionalitu pro vývojové rozhraní, tato funkcionalita je aktivní pouze pokud je vývojové rozhraní aktivní. Vývojová funkcionalita umožňuje dočasně ukládat matice z průběhu zpracování algoritmu do mapy, ale tato možnost funguje pouze pro jedno vlákno.

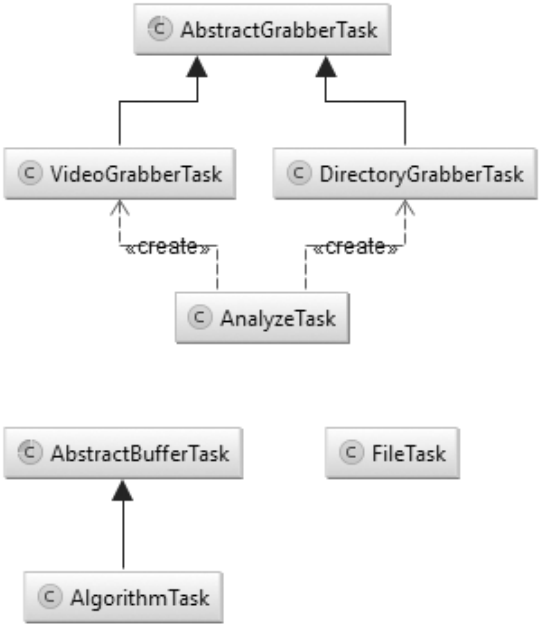
Třída Buffer slouží, jak již název napovídá, jako zásobník pro VideoFrame. Samotný zásobník je řešen pomocí rozhraní ConcurrentNavigableMap a konkrétní implementace ConcurrentSkipListMap. S touto třídou je možno pracovat na více vláknech bez rizika deadlocku. Pořadí hodnot v mapě je tříděno přirozeně pomocí klíče a datová struktura skip list umožňuje rychlé vyhledávání v mapě dle klíče. Velikost bufferu se dá řídit definovanou konstantou, po naplnění bufferu nelze přidávat další elementy. To zajišťuje metoda addFrame, která slouží pro vytvoření a vložení nového framu, pokud je již buffer naplněn tak nelze nový frame přidat. Čištění bufferu probíhá automaticky. Při vytvoření objektu je spuštěna čistící sekvence, která odmazává všechny framy s nulovou hodnotou zámku. Sekvence běží do té doby, než je buffer uzavřen a je prázdný. Uzavření bufferu je stav, kdy už do něj nelze přidat další hodnoty, stav je reprezentován atributem typu boolean.

5.1.3. Balíček Task

Další částí programu je balíček Task, jehož schéma je vidět na obrázku 5.4. Třídy v něm obsažené slouží k vykonávání asynchronních úkolů a implementují buď Runnable nebo Callable, v případě, že je potřeba čekat na výsledek. V třídách grabber je načítání souborů z disku. Jak název napovídá tak buď videa, anebo celé složky (kde se načítají obrázky). K

načtení je v obou případech použita knihovna OpenCV, která načte jednotlivé soubory a udělá z nich matice ve formátu BGR (pouze přeházené kanály o proti RGB, jinak je formát stejný). Načtená matice je pak vložena do třídy buffer, kde bude čekat na další zpracování.

Pro zpracování videa se využívá třída AlgorithmTask, která je potomkem AbstractBufferTask. V této třídě dochází ke zpracování dat algoritmem, metody pro sběr dat jsou definovány v abstraktním předkovi. Po vytažení dat se spustí metoda algoritmu, která zjistí, jestli se v konkrétním snímku nachází nějaká anomálie. V případě pozitivního výsledku se snímek uloží do přepravky, která se pak použije při generování výstupu. Jakmile je operace dokončena, z video snímku se odstraní zámek pro konkrétní algoritmus a pošle se signál pro aktualizaci průběhu.



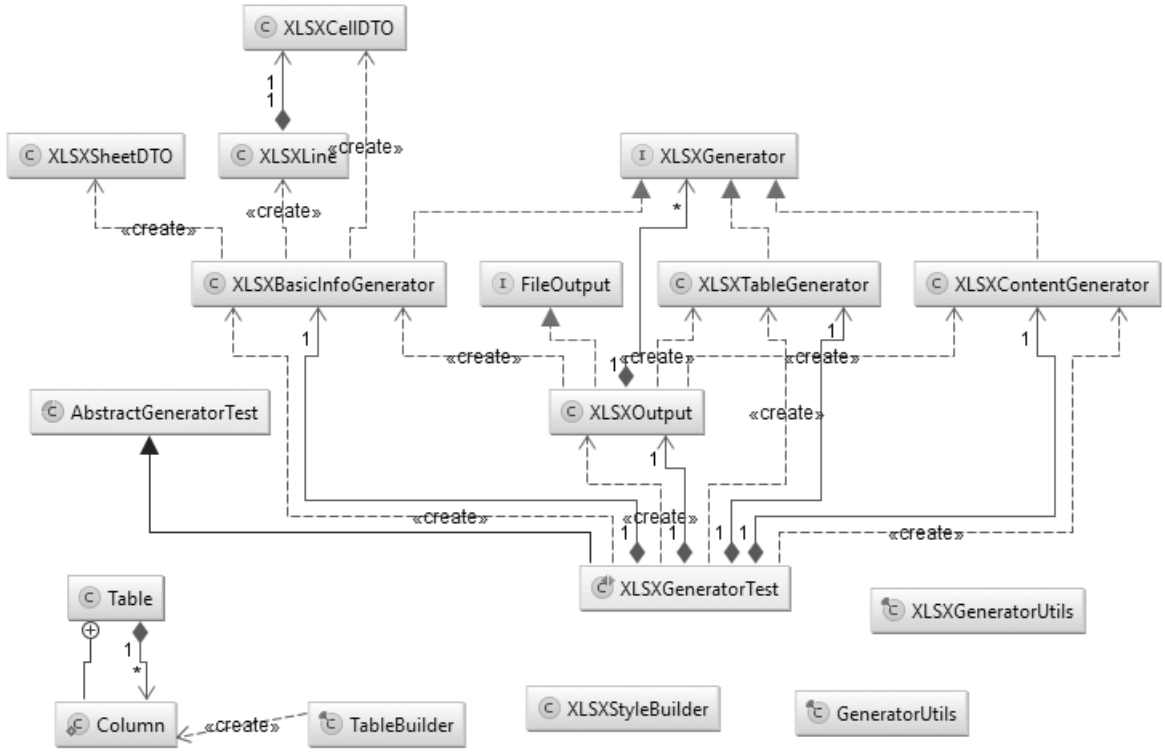
Obrázek 5.4.: Balíček Task

Třída AnalyzeTask spojuje funkcionalitu všech předchozích tasku. Pro každý soubor se vytvoří buffer a spustí se GrabberTask na novém vlákně. Jakmile je GrabberTask zapnut, tak se vyžádají od AlgorithmManagera všechny aktivní algoritmy a jejich tasky, kde se použije tentýž buffer. Ty se poté také spustí asynchronně. Z důvodu optimalizace bylo přidáno omezení na zpracování maximálně dvou souborů/složek zároveň. I to ale občas může

způsobovat problémy, tak do budoucího rozvoje práce by bylo dobré zlepšit optimalizace, např. použít knihovnu MapDB. Po dokončení všech operací se pošle signál k aktualizaci průběhu a vygeneruje se přepravka s výsledky analýzy.

5.1.4. Balíček Output

Poslední hlavní částí modulu Core je balíček output. Ten má v sobě obsaženou logiku pro generování výstupních souborů. V současné době je podporován formát XLSX a PDF. Generováni obou formátů je řešené podobně, každý formát však obsahuje specifické nástroje pro tvorbu. Schéma pro tvorbu XLSX je vidět na obrázku 5.5. Aplikace komunikuje s generátory pomocí rozhraní FileOutput. V rámci generování XLSX(i PDF) se vnitřní logika rozpadne na několik menších generátorů, které implementují rozhraní XLSXGenerator (případně PDFGenerator). Tyto generátory se pak spustí v hlavní třídě XLSXOutput v předem daném pořadí a vznikne tak výsledný soubor.

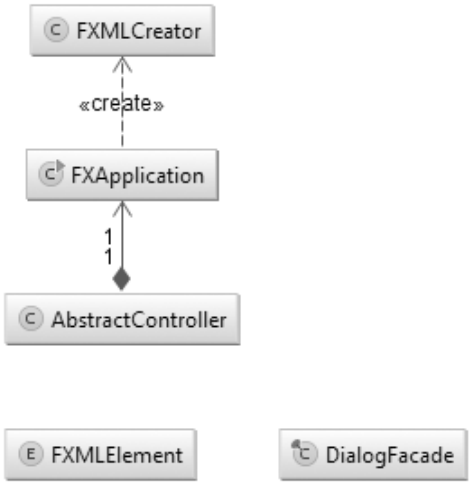


Obrázek 5.5.: Schéma generátoru XLSX

Tento balíček obsahuje, jako jediný, jednotkové testy. Je zde vytvořen mock proběhnuté analýzy a podle něj jsou generovány soubory. Testy ale nejsou spouštěny při sestavování programu, spíše jsou zde pro ulehčení a zrychlení vývoje.

5.2. Modul Fxml

Modul FXML je pomocný modul pro práci s JavaFX, který pak dále využívá modul Application a Dev. Obsahuje pouze několik společných třídy pro tyto dva moduly, schéma je vidět na obrázku 5.6. JavaFX vytváří GUI pomocí speciálního XML formátu FXML nebo je možné využít i programátorský přístup a udělat vše manuálně. Pro vytvoření nového prvku je potřeba znát název FXML souboru a také jeho kontrolor, aby se nemuselo pokaždé tyto dvě věci definovat, tak třída FXApplication obsahuje metody pro tvorbu nových prvků a přepínání kontextu. Stáčí tedy definovat nový prvek v enumu FXMLElement a zavolat patřičnou metodu v FXApplication. Ta vezme prvek z enumu a pomocí FXMLCreator vytvoří nový prvek, k němuž je pak přiřazen abstraktní kontrolor. Jediná podmínka je pak ta, že každý kontrolor musí být potomkem abstraktního předka a v aplikaci pak proběhne přetypování na konkrétní kontrolor.

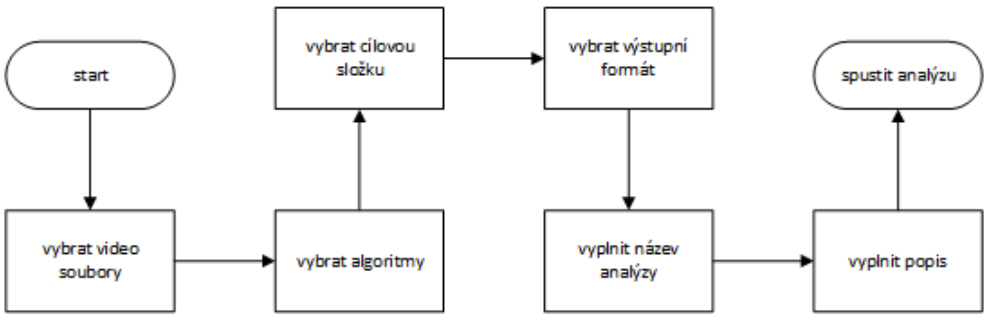


Obrázek 5.6.: Schéma modulu FXML

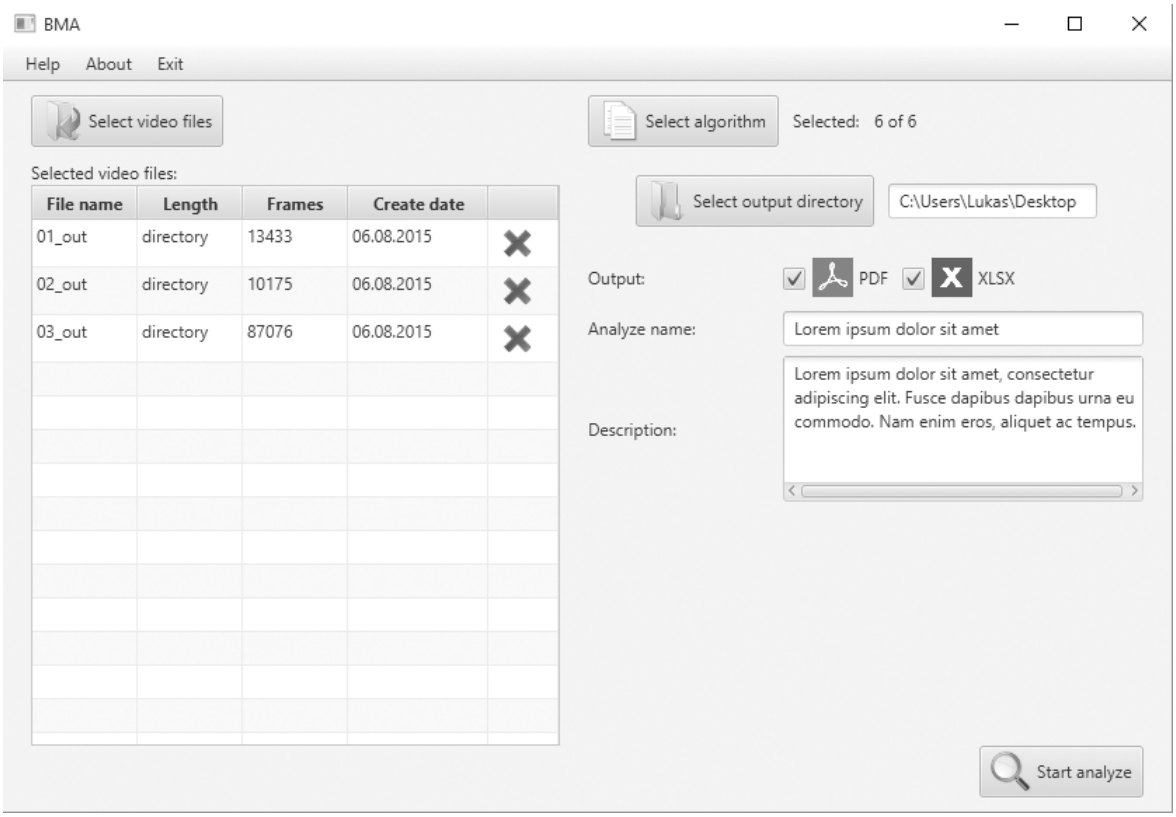
Při vytváření elementu ve třídě FXMLCreator se také nastavuje cesta k lokalizačním souborům, ta se bere z MessageUtils, který je v modulu Core. Aplikace je tímto uzpůsobena na možnost více jazyků. V této třídě je možné vytvořit i element, který bude spolupracovat s CDI. Toho bylo využito téměř po celou dobu vývoje aplikace a byla zvolena implementace WeldSE. Ale po problémech s pomalejším načítáním programu a pro zvláštní chování s použitím ProGuard a Shade pluginu pro Maven bylo od této implementace opuštěno (funkce těchto pluginů je popsána v kapitole 5.3). Tímto bohužel aplikace ztratila interceptory pro kontrolování výkonu jednotlivých metod, snazší logování programu pomocí log4j, řešení automatických závislostí a lepší práci se singletone objekty. Do dalšího možného vývoje práce je možné zkusit CDI implementovat pomocí Spring Boot, který by neměl dělat problémy, které vznikaly při použití WeldSE. A to zejména proto, že není definován pomocí XML, ale konfigurace se provádí přímo v Jave.

5.3. Modul Application

V modulu Application je definováno uživatelské rozhraní a komunikace s modulem Core. Aplikace je založena na návrhovém vzoru MVC. Jedná se pouze o několik kontrolorů a pomocných tříd, bez žádné složitější logiky, tu pak obstarává modul Core. Hlavním kontrolorem je třída MainController, jehož rozhraní je vidět na obrázku 5.8. Práce s programem je zobrazena na obrázku 5.7.

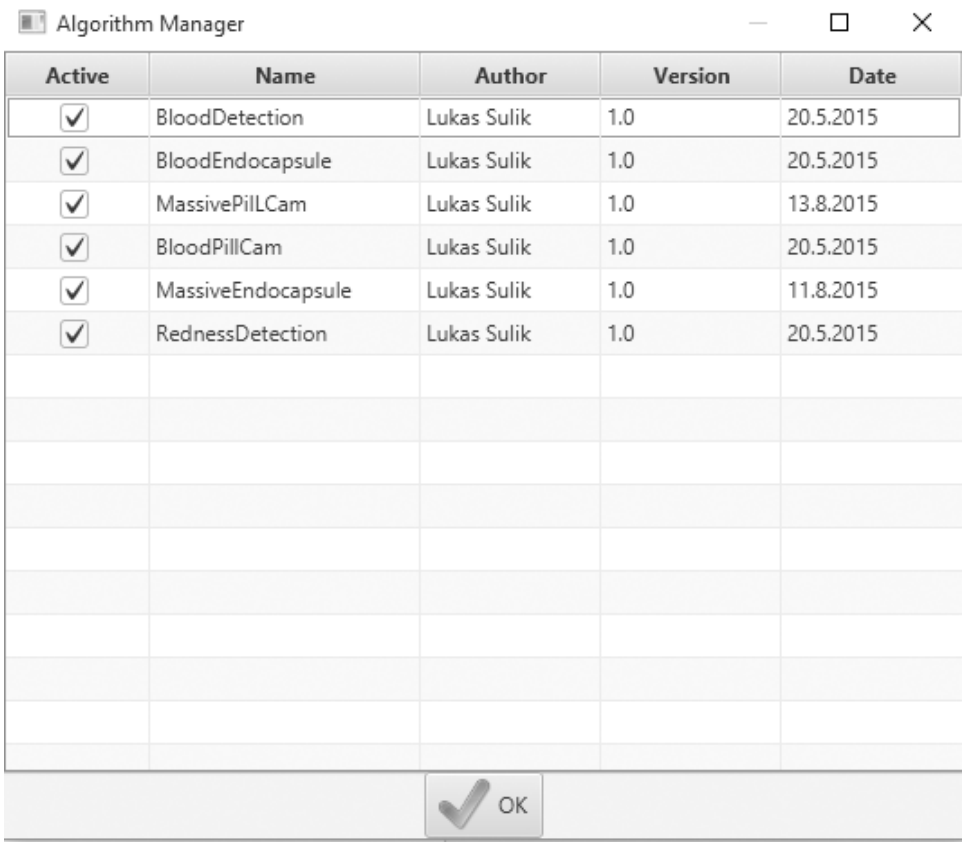


Obrázek 5.7.: Průběh práce s programem



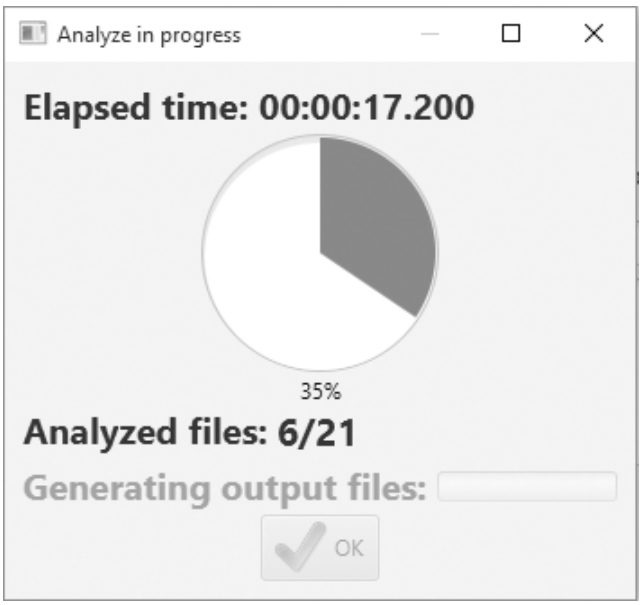
Obrázek 5.8.: Hlavní obrazovka aplikace

Z hlavní obrazovky se dá spustit AlgorithmManagerController, viz. obrázek 5.9. Kde se vyberou konkrétní algoritmy, které se mají použít pro aktuální analýzu. Dále se na hlavní obrazovce dá zvolit, které soubory se budou zpracovávat a kam se pak vygenerují výstupní soubory. Lze k analýze připsat i volitelné doplňkové údaje, ty se pak také vygenerují k výstupním souborům. Po spuštění analýzy je zobrazen dialog průběhu, který je vidět na obrázku 5.10.



Obrázek 5.9.: Správa algoritmů

ProgressDialogController implementuje rozhraní ProgressNotifier z modulu Core. Přes toto rozhraní jsou posílány signály z asynchronního zpracování souborů a na základě toho je v tomto případě aktualizováno uživatelské rozhraní. Při implementování jiného GUI je možné využít toto rozhraní jinak, např. pokud by se jednalo o webovou aplikaci, tak je možné ukládat záznam o průběhu do databáze a uživatel by se pak na něj mohl dotazovat pomocí ajaxových dotazů.



Obrázek 5.10.: Dialog průběhu

V modulu application jsou umístěné Maven skripty pro sestavování programu. Nejprve je využit Maven Shade plugin, který dle definice autoru - "Tento plugin má schopnost zabalit program do uber-jar, včetně jeho závislostí a zastínit ho - tj. přejmenovat balíčky nějakých závislostí."volně přeloženo z [25]. Plugin také odstraňuje duplicitní závislosti, to může vznikat v případě, že různé knihovny využívají další knihovny, které mohou být stejné. Dále je spuštěn ProGuard plugin, ten má definovanou konfiguraci v externím souboru a v Mavenu se pouze spustí. Z tohoto pluginu je využita pouze obfuskace kódu, ostatní možnosti jsou zakázány. Až na pár vyjímek nutných k funkčnosti je obfuskován celý kód a s ním i knihovna OpenCV, ukázka výsledku je na obrázku 5.11.

```java
public static Mat filterSmallContours(Mat originalMat, double minSize) {
    List<MatOfPoint> contours = new ArrayList<>();
    Imgproc.findContours(originalMat, contours, new Mat(), Imgproc.RETR_LIST,
            Imgproc.CHAIN_APPROX_SIMPLE);
    Mat contourFilter = Mat.zeros(originalMat.size(), originalMat.type());
    double contourArea;
    for (int i = 0; i < contours.size(); i++) {
        MatOfPoint contour = contours.get(i);
        contourArea = Imgproc.contourArea(contour);
        if (contourArea > minSize) {
            Imgproc.drawContours(contourFilter, contours, i, new Scalar(255), -1);
        }
        contour.release();
    }
    return contourFilter;
}
```

```java
public static Mat b(Mat paramMat, double paramDouble)
{
  ArrayList localArrayList = new ArrayList();
  Imgproc.a(paramMat, localArrayList, new Mat(), 1, 2);
  Mat localMat = Mat.d(paramMat.p(), paramMat.t());
  for (int i = 0; i < localArrayList.size(); i++)
  {
    aJ localaJ = (aJ)localArrayList.get(i);
    double d = Imgproc.b(localaJ);
    if (d > paramDouble) {
      Imgproc.a(localMat, localArrayList, i, new aT(255.0D), -1);
    }
    localaJ.n();
  }
  return localMat;
}
```

Obrázek 5.11.: Horní část ukázky je zdrojový kód. Dolní je dekompilovaný obfuskovaný kód.

Jakmile je připravena JAR knihovna, spustí se plugin Launch4J, který vytvoří spouštěcí soubor pro platformu Windows. Je možné spouštět aplikaci i přes JAR, jelikož je odděleno od exe souboru, ale takto je to pohodlnější pro uživatele Windows. V exe souboru je také zakomponována kontrola minimální verze Javy.

5.4. Modul Dev

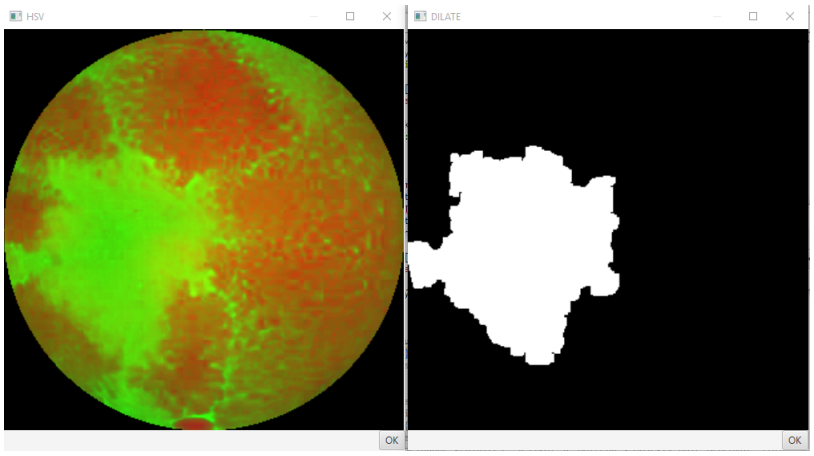
Modul Dev slouží pouze k vývoji algoritmů a není zahrnut do sestavení aplikace pro uživatele. Vznikl proto, že autor nemohl nalézt žádné vývojové nástroje pro Javu, které by byly kompatibilní s OpenCV a byly vhodné pro účel práce. Obsahuje pouze prosté GUI, které se z větší části generuje pomocí reflexe. Aby bylo možné pracovat s algoritmy v reálném čase

a sledovat tak přímo, co který algoritmus s určitou konfigurací na konkrétním snímku dělá bylo nutné zasáhnout do logiky programu. Je snaha tento zásah řešit pouze v tomto modulu, ale některé nutné úpravy byly udělány i ve třídě Buffer (neomezená kapacita a nemažou se data) a VideoFrame(metody pro ukládání snímku algoritmu), fungují však pouze pokud je globálně zapnutý vývojový mód, a tak není ovlivněn běžný chod programu.

V rozhraní je stejně tak jako v aplikaci spuštěn MainController, který obsahuje jen volbu algoritmu a souboru, který má být zpracován, lze zpracovávat pouze jeden soubor a algoritmus najednou. Po spuštění programu je vystavěno vývojové rozhraní, které je vidět na obrázku 5.14. Pomocí reflexe jsou získána všechna pole z třídy a ta, která obsahují anotaci @DevInterfaceField (ukázka na obrázku 5.3), se zpracují a vytvoří se z nich ovládací prvek na základě hodnoty v atributu fieldType. Ovládací prvky mohou být: zaškrtávací políčko (pro booelan), posuvník (int, double, ...), obyčejné pole (string, int, ...) a výčtový typ. Výčtový typ je použit pro ukládání snímku algoritmu v daném kroku např. obrázek 5.12.

```
Mat hsv = new Mat();
Imgproc.cvtColor(croppedSrc, hsv, Imgproc.COLOR_BGR2HSV);
frame.devCopy(BBConf.devNames.HSV.name(), hsv);
```

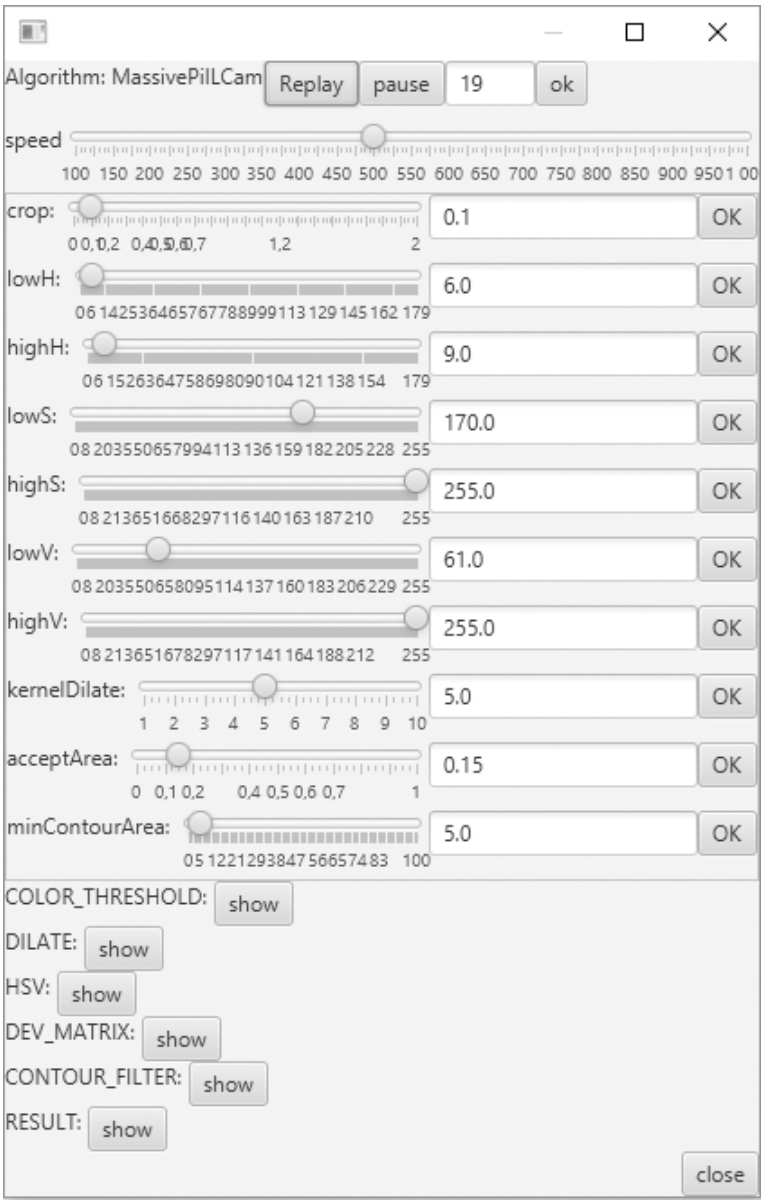
Obrázek 5.12.: Ukázka konverze z BGR do HSV a následné uložení snímku pro vývoj.



Obrázek 5.13.: Ukázka snímků algoritmu. Vlevo převod do HSV, vpravo dilatace.

Pro běh programu je spuštěn AnalyzeTask z modulu Core pro získání a zpracování dat. K tomu je spuštěno další vlákno, kde běží BufferPlayer, který je potomkem

AbstractBufferTask z modulu Core. Ten zpracovává data z Bufferu s aktuální konfigurací a také zobrazuje vybraná data snímků algoritmu v rozhraní, na obrázku 5.13 jsou vidět snímky z algoritmu. BufferPlayer také ukládá všechny detekované artefakty a k nim i snímky na disk pro rychlejší orientaci, co je detekováno. Jak je vidět na obrázku 5.14, tak rozhraní disponuje ještě pauzou, posuvníkem rychlosti, opakováním a přechodem na konkrétní snímek v bufferu.

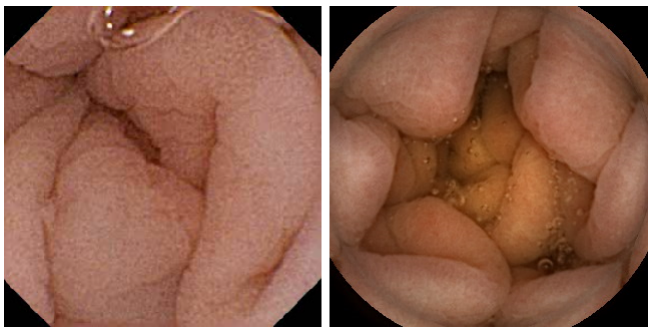


Obrázek 5.14.: Vývojové rozhraní konkrétního algoritmu.

6. Testování vyvinutého řešení

Testování a vývoj algoritmu byl prováděn především empiricky. K dispozici byl nejdříve malý vzorek dat, který byl dodán Fakultní nemocnicí Hradec Králové. Tento vzorek obsahoval 21 patologických videí, kde se vyskytovaly různé nemoci a 16 videí bez patologických výskytů. Každý snímek obsahoval 201 framů. Jelikož tato data byla extrahována za pomocí proprietárního softwaru dodávaného ke kamerám, tak kvalita dat nebyla nijak valná. U toho vzorku bylo lékaři označeno i, to co je na snímkách špatně.

Druhý vzorek dat byl získán od nemocnice až po vyvinutí algoritmů a softwaru. Měl tedy za úkol prokázat vyvinuté řešení na větším vzorku dat. U toho vzorku se však postrádá informace od profesionálů, jaká část obsahuje patologické nálezy a kvůli velkému množství dat to lze určit jen přibližně. Celkem se jedná o tři celé průchody kamerou, kde je patologický nález. Dva průchodu jsou z kamery PillCam celkem o 23608 framech a zbývající je z kamery od výrobce EndoCapsule o 87076 framech. Tento soubor dat je výrazně kvalitnější, jelikož ve spolupráci s akademií věd České republiky bylo použito jejich technologií pro extrakci dat z kamer. Na obrázku 6.1 je vidět rozdíl v kvalitě obou vzorků.



Obrázek 6.1.: Ukázka rozdílů kvality obou vzorků. Vlevo první vzorek, vpravo druhý vzorek

Vývoj a testování algoritmu detekce malých skvrn probíhal v různých etapách a od

původního návrhu algoritmu do dnešní podoby se změnilo hodně věcí a byly použity jiné přístupy. Původní návrh byl nejdříve založen na adaptivním binárním práhování, ale to bylo zcela neúspěšné. Později byla snaha detekovat krvavé skvrny pomocí hranových detektorů konkrétně Cannyho detektor. To vedlo k lepšímu výsledku, nicméně chybovost byla příliš veliká a hrany se občas ztráceli vůči pozadí. Nakonec bylo zvoleno filtrování barev v HSV spektru, kde bylo dosaženo nejlepších výsledků a v kombinaci s hranovým detektorem pro odstranění míst bez hran se jeví jako optimální řešení. Testování později odhalilo, zejména v druhém vzorku dat, že druhý filtr barev pro zpřesnění výsledků není tak důležitý a jeho dopad v konečném výsledku není příliš markantní.

Druhý algoritmus byl značně přímočařejší a vzhledem k tomu, že autor již měl zkušenosti z prvního algoritmu, tak se spíše jednalo o nastavování správných hodnot v konfiguraci.

6.1. Metodologie testování

Pro měření úspěšnosti algoritmů jsou použity standardní postupy v počítačovém vidění pro určení efektivity algoritmu. Je tedy vypočítána hodnota FRR a FAR dle následujících vzorců.

$$FRR = \frac{FN}{TP + TN + FP + FN}$$

$$FAR = \frac{FP}{TP + TN + FP + FN}$$

V klinických testech se v lékařské komunitě používají hodnoty Senzitivita (SE) a Specificita (SP)[26]. Senzitivita je procentuální šance na detekci pacienta, který je nemocný a není tak špatně označen jako zdravý. Dalo by se říct, že opakem senzitivity je specificita. Při specificitě se jedná o správné detekování pacienta, který není nemocný a není špatně označen, jako pozitivní. Určuje se dle následujících vzorců.

$$SE = \frac{TP}{TP + FN}$$

$$SP = \frac{TN}{TN + FP}$$

V tabulce 6.1 je vidět, kolik je skutečně pozitivních snímků v každém vzorku. Do tabulky jsou zahrnuty pouze ty vzorky, které obsahují krev. Ostatní vzorky buď obsahují jiné nemoci, nebo nejsou pozitivní. Každý vzorek také vždy neobsahuje krev, jež se dá snadno detekovat. Např. na vzorku z endocapsule je po cca 50% vzorku masivní krvácení, kde se pak na každém snímku od vzniku problému vyskytuje alespoň slabá, místy růžová krev, která je již smíchána s jinými tekutinami, a tak nevyznačuje známky krve definované v algoritmu (dalo by se říct, že často je to spíše růžová voda). Počet snímků s krví u všech případů není úplně přesný, protože autor nemá k dispozici přesné počty, a tak musel vzorky projít ručně a provést odhad, který může být lehce nepřesný.

Název souboru	Počet snímků	Počet snímků s krví
Endocapsule	87076	45320
PillCam 1	13433	32
PillCam 2	10175	15
PAT07.avi	201	5

Tabulka 6.1.: Jednotlivé vzorky s počtem krvavých snímků

6.2. Výsledky testování

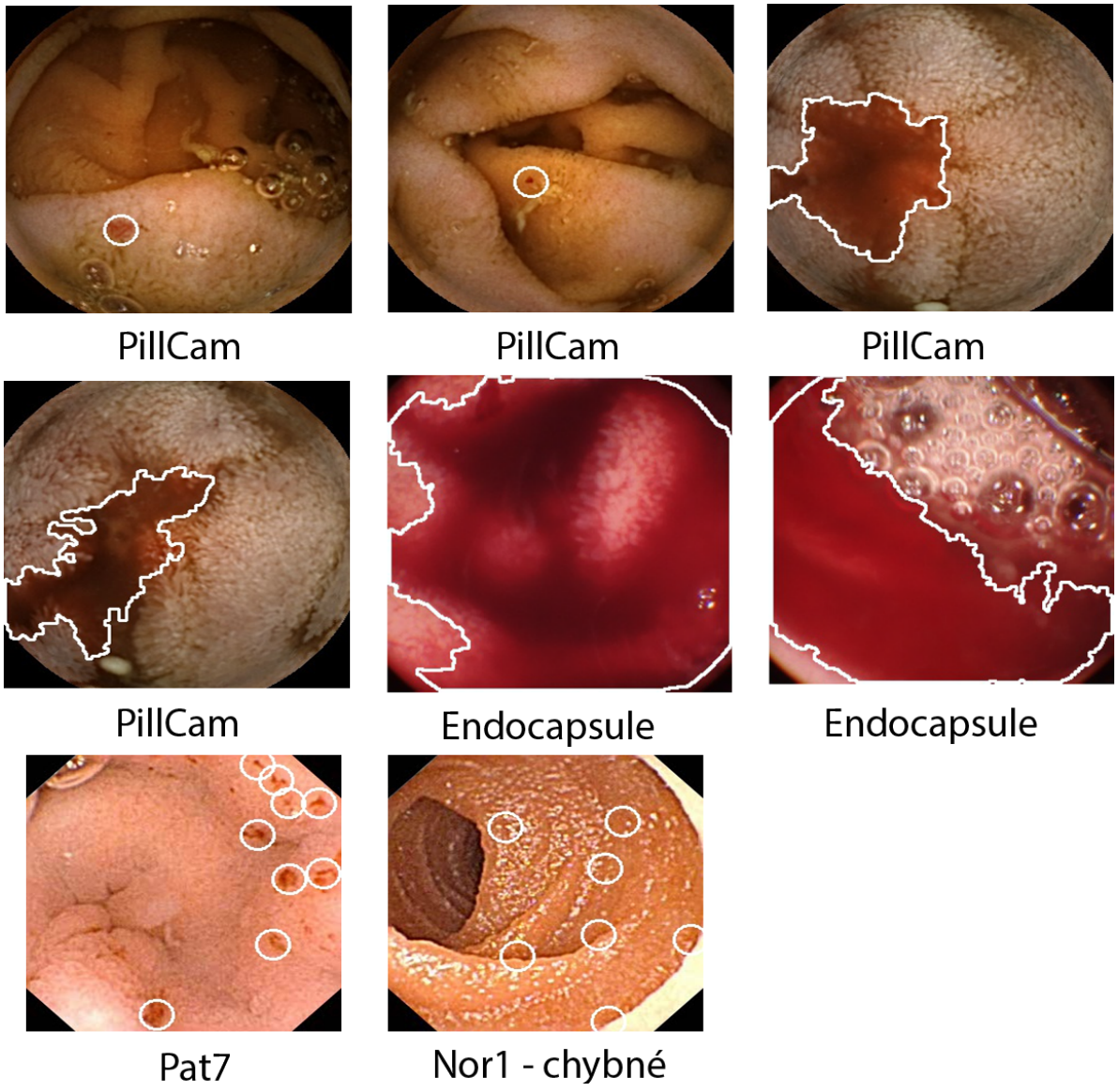
Následující tabulka 6.2 a 6.3 ukazuje výsledky testování. Byly testovány všechny vzorky, vždy dle konfigurace pro ně určené. Pro jaký vzorek byla která konfigurace použita, je vidět v příloze A. Ve výsledkových tabulkách jsou brány oba algoritmy jako jeden, a to z toho důvodu, že pro určení konečného výsledku je irelevantní, jestli to detekoval algoritmus A nebo B. Proto jsou vzaty výsledky obou algoritmů a je provedeno sjednocení. Také by bylo velmi problematické určit, co který algoritmus má detekovat a podle toho ho také hodnotit. Vybrané konkrétní výsledky algoritmů jsou vidět na obrázku 6.2.

Název souboru	Detekováno	TP	FP	TN	FN	FRR	FAR	SE	SP
PAT21.avi	0	0	0	201	0	0,00%	0,00%	N/A	100,00%
PAT20.avi	0	0	0	201	0	0,00%	0,00%	N/A	100,00%
PAT19.avi	0	0	0	201	0	0,00%	0,00%	N/A	100,00%
PAT18.avi	0	0	0	201	0	0,00%	0,00%	N/A	100,00%
PAT17.avi	1	0	1	200	0	0,00%	0,50%	N/A	99,50%
PAT16.avi	5	0	5	196	0	0,00%	2,49%	N/A	97,51%
PAT15.avi	1	0	1	200	0	0,00%	0,50%	N/A	99,50%
PAT14.avi	1	0	1	200	0	0,00%	0,50%	N/A	99,50%
PAT13.avi	0	0	0	201	0	0,00%	0,00%	N/A	100,00%
PAT12.avi	0	0	0	201	0	0,00%	0,00%	N/A	100,00%
PAT11.avi	0	0	0	201	0	0,00%	0,00%	N/A	100,00%
PAT10.avi	0	0	0	201	0	0,00%	0,00%	N/A	100,00%
PAT09.avi	0	0	0	201	0	0,00%	0,00%	N/A	100,00%
PAT08.avi	0	0	0	201	0	0,00%	0,00%	N/A	100,00%
PAT07.avi	4	4	0	196	1	0,50%	0,00%	80,00%	100,00%
PAT06.avi	0	0	0	201	0	0,00%	0,00%	N/A	100,00%
PAT05.avi	0	0	0	201	0	0,00%	0,00%	N/A	100,00%
PAT04.avi	0	0	0	201	0	0,00%	0,00%	N/A	100,00%
PAT03.avi	0	0	0	201	0	0,00%	0,00%	N/A	100,00%
PAT02.avi	3	0	3	198	0	0,00%	1,49%	N/A	98,51%
PAT01.avi	0	0	0	201	0	0,00%	0,00%	N/A	100,00%
NOR9.avi	0	0	0	201	0	0,00%	0,00%	N/A	100,00%
NOR8.avi	0	0	0	201	0	0,00%	0,00%	N/A	100,00%
NOR7.avi	2	0	2	199	0	0,00%	1,00%	N/A	99,00%
NOR6.avi	0	0	0	201	0	0,00%	0,00%	N/A	100,00%
NOR5.avi	0	0	0	201	0	0,00%	0,00%	N/A	100,00%
NOR4.avi	0	0	0	201	0	0,00%	0,00%	N/A	100,00%
NOR3.avi	0	0	0	201	0	0,00%	0,00%	N/A	100,00%
NOR2.avi	0	0	0	201	0	0,00%	0,00%	N/A	100,00%
NOR16.avi	0	0	0	201	0	0,00%	0,00%	N/A	100,00%
NOR15.avi	0	0	0	201	0	0,00%	0,00%	N/A	100,00%
NOR14.avi	0	0	0	201	0	0,00%	0,00%	N/A	100,00%
NOR13.avi	0	0	0	201	0	0,00%	0,00%	N/A	100,00%
NOR12.avi	0	0	0	201	0	0,00%	0,00%	N/A	100,00%
NOR11.avi	0	0	0	201	0	0,00%	0,00%	N/A	100,00%
NOR10.avi	0	0	0	201	0	0,00%	0,00%	N/A	100,00%
NOR1.avi	47	0	47	154	0	0,00%	23,38%	N/A	76,62%

Tabulka 6.2.: Výsledky testování prvního vzorku

Název souboru	Detekováno	TP	FP	TN	FN	FRR	FAR	SE	SP
Endocapsule	37889	35546	2343	39413	9774	11,22%	2,69%	78,43%	94,39%
PillCam 1	159	24	135	13266	8	0,06%	1,00%	75,00%	98,99%
PillCam 2	79	12	67	10093	3	0,03%	0,66%	80,00%	99,34%

Tabulka 6.3.: Výsledky testování druhého vzorku



Obrázek 6.2.: Ukázka vybraných výsledků

6.3. Zhodnocení testování

Testování algoritmů probíhalo ve dvou etapách. V první etapě se algoritmy vyvíjeli a testovali na prvním vzorku dat. Kde se dle tabulky 6.2 dosáhlo uspokojivých výsledků.

V druhé etapě byl obdržen druhý vzorek dat z reálných pacientů. Po aplikaci konfigurace z prvního vzorku a spuštění testů nad druhým došlo k téměř absolutnímu neúspěchu. Aplikace nebyla schopna detekovat ani výrazné krvácení. Bylo zapotřebí vytvořit konfiguraci, zejména na filtr dle barev, pro každý vzorek zvlášť. Stejné chování je zaznamenáno i mezi odlišnými výrobci kamer v druhém vzorku, každá kamera má odlišné barevné spektrum, které dokáže snímat a tak není možné použít univerzální řešení pro všechno. Výsledky testování druhého vzorku jsou vidět v tabulce 6.3.

Při testování jinak nedošlo ke změnám algoritmů či programu. Vše fungovalo, tak jak bylo očekáváno. Pouze se vytvořili nové konfigurace pro algoritmy. Porovnání nové metody o proti metodám zmíněných ve vědeckých článcích nebylo možné, protože není k dispozici stejný datový vzorek. V následující kapitole budou detailněji rozebrány a zhodnoceny samotné výsledky.

7. Diskuze nad výsledky

Z tabulky 7.1 je vidět, že průměrná senzitivita u pozitivních videí je 78.36%. To se může zdát jako málo, ale vzhledem k tomu, že se krev nachází ve střevech v různé podobě a tvarech, tak nelze vždy určit krev všechnu, jelikož je to mnohdy velmi obtížné. Co je důležité, že ve všech případech byl určen zdroj krvácení, případně jeho největší části. Ostatní nedetekované objekty pak mohou být např. smíchaná krev s tekutinami nebo její velmi malé částečky po okrajích snímku mnohdy již dříve detekované anomálie na snímcích předchozích. Pro relevantnost testů je však nutné tyto objekty zahrnout do FN, jelikož odborníci jsou schopni je určit za pomocí jejich predikce a zkušeností. Také pro vyšetření pacienta je podstatné vědět, kde se nachází zdroj krvácení nebo jeho velká část.

Název	FRR	FAR	SE	SP
Pozitivní videa	2,95%	1,09%	78,36%	98,18%
Negativní videa	0,00%	0,83%	N/A	99,17%

Tabulka 7.1.: Průměrné výsledky

Specificita a FAR u všech videí je, až na jednu výjimku, velice dobrá a je tak možné rychle určit, zda-li pacient je zdráv či nikoliv. Výjimkou je video NOR1.avi, kde bylo rozeznáno 23.28% mylně pozitivních snímků viz. tabulka 6.2. To je způsobeno tím, že se ve videu nachází hodně odlesků, které pak vytvářejí malé anomálie.

Hodnota FRR se drží v přijatelných mezích a nejsou příliš odmítány i snímky, které obsahují krev. Výjimkou je vzorek Endocapsule a to z důvodu, že se zde nachází masivní krvácení, a jak již bylo zmíněno výše, tak některé snímky již nelze přesněji určit a je podstatné, že se našel zdroj krvácení. Pravděpodobně by nevznikly z něčeho, co se nepodařilo určit, a test by tak závisel na tom, jestli se poznají vedlejší efekty krvácení.

Ohledně výsledků toho, jak si na tom stojí software, tak během testování byly zjištěny technické nedostatky a to zejména v ukládání dočasných výsledků algoritmu pro pozdější generování výstupních souborů. Při testování vzorku z Endocapsule, kde bylo detekováno 37889 objektů, tak byly nároky na paměť enormní a musel se testovaný vzorek rozdělit na více částí. Také občas působila problémy knihovna OpenCV. Jelikož se jedná o nativní přístup, tak zde byl problém s garbage collectorem, který neodstraňuje nativní objekty, a proto se muselo dávat pozor, aby se všechny vytvořené nativní objekty korektně zrušily za pomocí k tomu určených funkcí v OpenCV. Jinak výkonnost a optimalizace algoritmů je uspokojivá. Například oba vzorky z PillCam trvalo zpracovat dvě minuty a šestnáct vteřin na průměrném PC.

Naplnění funkčních a nefunkčních požadavků na software se podařilo naplnit s dvěma nedostatky. Z výsledků testování vyplynula nedostatečná optimalizaci při držení detekovaných hodnot a tím velké nároky na paměť. A software umí načítat nativní OpenCV knihovny pouze pro MS Windows. Pro ostatní OS, které OpenCV podporuje není hotové načítání knihoven z důvodu absence těchto prostředí při vývoji.

7.1. Možný směr budoucího vývoje

Budoucí vývoj by se mohl ubírat opravnou technických částí softwaru čí jeho vylepšením, na které už nebyl vzhledem časové náročnosti čas. V bodech by se jednalo hlavně o tyto záležitosti:

- Implementace MapDB pro vyřešení paměťové náročnosti při velkých datech nebo zaměnit způsob ukládání výsledků algoritmu, např. pouze jako číslo snímku, nikoliv celý snímek, a zpětně je pak dohledávat.
- Dodělat načítání nativních OpenCV knihoven pro ostatní OS.
- Implementace Spring Boot a nahrazení frameworku Weld, který musel být odstraněn.
- Větší škálování vláken pro algoritmy a čtení více dat z bufferu jedním algoritmem. Momentálně se snímky z bufferu zpracovávají jeden po jednom.

- Neanalyzovat již pozitivní snímek určený jiným algoritmem.
- Nastavení neměnných konstant algoritmu při prvním běhu a nepočítat je vždy znova pro každý snímek. Toto platí pouze za předpokladu, že všechny snímky budou ve vzorku stejné.
- Rozložit některé části algoritmu na více vláken a zpracovávat asynchronně části na sobě nezávislé.
- Implementovat webové rozhraní. Uživatel by přistupoval k programu pouze na bázi tenkého klienta a všechny operace by se děly na straně serveru. To by mohlo přinést rychlejší zpracování a případně prohlížení výsledků i na mobilních zařízení/tabletech.

Dále by bylo nutné otestovat software na ještě větším množství dat a případně podle toho doladit algoritmy. Co se týče algoritmů, tak z detekce malých skvrn by šla odstranit část pro zpřesnění výsledků, jelikož se ukázalo, že není tak důležitá. Nejdůležitějším bodem budoucího vývoje zůstává rozšíření programu o další algoritmy na jiné nemoci a vyzkoušení dalších technik, např. neuronové sítě pro detekci nových věcí.

8. Závěr

Cílem práce bylo analyzovat možnosti počítačového vidění v biomedicíně. Byla představena kapslová endoskopie, její využití a možnosti. Dále se práce zabývala představením počítačového vidění a jeho základními principy. Průzkum aktuálního stavu trhu sumarizoval již hotové řešení od výrobců kapslí, to bohužel vzhledem k uzavřenosti programů nepřineslo mnoho výsledků.

Praktická část se zabývala vývojem nového softwaru, který by byl alespoň částečně konkurence schopný a uměl detekovat krev, případně v budoucnu i další choroby. Vzorky pro testování byly dodány Fakultní nemocnicí Hradec Králové.

Samotný vývoj software se ukázal jako časově náročný, ale obešel se bez výraznějších problémů, jelikož zvolené technologie byly autorovi dobře známé. S algoritmy byla daleko větší práce. Autor měl sice předchozí zkušenosti s počítačovým viděním, ale nebyly nikterak četné. Neocenitelnou pomoc pro vývoj algoritmů skýtaly konzultace s Dr. Orcanem Alparem.

Testování ukázalo, že software je schopný detekovat krev ve střevech s vysokou přesností. Aby byly výsledky průraznější, bylo by vhodné v tomto projektu pokračovat i mimo tuto diplomovou práci a provést další, rozsáhlejší testy. Zatím je vývoj projektu na dobré cestě a myslím si, že výsledky jsou uspokojivé.

Literatura

[1] M. Appleyard, Z. Fireman, A. Glukhovsky, H. Jacob, R. Shreiver, S. Kadirkamanathan, A. Lavy, S. Lewkowicz, E. Scapa, R. Shofti, P. Swain a A. Zaretsky, "A randomized trial comparing wireless capsule endoscopy with push enteroscopy for the detection of small-bowel lesions", *Gastroenterology*, sv. 119, č. 6, s. 1431–1438, DOI: 10.1053/gast.2000.20844. WWW: http://www.gastrojournal.org/article/S0016-5085(00)49998-1/abstract (cit. 20. 01. 2015).

[2] *ENDOCAPSULE (MAJ-2027) | Olympus America | Medical*. WWW: http://medical.olympusamerica.com/products/endocapsule-maj-2027 (cit. 24. 06. 2015).

[3] *Intromedic*. WWW: http://www.intromedic.com/eng/sub_products_1.html (cit. 24. 06. 2015).

[4] G. I. Ltd., *PillCam® Capsule Endoscopy, User Manual, Rapid® v8.0*, 2013.

[5] *Capsule endoscopy camera - Mayo Clinic*. WWW: http://www.mayoclinic.org/tests-procedures/capsule-endoscopy/basics/definition/prc-20012773 (cit. 24. 06. 2015).

[6] Z Fireman, E Mahajna, E Broide, M Shapiro, L Fich, A Sternberg, Y Kopelman a E Scapa, "Diagnosing small bowel Crohn's disease with wireless capsule endoscopy", *Gut*, sv. 52, č. 3, s. 390–392, břez. 2003. DOI: 10.1136/gut.52.3.390. WWW: http://gut.bmj.com/content/52/3/390.abstract.

[7] G. R. Bradski a A. Kaehler, *Learning OpenCV: [computer vision with the OpenCV library]*, eng, 1. ed., [Nachdr.], ř. Software that sees. Beijing: O'Reilly, 2011, ISBN: 9780596516130 9780596516130.

[8] M. Sonka, *Image processing, analysis, and machine vision*, International student ed. Mason, OH: Thomson, 2007, ISBN: 0495244384.

[9] *OpenCV 3 Image Thresholding and Segmentation - 2015*. WWW: `http://www.bogotobogo.com/python/OpenCV_Python/python_opencv3_Image_Global_Thresholding_Adaptive_Thresholding_Otsus_Binarization_Segmentations.php` (cit. 07. 08. 2015).

[10] *Eroding and Dilating — OpenCV 2.4.9.0 documentation*. WWW: `http://docs.opencv.org/doc/tutorials/imgproc/erosion_dilatation/erosion_dilatation.html` (cit. 13. 01. 2015).

[11] *Opencv: Smoothing Images*. WWW: `http://docs.opencv.org/master/d4/d13/tutorial_py_filtering.html#gsc.tab=0` (cit. 10. 09. 2015).

[12] *RapidSoftware v8 Image*. WWW: `http://i.ytimg.com/vi/QHHuHvpYn6I/maxresdefault.jpg` (cit. 02. 08. 2015).

[13] *Olympuseuropa*. WWW: `http://www.olympus-europa.com/medical/en/medical_systems/applications/gastroenterology_1/capsule_endoscopy_and_enterology/capsule_endoscopy_and_enterology.html` (cit. 02. 08. 2015).

[14] *MiroView Image*. WWW: `http://i.ytimg.com/vi/J3uHafbQ6vk/maxresdefault.jpg` (cit. 02. 08. 2015).

[15] *About Mayo Clinic - About Us - Mayo Clinic*. WWW: `http://www.mayoclinic.org/about-mayo-clinic` (cit. 02. 08. 2015).

[16] *Gisentinel*. WWW: `http://www.xyken.com/Pages/MedicalSolutions.aspx` (cit. 02. 08. 2015).

[17] *Gumption: Recommender Systems*. WWW: `http://gumption.typepad.com/blog/recommender_systems/` (cit. 13. 01. 2015).

[18] A.-R. Amer a A. Abdelshakour, "Detection of bleeding in wireless capsule endoskopy images using range ratio color", *2*, sv. May 2010, č. 2, WWW: `http://arxiv.org/ftp/arxiv/papers/1005/1005.5439.pdf` (cit. 13. 01. 2015).

[19] P. Barbara, T. Tammam, G. Marco, M. Enrico a O. Gabriella, "A technique for blood detection in wireless capsule endoscopy images", sv. 2009, WWW: `http://www.eurasip.org/Proceedings/Eusipco/Eusipco2009/contents/papers/1569190828.pdf` (cit. 13. 01. 2015).

[20] *Maven – Welcome to Apache Maven*. WWW: `https://maven.apache.org/` (cit. 28. 08. 2015).

[21] *Apache POI - the Java API for Microsoft Documents*. WWW: `https://poi.apache.org/` (cit. 28. 08. 2015).

[22] *Apache PDFBox | A Java PDF Library*. WWW: `https://pdfbox.apache.org/` (cit. 28. 08. 2015).

[23] *Proguard*. WWW: `http://proguard.sourceforge.net/` (cit. 28. 08. 2015).

[24] *Launch4j - Cross-platform Java executable wrapper*. WWW: `http://launch4j.sourceforge.net/` (cit. 28. 08. 2015).

[25] *Apache Maven Shade Plugin – Introduction*. WWW: `https://maven.apache.org/plugins/maven-shade-plugin/` (cit. 06. 09. 2015).

[26] A. G. Lalkhen a A. McCluskey, “Clinical tests: Sensitivity and specificity”, *Continuing Education in Anaesthesia, Critical Care & Pain*, sv. 8, č. 6, s. 221–223, pros. 2008. DOI: 10.1093/bjaceaccp/mkn041. WWW: http://ceaccp.oxfordjournals.org/content/8/6/221.short.

Slovník

AVI Multimediální video formát.

CDI Contexts and Dependency Injection - standart pro vkládání závislostí a kontextu. Poprvé byl uveřejněn pro Javu EE 6.

CV Computer vision - počítačové vidění.

EER Equal error rate - přijatelná míra chyb.

FAR False acceptance rate - míra špatně detekovaných objektů.

FN False negativity - falešně nepozitivní anomálie.

FP False positivity - falešně pozitivní anomálie.

FRR False rejection rate - míra chybně odmítnutých objektů.

GUI Graphical user interface - uživatelské rozhraní.

JNA Java native access - funkcionalita pro volání nativních knihoven z Javy.

JPG Metoda komprese obrázku.

MVC Model-View-Controller - návrhový vzor.

ROI Rectangle of interest - čtverec/oblast zájmu.

SE Sensitivity - pravděpodobnost detekce.

SP Specificity - přesnost detekce.

TN True negativity - nepozitivní anomálie.

TP True positivity - pozitivní anomálie.

WCE Wireless capsule endoskopy - bezdrátová kapslová endoskopie.

Přílohy

Seznam obrázků

Seznam tabulek

A. Konfigurace algoritmů

V této příloze je konfigurace algoritmů a její vysvětlení.

Konfigurace	Význam	Málé skvrny	Velké skvrny
highH	maximální hranice filtru pro H kanál ve formátu HSV	Y	Y
lowH	minimální hranice filtru pro H kanál ve formátu HSV	Y	Y
highS	maximální hranice filtru pro S kanál ve formátu HSV	Y	Y
lowS	minimální hranice filtru pro S kanál ve formátu HSV	Y	Y
highV	maximální hranice filtru pro V kanál ve formátu HSV	Y	Y
lowV	minimální hranice filtru pro V kanál ve formátu HSV	Y	Y
kernelDilate	velikost jádra pro dilataci	Y	Y
minContourArea	minimální velikost spojité oblasti	Y	Y
highHPost	maximální hranice zpřesňujícího filtru pro H kanál ve formátu HSV	Y	N
lowHPost	minimální hranice filtru zpřesňujícího filtru pro H kanál ve formátu HSV	Y	N
highSPost	maximální hranice zpřesňujícího filtru pro S kanál ve formátu HSV	Y	N
lowSPost	minimální hranice filtru zpřesňujícího pro S kanál ve formátu HSV	Y	N
highVPost	maximální hranice zpřesňujícího filtru pro V kanál ve formátu HSV	Y	N
lowVPost	minimální hranice filtru zpřesňujícího filtru pro V kanál ve formátu HSV	Y	N
blurKernelSize	velikost jádra pro rozmazání	Y	N
cannyThreshMin	minimálníh hranice práhování pro canny	Y	N
cannyThreshMax	maximální hranice práhování pro canny	Y	N
apertureSize	velikost sobelova operátoru	Y	N
maxCircleRadius	maximální velikost kružnice	Y	N
crop	hodnota oříznutí	Y	Y
acceptArea	minimální procentuální velikost	N	Y

Tabulka A.1.: Konfigurace algoritmů

B. Konkrétní konfigurace algoritmů

V této příloze jsou konkrétní hodnoty konfigurace algoritmů, se kterou byli testovány data. Konfigurace jsou pouze jako textový výstup, jelikož konfigurační XML soubory jsou příliš velké.

Detekce malých skvrn - první vzorek

apertureSize-3; cannyThreshMax-175; lowSPost-195; blurKernelSize-3; lowH-8; lowVPost-200; lowHPost-8; highSPost-255; highH-10; lowS-190; lowV-167; highVPost-210; minContourArea-5; cannyThreshMin-140; highHPost-9; maxCircleRadius-15; kernelDilate-10; highS-255; highV-220; crop-0.1; —

Detekce malých skvrn - Endocapsule

apertureSize-3; cannyThreshMax-145; lowSPost-135; blurKernelSize-3; lowH-165; lowVPost-75; lowHPost-165; highSPost-255; highH-179; lowS-135; lowV-75; highVPost-191; minContourArea-5; cannyThreshMin-130; highHPost-179; maxCircleRadius-15; kernelDilate-10; highS-255; highV-191; crop-0.1; —

Detekce malých skvrn - PillCam

apertureSize-3; cannyThreshMax-145; lowSPost-190; blurKernelSize-3; lowH-8; lowVPost-160; lowHPost-8; highSPost-255; highH-11; lowS-190; lowV-150; highVPost-220; minContourArea-5; cannyThreshMin-130; highHPost-11; maxCircleRadius-15; kernelDilate-10; highS-255; highV-220; crop-0.1; —

Detekce velkých skvrn - první vzorek

lowV-167; acceptArea-0.15; minContourArea-5; lowH-8; kernelDilate-5; highS-255; highH-10; lowS-190; crop-0.1; highV-220; —

Detekce velkých skvrn - Endocapsule

lowV-75; acceptArea-0.15; minContourArea-5; lowH-165; kernelDilate-5; highS-255; highH-179; lowS-135; crop-0.1; highV-191; —

Detekce velkých skvrn - PilLCam

lowV-61; acceptArea-0.15; minContourArea-5; lowH-6; kernelDilate-5; highS-255; highH-9; lowS-170; crop-0.1; highV-255;

Univerzita Hradec Králové
Fakulta informatiky a managementu
Akademický rok: 2015/2016

Studijní program: Aplikovaná informatika
Forma: Prezenční
Obor/komb.: Aplikovaná informatika (ai2-p)

Podklad pro zadání DIPLOMOVÉ práce studenta

PŘEDKLÁDÁ:	ADRESA	OSOBNÍ ČÍSLO
Bc. Sulík Lukáš	Borovičky 3164, Dvůr Králové nad Labem	I1300499

TÉMA ČESKY:

Návrh a realizace software pro analýzu biomedicínských dat

TÉMA ANGLICKY:

Design and implementation of software for biomedical data analysis

VEDOUCÍ PRÁCE:

doc. Ing. Ondřej Krejcar, Ph.D. - CZAV

ZÁSADY PRO VYPRACOVÁNÍ:

Cíl práce: Analýza potencionálu počítačového vidění v oblasti kapsulární endoskopii za účelem detekce chorob. Vytvoření software pro detekci vybraných chorob ve střevech.

Osnova:

1. Úvod
   a. Cíle práce
   b. Kapsulární endoskopie
2. Počítačové vidění v endoskopii
   a. Úvod do počítačového vidění
   b. Použité algoritmy
   c. Aktuální stav trhu
3. Definice problému
   a. Související práce
4. Návrh a realizace software
   a. Návrh řešení
   b. Implementace řešení
5. Testování vyvinutého řešení
   a. Metodologie testování
   b. Výsledky testování
6. Diskuze nad výsledky
   a. Možný směr budoucího vývoje
7. Závěr

SEZNAM DOPORUČENÉ LITERATURY:

Podpis studenta: ..... **Datum:** .....

Podpis vedoucího práce: ..... **Datum:** .....