Středoškolská odborná činnost
Obor SOČ: 18 - Informatika

Aplikace pro iOS na rozpoznání hematomů

Škola: Gymnázium, Brno, třída Kapitána Jaroše 14
Kraj: Jihomoravský kraj

Autor:
Mikoláš Stuchlík

Vedoucí práce:
doc. Ing., Dipl.-Ing. Martin Drahanský, Ph.D.
Odborní konzultanti:
Mgr. Marek Blaha
MUDr. Milan Sepši, Ph.D.

Poděkování

Děkuji Mgr. Marku Blahovi, díky kterému jsem dostal příležitost na této práci pracovat. Děkuji docentu Drahanskému, doktoru Sepšimu a Gymnáziu, tř. Kapitána Jaroše, kteří mi poskytli prostředky, bez kterých bych práci dělat nemohl. A chtěl bych poděkovat své rodině, která mi dodala inspiraci.

Čestné prohlášení

Prohlašuji, že jsem svou práci SOČ vypracoval samostatně a použil jsem pouze podklady (literaturu, projekty, SW atd.) uvedené v seznamu vloženém v práci SOČ.
Prohlašuji, že tištěná verze a elektronická verze soutěžní práce SOČ jsou shodné.
Nemám závažný důvod proti zpřístupňování této práce v souladu se zákonem č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) v platném znění.

V Brně dne 5.3.2015:

Abstract

This thesis is about segmentation and recognizing of formations on human skin. This thesis is especially about possibilities, how to measure the area of formations called hematoma, using mobile device. We suppose, that mobile device has no input device, that can measure the area of hematoma, with exception for built-in camera.

Keywords: iOS, Hematoma, Image Processing, Recognizing ...

Abstrakt

Tato práce pojednává o problematice segmentace a rozpoznávání útvarů na lidské pokožce. Konkrétně se zabývá možnostmi, jak změřit plochu hematomu na pokožce člověka za použití mobilního zařízení. Předpokládá se, že mobilní zařízení je běžně vybavené, tedy má fotoaparát a nemá jiné další vstupní zařízení, které by mohlo nějakým způsobem pomáhat při měření.

Klíčová slova: iOS, Hematom, Zpracování obrazu, Rozpoznávání ...

Obsah

Kapitola 1

Úvod

V nemocničním prostředí je u pacientů po operacích srdce nutné sledovat, zda-li po operaci nedochází ke komplikacím. Jednou z těchto komplikací je prosakování stehů. Proto je nutné sledovat, zda-li se s postupem času průsak snižuje a hojení probíhá v pořádku, nebo zda-li se průsak nesnižuje, a je tedy potřeba podstoupit další operaci.

K určení, zda-li se průsak snižuje nebo ne, sledujeme hematom, který průsak vytváří. Pokud se v průběhu času hematom zvětšuje, víme, že k snižování průsaku nedochází.

Za tímto účelem lékaři kontrolují, zda-li se hematom zvětšuje. Proto je poptávka po mobilních aplikacích, které by lékaři při kontrole ulehčovaly práci. Tedy aby lékař měl možnost vést u každého pacienta jednoduše záznamy o průběhu hojení.

Mobilní aplikace, která by toto umožňovala, bude muset umět:

- vést na každého pacienta „složku“, kde by se shromažďovaly údaje
- tyto složky bezpečně ukládat a sdílet
- detekovat vzdálenost fotoaparátu od rány
- rozpoznat a změřit samotný hematom

Moje práce má za cíl vytvořit aplikaci, která bude umět rozpoznat a změřit hematom za použití pokročilých možností nastavování fotoaparátu a poskytnout výsledek ve formě nalezeného obrazce. Zároveň má mít možnost tyto informace uložit.

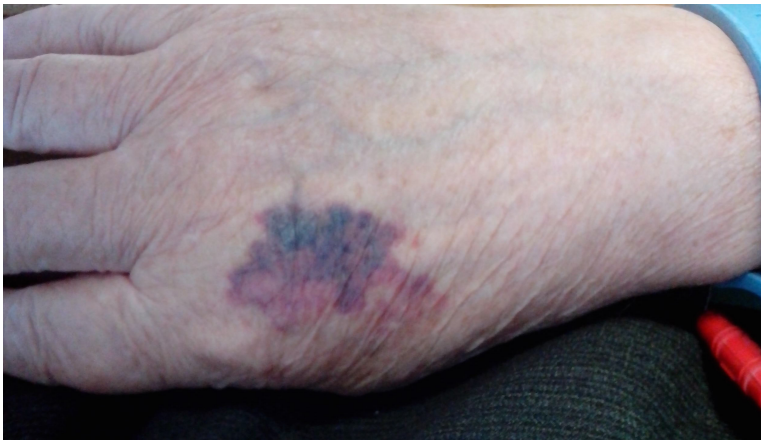
Kapitola 2

Teoretický rozbor

Tato kapitola obsahuje popis některých základních pojmů a technologií, kterých se bude v práci využívat.

2.1 Hematom

Hematom je rozsáhlejší únik krve do prostoru kůže, který může při dostatečné intenzitě způsobit otok citlivý na dotek. Na kůži se viditelně projevuje jako oblast, která se svou barvou viditelně liší od okolí.[**?**] Vzniká například v důsledku zranění. V našem případě vzniká při poškození kůže při zavedení kardiostimulátoru do podkoží. Charakteristická barva hematomu je způsobena hemoglobinem obsaženým v krvi a jeho degradací a vstřebáváním. Při postupném rozkladu krve, která tvoří hematom, se postupně mění jeho barva. Proto hematom nabývá nejen červené, červenohnědé a hnědé barvy, ale i odstínů žluté a dalších barev, která jsou ale vesměs příznakem vstřebávání hematomu.



Obrázek 2.1: Hematom na ruce pacienta.

2.2 Současný stav

V současnosti není na službách určených pro distribuci aplikací (App Store, Play Store, Amazon) dostupná žádná aplikace, která by se soustředila na detekci hematomů.

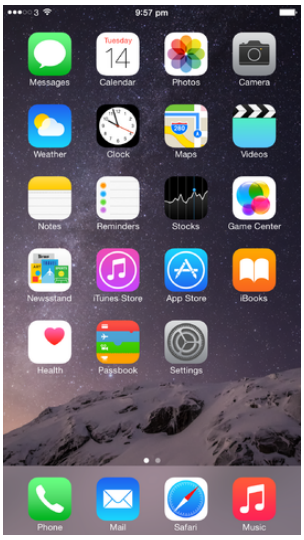
2.3 iOS

iOS je operační systém vyvýjený firmou Apple a vydaný v roce 2007 tehdy ještě pod názem iPhone OS. Je určen pro mobilní zařízení a téměř bez výjimky je nasazen na zařízeních, jejichž hlavním ovládacím prvkem je dotykový display. V historii se objevil například na zařízeních iPhone, iPad či Apple TV a dalších. K vývoji aplikací na iOS je nutné využít jazyka Objective-C, i když kompilátor umožňuje v některých případech zkompilovat i jazyk C++.

Obrázek 2.2: Vzhled iOS v průběhu času.

2.3.1 iOS 8

iOS 8 byl vydán pro veřejnost 17. listopadu 2014. Pro uživatele přinášel minimum změn, přinesl však mnohé změny a vylepšení pro vývojáře. Pro tuto práci je nejdůležitější, že přinesl značná rozšíření v oblasti ovládání kamery a práce s obrazem vůbec, a to skrze knihovnu AVFoundation. Aplikace pro iOS 8 mohou být vyvíjeny pouze za použití programu Xcode.



Obrázek 2.3: Domovská obrazovka iOS 8.

2.4 Použité technologie

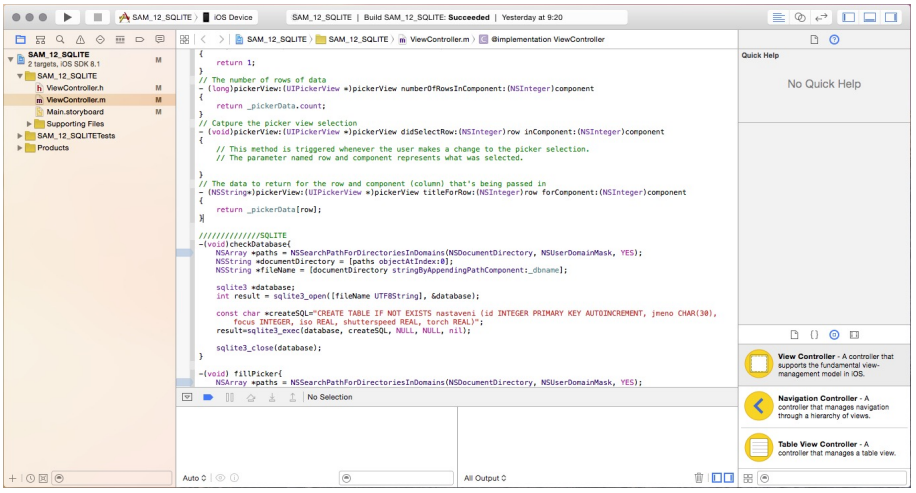
2.4.1 Mac OS X

Mac OS X je desktopový operační systém s grafickým rozhraním vyvíjený firmou Apple. Jedná se o systém postavený na operačním systému Unix. Mac OS X je nástupce Mac OS 9 a první verze Mac OS X byla vydána v roce 2004. Je vyvíjen jako uzavřený, ale využívá některých open source komponent. Mac OS X je nasazován pouze na počítačích Apple, jako například iMac, Mac Pro, Mac Book a další. K vývoji aplikace byla použita verze Mac OS X 10.10.2 Yosemite. Tento systém byl při vývoji využit, neboť je vyžadován pro spuštění Xcode 6.

2.4.2 Xcode 6

Xcode je integrované vývojové prostředí vyvíjené společností Apple. Obsahuje podporu pro mnoho jazyků, určen je především pro vývoj aplikací na iOS a Mac OS X v jazycích C++, Objective-C a Swift. Xcode je šířen zadarmo distribuční službou App Store, ale pouze pro uživatele Mac OS X. Xcode 6 obsahuje všechny moduly, které jsou nutné k vytvoření aplikace - kompilátor, vytváření a řízení projektu, vývoj kódu, kompletní dokumentaci Mac OS X a iOS a grafické rozhraní sloužící pro tvorbu uživatelského rozhraní známé jako „storyboard“.

Obrázek 2.4: Plocha systému Mac OS X 10.10 Yosemite



Obrázek 2.5: Pohled na kód v Objective-C v Xcode.

2.4.3 Objective-C

Objective-C je programovací jazyk, který na počátku 80. let 20. století navrhl Brad Cox. Jedná se o objektově orientovaný jazyk a vznikl jako jazyk C doplněný o další prvky. Tento jazyk si v roce 1988 licencovala firma NeXT vlastněná Stevem Jobsem - jednou z hlavních postav společnosti Apple - a vytvořila jeho knihovny a vývojové prostředí NEXTSTEP. Po převzetí NeXTu Applem se stal NEXTSTEP základem pro Mac OS X. Verze tohoto vývojového prostředí od firmy Apple nese název Cocoa. V roce 2007 byla uvolněna Applem aktualizce Objective-C, která nesla název Objective-C 2.0.[1]

2.4.4 AVFoundation

AVFoundation je framework, který poskytuje základní podporu pro práci s audiovizuálními médii na iOS a Mac OS X. Slouží k nastavování proudů dat mezi jednotlivými komponenty v zařízení, například při ukládání dat z fotoaparátu. Byl vytvořen v jazyce Objective-C. Pro účely této práce byl využit při ovládání fotoaparátu a nastavování parametrů expozice. Po vydání iOS 8 byl totiž mohutně rošířen právě v oblasti práce s fotoaparátem.

2.4.5 iOS Developer Program

Z předchozích kapitol tedy vyplývá, že k programování aplikací na iOS je zapotřebí počítač od Apple, na kterém bude operační systém Mac OS X 10.9 a novější. Na tento systém pak lze nainstalovat Xcode 6. V Xcode 6 je možno aplikaci pro iOS spustit na simulátoru. Pokud však bude potřeba aplikaci vyzkoušet na fyzickém zařízení s iOS, bude nutné zařadit se do tzv. iOS Developer Programu. Zařazení do tohoto programu je placené (99 USD za rok), vzdělávací instituce ho mají za účelem výuky po schválení licence zdarma. Po zařazení do programu lze získat certifikáty, pomocí kterých se podepisují aplikace a nahrávají se do vlastních iOS zařízení. Tyto aplikace pak lze spustit po určitý čas od nahrání, pakliže nejsou vydány na App Store.

2.4.6 OpenCV

OpenCV je open source knihovna, která slouží k počítačové práci s obrazem. Tato knihovna obsahuje více než 2500 algoritmů. Tuto knihovnu využívají ve svých produktech velké korporace jako Google, Microsoft nebo Sony či Honda. Tato knihovna má interface v C, C++, Pythonu a Javě. Lze ji použít na téměř všech platformách od desktopových distribucí Linuxu přes Android a Windows až po Mac OS X a iOS. V této práci bude využita pro zpracování obrazových dat pořízených fotoaparátem.

2.4.7 SQLite

SQLite je open source knihovna, která slouží k ukládání dat. Je velmi málo závislá na ostatních knihovnách, nevyžaduje server a nejsou nutné žádné další konfigurace. Pro interakci s uživatelem využívá jazyk SQL. V této práci se využívá k ukládání výsledků a nastavení aplikace.

Kapitola 3

Návrh

Tato kapitola se zabývá návrhem aplikace. Budou zde popsány hlavní funkce a uživatelské rozhraní.

3.1 Funkce aplikace

Aplikace, které se snažíme dosáhnout, by měla mít tyto základní funkce:

- pořizovat snímky v co nejvyšší možné kvalitě
- detekovat polohu hematomu a jeho plochu v pixelech
- ukládání snímků s výsledky, které aplikace poskytla
- přehlednost

Tento seznam a úspěšnost detekce budou na konci práce sloužit jako kritéria pro zhodnocení úspěšnosti práce.

3.2 Grafické uživatelské rozhraní

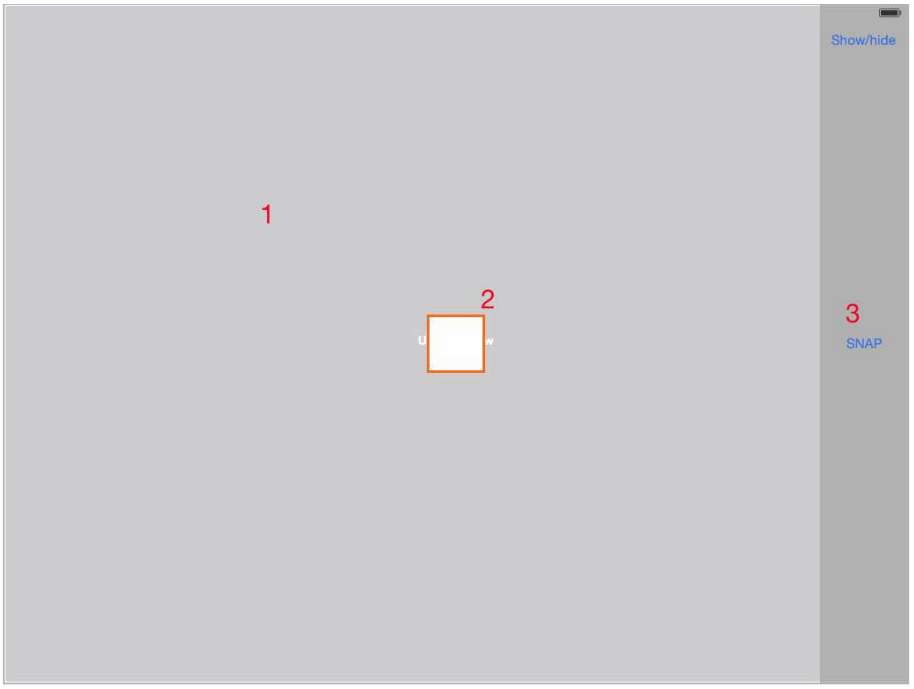
Grafické uživatelské rozhraní aplikace se dělí na dvě obrazovky: obrazovku „základní“, to jest tu, kterou uživatel vidí po zapnutí aplikace, a obrazovky „zpracování obrazu“, pomocí které je schopen změřit plochu hematomu.

Obě obrazovky vypadají podobně. Vždy na pozadí je roztažen přes celý display pracovní snímek nebo náhled a vpravo je šedý pásek obsahující obslužná tlačítka. Na obou obrazovkách je tlačítko k přepnutí z jedné na druhou.

Aplikace je orientována v režimu „landscape right“, tedy na šířku směrem doprava.

3.2.1 Základní obrazovka

Základní obrazovka obsahuje následující prvky: náhled fotografie (1), hledáček (2), tlačítko pro pořízení snímku (3).



Obrázek 3.1: Náhled v editoru

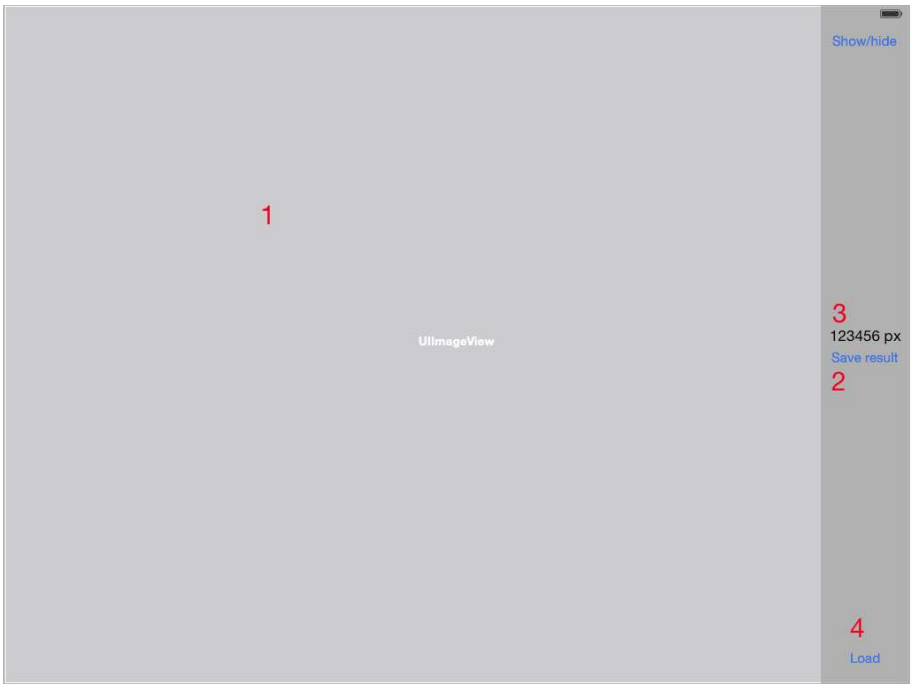
Náhled fotografie (1) slouží uživateli k náhledu na výsledný snímek.

Hledáček (2) musí uživatel umístit do středu sledovaného objektu. Do středu hledáčku je totiž zaměřen autofokus a plocha uvnitř hledáčku navíc slouží při pozdějším zpracování fotografie.

Tlačítko pro pořízení snímku (3) způsobí přepnutí ze základní obrazovky a pošle do fotoaparátu příkaz k získání dat.

3.2.2 Obrazovka zpracování obrazu

Obrazovka pro zpracování obrazu obsahuje tyto prvky: zpracovávaný snímek (1), tlačítko pro uložení výsledků (2), počet pixelů (3), tlačítko pro načtení již zpracovaných snímků (4).



Obrázek 3.2: Náhled v editoru

Zpracovávaný snímek (1) se zobrazí pro lepší orientaci jako vystínovaná mapa. Uživatel pak vybírá dotykem jednotlivé zóny, které se buďto přidají, nebo odeberou od celkové plochy. Tato funkce je možná pouze u nově pořízených snímků.

Tlačítko pro uložení výsledků (2) uloží snímek do knihovny fotografií a přiřadí jemu příslušný zápis do SQLite, který bude obsahovat informaci o vybrané ploše.

Počet pixelů (3) je text, který zobrazuje, kolik pixelů mají vybrané zóny dohromady. Toto číslo bude posléze sloužit k vypočítání plochy hematomu.

Tlačítko pro načtení již zpracovaných snímků (4) odkáže uživatele do knihovny, kde pomocí metod připravených Applem vybere snímek, který chce prohlížet. V případě, že tento nebude mít zápis v SQLite, bude sice zobrazen, ale uživatel bude upozorněn na chybějící data na místě, kde by se normálně zobrazoval počet pixelů.

3.3 Programování

V této sekci budu popisovat veškeré „psané“ části aplikace, tedy jak třídu HematomaDetektor, tak i kód kontroleru a jeho jednotlivé součásti.

3.3.1 Kontroler

Metodami kontroleru se již nebudeme dále zabývat nad rámec této kapitoly.

3.3.1.1 Metoda ViewDidLoad a funkce spouštěné po startu aplikace

Metody spouštěné po spuštění aplikace vesměs inicializují objekty a nastavují jejich instanční proměnné. Nastavují se tři základní oblasti: HUD - kde se upravuje barva a pozice objektů, AVFoundaton - objekty sloužící k práci s fotoaparátem a SQLite - ověřuje existenci databáze, popřípadě jednu vytvoří. Je v ní také inicializován objekt HematomuDetector.

V metodě, která nastavuje AVFoundation, je zajímavých pouze několik věcí: nastavuje kvalitu snímku na nejvyšší, nastavuje barevný prostor, ve kterém bude výstup z fotoaparátu, a přiřazuje datové proudy.

3.3.1.2 Metoda snapButtonEvent

Tato metoda zablokuje tlačítko pro focení, přepne obrazovku do režimu zpracování snímku a poskytne odkazy na oblasti paměti, kde jsou uložena data pořízeného snímku. Snímek se uloží na trvalé úložiště, kde bude po čas zpracovávání v plné velikosti. Po uložení snímku se na pozadí znovu připraví fotoaparát do režimu snímání. Obrázek se načte do paměti jako objekt UIImage a započne proces zpracovávání.

3.3.1.3 Metoda saveButtonEvent

Po dokončení zpracovávání se obrázek odstraní z trvalého úložiště a je nahrazen obrázkem o menší kvalitě. Pokud by se obrázek ukládal v plné kvalitě, představovalo by to obrovské nároky na kapacitu zařízení. Pořízený obrázek má 2592x1936 pixelů, kde každý pixel obsahuje 16 bitů informací. Jeden průměrný takto pořízený snímek má 9 - 11 MB. Uvážíme-li, že lékař může během jednoho dne zkontrolovat 10 pacientů, tak dojdeme k tomu, že by aplikace byla pro delší ukládání dat zcela nepoužitelná. Proto se snímek uloží v nižší kvalitě s několikanásobně nižšími nároky na prostor (průměrně 500 kB) a počet pixelů označených zón se pro přesnost uloží do databáze SQLite.

3.3.1.4 Metoda showHideButtonEvent

Po stisku tohoto tlačítka dojde k tomu, že se provede if then konstrukce, kde jako podmínka bude, zda-li je prvek imageView viditelný, nebo ne. Pokud ano, skryjí se prvky

příslušné obrazovce zpracovávání obrazu a zobrazí se prvky příslušné základní obrazovce. V případě, že je podmínka neplatná, provede se proces opačně.

3.3.1.5 Metoda loadButtonEvent

Metoda loadButtonEvent otevře galerii obrázků Apple. V momentě vybrání snímku bude aplikace upozorněna a zavře galerii. Načtený snímek se uloží do objektu HematomaDetector a načte se jeho příslušný zápis z databáze SQLite.

3.3.1.6 Metoda touchesBegan:withEvent:

Tato metoda je volána ve chvíli, kdy se uživatel dotkne obrazovky. V případě, že se uživatel dotýká uvnitř imageView a v paměti je uložen pracovní obrázek, je volána funkce objektu HematomaDetector pickAreaWithCoordsOnX:Y: a je provedena. V případě, že pracovní obrázek není k dispozici, tak se metoda neprovádí.

3.3.2 Třída HematomaDetector

Třída HematomaDetector je základním kamenem celé aplikace. Využívá obraz pořízený fotoaparátem a provádí na něm výpočty. Využívá knihovny OpenCV, a proto musí být kompilována C++ kompilátorem.

V této třídě je několik statických metod, které vesměs slouží jen jako podpůrné. Proto se v dalších kapitolách zaměříme pouze na skutečně důležité metody a v této kapitole bude uveden základní popis všech metod a instančních proměnných.

3.3.2.1 Instanční proměnné třídy

Objekt třídy HematomaDetector má čtyři instanční proměnné. Jedná se o tři objekty UIImage (photo, processPhoto, outputPhoto) a proměnnou long pixelAmount.

3.3.2.2 Konstruktor

Konstruktor této třídy přiřadí všem objektům UIImage hodnotu nil a proměnnou pixelAmount na -1.

3.3.2.3 Metoda UIImageFromCvMat:

Tato metoda získá na vstupu objekt cv::Mat, který slouží k reprezentaci obrázků a je poskytován v knihovně OpenCV. Na výstupu potom poskytne obrázek reprezentovaný objektem UIImage z UIKit.

3.3.2.4 Metoda cvMatFromUIImage:

Metoda cvMatFromUIImage: pracuje obdobně jako UIImageFromCvMat:, ale s tím rozdílem, že na vstupu je objekt UIImage a na výstupu objekt cv::Mat.

3.3.2.5 Metoda getMap

Tato metoda podrobí obrázek UIImage photo výpočtům, na jejichž konci uloží do UIImage processPhoto obrázek ve formě mapy, na kterém budou zvýrazněné jednotlivé segmenty pokožky a na kterém potom uživatel bude moct vybírat jednotlivé zóny. Více o této metodě v další kapitole.

3.3.2.6 Metoda pickAreaWithCoordsOnX:Y:

Tato metoda propočítá, kterého místa se uživatel dotkl na obrázku UIImage processPhoto. Poté zvýrazní dané místo v objektu UIImage outputPhoto. Pokud se uživatel dotkne již zvýrazněného místa, tak proběhne proces stejně, ale zvýraznění se zruší. Více o této metodě v další kapitole.

3.3.2.7 Metoda saveData

Metoda saveData uloží zkomprimovaný obrázek UIImage outputPhoto jako JPEG do galerie. Jeho jméno využije jako ID pro zápis který uloží do SQLite. Tento zápis bude obsahovat informaci, kolik pixelů na původním obrázku obsahovala zvýrazněná plocha. Tedy uloží se tam hodnota z long pixelAmount.

3.3.2.8 Metoda loadData

Metoda loadData načte do proměnné UIImage outputPhoto obrázek z knihovny a jeho jméno použije pro vyhledání příslušného zápisu v SQLite. Tento zápis načte a uloží hodnotu do long pixelAmount. Data poté zobrazí na obrazovce.

Kapitola 4

Vývoj a implementace

V této kapitole bude popsán způsob, kterým se zpracovává obraz.

4.1 Struktura vstupního snímku

K pořízení snímku se využívá knihovna AVFoundation. Zavoláním metody captureStillImageAsynchronouslyFromConnection:completionHandler: objektu AVCaptureStillImageOutput se získá přístup k CMSampleBufferRef, který obsahuje jak veškeré informace o fotce, tak fotku samotnou.

Světlo dopadá na fotočip skrze tzv. Bayerovu masku, která obsahuje světelné filtry pro modré, zelené a červené světlo v poměru 1:2:1. Proto pro každý pixel jsou dvě informace o zelené barvě, ale pouze jedna pro červenou a jedna pro modrou.[3] Proto se provádí proces nazývaný Bayerova interpolace[2], který poskytne strukturu, se kterou je již možno pracovat jako s digitální fotografií. Na tuto strukturu se ještě aplikují nastavení (jako například vyvážení bílé).

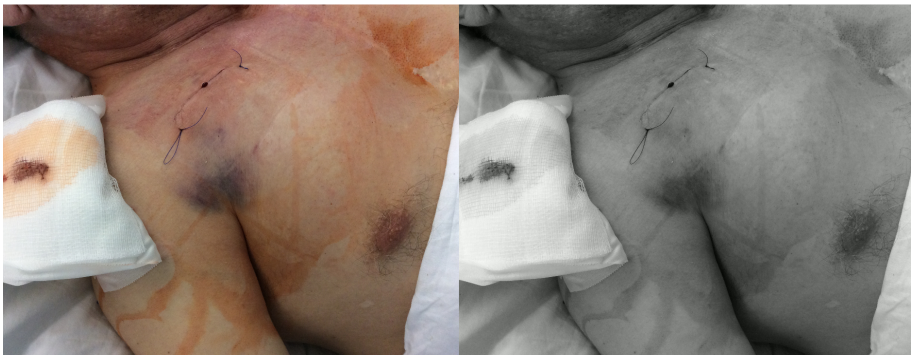
Na výstupu tedy nalezneme obraz ve formě proudu dat. Pro potřeby této práce byl nastaven výstupní barevný prostor na BGRA, vzhledem ke snažšímu ukládání do galerie jako .png pro pozdější zpracování. Proud dat je ve formě bitů a tyto bity jsou v pořadí $B_1, G_1, R_1, A_1, B_2, G_2, ...A_n$, kde n je počet bitů obrazu, tedy $pocetRadku * pocetSloupcu *$ 4. Teoreticky by bylo možné provádět výpočty už na tomto výstupu z fotoaparátu, ale dokud je odkaz pro data z výstupu aktivní, nemůže se vyfotit další snímek, popřípadě dojde ke zmatení výpočtů. Proto se data nejdříve uloží, aby se uvolnil fotoaparát a poté se snímek zpracuje.

4.2 Algoritmizace funkce getMap

Funkce getMap má za cíl vytvořit pro uživatele obrázek, který bude vypadat jako politická mapa. Na této mapě mají být vyznačeny segmenty obrázku s různou barvou, kde uživatel vybere dotykem plochu nebo plochy, které chce změřit.

4.2.1 Barevný prostor YCbCr

Barevný prostor YCbCr (popřípadě YCrCb) se používá u videa nebo digitálních fotografií. Složka Y vyjadřuje jas na intervalu $< 0, 255 >$ a složky Cb a Cr jsou modrý a červený chrominanční komponent, oba na stejném intervalu jako Y.



Obrázek 4.1: Vlevo původní obrázek, vpravo kanál Y.

Obrázek 4.2: Vlevo kanál Cr, vpravo kanál Cb.

Barevný prostor YCbCr pro potřeby této práce slouží jako základní prostor, na kterém jsou prováděny veškeré výpočty. Je to proto, že při detekci hematomu se nemusí počítat s osvětlením, protože osvětlení vyjadřuje kanál Y.[1]

[1] V případě, že je některé místo příliš tmavé nebo příliš světlé, např. odleskem po natření krémem nebo štěrbina mezi trupem a rukou, potom nepomůže samotné zanedbání kanálu Y.

Knihovna OpenCV vytváří barevný prostor YCrCb z prostoru RGB takto:

$$Y \leftarrow 0.299 \cdot R + 0.587 \cdot G + 0.114 \cdot B$$

$$Cr \leftarrow (R - Y) \cdot 0.713 + delta$$

$$Cb \leftarrow (B - Y) \cdot 0.564 + delta$$

$$R \leftarrow Y + 1.403 \cdot (Cr - delta)$$

$$G \leftarrow Y - 0.714 \cdot (Cr - delta) - 0.344 \cdot (Cb - delta)$$

$$B \leftarrow Y + 1.773 \cdot (Cb - delta)$$

kde $delta = 128$ pro 8-bit obrázek.[4]

4.2.2 Prahování

Segmentování ploch rozdílné barvy na pokožce má sice mnoho využití, přesto pro účely této práce je stěžejní segmentace hematomů. Proto bylo potřeba zjistit, co mají hematomy společného. Předpoklad byl, že hematomy mají vesměs společné to, že se liší výrazně alespoň v jedné barevné složce od pokožky. Za tímto účelem byla vytvořena aplikace, která tuto domněnku má ověřit.

4.2.2.1 Aplikace SOC_DOC_CAM_3

Tato aplikace je poměrně jednoduchá, proto bude popsána pouze hlavní funkce této aplikace. Na vstupu je snímek z fotoaparátu nebo galerie. Uživatel zadá velikost poloviny hrany čtvercové matice a její střed. Hodnoty kanálů Cb a Cr se zprůměrují a na výstupu je průměrná hodnota těchto kanálů.

Do této aplikace byly nahrány snímky a odečteny hodnoty pro plochu, kde je hematom a kde je pouze kůže. Z těchto hodnot byla sestavena tabulka.

Z této tabulky vyplývá, že lidé s vláčnější kůží s méně výrazným pigmentem mají výraznější rozdíl mezi kůží a hematomem v kanálu Cb, kdežto lidé s výraznějším pigmentem s pevnější kůží mají rozdíly viditelné jak v kanálu Cr, tak Cb. V kanálu Cr je ovšem často změna výraznější, ale jen o minimum.

Obrázek 4.3: Ukázka průměrné hodnoty barvy kůže. (hodnota -1 značí střed)

Hematom		**Kůže**		**Typ pokožky**
Cr	**Cb**	**Cr**	**Cb**	**-**
130	141	132	127	stará
135	114	135	124	stará
128	126	127	134	stará
128	120	129	125	stará
141	122	134	117	mladá
145	121	136	114	mladá

Obrázek 4.4: Tabulka naměřených hodnot - ukázka.

4.2.3 Prahování na základě snižování počtu barev

Prahování na základě snižování počtu barev se snaží vytvořit takový obrázek, kde budou místa s výraznějšími rozdíly „lépe vidět“. Toto má výhody jak pro uživatele, tak pro počítač. Pro uživatele je výběr z různých jasně rozlišených barevných ploch snadný a pro počítač relativně nenáročný. V praxi se použije tento vzorec:

$$novaBarva = starabarva - ((staraBarva - koeficient)\%delitel)$$

kde $koeficient$ slouží k vyvážení specifických rozdílů obrázků (většinou se jedná o lehké zabarvení celého obrázku), operace % znamená „zbytek po celočíselném dělení“ a *delitel* je číslo, které určuje, jak výrazné budou přechody a kolik barev tedy v kanálu zbyde.

4.2.3.1 Aplikace SOC_MODULO_1

Tato aplikace opět nebude podrobněji popsána. Vychází z principů popsaných výše v této sekci a na vstupu očekává snímek a čísla *delitel* pro kanály Cr a Cb. Aplikace je schopna pracovat i s kanálem Y, tento kanál však není při zpracovávání využit.

Obrázek 4.5: Ukázka výstupu aplikace pro *delitel* uvedený v nastavení

Experimentálně bylo zjištěno, že pro největší množství hematomů je možné spatřit plochu hematomu jasně pro nastavení hodnoty *delitel* na kanálu Cr i Cb na 18 nebo 19. Při detekci hematomu si tedy vytvořím čtyři obrázky - podle hodnoty *delitel* - Cb18, Cb19, Cr18 a Cr19. Nechť jsou tyto obrázky nazývány „podmapy“.

4.2.4 Detekce přijatelných útvarů

V momentě, kdy mám čtyři podmapy, z nichž některé obsahují vyrýsovaný hematom a jiné nikoliv, přichází na řadu otázka, jak zajistit, aby uživatel dostal ty důležité, a ty, které by byly matoucí, zahodit. Samozřejmě by bylo možné uživateli podat výslednou mapu poskládanou ze všech čtyř podmap. Tato výsledná mapa by ale byla méně přehledná.

Za tímto účelem využijeme zaměřovač. Na každé podmapě provedeme následující operaci:

- všechny soustavné plochy si postupně vybarvíme bílou a zároveň zjistíme počet takto vybarvených pixelů

- v případě, že v průběhu vybarvování se stane, že bílá dosáhne až na kraj obrázku, tak plochu, která toto učinila, necháme nevybarvenou
- v případě, že počet pixelů některé z ploch přesáhl 40% celkového počtu pixelů, pak tuto plochu také odbarvíme

Potom podmapy sloučíme do jedné mapy. Postup je následující:

- Pokud žádná mapa nevyhovuje, tj. na žádné není bíle vybarvený prostor, potom výsledek bude oznámen uživateli
- Potom se berou podmapy jedna po druhé a otiskují se (to znamená, že se přepíše daný kanál na mapě) do výsledné mapy
- První podmapa se otiskne tak, jak je
- Pokud existují další podmapy a jsou to posmapy shodného kanálu, potom se hodnota pro každý pixel zprůměruje
- Pokud se jedná o podmapy jiného kanálu, tak se tato podmapa otiskne do mapy tak jak je

Výsledná mapa se pokytne uživateli jako prostor, ve kterém může označovat plochy, u kterých chce spočítat povrch.

4.3 Výběr zón

Výběr zón je prováděn tak, že na vstupu je mapa a souřadnice, kde se jí uživatel dotkl. Objekt, který reprezentuje mapu, se nakopíruje do druhého objektu, který bude reprezentovat zóny, které uživatel označil. V momentě, kdy se uživatel dotkne obrazovky v určitém bodě, tak se počítač zeptá, zda-li je tento bod již označen. Počítač pak projde všechny pixely, které náleží souvislé stejnobarevné ploše na mapě a v objektu, který reprezentuje výsledek, všechny tyto pixely přebarví. Přebarvení závisí na tom, zda-li byla již plocha označena, nebo nikoliv. Pokud ne, přebarví pixely v objektu, který reprezentuje výsledek na jasně zelenou barvu - v kanálech Cr a Cb, v kanálu Y ponechá stejnou hodnotu. Je to z toho důvodu, že Y dokáže zvýraznit povrch kůže a uživatel se snáze orientuje. Pokud ano, nakopíruje do těchto pixelů hodnoty pixelů v mapě.

Kapitola 5

Výsledky

V této kapitole se popisují výsledky práce a komplikace. Popisují se zde i požadavky na prostředí.

5.1 Výsledky programování

Naneštěstí vzhledem ke komplikacím během programování nebyla výsledná aplikace doprogramována dokonce. Práce však již dospěla do fáze, kdy je již pouhým okem možné určit, zda-li segmentace byla provedena úspěšně, či nikoliv. Aplikace však ještě sama není schopna nabídnout uživateli prostředí pro výpočet samotné plochy a automaticky rozpoznat, na které podmapě se nachází výsledek, a na které ne.
Následuje přehled vlastností a funkcí aplikace s popisem:

- **Uživatelské rozhraní** bylo vytvořeno úspěšně.
- **Pořizování snímku** funguje správně.
- **Ukládání snímku a informace o segmentované ploše** funguje správně
- **Automatické rozpoznávání platných podmap** není plně funkční. Vyskytly se problémy s reprezentací dat v paměti, protože funkce cv::clone() neudělá dostatečně hluboké klonování objektu tak, aby šlo jednoduše zkopírovat jejich obsahy a potom s nimi pracovat jako s nezávislými objekty. Práce s podmapami je navíc náročná na paměť i čas. Vzhledem k popsaným komplikacím ke dni odevzdání práce nebyla aplikace ještě funkční. Přesto již tento problém byl překonán a jsou již k dispozici čtyři podmapy tak, jak byly popsány v předchozích kapitolách.

- **Slučování podmap a tvorba mapy** není plně funkční, protože její testování je přímo závislé na podmapách.
- **Rozpoznávání bodu, kde se uživatel dotkl** funguje správně.
- **Označování plochy** je funkční na jednoduchých obrázcích. Na mapě ještě nebyl otestován.

5.2 Fotografování

Uživatel by měl dbát při pořizování snímku na několik věcí, aby získaná data byla co nejkvalitnější. V místnosti by měl být dostatek světla, aby fotoaparát nemusel nastavovat příliš vysoké ISO, popřípadě aby nemusel nedostatek světla přinejhorším kompenzovat zvýšením času závěrky. Pokud by totiž bylo ISO příliš vysoké, mohlo by docházet k nepřesnostem vzhledem k šumu, který by vznikl. Vysoký čas závěrky by zase mohl způsobit, že by byl vstupní snímek rozmazaný. Zároveň by měl dbát na to, aby byla co největší plocha hematomu pokud možno rovnoběžná s plochou, na kterou se fotí.

5.3 Segmentace

V této sekci budou následovat ukázky segmentace a komplikace, které mohou při segmentaci vzniknout. Uvedu pro příklad pouze jeden obrázek, okomentuji důvody úspěchu. Přesto se budu více zabývat obrázky, u kterých nebyla segmentace z nějakého důvodu úspěšná.

5.3.1 Ukázky úspěšné segmentace

Ukázka úspěšně rozpoznatelného hematomu.

Obrázek 5.1: Vstupní snímek

Obrázek 5.2: Takto vypadá původní snímek po snížení počtu barev a zanedbání kanálu Y

Obrázek 5.3: Snímek po snížení počtu barev s kanálem Y, takto ho uvidí uživatel pro výběr zón.

Toto je téměř ideální snímek. Ze vstupního snímku je již na první pohled hematom jasně vidět. Kůže není nikde vyloženě přsvětlená nebo tmavá. Hematom nepřekrývají na kůži ani žádné otisky.

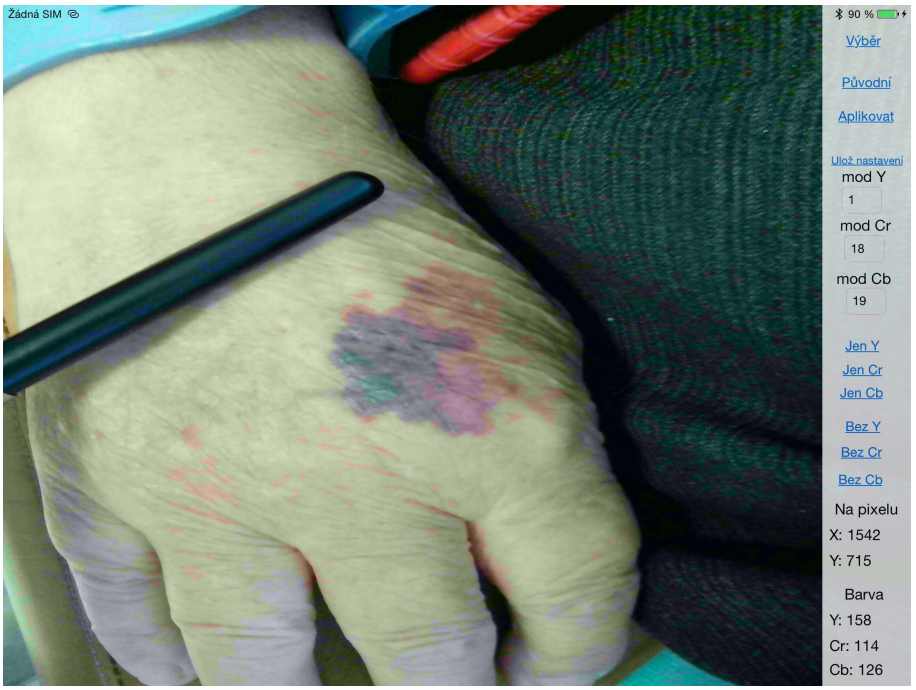
5.3.2 Ukázky neúspěšné segmentace

V této sekci jsou popisovány zjištené důvody, proč nebyl hematom správě zpozorován.

5.3.2.1 Neúspěch při vybírání podmap

Obrázek 5.4: Vstupní snímek

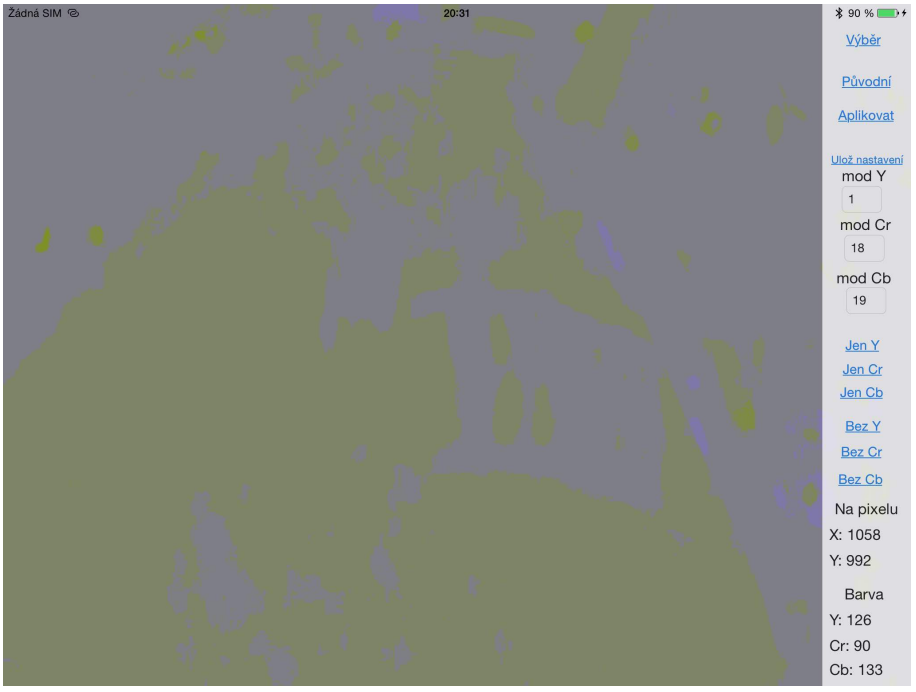
Obrázek 5.5: Takto vypadá původní snímek po snížení počtu barev a zanedbání kanálu Y



Obrázek 5.6: Snímek po snížení počtu barev s kanálem Y, takto ho uvidí uživatel pro výběr zón.

Toto je nejpodstatnější chyba. Z obrázku je 5.4 je totiž jasně patrné, že hematom má více barev. Proto je pravděpodobné, že uživatele bude zajímat i podmapa, která se nevyhodnotí jako podstatná. Uživatel tedy nedostane na výběr některé segmenty hematomu. Jedná se tedy o chybu ve fázi vybírání platných podmap.

Obrázek 5.7: Vstupní snímek



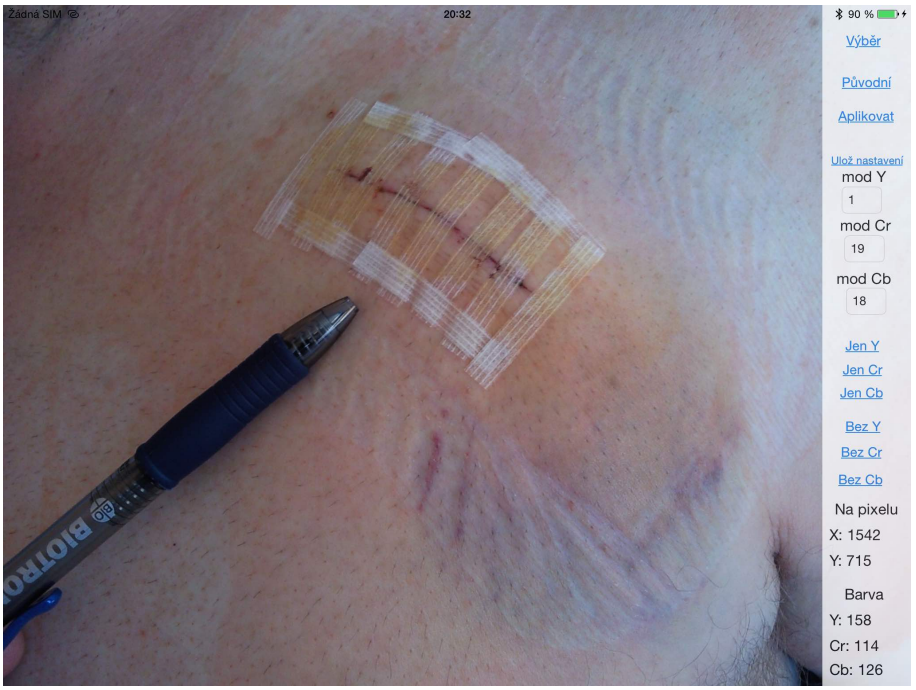
Obrázek 5.8: Takto vypadá původní snímek po snížení počtu barev a zanedbání kanálů Y a Cr

Obrázek 5.9: Snímek po snížení počtu barev s kanálem Y, takto ho uvidí uživatel pro výběr zón.

Druhý případ, který může negativně ovlivnit segmentaci, je, pokud se hematom nalézá na rozhraní podmap. V tomto případě lze hematom spatřit jako průnik dvou nebo více ploch podmap, kdy každá plocha ovšem sama o sobě nereprezentuje hematom (viz obrázek 5.8).

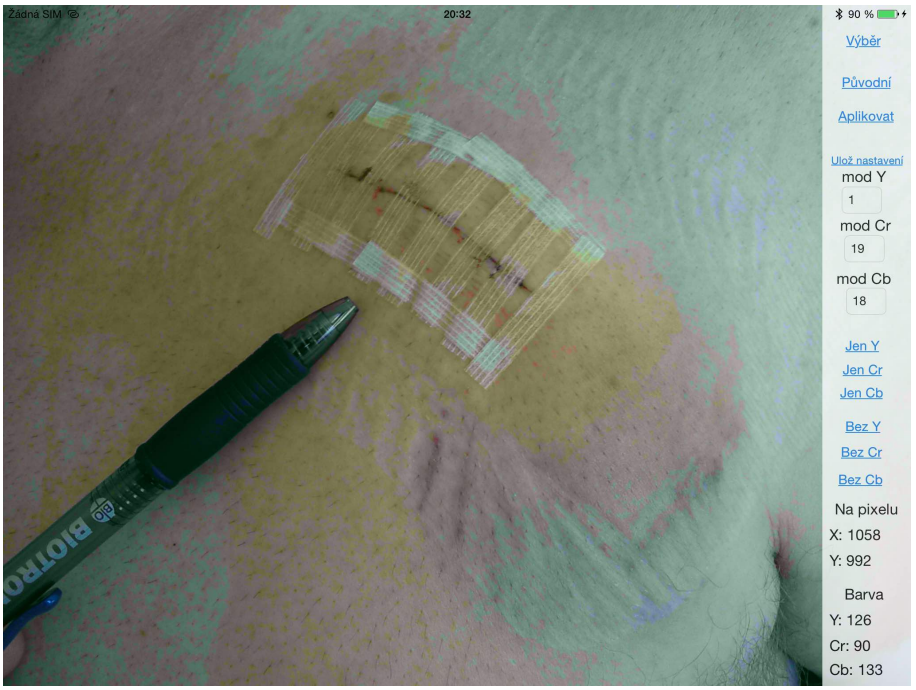
5.3.2.2 Neúspěch z důvodu přesvětlení

Následující obrázky obsahují za jiných okolností celkem dobře viditelný hematom. Protože ale na plochu hematomu působí silné boční nasvícení a navíc je zjevně fotografie pořizována pod poměrně ostrým úhlem vzhledem k ploše, na které se podstatná část hematomu rozkládá, dochází ke ztrácení hematomu a ani jedna kombinace podmap není schopna hematom zachytit.



Obrázek 5.10: Vstupní snímek

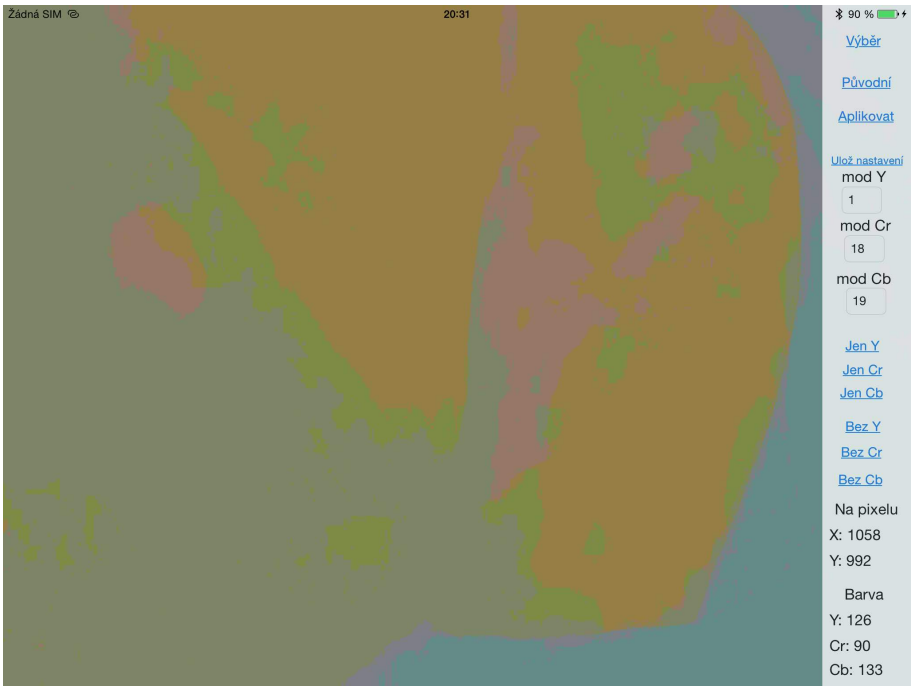
Obrázek 5.11: Takto vypadá původní snímek po snížení počtu barev a zanedbání kanálu Y



Obrázek 5.12: Snímek po snížení počtu barev s kanálem Y, takto ho uvidí uživatel pro výběr zón.

5.3.2.3 Neúspěch z důvodu nedostatku světla v prostoru mezi končetinou a trupem

Obrázek 5.13: Vstupní snímek



Obrázek 5.14: Takto vypadá původní snímek po snížení počtu barev a zanedbání kanálu Y

Obrázek 5.15: Snímek po snížení počtu barev s kanálem Y, takto ho uvidí uživatel pro výběr zón.

Na těchto obrázcích je celkem jasně vidět, že při přechodu od osvětleného místa do neosvětleného dochází ke značným zkreslením v segmentaci, což může vést ke komplikacím ve vedení statistiky o hojení hematomu.

Kapitola 6

Závěr

6.1 Zhodnocení práce

Cílem této práce bylo vytvořit aplikaci, která bude umět detekovat hematom za použití pokročilých možností nastavování fotoaparátu a poskytnout výsledek ve formě nalezeného obrazce.

Bohužel musím konstatovat, že tento cíl naplněn nebyl. Ke dni odevzdání práce bohužel nebyla aplikace plně funkční. Tato práce však přináší možnosti řešení problému.
Ke dni odevzdání práce aplikace:

- umí pořizovat snímky ve vysoké kvalitě
- neumí detekovat plochu hematomu
- umí zjistit, kolik pixelů má vyznačená plocha
- umí ukládat snímky s výsledky
- uživatelské rozhraní je minimalistické a není zde žádný matoucí prvek

6.2 Výstupy práce

Tato práce popisuje možnosti, jak detekovat hematom na základě rozdílných barev. Doporučuje také využití barevného prostoru YCbCr a vysvětluje jeho výhody. Popisuje problémy vzniklé při detekci a nastiňuje jejich řešení. Tato práce také shromažďuje galerii testovacích subjektů, na nichž je možné provádět další výzkum.

6.3 Problémy

Největší problém, na který jsem při psaní práce narazil je zajistit, aby byly uživately nabídnuty správné podklady pro výběr plochy, kde se hematom nachází. Jedná se o problém při zpracování, kdy podmínky, pro zahrnutí jednotlivých podmap do výsledné mapy, nejsou nastaveny dobře. Tento problém by mohlo vyřešit, pokud by se podmínky aplikovaly nejen na podmapy, ale i na kombinace vzniklé z různých podmap. Nevýhodou tohoto přístupu je vysoká časová náročnost.

6.4 Návrhy na rozšíření

Aplikace by mohla využívat vícevláknového programování, pro optimalizaci času zpracování snímku. Zároveň by mohla využívat knihovny a prostředí, které by směřovaly výpočet na grafický čip. Při pořizování snímku by mohlo být využito dalších nastavení fotoaparátu, například vyvážení bílé, aby tozdíly pokožka - hematom byly co nejvýraznější. Doporučuji také pokračovat v dalším získávání subjektů, na kterých by se aplikace testovala. Je nutné totiž zjistit, zda-li různé barvy a struktury kůže várazněji ovlivňují segmentaci.

Literatura

[1] S. G. Kochan, překlad Karel Voráček: Objective-C 2.0 *Výukový kurz programování pro Mac OS X a iPhone*, Computer Press a.s., 2010, první vydání, ISBN 978-80-251-2654-7

[2] *Bayerova maska* [online]. [cit. 2015-03-02]. Dostupné z: http://www.fotoroman.cz/glossary2/1_bayer_mask.htm

[3] *Bayerova maska* [online]. [cit. 2015-03-02]. Dostupné z: http://fyzika.jreichl.com/main.article/view/1533-bayerova-maska

[4] *Miscellaneous Image Transformations* [online]. [cit. 2015-03-02]. Dostupné z: http://docs.opencv.org/modules/imgproc/doc/miscellaneous_transformations.html

[5] *OpenCV 2.4.9.0 documentation* [online]. [cit. 2015-03-04]. Dostupné z: http://docs.opencv.org

[6] APPLE INC. AV *Foundation Programming Guide*. Dostupné z: https://developer.apple.com/library/ios/documentation/AudioVideo/Conceptual/AVFoundationPG/AVFoundationPG.pdf

[7] *Developing iOS 7 Apps for iPhone and iPad*. STANFORD UNIVERSITY. [online]. [cit. 2015-03-04]. Dostupné z: https://itunes.apple.com/us/course/developing-ios-7-apps-for/id733644550

[8] *Raw image data from camera like "645 PRO"*. [online]. [cit. 2015-03-04]. Dostupné z: http://stackoverflow.com/questions/11612647/raw-image-data-from-camera-like-645-pro/11615472#11615472

[9] *How to get Bytes from CMSampleBufferRef , To Send Over Network*. [online]. [cit. 2015-03-04]. Dostupné z: http://stackoverflow.com/questions/6189409/how-to-get-bytes-from-cmsamplebufferref-to-send-over-network