KKY/ZDO

Semestrální práce z předmětu Zpracování digitalizovaného obrazu

Západočeská univerzita v Plzni
Katedra Kybernetiky
ZDO

Martin Kraus, Jan Burian
24. května 2023

Obsah

1 Úvod

Tento dokument slouží jako dokumentace k semestrální práci z předmětu Zpracování digitalizovaného obrazu. Jako programovací jazyk pro vypracování byl zvolen Python s využitím následujících standardních knihoven:

- *numpy*
- *scikit-image*
- *matplotlib*
- *cv2 (openCV)*
- *os*
- *json*

2 Zadání

Incision quality evaluation

Introduction

Distance learning through online means has proven its value in the past. Its epidemiological and energy-economic benefits continue to increase its relevance. However, some subjects seem almost impossible to teach by video. Among such subjects is the teaching of surgery. The goal of our application is to teach this very subject.

Using AI and machine learning tools, we aim to reduce the demands on the teacher and facilitate an objective assessment of the accuracy of student performance. The teaching is then done by video chat, and the AI based measurement complements the evaluation of the results.

The assignment

The goal of this assignment is to evaluate the quality of surgical stitching based on the image of the incision and the stitch. Students will be provided with a set of images depicting incisions and stitches made during a surgical procedure. The task of the students will be to use computer vision methods to extract and segment image data, and then use machine learning methods to perform qualitative analysis of the stitching. Students will evaluate the quality of the stitching based on parameters such as evenness and perpendicularity. The results of the analysis will be presented using graphical visualizations and metrics for stitching quality.

- The submission is expected to be a link to git repository
- Repository name contain identification of your team
- The final report in PDF is in the repository root
- There is file in your repository `src/run.py`
- The first argument is always output `.json` file
- The second argument might be `-v`. It will start visual mode with debug images.
- The `src/run.py` accepts image filenames as arguments

Example: run

```
cd GithubProjects/ZDO_Team99/src
python run.py output.json incision001.jpg incision005.png incision010.JPEG
```

Example: run with visualization

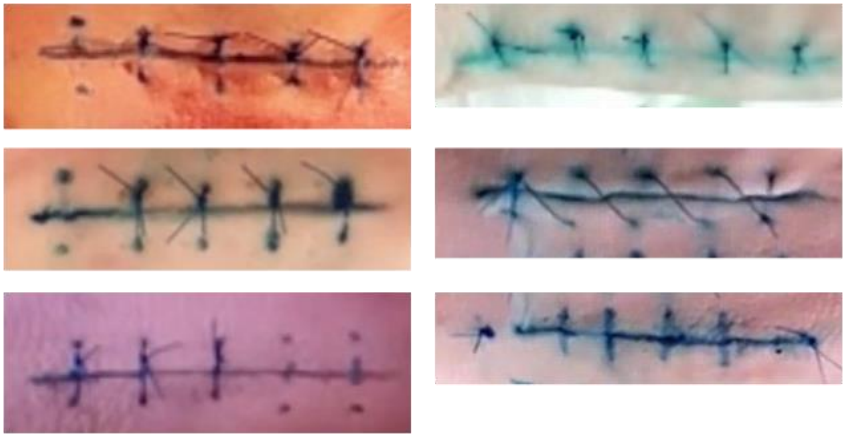
```
python run.py output.json -v incision001.jpg incision005.png inc
```

Output json

The structure of the output file describes the incision line and its crossings with the stitches. Crossings positions are measured in pixeles from the left beginning of incision line.

```
{
  [
    { "filename": "incision001.jpg",
      "incision_polyline": [[109.47,19.32],[111.88,42.19]]
      "crossing_positions": [13.8, 18.1, 19.0]
      "crossing_angles": [87.1, 92.3, 75.0]
    },
    { ...
    },
    { "filename": "incision010.JPEG",
      "incision_polyline": [...]
      "crossing_positions": [...]
      "crossing_angles": [...]
    },
  ]
}
```

Data



3 Poskytnutá data

Jedná se o medicínská obrazová data. V každém obrazu se zpravidla nachází jedna jizva a určitý počet stehů. V některých případech nachází pouze jizva bez stehů, v obrázku jsou pak často pouze "tečky", které značí začátek a konec potenciálního stehu. Celkem bylo poskytnuto 220 obrazů nepříliš dobré kvality. Průměrná velikost obrazu činila 295 pixelů (šířka) $\times$ 94 pixelů (výška). Jednotlivé obrazy byly pořízeny v různých světelných podmínkách různými snímači. Z důvodu malého počtu a rozmanitosti dat a bez potřebných expertních (lékařských) znalostí je tato úloha detekce velice obtížná.

Obrázek 1: Jizva a stehy př. 1

Obrázek 2: Jizva a stehy př. 2

Obrázek 3: Jizva a stehy př. 3

4 Vypracování

Cílem této semestrální práce je navržení algoritmu na nalezení jizvy a stehů v medicínských obrazových datech. Pro tento účel budou použity metody počítačového vidění, které byly z velké části předvedené na cvičeních a přednáškách z předmětu "Zpracování digitalizovaného obrazu (KKY/ZDO)". Odkaz na GitHub repozitář: `https://github.com/BerassHaggy/ZDO_krabu`.

Obrázek 4: Odkaz na GitHub repozitář.

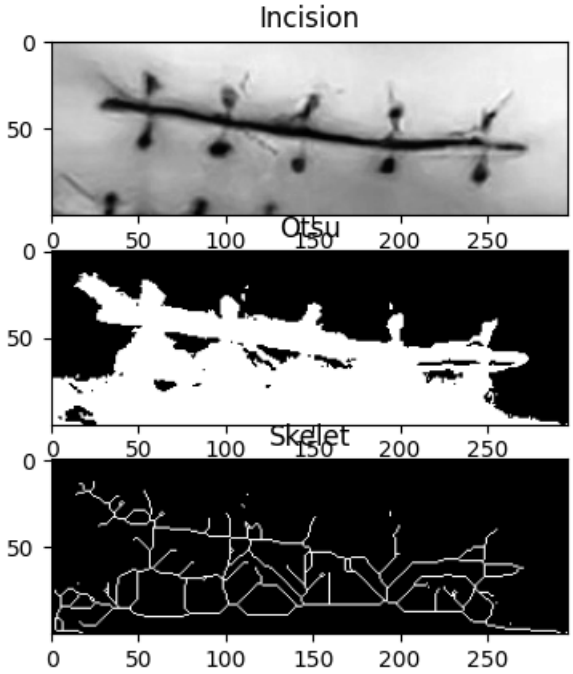
4.1 Prvotní návrh řešení (koncept)

Před samotným řešením této úlohy jsme nejprve zkusili vymyslet jeden z postupů, jakým by se tato úloha dala řešit. Nakonec jsem však zvolili jiné řešení. Přesto sem vkládáme i náš prvotní nápad na vyřešení úlohy.

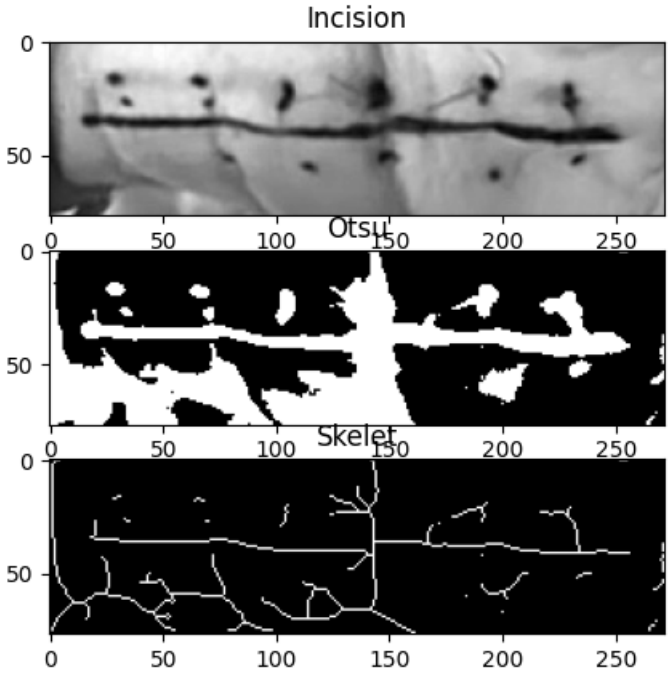
4.1.1 Předzpracování

Úplně nejdříve jsme si dané obrazy načetli v šedotónových barvách. Dále jsme načtené obrazy zkusili naprahovat. Pro co nejlepší určení hodnoty prahu jsme využili tzv. Otsuovu metodu, jejíž implementace je součástí knihovny scikit-image. Tímto jsme získali masku (binární obraz), ze které jsme následně získali skelet, použili jsme metodu *skeletonize()*, která je opět součástí modulu scikit-image. Tímto způsobem jsme získali zjednodušené tvary jizev, ale i stehů. V jednoduchých případech, kdy je v obraze jasně viditelná jizva a pozadí obrazu je v rámci možností jednolité, funguje tento postup vcelku spolehlivě. Problém nastane zejména u případů, kdy se v okolí kolem jizvy vyskytnou např. barevné "čáry" od fixy, stíny či struktura kůže, tím pádem dojde ke špatnému nastavení prahu a tedy i špatnému skeletu. Na získané vyprahované obrazy jsme se snažili aplikovat např. hranové detektory či Sobelův filtr, to však

vedlo k neuspokojivým výsledkům. Na následujících obrázcích jsou znázorněny příklady, u kterých dojde ke špatnému naprahování a následně ke špatné konstrukci skeletu:



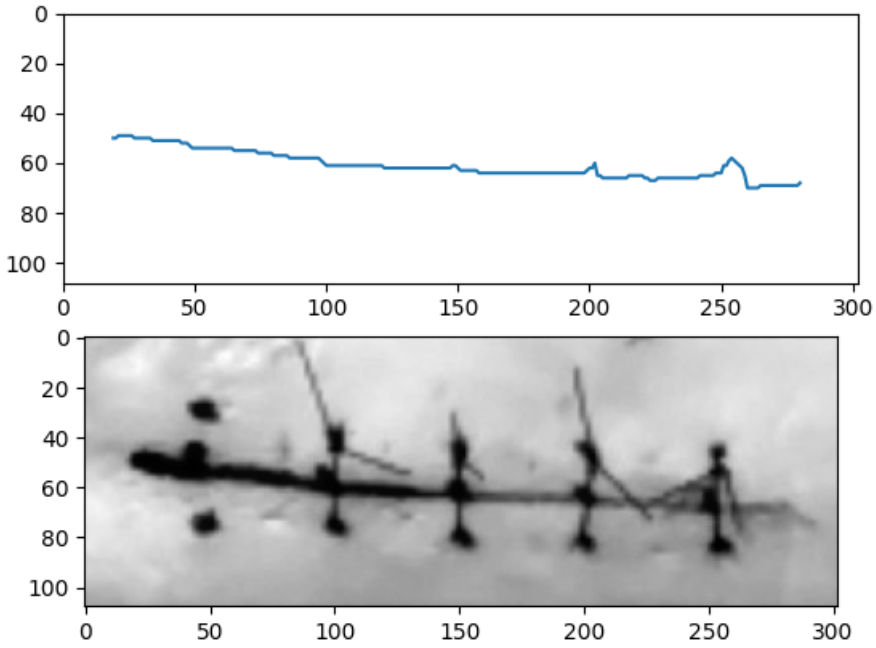
Obrázek 5: V pořadí ze shora: původní černobílý obraz; naprahovaný obraz a skelet.



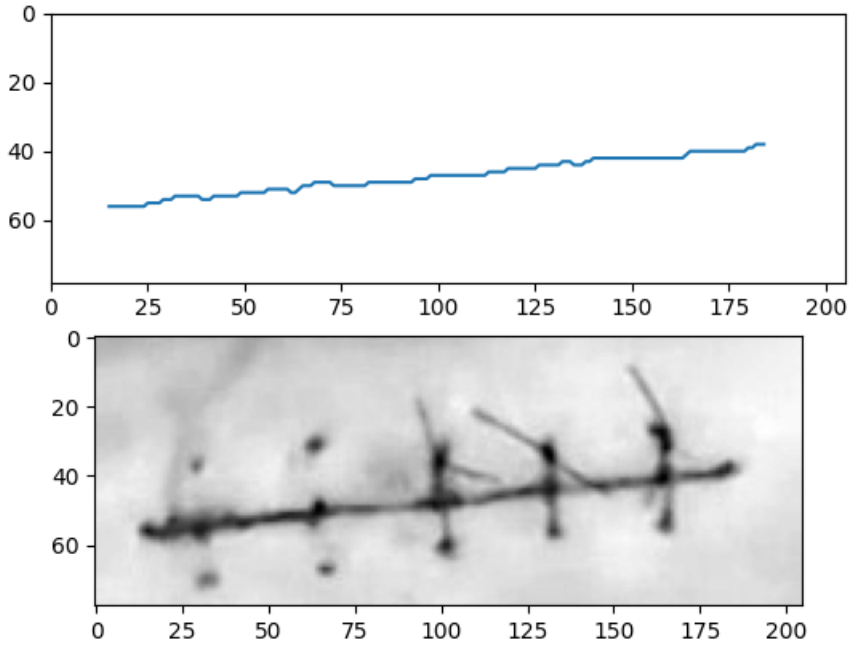
Obrázek 6: V pořadí ze shora: původní černobílý obraz; naprahovaný obraz a skelet.

4.1.2 Zpracování

V další části jsme se pokusili zpracovávat získané skelety. V tomto případě se jedná o úlohu nalezení cesty v matici, která obsahuje prvky 0 (False) a 1 (True). V našem algoritmu nejprve hledáme startovní bod (počátek jizvy). Po nalezení startovního bodu se pak v matici pohybujeme směrem zleva doprava. Kontrolujeme, zda se vedle startovního bodu nenachází prvek == 1 (True) přímo vedle vpravo, šikmo nahoře či šikmo dole. Takhle bychom chtěli pokračovat až dokud nenarazíme na poslední prvek. Samozřejmě se může stát, že daný skelet bude v různých směrech nespojitý, tím pádem je třeba nastavit různé tolerance. Pamatujeme si zároveň více potenciálních cest skrz matici a záchytné body, do kterých se algoritmus vrátí, pokud narazí na slepou větev. Jedná se o příklady, kdy se z "hlavní" cesty jizvy odbočí na steh, kde cesta dál nepokračuje, tak se vrátíme zpět na "rozcestí" a snažíme se pokračovat dál. Na obrázcích níže lze pozorovat správné výstupy algoritmu. Dokážeme tedy detekovat začátek a konec jizvy a vizuálně ověřit, zda jsou zhruba na správných místech.



Obrázek 7: V pořadí ze shora: detekovaná jizva; původní černobílý obrázek.

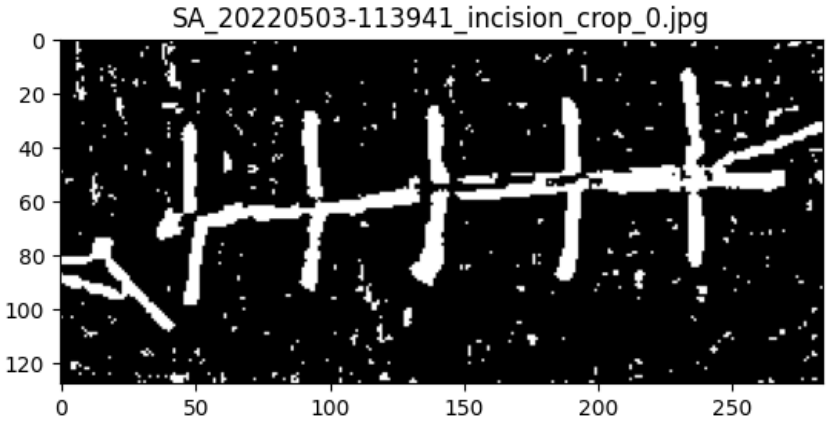


Obrázek 8: V pořadí ze shora: detekovaná jizva; původní černobílý obrázek.

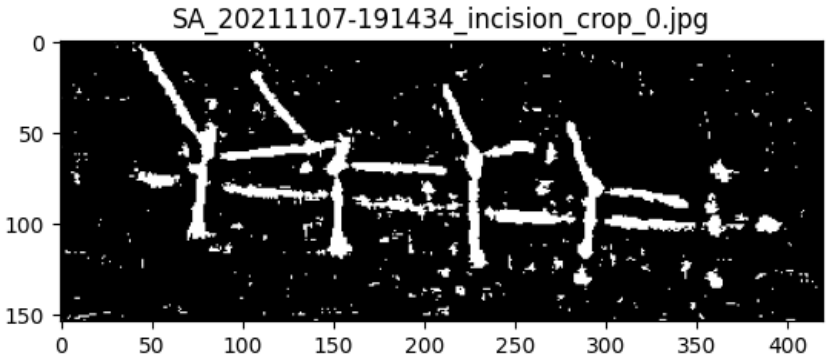
4.2 Finální řešení

4.2.1 Detekce jizev

V této fázi jsme nejprve provedli detekci jizev v jednotlivých obrazech. V případě detekce jizvy byl na vstupu černobílý obrázek. Na tento obraz byl následně aplikován upscaling ($cv2.resize()$), tzn. že tento obraz byl zvětšen/škálován a hodnoty jednotlivých pixelů byly dopočítány s využitím bikubické interpolace, jestliže byla překročena určitá prahová hodnota. V našem případě byl proveden upscaling obrazu, právě tehdy když byla jeho šířka menší než 200 pixelů. Tato hodnota byla určena experimentálně. Dále jsme obraz adaptivně naprahovali a to s využitím metody *cv2.adaptiveThreshold()*.



Obrázek 9: Výsledek adaptivního prahování.



Obrázek 10: Výsledek adaptivního prahování.

Dále jsme vyzkoušeli aplikaci hranových detektorů (Canny) jako přípravu pro další zpracování. Tento výstup však nakonec nebyl využit.

Na předzpracovaná data jsme následně zkusili aplikovat Houghovu transformaci. Pro tento účel jsme využili metodu *cv2.HoughLinesP()*, jejíž parametry byly určeny experimentálně. Metoda nám v mnoha případech vracela více než právě jednu detekovanou úsečku. Pokud ale metoda v obrazu nedetekovala ani jednu úsečku, pak byl na obraz znovu aplikován uspcaling společně s prahováním. Tento modifikovaný obraz sloužil následně opět jako vstup do metody *cv2.HoughLinesP()*. U detekovaných čar bylo dále kontrolováno, zda je jejich úhel vzhledem k horizontální ose menší než 30°, aby opravdu docházelo k detekci jizev a ne stehů (to až v následujících krocích).

Dále bylo třeba vyřešit případy, ve kterých bylo vygenerováno více úseček než právě jedna. To bylo vyřešeno zprůměrováním daných úseček, čímž jsme získali "referenční"úsečku. Následně jsme zjišťovali, zda se ostatní úsečky nachází v určitém pásmu referenční úsečky. Pokud ano, byly přidány do listu, který sloužil opět jako vstup do metody průměrování jednotlivých úseček, a tak byla vždy získána jen jedna úsečka reprezentující jizvu na daném obrázku. Zkoušeli jsme tento problém vyřešit i pomocí klasifikačního algoritmu k-means, to však nevedlo k dobrému výsledku.

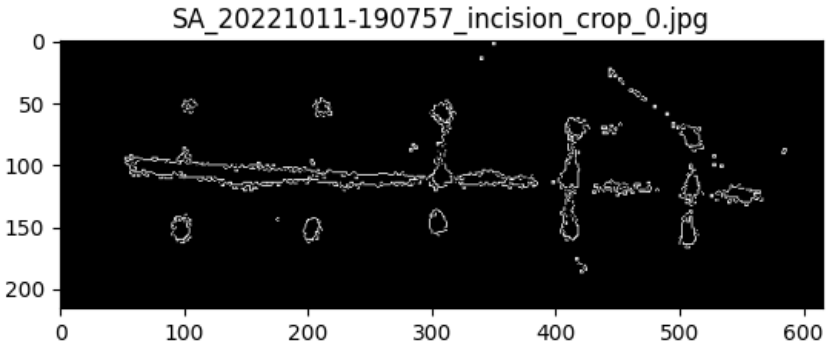
Detekce jizev fungovala v našem případě vcelku spolehlivě. Vždy velice záleželo na kvalitě zpracovávaného obrazu. Rovněž bylo velice důležité, aby vůbec samotná jizva byla v obrazu dobře viditelná, což bylo u některých obrazů trochu obtížnější. Námi implementovaná metoda

však měla problémy s detekcí jizev jiného tvaru než tvaru nějaké čáry/úsečky, např. ve tvaru "hokejky", atp. Toto lze brát jako jednu nevýhodu využití Houghovy transformace, jelikož tato transformace je vhodná především pro detekci čar v obrazu. Také se vyskytlo pár případů, kdy kromě detekce jizvy došlo k detekci jednotlivých teček (vyznačení vpichů) také jako jizvy. To způsobilo na některých obrázcích menší odchýlení výsledné detekované úsečky od reálné jizvy.

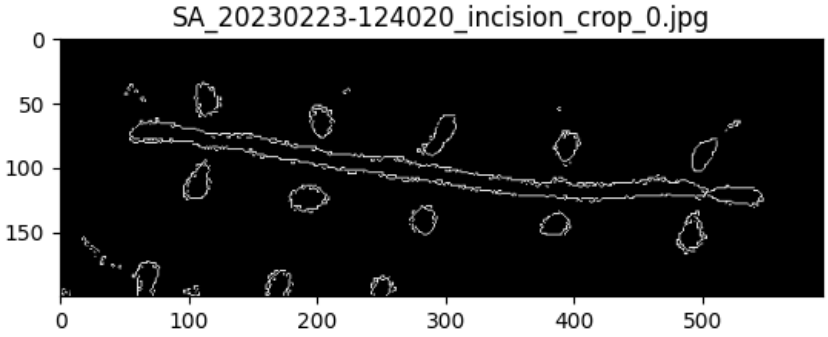
4.2.2 Detekce stehů

V další části jsme se věnovali detekci stehů v jednotlivých obrazech. Detekce stehů se nám jevila jako složitější než detekce jizev. Samotné stehy totiž mohou být různé (jde zejména o různé úhly vzhledem k jizvě). Dále mohou být problémy se zvrásněním kůže.

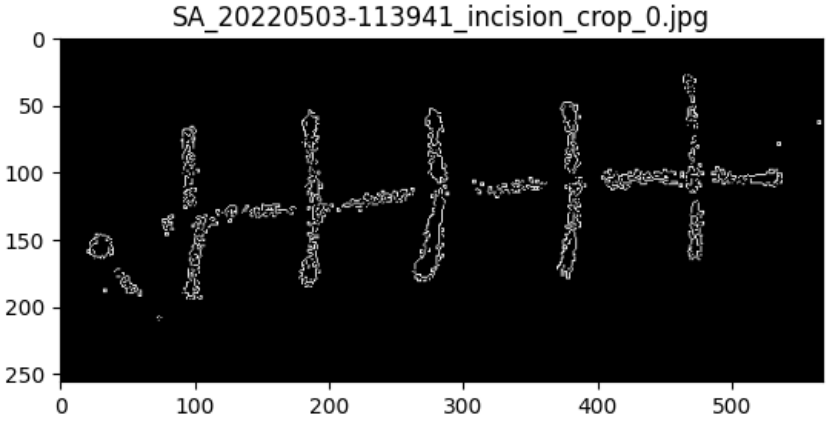
Na vstupu byl opět černobílý obrázek. Nejprve byl na tento obraz otestován totožný postup jako u detekce jizvy, nedosahovalo se však takových výsledků jako u detekce jizvy. Na načtený obraz bylo tedy nejprve aplikováno Gaussovské rozstrotření (z angl. *Gaussian blur*) s použitím metody *cv2.GaussianBlur()* za účelem filtrace šumu pro následné zpracování. Následně byl obraz škálován, tak jako v případě detekce jizvy, v tomto případě byl však snímek škálován vždy (bez závislosti na šířce obrazu), což u některých obrázků (především malé velikosti) vedlo k velkým zlepšením, co se týče kvality. V dalším kroku jsme na obraz aplikovali adaptivní prahování (opět metoda *cv2.adaptiveThreshold()*). Dále jsme použili, oproti detekci jizvy, Cannyho operátor (operátor pro detekci hran) s definovanými mezemi prahu spočítanými jako násobky průměrné hodnoty adaptivního prahu. Tento Cannyho operátor poskytl následující výsledky:



Obrázek 11: Výsledek po aplikaci Cannyho operátoru.



Obrázek 12: Výsledek po aplikaci Cannyho operátoru.

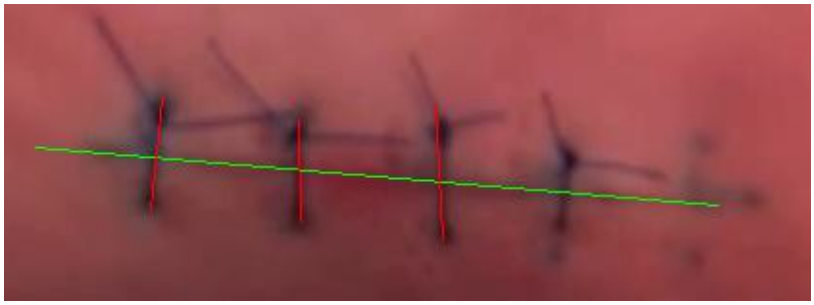


Obrázek 13: Výsledek po aplikaci Cannyho operátoru.

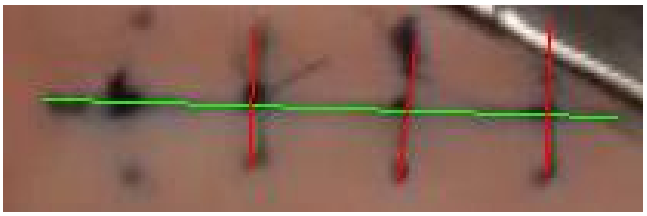
Následně jsme už opět na předzpracovaný obraz aplikovali Houghovu transformaci. Parametry Houghovy transformace byly opět nalezeny experimentálně vzhledem k našim datům. Dále byly kontrolovány úhly jednotlivých detekovaných úseček/čar. Pro případ stehů se zvolila dolní mez úhlu na hodnotu 60°. V případě, kdy při takovémto nastavení nedošlo k detekci žádné úsečky, snížila se hraniční hodnota na 20°, což pomohlo v případech, kdy byly stehy

"šikmo"vůči jizvě. Opět bylo třeba řešit případy, ve kterých bylo detekovaných čar více než byl skutečný počet stehů v obrazu nebo pokud detekované čáry zcela neodpovídaly stehům. Tento problém byl řešen s využitím algoritmu k-means, do kterého vstupovaly jako příznaky body (x,y) počátků detekovaných úseček reprezentující jednotlivé stehy. V k-means metodě jsme zjišťovali, pro který počet tříd (počet stehů) dosáhne kriteriální funkce nejvyššího skóre a následně jsme jednotlivé úsečky rozdělili do daných tříd a úsečky v jednotlivých třídách opět zprůměrovali a získali jsme tak opět pro jeden detekovaný steh jen jednu úsečku.

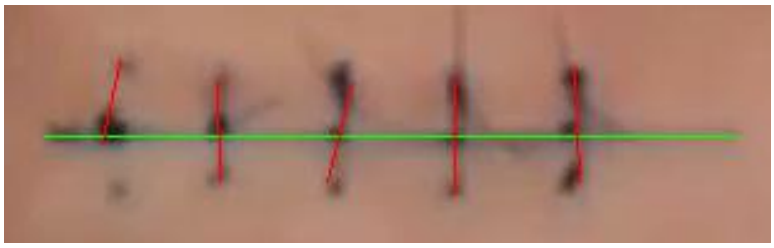
Na detekci stehů byly rovněž vyzkoušeny morfologické operace na spojení v obraze nakreslených "teček", kudy má procházet steh. Vyzkoušen byl rovněž i skelet. Dále jsme se pokusili použít color-based přístup, kde byla snaha o využití barevnosti a textury jizev a odlišit je tak od okolí. Tento postup byl však neúspěšný. Na závěr jsme vyzkoušeli zlepšit ostrost obrazu pomocí "Unsharp masking technique", která obecně vylepšuje okraje a detaily v obraze odečtením rozmazané verze obrazu od původního obrazu. Níže je zobrazeno několik získaných detekcí jizev a stehů:



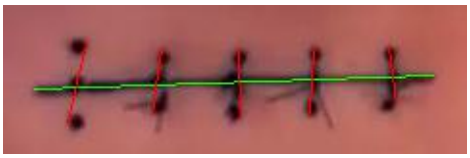
Obrázek 14: Detekovaná jizva (zeleně) a detekované stehy (červeně).



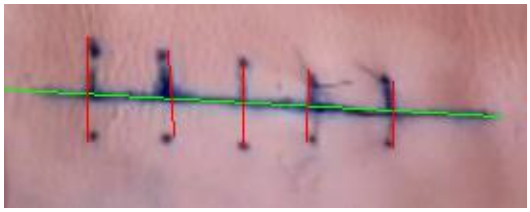
Obrázek 15: Detekovaná jizva (zeleně) a detekované stehy (červeně).



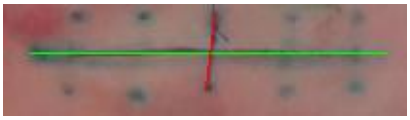
Obrázek 16: Detekovaná jizva (zeleně) a detekované stehy (červeně).



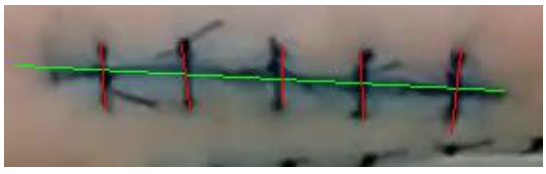
Obrázek 17: Detekovaná jizva (zeleně) a detekované stehy (červeně).



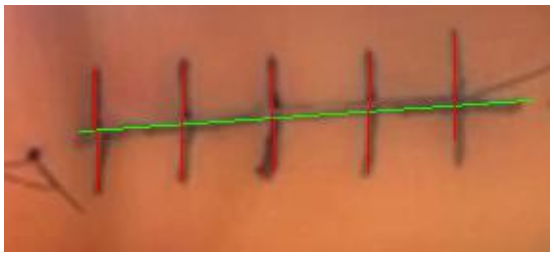
Obrázek 18: Detekovaná jizva (zeleně) a detekované stehy (červeně).



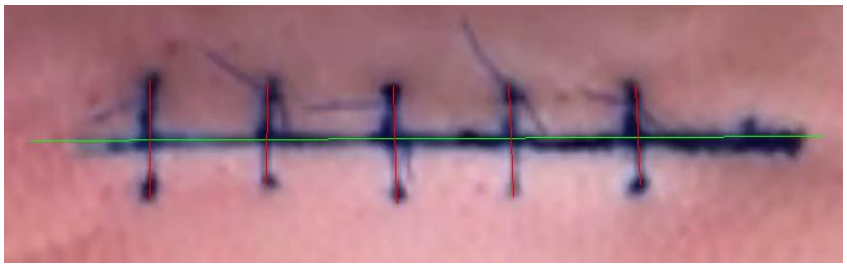
Obrázek 19: Detekovaná jizva (zeleně) a detekované stehy (červeně).



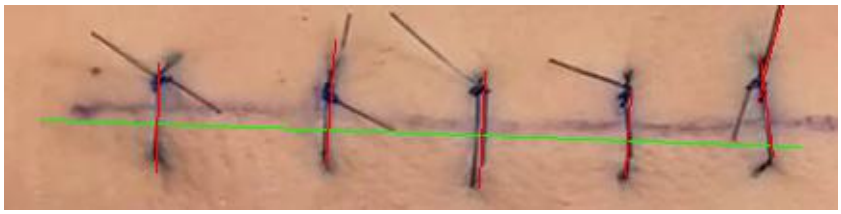
Obrázek 20: Detekovaná jizva (zeleně) a detekované stehy (červeně).



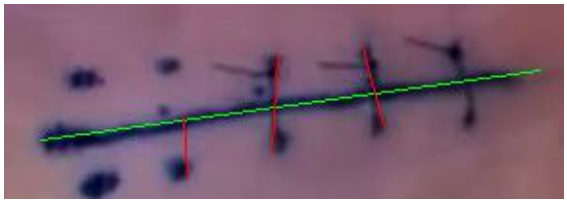
Obrázek 21: Detekovaná jizva (zeleně) a detekované stehy (červeně).



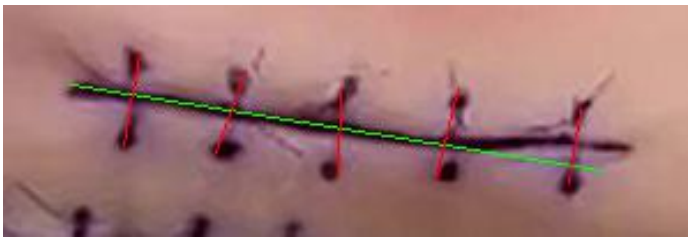
Obrázek 22: Detekovaná jizva (zeleně) a detekované stehy (červeně).



Obrázek 23: Detekovaná jizva (zeleně) a detekované stehy (červeně).



Obrázek 24: Detekovaná jizva (zeleně) a detekované stehy (červeně).

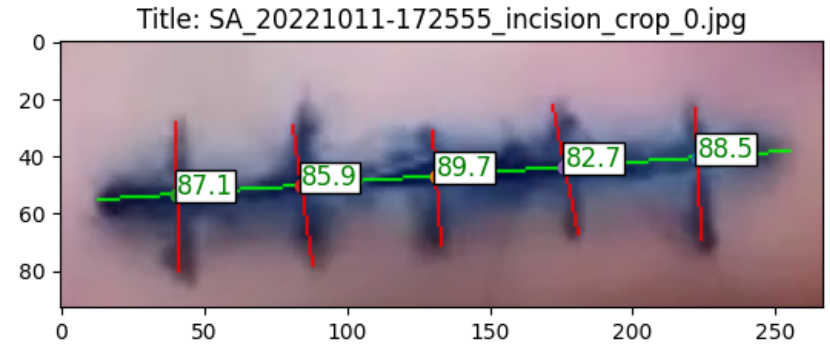


Obrázek 25: Detekovaná jizva (zeleně) a detekované stehy (červeně).

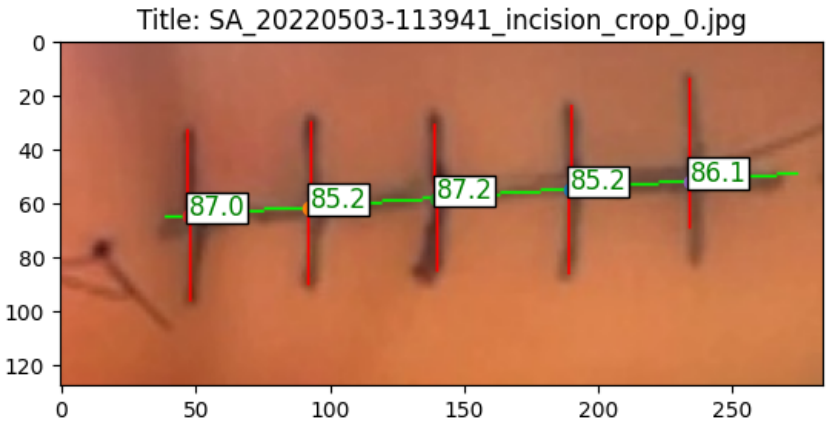
4.2.3 Výpočet průsečíků a úhlů

Na výsledky z předchozí části jsme aplikovali algoritmy poskytnuté v rámci zadání semestrální práce, které vypočítaly průsečíky stehů a jizev a příslušné úhly mezi nimi. Na následujících obrázcích jsou vidět dosažené výsledky.

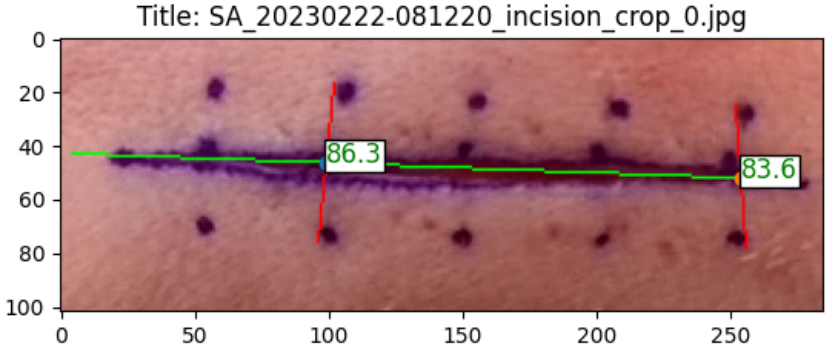
Obrázek 26: Detekovaná jizva (zeleně) a detekované stehy (červeně) včetně vypočítaných úhlů.



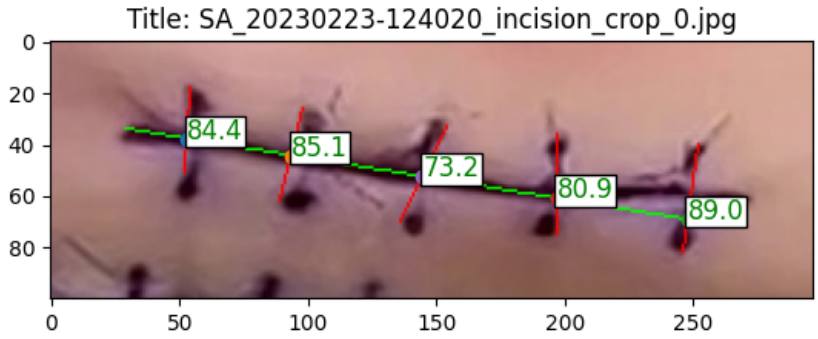
Obrázek 27: Detekovaná jizva (zeleně) a detekované stehy (červeně) včetně vypočítaných úhlů.



Obrázek 28: Detekovaná jizva (zeleně) a detekované stehy (červeně) včetně vypočítaných úhlů.



Obrázek 29: Detekovaná jizva (zeleně) a detekované stehy (červeně) včetně vypočítaných úhlů.



Obrázek 30: Detekovaná jizva (zeleně) a detekované stehy (červeně) včetně vypočítaných úhlů.

Obrázek 31: Detekovaná jizva (zeleně) a detekované stehy (červeně) včetně vypočítaných úhlů.

5 Spuštění programu

Pro spuštění programu je potřeba vstoupit do adresáře */src* daného GitHub repozitáře , který byl již zmíněn v úvodu tohoto dokumentu. Je také nutné nadefinovat vstupní argumenty. Jedním ze způsobů je využít konfigurace v IDE pro daný spouštěcí skript, což je *run.py*. Druhým způsobem je spuštění z příkazové řádky pomocí jednoho z následujících příkazů:

```
python run.py output.json incision001.jpg incision005.png

python run.py –v output.json incision001.jpg incision005.png
```

Záleží na tom, zda chce uživatel výsledky i zobrazit. Vstupní argumenty jsou tedy:

1. Název výstupního *.json* souboru
2. Verbose mode (nepovinný) (-v)
3. Názvy jednotlivých obrázku na zpracování

Všechny obrázky se také nacházejí v repozitáři a proto je možné jako argument zadat přímo názvy daných obrázků pro zpracování. Mimo volitelný vizuální výstup (zobrazení detekovaných jizev a stehů a úhlů mezi nimi) je výstupem také *.json* soubor, který nese pro každý zpracovaný (vstupní) obrázek jeho název, počátek a konec detekované jizvy, souřadnice průsečíků stehů a jizvy a vypočítané úhly mezi nimi. Příklad výstupního *.json* souboru:

```
[
    {
        "filename": "SA_20211107-191434_incision_crop_0.jpg",
        "incision_polyline": [
            16,
            74,
            371,
            104
        ],
        "crossing_positions": [
            [
                226.31,
                91.77
            ],
            [
                153.49,
                85.62
            ],
            [
                78.76,
                79.3
            ]
        ],
        "crossing_angles": [
            81.85,
            84.07,
            88.92
        ]
    }
]
```

6 Zhodnocení řešení

Detekce jizev v jednotlivých obrazech fungovala vcelku spolehlivě, pouze v jednom případě se stalo, že se obraze nenašla žádná čára, pomocí Houghovy transformace, která by reprezentovala jizvu. Také se někdy stalo, že výsledná čára byla v obraze výše či níže než se očekávalo, to bylo většinou způsobeno velkou hustotou "teček", znázorňujících začátky a konce stehů. U některých případů rovněž došlo k tomu, že detekovaná čára byla delší či kratší než skutečná jizva, tento jev nastával především u obrazů s horšími světelnými podmínkami (v obrazu byl např. viditelný stín).

Naopak u detekce stehů jsme dosáhli horších výsledků. Hodně záleželo na samotné viditelnosti stehů v obraze a rovněž na světelných podmínkách. Rovněž bylo složité určit hodnoty parametrů Houghovy transformace z důvodu výskytu většího množství druhů stehů v trénovacích datech, tudíž z tohoto hlediska by bylo lepší vytvořit nástroj pro detekci pouze jednoho druhu stehu.

Obecně lze konstatovat, že v případě lepší kvality obrazových dat, by námi navržený algoritmus na detekci jizev a stehů dosáhl zřejmě znatelně lepších výsledků.

7 Závěr

V této semestrální práci jsme měli možnost si prakticky vyzkoušet postupy a metody počítačového vidění představené z velké části na cvičeních a přednáškách z předmětu ZDO. Cílem této práce byla detekce jizev a stehů v poskytnutých medicínských obrazových datech. Nejprve jsme na data zkusili aplikovat především základní metody počítačového vidění. Zaměřili jsme se nejdříve na detekci jizvy.

Data jsme nejprve naprahovali, čímž jsme dostali masku, ze které jsme následně získali skelet. U případů, ve kterých byl jizva jasně viditelná fungoval tento postup vcelku spolehlivě. Bohužel, jelikož ve většině obrazů nebyla jizva až tak viditelná, byli jsme nuceni zvolit jiný postup.

Jako finální řešení jsme zvolili použití Houghovy transformace, pomocí které jsme detekovali jizvy i stehy. Parametry Houghovy transformace byly v obou případech zvoleny experimentálně s přihlédnutím k trénovacím datům. Detekce jizev fungovala pomocí tohoto postupu velice dobře, naopak u detekce stehů byly dosažené výsledky horší, než bylo očekáváno, což byl zřejmě důsledek špatné kvality trénovacích obrazových dat.