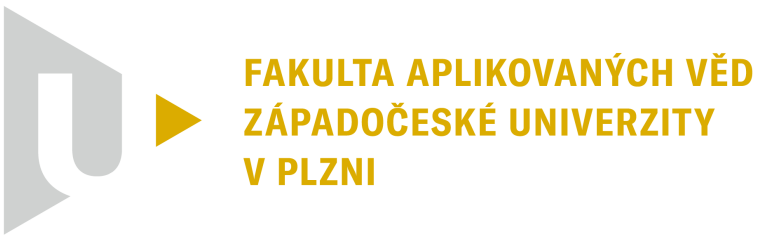
Vyhodnocení kvality zašití rány

Katedra kybernetiky

KKY/ZDO

Tomáš Hefler

Jakub Honzík



Fakulta aplikovaných věd Západočeské univerzity

27. května 2023

1 Zadání

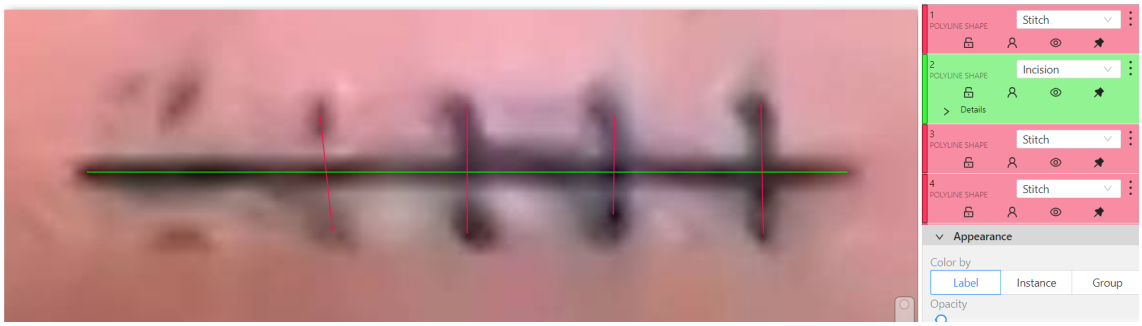
Zadáním semestrální práce je tvorba programu, který bude schopen vyhodnotit kvalitu stehů z jednotlivých fotek nácviku zašívání ran na prasečích nohách. Pomocí metod počítačového vidění navrhneme postup detekce jednotlivých stehů a vyhodnocení kvality vybraných charakteristik.

2 Vypracování

Semestrální práci jsme vypracovali v programovacím jazyce Python za použití knihoven scipy a skimage. Vypracování je dostupné na `https://github.com/Hefik/ZDO_hef_hon/tree/main` .

2.1 Anotace dat

Prvním úkolem naší semestrální práce bylo anotování dat pomocí programu CVAT. Každý z nás dostal několik obrázků, ve kterých pomocí anotačního programu vyznačil jizvu/ránu a stehy. Bylo nutné přihlížet ke kvalitě jednotlivých obrázků, protože v některých případech byly švy téměř nerozeznatelné kvůli horší kvalitě obrázků.

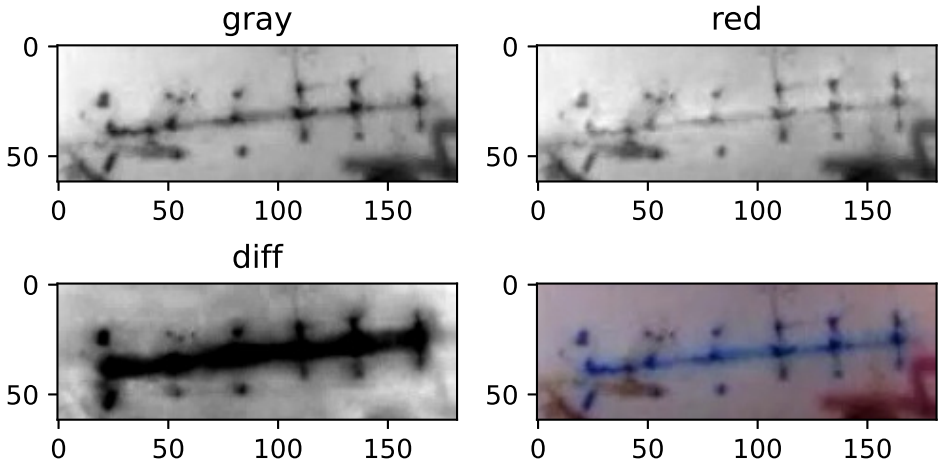


Obrázek 1: Anotace obrázků

2.2 Úprava obrázků

Metody, které jsme využívali pracují s 2D obrázky, nikoliv s RGB 3D obrázky. Z RGB obrázků jsme chtěli získat co nejvíce informace, proto jsme zkoušeli různé

způsoby převodu z RGB obrazu na 2D obraz. Jako první jsme zkoušeli běžně používanou metodu rgb2gray, která na každý pixel ukládá jas zprůměrovaný ze všech tří barev. Dále jsme zkoušeli metodu na základě pozorování, že mají rány a stehy často barvu do modra, zatímco prasečí kůže je více do červena, rány a stehy jsou také tmavší než okolí. Na základě tohoto pozorování jsme zkoušeli místo jasu využít pouze červenou složku spektra a také kombinace červené a modré. Od červené jsme odečítaly modrou barvu, která je více hojná v tmavých ranách, takže jsou stehy více tmavé a okolí světlejší. Namísto hodnot menších než nula jsme dosazovali nulu a modré jsme odečítali zmírněnou hodnotu, aby nedocházelo k moc velkému začernění obrazu.

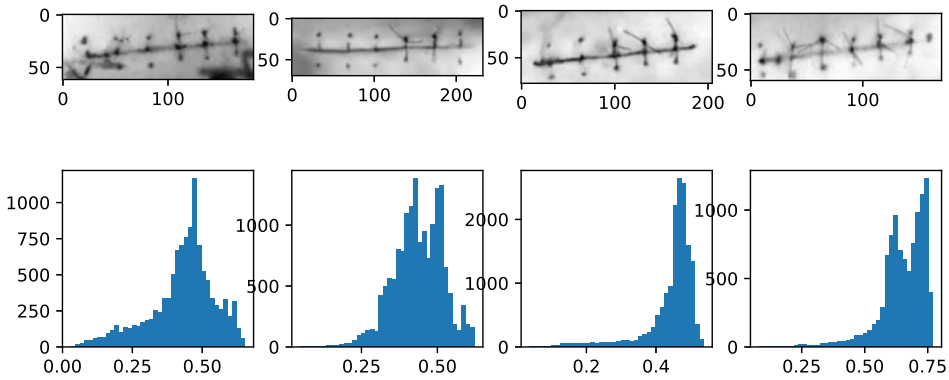


Obrázek 2: úpravy obrázku, vlevo nahoře jas, vpravo nahoře červená barva, vlevo dole obraz po rozdílové úpravě, vpravo dole původní obrázek

Na obrázku je vidět, že po rozdílové úpravě mizí červené fleky po stranách a rány jsou více výrazné. Přechody mezi tmavou a světlou částí jsou ale méně strmé a větší šířka jizvy a stehů limituje možnou přesnost v detekci. Nakonec jsme zůstali u jasu, který je spolehlivý i pro obrázky, které nemají diskutované barevné vlastnosti, kterých rozdílová úprava využívá. Červené fleky se v datech moc často nevyskytují a nejsou příliž problematické.

2.3 Prahování

Pro segmentaci jsme nejprve chtěli zkusit prahování. Po vykreslení histogramů jsme ale zjistily, že z nich moc nejde odhadnout hodnotu prahu.

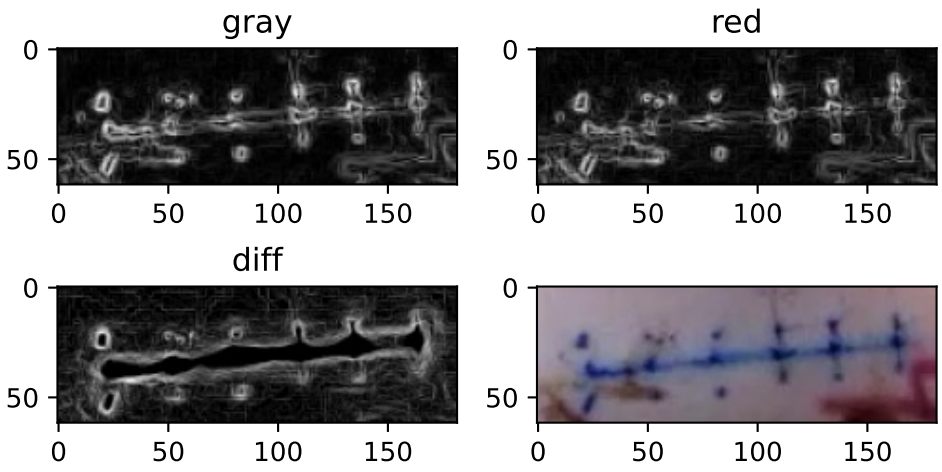


Obrázek 3: histogramy

Obrazy jsou dost zašuměné a histogramy odpovídají směsi více gaussovských šumů.

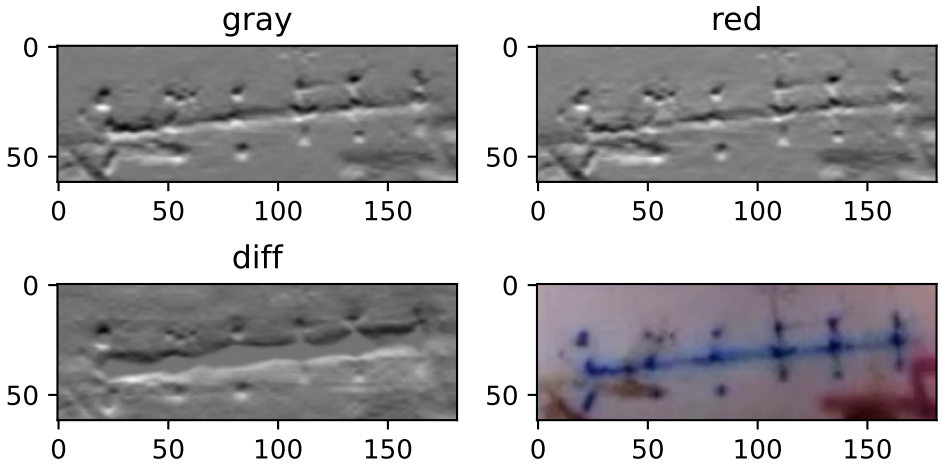
2.4 Hranové detekce a segmentace

Vyzkoušeli jsme více hranových detektorů. Nejprve gradientní operátory Roberts a Prewitt. Pro upravenou verzi obrázku 'diff' jsme museli používat jiné parametry kvůli většímu množství šumu v této verzi.

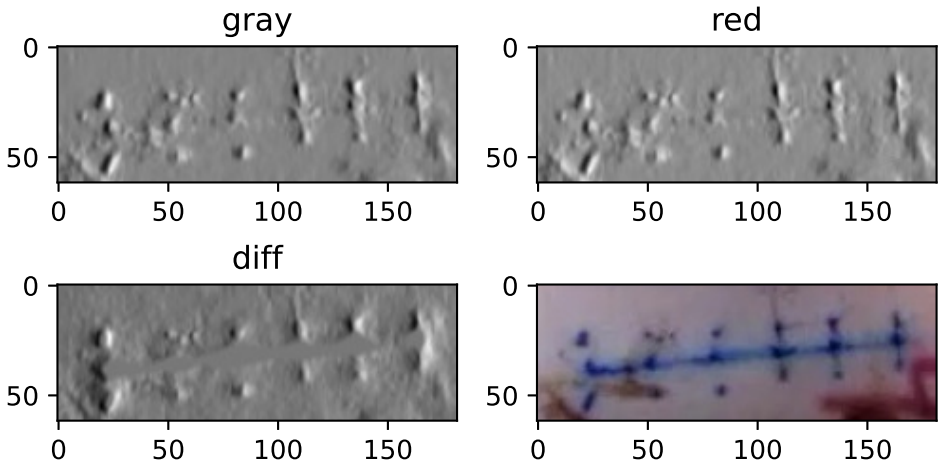


Obrázek 4: Detekce hran pomocí masky Roberts

Masku Prewitt jsme využily zvlášť pro detekci vodorovných hran na detekci jizvy a zvlášť pro detekci svislých hran pro detekci stehů.

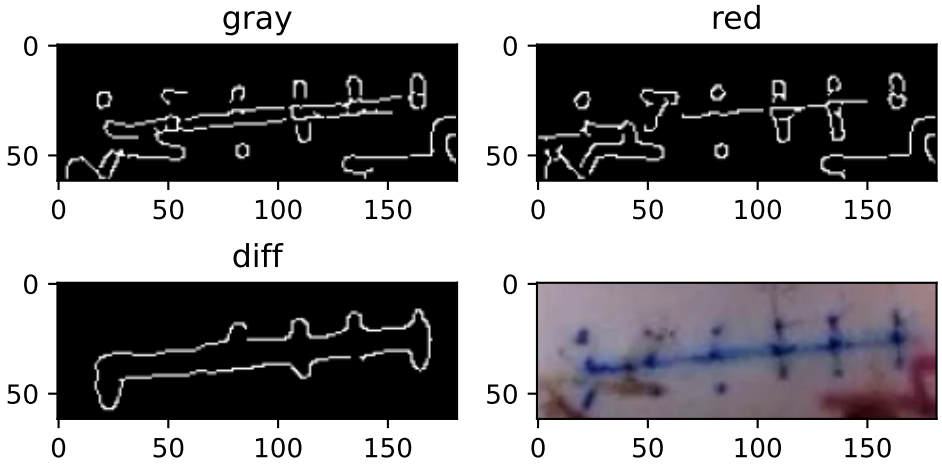


Obrázek 5: Detekce hran pomocí masky Prewitt pro detekci vodorovných hran



Obrázek 6: Detekce hran pomocí masky Prewitt pro detekci svislých hran

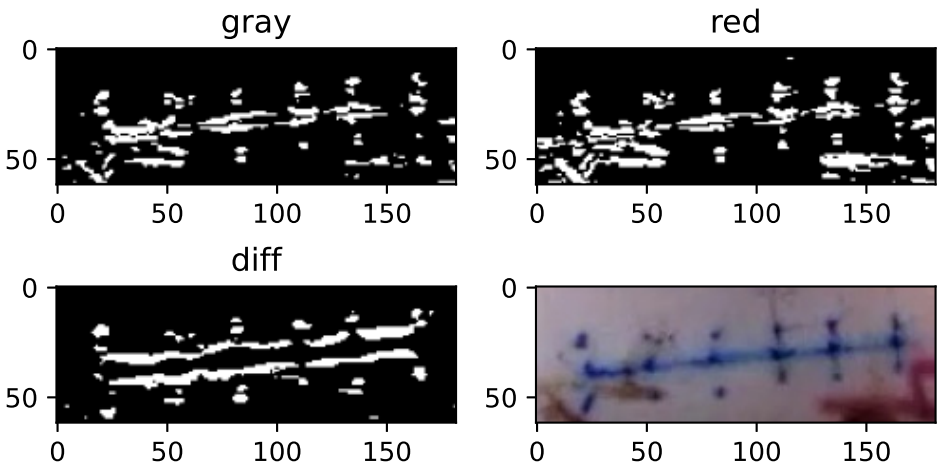
Také jsme vyzkoušeli operátor Canny, který rovnou rozdělil obraz do binárních hodnot.



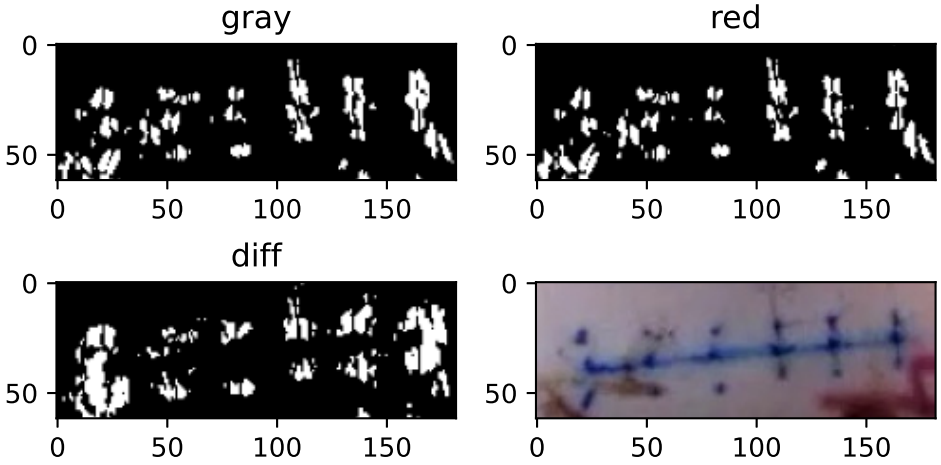
Obrázek 7: Detekce hran pomocí operátoru Canny

Výstup z operátoru Prewitt nabývá v místech bez hran hodnoty blízké nule a na hranách hodnoty vzdálenější od nuly na obě strany závisle na směru. Aby byly hrany z obou stran stejné, provedli jsme prahování podle vzdálenosti od nuly. Prahování jsme udělali adaptivní na jakýkoliv obrázek tak, aby byl poměr jdeniček a nul u všech obrázků zhruba stejný. Tento poměr se nastavuje parametrem intensity, který odpovídá počtu jedniček v binárním obrázku vydělený počtem pixelů v obrázku.

Experimentálně jsme došli k vyhovující intenzitě okolo 0.1.



Obrázek 8: Propojený Prewitt pro detekci vodorovných hran



Obrázek 9: Propojený Prewitt pro detekci svislých hran

Prahovaný Prewitt jsme dále využívali pro detekci čar.

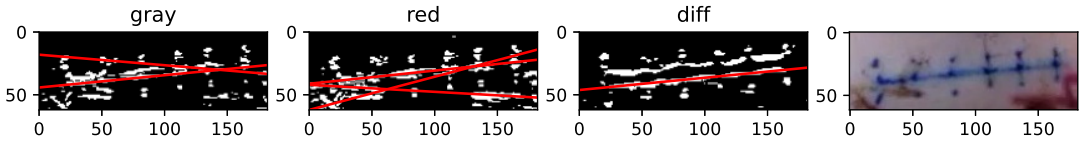
2.5 Detekce rovných čar

Po předzpracování obrázků pomocí prahování a hranových detektorů bylo třeba najít jednotlivé stehy a jizvu. Jelikož stehy by měli být na obrázcích vidět jako svislé

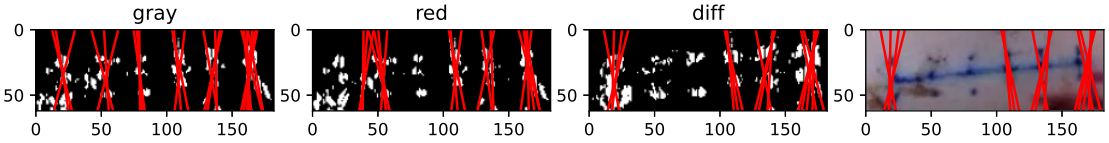
rovné čáry a jizva podle většiny obrázků byla jedna vodorovná čára, rozhodli jsme se využít detektor rovných linií pomocí Houghovy transformace.

2.5.1 Detekce pomocí hough_line_peaks()

Nejprve jsme vyzkoušeli metodu hough_line_peaks() z knihovny skimage. Tato metoda hledá rovné linie v zadaném rozmezí úhlů, tímto způsobem jsme mohli rozlišit detekci vodorovné jizvy od svislých stehů. Pro tuto metodu se nám nejvíce osvědčili výstupy hranových detektorů z obrazu po rozdílové úpravě.



Obrázek 10: Detekce jizvy pomocí metody hough_line_peaks()



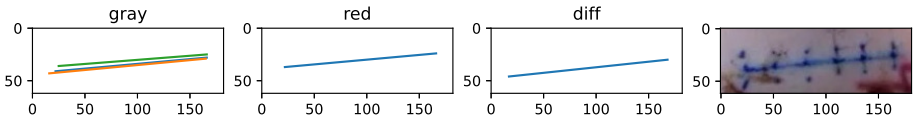
Obrázek 11: Detekce stehů pomocí metody hough_line_peaks()

Tyto obrázky zobrazují nalezené jizvy a stehy. V pozadí je výstupní obrázek z hranové detekce. Je vidět, že detektor nalezl jizvu i několik linií stehů, Ovšem každému stehu náleží hned několik přímek. Jako hlavní nevýhoda tohoto postupu se ukázal fakt, že tato metoda nedokáže nají úsečky, které by svojí délkou odpovídaly hledanému objektu. Z tohoto důvodu jsme se rozhodli tuto metodu pro finální řešení nevyužít.

2.5.2 Detekce pomocí probabilistic_hough_line()

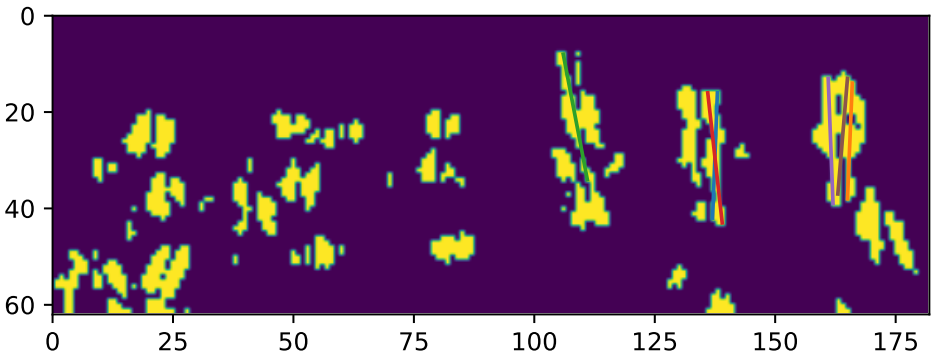
Stejně jako předchozí metoda i tato využívá Houghovu transformaci pro hledání rovných čar v obrázku. Obdobně jako hough_line_peaks() umožňuje hledání v zadaném rozmezí úhlů, kterými jsou úsečky v prostoru natočeny, ale navíc přijímá

další tři parametry. Tyto parametry nám umožnily blíže specifikovat hledané linie. Jednou z největších výhod této metody se ukázala skutečnost, že hledá úsečky, které odpovídají rozměrem i natočením hledaným čarám. Pro tuto metodu se nám nejvíce osvědčili výstupy hranových detektorů z šedotónového obrazu.



Obrázek 12: Detekce jizvy pomocí metody probabilistic_hough_line()

Na obrázku je ukázka detekování jizvy. Jelikož jsme předpokládali, že hranové detektory vytvoří pro jizvu dvě hrany, jednu horní a jednu spodní, požadovali jsme, aby pro šedotónový obrázek byly nalezeny alespoň dvě úsečky. Tohoto jsme docílili postupným iterativním upravováním parametru, který určuje minimální délku hledané úsečky.



Obrázek 13: Detekce stehů pomocí metody probabilistic_hough_line()

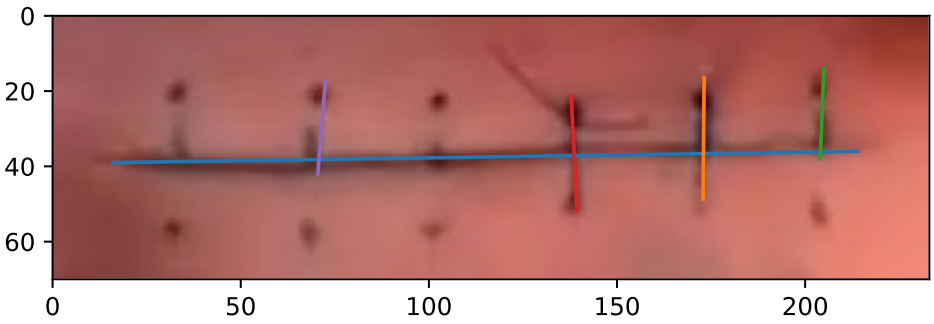
Na tomto obrázku je ukázka detekce stehů, na pozadí je výstup z hranové detekce, ve kterém jsou úsečky hledány. Stejně jako u hough_line_peaks() je vidět, že pro některé stehy bylo nalezeno hned několik úseček.

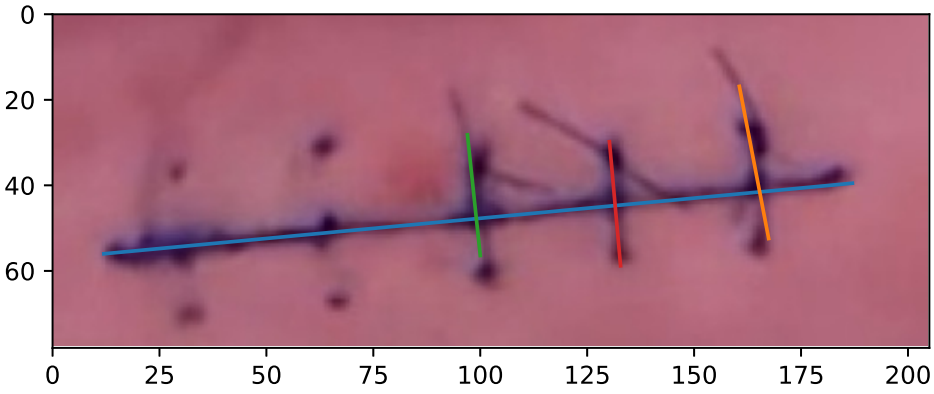
2.6 Úpravy nalezených čar

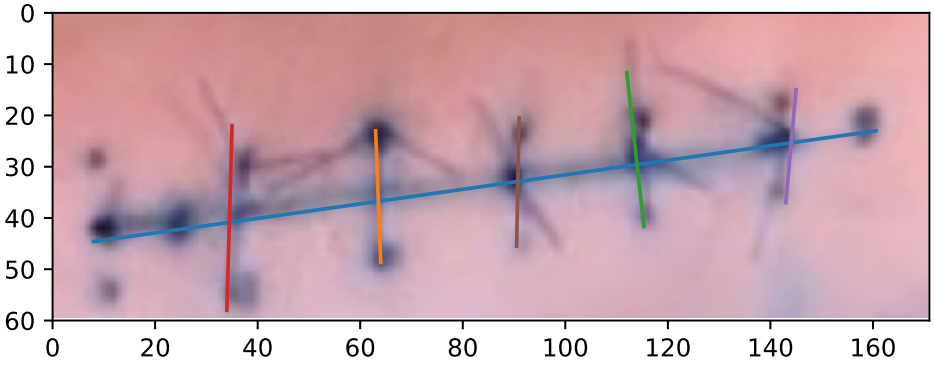
Jak bylo ukázáno v předchozí části práce, po aplikaci detektorů čar jsme zpravidla získali pro každý z hledaných objektů hned několik úseček. Bylo tedy nutné upravit nalezené linky. Pro úpravu stehů jsme vytvořili postup, který nejprve roztřídí úsečky do tříd, podle toho, ke kterému stehu by měli náležet Tento algoritmus funguje na základě vzorových úseček a prahové hodnoty. Nejprve se zvolí první vzorová úsečka, která vytvoří první třídu. Následně se prochází všechny ostatní úsečky. Pokud je vzdálenost jejích okrajových bodů od okrajových bodů vzorů všech tříd větší než zadaný práh, vznikne nová třída s touto úsečkou jako vzorem. V opačném případě je úsečka přiřazena do třídy do které má vzdálenost menší než je práh. Jelikož nelze dobře rozlišovat podle souřadnice svislé osy, rozhodli jsme se využívat

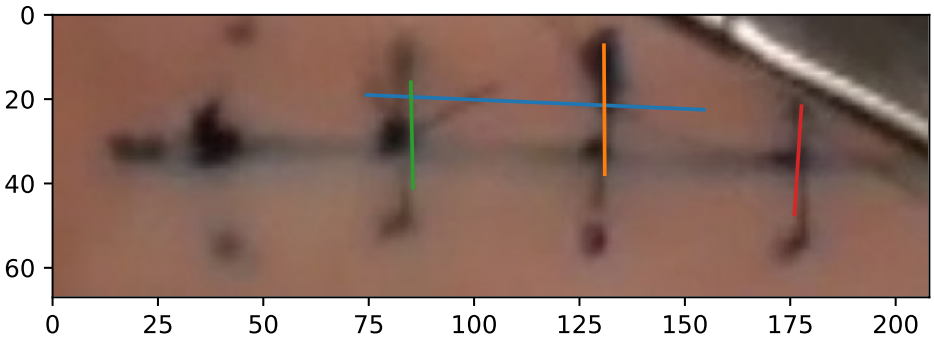
pouze jejich souřadnice ve vodorovné ose. Následně se vypočte aritmetický průměr ze všech úseček v každé ze tříd a vznikne jedna finální úsečka, která reprezentuje nalezený steh.

V případě úpravy nalezených úseček pro jizvy jsme využili stejnou metodu, jako pro stehy, jen jsme místo souřadnic ve vodorovné ose posuzovali ty ve svislé. Zjistili jsme, že po aplikaci této metody nemusí vždy dojít ke vzniku pouze jedné úsečky reprezentující jizvu. Například když došlo k nalezení úseček, které byly od sebe příliš daleko z důvodu nerovnosti jizvy, byly výsledkem dvě nebo více na sebe navazujících úseček, jejichž konce většinou neodpovídaly skutečné jizvě. Tento problém jsme se rozhodli vyřešit pomocí lineární regrese. Proložili jsme všechny body získané z detekce jizev přímkou, tak že celková odchylka těchto bodů od přímky byla minimální. Z takto získané přímky jsme získali jednu úsečku tak, že jsme našli z původních bodů nejnižší a nejvyšší hodnotu souřadnice na vodorovné ose. Tyto body jsme dosadili do naší přímky a získali finální úsečku reprezentující jizvu.




0
20
40
60
0
25
50
75
100
125
150
175
200


0
10
20
30
40
50
60
0
20
40
60
80
100
120
140
160



Obrázek 14: Ukázka nalezených stehů a jizev na původních obrázcích

Na obrázcích reprezentuje jizvy úsečka modré barvy, ostatní úsečky představují nalezené stehy. Na posledním obrázku je ukázka chybného nalezení jizvy, která následně povede k nepřesnosti výsledků.

2.7 Vyhodnocení kvality zašití rány

Cílem této semestrální práce nebylo pouze nalézt jednotlivé jizvy a stehy, ale hlavně vyhodnotit kvalitu zašití rány z barevného obrázku. Zvolili jsme dva hodnotící parametry: úhel a vzdálenost. Hodnotili jsme pouze detekované stehy, které mají průsečík s jizvou.

Prvním je úhel mezi stehy a jizvou, podle dostupných informací by při použití této techniky zašívání měli stehy s jizvou svírat pravý úhel. Tento úhel jsme určili ze směrového vektoru daného stehu a jizvy.

Druhý parametr, vzdálenost, reprezentuje rozestupy mezi mezi stehy. Stehy by měli být přibližně stejně daleko od sebe a krajní stehy by měli být dostatečně blízko začátku a konci rány. Tuto vzdálenost jsme pro každý steh určovali jako vzdálenost

jeho průsečíku s jizvou od levého kraje jizvy.

2.8 Uložení Výsledků

Výstupem našeho programu je soubor ve formátu JSON, jehož název udává uživatel, jako parametr při spouštění programu. Do tohoto souboru se pro každý ze zpracovávaných obrázků zapíše pro každý z obrázků okrajové body úsečky jizvy, vzdálenosti mezi průsečíky stehů a jizvy od začátku jizvy srovnané od nejnižší po nejvyšší a úhly mezi stehy a jizvou srovnané ve stejném pořadí jako vzdálenosti. Následující kód je ukázkou výstupu programu při analýze jednoho obrázku.

```
[
    [
        {
            "filename": "incision001.jpg",
            "incision_polyline": [
                [
                    [
                        20.0,
                        40.0
                    ],
                    [
                        166.0,
                        26.5
                    ]
                ]
            ],
            "crossing_positions": [
                89.94716132658796,
                117.5,
                144.17595363640683
            ],
            "crossing_angles": [
                94.17944665934945,
                84.71712445132383,
                85.51284800406312
            ]
        }
    ]
]
```

3 Závěr

V této semestrální práci jsme vytvořili program v jazyce Python, který dokáže v zadaných obrázcích z nácviku zašívání ran najít jizvu a stehy a poskytnout hodnocení jednotlivých stehů.

Vyzkoušeli jsme různé verze získání monochromatického obrázku a zjistili jsme, že nejlépe funguje program s klasickým šedotónovým obrázkem. Využili jsme několik hranových detektorů a jejich výstupy jsme následně prahovali. V získaných obrázcích obsahujících výraznější hrany jsme následně vyhledávali vodorovné a svislé úsečky. Z nalezených úseček jsme následně pomocí průměrování a lineární regrese určili úsečky jizev a stehů. Z těchto nalezených úseček jsme vyhodnotili kvalitu zašití rány pomocí úhlů mezi stehy a jizvou a vzdáleností mezi stehy a jizvou. Výsledky programu se zapíší do výstupního souboru.