Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra kybernetiky
Zpracování digitalizovaného obrazu

Referát k semestrální práci

Ondřej Valtr
studijní číslo: *A23N0061P*
ID „týmu“: *8*

letní semestr 2024
poslední úprava: *9. 5.*

Obsah

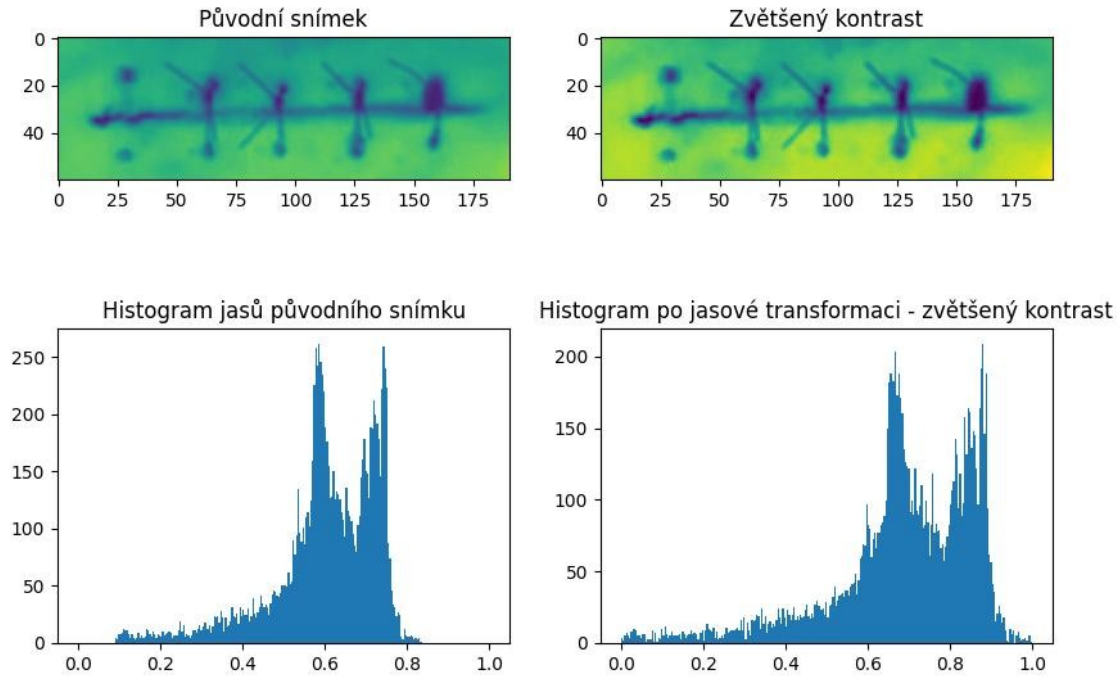
Návod pro úspěšné spuštění programu

1. v adresáři pro umístění repozitáře otevřít terminál a zadat
   git clone https://github.com/vao25/ZDO_Team8.git
2. *cd src/*
3. do adresáře images přidat obrázky pro zpracování (implicitně je adresář prázdný)
4. pak již lze program spustit dle pokynů v zadání SP, např.:
   python3 run.py output.csv incision001.jpg incision005.png

Popis klíčových myšlenek a metod programu

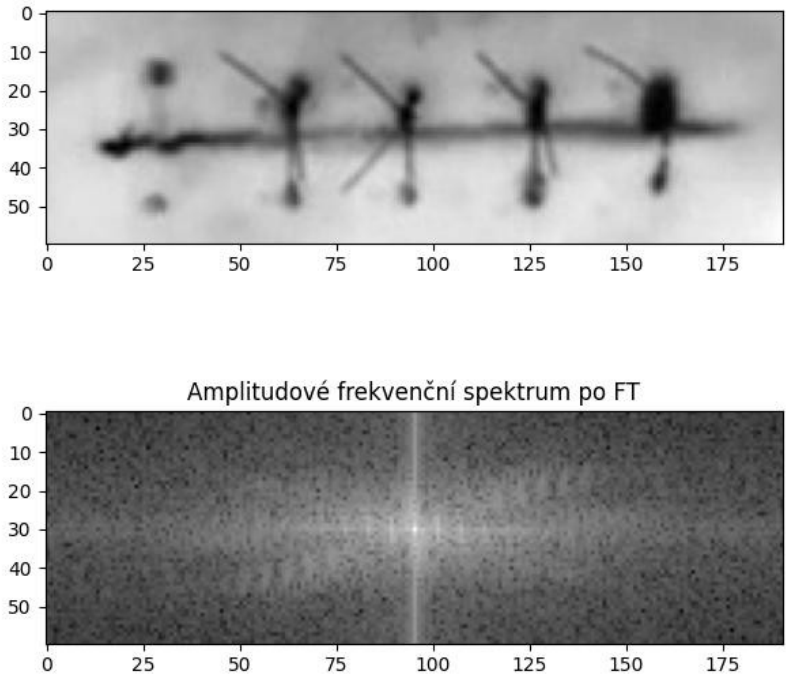
Byla snaha, aby byla každá metoda popsána a kód řádně okomentovaný tak, aby bylo zřejmé k čemu daná metoda slouží. V tomto referátu jsou uvedeny myšlenky, na kterých je program postaven. Bude popisován tak, jak je obrázek postupně zpracováván. Budou uvedeny i mezivýsledky, které by měly představu vylepšit. Pro úplnost a vhled do konkrétních implementací je vhodné si společně s tímto referátem procházet i zdrojové kódy.
Pozn.: zpracování obrázků je ukázáno na ilustrativním příkladu z trénovací množiny:
obr. *SA_20220620-103036_3vhpgd00n9vt_incision_crop_0.jpg*

Každý obrázek se načte jako šedotónový. Následně je předán metodě *zpracuj_obr(obr)*, která zajišťuje většinu zpracování obr. Nejdříve zvýší kontrast v obr., aby byl zvýrazněn rozdíl mezi ránou se stehy a pozadím. Následně je ke všem hodnotám jasů pixelů přičten rozdíl mezi průměrem jasů v daném obr. a tzv. referenčním průměrem jasů. Cílem je jasová korekce (když je potřeba ztmavení, zesvětlení obr.). Mezivýsledek po těchto krocích:



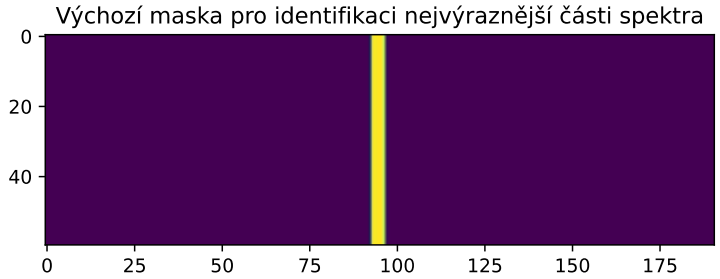
Obrázek 1: Ukázka prvního předzpracování vybraného snímku

Protože stehy a rána tvoří v obr. pravidelné struktury (stehy mají jen vyšší frekvenci než rána), je poté využita **frekvenční analýza - Fourierova transformace** a získáno frekvenční spektrum:



Obrázek 2: Frekvenční analýza

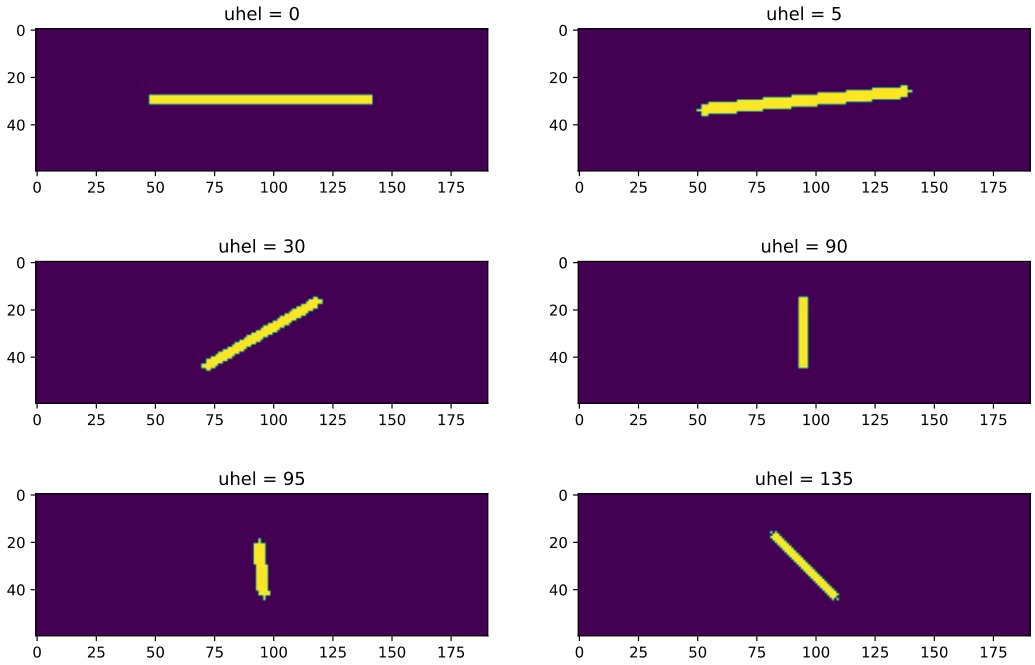
Pokud na sebe rána a stehy budou kolmé, budou ve spektru vytvářet charakteristický kříž, jehož frekvence reprezentují tyto příčné a podélné pravidelně se vyskytující struktury v podobě ran a stehů. Zvlášť ráně a zvlášť stehům pak odpovídá jedna část tohoto kříže. Pokud by na sebe rána se stehy nebyly kolmé, nebude pak ani kříž pravoúhlý. Pokud je struktura rány se stehy snímána na šikmo, bude kříž ve spektru pootočen. To jsou myšlenky, ze kterých se nadále vychází. Nyní je tedy úkolem identifikovat ve spektru tyto nejvýraznější části. To má na starost metoda *zjisti_smer()*. Vychází se z návrhu řešení pro tuto práci. Metoda vytvoří masku o šířce 4 pixely a délce odpovídající výšce obr.



Obrázek 3: Maska, která bude rotována okolo svého středu

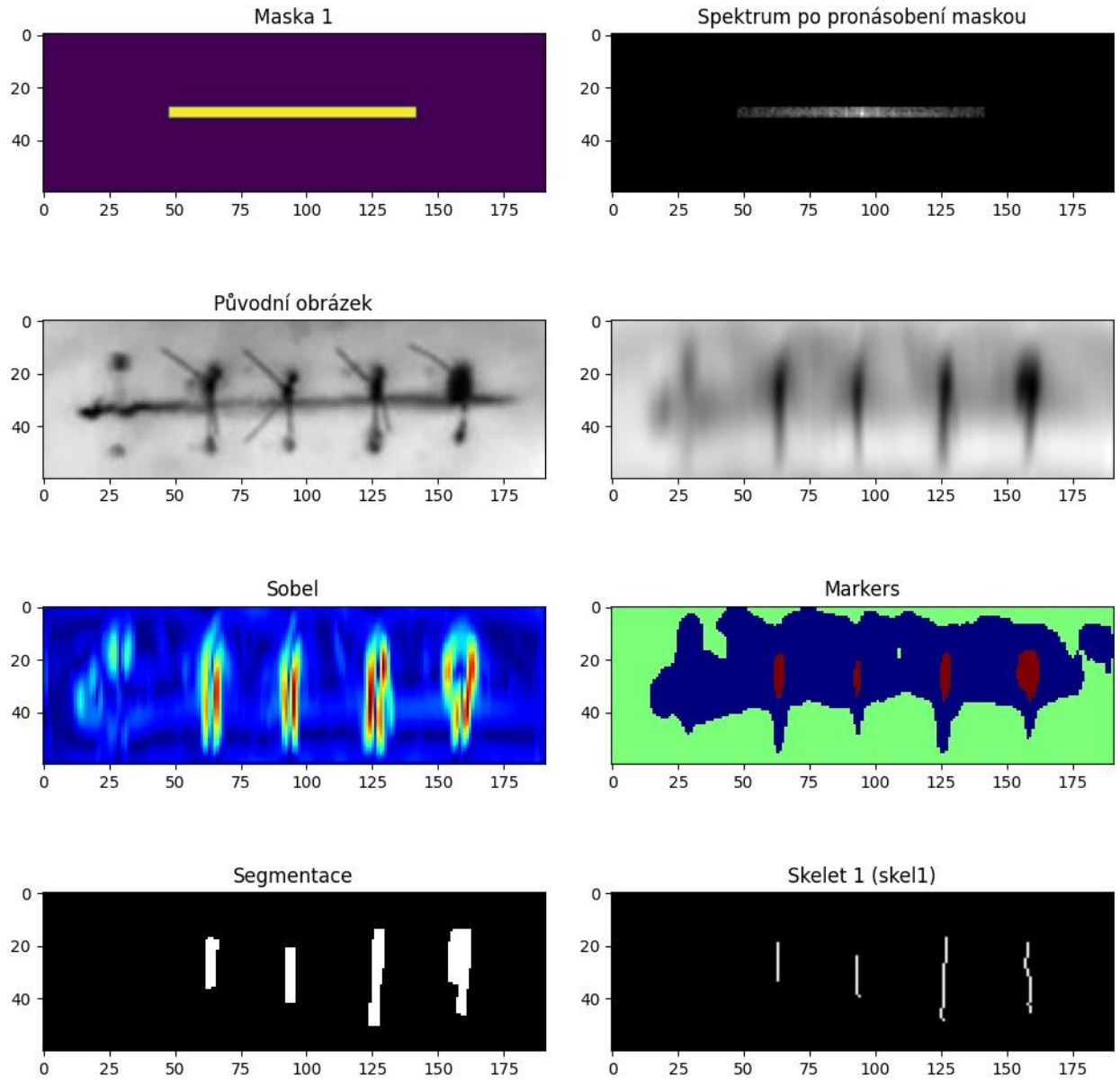
Obdélník je následně rotován po 5° okolo středu a vždy pronásoben (po prvcích, nikoliv maticově) se spektrem. To, co ze spektra zbude, je pak posčítáno. Jako „nejintenzivnější směr“ se pak určí maximum z těchto 36 (180/5) součtů. Aby nebyl zvýhodňován směr, kde by mohl být použit delší obdélník, je obdélník pro všechny směry stejný a to maximálně tak dlouhý, jak může být v „nejužším“ místě (určuje tedy výška obr.). Takto se identifikuje první část kříže, principiálně se může jednat buď o ránu nebo o stehy, záleží na konkrétním případě.

Další velmi využívanou metodou je metoda *vytvor_masku()*. Jejím úkolem je vytvořit masku (obdélník) v daném směru (pod zadaným úhlem). Volitelným parametrem je i delkaRel, jejíž hodnotou lze nastavit délku obdélníku. Může nabývat hodnot z intervalu $\langle 0;1 \rangle$, kde hodnota 1 značí maximální možnou délku – kdyby se místo obdélníku vzala pouze úsečka pod zadaným úhlem a omezila se velikostí obr. Význam tohoto parametru je ten, že je jím možné nastavit optimální délku obdélníku, protože vysoké hodnoty koeficientů spektra (vysoké frekvence) není vhodné brát (rána a stehy jsou spíše nižší frekvence). Zároveň pro každý směr může být užitečné vzít jinak dlouhý obdélník – typicky např. není nutné se omezovat poměrem strany dané výškou obr. pro horizontální směr ve spektru. Toto tedy nejde řešit jednoduše pomocí rotace jako v případě metody *zjisti_smer()*, stručný popis tvorby masky je uveden dále:
Je využit předpis přímky ($i = k \cdot j$), kde směrnice je dána hodnotou funkce *tangens* pro zadaný úhel. Hodnoty 0 – šířka obr. jsou touto směrnicí tedy pronásobeny. Pro strmý sklon přímky však vznikají mezi získanými body dost velké mezery (díry), je tedy nutné tyto místa zaplnit, aby vznikla nepřerušená úsečka. Poté je tato úsečka omezena na délku určenou již zmíněným parametrem *delkaRel* (hodnota přednastavena na 0,5: délka úsečky je tedy omezena na poloviční délku). „Obdélník“ je pak vytvořen tak, že ke každému bodu úsečky jsou přidány na každou stranu (nahoru, dolu, doleva a doprava) 2 další body. Jak je vidět na následující ukázce vytvořených masek, nejedná se tedy přímo o obdélník, spíše je to jakýsi „obdélníkovitý útvar“. Výsledky nevypadají příliš profesionálně, ale jde spíše o princip „zachycení frekvencí“, nikde není dáno, že se musí jednat přímo o obdélník.



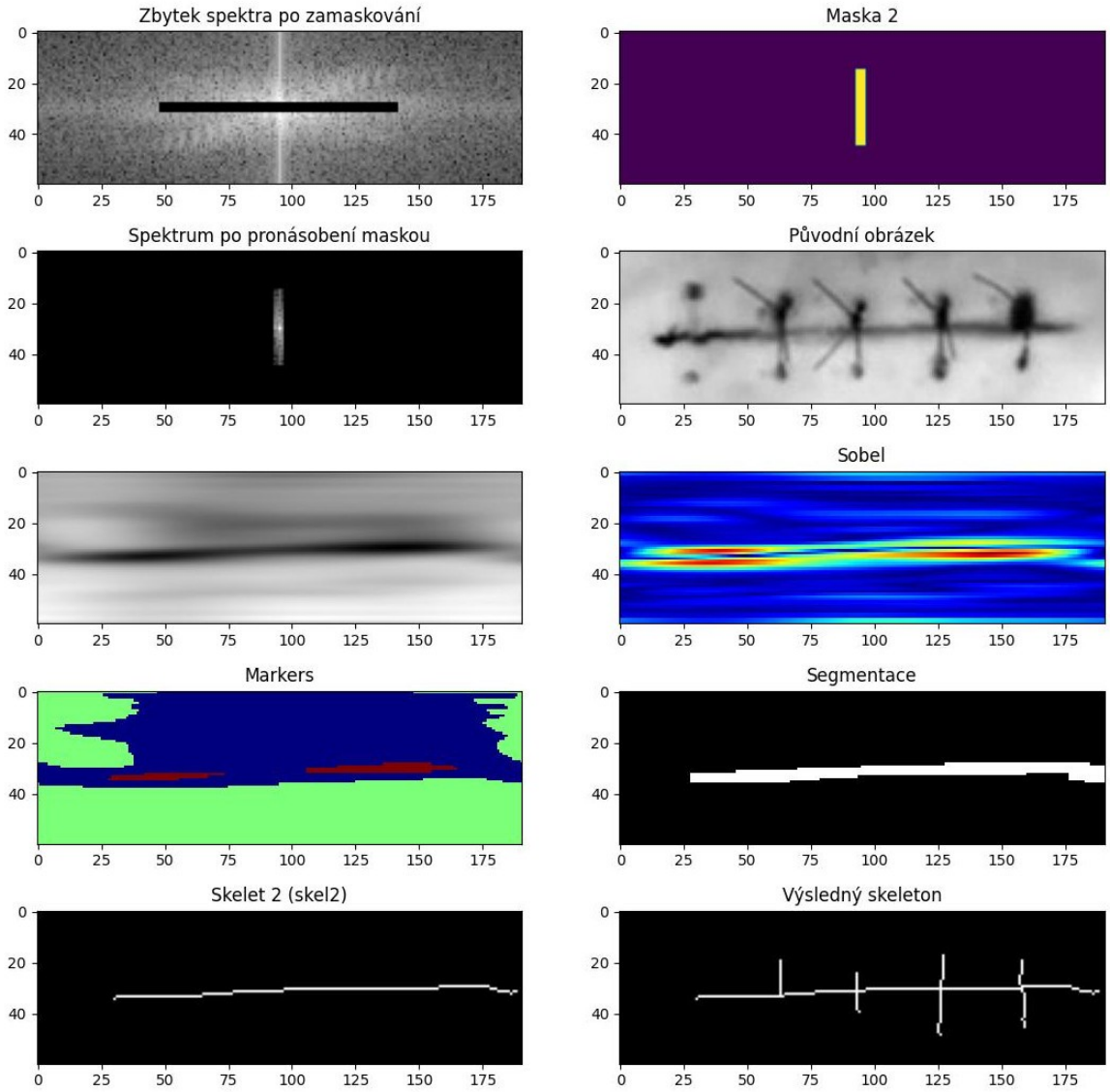
Obrázek 4: Ukázky vytvořených masek pomocí metody vytvor_masku(uhel, im.shape, delkaRel = 0.5) s danými parametry

Takto vytvořenou maskou je tedy vynásobeno spektrum a převedeno zpět do podoby obrázku. Výsledkem jsou více zvýrazněné struktury, které se budou lépe segmentovat. Pro tento účel byla nakonec zvolena **Watershed segmentace**. Mapa „pro zalití“ byla vytvořena pomocí hranového detektoru – konkrétně *Sobel* – na nově získanou podobu obr. Dále byly zvoleny 2 markery: jeden pro pozadí a jeden pro rány či stehy. Vychází se přitom z myšlenky, že hodnoty jasů pixelů rány nebo stehů budou daleko nižší než hodnoty jasů pozadí a zároveň, že pixelů pozadí bude pravděpodobně mnohonásobně více. Prahové hodnoty pro jednotlivé markery se v každém obr. určí následovně: vypočte se kumulativní histogram a jako markery pro pozadí se označí půlka nejjasnějších pixelů. Naopak jako markery rány (stehů) se označí 2,5% nejméně jasných bodů. Proč se zvolila přesně takováto čísla je vysvětleno později. Důležité je, že to zmenšuje závislost na konkrétních hodnotách jasu pro odlišné obr. Následně již může být zavolána knihovní funkce *skimage.segmentation.watershed(elevation_map, markers)*. Na výsledek se pak použije mediánový filtr o velikosti 3 pro filtraci šumu nebo malých segmentů, které nemohou být ránou (stehy). Z tohoto výsledku je pak utvořen skelet označený *skel1*. Tento skelet odpovídá buď ráně nebo stehům, to zatím ještě není jasné (záleží na konkrétním obr.) a určí se to později.



Obrázek 5: Posloupnost mezivýsledků popisovaných výše

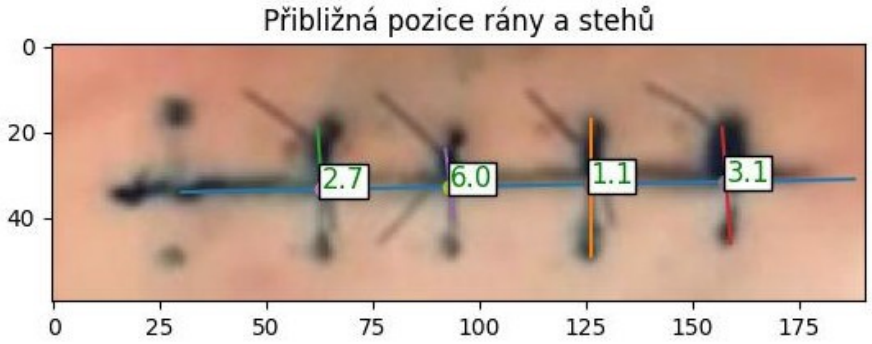
Následuje určení druhého nejintenzivnějšího směru ve spektru odpovídající buď ráně nebo stehům, což záleží na tom, jaký případ nastal předtím. Využije se k tomu v předchozím kroku získaná maska, a na pozice odpovídající 1 v masce se do spektra uloží nejmenší hodnota ve spektru (jakési vynulování, ale nuly to být nemohou, protože ve spektru se nacházejí i záporné hodnoty); tím dojde k „zakrytí“ nejvýraznější části a může být opět použita metoda *zjisti_smer()* pro určení druhého nejintenzivnějšího směru. Následuje stejný postup pro získání skeletu, jako v předchozím případě. Celkový skelet je pak dán součtem *skel1* a *skel2*. Ve výsledku však nemusí být souvislý skelet (mohou se vyskytovat i segmenty odpovídající větším částem, který mediánový filtr neodstranil, nebo třeba tečkám od fixů, kde nejsou stehy apod.). Ze skeletu se vybere největší (z hlediska počtu bodů) část a určí se jako výsledný skelet - skel (rána se stehy). Jím se pak vynásobí *skel1* a *skel2*, aby odpovídaly i jednotlivé části.
Teď jen již avizovaná poznámka, jak byly určeny hodnoty 0,5 a 0,025 pro markery. Byly utvořeny seznamy potenciálních hodnot pro pozadí [0.5 , 0.55, 0.6 , 0.65, 0.7 , 0.75, 0.8 , 0.85, 0.9 , 0.95] a pro ránu, stehy: [0.005, 0.01 , 0.015, 0.02 , 0.025, 0.03 , 0.035, 0.04]. Přes dva *for-cykly* pak byla pro dané dva parametry zjišťována úspěšnost na trénovací množině. Nejlépe tedy vyšly hodnoty 0,5 (50% pixelů) a 0,025 (2,5% pixelů).



Obrázek 6: Posloupnost mezivýsledků vedoucí k získání skel2 a výsledného skeletu

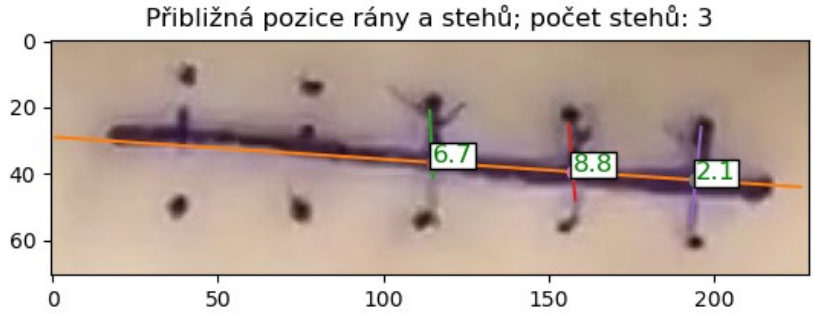
Již ve skriptu *run.py* se zjišťuje, jestli jednotlivé skelety nejsou prázdné (samé nuly). Pak žádný steh nebyl nalezen a jedná se pravděpodobně o případ 0 stehů.
Poté je použita metoda *priprav_data()*, která nejdříve určí, který z dvojice skeletonů je rána a který stehy. Vychází přitom z myšlenky, že tak, jak jsou obr. pořízeny, bude skelet s bodem nejvíce nalevo odpovídat ráně (samozřejmě by se stejně tak mohl vzít bod nejvíce vpravo). Může se stát, že bude-li steh velmi blízko okraji rány a obr. bude hodně pootočený (na šikmo), jako bod nejvíce nalevo se určí ze skeletu stehů. V takovém případě identifikace selže. Od této chvíle je tedy vždy *skel1* rána a *skel2* stehy. Dále metoda připraví data pro určení počtu stehů (*pozn.:* počet stehů už by šel určit i v této fázi, ale nakonec řešeno v modulu s vizualizací) a pro případnou vizualizaci. Vytvoří seznam *bodyRana*, do kterého se vloží dva seznamy se souřadnicemi bodu nejvíce nalevo a bodu nejvíce napravo rány (*skel1*). Jedná se tedy pouze o přibližnou pozici rány. Další seznam – *bodyStehy* – pak na každém indexu obsahuje další seznam odpovídající každému stehu. V nich jsou další seznamy pro dvojice souřadnic krajních bodů, které jsou ve stehu (v jednotlivých segmentech ve *skel2*) nejvíce nahoře a dole (předpokládá se, že rána bude v horizontálním směru, kdežto stehy spíše ve vertikálním, nebo budou jen nějak pootočeny). Od i-té souřadnice horního bodu je navíc odečtena 1, k i-té souřadnici dolního bodu pak 1 přičtena. Je to z důvodu toho, že posléze metoda určující průnik dvou úseček neurčí, že mají 2 úsečky průnik, pokud mají společný bod, který je v

jedné úsečce na úplném kraji. Je to tedy řešeno tímto malým zvětšením úseček stehů, protože je škoda přicházet o takto nadějně zpracované obr. Tento případ totiž nastával poměrně častokrát (je vidno i na posledním subplotu Obrázku 6). Samozřejmě, pokud stehy nejsou přesně ve vertikálním směru, může to vést k malému zkreslení úhlu průniku, ale to v této úloze tolik nevadí.
Metoda *spocti_stehy_viz()* se pak stará o spočítání stehů a případnou vizualizaci. Je k tomu využit kód poskytnutý k zadání této semestrální práce. Ústřední metodou pak je metoda *intersectLines()*, která mimo jiné určí to, jestli mají dvě úsečky společný průnik. Vždy se jí předávají dvojice souřadnic krajních bodů úseček rány a konkrétního stehu. Pokud mají společný průnik, zvýší se hodnota počtu stehů. Metoda *spocti_stehy_viz()* pak obstarává i případnou vizualizaci. Vizualizace je zobrazení úseček rány a stehů z dříve získaných krajních bodů. Zejména v případě rány se tedy nejedná o složení menších úseček zachycující přesný tvar (tedy nejen přímo úsečky) tak, jak je to v anotovaných datech. Vzhledem k povaze zadání (určit počet stehů) však má plnit spíše funkci ilustrativní, aby bylo vidno, že počty stehů nebyly určeny nějakým způsobem náhodně. Jedná se o přibližné udání pozice. Posledním příkazem ve spouštěcím skriptu run.py je pak pouze zavolání metody *zapis_vystup()*, která zapíše výsledky do požadovaného výstupního *.csv* souboru.

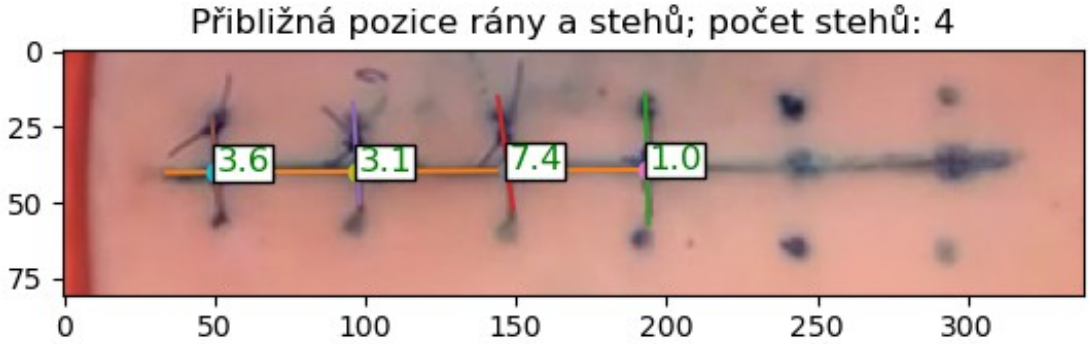


Obrázek 7: konečný výsledek zpracování obr. SA_20220620-103036_3vhpgd00n9vt_incision_crop_0.jpg (vizualizováno i s hodnotami úhlů, které jsou však spíše orientační)

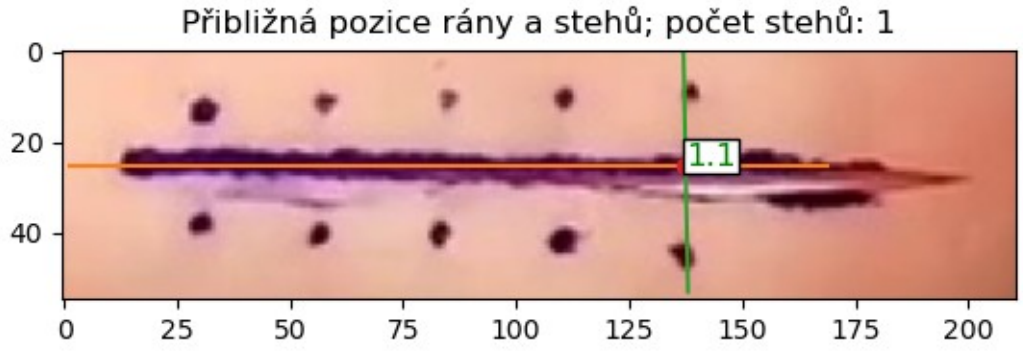
Další ukázky výstupu programu



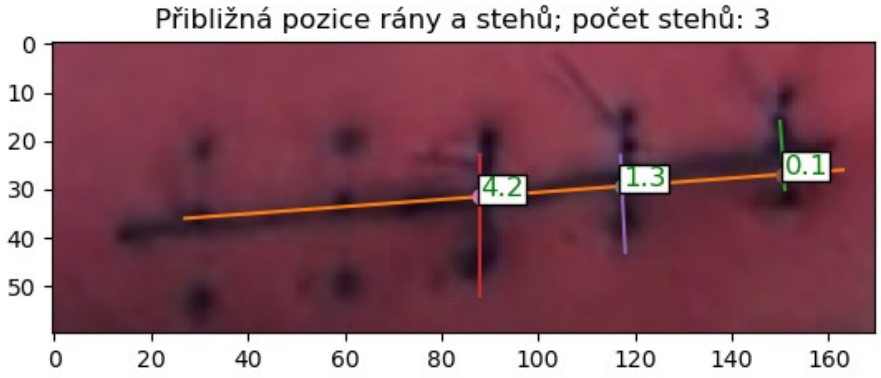
Obrázek 8: SA_20231011-104231_e3sdqld32ut6_incision_crop_0.jpg; určen správný počet stehů



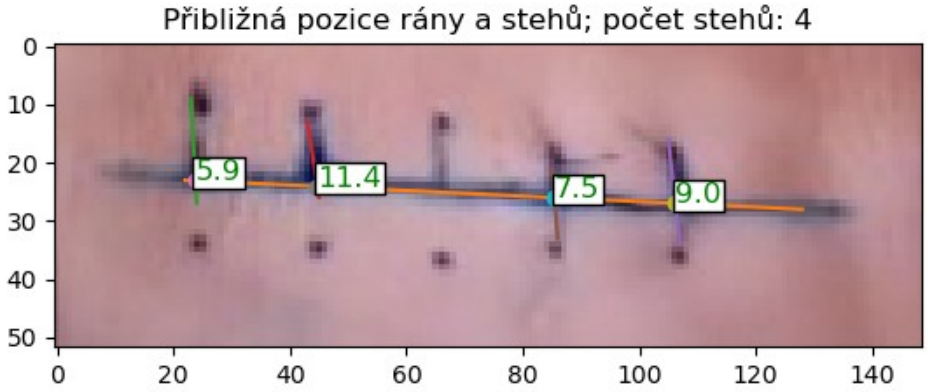
Obrázek 9: SA_20220801-104759_9xjxd9p268zh_incision_crop_0.jpg; určen správný počet stehů



Obrázek 10: SA_20230222-130319_c1g5pmjim3m6_incision_crop_0_start.jpg; určen špatný počet stehů – správně 0 stehů



Obrázek 11: SA_20220707-190320_tl4ev95290pp_incision_crop_0.jpg; určen správný počet stehů



Obrázek 12: SA_20220620-102621_8ka1kmwpywxv_incision_crop_0.jpg; určen špatný počet stehů – správně 2 stehy

Závěr

Výsledný program bohužel nefunguje ideálně. Z anotací byly pro každý obr. sečteny tagy <*polyline*> s *label="Stitch"*, pokud se žádný takový tag nevyskytoval, byl počet stehů určen jako nula. Pokud navíc nebyl přítomen ani tag <*polyline*> s *label="Incision"*, byla pro daný obr. dána hodnota -1 (obr. nelze zpracovat). Na trénovací množině pak program dosahuje **úspěšnosti 34,33 %** , což je nízké číslo (*ze 134 obr. ve 46 případech určen správný počet stehů*).
Z uvedených ukázek vyplývají i některé nedostatky tohoto řešení. Za prvé, pokud se v obr. nevyskytuje žádný steh (nebo třeba jeden, dva) a je zde hodně výrazně namalovaných teček (značek od fixy), bude je algoritmus považovat za pravidelný útvar o nějaké frekvenci a může je zvýraznit tak, jak se stalo třeba na obr. 10. Potom také, budou-li čáry od fixy příliš výrazné a budou-li protínat ránu, budou taktéž považovány za stehy (viz obr. 12).
Naopak toto řešení by mělo být vhodné i pro případ pootočené rány ve snímku, či stehům pod jiným úhlem (nejen okolo 90°). Nemělo by také být náchylné na různé jasové rozložení snímků (viz obr. 11). Výhoda zpracování ve frekvenční oblasti je i odstranění nití od uzlíků stehů, které jsou ve spektru vynulovány maskami.
Byť úspěšnost není dobrá (a výsledná vizualizace není úplně precizní), je vidět, že pro nějaké základní vstupní obrázky prokazuje program celkem dobrou (alespoň principiální) funkčnost.