

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

ZPRACOVÁNÍ OBRAZU

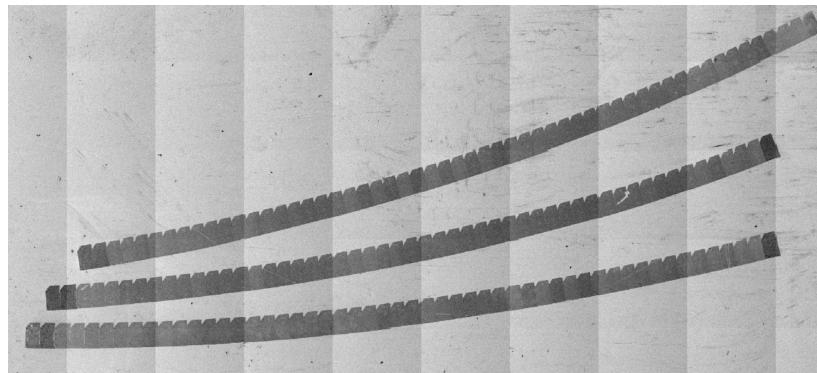
Technická zpráva projektu – *Slice Recognition in Array Tomography*

1 Úvod

Cílem projektu je implementovat a otestovat algoritmus pro automatické rozpoznávání řezů pro *Array Tomography*. Ideální algoritmus by měl být robustní vůči rotaci řezů, šumu a jiným artefaktům. Výstupem projektu je kromě samotné implementace také technická zpráva obsahující mimo jiné průzkum stávajících metod a dostupných dat a popis zvoleného postupu. Řešitel je pouze jeden (autor), není tedy uvedeno složení týmu, dělení práce, ani ostatní náležitosti relevantní pouze pro vícečlenné týmy.

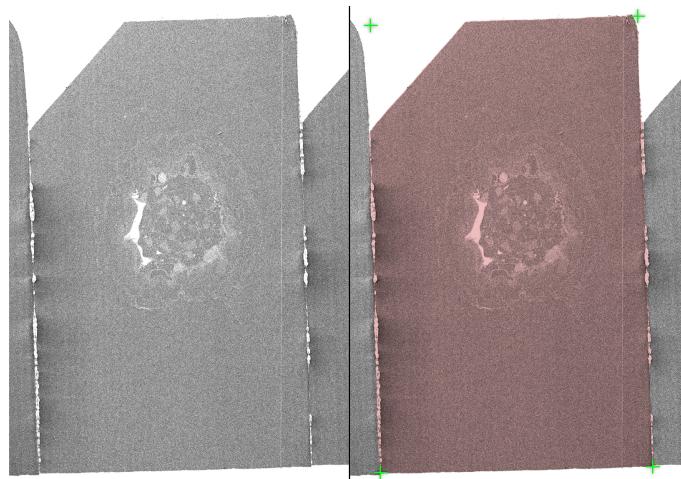
2 Stávající metody

Array Tomography (dále jen *tomografie*) je technika tvorby objemového modelu vzorku. V prvotních krocích je vzorek **kvádrovitého tvaru** nařezán na plátky a nasnímán rastrovacím elektronovým mikroskopem [5]. Snímání probíhá ve více fázích. Zaprvé je nasnímáno panorama všech řezů v nízkém rozlišení. Na základě panoramatu jsou manuálně či jinak určeny: křivka, na které jsou řezy rozloženy, a přibližné rozměry jednotlivých řezů [2], viz obrázek 1.



Obrázek 1: Ukázka pásku řezů nasnímaných v nižším rozlišení.

Na základě těchto parametrů hlava mikroskopu v přibližných lokacích nasnímá jednotlivé řezy ve vysokém rozlišení. Toto řešení se zaměřuje na poslední krok předzpracování snímků – lokalizace ROI (*Region Of Interest*) na snímcích jednotlivých řezů s vysokou mírou přesnosti, viz obrázek 2.



Obrázek 2: Příklad požadovaného výstupu detekce řezu [2, 8]. Vlevo: vstupní snímek řezu ve vysokém rozlišení, vpravo: výstup detekce ve podobě segmentovaného obrazu či vrcholů obalového čtyřúhelníku.

Takto zadaný problém detekce řezů lze formulovat jako sémantickou segmentaci obrazu, která je úspěšně realizována konvolučními neuronovými sítěmi [4]. Jelikož jsou však KNN a obdobné metody strojového učení mimo rozsah kurzu ZPO, budou prozkoumány pouze tradičně algoritmické přístupy lokalizace ROI.

Základní algoritmus, jak ho popisují [2, 8], spočívá v detekci hran *projekční profilovou analýzou*. Tento postup selhává na snímcích s narovenanými vzorky nebo vysokou mírou artefaktů, ovšem může být pro potřeby tomografie dostačující, pokud je na ostatní snímky v sekvenci aplikován *template matching* pro jejich relativní zarovnání vůči sobě. Vzhledem k nátuře dostupných dat se však toto řešení zaměřuje právě na detekci ROI na izolovaném snímku řezu.

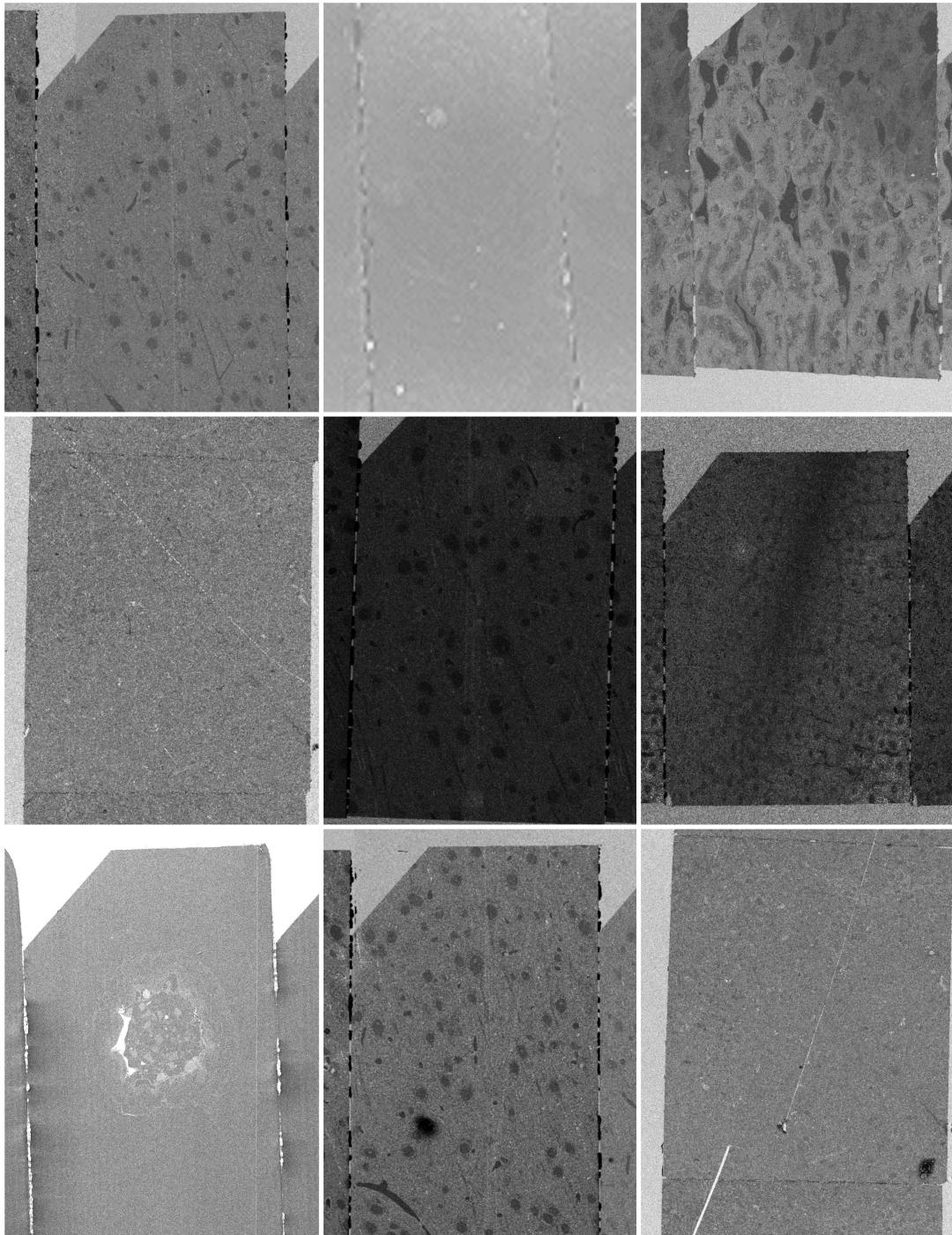
3 Plán implementace

Samotná implementace se věnuje algoritmu představenému v [2], který po aplikaci Cannyho či Sobelova filtru [9] pomocí paradigmatu RANSAC hledá vhodné úsečky vyjadřující hledané hrany ROI. Toto řešení pro implementaci využívá *Python* především v kombinaci s knihovnou *OpenCV* pro manipulaci s obrazovými daty.

Pro evaluaci výsledné implementace je použita metrika *Jaccardův koeficient*, neboli *průnik přes sjednocení*. Při výpočtu metriky jsou srovnávány dvě oblasti – ROI vypočtený implementovaným algoritmem a manuálně anotovaná *ground truth*.

Datasetsy

Pro potřeby vývoje bylo manuálně extrahováno a sesbíráno 15 vzorků, viz ilustrační výběr na obrázku 3. Akvizice a anotace vzorků je netriviální a časově náročná a volně dostupné on-line datasetsy jsou vzácné či zcela nedostupné. Pro potřeby evaluace je v plánu, kromě implementace, také akvizice dodatečných vzorků a jejich manuální anotace.



Obrázek 3: Výběr 9 vzorků z vývojového datasetu.

4 Popis metody

Metoda sestává z několika izolovaných kroků – redukce šumu, aplikace filtru pro detekci hran, odfiltrování nežádoucích objektů a hledání čtyř úseček pomocí RANSAC algoritmu. Algoritmus na vstupu kromě vzorku vyžaduje také přibližnou šířku a výšku hledaného řezu. Toto by však nemělo být problematické, jelikož stejné parametry jsou nutné k samotnému nasnímání vstupních vzorků.

Předzpracování

Zde jsou pojednávány: redukce šumu aplikací Gaussovského filtru, aplikace filtru pro detekci hran a odfiltrování nežádoucích objektů. Metoda popisuje exaktní vztahy pro určení parametrů Gaussovského filtru, kde w a h jsou dimenze obrazu, s je velikost konvolučního jádra a σ je směrodatná odchylka:

$$s = \left\lfloor \frac{\max(w, h)}{200} \right\rfloor \cdot 2 + 1 \quad (1)$$

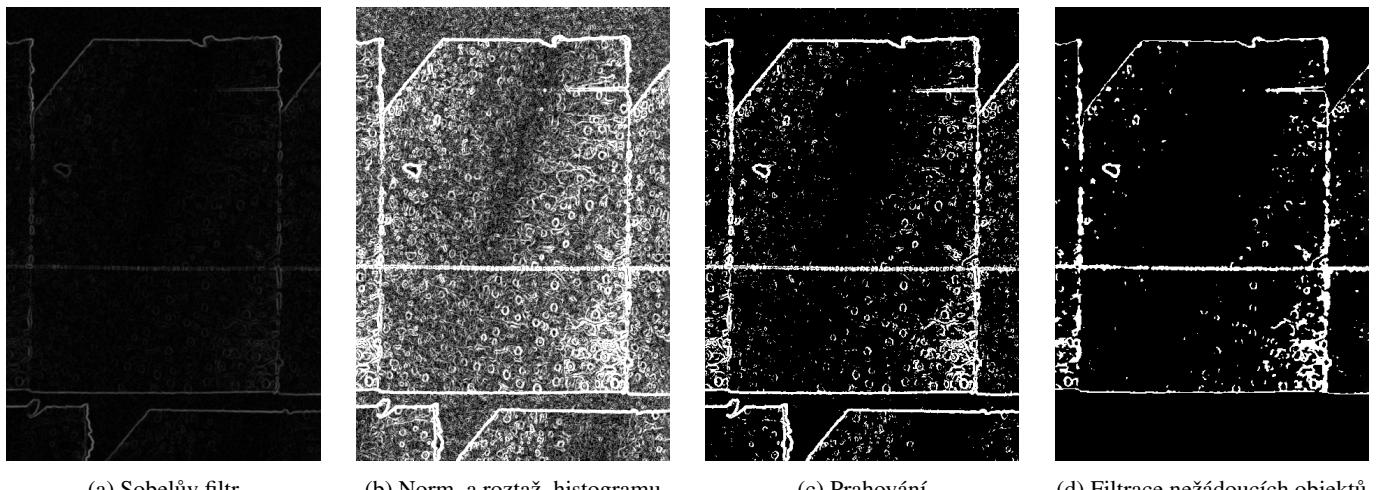
$$\sigma = 0.3 \cdot \left(\frac{s-1}{2} - 1 \right) + 0.8 \quad (2)$$

Jako filtr pro detekci hran metoda volí Sobelův filtr. Na takto vyfiltrovaném obrazu však navíc specifikuje specializovaný proces roztažení histogramu (podobný *ekvalizaci histogramu* [1]):

1. Je identifikováno 11 % nejjasnějších pixelů a všem těmto pixelům je přiřazena nová hodnota odpovídající nejnižší hodnotě z té samé skupiny pixelů (nikoliv hodnota 0). Tento krok má za cíl normalizovat extrémní hodnoty pro robustnější detekci hran.
2. Nový histogram je roztažen aby pokrýval interval 0 – 255, tzn. intenzity všech pixelů jsou normalizovány zpět do plného spektra.
3. Nakonec je aplikován prahový filtr s prahem hodnoty 254.

Poslední krok předzpracování odfiltruje nežádoucí objekty:

1. Na vstupní (nezpracovaný) vzorek je aplikován Otsuův prahový filtr [6], který každý pixel klasifikuje do tří pozadí (hodnota 0) / popředí (hodnota 1).
2. V popředí jsou identifikovány spojité komponenty. Spojité komponenty, jejichž celková plocha nedosahuje 15 % plochy obrazu, jsou převedeny na negativní masku, která je aplikována na binární obraz z předchozího kroku. Viz obrázek 4.



Obrázek 4: Výstupy jednotlivých kroků předzpracování (na obrazu již vyhlazeném Gaussovským filtrem).

Hledání úseček

Jako inicializační krok je binární obraz s hranami rozdělen na čtyři vzorky – na dvě vertikální a dvě horizontální poloviny (dále jen *semivzorky*). To má pomoci algoritmu hledání úseček izolací jednotlivých hran vstupního obrazu. Na každém semivzorku jsou následně vzorkovány dvojice bodů modelující hledanou úsečku. Vzorkování těchto bodů následuje dodatečná pravidla:

- Pro vertikální semivzorky je bod p_1 vzorkován z horní poloviny a bod p_2 z dolní poloviny obrazu. To má zajistit vertikalitu vzorkovaných úseček. Analogický postup platí pro horizontální semivzorky.
- Body jsou vzorkovány nerovnoměrnými rozloženími pravděpodobností, založenými na Gaussovském rozložení. Rozložení jsou určena pro každý semivzorek zvlášť a cílem je upřednostňovat body blíže středu obrazu, viz rovnice 3.

$$\mu_{\text{left}} = \left\lfloor \frac{w}{2} \right\rfloor - \left\lfloor \frac{s}{2} \right\rfloor, \quad \sigma_{\text{left}} = \frac{w-s}{3}, \quad (3a)$$

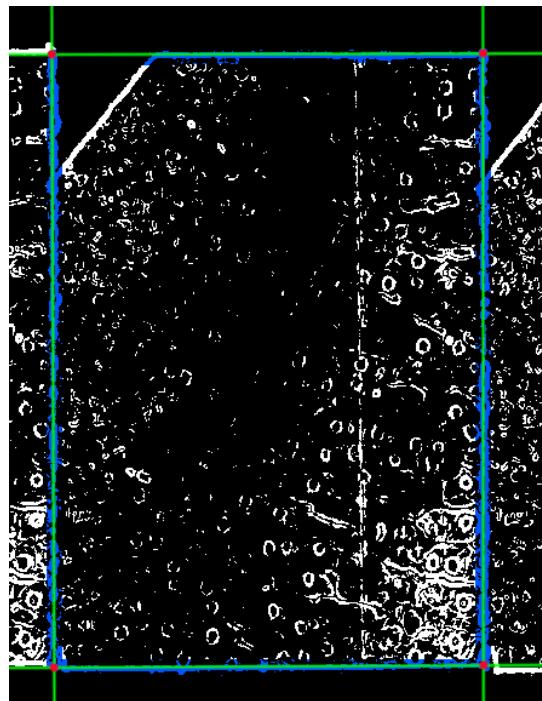
$$\mu_{\text{right}} = \left\lfloor \frac{w}{2} \right\rfloor + \left\lfloor \frac{s}{2} \right\rfloor, \quad \sigma_{\text{right}} = \frac{w-s}{3}, \quad (3b)$$

$$\mu_{\text{top}} = \left\lfloor \frac{h}{2} \right\rfloor - \left\lfloor \frac{l}{2} \right\rfloor, \quad \sigma_{\text{top}} = \frac{h-l}{3}, \quad (3c)$$

$$\mu_{\text{bottom}} = \left\lfloor \frac{h}{2} \right\rfloor + \left\lfloor \frac{l}{2} \right\rfloor, \quad \sigma_{\text{bottom}} = \frac{h-l}{3}, \quad (3d)$$

Rovnice 3: Parametry Gaussovských rozložení pro vzorkování bodů úseček nad jednotlivými semivzorky, kde w a h jsou šířka a výška obrazu a s a l jsou **přibližná šířka a výška hledaného řezu**.

Po navzorkování dvojic (p_1, p_2) nad každým semivzorkem je vždy zvolen nevhodnější kandidát. *Fitness* těchto kandidátních modelů je vypočtena jako počet *inliers*, v tomto případě bodů z binární masky o hodnotě 1, které leží ve specifikované vzdálenosti od úsečky. Jedná se o **Eukleidovskou vzdálenost bodu od úsečky** $d = \frac{\max(w,h)}{50}$ či nižší, kde w a h jsou rozměry obrazu. Počet vzorkovaných dvojic, tedy iterací, vychází v RANSAC paradigmatu z požadované pravděpodobnosti nalezení řešení p , poměru počtu *inliers* k celkové velikosti vzorku w a z minimálního počtu bodů n (2 pro úsečku): $\frac{\log(1-p)}{\log(1-w^n)}$. V práci [2] bylo empiricky rozhodnuto, že 1000 iterací na hranu dostačuje pro adekvátní robustnost algoritmu. Po nalezení každé úsečky zbývá najít průniky jejich extrapolací (průměk), viz obrázek 5.



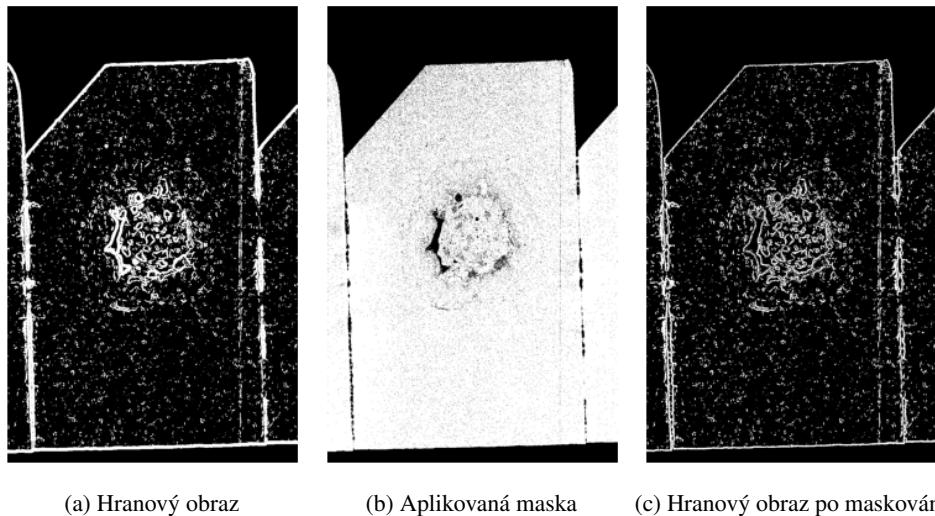
Obrázek 5: Ilustrace výsledného modelu. Modré jsou *inliers*, zeleně extrapolace výsledných úseček a červeně body výstupního polygonu.

5 Úpravy metody

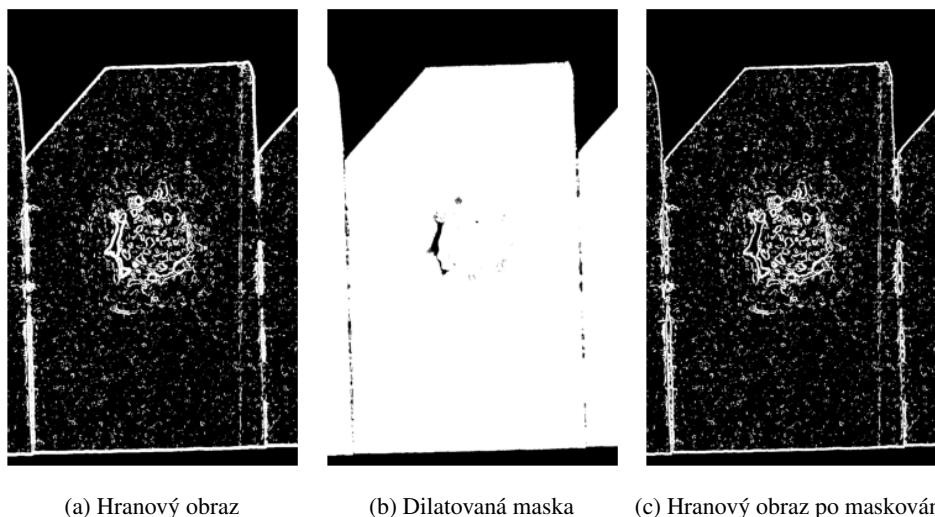
V rámci prototypování a následné implementace jsem se rozhodl pro několik úprav referenční metody, které by měly vést k lepším výsledkům lokalizace, jednoduššímu použití výsledného programu či snížení výpočetní složitosti.

Předzpracování obrazu

V původní metodě je nad vstupním obrazem aplikován Otsuův prahový filtr pro získání masky diskriminující pozadí a popředí řezu v obraze. Následně mají v této masce být identifikovány a eliminovány malé spojité komponenty, načež je maska použita pro očištění hranového obrazu od nadbytečných artefaktů, viz obrázky 6. Problém v této metodě lze zpozorovat v obrázku 6c – linie řezu, především vrchní a spodní, jsou částečně ořezány, což snižuje šanci na jejich detekci, obzvlášť pokud se uvnitř řezu nachází vysoké množství šumu. Tento jev nastává nesouladem v konstrukci hranového obrazu a masky; zatímco maska je konstruována z původního obrazu, hranový obraz je konstruován z obrazu rozostřeného gaussovským filtrem. Řešením je kompenzovat dilatacií masky. Jelikož jsou řezy zpravidla umístěny vedle sebe, a tedy pouze vrchní a spodní hrany jsou položeny proti pozadí, rozhodl jsem se masku dilatovat pouze ve vertikálním směru. Toho je docíleno použitím vertikálně orientovaného strukturovacího prvku obdélníkovitého tvaru (šířka = 1 px). Výška vychází z poloměru jádra použitého v gaussovském filtrovi, tedy $\max(w, h)/200$ ¹. Výsledek, když je zachována větší část původní hrany, lze vidět na obrázcích 7.



Obrázek 6: Maskování hranového obrazu v původní metodě.



Obrázek 7: Maskování hranového obrazu maskou dilatovanou strukturovacím prvkem vertikálně obdélníkovitého tvaru.

¹K výšce je následně přičtena 1 pokud je podíl sudý. Nenásobit poloměr 2 jsem se rozhodl proto, aby metoda nadměrně nepřečeovala výšku řezu.

Odhad rozměrů řezu

Původní metoda vyžaduje na vstupu kromě obrazu řezu, také přibližné rozměry řezu. Jelikož je tento parametr potřebný k samotnému nasnímání řezů, není toto příliš zásadní omezení. Přesto mě nutnost anotovat několik vzorků z různých dávek motivovala k implementaci automatického odhadu těchto rozměrů. Metoda odhadu je relativně triviální – ve zpracovaném a maskovaném hranovém obrazu jsou nalezeny dvojice extrémních souřadnic nenulových pixelů v horizontálním i vertikálním směru a jejich rozdíl dá odhad rozměru řezu. Dále jsou oba odhadnuté rozměry zastropovány na 80 % příslušné dimenze obrazu. Jelikož zpravidla řez zleva i zprava sousedí s jiným řezem, bude odhadnutá šířka též v každém případě 80 % šířky obrazu, nicméně i takto nepřesný odhad je dostatečný i díky úpravě popsané v následujícím odstavci.

Vzorkování přímek

V původní metodě jsou kandidátní přímkové vzorky výběrem dvou bodů v hranovém obrazu, kdy výběr má respektovat gaussovská rozložení se speciálními parametry, viz rovnice 3. Tato rozložení pomáhají při vzorkování vybírat relevantní body, ovšem prohledávaný stavový prostor je i tak relativně vysoký – při obrazu o velikosti 1000×800 bodů se jedná o 4×10^{10} možných úseček pro jednu ze čtyř hran. Analýzou hranových obrazů jsem zjistil, že vykazují hustotu zhruba 10 %, tedy pouhá desetina bodů je nenulová. Pouhou úpravou rozložení pravděpodobnosti $p(x|x=0) = 0$ lze na stejném příkladu zmenšit stavový prostor zhruba na 4×10^8 , obecně $\sim 10^2$ krát. Tato optimalizace je do jisté míry approximativní, ovšem považuji za relativně kvalitní heuristiku z které vychází, tj. že body popisující hledanou přímkou leží na některé z hran detekovaných Sobelovým operátorem.

Fitness kandidátních přímek

Při hledání kandidátních modelů (přímek) v RANSAC paradigmatu, jsou tyto modely ohodnocovány nějakou metrikou (dále jen *fitness*) a jako výsledný je zvolen právě kandidát s nejlepší (nejvyšší) fitness. Základní metriku (tedy počet *inlierů*) jak ji popisuje původní metoda, jsem zanechal, ovšem rozhodl jsem se ji při určení výsledné fitness váhovat dodatečnými parametry kandidátní přímek. Všechny 4 hrany ohodnocují tzv. **skóre náklonu**, které měří jak moc se náklon přímkového řezu blíží k „ideálnímu úhlu“, tj. zcela vertikální či zcela horizontální:

$$\text{score}_{\text{slope}} = \mathcal{N}(\delta; 0, \sigma^2),$$

$$\delta = \begin{cases} \left| \theta - \frac{\pi}{2} \right|, & \text{pro levou/pravou hranu,} \\ \min\{\theta, \pi - \theta\}, & \text{pro vrchní/spodní hranu,} \end{cases} \quad (4)$$

$$\theta = |\arctan(a/b)|, \quad \sigma = 20^\circ \text{ (v rad)}$$

Dále, při hledání vrchní či spodní hrany, kandidátní přímkové řezy ohodnocují poměrem nenulových a nulových bodů nad či pod ní a to v masce diskriminující popředí a pozadí. Význam této metriky je přiblížit výslednou hranu co nejbliže skutečným okrajům řezu a nazývám ji **score_{hug}**, viz vizualizace na obrázku 8. Opět zde vycházím z předpokladu, že řezy spolu při snímání shora a zdola nesousedí.



Obrázek 8: Vizualizace oblastí pro výpočet $\text{score}_{\text{hug}}$ kandidátních přímek pro vrchní a spodní hranu.
Pro obě přímkové řezy je zde $\text{score}_{\text{hug}} > 0.5$, tedy v příslušné oblasti je nízký poměr nenulových bodů vůči nulovým.

Jelikož obě metriky nemusí být obecně prospěšné pro lokalizaci nad arbitrárními vzorky, rozhodl jsem se je parametrisovat:

$$\text{fitness} = (1 - \alpha) \cdot \text{inliers} + \alpha \cdot (\text{score}_{\text{slope}} \cdot \text{inliers}) \quad (5)$$

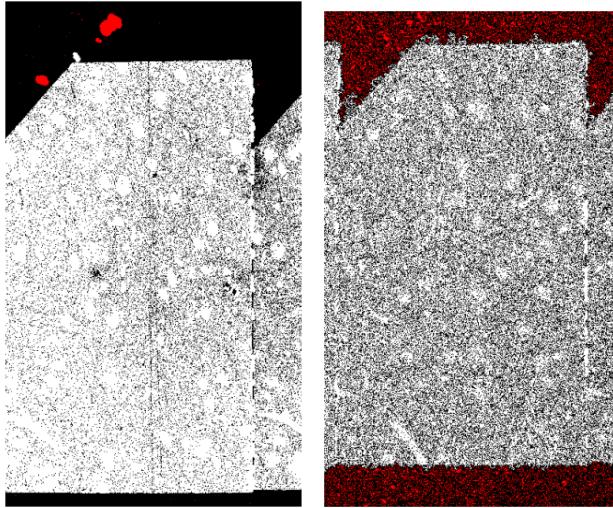
$$\text{fitness}_{\text{horizontal}} = (1 - \beta) \cdot \text{fitness} + \beta \cdot (\text{fitness} \cdot \text{score}_{\text{hug}}^2) \quad (6)$$

6 Implementace

Implementace proběhla ve dvou fázích – prototypování v Jupyter notebooku nad Python 3.11 a následně implementace komplikané verze v C++. Nad oběma technologiemi jsem využil metod a struktur z knihovny OpenCV, přičemž komplikanou verzi jsem dále optimalizoval použitím OpenMP. Obě zmíněné implementace by měly být funkčně identické, budu tedy popisovat jeden společný algoritmus. Většina popsaných kroků, jak v původní, tak v upravené metodě, je při použití OpenCV implementačně přímočará, zaměřím se tedy pouze na netriviální části, jejichž implementace nebyla explicitně popsána v původní práci [2] či v sekci 4.

Odstanění malých komponent z binární masky

Nad vstupním obrazem řezu je aplikován Otsuův prahový filtr a ve výsledné masce jsou odstraněny spojité komponenty, jejichž plocha nedosahuje 11 % celkové plochy obrazu. Identifikaci těchto komponent realizuji metodou pro hledání kontur, vestavěnou do OpenCV, implementující algoritmus popsány v [7]. Parametry volání jsou RETR_EXTERNAL pro režim hledání kontur (pouze vnější obvodové kontury) a CHAIN_APPROX_SIMPLE pro snížení paměťové náročnosti. Pro výpočet plochy lze opět využít OpenCV metodu contourArea, nad jejíž návratovou hodnotou lze triviálně odfiltrovat nežádoucí komponenty, viz obrázky 9. Je důležité zmínit, že hledání těchto komponent je realizováno nad maskou před její dilatací.



Obrázek 9: Hledání malých spojitých komponent (červeně) pomocí kontur.
Tímto krokem lze spolehlivě odfiltrovat větší nežádoucí objekty i šum z pozadí řezu.

Vzorkování přímek

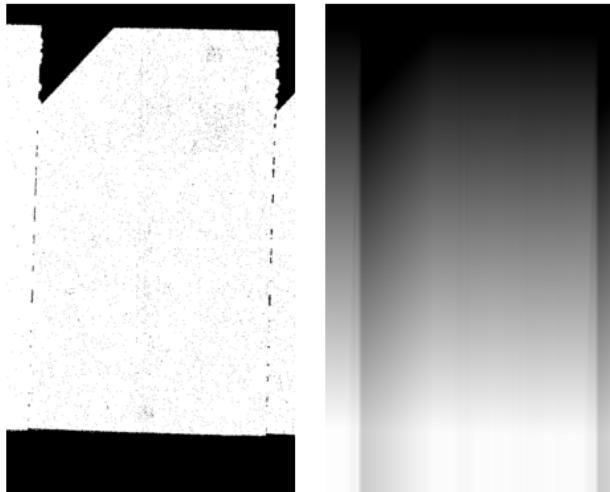
Jak bylo naznačeno v sekci 5, oproti původní metodě jsem upravil rozložení pravděpodobnosti pro vzorkování bodů kandidátních přímek. Logiku pro sestavení tohoto rozložení a následné vzorkování implementuje metoda find_edge_line_ransac/5. Sestavení rozložení se odehrává v těchto hlavních krocích:

1. Z hranového obrazu jsou vyselektovány pouze nenulové body.
2. Z této množiny jsou dle souřadnic vyselektovány pouze body náležející danému semivzorku (viz sekce 4).
3. Tato množina je dle souřadnic rozdělena na 2 pomocná pole g_1 a g_2 , a to pro obě poloviny semivzorku.
4. Jsou sestaveny množiny $\{\mathcal{N}(x; \mu_{edge}, \sigma_{edge}) \mid x \in g_1\}$ a $\{\mathcal{N}(x; \mu_{edge}, \sigma_{edge}) \mid x \in g_2\}$ kde \mathcal{N} je gaussovské rozložení s odpovídajícími parametry pro danou hranu.
5. Hodnoty těchto množin, mapující pravděpodobnost navzorkování každému z bodů v g_1 a g_2 , jsou normalizovány.

S daným počtem iterací jsou vzorkovány dvojice bodů z g_1 a g_2 a přímka, kterou popisují, je vyjádřena v obecném tvaru $0 = ax + by + c$. Tuto parametrizaci jsem zvolil, jelikož se přímo nabízí pro vyjádření z dvojice bodů a navíc je výpočetně triviální při této parametrizaci odvozovat souřadnice bodů, kterými prochází při scanning průchodu, což je výhodné pro výpočet score_{hug} (viz dále).

Výpočet score_hug přímky

Tato metrika je počítána pouze pro vrchní a spodní hranu, viz sekce 5. Naivní implementace by spočívala v *scanning* průchodu bodů, kterými přímka prochází, a počítání nenulových bodů ve sloupci nad/pod ní. Tento přístup však do vyhodnocení v každé iteraci zanáší velké množství operací včetně paměťových přístupů, přičemž velké množství z nich je duplicitní. Implementovaná optimalizace spočívá v předvypočítání kumulativní sumy nad řádky binární masky, viz vizualizace na obrázku 10. Tuto matici lze následně referencovat pro zjištění počtu nenulových bodů ve sloupci nad libovolným bodem, v konstantním čase.



Obrázek 10: Očištěná binární maska (vlevo) a vizualizace její kumulativní sumy přes řádky (vpravo).

7 Evaluace

Nad dostupnými vzorky (15 obrazů řezu) byla provedena evaluace metrikou *Intersection Over Union*, dále jen *IoU* s parametry $\alpha = \beta = 0.8$. Rozlišení snímků bylo v rozsahu 94×183 až 815×1284 bodů. Sumarizaci pro všechny řezy kromě slice11.png lze vidět na obrázku 11. Ze summarizace lze pozorovat, že metoda spolehlivě konverguje již před hranicí 100 iterací. To, že se jedná o desetinu iterací, kterou doporučuje autorka původní metody [2], odpovídá hypotéze v sekci 5, která vychází z předpokladu, že stačí vzorkovat pouze 10 % bodů celého obrazu.

Důvod, proč v summarizaci nebyl zahrnut vzorek slice11.png lze pozorovat na obrázku 12 – na tomto vzorku je některá z hran po celé své délce mimo snímek, a pro takový vzorek metoda nikdy nezkonverguje, jelikož alespoň dva body hrany musí ležet v obraze.

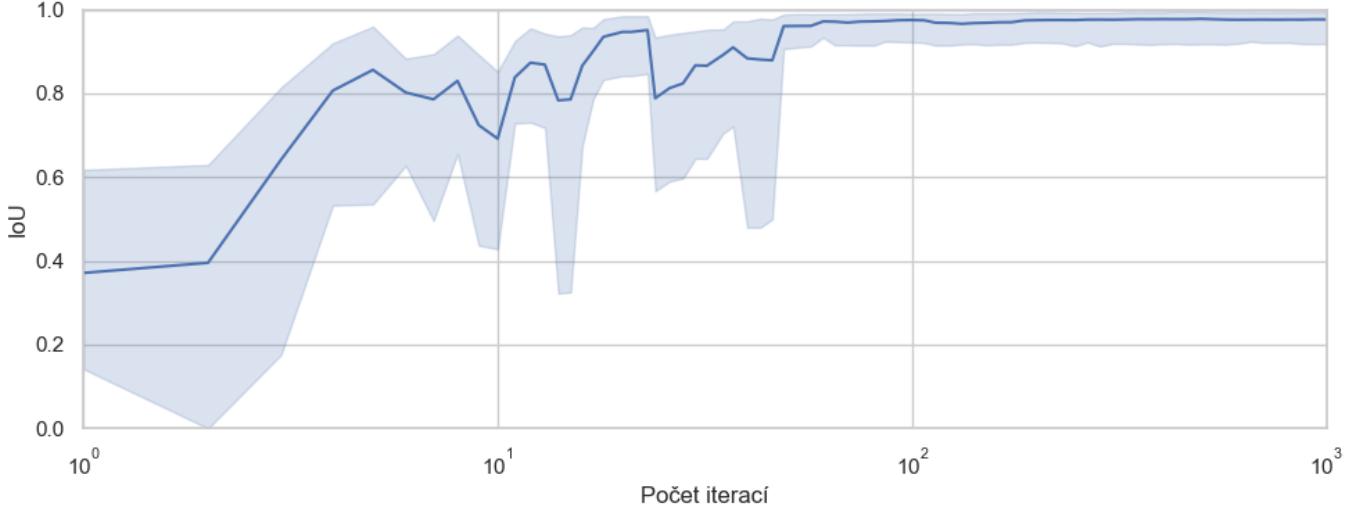
Když pomineme omezení na obrazech s touto vadou, průměrná IoU vychází 0.9764 ($\sigma = 0.0007$). Nízkou chybu $\epsilon_\mu = 0.0236$ lze vysvětlit i nepřesností v anotacích, z tohoto hlediska tedy implementaci považují za úspěšnou.

8 Překlad a použití programu

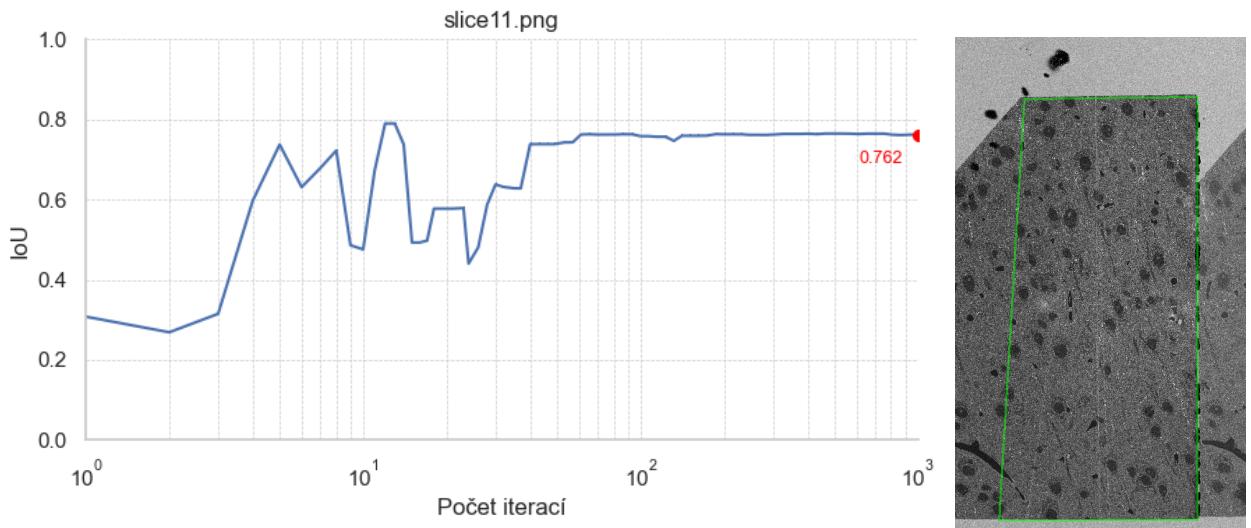
Zdrojové soubory komplikované verze s paralelizací pomocí OpenCV, jsou umístěny v adresáři `bin`. V tomto adresáři je také `Makefile` soubor, s jehož pomocí lze příkazem `make` program sestavit. Program přijímá následující argumenty příkazové řádky:

- **--iterations** (nepovinný, výchozí hodnota: 1000) – Počet iterací pro hledání každé hrany RANSAC algoritmem.
- **--slopeFactor** (nepovinný, výchozí hodnota: 0.8) – Parametr α pro evaluaci fitness přímek (viz sekce 5).
- **--hugFactor** (nepovinný, výchozí hodnota: 0.8) – Parametr β pro evaluaci fitness přímek (viz sekce 5).
- **--seed** (nepovinný, výchozí hodnota: 42) – *Seed* pro generátor pseudo-náhodných čísel.
- **<path_in>** (povinný) – Cesta ke zdrojovému obrazu ve formátu `.png`.
- **<path_out>** (nepovinný) – Cesta k výstupnímu obrazu ve formátu `.png`. Budou zde vizualizovány nalezené hrany.

Program po nalezení hran vypíše souřadnice každého z rohů výsledného polygonu na standardní výstup ve formátu `<coord_x> <coord_y>\n(x4)`. Sestavitelnost a funkčnost programu byla testována na referenčním stroji *merlin*.



Obrázek 11: Sumarizace vývinu IoU vůči počtu iterací pro všechny vzorky kromě slice11.png.
Hlavní linka vyobrazuje μ a errorbar vyobrazuje 95 % interval spolehlivosti.



Obrázek 12: Vývin IoU pro vzor slice11, jehož hrana leží mimo obraz (vlevo) a model nalezený algoritmem (vpravo).

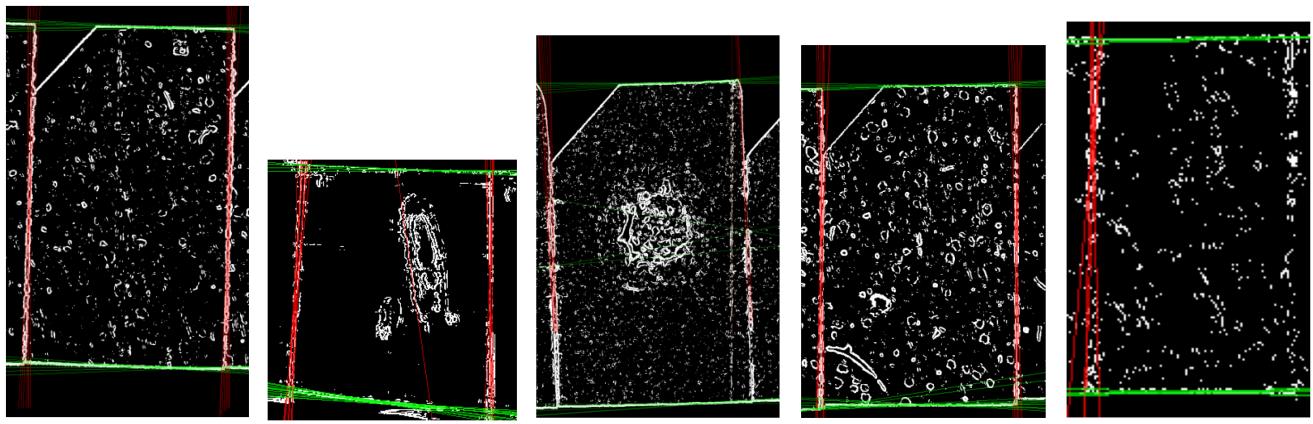
9 Prostor pro zlepšení

Přestože implementaci považuji za úspěšnou, stále existuje prostor pro zlepšení v každé dílčí části této práce. Za nejvíce problematický považuji samotný způsob hledání přímek. I když jsem algoritmus do jisté míry optimalizoval a nakonec se projevil jako robustní, stále stojí za zvážení využití některého z konvenčních způsobů hledání čar v obraze, především Houghovy transformace, viz experiment na obrázcích 13. Vidím zde potenciál na zkrácení výpočetního času, především použitím některé z vysokonáložnostních variant, např. PCLines [3].

I když se vrátíme k předzpracování snímků, vidím prostor pro zlepšení odšumovací techniky. Momentálně je plošně aplikován gaussovský filtr se speciálními parametry, ovšem tento přístup neuvažuje několik předpokladů, které lze o vstupním obrazu udělat. Jako inspiraci můžu např. uvést, že nejvíce nás při vzorkování zajímají právě oblasti dané gaussovským rozložením pravděpodobnosti se speciálními parametry, použitým při vzorkování bodů. Toto rozložení zhruba vymezuje, kde se v obraze nachází hrany a jinde (ve středu a na okrajích) je nejvíce pouze šum.

Zřejmě omezení se týká vzorů, kde se hrany nacházejí mimo obraz. Je sice diskutabilní, zda je v takových případech vůbec produktivní řezy lokalizovat, ovšem pro úplnost je nutné zmínit, že by techniku, která toto alespoň do jisté míry realizuje, šlo implementovat. Protože to však vyžaduje extrakci kontextu z celého obrazu a celkové „porozumění“ toho, jak řezy vypadají, intuitivně řešení tohoto problému gravituje ke strojovému učení.

V neposlední řadě by výsledkům prospěla akvizice adekvátního objemu testovacích dat, což by umožnilo skutečně empiricky vyhodnotit, jak se algoritmus chová. Kromě vyhodnocení by to mohlo vést také k odladění výchozích hodnot parametrů tak, aby program šlo co nejjednodušši používat na arbitrárních vzorech.



Obrázek 13: 20 nejlepších kandidátních přímek identifikovaných Hough. transformací pro $\theta_1 \in (-20^\circ, 20^\circ)$ a $\theta_2 \in (80^\circ, 100^\circ)$.

10 Zdroje

Čerpal jsem z internetových zdrojů a jedné přednášky do kurzu ZPO.

Reference

- [1] Vítězslav Beran. Bodové transformace obrazu. Přednáška v rámci kurzu Zpracování obrazu, Fakulta informačních technologií, Vysoké učení technické, Brno, 2025. Dostupné z: https://moodle.vut.cz/pluginfile.php/871396/course/section/99100/zpo_point_image_transforms.cs.pdf.
- [2] Dagmar Danihelová. Array tomography-slice recognition. Diplomová práce, Masarykova univerzita, Fakulta informatiky, Brno, 2024.
- [3] Markéta Dubská, Adam Herout, and Jiří Havel. Pclines—line detection using parallel coordinates. In *CVPR 2011*, pages 1489–1494. IEEE, 2011.
- [4] Yanming Guo, Yu Liu, Theodoros Georgiou, and Michael S Lew. A review of semantic segmentation using deep neural networks. *International journal of multimedia information retrieval*, 2018.
- [5] Kristina D Micheva and Stephen J Smith. Array tomography: a new tool for imaging the molecular architecture and ultrastructure of neural circuits. *Neuron*, 2007.
- [6] Nobuyuki Otsu et al. A threshold selection method from gray-level histograms. *Automatica*, 1975.
- [7] Satoshi Suzuki et al. Topological structural analysis of digitized binary images by border following. *Computer vision, graphics, and image processing*, 30(1):32–46, 1985.
- [8] Vladimíra Valíčková. Array tomography: Rekonstrukce 3D obrazu. Diplomová práce, Masarykova univerzita, Fakulta informatiky, Brno, 2020.
- [9] S Vijayarani and M Vinupriya. Performance analysis of canny and sobel edge detection algorithms in image mining. *International Journal of Innovative Research in Computer and Communication Engineering*, 2013.