VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ
ÚSTAV MATEMATIKY
FACULTY OF MECHANICAL ENGINEERING
INSTITUTE OF MATHEMATICS

DETEKCE A ROZPOZNÁVÁNÍ KMENE STROMŮ V OBRAZECH

TREE TRUNK DETECTION AND RECOGNITION IN IMAGES

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR
FILIP ŠALOMON

VEDOUCÍ PRÁCE
SUPERVISOR
Mgr. JANA PROCHÁZKOVÁ, Ph.D.

BRNO 2023

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA STROJNÍHO INŽENÝRSTVÍ

Zadání bakalářské práce

Ústav:	Ústav matematiky
Student:	**Filip Šalomon**
Studijní program:	Matematické inženýrství
Studijní obor:	bez specializace
Vedoucí práce:	**Mgr. Jana Procházková, Ph.D.**
Akademický rok:	2022/23

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách a se Studijním a zkušebním řádem VUT v Brně určuje následující téma bakalářské práce:

Detekce a rozpoznávání kmene stromů v obrazech

Stručná charakteristika problematiky úkolu:

Student se seznámí se základními algoritmy z analýzy obrazu pro clusterování a vyhledávání objektů v obraze. Poté pomocí vybrané navržené metody určí, kde jsou v obraze kruhovité tvary značící kmeny stromů a vypočítá jejich obvod pro použití v lesnické praxi.

Cíle bakalářské práce:

1. Seznámení se s algoritmy zpracování obrazu, načtení obrazu, konvolucí, morfologickými operacemi.
2. Návrh a pochopení principu algoritmu RANSAC.
3. Výběr vhodné metody pro detekci objektů v obraze a clusterovací algoritmus.
4. Implementace na reálných datech získaných ve spolupráci s Výzkumným ústavem Silva Tarouc y pro krajinu a okrasné zahradnictví, veřejná výzkumná instituce (VÚKOZ).

Seznam doporučené literatury:

FISCHLER, M. A. a R. C. BOLLES. Random sample consensus. Communications of the ACM. 1981, 24(6), 381-395. ISSN 0001-0782. Dostupné z: doi:10.1145/358669.358692

GONZALEZ, R. C. a R. E. WOODS. Digital image processing. 3rd ed. Upper Saddle River, N.J.: Prentice Hall, c2008. ISBN 978-0-13-168728-8.

LAMIROY, B., L. FRITZ a O. GAUCHER. Robust Circle Detection. Ninth International Conference on Document Analysis and Recognition (ICDAR 2007). IEEE, 2007, 2007, 526-530. ISBN 0-7695-2822-8. ISSN 1520-5363. Dostupné z: doi:10.1109/ICDAR.2007.4378765

Termín odevzdání bakalářské práce je stanoven časovým plánem akademického roku 2022/23

V Brně, dne

L. S.

doc. Mgr. Petr Vašík, Ph.D. ředitel ústavu	doc. Ing. Jiří Hlinka, Ph.D. děkan fakulty

Abstrakt

Tato práce se zabývá detekcí kruhovitých tvarů v obrazech – řezech skenů lesa. Pro segmentaci obrazu je použit algoritmus Freemanova řetězce. Pro měření průměrů je použit algoritmus Ransac (Random Sample Consensus). Navržený algoritmus je testován na datech z Žofínského pralesa.

Abstract

This thesis deals with detection of circular shapes in images – forest scan cross-sections. Freeman chain code algorithm is used for image segmentation. Ransac (Random Sample Consensus) algorithm is used for diameter measurement. The designed algorithm is tested on data from Žofín Forest.

Klíčová slova

Ransac, Freemanův řetězec, analýza obrazu, konvoluce, morfologické operace

Keywords

Ransac, Freeman chain code, image analysis, convolution, morphological operations

ŠALOMON, Filip. *Detekce a rozpoznávání kmene stromů v obrazech*. Brno, 2023. Dostupné také z: https://www.vut.cz/studenti/zav-prace/detail/148800. Bakalářská práce. Vysoké učení technické v Brně, Fakulta strojního inženýrství, Ústav matematiky. Vedoucí práce Mgr. Jana Procházková, Ph.D.

Prohlašuji, že jsem bakalářskou práci *Detekce a rozpoznávání kmene stromů v obrazech* vypracoval samostatně pod vedením Mgr. Jany Procházkové, Ph.D. s použitím materiálů uvedených v seznamu literatury.

Filip Šalomon

Poděkování patří Davidu Janíkovi, Kamilu Královi a Martinu Krůčkovi za téma bakalářské práce a příležitost spolupracovat. Martinu Krůčkovi dále patří velké díky za poskytnutí testovacích dat.
Největší poděkování patří vedoucí práce Janě Procházkové za výborné vedení s nepřehlédnutelným zájmem a přínosné konzultace vždy v dobrém duchu.

Filip Šalomon

Obsah

1 Úvod

V pralese jsou stromy. Stromy přitahují pozornost mimo jiné i Výzkumného ústavu Silva Taroucy pro krajinu a okrasné zahradnictví – konkrétně Odboru ekologie lesa. Tito odborníci vystupují pod názvem Pralesní tým Modrá kočka a ve snaze zachytit co nejvěrněji prostorovou situaci lesa využívají 3D laserového skenování. Pro zpracování skenů vyvíjejí vlastní open-source software zvaný 3D Forest.

Základním parametrem, který se u stromů sleduje, je průměr v prsní výšce, tzv. DBH. Existuje řada algoritmů pro získání DBH ze skenů, mezi nimi je i *Random Sample Consensus* (Ransac), který ve zmiňovaném programu 3D Forest není. Implementace Ransacu na skenech lesa tvoří základ tato práce.

Vstupem do mého programu je obraz – řez skenem ve výšce 130 cm nad zemí. Snahou je detekovat kruhovité tvary a změřit jejich velikost. Výstupem jsou pak pozice a průměry stromů.

První část práce je o obrazu samotném, jak se s obrazem pracuje a jak lze obraz upravovat pomocí konvoluce nebo morfologických operací. Dále jsou představeny algoritmy pro rozčlenění obrazu (Freemanův řetězec) a hledání kruhovitých tvarů (Ransac).

V další části je navržen a sestaven výsledný algoritmus podle povahy dat, se kterými se bude pracovat. Významná pozornost je věnována aplikaci algoritmu Ransac a jeho implementaci v programovacím jazyce C++.

Nakonec je navržený algoritmus testován na datech, která poskytl Ing. Martin Krůček, Ph.D. z Odboru ekologie lesa. Data jsou z jádrové oblasti národní přírodní rezervace Žofínský prales.

2 Teorie

V teoretické části této práce zavedeme obraz a jak s ním pracovat. Uvedeme jak obraz upravovat pomocí konvoluce nebo morfologických operací. Představíme algoritmus Freemanova řetězce a jeho použití při hledání obrysů. Nakonec uvedeme algoritmus Random Sample Consensus (Ransac) a jeho použití při zpracování dat. Zdroje pro tuto část jsou [1–6].

2.1 Obraz

V této kapitole zavedeme obraz a jeho reprezentaci pomocí obrazové matice. Také zde zmíníme způsoby převodu obrazu mezi jednotlivými typy. Tato kapitola vychází z [1, 2].

Definice 2.1.1 (Spojitý obraz)**.** Mějme funkci dvou proměnných $f(x, y)$, kde $x, y \in \mathbb{R}$. Nabývá-li funkce hodnot z nějakého intervalu $\langle 0, \omega - 1\rangle$, $\omega \in \mathbb{N}$, hovoříme o spojitém obrazu v odstínech šedi (příp. o obrazu v ω odstínech šedi).

Takto formulovaná definice může reprezentovat skutečný obraz, avšak digitální zobrazovací zařízení ze své podstaty vyžadují diskrétní definiční obor a diskrétní obor hodnot. Proto zavedeme přísnější definici digitálního obrazu s jistými předpoklady. Jedním z nich je rozsah hodnot, kterých souřadnice x a y nabývají – tedy znalost velikosti obrazu.

Definice 2.1.2 (Digitální obraz)**.** Nechť $m, n \in \mathbb{N}$ a $f(x, y)$ je obraz. Jestliže $x \in \{1, 2, \dots m\}$, $y \in \{1, 2, \dots n\}$ a $f \in \{0, 1, \dots \omega - 1\}$, řekneme, že f je digitální obraz v odstínech šedi.

Nyní máme obraz konečné velikosti s přirozenými hodnotami. Takový obraz lze reprezentovat pomocí maticového zápisu, tzv. *obrazové matice*:

$$\begin{pmatrix} a_{11} & a_{12} & \dots & a_{1y} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2y} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ a_{x1} & a_{x2} & \dots & a_{xy} & \dots & a_{xn} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{my} & \dots & a_{mn} \end{pmatrix}, \; f(x, y) = a_{xy}. \tag{2.1.1}$$

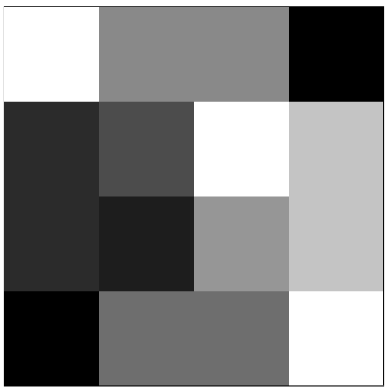
Prvky matice a_{xy} nazýváme pixely. Výraz pochází z anglického *picture element* (prvek obrázku, nebo obrázková jednotka).

Na obrázku 1 je příklad obrazu v odstínech šedi, kde maximální hodnota je 255 (tj. $\omega - 1$). Zápis pro tento případ ukazuje matice:

$$\begin{pmatrix} 255 & 137 & 137 & 0 \\ 43 & 76 & 255 & 196 \\ 43 & 29 & 150 & 196 \\ 0 & 110 & 110 & 255 \end{pmatrix}. \tag{2.1.2}$$

Stále jsme však omezeni na šedou barvu. Chtěli bychom-li obraz barevný, přidáme další rozměr.

Definice 2.1.3 (Barevný obraz)**.** Mějme tři digitální obrazy se stejnými hodnotami x, y, ω. Potom tuto trojici obrazů nazýváme (digitálním) barevným obrazem (r, g, b). Kde r reprezentuje červený (red) kanál, g zelený (green) a b modrý (blue).



Obrázek 1: Obraz v odstínech šedi. $\omega = 256$.

Maticový zápis barevného obrazu, jak jsme před chvílí nastínili, spočívá v použití třetí dimenze (viz (2.1.3)). Namísto číselné indexace třetího rozměru využijeme příslušných písmen, tj. $r := 1, g := 2, b := 3$.

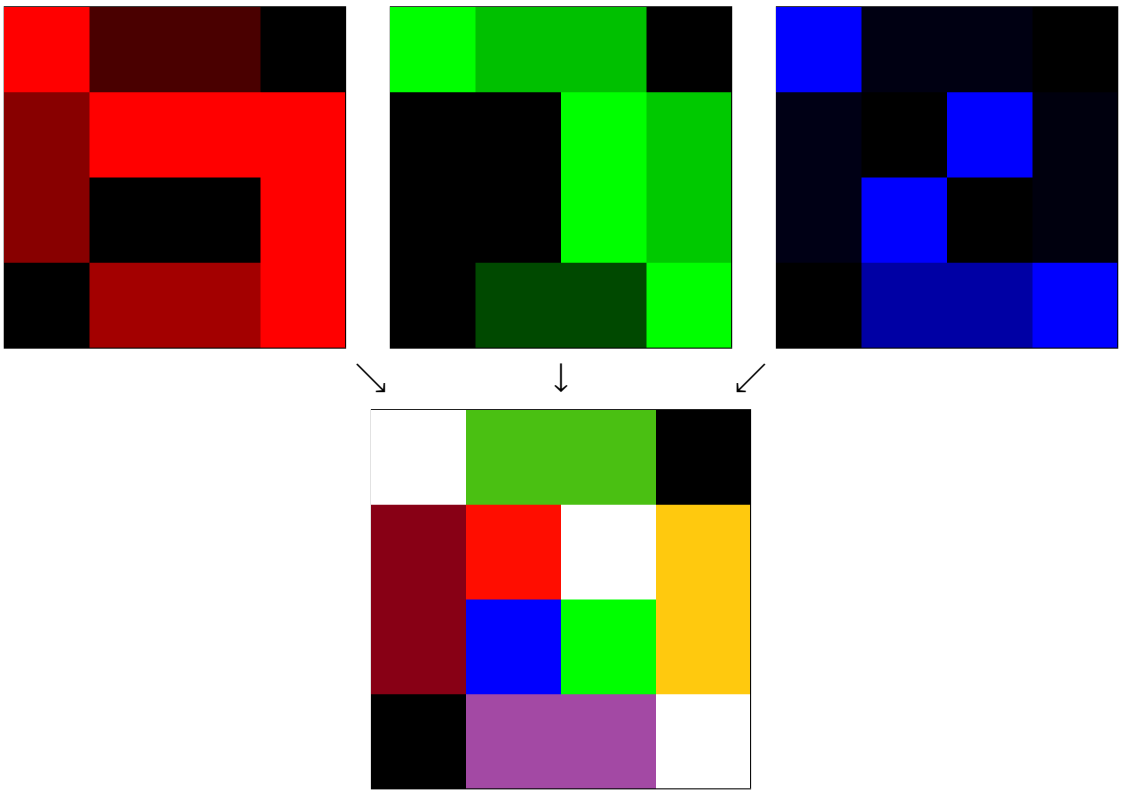
$$\boldsymbol{R} = \begin{pmatrix} a_{11r} & a_{12r} & \dots & a_{1yr} & \dots & a_{1nr} \\ a_{21r} & a_{22r} & \dots & a_{2yr} & \dots & a_{2nr} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ a_{x1r} & a_{x2r} & \dots & a_{xyr} & \dots & a_{xnr} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ a_{m1r} & a_{m2r} & \dots & a_{myr} & \dots & a_{mnr} \end{pmatrix}$$

$$\boldsymbol{G} = \begin{pmatrix} a_{11g} & a_{12g} & \dots & a_{1yg} & \dots & a_{1ng} \\ a_{21g} & a_{22g} & \dots & a_{2yg} & \dots & a_{2ng} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ a_{x1g} & a_{x2g} & \dots & a_{xyg} & \dots & a_{xng} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ a_{m1g} & a_{m2g} & \dots & a_{myg} & \dots & a_{mng} \end{pmatrix} \tag{2.1.3}$$

$$\boldsymbol{B} = \begin{pmatrix} a_{11b} & a_{12b} & \dots & a_{1yb} & \dots & a_{1nb} \\ a_{21b} & a_{22b} & \dots & a_{2yb} & \dots & a_{2nb} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ a_{x1b} & a_{x2b} & \dots & a_{xyb} & \dots & a_{xnb} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ a_{m1b} & a_{m2b} & \dots & a_{myb} & \dots & a_{mnb} \end{pmatrix}$$

Na obrázku 2 je příklad spojení tří barevných kanalů v barevný obraz s maximální hodnotou 255 (tj. $\omega - 1$). Zápis pro tento případ ukazuje matice:

$$\boldsymbol{R} = \begin{pmatrix} 255 & 74 & 74 & 0 \\ 136 & 255 & 255 & 255 \\ 136 & 0 & 0 & 255 \\ 0 & 163 & 163 & 255 \end{pmatrix}, \ \boldsymbol{G} = \begin{pmatrix} 255 & 192 & 192 & 0 \\ 0 & 0 & 255 & 201 \\ 0 & 0 & 255 & 201 \\ 0 & 73 & 73 & 255 \end{pmatrix}, \ \boldsymbol{B} = \begin{pmatrix} 255 & 18 & 18 & 0 \\ 21 & 0 & 255 & 14 \\ 21 & 255 & 0 & 14 \\ 0 & 164 & 164 & 255 \end{pmatrix}. \tag{2.1.4}$$



Obrázek 2: Spojení tří barevných kanálů v barevný obraz. $\omega = 255$.

Nyní máme definované obrazy v odstínech šedi a barevné. Zbývá doplnit, jak získat obraz binární. Binární obraz obsahuje jen dvě barvy, např. bílou a černou barvu, žádnou šedou. Proto jej lze snadno získat z obrazu v odstínech šedé prahováním.

Definice 2.1.4 (Prahování)**.** Mějme obraz $f(x, y)$ v odstínech šedé, číslo t z oboru hodnot obrazu f. Binární obraz g získáme pomocí následujícího vztahu.

$$g\left(f(x,y)\right) = \begin{cases} 0, & f(x,y) < t, \\ 1, & \text{jinak.} \end{cases}$$

Číslo t nazýváme prahem a tento proces prahováním. Čísla 0 a 1 reprezentují libovolné dvě barvy.

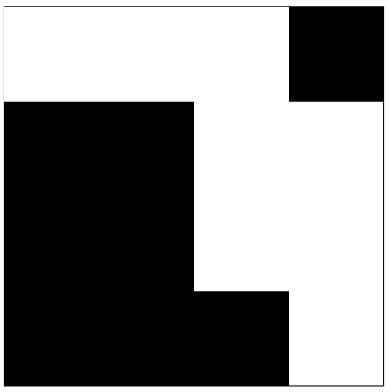
Poznámka. Nerovnost $<$ v definici (2.1.4) lze nahradit za $\leq$ bez výrazné změny.

Určit vhodný práh je důležité pro zachování rozlišitelnosti objektů či pro odlišení pozadí od popředí. Lze využít např. Otsuovy metody[1], která stojí na analýze histogramu obrazu.

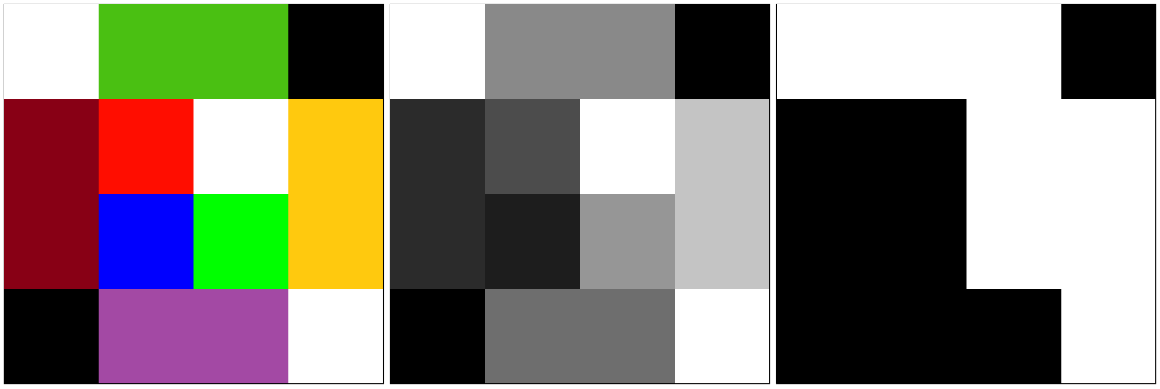
Obrázek 3 je příklad binárního obrazu. Hodnoty pixelů je možné vidět v matici:

$$\begin{pmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{2.1.5}$$

[1]OTSU, Nobuyuki. A Threshold Selection Method from Gray-Level Histograms. IEEE Transactions on Systems, Man, and Cybernetics [online]. 1979, 9(1), 62-66 [cit. 2022-11-06]. ISSN 0018-9472. Dostupné z: doi:10.1109/TSMC.1979.4310076



Obrázek 3: Binární obraz.



Obrázek 4: Barevný obraz (vlevo), převedený do odstínů šedé (uprostřed), dále převedený do binárního obrazu (vpravo).

Víme, jak získat binární obraz z odstínů šedé. Zbývá otázka, jak získat odstíny šedé z barevného.

Nejjednodušší způsob by pravděpodobně byl zprůměrovat hodnoty všech kanálů pro daný pixel. Tato varianta by ovšem neodpovídala tomu, jak lidské oko vnímá intenzity barev. Proto se používá lineární kombinace s vhodně zvolenými koeficienty. Rovnice (2.1.6) je jedním z několika způsobů výpočtu, který stále není dokonalý. Jestliže bychom se chtěli dostat k přesnějším přepočtům, rovnice se velmi rychle komplikuje.

$$\begin{aligned} a_{xy} &= 0,299 \cdot a_{xyr} + 0,587 \cdot a_{xyg} + 0,114 \cdot a_{xyb}, \\ &\text{kde } a \text{ je prvkem šedé obrazové matice,} \\ &\text{respektive barevné v příslušném kanálu.} \end{aligned} \tag{2.1.6}$$

Hodnota pixelu dle výše uvedené lineární kombinace (2.1.6), nemusí být přirozené číslo. To však lze snadno vyřešit zaokrouhlením.

V této části textu o obrazové matici a obrazech obecně jsme si ukázali, jak vypadá obraz barevný, v odstínech šedé a binární. Viděli jsme, jak je možné jej zapsat do obrazové matice a jak mezi typy obrazů v jednom směru přecházet.

2.2 Konvoluce

Konvoluce je silným nástrojem, který se uplatňuje i v oblasti zpracování obrazu. Obecně se jedná o binární operaci na funkcích, se kterou je možné řešit např. i diferenciální rovnice. Zdrojem této kapitoly je [2].

Jelikož pracujeme s obrazem, využijeme konvoluci dvojrozměrnou, a protože pracujeme s obrazem digitálním, budeme definovat pouze diskrétní dvojrozměrnou konvoluci.

Definice 2.2.1 (Diskrétní dvojrozměrná konvoluce)**.** Mějme dva digitální obrazy $f(x, y)$ a $g(x, y)$. Rozšíříme jejich definiční obory na celé $\mathbb{Z}^2$ tak, že uvažujeme nuly tam, kde původní funkce nejsou definovány. Diskrétní dvojrozměrná konvoluce funkcí f, g je dána následujícím vztahem.

$$(f * g)(x, y) = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} f(i, j) \cdot g(x - i, y - j). \tag{2.2.1}$$

Výsledná funkce diskrétní dvojrozměrné konvoluce se většinou uvažuje na větším z definičních oborů původních funkcí vstupujících do operace. Vstupující obraz s menším definičním oborem se pak nazývá *jádrem konvoluce* nebo *maskou*. Pro praktický výpočet není nutné rozšíření funkcí na celé $\mathbb{Z}^2$, ale stačí podle velikosti masky rozšířit obraz do všech směrů.

Poznámka (Padding). Tomuto rozšíření obrazu se říká *padding* a existuje více možností, jakými hodnotami naplnit nové pixely. Kromě vyplnění nulami je například možné zrcadlit krajní pixely samotného obrazu.

Dříve zmíněná síla konvoluce spočívá ve výběru masky. Obraz je možné rozmazat i zostřit, příp. je možné v obrazu detekovat hrany.

$$\boldsymbol{A} = \frac{1}{9} \cdot \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}, \ \boldsymbol{B} = \begin{pmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{pmatrix}, \ \boldsymbol{C} = \begin{pmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{pmatrix}. \tag{2.2.2}$$

Ve výše uvedené rovnici jsou příklady konvolučních masek. Matice $\boldsymbol{A}$ rozmazává obraz, $\boldsymbol{B}$ zostřuje a $\boldsymbol{C}$ zvýrazňuje hrany. Viz obrázek 5.

Matice $\boldsymbol{A}$ se násobí koeficientem $\frac{1}{9}$, aby byly zachovány barvy. Součet všech prvků matice je pak roven 1, zachovává se tedy celková intenzita obrazu, nedochází k umělému přesvětlení. Zbylé dvě masky koeficient nepotřebují, součet jejich prvků již je 1.

2.3 Morfologické operace

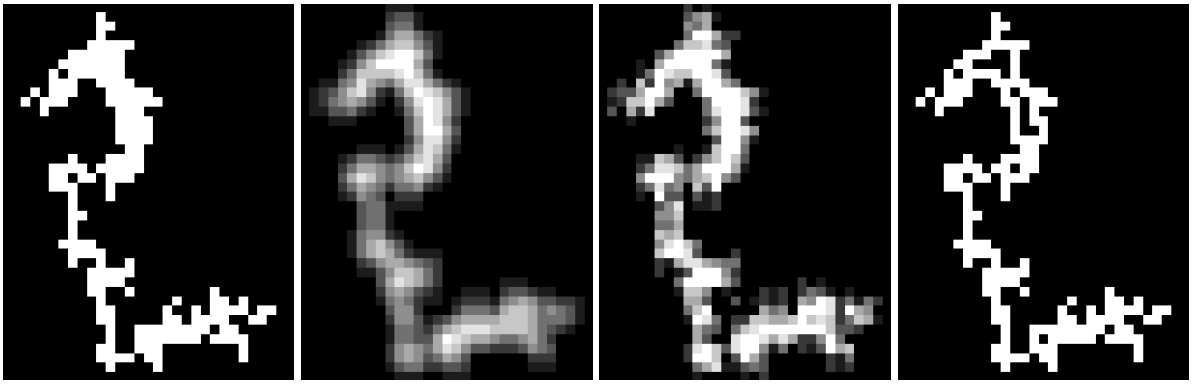
Morfologické zpracovávání obrazu nahlíží na obrazy jako na množiny. V našem případě budeme uvažovat podmnožiny $\mathbb{Z}^2$, kde jednotlivé složky jsou souřadnice v rovině. Dále budeme uvažovat binární obrazy s bílým pozadím a černým popředím, tj. objekty zájmu. Znamená to tedy, že množina obsahuje souřadnice bodů objektu, který nás zajímá, a body pozadí nejsou nikde zaznamenány.

Zdrojem této kapitoly je [2].

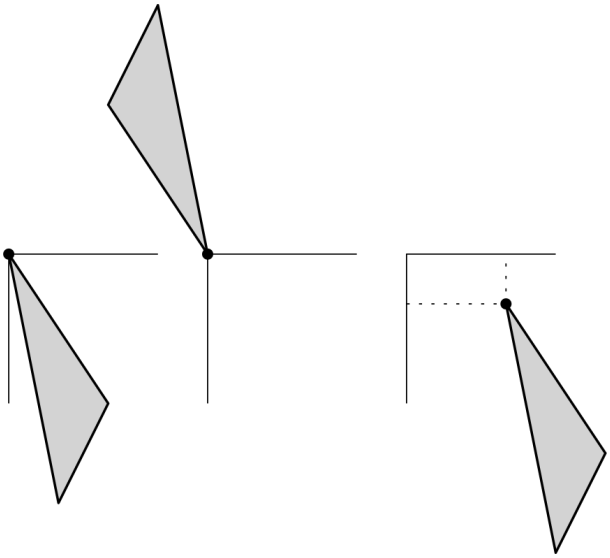
V této části jsou představeny základní binární morfologické operace – eroze a dilatace. Dále jsou uvedeny složené operace opening a closing. Nejprve je však nutné zavést translaci a reflexi.

Definice 2.3.1 (Reflexe)**.** Reflexi (zrcadlení) množiny B značíme $\hat{B}$ a je dána vztahem

$$\hat{B} = \{w \mid w = -b, \ b \in B\}. \tag{2.3.1}$$



Obrázek 5: Použití konvolučních masek z rovnice (2.2.2). Zleva: originální obraz, maska $\boldsymbol{A}$ aplikovaná na originální obraz, maska $\boldsymbol{B}$ aplikovaná na rozmazaný obraz, maska $\boldsymbol{C}$ aplikovaná na originální obraz.



Obrázek 6: Vlevo původní množina B, uprostřed reflexe $\hat{B}$, vpravo translace B_z.

Poznámka. Za předpokladu, že provádíme reflexi na obraze B, kde prvky $b \in B$ jsou souřadnice (x, y), pak $\hat{B}$ obsahuje souřadnice $(-x, -y)$.

Příklad reflexe je na obrázku 6.

Definice 2.3.2 (Translace)**.** Nechť $z \in \mathbb{Z}^2$ a O je počátek $(0, 0)$. Translaci (posunutí) množiny $B \subset \mathbb{Z}^2$ o vektor $(O - z)$ značíme B_z a je dána vztahem

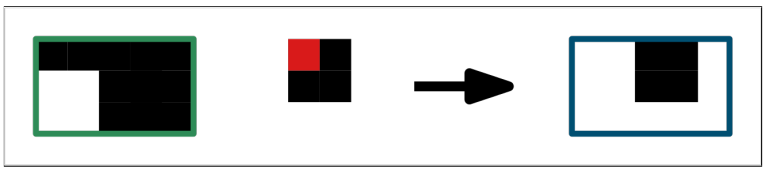
$$B_z = \{w \mid w = b + z,\ b \in B\}\,. \tag{2.3.2}$$

Příklad translace je na obrázku 6.

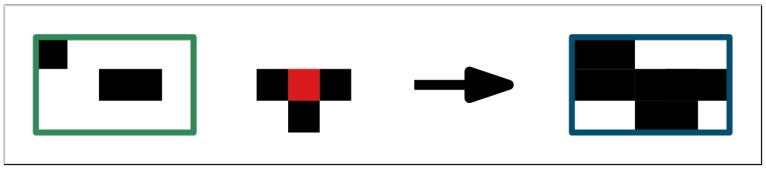
Definice 2.3.3 (Eroze)**.** Mějme $A, B \subset \mathbb{Z}^2$. Erozi množiny A množinou B značíme $A \ominus B$ a je dána vztahem

$$A \ominus B = \{z \mid B_z \subseteq A\}\,. \tag{2.3.3}$$

Eroze zmenšuje objekty a zbavuje výběžků. Příklad je na obrázku 7.



Obrázek 7: Eroze množiny v zeleném rámečku strukturním prvkem s červeným počátkem. Výsledek v modrém rámečku.



Obrázek 8: Dilatace množiny v zeleném rámečku strukturním prvkem s červeným počátkem. Výsledek v modrém rámečku.

Definice 2.3.4 (Dilatace)**.** Mějme $A, B \subset \mathbb{Z}^2$. Dilataci množiny A množinou B značíme $A \oplus B$ a je dána vztahem

$$A \oplus B = \left\{ z \mid (\hat{B})_z \cap A \neq \emptyset \right\}. \tag{2.3.4}$$

Dilatace zvětšuje objekty, vyplňuje díry a mezery. Příklad je na obrázku 8.

Poznámka. Množina B z předchozích definic (2.3.3), (2.3.4) se nazývá *strukturní prvek* (z angl. structure element). Strukturní prvky jsou množiny menší než množiny obrazů samotných a jejich funkce je do jisté míry podobná funkci masek z předchozí kapitoly o konvoluci (str. 17). Pro strukturní prvky je důležitá pozice počátku, která se u symetrických volí většinou v těžišti.

Základní morfologické operace eroze a dilatace se skládají v operace opening a closing. Ty se liší pořadím základních operací.

Definice 2.3.5 (Opening)**.** Opening množiny A strukturním elementem B značíme $A \boxminus B$ a je dán vztahem

$$A \boxminus B = (A \ominus B) \oplus B. \tag{2.3.5}$$

Jedná se o erozi, po které následuje dilatace. Eroze odstraní výběžky a jiné tenké linie, avšak zároveň zmenší objekty. Následná dilatace objekty zvětší na původní velikost. Opening tedy odstraňuje tenké struktury a zachovává velikost objektů. Příklad je na obrázku 9.

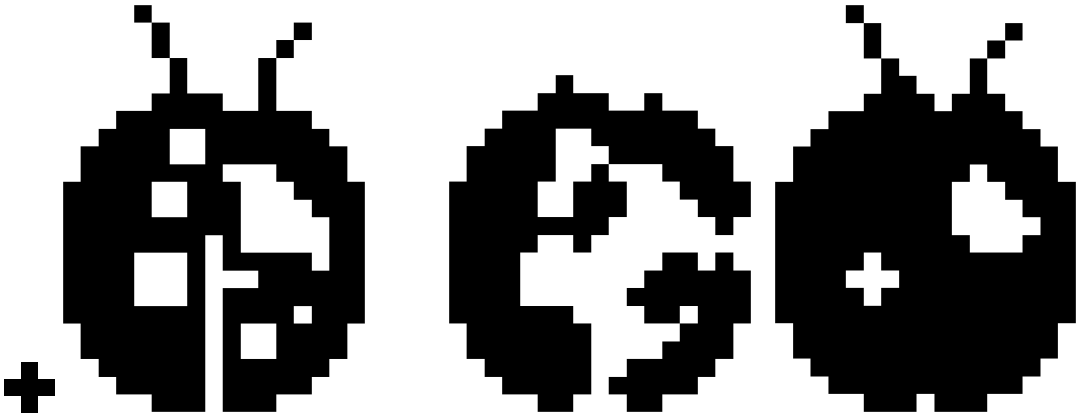
Definice 2.3.6 (Closing)**.** Closing množiny A strukturním elementem B značíme $A \boxplus B$ a je dán vztahem

$$A \boxplus B = (A \oplus B) \ominus B. \tag{2.3.6}$$

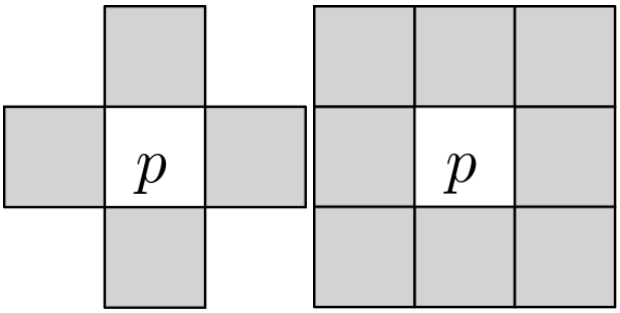
Jedná se o dilataci, po které následuje eroze. Dilatace vyplní díry a mezery a objekty zvětší. Následná eroze objekty zmenší na původní velikost. Closing tedy vyplňuje díry a zachovává výběžky bez změny velikosti objektů. Příklad je na obrázku 9.

2.4 Freemanův řetězec

V této kapitole bude představen algoritmus Freemanova řetězce (též Freeman chain code). Nejprve však zavedeme okolí pixelu. Zdroje této kapitoly jsou [2–5].



Obrázek 9: Zleva strukturní element, původní množina, opening a closing.



Obrázek 10: 4-okolí pixelu p vlevo a 8-okolí vpravo.

Okolí

Definice 2.4.1 (4-okolí)**.** Mějme pixel p se souřadnicemi (x, y), pak množinu pixelů $\{(x+1, y), (x, y+1), (x-1, y), (x, y-1)\}$ nazýváme 4-okolí pixelu p a značíme $N_4(p)$.

Příklad 4-okolí je na obrázku 10 vlevo.

Definice 2.4.2 (D-okolí)**.** Mějme pixel p se souřadnicemi (x, y), pak množinu pixelů $\{(x+1, y+1), (x-1, y+1), (x-1, y-1), (x+1, y-1)\}$ nazýváme D-okolí pixelu p a značíme $N_D(p)$.

Definice 2.4.3 (8-okolí)**.** Mějme pixel p se souřadnicemi (x, y), pak množinu pixelů $N_4(p) \cup N_D(p)$ nazýváme 8-okolí pixelu p a značíme $N_8(p)$.

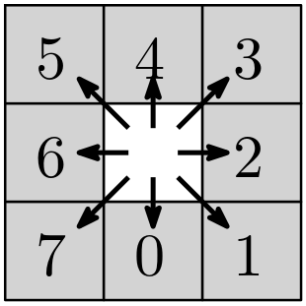
Příklad 8-okolí je na obrázku 10 vpravo.

Poznámka. V případě, kdy je pixel na hranici obrazu, by jeho okolí zahrnovalo pixely mimo obraz. Tomu lze předejít požadavkem na průnik daného okolí s pixely obrazu.

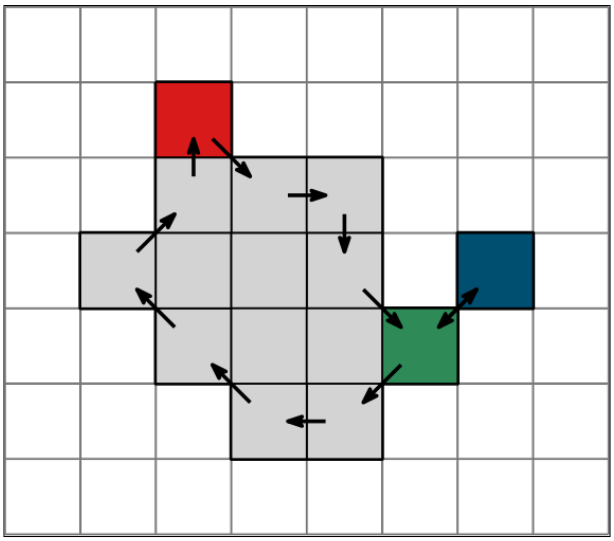
Freemanův řetězec

Freemanův řetězec slouží k popisu obrysu objektu v binárním obraze. Řetězec nese informaci o směrech, ve kterých se musíme přesunovat z pixelu na pixel, abychom se po hranici objektu dostali zpět na počáteční pixel. V řetězci nenalezneme informaci o souřadnicích pixelů, avšak stačí znalost jednoho (např. prvního), abychom zjistili všechny.

Pro jednotlivé přesuny z pixelu na pixel se využívá 8-okolí. Označíme směry dle obrázku 11.



Obrázek 11: Číselně označené směry na 8-okolí.



Obrázek 12: Detekce hranice objektu pomocí algoritmu Freemanova řetězce. Počáteční pixel je červený. Výsledný řetězec: [1, 2, 0, 1, 3, 7, 7, 6, 5, 5, 3, 4].

Algoritmus začíná na libovolném hraničním pixelu. Pro naše účely budeme uvažovat, že jsme obraz procházeli po řádcích zleva doprava a shora dolů, dokud jsme se nedostali na pixel objektu. Z povahy způsobu hledání počátečního pixelu víme, že ve směrech 3,4,5 a 6 od počátečního pixelu není žádný pixel objektu. Zvolme směr procházení hranice ve směru hodinových ručiček. Dále se kontroluje 8-okolí počátečního pixelu. Začneme ve směru 2, pokud nenalezneme pixel objektu, tak v dalším kroku ve směru 1, poté případně 0, pak 7... Když nalezneme pixel objektu, tak se na něj přesuneme a do řetězce zaznamenáme směr, ve kterém jsme se posunuli. Dále prohledáváme 8-okolí nového pixelu a začínáme ve směru
[směr posunutí + 2, mod 8]. Opět v případě neúspěchu kontrolujeme směry sestupně.

Po každém kroku, tj. změny kontrolovaného směru, nebo přesunu na nový pixel (také se mění kontrolovaný směr) se provádí kontrola, zda je aktuální pixel stejný jako počáteční a zároveň zda aktuální kontrolovaný směr je stejný jako počáteční (v našem případě 2). Tyto podmínky zaručují ukončení algoritmu po průchodu celou hranicí objektu.

Na obrázku 12 jsou vyznačeny směry procházení hranice objektu s počátečním pixelem označeným červeně. Přechod ze zeleného pixelu na modrý ukazuje, proč je nutné v algoritmu kontrolovaný směr zvýšit o 2 ne pouze o 1.

Poznámka. Pomocí řetězce je možné odhadovat délku obvodu objektu tak, že sečteme počet sudých směrů a $\sqrt{2}$-násobek počtu lichých směrů. Podrobná analýza tohoto odhadu je v [5].

2.5 Ransac

Název algoritmu Ransac pochází z anglického jazyka. Random sample consensus, volně přeloženo jako shoda náhodných vzorků, je metoda zpracování dat, přesněji proložení experimentálních dat teoretickým modelem. Velkou předností této metody je práce na datech se značným šumem (příp. chybami), avšak za cenu toho, že není jisté dosažení dobrého výsledku. Z názvu totiž vyplývá, že se využívá náhodného výběru. Zdrojem této kapitoly je [6].

Popis algoritmu Ransac

Mějme model M, který vyžaduje n bodů pro určení parametrů. Dále mějme množinu P naměřených bodů takovou, že $\text{card}(P) \geq n$.[2] Náhodně vyberme podmnožinu $S_1 \subseteq P$ tak, že $\text{card}(S_1) = n$. Pomocí prvků z S_1 stanovme parametry modelu M_1. Vytvořme *shodovou* podmnožinu $S_1^* \subseteq P$ tak, že prvky z S_1^* jsou ve zvolené chybové toleranci od modelu M_1.
Jestliže $\text{card}(S_1^*) \geq t$, kde t je zvolený práh závislý na předpokládaném množství chybových bodů, použijme množinu S_1^* pro nový model M_1^* (např. pomocí metody nejmenších čtverců).
Jestliže $\text{card}(S_1^*) < t$, náhodně vyberme novou podmnožinu $S_2 \subseteq P$ a opakujme algoritmus. Pokud po zvoleném počtu pokusů nedospěje algoritmus ke vhodné shodové množině, vytvořme model s největší shodovou množinou, nebo ukončeme proces neúspěchem.

Zapsáno symbolicky:
Model M (n bodů), množina $P : \text{card}(P) \geq n$.
1. $S_1 \subseteq P$: $\text{card}(S_1) = n$.
2. $S_1 \longrightarrow M_1$.
3. $S_1^* \subseteq P$: $s \in S_1^*$: $\|s - M_1\| \leq \varepsilon$.
4. $\text{card}(S_1^*) \geq t \quad \Rightarrow \quad S_1^* \longrightarrow M_1^*$.
- $\text{card}(S_1^*) < t \quad \Rightarrow \quad 1.\ S_2 \ldots$

Poznámka. V závislosti na aplikaci algoritmu a požadované přesnosti lze jako výsledný model považovat i M_i, pokud je splněna prahová podmínka $\text{card}(S_i^*) \geq t$.

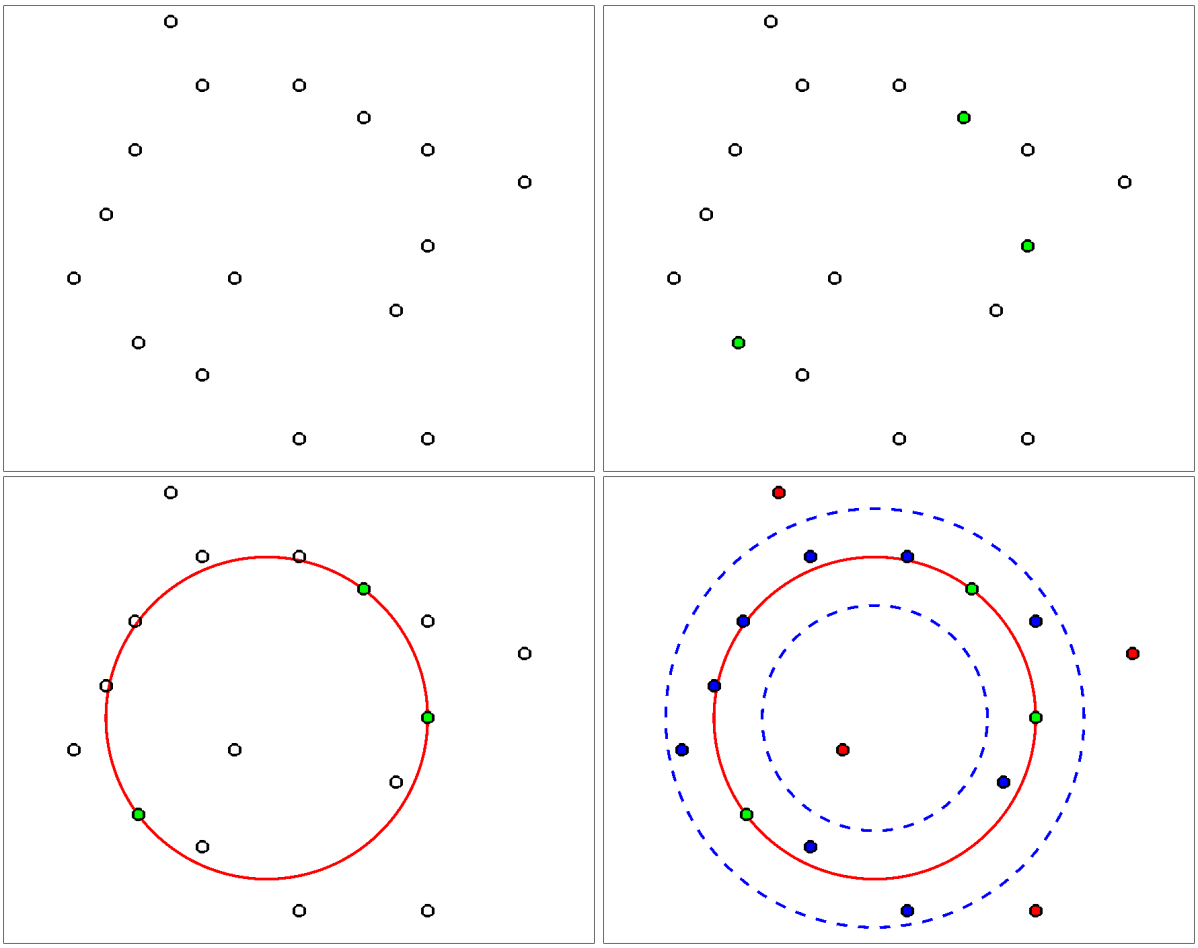
Ukažme si na příkladu, jak by algoritmus fungoval. Začneme s množinou 16 bodů v rovině podle obrázku 13. Náhodně vybereme požadovaný počet bodů. Jelikož naším modelem je kružnice, vybereme tři body (zelené). Tím máme jednoznačně určenou kružnici (červená). Mezikruží ohraničené čárkovanými kružnicemi (modré) vymezuje zvolenou chybovou toleranci. Body, které leží uvnitř mezikruží, nazýváme *inliers* (z angl. volně přeloženo jako náležící body). Body, které leží mimo mezikruží, nazýváme *outliers* (z angl. volně přeloženo jako mimoležící body). V našem případě máme 12 inlierů. Je-li pro naše účely tento počet inlierů nedostatečný, začíná se znovu s novými třemi náhodně vybranými body. Splňuje-li počet inlierů prahovou podmínku, lze za výsledný model považovat červenou kružnici, nebo množinu všech inlierů (dříve označenou jako shodovou množinu) pomocí metody nejmenších čtverců proložit novou kružnicí.

V poslední větě popisu algoritmu (2.5) je zmínka o zvoleném počtu pokusů. Tento počet pokusů je možné určit následujícím výpočtem, jehož odvození je v [6].

$$I = \frac{\log(1-p)}{\log\left(1-(1-\overline{p})^n\right)}, \tag{2.5.1}$$

kde I je hledaný maximální počet pokusů potřebný k nalezení alespoň jednoho dobrého modelu s pravděpodobností úspěchu p. Dále n je počet bodů nutných k sestavení modelu a $\overline{p}$ je pravděpodobnost, že náhodně vybraný bod je outlier.

[2] Mějme množinu M, pak $\text{card}(M)$ vyjadřuje počet prvků množiny M.



Obrázek 13: Příklad použití algoritmu Ransac. Zleva nahoře: vstupní body, náhodný výběr tří bodů, proložení kružnicí, stanovení inlierů a outlierů.

Navážeme-li na příklad s kružnicemi, víme, že $n = 3$, podle dostupných dat odhadneme $\overline{p} = 0,2$ a zvolíme $p = 0,99$.

$$I = \frac{\log(1 - 0,99)}{\log\left(1 - (1 - 0,2)^3\right)} = 6,42$$

Po zaokrouhlení nahoru nám výpočet říká, že s 99% pravděpodobností dostaneme v 7 pokusech alespoň jeden dobrý model, tj. model splňující námi zvolené podmínky.

Poznámka. Číslo I lze také použít ne jako maximální dovolený počet pokusů, ale jako počet pokusů, které se provedou. Z dobrých modelů se nakonec vybere ten nejlepší. Vybírání může být například na základě počtu inlierů, nebo podle průměrné vzdálenosti inlierů od modelu.

Obrázek 14: Point-cloud z *3D Forest.* Dostupné z: www.3dforest.eu/

3 Implementace

V kapitole se zaměříme na použití zavedeného matematického aparátu z předchozí kapitoly za účelem nalezení kmenů stromů a změření průměrů (příp. obvodů). Pojednáme o snaze zbavit se šumu a zvýraznit žádoucí tvary, o segmentaci a o následném měření velikosti kmenů. Nejprve se však seznámíme s daty a jejich původem.

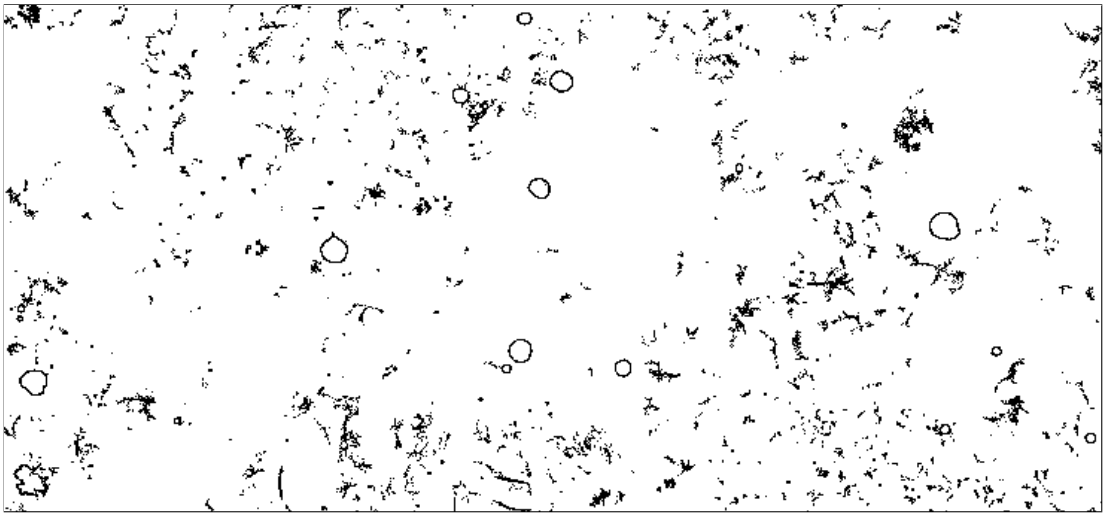
3.1 Původ dat

Pro zachycení prostorové situace v lese se hojně využívá technologie 3D laserového skenování [7–9]. Princip laserového skeneru – *lidaru*[3] – spočívá v měření času, za který se paprsek vyslaný ze zařízení vrátí po odrazu od skenovaného objektu zpět. Paprsky jsou zařízením vysílány do mnoha směrů pomocí soustavy zrcátek. Samotný skener je pak instalován např. na stativ pro pozemní skenování, nebo na dron pro letecké skenování.

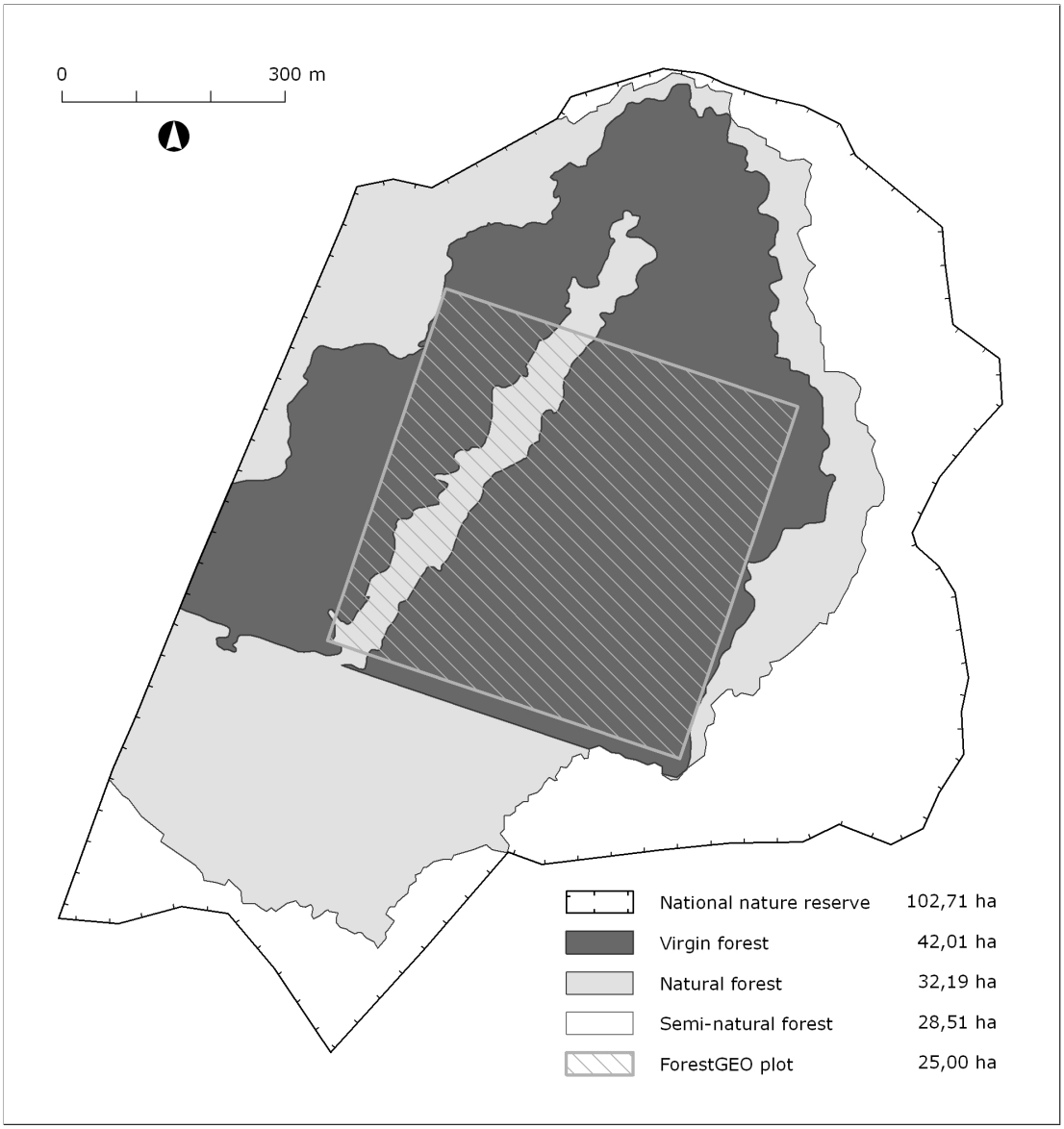
Výstupem této metody je point-cloud (česky mračno bodů, obrázek 14), který je nutné dále zpracovat. Pro účely této práce vyžadujeme obraz, proto point-cloudem vedeme řez. Řez je ve výšce 130 cm nad zemí. Znamená to, že budeme měřit tzv. DBH – angl. diameter at breast height, tj. průměr v prsní výšce. Dále zpracovávat tedy budeme binární obraz, jehož příklad lze vidět na obrázku 15.

Data zpracovávaná v této práci poskytl Odbor ekologie lesa, VÚKOZ[4]. Skeny pochází z jádrové oblasti Žofínského pralesa – národní přírodní rezervace v Novohradských horách. Na obrázku 16 je jádrová oblast označena jako „ForestGEO plot“ vyšrafována.

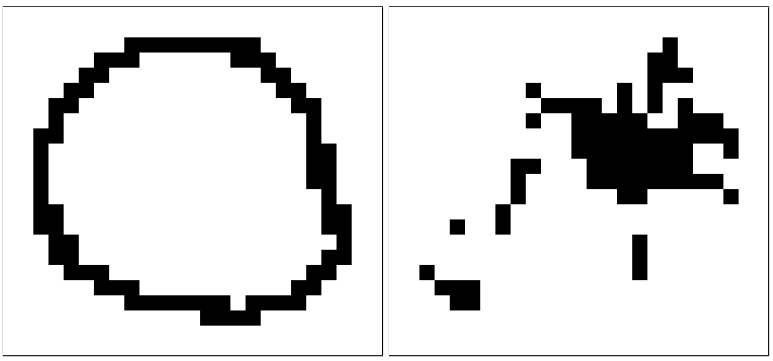
[3] lidar – Laser imaging, detection, and ranging.
[4] VÚKOZ – Výzkumný ústav Silva Tarouc y pro krajinu a okrasné zahradnictví, v.v.i.



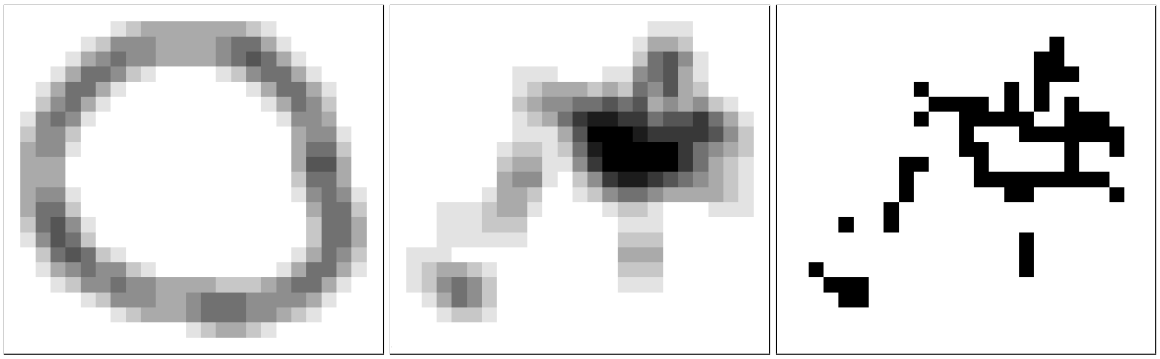
Obrázek 15: Řez point-cloudem ve výšce 130 cm. Sken z Žofínského pralesa.



Obrázek 16: Mapa Žofínského pralesa. [11]



Obrázek 17: Výřezy z obrázku 15. Vlevo žádoucí tvar. Vpravo nežádoucí.



Obrázek 18: Rozmazání a detekce hran aplikovány na obrázek 17.

.

3.2 Úprava obrazu

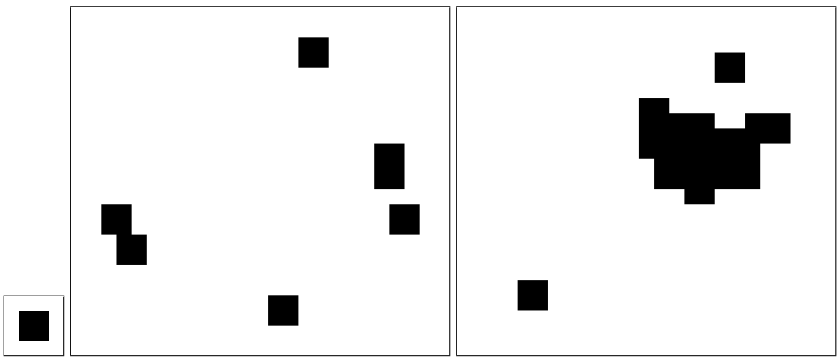
Na obrázku 15 je patrné, že obsahuje velké množství šumu — nežádoucích bodů. Chtěli bychom zachovat kružnice a jim podobné tvary, zatímco osamělé body, linie a chaotické tvary potřebujeme odstranit. Na obrázku 17 je příklad žádoucího tvaru, kmene stromu, a nežádoucího šumu, pravděpodobně větve malého stromu, který nelze s daným rozlišením měřit.

Začneme úvahou, zda by pomohla konvoluce. Jak je popsáno na str. 17, s konvolucí je možné obraz rozmazat, zaostřit, či detekovat v obraze hrany. Rozmazat obraz nechceme, komplikovalo by to následné snahy o změření průměru kmene. Navíc se tak nezbavíme velkých plných tvarů, jako je např. na obrázku 17. Rozmazání je na obrázku 18. Zostřovat obraz není možné, protože na našich datech je obraz po zaostření stále stejný.

Detekování hran má ovšem na první pohled velký potenciál. Nicméně žádoucí tvary v našich datech jsou již pouze hrany, a tak hranový detektor obraz nezmění. U větších souvislých oblastí nežádoucích tvarů po aplikaci hranového detektoru zmizí vnitřní pixely (viz obrázek 18, což na další krok zpracování obrazu – segmentaci – nemá žádný vliv. Hranový detektor má uplatnění především na obrazech v odstínech šedi ne na obrazech binárních.

Kromě všech předchozích důvodů, proč nelze využít konvoluci, je tu i ten, že pracujeme na binárním obraze, přičemž konvoluce by jej dostala do odstínů šedi. Konvoluce je tedy pro tuto aplikaci nevhodná.

Dále zauvažujeme o morfologických operacích (kap. 2.3). Využili bychom zbavování se výběžků, což nás vede k erozi. Nechceme však obrázek zmenšit, proto po erozi aplikujeme dilataci, tudíž jsme vybrali opening. Následujícím krokem je volba strukturního prvku (dále SP). Abychom eliminovali co největší šum, budeme chtít co největší SP.



Obrázek 19: Vlevo strukturní prvek 2×2 pixely, dále žádoucí tvar a nežádoucí tvar z obrázku 17 po aplikaci openingu.

Zároveň nesmíme ztratit kružnice, proto se SP musí „vejít“ do tloušťky linie, která kružnice tvoří. Při pohledu na obrázek 17 vidíme, že spolehlivě celou kružnici ani její části neztratíme při použití SP o velikosti jednoho pixelu. Samotný pixel však nemá žádný vliv na žádný obraz. Lze to přirovnat k násobení libovolného čísla číslem 1.

Na obrázku 19 je příklad použití SP 2×2 pixely. Na obrázku vidíme, že strom je nedetekovatelný, zatímco šum eliminován nebyl.

Konvoluce ani morfologické operace nám tedy nepomohou. Samotná povaha dat to neumožňuje. Pracujeme s malým množstvím pixelů a šum je podobně velký jako kmeny samotné. Budeme tedy muset do dalšího kroku pokračovat s neupravenými daty.

3.3 Segmentace

Následujícím krokem zpracování obrazu je segmentace. Segmentace je rozdělení obrazu na menší obrazy (segmenty), se kterými následně pracujeme zvlášť. Segment obsahuje jeden celý objekt. Příkladem segmentace je obrázek 17, kde vlevo je ukázka povedeného segmentu, zatímco vpravo by bylo žádoucí rozdělit obraz na více částí.

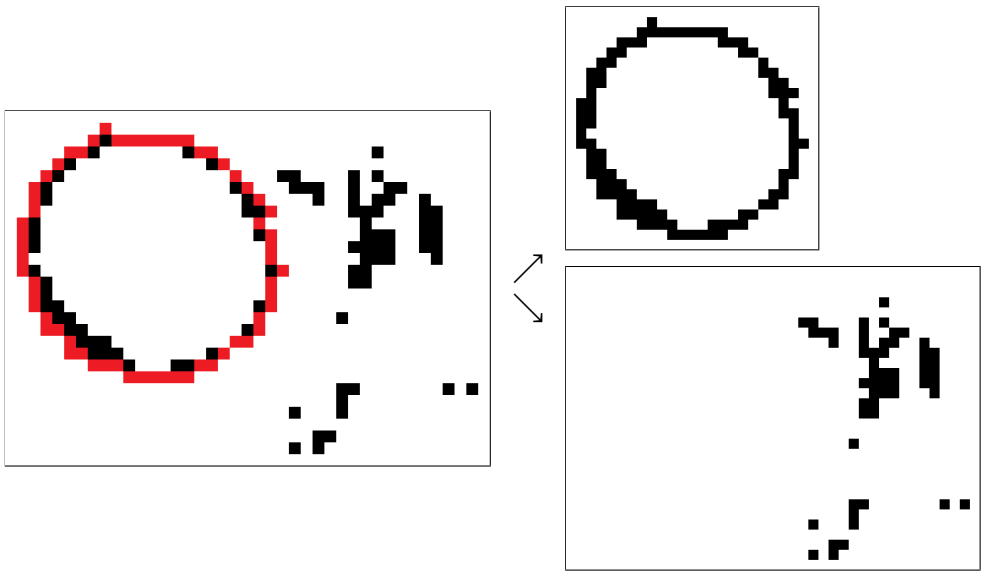
Segmentaci umožňuje detekce obrysu pomocí Freemanova řetězce (kap. 2.4). Aby algoritmus mohl začít, potřebuje počáteční pixel objektu. Proto v našem případě proces segmentace začíná v levém horním rohu. Po řádcích kontrolujeme, zda je pixel prázdný (v barvě pozadí), a když narazíme na pixel nějakého objektu, započneme na něm algoritmus Freemanova řetězce. Jakmile získáme obvod, z obrazu vyjmeme detekované pixely objektu spolu s pixely, které ohraničují, a považujeme jej za nový segment. Příklad je na obrázku 20.

Po vyjmutí segmentu pokračujeme po řádcích prohlížením pixelů, dokud nenarazíme na další objekt. Algoritmus výběru segmentu pokračuje, dokud nejsou všechny pixely zpracovány.

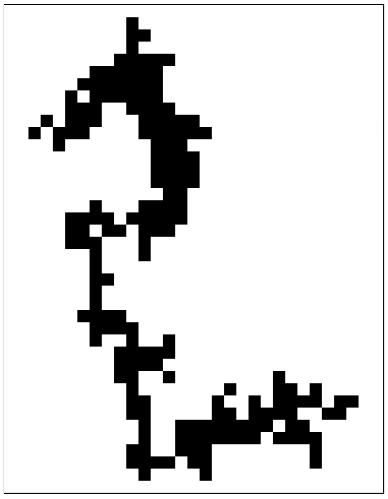
3.4 Měření průměrů Ransacem

Nyní se zaměříme na zpracování jednoho segmentu. Z kapitoly 3.2 víme, že dosud nebylo možné odstranit šum. Pojem *šum* v tomto případě neoznačuje grafické znečištění způsobené skenovacím zařízením, ale objekty, které si nepřejeme měřit, tj. větve, listí, padené stromy a příliš malé stromky.

Nejprve zvolíme práh – minimum pixelů segmentu. Tím odstraníme velkou část šumu. Jak vidíme na obrázcích 20 a 17, kromě zjevného stromu obsahují i šum o 1 až 10 pixelech. Tyto malé segmenty tedy odstraníme prahováním.



Obrázek 20: Červeně ohraničený objekt v obraze → nový segment a obraz bez segmentu.



Obrázek 21: Velká oblast šumu v jednom segmentu.

Zůstanou nám stromy a větší oblasti šumu (např. obrázek 21). Na ty již použijeme algoritmus Ransac. V případě, že objekt nemá přibližný tvar kružnice, bude Ransac neúspěšný, tím odstraníme oblasti šumu, ale i některé stromy. Abychom minimalizovali ztrátu stromů, musíme vhodně zvolit parametry Ransacu.

Aplikace Ransacu

V této části bude popsán algoritmus Ransac v podobě, v jaké je použit v této práci. Od algoritmu popsaného v kapitole 2.5 se liší hlavně tím, že vynecháváme použití metody nejmenších čtverců. Uveďme stručný popis algoritmu:

Model M (n pixelů), množina $P : \mathrm{card}(P) \geq n$.

1. $S_1 \subseteq P$: $\mathrm{card}(S_1) = n$.
2. $S_1 \longrightarrow M_1$.
3. $S_1^* \subseteq P$: $s \in S_1^*$: $\|s - M_1\| \leq \varepsilon$.
4. Jestliže $\mathrm{card}(S_1^*) < t \quad \Rightarrow \quad$ 1. $S_2 \ldots$

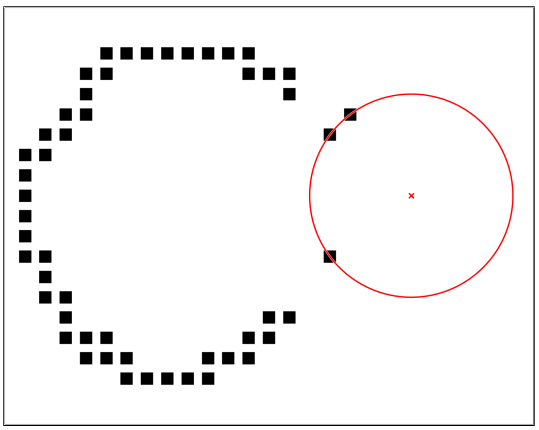
Segmenty budeme prokládat kružnicí, potřebujeme tedy 3 pixely pro stanovení modelu M. Pixely, které patří objektu v daném segmentu, tvoří množinu P. Pokud by nebyla splněna podmínka $\text{card}(P) \geq n$, nebylo by možné určit model.

V kroku 1 se náhodně vyberou 3 pixely z množiny P. V kroku 2 se určí parametry modelu. V kroku 3 se ověří, kolik pixelů je dostatečně blízko modelu. V kroku 4 se testuje podmínka, zda je model dostatečný. Pokud model není dobrý, začíná se opětovně krokem 1.

Každý krok je důkladně popsán na stranách 31 a 32. Nyní se podrobněji podívejme na Ransac implementovaný v kódu.

Kód 1: Metoda Ransac v C++.

```
Circle Segment::Ransac(vector<Pixel> cloud, uint8_t maxIter, double
    thr_distanceInliers, double thr_percentInliers)
{
  vector<unsigned> rng(3);
  Circle adeptCircle, bestCircle;
  vector<Circle> adeptCircles;
  bestCircle.averageDistance = thr_distanceInliers * 10.0;

  for (uint8_t i = 0; i < maxIter; i++){//Hlavni cyklus Ransacu.
    rng = Rng::Get3Rng(cloud.size());
        //3 nahodna cisla v rozsahu daneho objektu.
    adeptCircle = FitCircle(cloud[rng[0]], cloud[rng[1]], cloud[rng
        [2]]);//Vypocet parametru kruznice.

    if (!adeptCircle.success)
      continue;//Kontrola na vytvoreni kruznice.

    Evaluate(adeptCircle, cloud, thr_distanceInliers,
        thr_percentInliers);//Ohodnoceni modelu.

    adeptCircles.push_back(adeptCircle);
        //Prirazeni aktualniho modelu k predchozim.
    if (!adeptCircle.success)
      continue;//Kontrola na dobry model.

    if (adeptCircle.averageDistance < bestCircle.averageDistance)
      bestCircle = adeptCircle;//Vyber dosud nejlepsiho modelu.
  }

  if (bestCircle.success == false){
    bestCircle.inliers = 0;//Pokud nebyl Ransac uspesny,

    for (Circle& element : adeptCircles){
      if (element.inliers > bestCircle.inliers)
        bestCircle = element;//vyhodnoti se nejlepsi neuspesny model.
    }
  }
  return bestCircle;
}
```



Obrázek 22: Chybný model.

V kódu 1 do metody `Ransac` vstupují 4 parametry. Parametr `cloud` je množinou pixelů objektu v segmentu, tedy množinou P. Číslo `maxIter` je číslo I z rovnice (2.5.1). Toto číslo udává počet pokusů, které algoritmus provede. Parametr `thr_distanceInliers`, v symbolickém popisu algoritmu jako ε, je maximální dovolenou vzdáleností pixelu od kružnice, aby byl daný pixel považován za inlier. Poslední parametr `thr_percentInliers`, z popisu algoritmu jako t, je minimální poměr inlierů ku outlierům, aby byla kružnice považována za dobrý model.

Na řádku 8 začíná hlavní cyklus metody, který proběhne právě `maxIter`-krát. Na dalším řádku se volá metoda `Get3Rng`, která vrací 3 náhodná celá čísla z intervalu $\langle 0, \mathrm{card}(P) - 1\rangle$. Tato čísla jsou využita pro výběr tří pixelů, které vstupují do metody `FitCircle`. V této metodě se vypočítají potřebné parametry pro stanovení kružnice – modelu.

Na řádku 13 se kontroluje, zda se podařilo model vytvořit. K neúspěchu by došlo například v případě, když by vybrané pixely ležely na společné přímce. Pokud se model nepodařil, ukončuje se pokus a začíná se další.

V případě úspěchu, metoda `Evaluate` ohodnotí kružnici podle dodaných parametrů `thr_distanceInliers` a `thr_percentInliers`. Ohodnocením se myslí stanovení počtu inlierů a výpočet poměru inlierů ku outlierům, dále pak určení průměrné vzdálenosti inlierů od kružnice a zhodnocení, zda je poměr dostatečný. Následně se kružnice přidá k předchozím modelům pro případ celkového neúspěchu algoritmu.

Na řádku 20 se kontroluje kružnice, jestli splnila podmínku na dostatek inlierů. Pokud ne, pokus končí a začíná další.

Na konci cyklu se porovnává aktuální model s nejlepším předchozím modelem. Kritériem pro porovnání je průměrná vzdálenost inlierů od kružnice. Tímto končí hlavní cyklus metody.

Pokud Ransac nebyl úspěšný a není vybrána nejlepší kružnice, z modelů, které vznikly, se vybere ten nejlepší podle počtu inlierů. V krajním případě, kdy celý objekt leží na jedné přímce a nevznikne žádná kružnice, se z metody vrací kružnice o poloměru 0.

Pokud bychom vynechali kontrolu z řádku 20 s předpokladem, že je zbytečná, mohl by vzniknout falešný nejlepší model, viz obr. 22. Takový model by měl průměrnou vzdálenost inlierů od kružnice 0, i přesto, že nesplňuje požadavek na množství inlierů. Nemohl by tedy vzniknout žádný lepší model. Nutno podotknout, že přesně tento případ nastat nemůže, protože pixely nejsou spojeny a nebyly by v jednom segmentu.

Náhodný výběr

V 1. kroku algoritmu Ransac náhodně vybíráme 3 pixely. Programovací jazyky běžně mají nástroje k pseudo-náhodnému generování čísel. V jazyku C++ je například funkce `rand`. [10] Tato funkce vrací celé nezáporné číslo z předem dané posloupnosti, která je při každém spuštění programu stejná. Tento problém lze vyřešit číslem – tzv. *seed* (z angl. – semínko), který zajistí, že vygenerovaná čísla jsou z jiné posloupnosti. Stejný seed však při každém spuštění dává opět stejné výsledky. Je tedy nutné, aby se pro každé spuštění použil jiný seed. Jednoduše toho lze dosáhnout použitím funkce `time`, která vrátí čas v sekundách, který uplynul od 00:00 hodin 1.ledna 1970. V takovémto případě zbývá ohlídat, aby se seed použil jednou na začátku spuštění programu.

Nežádoucí výsledek by mohl nastat v případě, kdy by se například při každém volání metody Ransac použil nový seed. Jelikož metoda Ransac je volána několikrát v řádech milisekund, všechny posloupnosti generovaných čísel by byly po celou dobu trvání jedné sekundy stejné.

Stanovení modelu

Krok 2 algoritmu Ransac určuje parametry modelu. Naším modelem je kružnice daná rovnicí:

$$(x-a)^2 + (y-b)^2 = r^2, \tag{3.4.1}$$

kde x, y jsou souřadnice pixelu, a, b jsou souřadnice středu a r je poloměr kružnice. Neznámé parametry jsou a, b, r, proto potřebujeme tři rovnice a tři body.

Řešení soustavy lineárních rovnic (SLR) je značně jednodušší, než řešení soustavy nelineárních rovnic. Proto rovnici kružnice (3.4.1) upravíme na nový tvar pomocí substituce $c = a^2 + b^2 - r^2$.

$$x^2 + y^2 - 2ax - 2by + a^2 + b^2 - r^2 = 0 \tag{3.4.2}$$

$$x^2 + y^2 - 2ax - 2by + c = 0 \tag{3.4.3}$$

$$2xa + 2yb - c = x^2 + y^2 \tag{3.4.4}$$

Rovnice (3.4.4) je pro neznámé a, b, r lineární, lze tedy řešit SLR s dosazenými body $[x_1, y_1]$, $[x_2, y_2]$ a $[x_3, y_3]$.

$$\begin{pmatrix} 2x_1 & 2y_1 & -1 \\ 2x_2 & 2y_2 & -1 \\ 2x_3 & 2y_3 & -1 \end{pmatrix} \begin{pmatrix} a \\ b \\ c \end{pmatrix} = \begin{pmatrix} x_1^2 + y_1^2 \\ x_2^2 + y_2^2 \\ x_3^2 + y_3^2 \end{pmatrix} \tag{3.4.5}$$

Vzdálenost

V kroku 3 algoritmu Ransac se počítá, kolik pixelů je v dostatečné blízkosti modelu. Parametr označený jako ε je nutné vhodně zvolit v závislosti na povaze sledovaných dat. Příliš velká hodnota zvyšuje šanci na úspěšně provedený Ransac pro šum. Příliš malá hodnota zamezí úspěšnému Ransacu na nepravidelných stromech. V této práci je $\varepsilon = 1$, to znamená šířka jednoho pixelu na obě strany od modelu. Pro výpočet vzdálenosti d bodu od kružnice použijeme rovnici:

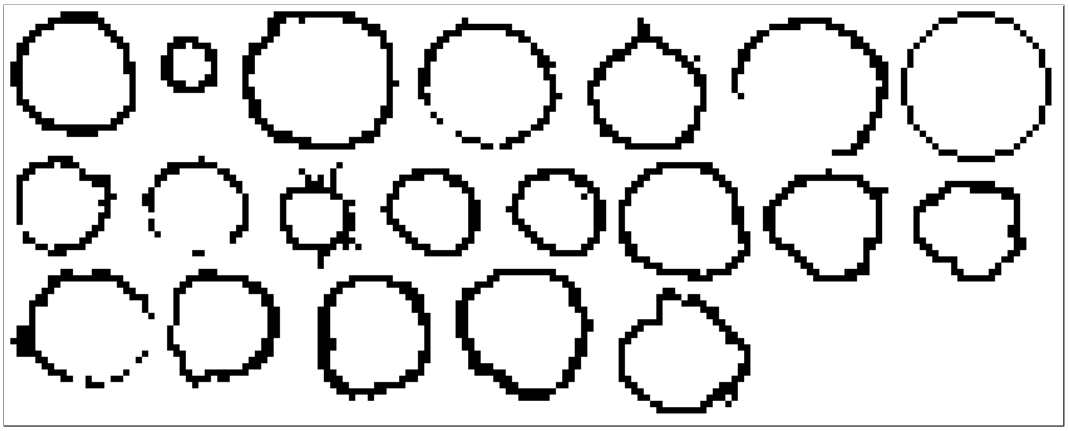
$$d = \left| r - \|\overrightarrow{(B,S)}\| \right|, \tag{3.4.6}$$

kde r je poloměr kružnice, B měřený bod a $S = [a, b]$ střed naší kružnice. Dále $\|\overrightarrow{(B,S)}\|$ je velikost vektoru.

Pokud pro bod B platí $d \leq \varepsilon$, pak bod označíme za inlier, v opačném případě outlier.

Dobrý model

Krok 4 algoritmu Ransac odlišuje špatný model od dobrého podle poměru inlierů ku outlierům. Parametr t, stejně jako ε v předchozí části, musíme vhodně zvolit podle dat, která zpracováváme. Jestliže předpokládáme velké množství výběžků z kmenů stromů, bude t menší, pokud hledáme stromy téměř dokonalé, nastavíme t vysoké. V této práci je $t = 0.7$, tj. požadujeme alespoň 70 % pixelů celého segmentu, aby byly inliers. Potom model označíme za dobrý.



Obrázek 23: Příklady stromů.

Obrázek 24: Příklady oblastí šumu.

4 Testování

V této kapitole se zaměříme na hledání vhodného nastavení parametru t – minimální poměr inlierů ku outlierům – a parametru ε – maximální vzdálenost pixelu od modelu – tak, aby se maximalizoval počet nalezených stromů a minimalizoval počet chybně nalezených oblastí šumu.

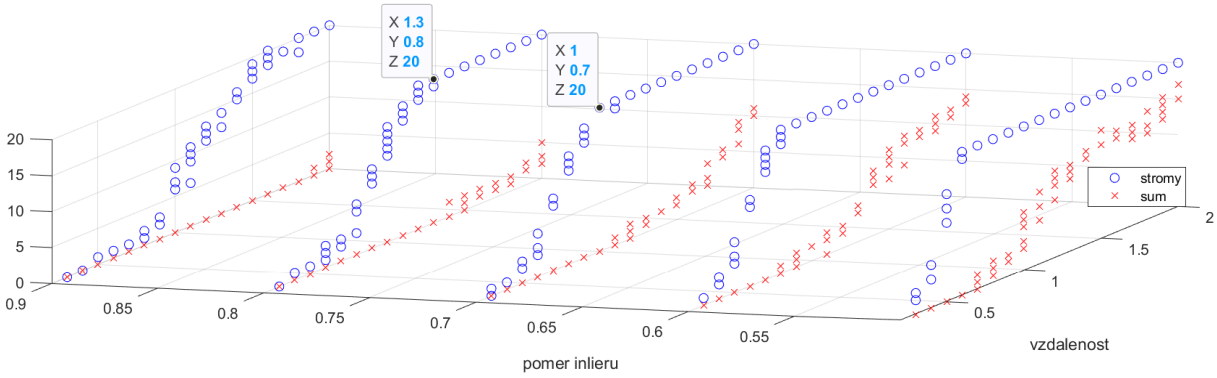
Dále algoritmus otestujeme na 70 různě velkých výřezech ze 7 hektarů Žofínského pralesa.

Nakonec zhodnotíme dosažené výsledky.

4.1 Nastavení parametrů

Pro kvalitu výsledků z Ransacu je klíčové správné nastavení parametrů t a ε. Z řezů byly vybrány příklady stromů (obr. 23) a příklady oblastí šumu (obr. 24). Oba obrázky byly zpracovány Ransacem při různém nastavení parametrů. Poměr inlierů: $t \in \{0,5;\ 0,6 \ldots\ 0,9\}$. Vzdálenost pixelu od kružnice: $\varepsilon \in \{0,3;\ 0,4 \ldots\ 2,0\}$.

Pro každé nastavení proběhl algoritmus 5 krát. Výsledek je na obrázku 25, kde modře jsou počty detekovaných stromů z obrázku 23 a červeně počty detekovaných oblastí šumu z obrázku 24. Dobré nastavení parametrů je takové, které má maximum detekovaných stromů a žádný detekovaný šum. Na obrázku jsou označeny dva body při parametrech $[1;\ 0,7]$ a $[1,3;\ 0,8]$. Tato nastavení dosáhla dobrých výsledků. S parametry $[1,3;\ 0,8]$ byl v jednom případě vynechán jeden strom, proto je nastavení $[1;\ 0,7]$ nejlepší (obr. 26).



Obrázek 25: Počty detekovaných stromů pro daná nastavení parametrů. Modře stromy, červené šum.

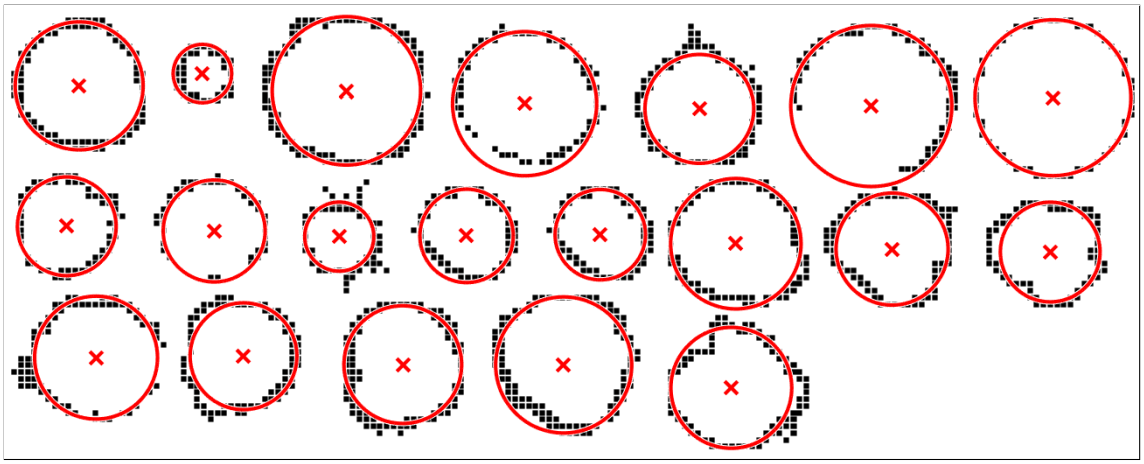
4.2 Data

Tato kapitola navazuje na kapitolu 3.1.

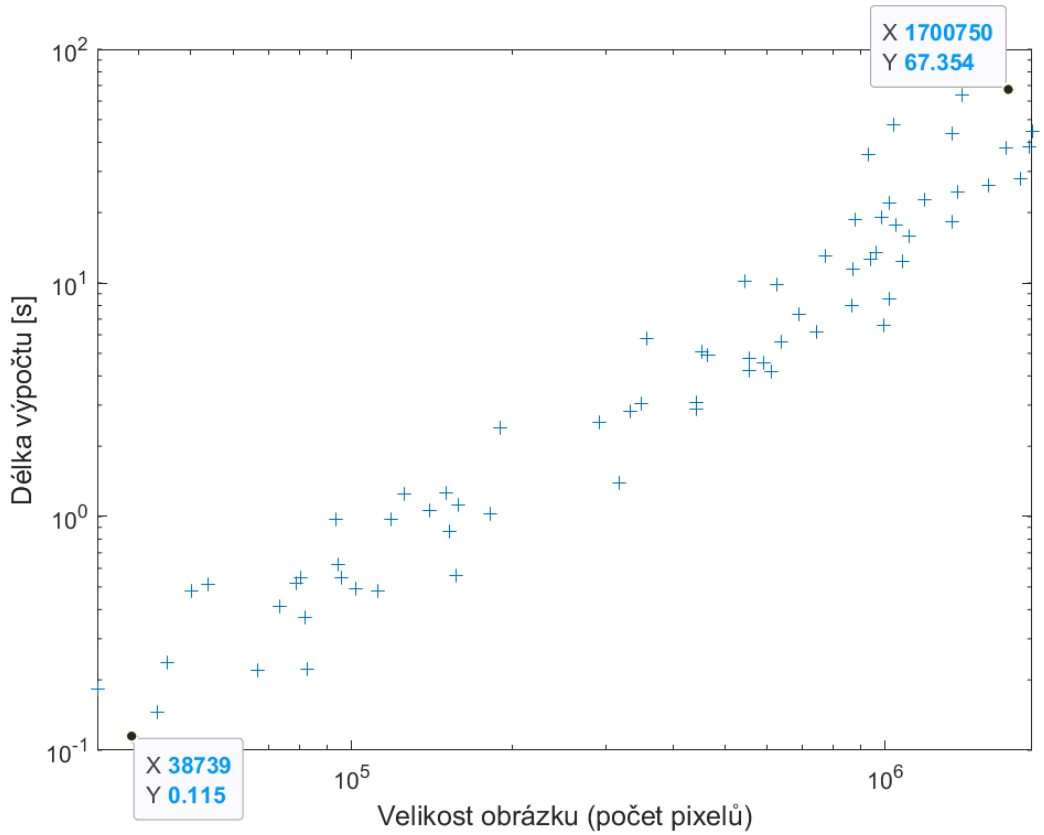
Na ploše 1 hektaru lze očekávat průměrně 3000 stromů od DBH[5] 1 cm [11]. V řezech, které máme k dispozici, 1 pixel reprezentuje 5×5 cm. Proto není možné detekovat menší stromy než přibližně 40 cm (2. strom z obr. 26 má kružnici o průměru 46 cm). Navržený algoritmus je vhodný pro detekci vzrostlých stromů.

Na druhou stranu pro jemnější rozlišení potřebujeme point-cloud s vyšší hustotou bodů, protože pro segmentaci vyžadujeme souvislé obrysy kmenů. Jemný rastr by na řídkém skenu měl za následek velké množství osamocených pixelů, mezery v obrysech kmenů.

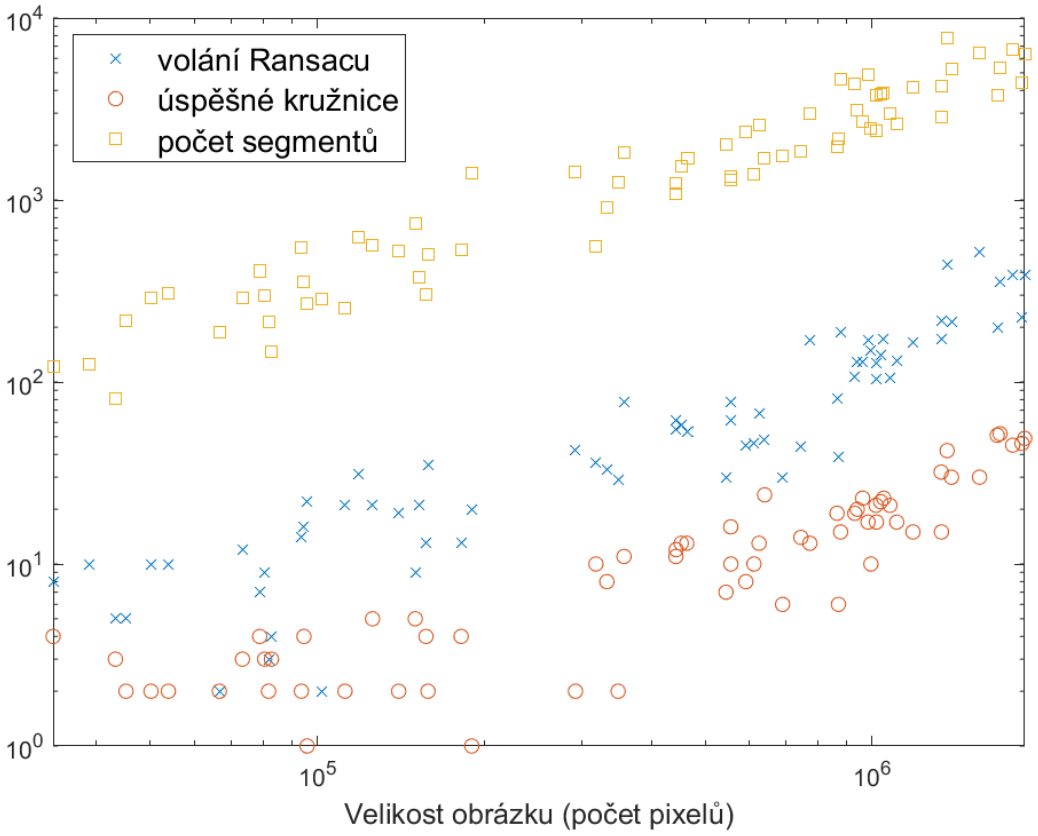
[5]DBH – Diameter at Breast Height – průměr v prsní výšce, tj. 130 cm od země.



Obrázek 26: Úspěšně detekované všechny stromy z obrázku 23. Pro $\varepsilon = 1$ a $t = 0,7$.



Obrázek 27: Graf závislosti délky výpočtu na velikosti (počtu pixelů) vstupního obrázku.



Obrázek 28: Graf závislosti počtu volání algoritmu Ransac, počtu úspěšně spočítaného modelu kružnice a počtu segmentů na velikosti (počtu pixelů) vstupního obrázku.

4.3 Zpracování dat

Algoritmus Ransac s parametry $\varepsilon = 1$ a $t = 0,7$, byl použit na 70 různě velkých výřezech ze 7 hektarů jádrové oblasti Žofínského pralesa. Vybrané řezy jsou v příloze, str. 40.

Pro každý řez byl měřen čas, po který byl zpracováván. Grafická závislost délky výpočtu na velikosti řezu je na obrázku 27. V grafu jsou zvýrazněny dva body – nejrychlejší a nejdelší zpracovávání. Řez s počtem pixelů 1 700 750 odpovídá ploše 42,5 aru a byl zpracován za 67 sekund. Provedeme-li aproximaci, program zpracuje půl hektaru za minutu. Celkový čas zpracování všech 70 výřezů je 13 minut a 3,9 sekundy.

Kromě času byl zaznamenáván počet volání metody Ransac, počet úspěšně vypočítaných kružnic a počet segmentů v obrázku – vše v závislosti na velikosti řezu (obrázek 28). Z grafu vidíme, že při našem rozlišení je počet volání Ransacu řádově menší než počet segmentů.

Jak bylo zmíněno v kapitole o měření průměrů 3.4, aby byla volána metoda Ransac, segment musí mít dostatečný počet pixelů. Tím jsou odstraněny nejmenší oblasti šumu. Z toho vyplývá, že zpracovávaná data obsahují velice velké množství malého šumu.

Počet úspěšně vypočítaných kružnic je zřejmě menší než počet volání metody Ransac. Nicméně tento parametr plně nevystihuje počet úspěšně detekovaných stromů. Kružnice může být napočítána na oblast šumu, proto byly vybrané menší výřezy podrobeny manuální kontrole (viz tab. 1). V příloze jsou tyto obrázky pod čísly 30 – 40. Červené jsou kružnice, které vypočítal Ransac. V modrých obdélnících jsou zvýrazněny nedostatky algoritmu, jako jsou nedetekované stromy a chybně detekované šumy.

č. obr.	č. řezu	$\oplus$	$f\oplus$	$f\ominus$	$\ominus$
30	22	2	0	0	187
31	23	5	0	2	557
32	24	4	2	0	118
33	26	3	1	2	287
34	33	1	0	0	270
35	34	3	0	0	78
36	36	5	0	1	737
37	37	2	0	0	548
38	40	2	0	0	212
39	42	4	0	0	407
40	44	2	0	0	287

Tabulka 1: Počty úspěšně detekovaných stromů $\oplus$, chybně detekovaných oblastí šumu $f\oplus$, chybně nedetekované stromy $f\ominus$ a úspěšně nedetekovaných oblastí šumu $\ominus$ pro vybrané řezy.

4.4 Zhodnocení

Algoritmus úspěšně odfiltruje velké množství šumu všech velikostí. Úspěšně detekuje izolované pravidelné vrostlé stromy.

Na obrázku 29 jsou stromy z vybraných řezů, které Ransac nedetekoval. Na nich si ukážeme problematické charakteristiky segmentů z pohledu algoritmu.

První segment zleva je příliš nepravidelný. Aby na něm byl Ransac úspěšný, snížili bychom požadavek na poměr inlierů.

Druhý segment je spojen ze dvou kmenů. Algoritmus počítá pouze jednu kružnici na každý segment a rozdělit segment na dva algoritmus nezvládne. Detekce jednoho z nich lze opět zajistit snížením požadavku na poměr inlierů, nicméně druhý kmen detekován nebude.

Třetí segment je napojený na šum. Pokud je šumu dostatečně málo, algoritmus je úspěšný, avšak větší oblasti šumu zapříčiní neúspěch. Řešením by opět bylo snížení požadavku na poměr inlierů.

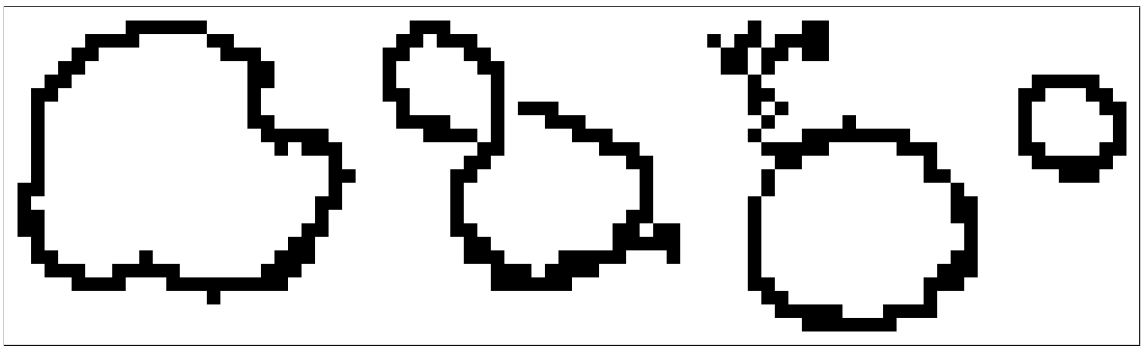
Poslední segment se skládá z 29 pixelů. Algoritmus má nastaven práh pro rozlišení malých segmentů od dostatečně velkých na 30. Jasným řešením by bylo snížení prahu.

Snížení požadavku na poměr inlierů nebo snížení prahu by bylo řešením pro tyto problematické stromy, nicméně zvolnění jakékoliv podmínky způsobí větší počet chybných detekcí oblastí šumu.

Narážíme tedy na limit samotného algoritmu. Ransac je vhodný pro zpracování dat s šumem, nicméně stále musí převažovat množství správných bodů. Tuto převahu ztrácíme přechodem z point-cloudu na obraz, kdy několik blízkých bodů na kmeni je redukováno na jeden pixel, přičemž stačí jeden bod šumu, aby byl též reprezentován jedním pixelem.

Dále v tomto algoritmu Ransac plní dvě funkce. Kromě počítání kružnic, je využíván i pro filtraci dat. Podmínka na dostatečný poměr inlierů by měla odlišit dobře vypočítaný model od špatného, zde však dále rozlišuje oblast šumu od stromu. Snaha nastavit parametr tak, aby plnil obě funkce, způsobí snížení kvality výsledku.

Řešením tedy je aplikovat Ransac přímo na point-cloudy ne obrazy. Dále je vhodné Ransac používat na počítání modelu ne filtraci dat.



Obrázek 29: Problematické stromy z vybraných řezů. Zleva: příliš nepravidelný strom, dva stromy spojeny do jednoho segmentu, strom napojený na šum, příliš malý strom.

5 Závěr

Cílem této práce bylo navrhnout algoritmus, který v řezech skenů lesa nalezne stromy, určí jejich pozici a průměr.

Za tímto účelem byl vystavěn aparát analýzy obrazu. Ukázalo se, že konvoluce ani morfologické operace na řezy nelze použít.

Výsledný algoritmus je převážně složen z algoritmu Freemanova řetězce (segmentace) a algoritmu Ransac (měření průměrů).

Testování na datech z Žofínského pralesa, která poskytl Ing. Martin Krůček, Ph.D., ukázalo, že navržený algoritmus je vhodný pro detekci vzrostlých stromů. Menší stromy jsou kvůli rozlišení řezu nedetekovatelné.

Z hlediska chyb – vynechání stromu nebo neodstranění šumu – je algoritmus poměrně spolehlivý. Výrazné zvýšení spolehlivosti není možné, neboť Ransac plní dvě funkce. Kromě počítání kružnic je využíván i pro filtraci dat. Snaha nastavit parametry Ransacu tak, aby plnil obě funkce, způsobí snížení kvality výsledku.

Práce bude dále pokračovat použitím algoritmu Ransac přímo na point-cloudu, aby bylo možné měřit i menší stromy. Data by měla být filtrována jiným algoritmem, aby parametry Ransacu byly nastaveny za jediným účelem.

Spustitelná aplikace je přílohou této práce a komentář k jejímu ovládání je v přiloženém *readme* souboru.

Reference

[1] KVĚTNÝ, Michal. *Metody mapování textur na mračna bodů*. Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2022, 89 s. Vedoucí práce Mgr. Jana Procházková, Ph.D.

[2] GONZALEZ, Rafael C. a Richard E. WOODS. *Digital image processing*. 3rd ed. Upper Saddle River: Pearson, c2008. ISBN 978-0-13-168728-8.

[3] FREEMAN, Herbert. On the Encoding of Arbitrary Geometric Configurations. *IEEE Transactions on Electronic Computers* [online]. 1961, EC-10(2), 260-268 [cit. 2022-11-06]. ISSN 0367-7508. Dostupné z: doi:10.1109/TEC.1961.5219197

[4] LUENGO, Cris. How to obtain the chain code. *Cris' Image Analysis Blog: theory, methods, algorithms, applications* [online]. 27 September 2010 [cit. 2023-05-01]. Dostupné z: https://www.crisluengo.net/archives/324/

[5] LUENGO, Cris. Measuring boundary length. *Cris' Image Analysis Blog: theory, methods, algorithms, applications* [online]. 14 September 2010 [cit. 2023-05-01]. Dostupné z: https://www.crisluengo.net/archives/310/

[6] FISCHLER, Martin A. a Robert C. BOLLES. Random sample consensus. *Communications of the ACM* [online]. 1981, 24(6), 381-395 [cit. 2022-11-06]. ISSN 0001-0782. Dostupné z: doi:10.1145/358669.358692

[7] KRŮČEK, Martin, Kamil KRÁL, KC CUSHMAN, Azim MISSAROV a James R. KELLNER. Supervised Segmentation of Ultra-High-Density Drone Lidar for Large-Area Mapping of Individual Trees. *Remote Sensing* [online]. 2020, 12(19), 16 [cit. 2023-02-21]. ISSN 2072-4292. Dostupné z: doi:10.3390/rs12193260

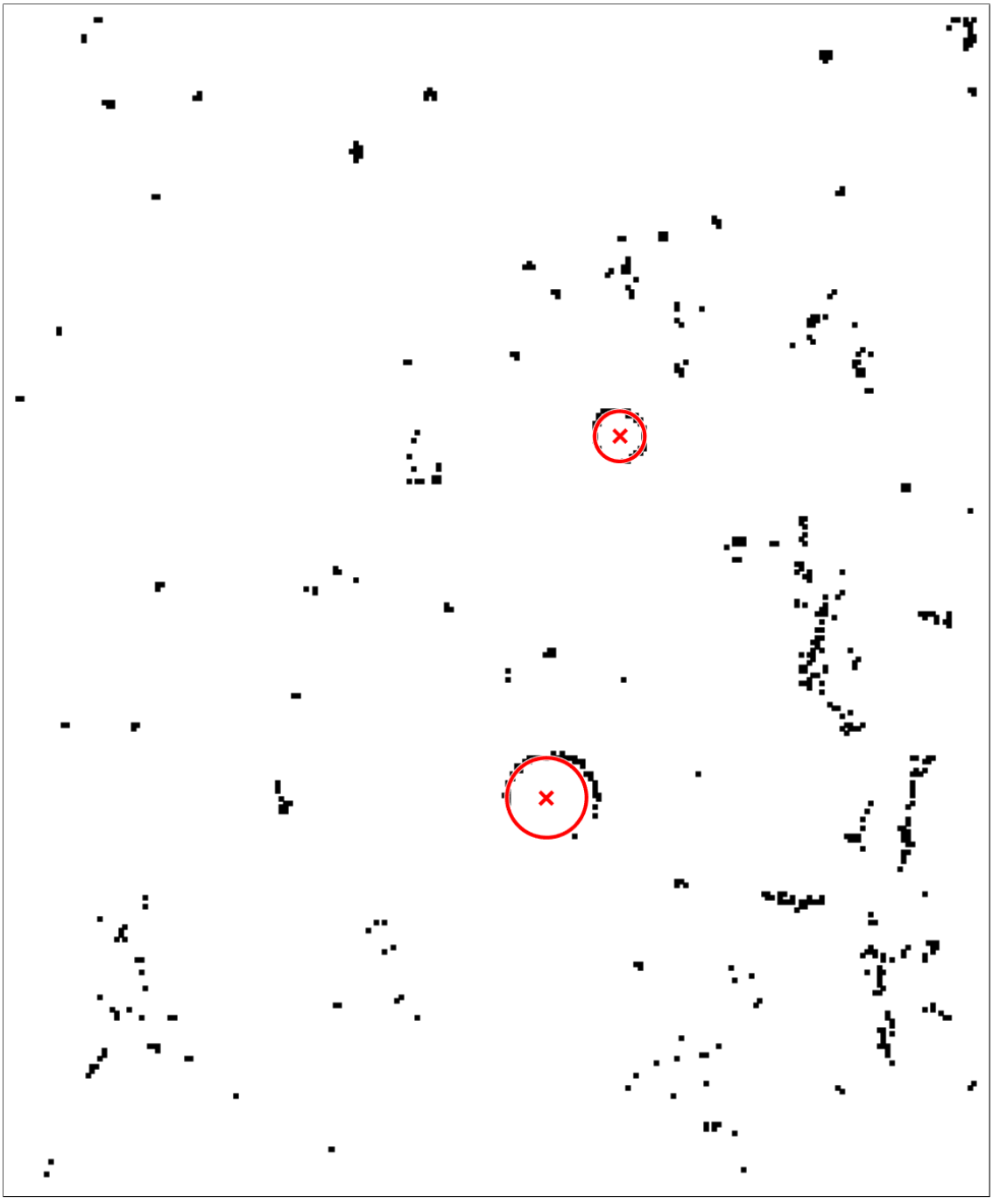
[8] KRŮČEK, Martin, Jan TROCHTA, Miloš CIBULKA a Kamil KRÁL. Beyond the cones: How crown shape plasticity alters aboveground competition for space and light—Evidence from terrestrial laser scanning. *Agricultural and Forest Meteorology* [online]. 2019, 264, 188-199 [cit. 2023-02-21]. ISSN 01681923. Dostupné z: doi:10.1016/j.agrformet.2018.09.016

[9] TROCHTA, Jan, Martin KRŮČEK, Tomáš VRŠKA, Kamil KRÁL a Jian YANG. 3D Forest: An application for descriptions of three-dimensional forest structures using terrestrial LiDAR. *PLOS ONE* [online]. 2017, 12(5) [cit. 2023-02-21]. ISSN 1932-6203. Dostupné z: doi:10.1371/journal.pone.0176871

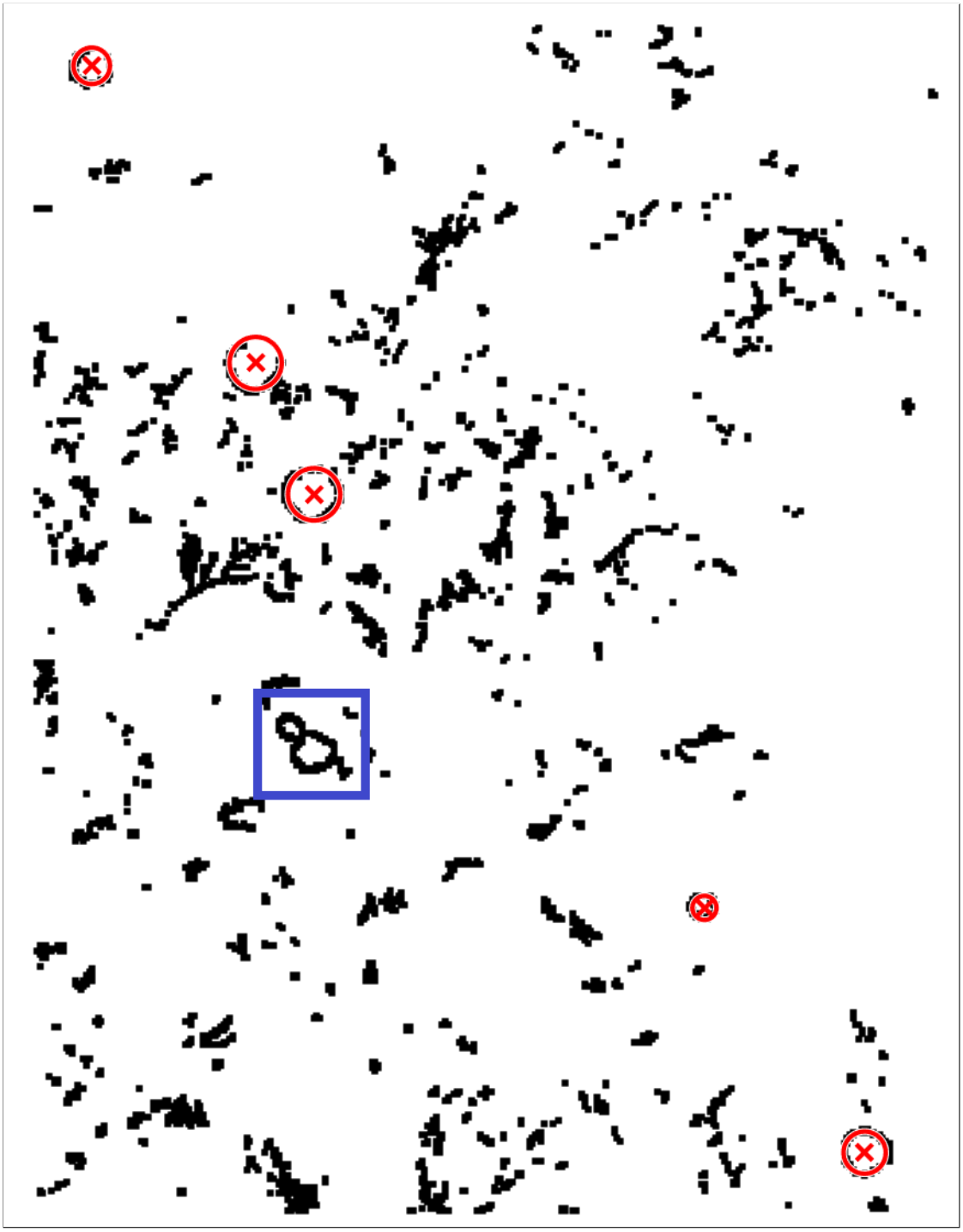
[10] Rand. *Cplusplus.com* [online]. 2000 [cit. 2023-03-15]. Dostupné z: https://cplusplus.com/reference/cstdlib/rand/

[11] Žofín ForestGEO. *Naturalforests.cz* [online]. [cit. 2023-04-26]. Dostupné z: https://naturalforests.cz/zofin-forest-geo

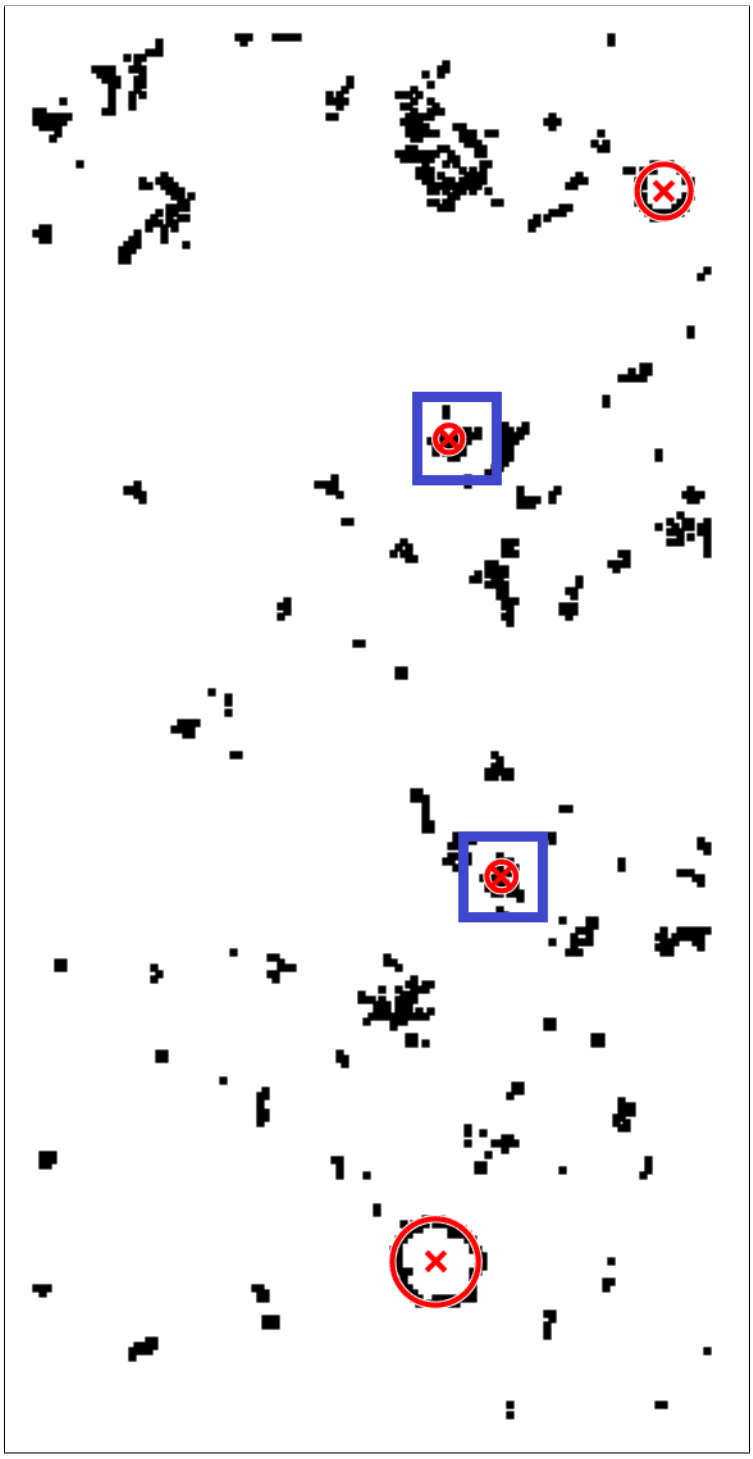
Příloha



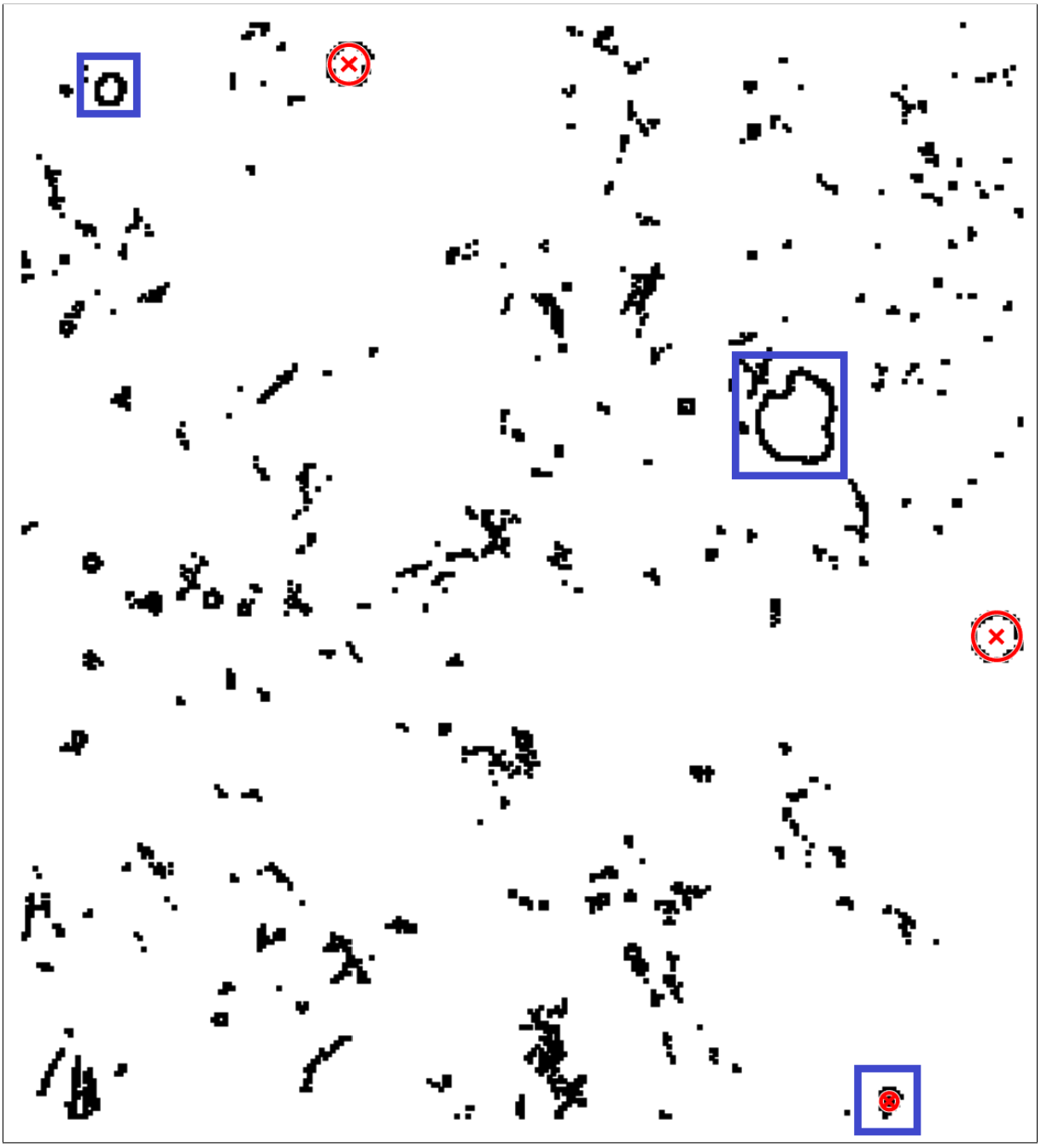
Obrázek 30: Zpracovaný řez č. 22.



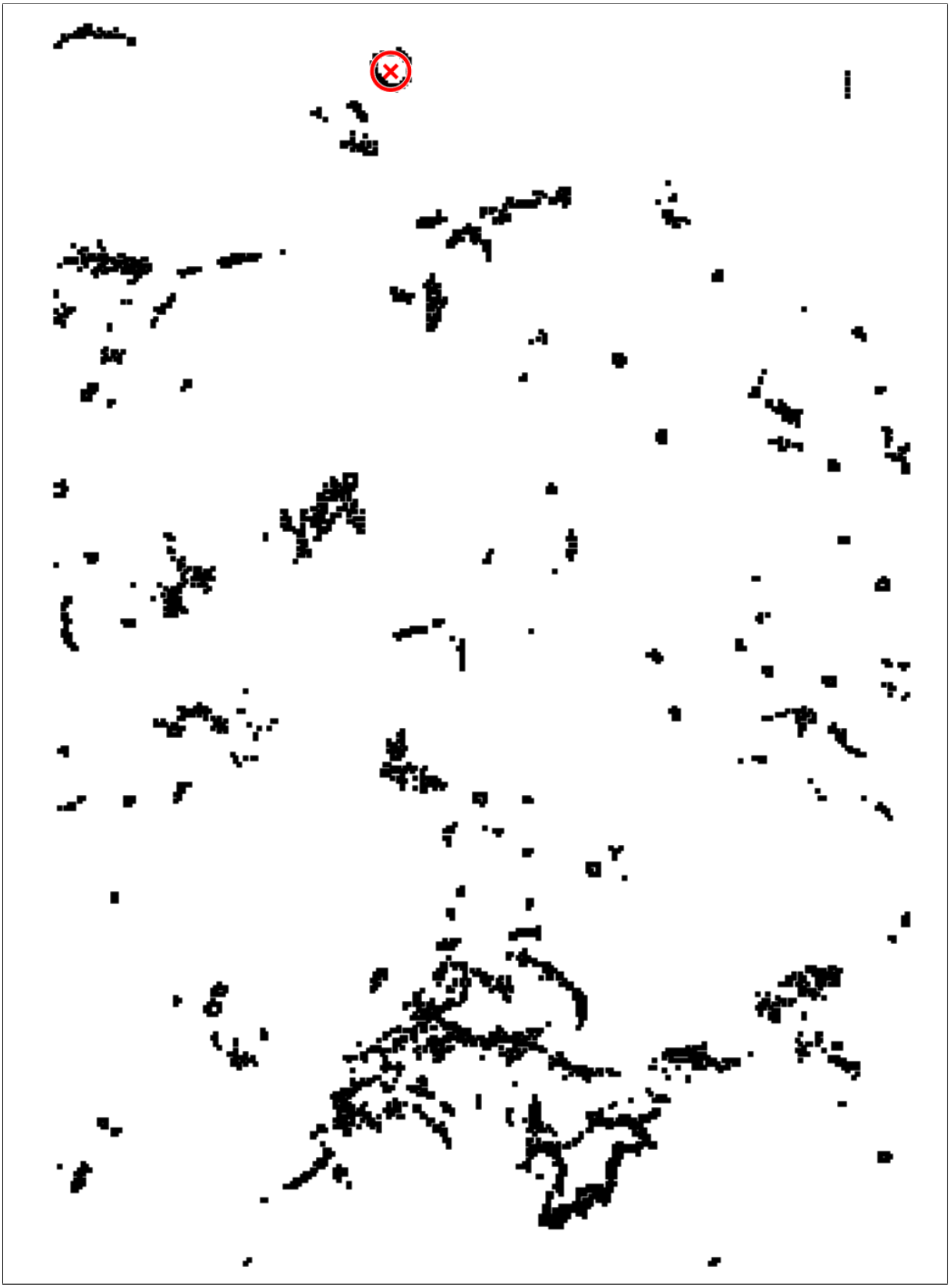
Obrázek 31: Zpracovaný řez č. 23.



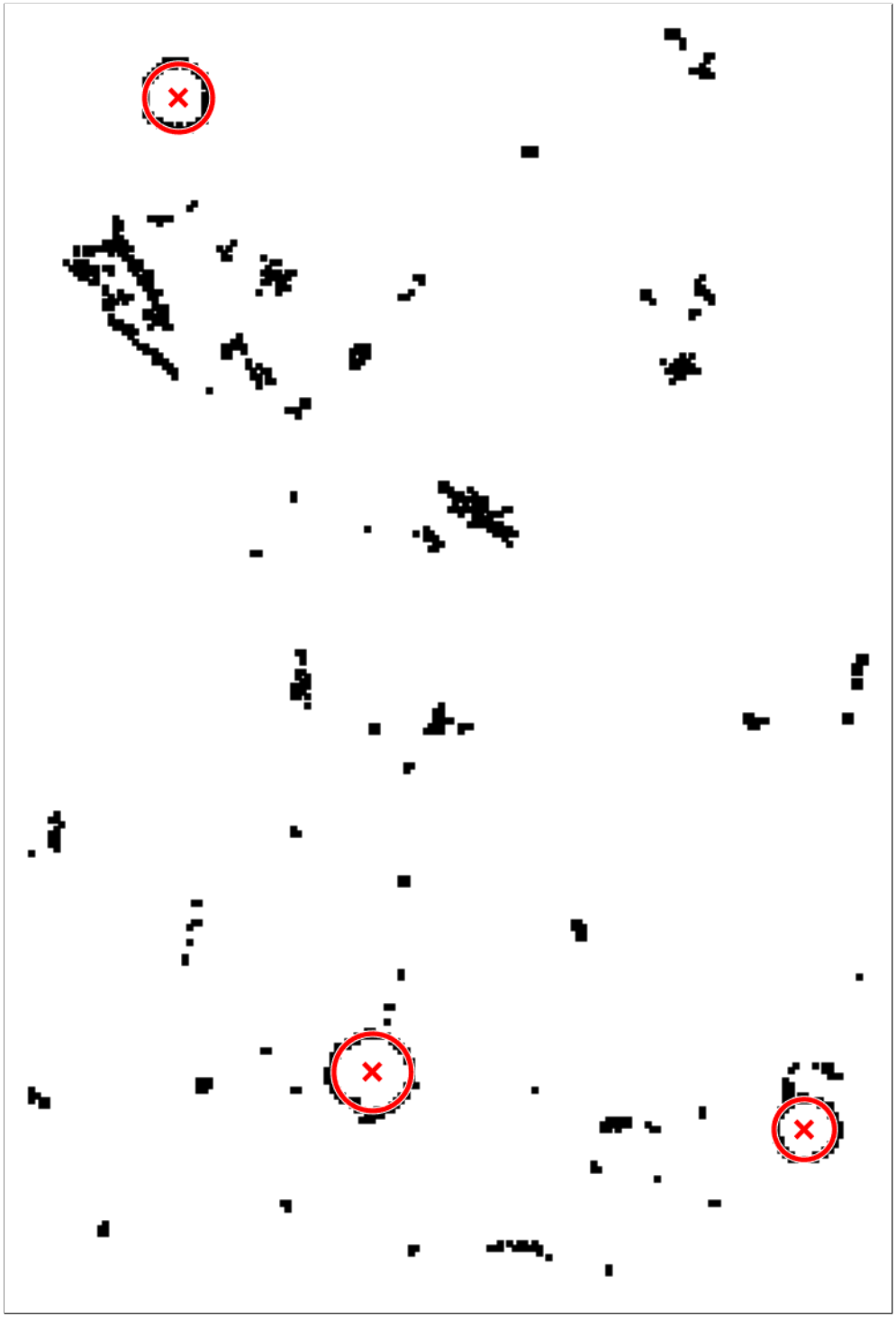
Obrázek 32: Zpracovaný řez č. 24.



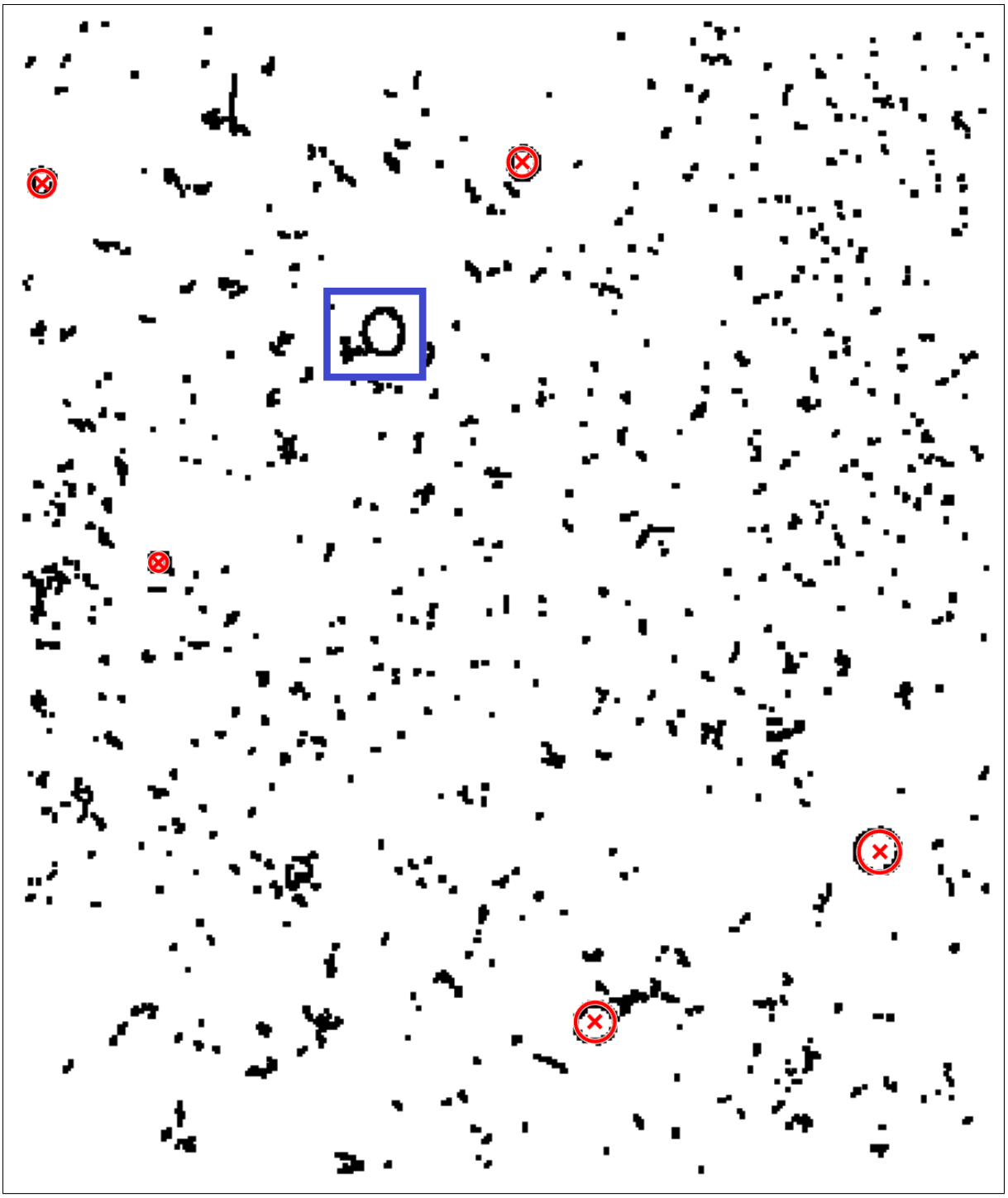
Obrázek 33: Zpracovaný řez č. 26.



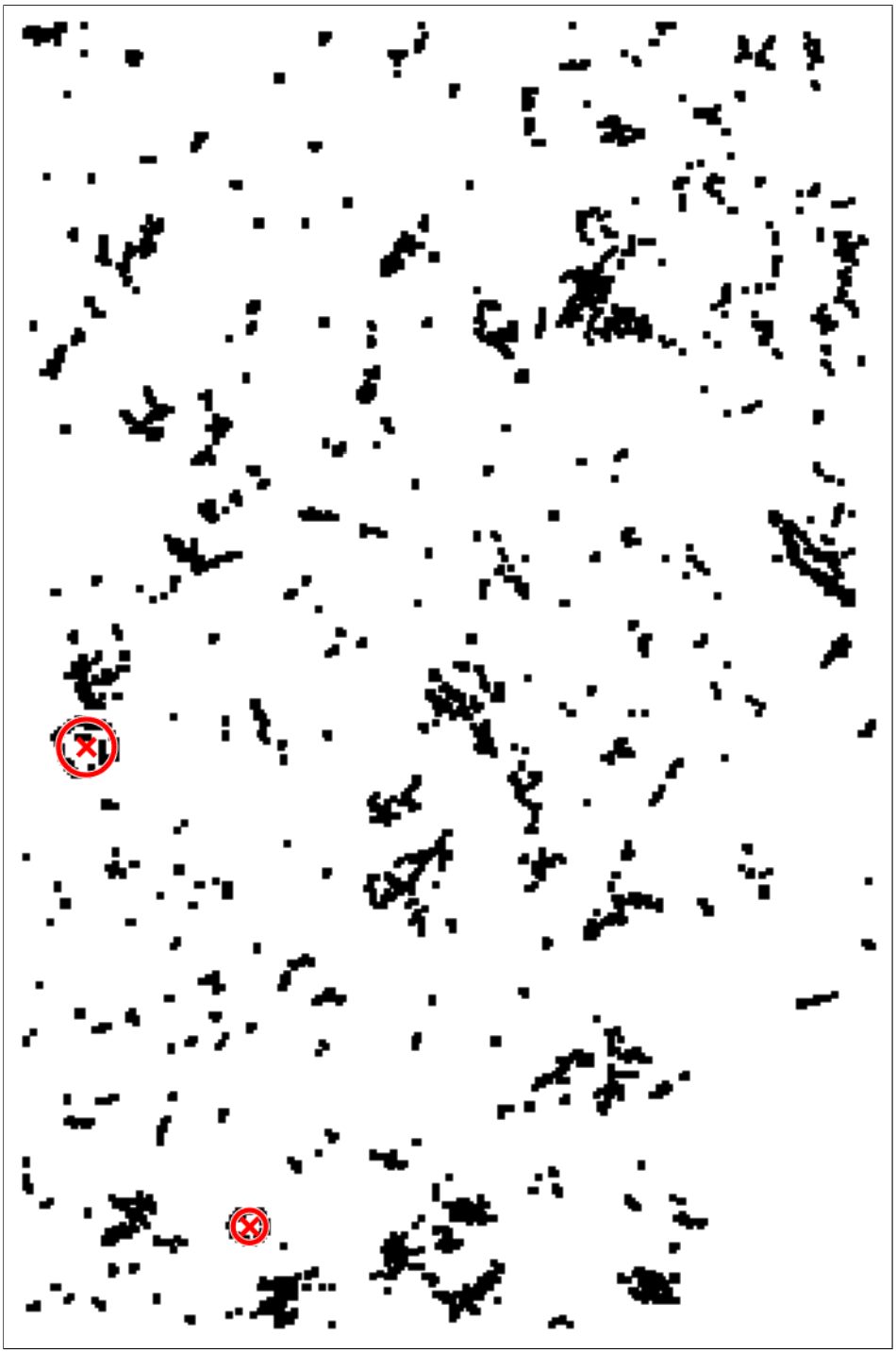
Obrázek 34: Zpracovaný řez č. 33.



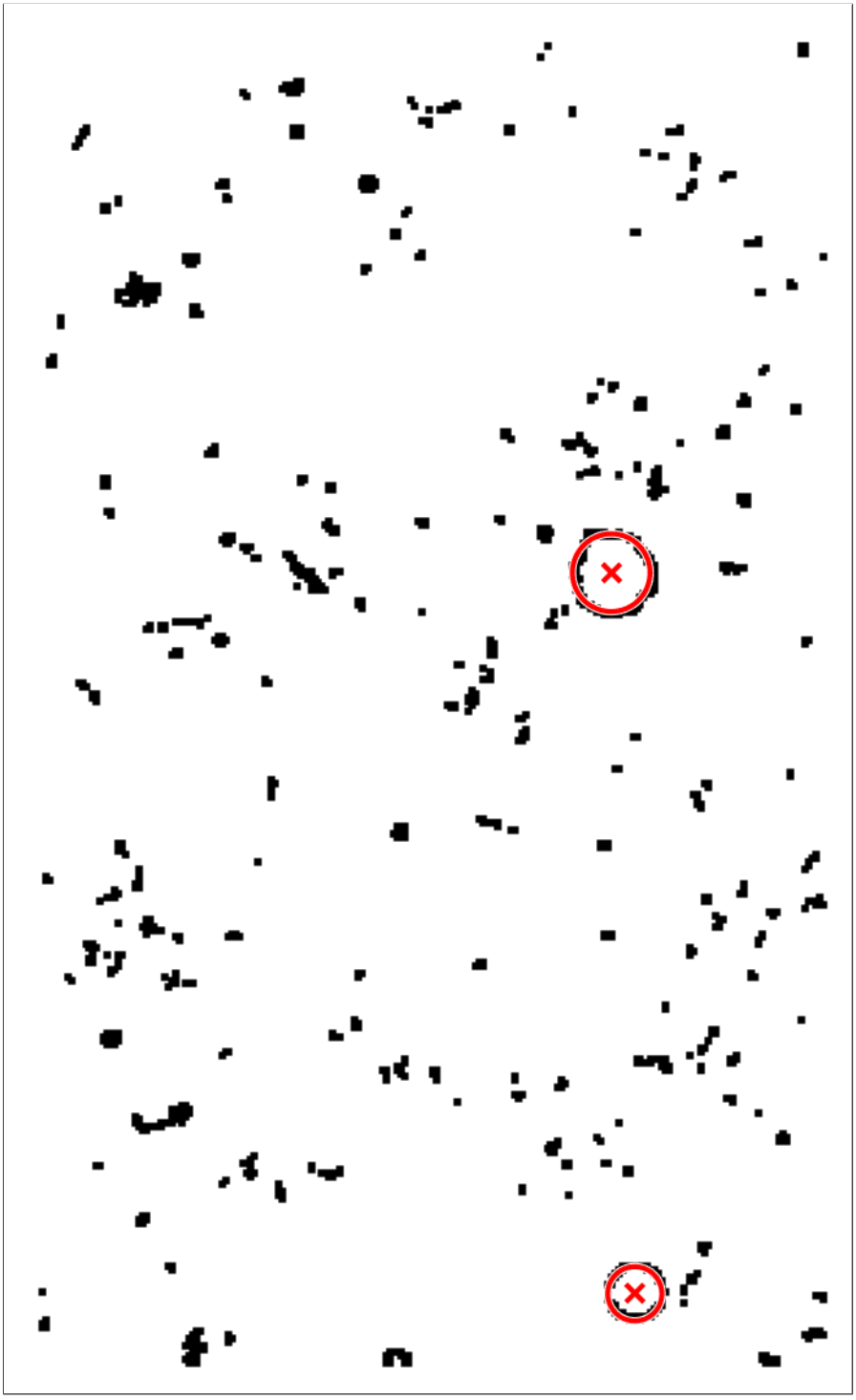
Obrázek 35: Zpracovaný řez č. 34.



Obrázek 36: Zpracovaný řez č. 36.

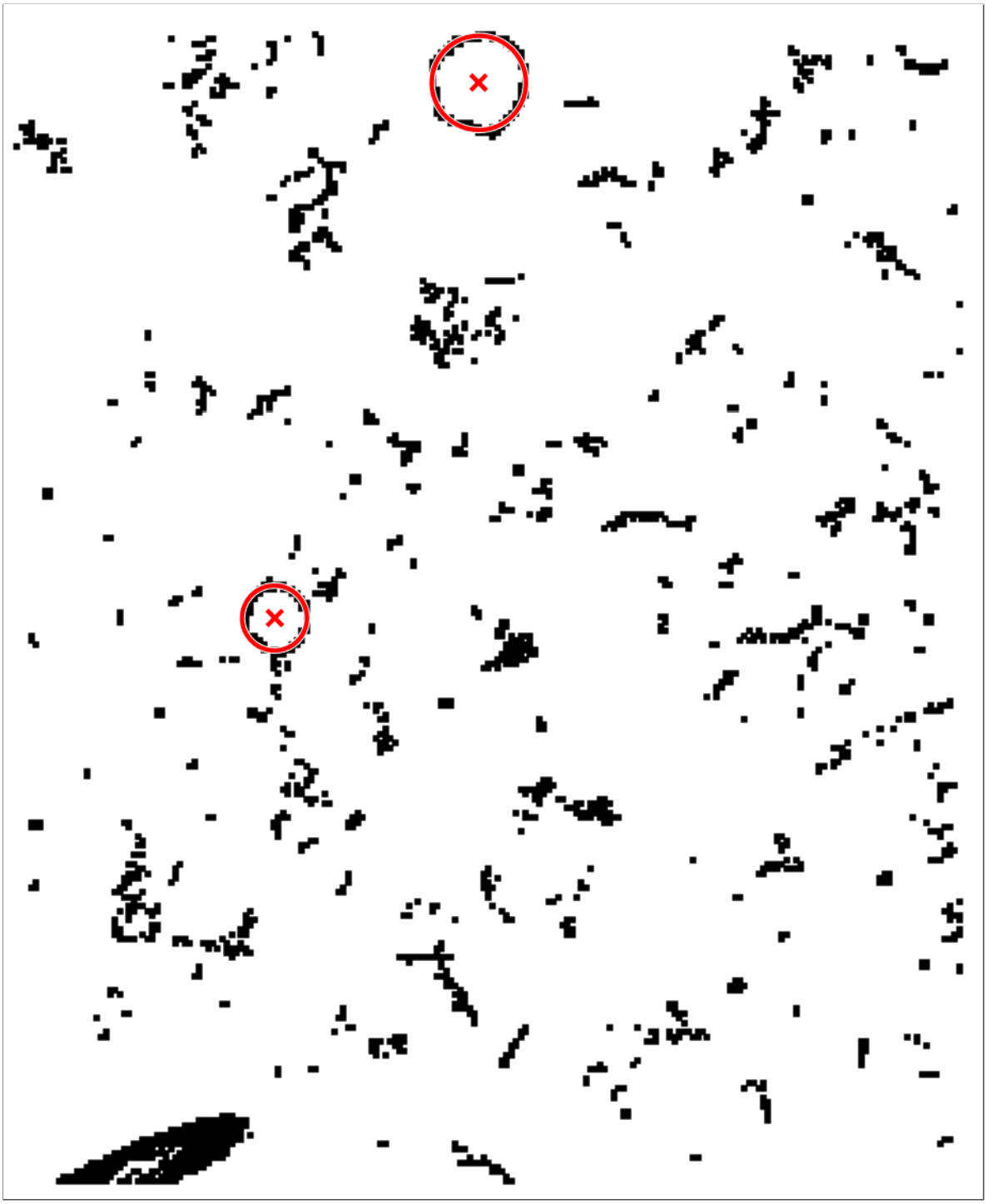


Obrázek 37: Zpracovaný řez č. 37.



Obrázek 38: Zpracovaný řez č. 40.

Obrázek 39: Zpracovaný řez č. 42.



Obrázek 40: Zpracovaný řez č. 44.