FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Zadání bakalářské práce

Název:	Optimalizace využití tradičních segmentačních algoritmů pro úlohy detekce defektů v průmyslu
Student:	Jiří Szkandera
Vedoucí:	Ing. Jakub Novák
Studijní program:	Informatika
Obor / specializace:	Znalostní inženýrství
Katedra:	Katedra aplikované matematiky
Platnost zadání:	do konce letního semestru 2022/2023

Pokyny pro vypracování

Cílem práce je zmapovat segmentační algoritmy zpracování obrazu na bázi kontur či spojených komponent a nalézt postup a využití takových algoritmů pro úlohy detekce objektů (nejčastěji defektů) z průmyslu.

Průmyslové úlohy jsou specifické velikostí obrazových dat, šumem a požadavkem na rychlost. Aktuálně používané implementace (např. v OpenCV) pro detekce kontur či spojených komponent jsou postavené na starších metodách a nezahrnují modernější algoritmy.

Úkoly:
1) Proveďte rešerši v oblasti tradičních segmentačních algoritmů a jejich využití.
2) Seznamte se s implementacemi algoritmů v používaných knihovnách pro strojové vidění.
3) Seznamte se s workflow zpracování obrazových dat v průmyslových úlohách.
4) Navrhněte postupy využití vhodných algoritmů a implementujte je.
5) Otestujte výsledky algoritmů a zhodnoťte úspěšnost.
6) Vizualizujte výsledky algoritmů na obrazových datech.

Elektronicky schválil/a Ing. Karel Klouda, Ph.D. dne 2. února 2022 v Praze.

Bakalářská práce

OPTIMALIZACE VYUŽITÍ TRADIČNÍCH SEGMENTAČNÍCH ALGORITMŮ PRO ÚLOHY DETEKCE DEFEKTŮ V PRŮMYSLU

Jiří Szkandera

Fakulta informačních technologií
Katedra aplikované matematiky
Vedoucí: Ing. Jakub Novák
12. května 2022

Obsah

Seznam obrázků

Seznam tabulek

Seznam algoritmů

Rád bych poděkoval svému vedoucímu Ing. Jakubu Novákovi za nesmírnou vstřícnost, ochotu a jasnou vizi při tvorbě této práce.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů, zejména skutečnost, že České vysoké učení technické v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 citovaného zákona.

V Praze dne 12. května 2022 .....

Abstrakt

Práce porovnává algoritmy určené k segmentaci objektu v obraze. Porovnány jsou tři algoritmy řadící se do kategorie superpixels (felzenszwalb, SLIC a quickshift), dva zástupci active contour models (snakes a level sets), random walker, region adjacency graphs a Otsu prahování. K tomuto účelu jsou zmapovány defekty objevující se v průmyslu. Nad defekty je vytvořena obecnější kategorizace. Následně jsou z každé kategorie vybrány dvě vady. Na první vadě je nalezena vhodná kombinace parametrů. U hledání jsou zohledněny efekty různého předzpracování a reprezentace snímku pomocí odlišných barevných prostorů. S nalezenými parametry je provedena segmentace druhé vady. Tak je otestována schopnost generalizace a vhodnost použití algoritmu pro vady dané kategorie. Úspěšnost segmentace je měřena metrikou IOU. Úspěšnější algoritmy dosáhly průměrného IOU měřeného přes všechny snímky jedné vady 90 %. U testu generalizace bylo v některých případech dosaženo průměrného IOU 53 %.

Klíčová slova porovnání algoritmů, segmentace objektů, detekce defektů, zpracování obrazu, strojové vidění, active contour, random walker, superpixels, region adjacency grafy, prahování Otsu

Abstract

The paper compares algorithms designed for image segmentation. Three algorithms belonging to the category of superpixels (felzenszwalb, SLIC and quickshift), two representatives of active contour models (snakes and level sets), random walker, region adjacency graphs and Otsu thresholding are compared. For this purpose, defects appearing in the industry are described. A more general categorization splitting defects into groups is made. Subsequently, two defects are selected from each category. A suitable combination of algorithm parameters is found on the first defect. Different preprocessing and color representation of the image are accounted for when the search is carried out. With the found parameters, a segmentation of the second defect is performed. Thus, the generalization capability and the suitability of the algorithm for the defects of the given category are tested. The success of the segmentation is measured by the IOU metric. The more successful algorithms achieved an average IOU measured over all images of a single defect of 90 %. In generalization test an average IOU of 53 % was achieved in some cases.

Keywords algorithm comparison, object segmentation, defect detection, image processing, computer vision, active contour, random walker, superpixels, region adjacency graphs, Otsu thresholding

Kapitola 1
Úvod

Segmentace je jeden ze základních kroků při zpracování obrazu, na kterém stojí následná extrakce příznaků či klasifikace. Pokud se podaří úspěšně vysegmentovat objekt zájmu, mohou být následující kroky snadnější.

Na zkoumání nových postupů nemusí být v průmyslu čas ani prostředky. Při řešení problému se většinou zvolí algoritmus, který se již osvědčil dříve. V praxi se využívají algoritmy, které byly vymyšleny před 20–30 lety. Novější, většinou sofistikovanější algoritmy by mohly dosahovat lepších výsledků.

Jednodušší algoritmy využívají omezené množství informací obsažené v obrázku (například thresholding nevyužívá prostorovost). Většinou dělají předpoklady ohledně struktury dat (Otsu thresholding předpokládá, že histogram je bimodální). Pokud je řešen problém, kde daný předpoklad nemůže být splněn, konkrétní algoritmus je nepoužitelný. Problém je poté nutné řešit jinou metodou.

Novější algoritmy spojují více postupů dohromady nebo fungují oproti starším algoritmům na kompletně odlišné bázi. Proto by mohly nabídnout spolehlivější výsledky. Ačkoliv doba běhu a paměťová náročnost algoritmu zůstávají stále důležité (speciálně pro systémy reálného času), s aktuálně dostupným výpočetním výkonem lze využít algoritmus, který má marginálně delší dobu běhu. Proto je vhodné přezkoumat, zdali aktuálně používaná řešení v průmyslových aplikacích jsou nejlepší možná.

Kapitola 2
Rešerše

Segmentační algoritmy mají rozsáhlé využití v průmyslu. Pro segmentaci lze využít neuronové sítě, což je přístup, který se v posledních letech stal populárním díky stále se zvyšujícímu výpočetnímu výkonu počítačů. Algoritmy nevyužívající neuronovou síť se řadí do kategorie tradičního přístupu. Typicky jsou méně výpočetně náročné, méně sofistikované a díky tomu více transparentní. Oba přístupy mají své výhody i nevýhody, nelze říct, že některý z postupů je lepší.

2.1 Tradiční přístup a neuronové sítě

Autoři [1] porovnávají charakteristiky tradičních technik s neuronovými sítěmi. Ačkoliv za poslední roky se neuronové sítě zlepšily natolik, že dosahují ohromujících výsledků a v mnoha úlohách předčily tradiční techniky, nejsou bez svých nedostatků. Neuronové sítě jsou tak dobré, jako vstupní data. U tradičních technik není potřeba velké množství dat k jejich natrénování. Do postupu zpracování je typicky vnášena expertní znalost. Navíc díky explicitně vyjádřenému postupu je možno vysvětlit koncové rozhodnutí tradičního algoritmu. U neuronových sítí činí interpretace výsledného rozhodnutí problém. Většinou se neví, jakým způsobem se k rozhodnutí došlo. Navíc může být výsledek v závislosti na vstupních datech nerobustní (drobná změna v datech může dramatickým způsobem změnit výsledek).

Ke zmíněným myšlenkám [2] přidávají, že tradiční přístup může být k řešení problému výhodnější, protože není určen trénovacími daty a tedy je více generalizovatelný. Tradiční přístup bývá transparentnější, rychlejší a efektivnější z pohledu spotřebované energie. Trénování neuronových sítí trvá dlouho a je nákladné. Vstupní data musí být anotována, což vyžaduje lidskou práci. Tradiční techniky ale nedokážou vyřešit náročnější problémy strojového vidění. Je potřeba zvolit techniku vhodnou pro řešený problém. Někdy je použit hybridní přístup, kdy je využito tradičních technik k předzpracování dat pro neuronové sítě. Autoři usoudili, že tradiční techniky strojového vidění zůstávají i dnes relevantní.

2.2 Porovnání tradičních metod

Autoři [3] zmapovali využití segmentačních metod v lékařském průmyslu. Algoritmy rozdělili do čtyřech kategorií: *prahovací*, *podobnostní*, *založené na hranové detekci* a *založené na shlukování*. U každého algoritmu popsali jeho výhody i nevýhody. Pro algoritmy vymykající se zmíněnému dělení vytvořili autoři kategorii *ostatní*, do které jsou mimo jiné zařazené level sets. Autoři také zmiňují algoritmy Otsu, k-means a meanshift, který je úzce spojen s algoritmem quickshift. Dále komentují vhodnost použití algoritmů na základě druhu snímkování (CT, MRI, rentgen a ultrazvuk). Svá doporučení zakládají na vlastnostech algoritmu, efektivitu netestují na reálných

datech. Došli k závěru, že každá z metod má své výhody a nevýhody, neexistuje univerzální metoda vhodná pro řešení všech problémů. Jako největší slabiny algoritmů označili přesnost a rychlost zpracování.

Článek [4] představuje rozsáhlé a hluboké zmapování dostupných algoritmů spolu s popisem jejich vlastností. U motivace segmentace dávají autoři důraz na lidské vnímání kontur. Došli k závěru, že detekce kontur dosáhla vysoký stupeň sofistikovanosti. V průběhu let lze pozorovat pokrok v robustnosti algoritmů. Většině segmentačních algoritmů se ale nedaří zachytit vizuální informace vyšší třídy jako symetrie, pokračování chybějící kontury nebo tvar objektu. Myslí si, že výzkum algoritmů zachycující zmíněné vizuální podněty má vysoký potenciál.

Článek [5] řeší problém detekce defektů při automatickém umisťování vláken v leteckém průmyslu. Za tímto účelem autoři provedli porovnání 30 algoritmů. Nejlepší se ukázal postup využívající buď adaptivního prahování, nebo prahování založeného na výpočtu standardní odchylky. Algoritmy dokázaly detekovat defekt v obrázku s přesností 97 %. Samotná segmentace vady se ukázala složitější s průměrnou mírou překrytí s ground truth 20 %. Díky dostatečně dobré lokalizaci defektu a dalšímu zpracování to autoři nepovažují za problém.

2.3 Využití segmentačních algoritmů

Autoři [6] mapují použití strojového vidění v průmyslu. Tradičně úkoly vizuální inspekce a kontroly kvality plnili lidé. Lidská práce je ale pomalá a náchylná k chybám kvůli repetitivní povaze činnosti. Strojové vidění řeší zmíněné problémy a tak přináší snížení celkových nákladů na výrobu. Jako konkrétní příklady průmyslů využívající strojové vidění uvádějí potravinářský průmysl, výrobu automobilů, lékařský průmysl, inspekci tištěných obvodů, zpracování oceli, dřeva a navádění robotů.

Článek [7] řeší problém segmentace listů rostlin za účelem fenotypizace. Srovnává 4 postupy použité pro řešení problémů v soutěži Leaf Segmentation Challenge. První postup převádí obrázek do barevného prostoru LAB. K oddělení rostliny od pozadí využívá 3D histogram. Pro segmentaci samotných listů používá vzdálenostní transformaci spolu s algoritmem region growing. Druhý postup opět převádí obrázek do barevného prostoru LAB. Obrázek je následně segmentován pomocí superpixelů za účelem najít středy listů. Středy využívá jako počáteční body pro zaplavování pomocí watershed transformace. Třetí postup využívá Chamfer matching. Postup porovnává listy v obrázku s databází šablon. Čtvrtý postup je dvoufázový. Rostlinu od pozadí odděluje za pomocí neuronové sítě. Listy odděluje za pomocí watershed transformace. Navržené postupy byly schopny segmentovat rostlinu od pozadí s průměrnou účinností 90 % (Dice score). Segmentace individuálních listů již nebyla tak úspěšná, průměrná účinnost byla 62 %. Problém představovaly překrývající se listy a listy s malou velikostí.

V [8] se autoři zabývali segmentací želatinové kapsle. Pro jejich případ použití navrhli segmentaci za použití Sobelova filtru (filtr slouží k nalezení hran v obrázku) v kombinaci s Otsu prahováním. Postup srovnali se segmentací využívající neuronovou síť. Ukázalo se, že segmentace jednoho obrázku pomocí neuronové sítě trvala přibližně 2,54 s, za to segmentace využívající Sobelův filtr trvala 80 ms, přičemž dosahovala dostatečné kvality. Pro řešený problém byla doba běhu algoritmu kritická, proto se autoři rozhodli využít první postup.

Autoři [9] se zabývali segmentací potravin, přesněji hranolek a plátků jablek. K tomuto účelu navrhli nový algoritmus. Obrázek byl převeden do barevného prostoru LAB. Díky tomu byla segmentace rezistentní vůči změně světlosti segmentovaného objektu. Jednotlivé hodnoty pixelů A a B byly zaneseny do grafu (složka L nebyla použita). Použitím polynomiální regrese byly body proloženy polynomem. Pixely byly rozděleny do dvou tříd na základě vzdálenosti od proloženého polynomu. Pro jejich případ použití se ukázalo, že navržený algoritmus je robustnější než prahování.

V [10] autoři řešili problém detekce plísně na vinných hroznech Chardonnay. K pořízení snímků využili hyperspektrální kamery. Kvůli obrovskému množství dat využili redukci dimen-

zionality pro nalezení spektrálních pásem obsahující data potřebná k segmentaci. K následné segmentaci využili náhodné lesy. Klasifikační přesnost pro samotné bobule činila 99,8 %, pro celý svazek pak 87 %.

[11] řeší úlohu segmentace části plic ve snímku výpočetní tomografie (CT). Byl navržen postup segmentace za pomoci fuzzy množin. Dosáhli přesnosti 98 %, metrika pro měření přesnosti je ale nestandardní.

Fuzzy množin také využili [12] pro detekci prahu použitého k segmentaci defektů na kovových odlitcích s využitím rentgenového snímkování. Rentgenové snímky nejsou většinou dostatečně kontrastní k tomu, aby bylo možno využít běžně používaných metod. Jejich metoda podávala lepší výsledky než Otsu prahování.

2.4 Architektura systému strojového vidění

Proces průmyslové kontroly se typicky skládá z následující sekvence kroků:

Pořízení obrazu Jakýkoliv nedostatek může mít negativní dopad dále. Kromě výběru samotné kamery a objektivu ovlivňuje kvalitu pořízení obrazu také výběr osvětlení. Konstantní podmínky při pořizování snímku zaručují opakovatelnost predikce. Proto je žádané, aby bylo osvětlení objektu a nastavení kamery mezi snímky neměnné. Pro zaručení konstantních podmínek může být snímkovací soustava oddělená od okolních vlivů (jako je denní světlo).

Preprocessing Příprava pixelů digitálního obrázku do podoby vhodné pro další zpracování. Typickým příkladem preprocessingu je odstranění šumu, oříznutí, rotace, zvětšení/zmenšení, modifikace jasu a kontrastu, převod do jiného barevného prostoru, ekvalizace histogramu…

Segmentace Nalezení objektu a oddělení objektu od pozadí.

Extrakce vlastností Extrakce podstatných charakteristik objektů nalezených v předchozím kroku. Jedná se o redukci dimenzionality.

Klasifikace Objekt je zařazen do některé z možných skupin. Příkladem může být dělení na dobré a vadné kusy výrobku.

Různé zdroje uvádí různý počet kroků. Některé kroky jsou sloučené či rozdělené, mohou mít i odlišné pojmenování. Podstata zpracování avšak zůstává stejná [6, 13, 14].

Kapitola 3

Zkoumané algoritmy

Ke zkoumání bylo vybráno celkem 8 segmentačních algoritmů. Active contour models jsou iterační algoritmy, které deformují dodanou konturu. Podle reprezentace kontury se dělí na snakes a level sets. Random walker je grafový algoritmus, který bere inspiraci z náhodných procházek v grafu. Mezi grafové algoritmy patří také region adjacency graphs. Dále jsou popsány tři algoritmy řadící se do kategorie superpixelů: felzenszwalb, SLIC a quickshift. Otsu se řadí do kategorie prahovacích algoritmů. Nakonec je popsána filtrace pomocí Gaussianu, která se využívá pro účely předzpracování.

3.1 Konvence

Napříč kapitolami je kvůli přehlednosti dodržováno jednotné značení některých matematických objektů.

3.1.1 Značení

V následujícím textu jsou kvůli přehlednosti rozlišena čísla od vektorů. Čísla jsou sázena výchozím písmem (x), vektory jsou sázeny tučně ($\boldsymbol{x}$). Stejným způsobem jsou rozlišeny funkce o jedné proměnné ($f(x)$) od funkcí více proměnných ($f(\boldsymbol{x})$).

Symbol $\mathbb{R}$ označuje množinu reálných čísel. $\langle a, b\rangle$ reprezentuje uzavřený interval reálných čísel od a do b. Symbol $\|\cdot\|$ označuje Euklidovskou normu vektoru. Funkce $I(x, y)$ poskytuje pixel obrázku na pozici (x, y).

3.1.2 Grafy

Některé algoritmy fungují na grafové bázi. Graf G je definován jako dvojice $G = (V, E)$, kde

- V je konečná množina vrcholů,
- E je množina hran.

Pokud je graf hranově ohodnocený, přiřazuje ohodnocení hranám funkce $w : E \to \mathbb{R}$. Způsob sestavení a využití grafu je specifikován separátně u každého algoritmu.

3.2 Filtrace pomocí Gaussianu

Filtrace pomocí filtru Gaussian slouží k rozmazání obrázku. Gaussova funkce je definována jako

$$g(x, y) = Ke^{-(x^2+y^2)/2\sigma^2}, \tag{3.1}$$

kde $K \in \mathbb{R}$ je konstanta. Hyperparametr σ určuje míru rozmazání (větší hodnoty odpovídají většímu rozmazání). Následně je vytvořená středově symetrická matice G tvaru $2n-1 \times 2n-1$ pomocí vztahu

$$G(x, y) = g(x - n, y - n). \tag{3.2}$$

Prvky matice jsou normalizovány podělením každého prvku součtem prvků celé matice. Účel normalizace je zachovat energii v obrázku (součty pixelů originálního a filtrovaného obrázku jsou stejné). Filtr je na obrázek aplikován pomocí konvoluce, která je definovaná jako

$$I'(x, y) = \sum_{i,j}^{n} I(x + i, y + j) \cdot G(i, j). \tag{3.3}$$

Typicky se filtr používá k redukci malých irelevantních detailů. Také může být použit pro odstranění či zmírnění šumu [15].

3.3 Active contour models

Active contour models jsou algoritmy, které iteračně deformují konturu na základě sil v typickém případě působících z obrázku. Předpokládá se, že na rozhraní dvou regionů bude síla jiná než uvnitř jednoho regionu.

Na nalézání výsledné kontury je možno se dívat jako na problém optimalizace. Model hledá konturu, která minimalizuje funkci udávající energii kontury. Analytická řešení jsou ale složitá a nejspíš nemožná, proto jsou při řešení úloh užívány iterační algoritmy. Je důležité si uvědomit, že nalezené minimum funkce nemusí být globální (a téměř nikdy nebude).

Active contour models se dělí na dvě kategorie na základě toho, jakým způsobem reprezentují konturu: snakes a level sets. Snakes využívají explicitní (parametrickou) reprezentaci. Level sets naopak reprezentují konturu jako průnik 3D křivky a roviny. Každá z těchto reprezentaci má své výhody [15].

3.3.1 Snakes

Pro reprezentaci kontury používají snakes [16] explicitní neboli parametrickou reprezentaci kontury. Odvození algoritmu a příklad sleduje [15]. Parametrická křivka je definovaná jako

$$\boldsymbol{c}(s) = \begin{pmatrix} x(s) \\ y(s) \end{pmatrix},$$

kde $s \in \langle 0, 1 \rangle$ je reálný parametr a x, y jsou funkce z $\langle 0, 1 \rangle$ do $\mathbb{R}$ přiřazující souřadnice v rovině.

Na začátku algoritmu je definován tvar a poloha křivky v obrázku. Tradičně jsou počáteční podmínky dodány uživatelem, avšak lze je definovat automaticky. Následně je parametrické křivce přiřazena energie, která bude minimalizována.

$$E(\boldsymbol{c}) = E_{internal}(\boldsymbol{c}) + E_{external}(\boldsymbol{c}). \tag{3.4}$$

Je možné si všimnout, že energie křivky se skládá ze dvou částí.

$E_{internal} = E_{elastic} + E_{bending}$ reprezentuje vnitřní stav křivky. Díky elastické energii se snake chová jako gumička, snaží se scvrknout dovnitř. Aby nedošlo ke scvrknutí do jednoho bodu, musí naproti energii $E_{internal}$ působit energie $E_{external}$. Pro celou křivku se $E_{elastic}$ dá vypočítat jako

$$E_{elastic} = \frac{1}{2} \int_0^1 \alpha \left\| \frac{\partial \boldsymbol{c}(s)}{\partial s} \right\|^2 \mathrm{d}s. \tag{3.5}$$

Z technických důvodů byla před integrál přidána konstanta 1/2. Konstanta α je hyperparametr (parametr, který zadává uživatel), který kontroluje, jak moc je elastická energie silná. Čím větší α, tím se snake bude zmenšovat rychleji. $E_{bending}$ kontroluje kulatost celé kontury. Dá se vypočítat jako

$$E_{bending} = \frac{1}{2} \int_0^1 \beta \left\| \frac{\partial^2 \boldsymbol{c}(s)}{\partial s^2} \right\|^2 \mathrm{d}s. \tag{3.6}$$

Konstanta β je opět hyperparametr. Čím větší β, tím víc bude snake penalizován za ostré hrany v kontuře.

Externí energie reprezentuje síly působící z obrázku. Pro všechny body křivky pak se dá energie vypočítat pomocí vzorce

$$E_{external} = \int_0^1 E_{image}(\boldsymbol{c}(s))\, \mathrm{d}s. \tag{3.7}$$

Na volbě E_{image} záleží chování celého algoritmu. V původním článku [16] autoři zkoumali tři energie, díky kterým byly snakes přitahovány k charakteristickým rysům v obrázku: *line*, *edges* a *terminations*. Metoda ze scikit-image [17], která je použita v praktické části, implementuje pouze energie *line* a *edges*. Dohromady se E_{image} dá popsat jako

$$E_{image}(x, y) = w_{line} E_{line}(x, y) + w_{edge} E_{edge}(x, y), \tag{3.8}$$

kde w_{line} a w_{edge} jsou hyperparametry. E_{line} je funkcionál vracející jas obrázku:

$$E_{line}(x, y) = I(x, y). \tag{3.9}$$

Snake je přitahován k světlejším nebo tmavším regionům podle toho, jestli je w_{line} kladné nebo záporné. E_{edge} se snaží využít hran v obrázku. K tomu využívá gradientu.

$$E_{edge}(x, y) = -\|\nabla I(x, y)\|^2. \tag{3.10}$$

V místech, kde je hrana, bude gradient největší. E_{edge} bude na hranách díky znaménku minus nejmenší. Výsledkem je to, že snake je přitahován do míst, kde jsou hrany.

Celkově je možno kontrolovat tyto hyperparametry:

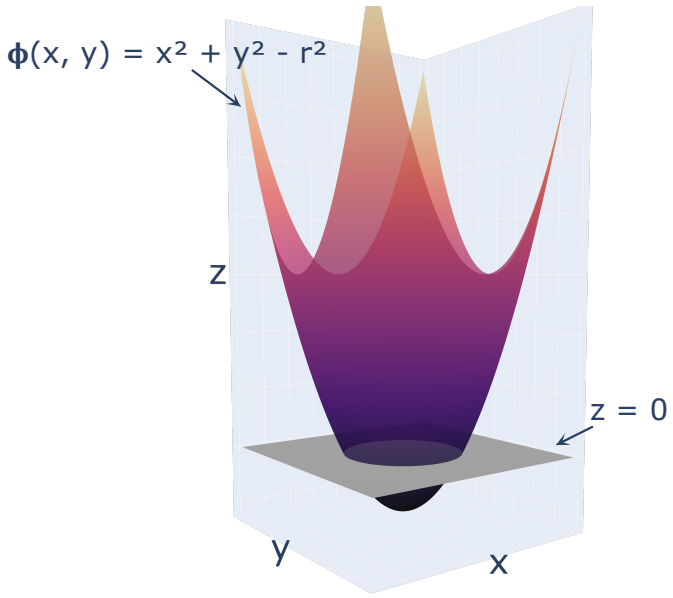
α koeficient elasticity,

β koeficient energie ohýbání,

w_{line} koeficient energie jasu,

w_{edge} koeficient energie na hranách.

Energie křivky může být reprezentovaná různými způsoby, popsán byl pouze jeden z možných způsobů. Podle volby funkce E se snakes dělí na poddruhy. Příkladem poddruhů jsou balloon snake nebo gradient vector flow snake [15].



Obrázek 3.1 Ukázka implicitního práce s konturou. Množina bodů průniku funkce ϕ s rovinou $z = 0$ tvoří zero level set.

3.3.2 Level sets

Level sets [18] jsou založeny na implicitním způsobu reprezentace kontury. Kruh se středem v počátku v xy-rovině o poloměru r lze popsat rovnicí

$$x^2 + y^2 = r^2.$$

Rovnici lze přepsat do podoby

$$x^2 + y^2 - r^2 = 0.$$

Nyní můžeme definovat funkci

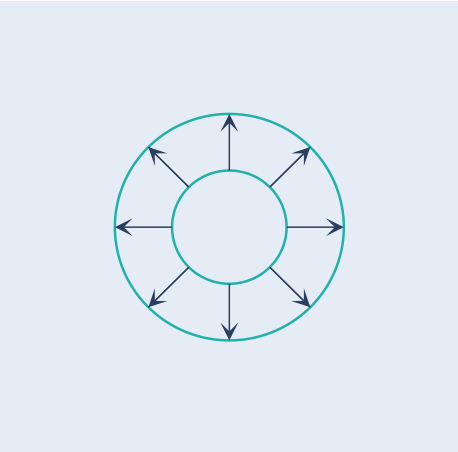
$$\phi(x, y) = x^2 + y^2 - r^2.$$

Body splňující $\phi(x, y) = 0$ tvoří počáteční kruh. Tvoří takzvaný *(zero) level set*, což je průnik funkce $\phi(x, y)$ s rovinou $z = 0$ viz obrázek 3.1. Dostanou označení Ω_0. Body, pro které platí $\phi(x, y) < 0$ budou považovány za vnitřek kontury a dostanou označení Ω^-. Podobně, body, pro které platí $\phi(x, y) > 0$ budou považovány za vnějšek kontury a dostanou označení Ω^+. [15]

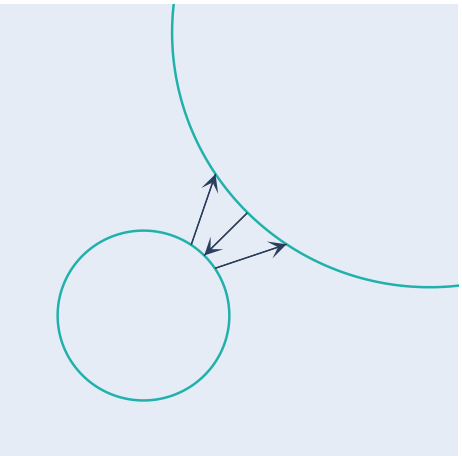
Implicitní reprezentace kontury má oproti explicitní mnoho výhod. Nechť je uvážena situace, kdy doprostřed rybníka je vhozen kámen. Vlnu, která se bude šířit od místa vhození do všech stran lze modelovat jako kruh, který se rozpíná od počátku (v čase t má poloměr t) viz obrázek 3.2. Pro účely simulace je možno kruh modelovat explicitním způsobem. Bude tedy navzorkován body na dostatečně mnoha místech. Rozpínání kruhu bude modelováno pohybem jednotlivých bodů.

Zde ale nastává první problém. Kruh se postupem času zvětšuje a proto se body od sebe vzdalují. Efektivně se tím snižuje rozlišení, jakým je kruh reprezentován. Kromě problému s rozlišením by pro simulaci rozpínání jednoho kruhu mohl model dostačovat.

Další problém ale nastane, když budou místo jedné simulovány vlny dvě (viz obrázek 3.3). Reprezentovat spojení a rozpojení (topologické změny) kontur vyjádřených explicitním způsobem je náročné. Implicitní způsob reprezentace tímto problémem netrpí, zvládne vyjádřit topologické změny bez větších problémů. Dva kruhy budou modelovány jako průnik funkce ϕ s rovinou. Spojení vln se promítne pouze do změny tvaru funkce ϕ (viz obrázek 3.4a). Důležité ale je, že před i po spojení je funkce ϕ pouze jedna (viz obrázek 3.4b). [19]



Obrázek 3.2 Modelování vln: vlna se šíří od počátku. Převzato z [19].



Obrázek 3.3 Modelování vln: dvě vlny před spojením. Převzato z [19].

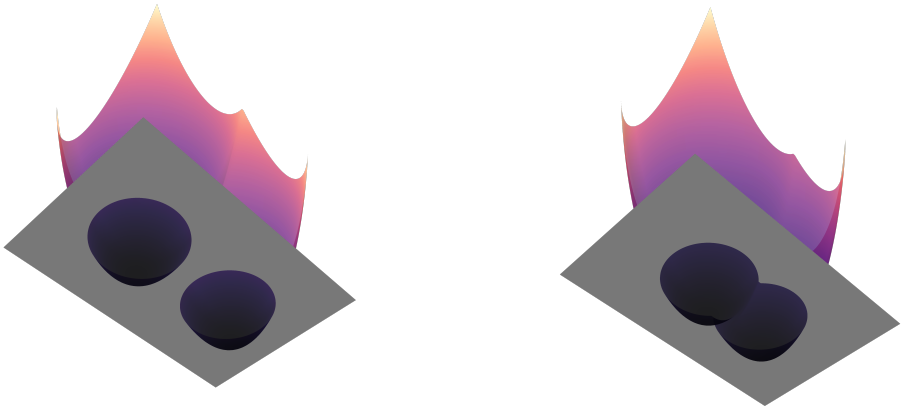
3.3.2.1 Level set equation

Pro účely sledování vývoje kontury je zaveden funkci ϕ ještě parametr t, který bude reprezentovat čas. Uvažme bod $\boldsymbol{w} = (x, y)^T$, který po celý čas patří do kontury. Funkce $\boldsymbol{w}(t)$ pak bude sledovat, jakým způsobem se pozice bodu na kontuře vyvíjí. Pro tuto funkci platí

$$\phi(\boldsymbol{w}(t), t) = 0.$$

Pokud je známa počáteční kontura $\phi_0 = \phi(\boldsymbol{w}(0), 0)$, bylo by možné sledovat vývoj kontury díky silám, které na konturu v obrázku působí. Vývoj lze přirozeně popsat derivací, tedy $\frac{\partial \phi}{\partial t}$. Použitím řetězového pravidla lze získat

$$\begin{aligned} \frac{\partial \phi(\boldsymbol{w}(t), t)}{\partial t} &= 0, \\ \frac{\partial \phi}{\partial \boldsymbol{w}} \cdot \frac{\partial \boldsymbol{w}}{\partial t} + \frac{\partial \phi}{\partial t} \cdot \frac{\partial t}{\partial t} &= 0, \\ \frac{\partial \phi}{\partial \boldsymbol{w}} \cdot \frac{\partial \boldsymbol{w}}{\partial t} + \frac{\partial \phi}{\partial t} &= 0. \end{aligned} \tag{3.11}$$



(a) Dvě vlny před spojením. **(b)** Dvě vlny po spojení.

Obrázek 3.4 Ukázka modelování topologických změn pomocí level setů. Převzato z [19].

V kontextu rovnice 3.11 jsou výrazy $\frac{\partial \phi}{\partial \boldsymbol{w}}$ a $\frac{\partial \boldsymbol{w}}{\partial t}$ vektory, $\cdot$ reprezentuje standardní skalární součin.

$$\frac{\partial \phi}{\partial \boldsymbol{w}} = \begin{pmatrix} \frac{\partial \phi}{\partial x} \\ \frac{\partial \phi}{\partial y} \end{pmatrix}, \quad \frac{\partial \boldsymbol{w}}{\partial t} = \begin{pmatrix} \frac{\partial x}{\partial t} \\ \frac{\partial y}{\partial t} \end{pmatrix}.$$

Nyní si je možno všimnout, že $\frac{\partial \phi}{\partial \boldsymbol{w}} = \nabla \phi$ a tedy

$$\begin{aligned} \frac{\partial \phi}{\partial \boldsymbol{w}} \cdot \frac{\partial \boldsymbol{w}}{\partial t} + \frac{\partial \phi}{\partial t} = \\ \nabla \phi \cdot \frac{\partial \boldsymbol{w}}{\partial t} + \frac{\partial \phi}{\partial t}. \end{aligned} \tag{3.12}$$

Výraz $\frac{\partial \boldsymbol{w}}{\partial t}$ představuje sílu působící na bod. Pokud se předpokládá, že je tato síla na konturu kolmá, výraz lze vyjádřit ve tvaru

$$\frac{\partial \boldsymbol{w}}{\partial t} = F(\boldsymbol{w}) \frac{\nabla \phi}{\|\nabla \phi\|}, \tag{3.13}$$

kde $F(\boldsymbol{w})$ udává velikost síly a $\frac{\nabla \phi}{\|\nabla \phi\|}$ udává směr síly, tedy vektor, který je znormován na jedničku. Právě funkce F je prvek, který řídí vývoj kontury a do kterého se promítnou vlastnosti obrázku či křivky. S touto znalostí je možno přepsat rovnici 3.12 do tvaru

$$\begin{aligned} F \cdot \nabla \phi \cdot \frac{\nabla \phi}{\|\nabla \phi\|} + \frac{\partial \phi}{\partial t} = 0, \\ F \cdot \|\nabla \phi\| + \frac{\partial \phi}{\partial t} = 0, \end{aligned} \tag{3.14}$$

protože $\nabla \phi \cdot \nabla \phi = \|\nabla \phi\|^2$. Rovnice 3.14 se nazývá *level set equation*. Ačkoliv byla rovnice odvozena pro nulový level set ($\phi(\boldsymbol{w}, t) = 0$), stejný vztah platí pro level set protínající se s rovinou v jakékoliv výšce ($\phi(\boldsymbol{w}, t) = c$ pro $c \in \mathbb{R}$). Díky této rovnici lze popsat vývoj kontury ϕ_0 a získat tak tvar ϕ v libovolném čase t. Fakt je více zřejmý, pokud je rovnice přepsána do tvaru

$$\frac{\partial \phi}{\partial t} = -F \cdot \|\nabla \phi\|. \tag{3.15}$$

Jedná se o diferenciální rovnici, která se řeší numerickými metodami [15].

3.3.2.2 Silová funkce

Evoluce kontury se kontroluje pomocí funkce F. Funkci lze zvolit mnoha způsoby, podobně jako u Snakes se tak level sets dělí na poddruhy [15].

Funkce minimalizována v původním článku [18] je zavedena jako

$$\begin{aligned} F(\phi) = \mu \cdot \mathrm{Length}(\Omega_0) \\ + \nu \cdot \mathrm{Area}(\Omega^-) \\ + \lambda_1 \int_{\Omega^-} |I(x,y) - \overline{\Omega^-}|^2 \,\mathrm{d}x\,\mathrm{d}y \\ + \lambda_2 \int_{\Omega^+} |I(x,y) - \overline{\Omega^+}|^2 \,\mathrm{d}x\,\mathrm{d}y \end{aligned} \tag{3.16}$$

Symboly $\overline{\Omega^-}$ a $\overline{\Omega^-}$ označují průměry hodnot pixelů uvnitř a vně kontury. První výraz sleduje délku nulového level setu, druhý výraz pak sleduje plochu vnitřku kontury. Druhá půlka funkce F využívající integrály počítá rozptyl hodnot pixelů uvnitř a vně kontury, výsledek ale není normalizován. Konstanty λ_1, λ_2, μ a ν jsou hyperparametry. Typicky se volí $\lambda_1 = \lambda_2 = 1$.

3.4 Random walker

Algoritmus random walker nahlíží na obrázek jako na hranově ohodnocený graf. Graf je sestaven z obrázku tak, že pro každý pixel je vytvořený vrchol a vrcholy jsou propojeny ve čtyřokolí. Na hranách jsou váhy reprezentující, jak moc se pixely mezi sebou liší (přesná funkce je zavedena později).

Nechť pro motivaci algoritmu random walker poslouží příklad obrázku velikosti 4×4. V obrázku se nejprve vyznačí pixely, které budou patřit do separátních segmentů. Takto označené pixely se nazývají seed pointy (viz obrázek 3.5a).

Algoritmus random walker bere inspiraci z náhodných procházek. Z každého vrcholu, který není seed point, je vypuštěn random walker. Random walker má větší šanci vydat se po hraně s vyšším ohodnocením. Je spočtena pravděpodobnost, že random walker dojde do každého ze seed pointů (procházka random walkera končí, jakmile dojde do některého ze seed pointů). Daný pixel/vrchol bude patřit do segmentu, k jehož reprezentativnímu seed pointu se random walker dostane s největší pravděpodobností.

Takto formulovaný problém je ovšem těžký na výpočet. Ukáže se, že řešení je ekvivalentní s výpočtem potenciálu v elektrických obvodech, kde hrany reprezentují rezistory s odporem rovným převrácené hodnotě původní váhy hrany. Tvorba obvodu vypadá následovně: Na jeden ze seed pointů je připojen zdroj s napětím 1, ostatní seed pointy jsou uzemněny. Potenciály na vrcholech se přesně rovnají pravděpodobnostem, že random walker dojde do seed pointu s připojeným jednotkovým zdrojem. Úlohu je možno vyřešit pomocí k lineárních rovnic[1], kde k je počet seed pointů.

Váhy na hranách jsou určeny funkcí

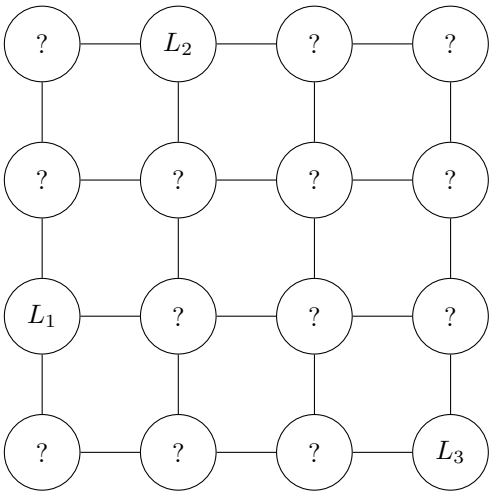
$$w(v_i, v_j) = \exp(-\beta(v_i - v_j)^2),$$

kde v_i, v_j jsou hodnoty pixelů a β je hyperparametr. Čím je rozdíl hodnot pixelů větší, tím má hrana menší hodnotu a šance na to, že se random walker po ní vydá je menší.

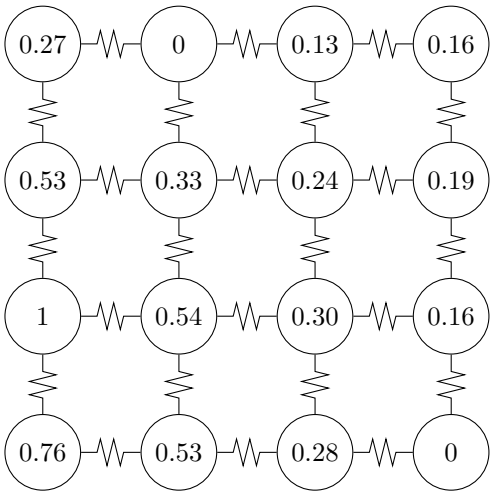
3.5 Superpixels

Algoritmy řadicí se do kategorie superpixels nejsou používané samostatně, ale ve spojení s jinou metodou. Poskytují užitečnou základní segmentaci, ale obrázek je většinou přesegmentován (má

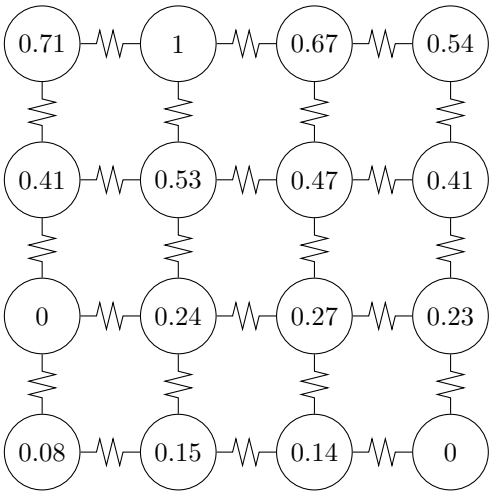
[1] Stačí $k - 1$ rovnic, pokud se využije fakt, že pravděpodobnosti se musí sečíst na 1.



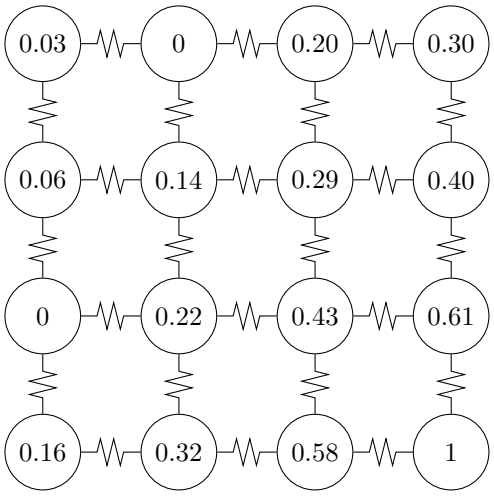
(a) Rozmístěné seed pointy.



(b) Pravděpodobnosti, že random walker z každého vrcholu dorazí k seed pointu L_1.



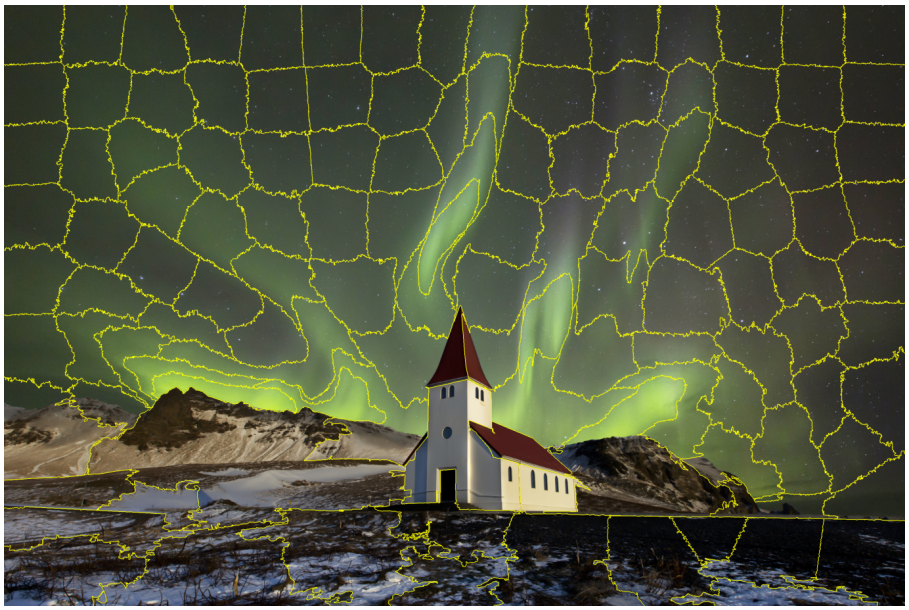
(c) Pravděpodobnosti, že random walker z každého vrcholu dorazí k seed pointu L_2.



(d) Pravděpodobnosti, že random walker z každého vrcholu dorazí k seed pointu L_3.

Obrázek 3.5 Ilustrace segmentace pomocí algoritmu random walker. Pro ilustrační účely byly všechny váhy hran nastaveny na jedničku [20].

více regionů, než je požadováno).



Obrázek 3.6 Ukázka přesegmentovaného obrázku za pomocí superpixelů (algoritmus SLIC). Obrázek převzat z [21].

3.5.1 Felzenszwalb

Felzenszwalb [22] je segmentační algoritmus pojmenovaný po svém tvůrci. Jedná se o grafový algoritmus, graf je hranově ohodnocený. Váhy $w(e)$ odpovídají Euklidovské vzdálenosti mezi hodnotami pixelů.

Vnitřní rozdíl v komponentě $C \subseteq V$ je definován jako maximální váha hrany v minimální kostře grafu komponenty $\mathrm{MST}(C, E)$

$$\mathrm{Int}(C) = \max_{e \in \mathrm{MST}(C,E)} w(e). \tag{3.17}$$

Intuice za hodnotu $\mathrm{Int}(C)$ je taková, že komponenta zůstane souvislá, pokud jsou odebrány hrany váhy větší než $\mathrm{Int}(C)$.

Vnější rozdíl mezi komponentami $C_1, C_2 \subseteq V$ je definován jako váha minimální hrany spojující tyto dvě komponenty

$$\mathrm{Dif}(C_1, C_2) = \min_{v_i \in C_1, v_j \in C_2 : (v_i, v_j) \in E} w(v_i, v_j). \tag{3.18}$$

Pokud neexistuje hrana spojující komponenty C_1 a C_2, potom $\mathrm{Dif}(C_1, C_2) = \infty$.

Minimální vnitřní rozdíl je zaveden jako

$$\mathrm{MInt}(C_1, C_2) = \min(\mathrm{Int}(C_1) + \tau(C_1), \mathrm{Int}(C_2) + \tau(C_2)). \tag{3.19}$$

Funkce τ slouží jako prahovací funkce, která kontroluje, o kolik musí být vnější rozdíl mezi komponentami větší než vnitřní rozdíl na to, aby algoritmus uvažoval mezi komponentami hranici. Funkce je definovaná jako

$$\tau(C) = k/|C|. \tag{3.20}$$

kde $|C|$ značí počet vrcholů v komponentě C. Větší hodnota kladného hyperparametru k vyjadřuje preferenci pro větší výsledné komponenty.

Implementace algoritmu v knihovně scikit-image [17] nabízí k ladění hyperparametr `min_size` kontrolující velikost výsledných komponent. Není součástí originálního algoritmu, jeho realizaci zajišťuje postprocessing.

Samotný segmentační algoritmus bere inspiraci z Kruskalova algoritmu pro hledání minimální kostry grafu. Bylo dokázáno, že ačkoliv algoritmus dělá pouze hladová rozhodnutí, výsledná segmentace není ani moc jemná, ani moc hrubá[2].

Algoritmus 1 Felzenszwalb [22]

function FELZENSZWALB(Graf $G = (V, E)$ s n vrcholy a m hranami)
  Seřaď hrany E vzestupně podle vah do $\pi = (e_1, \dots e_m)$
  Každému vrcholu vytvoř vlastní komponentu
  for $q := 1$ **to** m **do**
    Označ koncové vrcholy e_q jako $e_q = (v_i, v_j)$
    C_i je komponenta obsahující v_i
    C_j je komponenta obsahující v_j
    if $C_i \neq C_j$ **and** $w(e_q) \leq \mathrm{MInt}(C_i, C_j)$ **then**
      Spoj komponenty C_i a C_j
    end if
  end for
  return komponenty, které tvoří výslednou segmentaci
end function

3.5.2 SLIC

Základní fungování algoritmu SLIC [23] stojí na shlukovacím algoritmu k-means. Nechť $\mathcal{X} = \{\boldsymbol{x}_i \in \mathbb{R}^d \mid i \in \{1, \dots, M\}\}$ je množina datových bodů. Na začátku rozmístí k-means do prostoru k bodů, které budou považovány za středy shluků. Následně iteruje ve smyčce; Napřed se každému středu vytvoří množina obsahující body, které jsou k uvažovanému středu blíž než k ostatním středům. Pak se středy přesunou do geometrických středů těchto množin. Velikost všech posunutí (ve smyslu normy vektoru posunutí) je sečtena a označena jako reziduální error E. Algoritmus končí, jakmile E klesne pod specifikovaný práh.

Algoritmus 2 k-means [23]

function k-MEANS(množina bodů $\mathcal{X}$)
  Je rozmístěno k středů $\mu_1, \dots, \mu_k$
  repeat
    Body datasetu jsou rozmístěny do shluků $C_i = \{\boldsymbol{x} \in \mathcal{X} \mid i = \arg\min_{j \in \{1,\dots,k\}} \|\boldsymbol{x} - \boldsymbol{\mu}_j\|\}$
    Středy jsou přesunuty do geometrických středů množin $C_1, \dots, C_k$
    Je vypočten reziduální error E
  until $E \leq$ práh
end function

Algoritmus SLIC přizpůsobuje k-means úloze segmentace v obraze. Středy inicializuje po celém obrázku rovnoměrně (na mřížce). Pro čtvercový obrázek o velikosti jedné strany N je vzdálenost dvou středů $S = \sqrt{N/k}$, Protože se předpokládá, že výsledný shluk/superpixel bude mít velikost kolem $S \times S$, hledání nejbližších sousedů je provedeno pouze v regionu o velikosti $2S \times 2S$ kolem středu shluku. Díky této úpravě nezávisí výpočetní složitost algoritmu na počtu sledovaných středů k.

[2] V původním článku je pojem „ani moc jemná, ani moc hrubá“ pevně definovaný. Šikovně je zvolen predikát, pomocí kterého jsou tyto vlastnosti dokázány.

Před začátkem shlukováním spojí SLIC dohromady prostorovou a barevnou složku pixelu. Nechť $\begin{pmatrix} l_i & a_i & b_i & x_i & y_i \end{pmatrix}^T$ je vektor i-tého pixelu, kde část $\begin{pmatrix} l_i & a_i & b_i \end{pmatrix}^T$ tvoří barevnou část a $\begin{pmatrix} x_i & y_i \end{pmatrix}^T$ tvoří prostorovou část. Jelikož hodnoty barev a souřadnic pixelů mají jiné rozsahy, vzdálenost mezi barvou a prostorem je uvažována separátně.

$$d_c = \sqrt{(l_j - l_i)^2 + (a_j - a_i)^2 + (b_j - b_i)^2}, \tag{3.21}$$

$$d_s = \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2}. \tag{3.22}$$

Následně je definovaná nová míra vzdálenosti, která spojuje d_c a d_s do jednoho čísla

$$D = \sqrt{\left(\frac{d_c}{N_c}\right)^2 + \left(\frac{d_s}{N_s}\right)^2}. \tag{3.23}$$

N_s je zvolena jako $S = \sqrt{(N/k)}$, což odpovídá maximální vzdálenosti ve shluku. Konstantu N_c, která by měla odpovídat rozsahu barev, je v praxi těžké určit, jelikož se dopředu neví, jaký rozsah barev bude obrázek a shluk obsahovat. Proto se $N_c = m$ prohlásí za hyperparametr a rovnice se přepíše do tvaru

$$D = \sqrt{d_c^2 + \left(\frac{d_s}{S}\right)^2 m^2}. \tag{3.24}$$

Hyperparametrem m lze kontrolovat váhu mezi prostorovostí a barevností (čím větší je parametr m, tím větší je důraz na prostorovost).

3.5.3 Quickshift

Quickshift [24] se řadí do rodiny shlukovacích algoritmů, které hledají modus (nejčastější hodnotu) v rozdělení dat. Bere inspiraci z algoritmů mean shift [25] a medoid shift [26], které budou pro pochopení quickshift popsány jako první.

Nechť $\mathcal{X} = \{\boldsymbol{x}_i \in \mathbb{R}^d \mid i \in \{1, \ldots, N\}\}$ je množina datových bodů. Hledání modu využívá koncept zvaný *kernel density estimation* nebo také *Parsen density estimation*, který slouží pro ohodnocení hustoty bodů z $\mathcal{X}$ v daném bodě pomocí

$$f(\boldsymbol{x}) = \frac{1}{N} \sum_{i=1}^{N} \Phi(\boldsymbol{x} - \boldsymbol{x}_i), \quad \boldsymbol{x} \in \mathbb{R}^d, \tag{3.25}$$

kde $\Phi(\boldsymbol{x})$ je kernel, typicky Gaussian. Algoritmy, které budou následovat předpokládají, že $\Phi(\boldsymbol{x})$ se dá napsat ve tvaru $\phi(\|\boldsymbol{x}\|^2)$, kde $\|\cdot\|$ je Euklidovská norma. Pro nalezení modu bude pro každý bod $\boldsymbol{x}_i$ vytvořena cesta $\{\boldsymbol{y}_k \in \mathbb{R}^d \mid k \in \{1, \ldots, K\}\}$. Počáteční bod bude položen jako $\boldsymbol{y}_0 = \boldsymbol{x}_i$. Body, které zkonvergují ke stejnému modu budou ve stejné třídě. Volba cesty k odhadovanému modu odlišuje jednotlivé algoritmy od sebe.

Mean shift volí další krok na cestě bodu jako

$$\boldsymbol{y}_{k+1}^{\text{mean}} = \arg\max_{\boldsymbol{y} \in \mathbb{R}^d} \sum_{i=1}^{N} \|\boldsymbol{x}_i - \boldsymbol{y}\|^2 \phi(\|\boldsymbol{x}_i - \boldsymbol{y}_k\|^2), \tag{3.26}$$

Skok odpovídá posunutí ve směru gradientu. Hledané $\boldsymbol{y}$, které maximalizuje výraz se dá explicitně vyjádřit jako

$$\boldsymbol{y}_{k+1}^{\text{mean}} = \frac{\sum_{i=1}^{N} \boldsymbol{x}_i \phi(\|\boldsymbol{x}_i - \boldsymbol{y}_k\|^2)}{\sum_{i=1}^{N} \phi(\|\boldsymbol{x}_i - \boldsymbol{y}_k\|^2)}, \tag{3.27}$$

což se dá interpretovat jako průměr vážený pomocí kernelu ϕ. Pro mean shift je potřeba specifikovat zastavovací kritérium a shlukovací strategii, která po zastavení spojí body do jedné třídy.

Medoid shift volí další krok jako

$$\boldsymbol{y}_{k+1}^{\text{medoid}} = \arg\max_{\boldsymbol{y}\in\mathcal{X}} \sum_{i=1}^{N} \|\boldsymbol{x}_i - \boldsymbol{y}\|^2 \phi(\|\boldsymbol{x}_i - \boldsymbol{y}_k\|^2). \tag{3.28}$$

Na rozdíl od mean shiftu uvažuje medoid shift pouze body z $\mathcal{X}$. Při výpočtu je potřeba díky této vlastnosti výraz počítat pro každý bod pouze jedenkrát. Jako zastavovací kritérium slouží podmínka $\boldsymbol{y}_k = \boldsymbol{y}_{k+1}$.

Quickshift se podobně jako medoid shift pohybuje pouze po bodech z $\mathcal{X}$. Jako další krok na cestě ale volí nejbližšího souseda, který má větší ohodnocení funkce P

$$\boldsymbol{y}_{k+1}^{\text{quick}} = \arg\min_{\boldsymbol{x}_i : P(\boldsymbol{x}_i) > P(\boldsymbol{y}_k)} \|\boldsymbol{y}_k - \boldsymbol{x}_i\|^2, \quad P(\boldsymbol{x}) = \frac{1}{N} \sum_{i=1}^{N} \phi(\|\boldsymbol{x} - \boldsymbol{x}_i\|^2). \tag{3.29}$$

Algoritmus spojuje všechny body do jednoho orientovaného stromu (graf je acyklický, protože algoritmus vždy volí bod, který zvyšuje ohodnocení P). Rodič vrcholu je nejbližší soused s větším P. Pro rozdělení bodů do shluků je definován práh τ, který udává maximální vzdálenost mezi body. Strom je pak prořezán (hrany mezi vrcholy, jejichž vzdálenost přesahuje τ, budou odstraněny) a vznikne les, jehož vrcholy budou tvořit jednotlivé shluky. Při ladění algoritmu je možné vyzkoušet vícero hodnot τ, ale počáteční strom bude vždy stejný.

Před začátkem shlukování se spojí prostorová a barevná část pixelu do jednoho vektoru (podobně, jako to dělá algoritmus SLIC). Vyvažování mezi příspěvkem prostorové a barevné složky řeší implementace algoritmu ve scikit-image [17] tak, že barevná složka je před začátkem algoritmu přenásobena konstantou.

3.6 Region adjacency graphs

Region adjacency graphs (RAGs) [27] jsou podobné přístupu se jménem region growing. Z názvu je patrné, že se jedná o grafový přístup.

V prvé řadě je obrázek předsegmentován. Autoři původního článku využívají region growing nebo watershed, avšak v této práci je k počáteční segmentaci využito superpixel algoritmů Quickshift, SLIC nebo Felzenszwalb. Počítá se s tím, že tato segmentace je moc jemná.

Nad počáteční segmentací vytvoříme hranově ohodnocený graf. Každý vrchol reprezentuje předsegmentovaný region. Mezi vrcholy je natažená hrana, pokud regiony spolu sousedí. Váhy hran jsou zvoleny jako

$$w(v_i, v_j) = \|\boldsymbol{\mu}_i - \boldsymbol{\mu}_j\|^2, \tag{3.30}$$

kde $\|\cdot\|$ je Euklidovská norma, $\boldsymbol{\mu}_i$, $\boldsymbol{\mu}_j$ představují průměrné hodnoty barev v regionech R_i, R_j a $v_i \in R_i$, $v_j \in R_j$. Pro upřesnění, pokud je obrázek reprezentován v barevném prostoru RGB, vektor $\boldsymbol{\mu}_i$ bude po složkách obsahovat průměrné hodnoty červené, zelené a modré barvy obsažené v regionu R_i.

Pro úlohu detekce defektů je využit algoritmus, který prochází hrany v grafu vzestupně podle vah. Koncové vrcholy hran jsou spojovány, dokud v grafu nezbudou pouze dva vrcholy. Tak je obrázek vysegmentován na popředí a pozadí. Jako popředí se bere vrchol, ve kterém je méně pixelů. Popsaný přístup se podobá hierarchickému shlukování. Přístup ke spojování není jediný možný, autoři původního článku proces spojování zastavují, když váha hrany přesáhne zadaný práh.

3.7 Otsu

Prahování je technika, která dělí pixely typicky do dvou tříd[3] podle jejich hodnoty. Na začátku je zvolen práh, což je číslo. Pixely jsou následně rozděleny podle určeného kritéria. V tomto případě jsou pixely děleny podle toho, zdali mají menší/větší hodnotu než zvolený práh.

Práh lze volit manuálně, pak je ale do postupu vnášena apriorní znalost. Prahování Otsu je metoda, která se snaží nalézt optimální globální práh. K tomu využívá koncept ze statistiky zvaný *between class variance*. Původní článek [28] vyšel v roce 1979, jedná se o starou, ale dosud hojně používanou techniku.

Nechť pixely v obrázku nabývají L šedých hodnot $\{1, \ldots, L\}$. Nechť n_i je počet pixelů s hodnotou i a nechť $N = n_1 + n_2 \cdots + n_L$. p_i bude představovat relativní četnost hodnoty pixelu v obrázku

$$p_i = \frac{n_i}{N}.$$

Nyní bude obrázek rozdělen na dvě třídy C_0 a C_1 prahem hodnoty k. Pravděpodobnost výskytu pixelu třídy C_1 nebo C_2 je určena výrazy

$$\omega_0(k) = \sum_{i=1}^{k} p_i, \tag{3.31}$$

$$\omega_1(k) = \sum_{i=k+1}^{L} p_i. \tag{3.32}$$

Střední hodnoty hodnot pixelů uvnitř tříd jsou dány výrazy

$$\mu_0(k) = \sum_{i=1}^{k} \frac{ip_i}{\omega_0}, \tag{3.33}$$

$$\mu_1(k) = \sum_{i=k+1}^{L} \frac{ip_i}{\omega_1}. \tag{3.34}$$

Střední hodnotu hodnot pixelů v celém obrázku nám dá výraz

$$\mu = \sum_{i=1}^{L} ip_i. \tag{3.35}$$

Rozptyly hodnot uvnitř tříd jsou vypočteny jako

$$\sigma_0^2(k) = \sum_{i=1}^{k} (i - \mu_0)^2 \, \mathrm{P}(i|C_0) = \sum_{i=1}^{k} (i - \mu_0)^2 \frac{p_i}{\omega_0} \tag{3.36}$$

$$\sigma_1^2(k) = \sum_{i=k+1}^{L} (i - \mu_1)^2 \, \mathrm{P}(i|C_1) = \sum_{i=k+1}^{L} (i - \mu_1)^2 \frac{p_i}{\omega_1} \tag{3.37}$$

Konečně, intra-class variance (rozptyl uvnitř tříd) se spočítá jako

$$\sigma_W^2(k) = \omega_0(k)\sigma_0^2(k) + \omega_1(k)\sigma_1^2(k).$$

Toto je kritérium, které je minimalizováno. [28] ale ukáže, že minimalizace intra-class variance je ekvivalentní s maximalizací between class variance σ_B^2 (rozptylu mezi třídami), kde

$$\begin{aligned} \sigma_B^2(k) &= \omega_0(k)(\mu_0(k) - \mu)^2 + \omega_1(k)(\mu_1(k) - \mu)^2 \\ &= \omega_0(k)\omega_1(k)(\mu_1(k) - \mu_0(k))^2. \end{aligned} \tag{3.38}$$

[3] Pixely je možno dělit i do více tříd.

Maximalizace between class variance vede na rychlejší algoritmus. [28] následně výraz upraví tak, aby byl pro výpočet co nejefektivnější.

Kapitola 4
Metodika testování

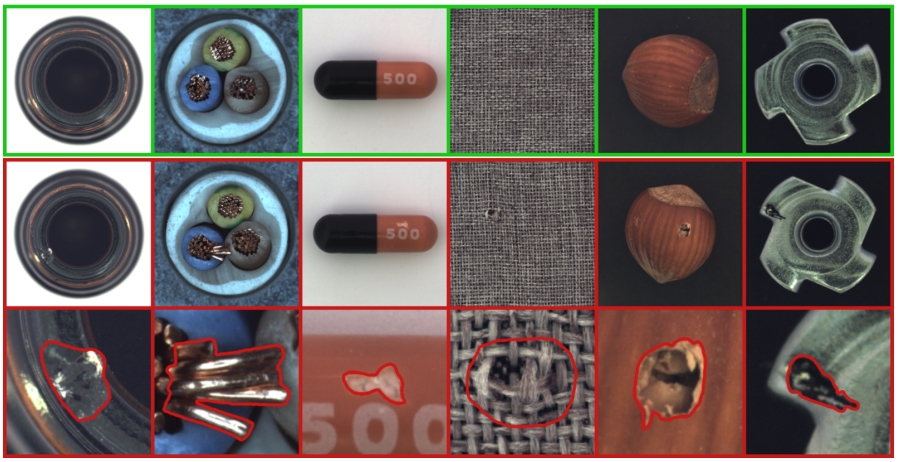
K účelu otestování algoritmů byl navržen obecný systém zpracování obrazových dat. Aby bylo zajištěno objektivity měření, jeho struktura je neměnná. Dále byly zmapovány typické vady v průmyslu a byl vybrán dataset obsahující vstupní obrazová data.

4.1 Vady v průmyslu

Pro otestování účinnosti algoritmů bylo nutno vybrat vhodný a dostatečně rozmanitý dataset. Byla vytvořena obecnější kategorizace průmyslových defektů, podle které byl dataset rozdělen.

4.1.1 Dataset

Díky jeho přednostem oproti jiným datasetům, které byly zváženy, byl pro otestování algoritmů vybrán dataset MVTec [29]. Dataset se zaměřuje na úlohu průmyslové kontroly. Typicky slouží pro porovnávaní algoritmů určených k detekci anomálií. Obsahuje obsáhlou sadu obrázků napodobující reálné problémy, které jsou v průmyslu řešeny. Dataset není syntetický, po využití jiného uměle vygenerovaného datasetu by bylo těžké říct, zdali je výsledek aplikovatelný v reálném světě.



Obrázek 4.1 Ukázka objektů a defektů objevujících se v datasetu MVTec [29]. První řádek obsahuje obrázky bez vady. Ve druhém řádku se na objektu objevuje vada. Ve třetím řádku je pak ta samá vada přiblížena a kolem vady je nakreslená kontura.

Dataset MVTec obsahuje více než 5000 snímků ve vysokém rozlišení (rozlišení se pohybuje okolo 1024×1024) rozdělených do patnácti různých kategorií podle druhu objektu či textury. Druh objektů je rozmanitý, objektem může být koberec, skleněná láhev, dřevěná deska, pilulka nebo kartáček na zuby. Obrázek 4.1 slouží jako ukázka obrazových dat obsažených v datasetu. Každý objekt zahrnuje sadu trénovacích snímků bez vad a testovacích snímku, na kterých může, ale nemusí být vada. Trénovací snímky slouží pro počáteční natrénování klasifikátorů.

Vady jsou rozděleny podle kategorie, příkladem vady na dřevě může být škrábanec, vylitá tekutina nebo vyvrtaná díra. Ke každému obrázku s vadou je dodaná ground truth, což je binární maska obsahující informaci, kde přesně se vada nachází. Ground truth není na rozdíl od jiných datasetů tvořená bounding boxem, je na pixel přesná. V práci jsou využívány pouze obrázky obsahující vadu.

4.1.2 Taxonomie defektů

Pro rozdělení vad datasetu (a vad, které se průmyslu objevují) do obecnějších kategorií bylo jako základ použito dělení navržené v [30]. Autoři navrhli rozdělení defektů na viditelné a hmatatelné. To je nejhrubější dělení, které se dále dělí na podkategorie. Celkový přehled navrženého dělení je k dispozici v tabulce 4.1.

K původnímu dělení byly přidány kategorie

- Viditelné defekty — konkrétní tvar — chyba pozice,
- Viditelné defekty — konkrétní tvar — text,
- Hmatatelné defekty — konkrétní tvar — chyba tvaru.

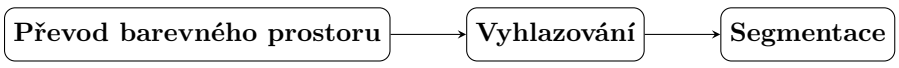
Naopak vady kategorie náraz (bump) a jáma (pit) se v datasetu neobjevují, proto jsou v tabulce 4.1 vynechány.

Význam většiny kategorií je zřejmý. Za vysvětlení stojí rozdíl mezi kategoriemi *důlek* a *díra*; Jasový průběh důlku je pozvolný, zato jasový průběh díry je okamžitý. U díry chybí povrchový materiál, který je za normálních okolností přítomen. U důlku nechybí povrchový materiál, je ale deformován. Kategorie *konkrétní tvar* obsahuje vady, které nemají ani tvar čáry, ani tvar bodu/kapky. Typicky se pro detekci těchto typů vad používají algoritmy, které nějakým způsobem zachycují nedeformovaný tvar objektu (například obecná Houghova transformace nebo algoritmy pro čtení textu).

Dataset byl rozdělen dle navrženého dělení. Kategorizace defektů je vysoce subjektivní záležitost. Některé defekty mohou nabývat charakteristik více kategorií současně, proto jsou v tabulce zařazeny současně do obou kategorií. Jak autoři [30] poznamenali, kvůli náhodné povaze výskytu defektu je každý defekt unikátní, proto je vytvoření univerzálního dělení náročné. Kompletní rozdělení datasetu lze nalézt v příloze A.

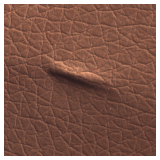
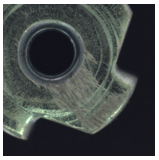
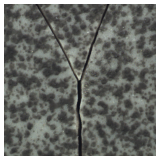
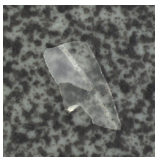
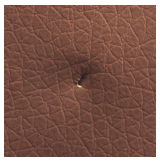
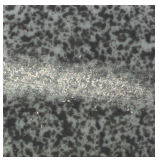
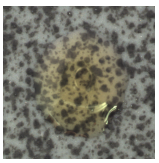
4.2 Systém zpracování dat

K ohodnocení kvalit algoritmů byl zvolen fixní systém zpracování obrazových dat. Systém má tři části: převod barevného prostoru, vyhlazování a segmentace. Části jsou zařazeny za sebou, jedna navazuje na druhou. Kroky převodu barevného prostoru a vyhlazování se považují za preprocessing. K vyhlazení obrázku je použit Gaussův filtr.



Obrázek 4.2 Architektura systému zpracování obrazových dat.

Tabulka 4.1 Rozdělení typů vad do obecnějších kategorií. Kategorie postupují zleva doprava od obecnějším ke konkrétnějším. Obrázky ukazují vady z datasetu MVTec [29].

Kategorizace			Vzorek	Kategorizace			Vzorek
Viditelné defekty	Tvar čáry	Škrábanec		Hmatatelné defekty	Tvar čáry	Záhyb	
		Škrábance				Prasklina/roztržení	
	Tvar bodu/kapky	Kontaminace			Tvar bodu/kapky	Důlek	
		Dření				Díra	
	Závislé na vzoru	Chyba tvaru				Tekutina	
		Chyba barvy			Konkrétní tvar	Chyba tvaru	
	Konkrétní tvar	Chyba pozice					
		Text					

V části segmentace je využito konkrétních segmentačních algoritmů. Snakes, level sets, random walker a Otsu do této části vstupují samostatně. Superpixel algoritmy (felzenszwalb, SLIC a quickshift) slouží k počáteční segmentaci, ze které je sestaven RAG. Část segmentace je v tomto případě dvoufázová.

4.2.1 Ladění

Ladění systému zpracování je prováděno individuálně nad každou vadou. Zahrnuje výběr a hledání nastavení segmentačního algoritmu. Prozkoušeny jsou všechny kombinace algoritmů a jejich parametrů (ve smyslu kartézského součinu). Algoritmům je možno ladit tyto parametry (značení odpovídá parametrům funkcí implementující algoritmy ve scikit-image [17]).

Snakes `alpha`, `w_edge`.

Level sets `mu`.

Random walker `beta`.

Felzenszwalb `scale`, `min_size`.

SLIC `n_segments`, `compactness`.

Quickshift `max_dist`, `ratio`, `kernel_size`.

Prahování Otsu a RAGs nemají hyperparametry, které by se daly ladit.

Většina segmentačních algoritmů je citlivá na druh předzpracování. Různé algoritmy ale mohou vyžadovat rozdílné druhy předzpracování pro svou optimální funkci. Aby byl vyloučen možný vliv předzpracování na výslednou segmentaci, inkluze a nastavení algoritmů předzpracování je zahrnuto v hledání vhodné konfigurace. Tak je nalezeno předzpracování, které je pro každý segmentační algoritmus co nejlepší. Ze stejného důvodu jsou prozkoušeny různé barevné prostory, ve kterém je obrázek reprezentován.

4.2.2 Ohodnocení úspěšnosti

Úspěšnost systému je hodnocena na dvou vadách stejné kategorie (vady jsou rozděleny dle dělení navrženého v kapitole 4.1.2). První vada slouží k nalezení nejlepší kombinace parametrů systému zpracování dat.

Jelikož jsou obě vady vybrány ze stejné kategorie, sdílejí společné charakteristické znaky. Test generalizace spočívá v aplikování systému zpracování na druhou vadu s nastavením, které bylo nalezeno na první vadě. Umožní získat hrubý odhad vhodnosti algoritmu pro segmentaci všech vad uvnitř dané kategorie. Je proveden pouze na algoritmech, které první vadu segmentovaly nejúspěšněji.

Pro ohodnocení úspěšnosti segmentace je použito metriky intersection over union (IOU). Počítá se průměrné IOU přes všechny obrázky dané vady daného objektu.

Kapitola 5
Testování

Systém zpracování dat, který hodnotí efektivitu segmentačních algoritmů je implementován a je učiněna specifikace jeho nastavení. Pro účely testování algoritmů jsou z datasetu vybrány konkrétní vady.

5.1 Implementace systému zpracování

Implementace systému zpracování a následné testování algoritmů bylo prováděno v programovacím jazyce Python. Byly využity knihovny

- scikit-image (verze 0.19.2) [17],
- scikit-learn (verze 1.0.2) [31].

Knihovna scikit-image obsahuje algoritmy určené ke zpracování obrazu. Jsou v ní implementovány všechny algoritmy zmiňované v kapitole 3. Jedná se o knihovnu s bohatou dokumentací a dobře promyšleným API. Je dbáno na přehlednost nejen API, ale i zdrojového kódu.

Knihovna scikit-learn slouží primárně pro řešení úloh strojového učení. Obsahuje ale systém pro prohledávání kombinací hyperparametrů maximalizující zadanou metriku zvaný pipelines. Pipeline je tvořena transformátory a klasifikátory, které jsou sekvenčně seřazeny za sebou (výstup jednoho objektu je vstupem druhého). Svým sekvenčním zpracováním dat koncepčně připomínají architekturu běžného systému strojového vidění.

Vstup klasifikátoru tvoří obrázek, jeho výstupem je binární maska, která označuje, zdali je pixel vadný nebo ne. Vstupem i výstupem transformátoru je obrázek. Jeho účelem je převádět data z jedné podoby do druhé.

Aby bylo možné pipelines využít pro úlohy zpracování obrazu, byly vytvořeny vlastní transformátory a klasifikátory, které zapouzdřují metody ze scikit-image. Transformátory byly použity pro zapouzdření metod preprocessingu, klasifikátory byly použity pro zapouzdření segmentačních algoritmů. Použití pipelines umožňuje běh programu hledajícího hyperparametry na více jádrech a snadné zadávání rozsahu parametrů k prohledání.

5.2 Vady vybrané k testování

Bylo využito kategorizace druhé úrovně (viz kategorizace navržená v kapitole 4.1.2). Každou kategorii reprezentují dvě konkrétní vady datasetu; první je využita pro nalezení nejlepší kombinace parametrů systému, druhá slouží k provedení testu generalizace. Jednotlivé vady byly vybrány

na základě dělení datasetu, které je možno nalézt v příloze A. Soupis vybraných vad lze nalézt v tabulkách 5.1 a 5.2.

Z testování byla vyloučena kategorie *konkrétní tvar* (viditelná i hmatatelná verze). Pro úlohu segmentace vad této kategorie se používají algoritmy odlišné od těch, které jsou zahrnuty v testované množině algoritmů. Ze stejného důvodu byla také vyřazena kategorie *závislé na vzoru — chyba tvaru*.

Tabulka 5.1 Párování kategorie s defektem datasetu použitých k testování (viditelné defekty).

Kategorie vady	Nalezení parametrů	Test generalizace
Tvar čáry	`hazelnut/cut`	`metal_nut/scratch`
Tvar bodu/kapky	`tile/glue_strip`	`metal_nut/bent`
Závislé na vzoru (chyba barvy)	`tile/oil`	`wood/color`

Tabulka 5.2 Párování kategorie s defektem datasetu použitých k testování (hmatatelné defekty).

Kategorie vady	Nalezení parametrů	Test generalizace
Tvar čáry	`leather/cut`	`tile/crack`
Tvar bodu/kapky	`leather/glue`	`wood/liquid`

5.3 Ladění systému zpracování

Implementovanému systému zpracování jsou dodány rozsahy pro hledání vhodného nastavení a jsou specifikovány počáteční podmínky algoritmů.

5.3.1 Hledání parametrů systému zpracování

Na začátku jsou systému stanoveny rozumné parametry, pro které má šanci uspět. Počáteční konfigurace je dodána ručně na základě úspěšnosti segmentace na jednom obrázku. Kolem hodnoty parametru, která byla stanovená počáteční konfigurací je následně vytvořen rozsah v obou směrech, na kterém je prováděno hledání parametrů. Jinými slovy je vytvořen interval, jehož levý bod je menší a pravý bod je větší, než počáteční hodnota. Dále je učiněn výběr vhodných kanálů v konkrétní barevné reprezentaci. Přesněji jsou prozkoušeny barevné prostory RGB, HSV a LAB. Parametry vyhlazování zůstávají napříč testovanými vadami stejné: vyzkoušeno je vyhlazování pomocí Gaussianu s parametry $\sigma \in \{1, 2, 3\}$ a dále je vyzkoušena varianta bez vyhlazování. Příklad konkrétního nastavení rozsahu parametrů lze nalézt v tabulce 5.3. Soupis všech rozsahů parametrů uvedeného ke každé vadě lze nalézt v příloze B.

Názvy parametrů v tabulce 5.3 odpovídají názvům funkčních parametrů knihovny scikit-image. Pro značení rozsahů je použita notace podobná funkcím z knihovny numpy. Místo desetinné čárky je použita desetinná tečka, protože čárka je užita pro oddělení jednotlivých čísel. `linspace(start, stop, num)` značí rovnoměrně rozprostřených `num` čísel v rozsahu od `start`

po `stop`. `logspace(start, stop, num)` označuje rovnoměrně rozprostřených `num` čísel na logaritmické stupnici. Použit je logaritmus o základu deset. Rozsah čísel je tedy od $10^{\texttt{start}}$ do $10^{\texttt{stop}}$. Hranaté závorky `[]` značí výčet čísel.

Algoritmus quickshift v implementaci knihovny scikit-image pracuje pouze s barevnými obrázky. Proto je vždy otestována kombinace barevných prostorů RGB, LAB a HSV. Na rozdíl od ostatních algoritmů nemůže být učiněn výběr konkrétního kanálu.

Algoritmus level sets v implementaci knihovny scikit-image naopak pracuje pouze s jednokanálovými obrázky. Proto je vstupní obrázek vždy převeden do šedotónového prostoru nebo je použita jedna ze složek RGB, LAB nebo HSV.

Tabulka 5.3 Nastavení parametrů pro testování vady: viditelné defekty — závislé na vzoru (chyba barvy).

Algoritmus	**Rozsahy parametrů**
Snakes	`alpha: linspace(0.7, 1.3, num=5)` `w_edge: linspace(0.5, 2.5, num=5)`
Level sets	`mu: logspace(-2, 1, num=4)`
Random walker	`beta: logspace(1, 2.5, num=20)`
Felzenszwalb + RAG	`scale: logspace(0, 2, num=3)` `min_size: linspace(20, 120, num=4)`
SLIC + RAG	`n_segments: linspace(100, 200, num=2)` `compactess: logspace(-4, 0, num=4)`

Quickshift + RAG	`max_dist: linspace(5, 40, num=4)`
	`ratio: linspace(0.8, 2.5, num=3)`
	`kernel_size: [3, 5, 7]`

Barevné prostory: RGB, LAB (složka B), HSV (složka H a S)	

5.3.2 Inicializace počátečních podmínek

Algoritmy snakes, level sets a random walker na začátku segmentace vyžadují dodání počátečních podmínek. Ty jsou nastaveny následujícím způsobem. K vadě je pomocí ground truth nalezen minimální ohraničující obdélník. Nechť (w, h) je šířka a výška obdélníku v pixelech. Počáteční konturu algoritmu snake tvoří elipsa sdílející střed obdélníku o poloosách s velikostmi $(w/2 + 25, h/2 + 25)$.

Random walker taktéž využívá ohraničující obdélník. Jako pozadí je označeno vše, co je vně elipsy sdílející střed obdélníku s poloosami velikosti $(w/2+25, h/2+25)$. Jako popředí je označen vnitřek elipsy sdílející střed obdélníku s poloosami velikosti $(w/2-50, h/2-50)$. Minimální délka poloosy je 5 pixelů.

Počáteční zero level set opět využívá ohraničující obdélník. Vnitřek kontury je tvořen vnitřkem elipsy sdílející střed obdélníku s poloosami velikosti $(w/2 - 100, h/2 - 100)$. Minimální délka poloosy je 5 pixelů. Za vnějšek kontury jsou označeny ostatní pixely.

Kapitola 6
Výsledky

Systém zpracování dat byl spuštěn a byly ohodnocena jeho efektivita. Dále byl proveden test generalizace na každé vybrané kategorii. Ke každé testované kategorii je dodána tabulka obsahující testované rozsahy parametrů, barevné prostory a preprocessing. Tabulky lze najít v příloze B.

Seznam nalezených parametrů je z důvodu přehlednosti pouze u defektů kategorie *viditelné defekty — závislé na vzoru (chyba barvy)*. Úplný přehled dokumentující nalezené parametry ostatních kategorií lze nalézt v příloze C.

6.1 Barevné značení v obrázcích předvádějící výslednou segmentaci

Výsledky segmentace algoritmů jsou zobrazovány průhlednou vrstvou nad zdrojovým obrázkem. Zelená barva označuje true positive (TP), tedy pixel, který algoritmus označil správně jako závadný. Červená barva označuje false positive (FP), jedná se o pixel, který algoritmus označil jako závadný, ale ve skutečnosti se jedná o nezávadný pixel. Šedá barva označuje false negative (FN), což je pixel označen jako nezávadný, který je ve skutečnosti závadný. Absence barvy indikuje true negative (TN), tedy pixel, který je správně označen jako nezávadný.

6.2 Viditelné defekty

V kategorii viditelné defektů byly otestovány tři podkategorie: chyba závislá na vzoru (chyba barvy), chyba tvaru čáry a chyba tvaru bodu/kapky.

6.2.1 Závislé na vzoru (chyba barvy)

Nalezení nejlepší kombinace parametrů bylo provedeno na vadě `tile/oil`. Tabulka 6.1 obsahuje seznam nalezených parametrů každého algoritmu. Vada zabírá na většině obrázků v datasetu velkou část obrázku a je dostatečně barevně odlišená. Oproti předchozím vadám má proto většina algoritmů relativně vysokou úspěšnost.

Dobře zafungovaly superpixel algoritmy. Úspěch by se dal přisuzovat velikosti vady. Navíc superpixel algoritmy oproti ostatním algoritmům nedávají takový důraz na oddělení vady hranou. Algoritmus felzenszwalb spolu s RAG dokázal vadu vysegmentovat s průměrným IOU až 89 % viz obrázek 6.3a.

Algoritmus random walker dává moc velký důraz na prostorovost. Častý problém segmentace způsobuje vada, která je v rohu. Díky nejasnému konci vady (a tedy chybějící oddělující hraně) vysegmentuje algoritmus celý roh viz obrázek 6.3b.

Otsu segmentace je v některých případech úspěšná (IOU 88 %), algoritmus ale nezpracovává prostorové informace. Proto jsou v typickém případě kromě vady vysegmentovány i světlejší oblasti obrázku (v barevném prostoru HSV) viz obrázek 6.3c. Relativně vysoké průměrné IOU 46 % je způsobeno velikostí vady.

Výsledek segmentace level sets připomíná segmentaci Otsu. Díky zahrnuté prostorové informaci je zde nesprávně vysegmentovaných světlých oblastí minimum. Oproti malé efektivitě poskytnuté algoritmem na předchozích vadách zafungovaly zde level sets dobře (průměrné IOU 76 %), protože vada je velká a barevně oddělená od pozadí. Nejasné hrany algoritmu nevadí (viz obrázek 6.3d).

Algoritmus snakes v tomto případě nevyniká, segmentace trpí na nejasné hrany.

Test generalizace byl proveden na vadě `wood/color`. Průměrné IOU kombinace algoritmů felzenszwalb + RAG je 27 %. Segmentace přiléhá na konturu vady, RAGs ale nejsou schopny kvůli povaze spojování sousedních regionů vysegmentovat více vad v jednom obrázku (viz obrázek 6.6a). Pro úspěšnou segmentaci by bylo třeba upravit spojovací politiku.

Level sets provedly segmentaci s úspěšností IOU 43 %. Velkou část práce ale provede za algoritmus počáteční inicializace. U malých vad se dokázala expanze kontury zastavit a tak inicializaci nezkazit. Naopak u větších vad dokázal algoritmus vysegmentovat vadu celou. Kvůli postupnému jasovému průběhu a nejasnosti začátku a konce vady v některých případech algoritmus vysegmentoval konturu, která by se dala považovat za alternativní té, která je označená jako ground truth (viz obrázek 6.6b).

6.2.2 Tvar čáry

K nalezení kombinace parametrů byla použita vada `hazelnut/cut`. Nejúspěšněji vadu vysegmentoval algoritmus random walker s průměrným IOU 65 % viz obrázek 6.1a. Problémy způsobuje nedostatečně výrazná hrana mezi vadou a zbytkem objektu. Pokud je více vad blízko u sebe, algoritmus je spojuje do jediné vady viz obrázek 6.1b.

Úspěšnou segmentaci poskytuje také algoritmus snakes s průměrným IOU 51 %. Neúspěšná segmentace je způsobená kolapsem kontury do jednoho bodu kvůli nedostatečně výrazným oddělujícím hranám viz obrázek 6.1c. Někde se naopak snake zachycuje o falešnou hranu viz obrázek 6.1d. Algoritmus level set by byl vhodný pro segmentaci celého ořechu, funkce pro evoluci kontury není pro segmentaci škrábance vhodně zvolená. S nedostatečně výraznou hranou měly problém i superpixel algoritmy v kombinaci s RAGs, které místo vady oddělují ořech od pozadí. Mezi vadou a ořechem/pozadím není dostatečný kontrast na to, aby algoritmus Otsu uplatnil předpoklad bimodality histogramu.

Test generalizace byl proveden na vadě `metal_nut/scratch` s algoritmy random walker a snakes. Průměrné IOU algoritmu random walker je 45 %, průměrné IOU pro snakes je 34 %. Relativní úspěch je ale způsoben inicializačními podmínkami, výsledná segmentace (viz obrázek 6.4) nijak nekopíruje hranici vady v obrázku. Proto je výsledek testu považován za neúspěšný.

6.2.3 Tvar bodu/kapky

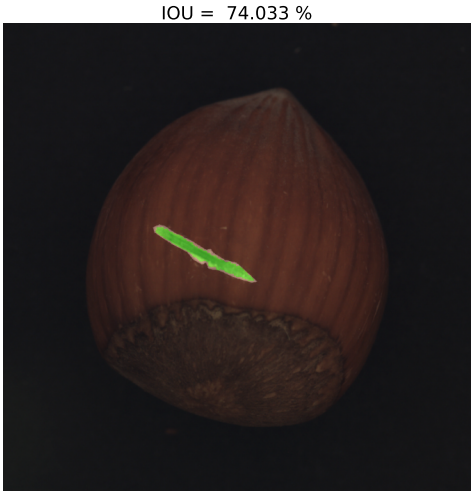
Pro nalezení nejlepší kombinace parametrů byla použita vada `tile/glue_strip`. Ačkoliv random walker prokazuje ze všech největší průměrné IOU 70 %, výsledná kontura kvůli nedostatečně výrazným hranám v některých částech na vadu nepřiléhá viz obrázek 6.2a. Úspěch měla kombinace algoritmů quickshift s RAGs s průměrným IOU 57 %. Algoritmus quickshift dokázal ve většině případů vadu lokalizovat a rozpoznat její hranu viz obrázek 6.2b. V některých případech ale segmentace kompletně selhala (IOU = 0 %) viz obrázek 6.2c. Level sets také dokázaly rozpoznat

Tabulka 6.1 Nalezená kombinace parametrů a průměrné IOU pro kategorii vady: viditelné defekty — závislé na vzoru (chyba barvy).

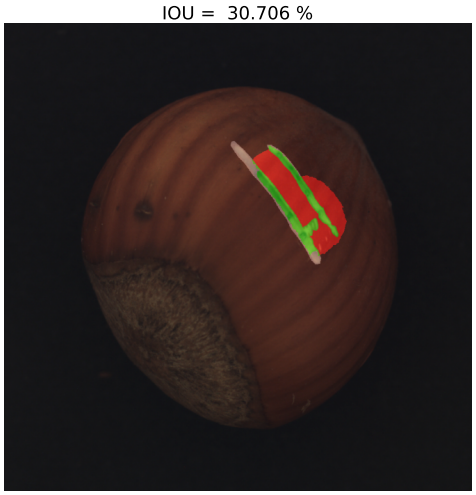
Algoritmus	Nalezené parametry	Průměrné IOU
Snakes	`alpha`: 0.85 `w_edge`: 0.5 Preprocessing: Gaussian ($\sigma = 1$) Barevný prostor: LAB (složka B)	77.39 %
Level sets	`mu`: 0.01 Preprocessing: Gaussian ($\sigma = 3$) Barevný prostor: HSV (složka H)	76.21 %
Random walker	`beta`: 220 Preprocessing: žádný Barevný prostor: HSV (složka H)	68.43 %
Felzenszwalb + RAG	`scale`: 100 `min_size`: 20 Preprocessing: Gaussian ($\sigma = 2$) Barevný prostor: HSV (složka S)	89.76 %
SLIC + RAG	`n_segments`: 200 `compactness`: 0.046415 Preprocessing: Gaussian ($\sigma = 1$) Barevný prostor: HSV (složka S)	73.74 %
Quickshift + RAG	`max_dist`: 5 `ratio`: 1.65 `kernel_size`: 3 Preprocessing: Gaussian ($\sigma = 3$) Barevný prostor: HSV	80.72 %
Otsu	Preprocessing: Gaussian ($\sigma = 3$) Barevný prostor: HSV (složka H)	46.03 %

hranu vady, ale díky relativně stejné barvě pozadí přibírá výsledná segmentace ostatní objekty v obrázku (viz obrázek 6.2d). Segmentace ostatních algoritmů se ukázala jako nedostatečná.

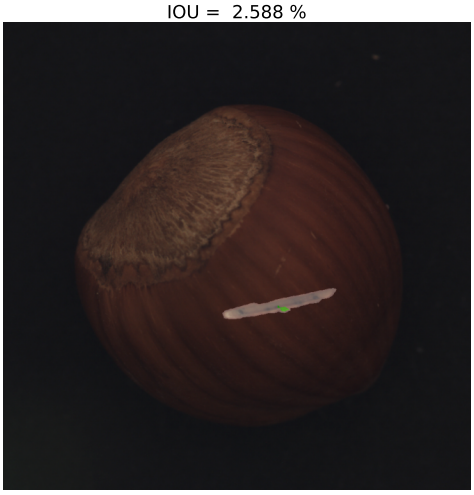
Test generalizace byl proveden na vadě `metal_nut/bent`. Kvůli vlastnostem algoritmu segmentoval RAG střed matky. To je způsobeno malou velikosti tmavého středu a ostrou hranou oddělující střed od oceli (viz obrázek 6.5a). Proto je průměrné IOU algoritmů quickshift + RAG pouze 0,1 %. Segmentace algoritmem random walker má průměrné IOU 53 %. Relativní úspěch segmentace byl primárně způsoben zvolenými počátečními podmínkami (viz obrázek 6.5b).



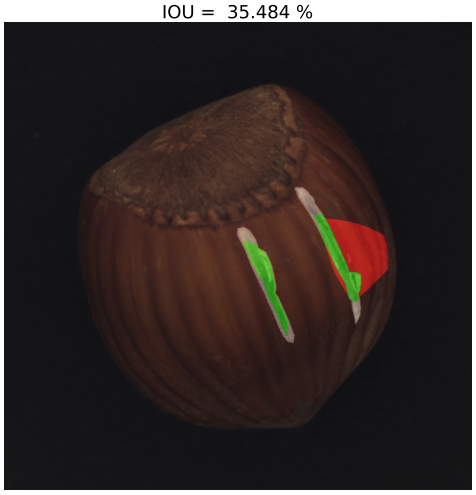
(a) Algoritmus random walker. Ukázka úspěšné segmentace.



(b) Algoritmus random walker. Dva defekty blízko sebe jsou spojeny do jednoho.

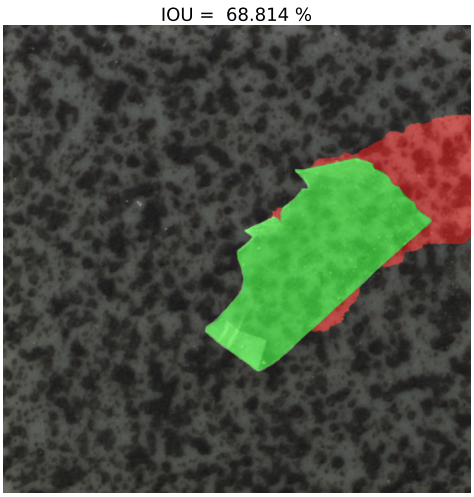


(c) Algoritmus snakes. Kontura se zbortila do jednoho bodu.

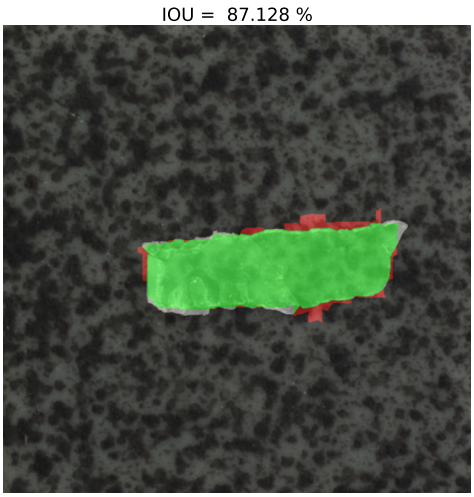


(d) Algoritmus snakes. Snake se zachytil o falešnou hranu.

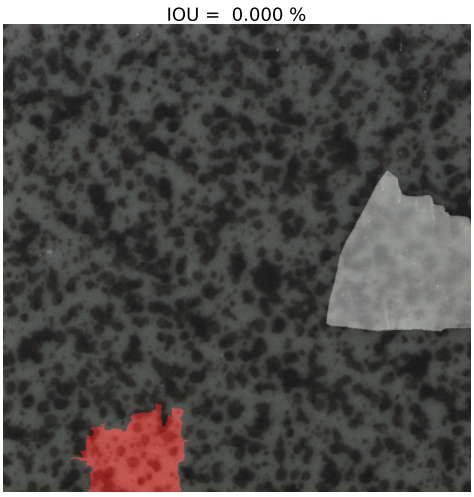
Obrázek 6.1 Segmentace vady `hazelnut/cut` reprezentující kategorii viditelné defekty — tvar čáry.



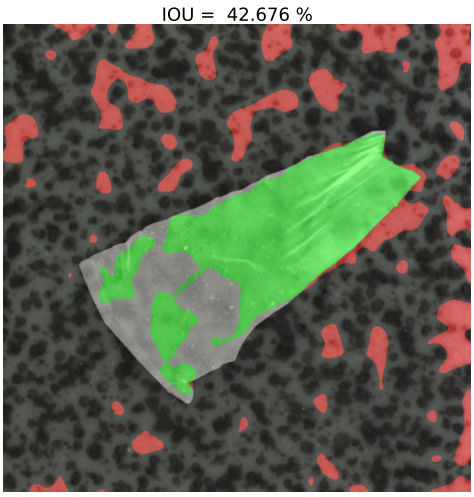
(a) Algoritmus random walker. Typický případ, kdy výsledná kontura na vadu nepříléhá.



(b) Algoritmy quickshift + RAG. Ukázka úspěšné segmentace.

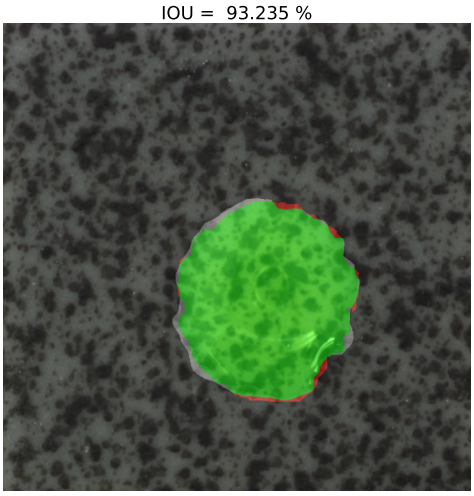


(c) Algoritmy quickshift + RAG. Výsledná segmentace defekt kompletně minula.

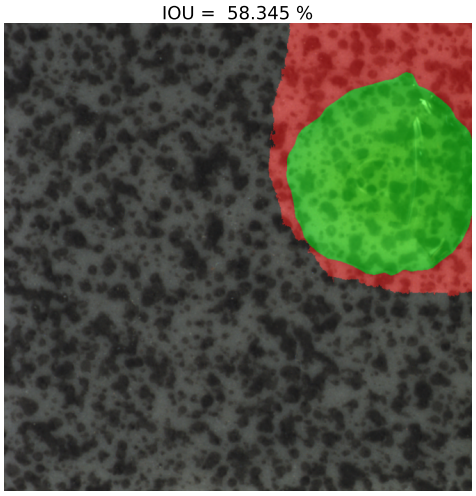


(d) Algoritmus level sets. Segmentace přibírá ostatní objekty v obrázku.

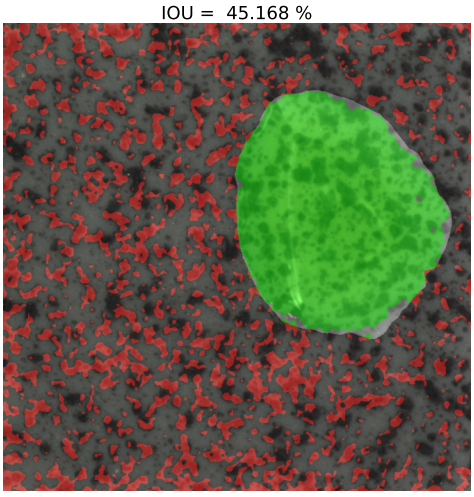
Obrázek 6.2 Segmentace vady `tile/glue_strip` reprezentující kategorii viditelné defekty — tvar bodu/kapky.



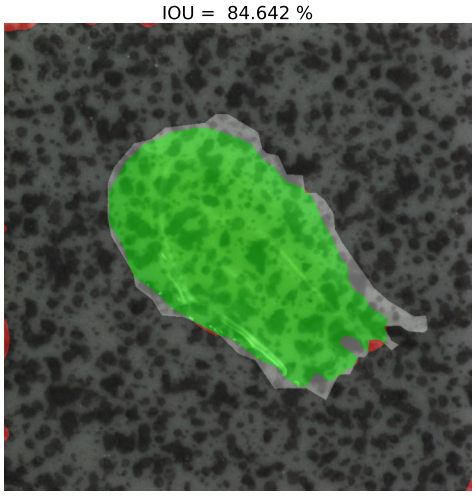
(a) Kombinace algoritmů felzenszwalb + RAG dokázala vadu vysegmentovat s největší úspěšností.



(b) Algoritmus random walker při umístění vady v rohu segmentuje celý roh.

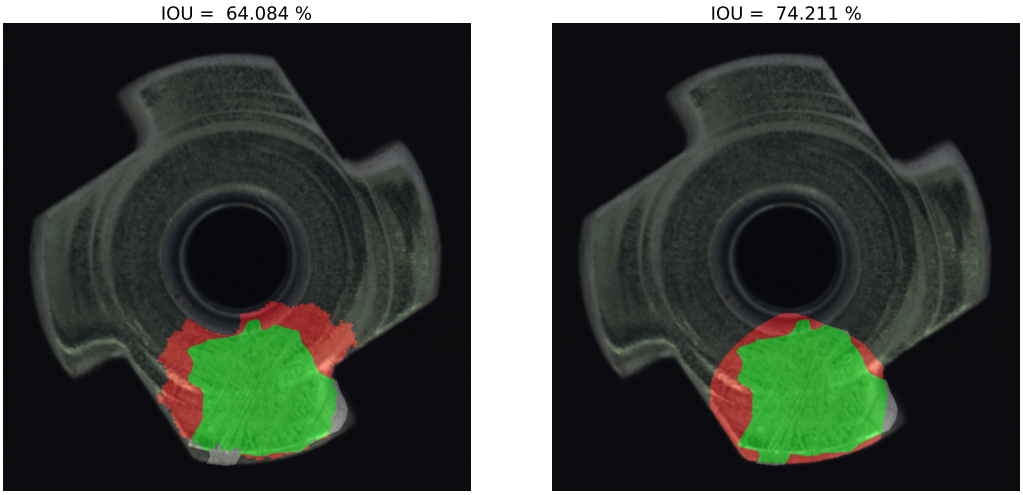


(c) Typický výsledek segmentace pomocí algoritmu Otsu.



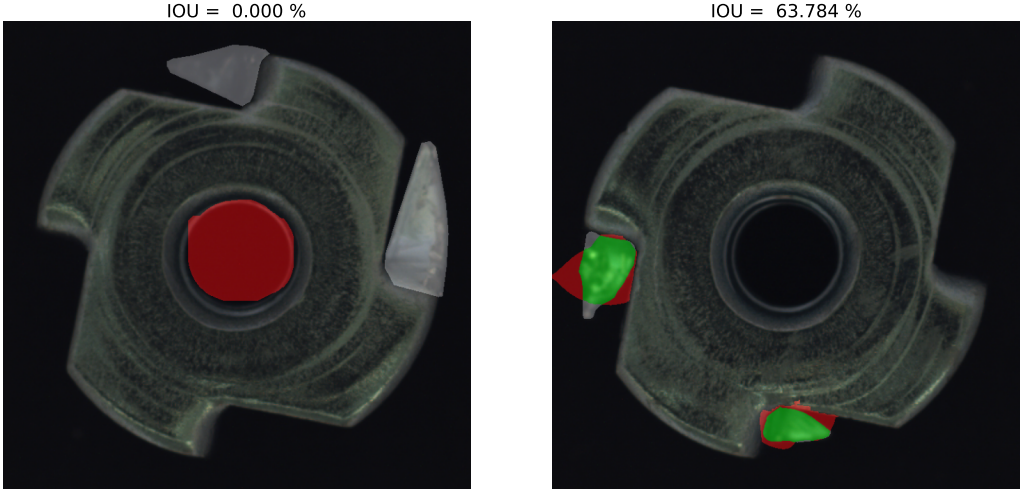
(d) Typický výsledek segmentace pomocí algoritmu level sets. Narozdíl od algoritmu Otsu segmentace nepříbírá světlejší oblasti obrázku.

Obrázek 6.3 Segmentace vady `hazelnut/cut` reprezentující kategorii viditelné defekty — závislé na vzoru (chyba barvy).



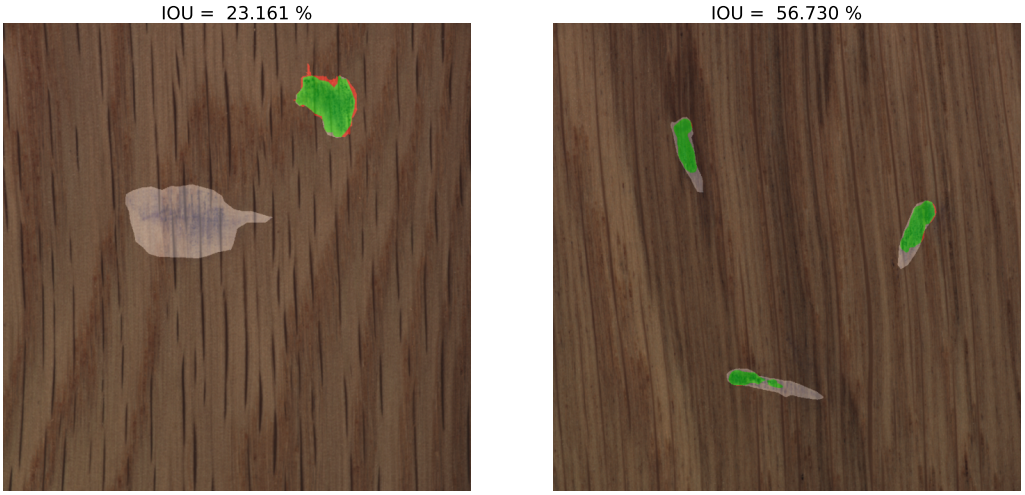
(a) Algoritmus random walker. **(b)** Algoritmus snakes.

Obrázek 6.4 Test generalizace na vadě `metal_nut/scratch`. Výsledná segmentace nekopíruje konturu vady, vysoké IOU je způsobeno inicializačními podmínkami.



(a) Algoritmy quickshift + RAG. Kvůli vlastnostem spojování RAG je vysegmentován střed matice. **(b)** Algoritmus random walker. Relativně vysoké IOU je způsobeno inicializačními podmínkami.

Obrázek 6.5 Test generalizace na vadě `metal_nut/bent`.



(a) Segmentace algoritmy felzenszwalb + RAG. Kvůli spojovácímu kritériu RAGs nedokáže algoritmus vysegmentovat vícero vad současně.

(b) Typická segmentace level sets. Začátek a konec vady je v tomto případě nejasný.

Obrázek 6.6 Test generalizace na vadě `wood/color`.

6.3 Hmatatelné defekty

V kategorii hmatatelné defektů byly otestovány dvě podkategorie: chyba tvaru čáry a chyba tvaru bodu/kapky.

6.3.1 Tvar čáry

Nalezení nejlepší kombinace parametrů bylo provedeno na vadě `leather/cut`. Kvůli malým rozměrům vady měly snakes tendenci ke kolapsu. Taktéž level setům činily rozměry vady problém. Na rozdíl od snakes měly tendenci do segmentace zahrnovat pozadí. I random walker měl problém s velikostí defektu. Algoritmus měl tendenci dělat defekty kulatější, než ve skutečnosti jsou, což bylo nejspíš způsobeno elipsovitou inicializací (viz obrázek 6.7a). Nejlepší ze všech se ukázala kombinace algoritmů felzenszwalb + RAG s průměrným IOU 39 %. Algoritmy dokázaly vadu lokalizovat, výsledná segmentace ale někdy nezabírá celou vadu viz obrázek 6.7b, což je způsobeno tím, že průměrná barva na hranici vady je více podobná pozadí než prostředku vady. Segmentace ostatních algoritmů je nedostatečná a neprokazuje zajímavé vlastnosti.

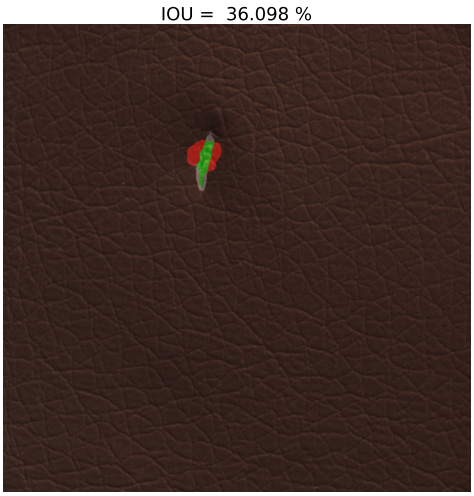
Generalizace kombinace algoritmů felzenszwalb + RAG byla testována na vadě `tile/crack`. Díky kompletně odlišnému tvaru defektu segmentace selhala s průměrným IOU 13 %. Objevují se ale náznaky úspěšné segmentace. Algoritmus má tendenci v obrázku vybírat podlouhlé oblasti souvislé barvy viz obrázek 6.9.

6.3.2 Tvar bodu/kapky

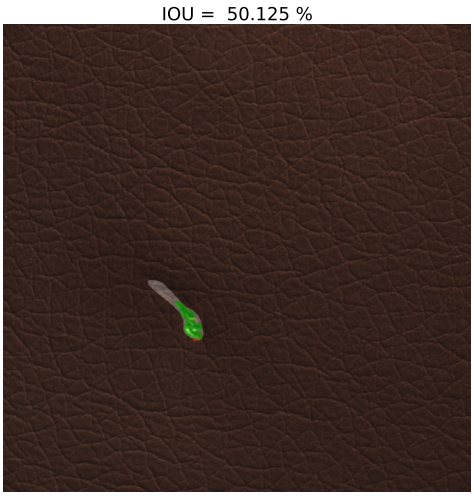
Počáteční nalezení parametrů bylo provedeno na vadě `leather/glue`. Díky vrásčitému povrchu kůže měly algoritmy problém s odlišením falešné hrany od vady hrany. Snakes se o tyto falešné hrany zachycovaly. Stejný problém nastal u random walker, ovšem v omezené míře (viz obrázek 6.8a). Ze superpixel algoritmů se nejlepší ukázal felzenszwalb, který *na všech snímcích* dokázal správně defekt lokalizovat. Některé obrázky obsahují více defektů, ale kvůli modus operandi RAG dokáže vysegmentovat pouze jednu vadu (viz obrázek 6.8b). Segmentace level sets

také dobře kopíruje konturu vady, avšak inicializace algoritmu hodně pomohla. K testu generalizace byl vybrán algoritmus random walker s největším průměrným IOU 66 % a kombinace felzenszwalb + RAG s průměrným IOU 64 %.

Generalizace byla provedena na vadě `wood/liquid`. Oproti dřevu není tekutina nikterak kontrastní. I přes to dokázal random walker tekutinu vysegmentovat s průměrným IOU 58 %. Má ale tendenci dělat defekty kulatější, než ve skutečnosti jsou (viz obrázek 6.10a). Na nedostatečný kontrast vady utrpěla segmentace algoritmy felzenszwalb + RAG s průměrným IOU 0,6 %, která na většině snímků nebyla schopna ani vadu lokalizovat (viz obrázek 6.10b).

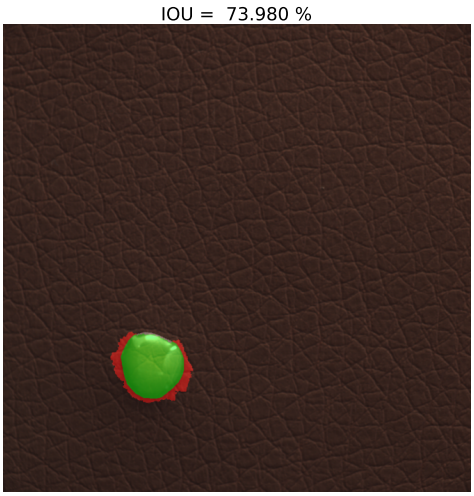


(a) Výsledek segmentace random walker je kulatější, než samotná vada, protože inicializace je elipsovitá.

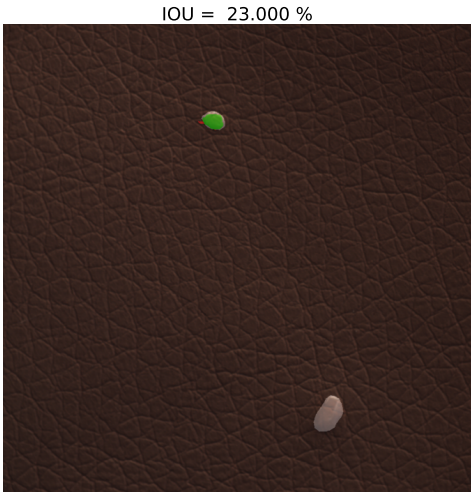


(b) Algoritmy felzenszwalb + RAG někdy nezabere celou vadu.

Obrázek 6.7 Segmentace vady `leather/cut` reprezentující kategorii hmatatelné defekty — tvar čáry.

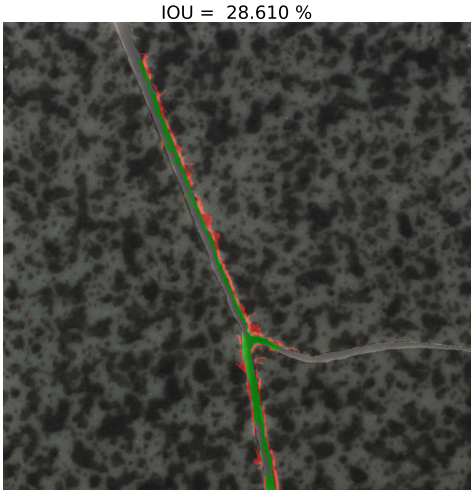


(a) Random walker se zachycuje o falešnou hranu díky vrásčitosti kůže.

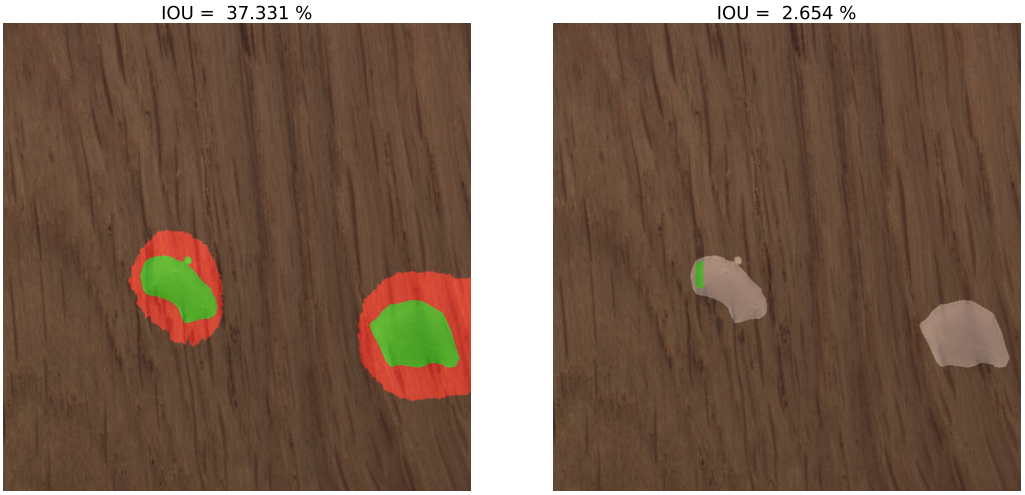


(b) Algoritmy felzenszwalb + RAG dokážou vysegmentovat pouze jednu vadu.

Obrázek 6.8 Segmentace vady `leather/glue` reprezentující kategorii hmatatelné defekty — tvar bodu/kapky.



Obrázek 6.9 Test generalizace na vadě `tile/crack`.



(a) Random walker má tendenci dělat defekty kulatější, něž ve skutečnosti jsou.

(b) Segmentace pomocí felzenszwalb + RAG selhala kvůli nedostatečnému kontrastu mezi vadou a pozadím.

Obrázek 6.10 Test generalizace na vadě `wood/liquid`.

Kapitola 7
Diskuze

Typické problémy algoritmům způsoboval malý kontrast mezi vadou a pozadím, nejasné či falešné hrany, podobná barva vady a zbytku obrázku nebo malá velikost vady. U snakes to vedlo ke zborcení kontury do jednoho bodu. Naopak level sets začaly přibírat k vadě kousky pozadí. Algoritmus random walker kvůli nejasné hraně vracel segmentaci, která oproti skutečnosti byla kulatější. Superpixely a RAGs měly problém s lokalizací defektu. Z těchto důvodů nebylo pro některé vady možné najít parametry, pomocí kterých by algoritmy vadu vysegmentovaly s chtěnou přesností.

U algoritmu random walker by problém s kulatými defekty mohlo vyřešit přenastavení koeficientu β určující váhu hran. Kriteriální funkce snakes a level sets také nebyla optimální. Zvláště u level sets kriteriální funkce nebere ohled na drobné detaily v obrázku, které jsou pro segmentaci defektů malé velikosti kritické. Jistě by bylo možno experimentovat s různými poddruhy algoritmů vybavenými jinou kriteriální funkcí.

Úspěšnost algoritmů v testu generalizace byla také smíšená. Největším faktorem určující úspěšnost byla vizuální podobnost testované vady a textury s trénovacími podmínkami. Příkladem neúspěšné generalizace je vada kategorie viditelné defekty — tvar čáry. Škrábanec na ořechu, který tvoří souvislý vryp je naprosto odlišný od škrábanců na matce, které jsou tvořeny množstvím povrchových čar. Plocha škrábance na matce je minimální. Podobně dopadl test generalizace vad kategorie viditelné defekty — tvar bodu/kapky. Ačkoliv se v případě i trénovací i testovací vady jedná o kontaminaci objektem barvy podobné textuře, v trénovacím případě se jedná o velký cizí objekt, který je položen na povrchu textury, v testovacím případě špatný záhyb kovu matky způsobil propojení vady s bokem matky.

Nejúspěšnější generalizace proběhla u vad kategorie viditelné defekty — závislé na vzoru. Ukazuje se, že barevná odlišnost vady poskytuje algoritmům informace potřebné k úspěšné segmentaci. Poměrně úspěšná generalizace nastala také u hmatatelných defektů. Parametry algoritmů segmentující generalizační vadu by bylo možné dále zlepšovat. Navržená kombinace ale může posloužit jako počáteční bod pro následné ladění. Velkou roli hrály počáteční podmínky, které algoritmům pomáhaly. Při reálném nasazení by bylo potřeba navrhnout systém, který tyto podmínky stanovuje.

V praxi je možno výrazným způsobem vylepšit účinnost segmentačních algoritmů vhodnou volbou kamery, objektivu či vhodným nasvícením objektu. Příkladem konkrétních technik může být nasvícení objektu z temného pole, nasvícení objektu pomocí koaxiálních světel, použití jiných úhlu pro zachycení objektu nebo využití hyperspektrálních kamer. Způsob pořízení obrazu v práci nebyl řešen, proto by mohlo být zajímavé do uvažovaných proměnných ovlivňující kvalitu výsledku segmentace zakomponovat tyto techniky.

Množina zkoumaných algoritmů by mohla být rozšířena o další tradiční metody. Pro segmentaci vad by mohly lepší výsledky podávat algoritmy pro detekci anomálií využívající autokore-

lace nebo Fourierovy transformace. Další výzkum nemusí být omezen pouze na tradiční metody, k segmentaci by mohly být použity neuronové sítě či hybridní přístup kombinující neuronové sítě s tradičními technikami.

Z úvah vyplývá, že ve většině případech je potřeba primárně respektovat vlastnosti algoritmů. Již při výběru algoritmu je do způsobu zpracování vnášena apriorní znalost. Mohla by být vytvořena kategorizace defektů, která se nesnaží defekty rozdělit na základě jejich charakteristických znaků, ale na základě vlastností charakterizující algoritmy. Správný výběr algoritmu je kritický pro úspěch při řešení daného problému, na výběr ale v této chvíli neexistuje univerzální recept.

Kapitola 8
Závěr

V práci byly zmapovány typické defekty objevující se v průmyslu, pro které bylo navržené obecnější dělení. Pro účely testování algoritmů byl vybrán dataset obsahující obrázky s průmyslovými defekty, jehož vady byly roztříděny dle navrženého dělení. Byl navržen systém zpracování obrazových dat. Ke každé kategorii byla pro systém zpracování dat nalezena nejlepší možná kombinace parametrů. Celkově bylo na pěti kategoriích odzkoušeno osm vybraných algoritmů s různým nastavením a byla zhodnocena jejich úspěšnost. Při hledání parametrů systému zpracování bylo dohlíženo i na různé druhy předzpracování obrázku. Dále byl proveden test generalizace, díky němuž bylo možno získat odhad vhodnosti algoritmů pro segmentaci vad dané kategorie. Výsledky segmentace byly vizualizovány.

Příloha A

Rozdělení vad datasetu MVTec podle navržené kategorizace

Tabulka A.1 Vady datasetu MVTec rozděleny do kategorií podle navrženého dělení.

Kategorie vady			**Vady datasetu**

Viditelné defekty	Tvar čáry	Škrábanec	`capsule/scratch`
			`hazelnut/cut`
			`pill/scratch`

		Škrábance	`metal_nut/scratch`
			`screw/scratch_head`
			`screw/scratch_neck`
			`wood/scratch`

	Tvar bodu/kapky	Kontaminace	`bottle/contamination`
			`pill/contamination`
			`tile/glue_strip`
			`metal_nut/bent`

		Dření	`tile/rough` `zipper/rough`

	Závislé na vzoru	Chyba tvaru	`bottle/broken_large`
			`bottle/broken_small`
			`carpet/cut`
			`carpet/hole`
			`carpet/metal_contamination`
			`carpet/thread`
			`grid/bent`
			`grid/broken`
			`grid/metal_contamination`
			`grid/thread`

Kategorie vady			Vady datasetu
Viditelné defekty	Závislé na vzoru	Chyba tvaru	zipper/broken_teeth
			zipper/fabric_border
			zipper/fabric_interior
			zipper/split_teeth
			zipper/squeezed_teeth
			metal_nut/bent
			screw/manipulated_front
			screw/thread_side
			screw/thread_top
		Chyba barvy	carpet/color
			hazelnut/print
			leather/color
			metal_nut/color
			pill/color
			pill/pill_type
			tile/gray_stroke
			tile/oil
			wood/color
			zipper/rough
			cable/missing_cable
			cable/cable_swap
			cable/missing_wire
	Konkrétní tvar	Chyba pozice	metal_nut/flip
			transistor/misplaced
		OCR	pill/faulty_imprint
			capsule/faulty_imprint
Hmatatelné defekty	Tvar čáry	Záhyb	leather/fold
		Prasklina/roztržení	capsule/crack
			leather/cut
			pill/crack
			tile/crack
			cable/cut_inner_insulation
			cable/cut_outer_insulation
			hazelnut/cut
	Tvar bodu/kapky	Důlek	capsule/poke
			hazelnut/crack
			leather/poke
			cable/poke_insulation
		Díra	hazelnut/hole
			wood/hole
		Tekutina	grid/glue

Kategorie vady			Vady datasetu

Hmatatelné defekty	Tvar bodu/kapky	Tekutina	`leather/glue`
			`wood/liquid`
			`tile/oil`

	Konkrétní tvar	Chyba tvaru	`capsule/squeeze`
			`transistor/damage_case`
			`transistor/cut_lead`
			`transistor/bent_lead`
			`toothbrush/defective`
			`screw/manipulated_front`
			`screw/thread_side`
			`screw/thread_top`
			`cable/bent_wire`

Příloha B

Testované rozsahy parametrů

Tabulka B.1 Nastavení parametrů pro testování vady: viditelné defekty — tvar čáry.

Algoritmus	**Rozsahy parametrů**
Snakes	`alpha: linspace(0.7, 1.3, num=5)` `w_edge: linspace(0.5, 2.5, num=5)`
Level sets	`mu: logspace(-2, 1, num=4)`
Random walker	`beta: logspace(1, 2.5, num=20)`
Felzenszwalb + RAG	`scale: logspace(0, 2, num=3)` `min_size: linspace(20, 100, num=2)`
SLIC + RAG	`n_segments: linspace(100, 200, num=2)` `compactess: logspace(-4, 0, num=4)`

Quickshift + RAG	`max_dist: linspace(5, 40, num=4)`
	`ratio: linspace(0.8, 2.5, num=3)`
	`kernel_size: [3, 5, 7]`

Barevné prostory: RGB, LAB (složka A), HSV (složka S)	

Tabulka B.2 Nastavení parametrů pro testování vady: viditelné defekty — tvar bodu/kapky.

Algoritmus	**Rozsahy parametrů**
Snakes	`alpha: linspace(0.7, 1.5, num=5)` `w_edge: linspace(0.2, 1.5, num=5)`
Level sets	`mu: logspace(-2, 1, num=4)`
Random walker	`beta: logspace(1, 2.5, num=20)`
Felzenszwalb + RAG	`scale: logspace(0, 2, num=3)` `min_size: linspace(20, 120, num=4)`
SLIC + RAG	`n_segments: linspace(100, 200, num=2)` `compactess: logspace(-4, 0, num=4)`

Quickshift + RAG	`max_dist: linspace(5, 40, num=4)`
	`ratio: linspace(0.8, 2.5, num=3)`
	`kernel_size: [3, 5, 7]`

Barevné prostory: grayscale	

Tabulka B.3 Nastavení parametrů pro testování vady: viditelné defekty — závislé na vzoru (chyba barvy).

Algoritmus	**Rozsahy parametrů**
Snakes	`alpha: linspace(0.7, 1.3, num=5)` `w_edge: linspace(0.5, 2.5, num=5)`
Level sets	`mu: logspace(-2, 1, num=4)`
Random walker	`beta: logspace(1, 2.5, num=20)`
Felzenszwalb + RAG	`scale: logspace(0, 2, num=3)` `min_size: linspace(20, 120, num=4)`
SLIC + RAG	`n_segments: linspace(100, 200, num=2)` `compactess: logspace(-4, 0, num=4)`

Quickshift + RAG	`max_dist: linspace(5, 40, num=4)`
	`ratio: linspace(0.8, 2.5, num=3)`
	`kernel_size: [3, 5, 7]`

Barevné prostory: RGB, LAB (složka B), HSV (složka H a S)	

Tabulka B.4 Nastavení parametrů pro testování vady: hmatatelné defekty — tvar čáry.

Algoritmus	**Rozsahy parametrů**
Snakes	`alpha: linspace(0.3, 1.3, num=5)` `w_edge: linspace(0.5, 2.5, num=5)`
Level sets	`mu: logspace(-2, 1, num=4)`
Random walker	`beta: logspace(1, 2.5, num=20)`
Felzenszwalb + RAG	`scale: logspace(0, 2, num=3)` `min_size: linspace(20, 120, num=4)`
SLIC + RAG	`n_segments: linspace(100, 200, num=2)` `compactess: logspace(-4, 0, num=5)`

Quickshift + RAG	`max_dist: linspace(5, 40, num=4)`
	`ratio: linspace(0.8, 2.5, num=3)`
	`kernel_size: [3, 5, 7]`

Barevné prostory: RGB, LAB (složka L), HSV (složka S)	

Tabulka B.5 Nastavení parametrů pro testování vady: hmatatelné defekty — tvar body/kapky.

Algoritmus	**Rozsahy parametrů**
Snakes	`alpha: linspace(0.2, 1.3, num=5)` `w_edge: linspace(0.5, 2.5, num=5)`
Level sets	`mu: logspace(-2, 1, num=4)`
Random walker	`beta: logspace(1, 2.5, num=20)`
Felzenszwalb + RAG	`scale: logspace(0, 2, num=4)` `min_size: linspace(20, 120, num=4)`
SLIC + RAG	`n_segments: linspace(100, 200, num=2)` `compactess: logspace(-4, 0, num=4)`

Quickshift + RAG	`max_dist: linspace(5, 40, num=4)`
	`ratio: linspace(0.8, 2.5, num=3)`
	`kernel_size: [3, 5, 7]`

Barevné prostory: RGB, LAB (složka A a B), HSV (složka S)	

Příloha C

Nalezené kombinace parametrů spolu s průměrným IOU

Tabulka C.1 Nalezená kombinace parametrů a průměrné IOU pro kategorii vady: viditelné defekty — tvar čáry.

Algoritmus	**Nalezené parametry**	**Průměrné IOU**
Snakes	`alpha`: 0.85 `w_edge`: 1.5 Preprocessing: Gaussian ($\sigma = 1$) Barevný prostor: RGB	51.12 %
Level sets	`mu`: 0.1 Preprocessing: Gaussian ($\sigma = 1$) Barevný prostor: grayscale	4.06 %
Random walker	`beta`: 316 Preprocessing: Gaussian ($\sigma = 1$) Barevný prostor: LAB (složka A)	65.27 %
Felzenszwalb + RAG	`scale`: 1 `min_size`: 20 Preprocessing: Gaussian ($\sigma = 1$) Barevný prostor: RGB	12.49 %
SLIC + RAG	`n_segments`: 200 `compactness`: 0.046415 Preprocessing: Gaussian ($\sigma = 2$) Barevný prostor: RGB	5.92 %
Quickshift + RAG	`max_dist`: 16.66 `ratio`: 2.5 `kernel_size`: 5 Preprocessing: žádný Barevný prostor: LAB	9.39 %
Otsu	Preprocessing: žádný Barevný prostor: LAB (složka A)	0.31 %

Tabulka C.2 Nalezená kombinace parametrů a průměrné IOU pro kategorii vady: viditelné defekty — tvar bodu/kapky.

Algoritmus	**Nalezené parametry**	**Průměrné IOU**
Snakes	`alpha`: 1.1 `w_edge`: 1.5 Preprocessing: žádný Barevný prostor: grayscale	45.65 %
Level sets	`mu`: 0.1 Preprocessing: Gaussian ($\sigma = 3$) Barevný prostor: grayscale	32.68 %
Random walker	`beta`: 316.66 Preprocessing: žádný Barevný prostor: grayscale	70.18 %
Felzenszwalb + RAG	`scale`: 10 `min_size`: 120 Preprocessing: Gaussian ($\sigma = 2$) Barevný prostor: grayscale	22.63 %
SLIC + RAG	`n_segments`: 100 `compactness`: 1 Preprocessing: Gaussian ($\sigma = 3$) Barevný prostor: grayscale	36.59 %
Quickshift + RAG	`max_dist`: 40 `ratio`: 0.8 `kernel_size`: 7 Preprocessing: Gaussian ($\sigma = 3$) Barevný prostor: LAB	57.03 %
Otsu	Preprocessing: Gaussian ($\sigma = 3$) Barevný prostor: grayscale	23.34 %

Tabulka C.3 Nalezená kombinace parametrů a průměrné IOU pro kategorii vady: viditelné defekty — závislé na vzoru (chyba barvy).

Algoritmus	**Nalezené parametry**	**Průměrné IOU**
Snakes	`alpha`: 0.85 `w_edge`: 0.5 Preprocessing: Gaussian ($\sigma = 1$) Barevný prostor: LAB (složka B)	77.39 %
Level sets	`mu`: 0.01 Preprocessing: Gaussian ($\sigma = 3$) Barevný prostor: HSV (složka H)	76.21 %
Random walker	`beta`: 220 Preprocessing: žádný Barevný prostor: HSV (složka H)	68.43 %
Felzenszwalb + RAG	`scale`: 100 `min_size`: 20 Preprocessing: Gaussian ($\sigma = 2$) Barevný prostor: HSV (složka S)	89.76 %
SLIC + RAG	`n_segments`: 200 `compactness`: 0.046415 Preprocessing: Gaussian ($\sigma = 1$) Barevný prostor: HSV (složka S)	73.74 %
Quickshift + RAG	`max_dist`: 5 `ratio`: 1.65 `kernel_size`: 3 Preprocessing: Gaussian ($\sigma = 3$) Barevný prostor: HSV	80.72 %
Otsu	Preprocessing: Gaussian ($\sigma = 3$) Barevný prostor: HSV (složka H)	46.03 %

Tabulka C.4 Nalezená kombinace parametrů a průměrné IOU pro kategorii vady: hmatatelné defekty — tvar bodu.

Algoritmus	**Nalezené parametry**	**Průměrné IOU**
Snakes	`alpha`: 0.8 `w_edge`: 1.5 Preprocessing: Gaussian ($\sigma = 3$) Barevný prostor: LAB (složka L)	24.40 %
Level sets	`mu`: 0.01 Preprocessing: Gaussian ($\sigma = 2$) Barevný prostor: HSV (složka S)	14.86 %
Random walker	`beta`: 316.66 Preprocessing: Gaussian ($\sigma = 3$) Barevný prostor: LAB (složka L)	38.90 %
Felzenszwalb + RAG	`scale`: 100 `min_size`: 53 Preprocessing: Gaussian ($\sigma = 1$) Barevný prostor: HSV (složka S)	39.16 %
SLIC + RAG	`n_segments`: 200 `compactness`: 0.1 Preprocessing: Gaussian ($\sigma = 2$) Barevný prostor: HSV (složka S)	22.93 %
Quickshift + RAG	`max_dist`: 5 `ratio`: 2.5 `kernel_size`: 3 Preprocessing: Gaussian ($\sigma = 2$) Barevný prostor: LAB	12.72 %
Otsu	Preprocessing: Gaussian ($\sigma = 1$) Barevný prostor: HSV (složka S)	0.80 %

Tabulka C.5 Nalezená kombinace parametrů a průměrné IOU pro kategorii vady: hmatatelné defekty — tvar bodu/kapky.

Algoritmus	**Nalezené parametry**	**Průměrné IOU**
Snakes	`alpha`: 1.03 `w_edge`: 2.5 Preprocessing: žádný Barevný prostor: HSV (složka S)	39.50 %
Level sets	`mu`: 1 Preprocessing: Gaussian ($\sigma = 2$) Barevný prostor: HSV (složka S)	57.53 %
Random walker	`beta`: 316.66 Preprocessing: žádný Barevný prostor: HSV (složka S)	66.40 %
Felzenszwalb + RAG	`scale`: 100 `min_size`: 86 Preprocessing: Gaussian ($\sigma = 1$) Barevný prostor: HSV (složka S)	64.38 %
SLIC + RAG	`n_segments`: 200 `compactness`: 0.046415 Preprocessing: Gaussian ($\sigma = 2$) Barevný prostor: HSV (složka S)	46.39 %
Quickshift + RAG	`max_dist`: 5 `ratio`: 2.5 `kernel_size`: 3 Preprocessing: žádný Barevný prostor: RGB	31.59 %
Otsu	Preprocessing: Gaussian ($\sigma = 3$) Barevný prostor: grayscale	0.87 %

Bibliografie

1. BELTHANGADY, Chinmay; ROYER, Loic A. Applications, promises, and pitfalls of deep learning for fluorescence image reconstruction. *Nature Methods*. 2019, roč. 16. ISSN 15487105. Dostupné z DOI: 10.1038/s41592-019-0458-z.
2. O'MAHONY, Niall; CAMPBELL, Sean; CARVALHO, Anderson; HARAPANAHALLI, Suman; HERNANDEZ, Gustavo Velasco; KRPALKOVA, Lenka; RIORDAN, Daniel; WALSH, Joseph. Deep Learning vs. Traditional Computer Vision. In: 2020, sv. 943. ISSN 21945365. Dostupné z DOI: 10.1007/978-3-030-17795-9_10.
3. LEE, Lay Khoon; LIEW, Siau Chuin; THONG, Weng Jie. A review of image segmentation methodologies in medical image. In: 2015, sv. 315. ISSN 18761119. Dostupné z DOI: 10.1007/978-3-319-07674-4_99.
4. PAPARI, Giuseppe; PETKOV, Nicolai. *Edge and line oriented contour detection: State of the art*. Sv. 29. 2011. ISSN 02628856. Dostupné z DOI: 10.1016/j.imavis.2010.08.009.
5. MEISTER, Sebastian; WERMES, Mahdieu A.M.; STÜVE, Jan; GROVES, Roger M. *Review of image segmentation techniques for layup defect detection in the Automated Fiber Placement process: A comprehensive study to improve AFP inspection*. Sv. 32. 2021. ISSN 15728145. Dostupné z DOI: 10.1007/s10845-021-01774-3.
6. CIORA, Radu; SIMION, Carmen. Industrial Applications of Image Processing. *ACTA Universitatis Cibiniensis*. 2014, roč. 64. Dostupné z DOI: 10.2478/aucts-2014-0004.
7. SCHARR, Hanno; MINERVINI, Massimo; FRENCH, Andrew P.; KLUKAS, Christian; KRAMER, David M.; LIU, Xiaoming; LUENGO, Imanol; PAPE, Jean Michel; POLDER, Gerrit; VUKADINOVIC, Danijela; YIN, Xi; TSAFTARIS, Sotirios A. Leaf segmentation in plant phenotyping: a collation study. *Machine Vision and Applications*. 2016, roč. 27. ISSN 14321769. Dostupné z DOI: 10.1007/s00138-015-0737-3.
8. ISLAM, Mohammed J.; BASALAMAH, Saleh; AHMADI, Majid; SID-AHMED, Maher A. Capsule image segmentation in pharmaceutical applications using edge-based techniques. In: 2011. ISSN 21540357. Dostupné z DOI: 10.1109/EIT.2011.5978613.
9. KANG, S. P.; SABAREZ, H. T. Simple colour image segmentation of bicolour food products for quality measurement. *Journal of Food Engineering*. 2009, roč. 94. ISSN 02608774. Dostupné z DOI: 10.1016/j.jfoodeng.2009.02.022.
10. KNAUER, Uwe; MATROS, Andrea; PETROVIC, Tijana; ZANKER, Timothy; SCOTT, Eileen S.; SEIFFERT, Udo. Improved classification accuracy of powdery mildew infection levels of wine grapes by spatial-spectral analysis of hyperspectral images. *Plant Methods*. 2017, roč. 13. ISSN 17464811. Dostupné z DOI: 10.1186/s13007-017-0198-y.

11. PAULRAJ, Tharcis; CHELLIAH, Kezi Selva Vijila; CHINNASAMY, Sundar. Lung computed axial tomography image segmentation using possibilistic fuzzy C-means approach for computer aided diagnosis system. *International Journal of Imaging Systems and Technology*. 2019, roč. 29. ISSN 10981098. Dostupné z DOI: 10.1002/ima.22340.
12. TANG, Yinggan; ZHANG, Xiumei; LI, Xiaoli; GUAN, Xinping. Application of a new image segmentation method to detection of defects in castings. *International Journal of Advanced Manufacturing Technology*. 2009, roč. 43. ISSN 02683768. Dostupné z DOI: 10.1007/s00170-008-1720-1.
13. GOLNABI, H.; ASADPOUR, A. Design and application of industrial machine vision systems. *Robotics and Computer-Integrated Manufacturing*. 2007, roč. 23. ISSN 07365845. Dostupné z DOI: 10.1016/j.rcim.2007.02.005.
14. VELESACA, Henry O.; SUÁREZ, Patricia L.; MIRA, Raúl; SAPPA, Angel D. Computer vision based food grain classification: A comprehensive survey. *Computers and Electronics in Agriculture*. 2021, roč. 187. ISSN 01681699. Dostupné z DOI: 10.1016/j.compag.2021.106287.
15. GONZALEZ, R.C.; WOODS, R.E. *Digital Image Processing*. Pearson, 2018. ISBN 9780133356724. Dostupné také z: https://books.google.cz/books?id=0F05vgAACAAJ.
16. KASS, Michael; WITKIN, Andrew; TERZOPOULOS, Demetri. Snakes: Active contour models. *International Journal of Computer Vision*. 1988, roč. 1. ISSN 09205691. Dostupné z DOI: 10.1007/BF00133570.
17. WALT, Stéfan van der; SCHÖNBERGER, Johannes L.; NUNEZ-IGLESIAS, Juan; BOULOGNE, François; WARNER, Joshua D.; YAGER, Neil; GOUILLART, Emmanuelle; YU, Tony; CONTRIBUTORS, the scikit-image. scikit-image: image processing in Python. *PeerJ*. 2014, roč. 2, e453. ISSN 2167-8359. Dostupné z DOI: 10.7717/peerj.453.
18. CHAN, Tony F.; VESE, Luminita A. Active contours without edges. *IEEE Transactions on Image Processing*. 2001, roč. 10. ISSN 10577149. Dostupné z DOI: 10.1109/83.902291.
19. KRISTIADI, Agustinus. *Level Set Method Part I: Introduction* [online]. 2016-11-05. [cit. 2022-04-30]. Dostupné z: https://agustinus.kristia.de/techblog/2016/11/05/levelset-method/.
20. GRADY, Leo. Random walks for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 2006, roč. 28. ISSN 01628828. Dostupné z DOI: 10.1109/TPAMI.2006.233.
21. COMMONS, Wikimedia. *Northern lights (Auroa borealis) over the Víkurkirkja church at Vik, Iceland*. 2017. Dostupné také z: https://commons.wikimedia.org/wiki/File:Church_of_light.jpg. File: Church of light.jpg.
22. FELZENSZWALB, Pedro F.; HUTTENLOCHER, Daniel P. Efficient graph-based image segmentation. *International Journal of Computer Vision*. 2004, roč. 59. ISSN 09205691. Dostupné z DOI: 10.1023/B:VISI.0000022288.19776.77.
23. ACHANTA, Radhakrishna; SHAJI, Appu; SMITH, Kevin; LUCCHI, Aurelien; FUA, Pascal; SÜSSTRUNK, Sabine. SLIC superpixels compared to state-of-the-art superpixel methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 2012, roč. 34. ISSN 01628828. Dostupné z DOI: 10.1109/TPAMI.2012.120.
24. VEDALDI, Andrea; SOATTO, Stefano. Quick shift and kernel methods for mode seeking. In: 2008, sv. 5305 LNCS. ISSN 16113349. Dostupné z DOI: 10.1007/978-3-540-88693-8_52.
25. CHENG, Yizong. Mean Shift, Mode Seeking, and Clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 1995, roč. 17. ISSN 01628828. Dostupné z DOI: 10.1109/34.400568.

26. YASER, Ajmal Sheikh; ERUM, Arif Khan; KANADE, Takeo. Mode-seeking by medoidshifts. In: 2007. Dostupné z DOI: 10.1109/ICCV.2007.4408978.

27. TRÉMEAU, Alain; COLANTONI, Philippe. Regions adjacency graph applied to color image segmentation. *IEEE Transactions on Image Processing*. 2000, roč. 9. ISSN 10577149. Dostupné z DOI: 10.1109/83.841950.

28. OTSU, Nobuyuki. THRESHOLD SELECTION METHOD FROM GRAY-LEVEL HISTOGRAMS. *IEEE Trans Syst Man Cybern*. 1979, roč. SMC-9. ISSN 00189472. Dostupné z DOI: 10.1109/tsmc.1979.4310076.

29. BERGMANN, Paul; BATZNER, Kilian; FAUSER, Michael; SATTLEGGER, David; STEGER, Carsten. The MVTec Anomaly Detection Dataset: A Comprehensive Real-World Dataset for Unsupervised Anomaly Detection. *International Journal of Computer Vision*. 2021, roč. 129. ISSN 15731405. Dostupné z DOI: 10.1007/s11263-020-01400-4.

30. CZIMMERMANN, Tamás; CIUTI, Gastone; MILAZZO, Mario; CHIURAZZI, Marcello; ROCCELLA, Stefano; ODDO, Calogero Maria; DARIO, Paolo. *Visual-based defect detection and classification approaches for industrial applications—A SURVEY*. Sv. 20. 2020. ISSN 14248220. Dostupné z DOI: 10.3390/s20051459.

31. PEDREGOSA, F.; VAROQUAUX, G.; GRAMFORT, A.; MICHEL, V.; THIRION, B.; GRISEL, O.; BLONDEL, M.; PRETTENHOFER, P.; WEISS, R.; DUBOURG, V.; VANDERPLAS, J.; PASSOS, A.; COURNAPEAU, D.; BRUCHER, M.; PERROT, M.; DUCHESNAY, E. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*. 2011, roč. 12, s. 2825–2830.

Obsah přiloženého média