VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ
DEPARTMENT OF COMPUTER SYSTEMS

APLIKACE MRAVENČÍCH ALGORITMŮ V ÚLOZE ZPRACOVÁNÍ OBRAZU

ANT COLONY OPTIMIZATION FOR IMAGE PROCESSING

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **MATĚJ PRÁŠEK**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. MICHAL BIDLO, Ph.D.**
SUPERVISOR

BRNO 2020

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačových systémů (UPSY) Akademický rok 2019/2020

Zadání bakalářské práce

Student: **Prášek Matěj**
Program: Informační technologie
Název: **Aplikace mravenčích algoritmů v úloze zpracování obrazu**
Ant Colony Optimization for Image Processing
Kategorie: Umělá inteligence

Zadání:

1. Nastudujte problematiku mravenčích algoritmů (Ant Colony Optimization - ACO) a jejich možností použití v úlohách zpracování obrazu.
2. Zpracujte studii na uvedené téma.
3. Zvolte vhodnou úlohu z oblasti zpracování obrazu a pro tuto navrhněte a implementujte variantu mravenčího algoritmu s cílem realizace vybrané operace nad danými obrazy.
4. Proveďte sadu experimentů pro různé konfigurace systému.
5. Zhodnoťte dosažené výsledky a diskutujte další možnosti projektu.

Literatura:

- M. Dorigo, T. Stützle: Ant Colony optimization. The MIT Press, Cambridge, MA, USA, 2004
- S. Gullipalli: Search Improvemnt Based on Ants Performance in Image Edge Detection using ACO. University of Paderborn, DE, 2015

Pro udělení zápočtu za první semestr je požadováno:

- Splnění bodů 1 a 2 zadání, demonstrace prototypu aplikace z bodu 3 zadání.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Bidlo Michal, Ing., Ph.D.**
Vedoucí ústavu: Sekanina Lukáš, prof. Ing., Ph.D.
Datum zadání: 1. listopadu 2019
Datum odevzdání: 28. května 2020
Datum schválení: 25. října 2019

Abstrakt

Tato bakalářské práce je zaměřena na detekci hran v obrazu pomocí algoritmu Ant Colony Optimization. Zaměřuji se na různé způsoby redukce šumu, hledání izolovaných hran a nalezení optimálních parametrů pro většinu obrazů.

Abstract

This bachelor thesis is focused to edge detection using the Ant Colony Optimization algorithm. I focus on noise reduction, searching for isolated edges and finding optimal parameters for the majority of pictures.

Klíčová slova

mravenčí algoritmus, optimalizace mravenčí kolonií, detekce hran, redukce šumu

Keywords

ant algorithm, ant colony optimization, edge detection, noise reduction

Citace

PRÁŠEK, Matěj. *Aplikace mravenčích algoritmů v úloze zpracování obrazu.* Brno, 2020. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Michal Bidlo, Ph.D.

Aplikace mravenčích algoritmů v úloze zpracování obrazu

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Michala Bidla, Ph.D. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Matěj Prášek
28. května 2020

Poděkování

Rád bych poděkoval Ing. Michalu Bidlovi, Ph.D. za vedení této bakalářské práce, za jeho rady, poznatky a postřehy.

Obsah

Kapitola 1

Úvod

V současné době je v řadě aplikací kladen důraz na nalezení optimálního řešení určitého problému. Inženýrské postupy znají efektivní metody, jak řešit různé (i náročné) problémy. Pro některé úlohy je však typické, že se jejich náročnost stupňuje exponenciálně v závislosti na velikosti instance. Typickým příkladem takového problému je problém obchodního cestujícího (Traveling Salesman Problem), zkráceně TSP. Cílem jeho řešení je nalézt optimální (tj. nejkratší) cestu obchodního cestujícího mezi množinou měst, přičemž je dovoleno navštívit každé město právě jedenkrát. Problémem však je, že s narůstajícím počtem měst se náročnost nalezení řešení enormně zvyšuje a od jisté hranice je nalezení optimální cesty prakticky nemožné. Podobné rysy jsou v informatice typické tzv. NP-úplným problémům, mezi které patří také TSP. V současné době je jedinou možností, jak nalézt řešení takového problému v rozumném čase, použití různých heuristických postupů.

Jednou třídou metod, které se ukázaly být v řešení TSP velmi nápomocné, jsou například mravenčí algoritmy. Dorigo a spol. poprvé publikovali studii, v níž ukázali možnost řešení TSP pomocí principů vysledovaných v chování mravenců v přírodě [5]. Jejich první algoritmus, nazvaný Ant System, byl později rozšířen a doplněn o další prvky (souhrnně např. v [10]).

Kromě problému obchodního cestujícího byly mravenčí algoritmy časem aplikovány i na jiné problémy. V poslední době byly úspěšně aplikovány pro ovládání kvantových systémů [24], dále pro detekci hran v obraze [12], plánování cesty při parkování [23] a při rozhodování pohybu robota [13].

Cílem této práce je použití mravenčích algoritmů pro detekci hran v obraze. Nevýhodou původního přístupu z [12] je velký šum v případě náhodného rozmístění mravenců a náročnost detekce izolovaných hran v případě rozmístění mravenců na místa s velkým rozdílem intenzit sousedních pixelů. Proto budou v rámci této práce zkoumány nové přístupy, které by dovolily kvalitu výsledků dále vylepšit. Důraz bude kladen též na objevení metody a jejích parametrů, která by zajistila přijatelný výsledek nezávisle na typu obrazu.

Předmětem této práce bude vylepšení metody detekce hran k získání příznivých výsledků pomocí různých úprav algoritmu a jeho parametrů a experimentování s parametry s cílem nalezení jejich optimálního nastavení.

Kapitola 2

Mravenčí algoritmy

Tato sekce vychází z knihy Ant Colony Optimization [10], ve které se Marco Dorigo spolu s Thomasem Stützlem zabývají do hloubky mravenčími algoritmy.

2.1 Od reálných mravenců k umělým

Ač se můžou mravenci zdát jako jednoduchý hmyz, jejich kolonie jsou ve skutečnosti distribuované a tvoří strukturovanou společnost. V důsledku této organizace jsou mravenčí kolonie schopny plnit složité úkoly, které mnohdy přesahují schopnosti mravenčího jedince.

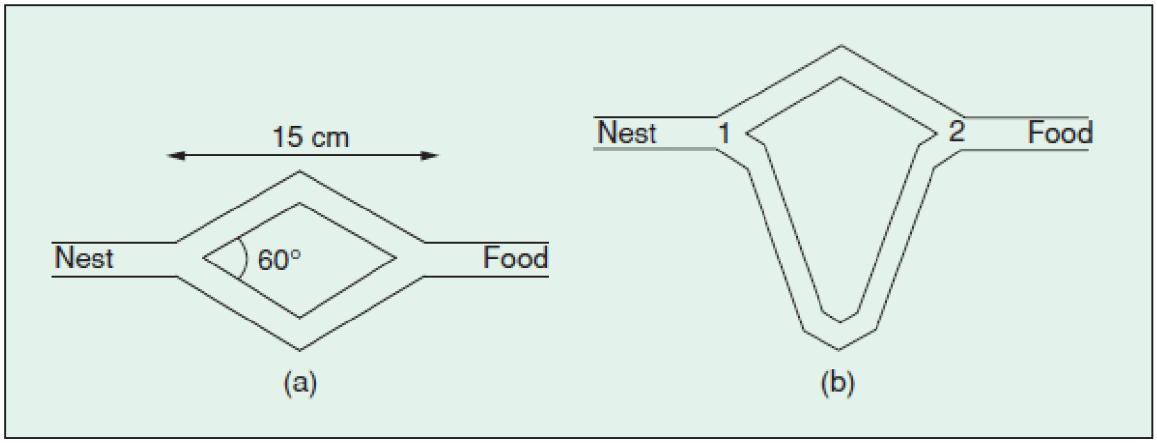
Odvětví mravenčích algoritmů studuje chování těchto mravenců a inspiruje se jimi při návrhu řešení optimalizačních problémů. Nejrůznější části chování mravenčích kolonií daly vzniknout mnoha druhům mravenčích algoritmů. Sledovaným chováním je například dělba práce, třídění larev, hledání potravy a jiné. Většina druhů mravenců má jen základní vizuální vnímání a některé druhy jsou úplně slepé, proto se mravenci dorozumívají pomocí nepřímé komunikace, které dosahují úpravou prostředí. Například mravenec při hledání potravy vypouští na zem chemikálie zvané *feromony*. V tomto případě se jedná konkrétně o *feromonové stezky* (trail pheromone), kterými značí cesty mezi hnízdem a potravou a zvyšují pravděpodobnost, že ostatní mravenci půjdou stejnou cestou. Pokud toto chování chceme použít i v případě umělých mravenců, musíme k jejich organizaci použít také nepřímou komunikaci. Jeden z nejúspěšnějších konceptů inspirace optimalizačních technik chováním mravenců jsou právě mravenčí algoritmy (ant colony optimization), zkráceně ACO, inspirující chováním mravenců při hledání potravy a jsou určený k řešení diskrétních optimalizačních problémů.

2.1.1 Experiment dvou mostů

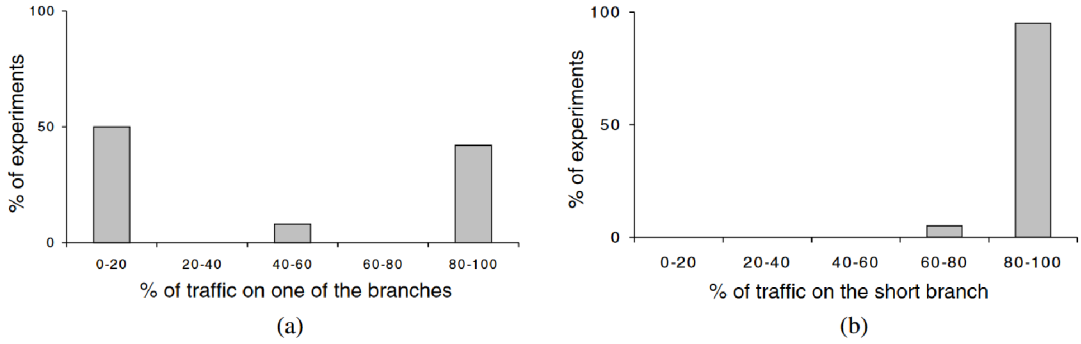
Experiment dvou mostů byl poprvé proveden J. -L. Deneubourgem a kolegy v roce 1990 [4], při kterém pozorovali chování argentinských mravenců při přenosu potravy do mraveniště, tvorbu feromonové cestičky a její následné sledování v laboratorních podmínkách. Experiment je založen na dvou „mostech“ spojujících mraveniště a zdroj potravy. Při sestavování mostů byl použit poměr $r = l_l/l_s$, kde l_l je délka delšího mostu a l_s je délka kratšího mostu.

V prvním experimentu měly mosty stejnou délku ($r = 1$; obr 2.1a). Mravenci si ze začátku vybírali cestu náhodně, protože na žádném z mostů nebyly feromonové stopy. Po chvíli se mravenci uchýlili k cestě pouze po jednom mostu. Je to z důvodu feromonové cestičky, která získává na intenzitě s každým průchodem každého mravence, z čehož vyplývá, že pokud přes most projde více mravenců, bude obsahovat více feromonů a tím pádem další mravenci projdou s větší pravděpodobností přes ten samý most. Výsledky experimentu

si můžete prohlédnout na obrázku 2.2a, kde nám svislá osa udává procento experimentů a vodorovná osa udává procento mravenců přecházející přes sledovaný most.



Obrázek 2.1: Experiment dvou mostů, kde mají mosty (a) stejnou délku ($r = 1$), (b) různou délku ($r = 2$) [6].

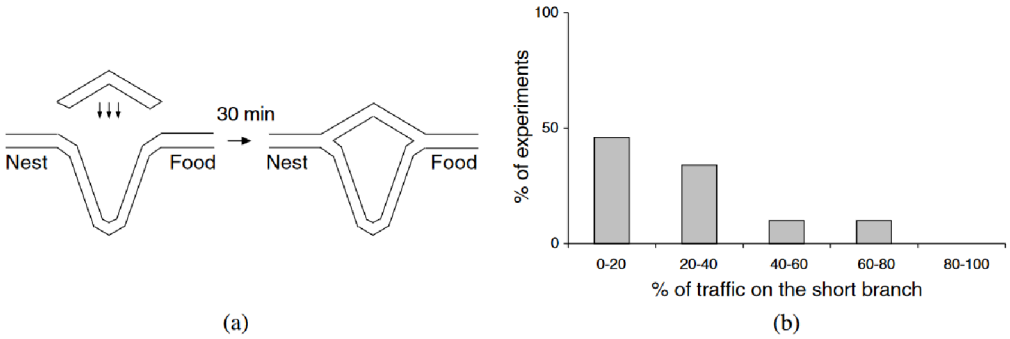


Obrázek 2.2: Výsledky získané z experimentu dvou mostů zobrazující procento mravenců přecházející po jednom z mostů. (a) Výsledky, kdy měly mosty stejnou délku ($r = 1$) a (b) výsledky, kdy měly mosty různou délku ($r = 2$).

Ve druhém experimentu, obr 2.1b, byl poměr mostů určen jako $r = 2$, což znamená, že delší most byl dvojnásobně delší oproti kratšímu. Z počátku, kdy nebyly ani na jednom mostě žádné feromony, mravenci vybírali most náhodně, ale už na zpáteční cestě od zdroje potravy byla na kratším mostě větší intenzita feromonů, proto mravenci preferovali ten. Oproti experimentu 2.1a byl výběr kratšího mostu značně rychlejší z toho důvodu, že za stejný čas přešlo přes kratší most více mravenců, takže intenzita feromonů na kratším mostě jednoduše převážila intenzitu feromonů na delším mostě. Jak můžeme vidět na grafu 2.2b, kde je na svislé ose opět procento experimentů a na vodorovné ose procento mravenců přecházející kratší most, ve všech případech většina mravenců zvolila kratší most a ve většině případů se jednalo o více než 80 % všech mravenců.

Za zmínku stojí i třetí experiment, obr 2.3a, kde měli mravenci z počátku k dispozici jen delší most, přičemž kratší most byl přidán po 30 minutách. V tomto případě, jak můžeme vidět na grafu 2.3b, byl delší most preferovaný i nadále a kratší most byl využit jen hrstkou mravenců. Mravenci stále využívali delší most z důvodu vysoké koncentrace feromonů.

Evaporace feromonů je příliš pomalá na to, aby mravenci mohli „zapomenout“ delší trasu a objevili trasu kratší.



Obrázek 2.3: Experiment dvou mostů, (a) kdy je mravencům poskytnut jen delší most a kratší je přidaný až po 30 minutách. (b) Výsledky experimentu zobrazující procento mravenců využívající kratší most.

Experiment dvou mostů nám dokázal, že mravenci mají určitou schopnost optimalizace, díky které dokáží najít nejkratší cestu mezi dvěma body. Abychom mohli tuto vlastnost využít také u našich umělých mravenců, potřebujeme vytvořit graf, jehož uzly představují místa, která nás zajímají a hrany představují cesty mezi nimi. Dále budeme předpokládat diskrétní čas a pohyb mravenců mezi dvěma uzly jako jeden krok.

2.2 Problém obchodního cestujícího

Problém obchodního cestujícího (Traveling Salesman Problem), zkráceně TSP, je grafem s ohodnocenými hranami, kde jsou města označována jako uzly, cesty mezi městy jako hrany a vzdálenost mezi městy jako ohodnocení hrany. Často se jedná o úplný graf, to znamená, že je každý uzel propojený s každým. Graf může být buď neorientovaný, pak mluvíme o symetrickém TSP a znamená to, že je vzdálenost d mezi městy i a j definována vztahem $d_{ij} = d_{ji}$, nebo orientovaný, pak mluvíme o asymetrickém TSP a vzdálenost mezi městy je definována vztahem $d_{ij} \neq d_{ji}$. Pro demonstraci algoritmu Ant System se v této práci zaměřím pouze na řešení symetrického TSP.

Řešením TSP je nejkratší cesta, v rámci které navštívíme všechna města právě jednou. Jelikož je počet řešení roven $O(n!)$, řešení prohledávání hrubou silou se stává nepraktickým již při 20 městech, což dělá z TSP NP-úplný problém. TSP s větším počtem měst není možné vyřešit v rozumném čase, proto vznikly různé aproximační algoritmy, které mají odchylku od optimálního řešení v řádu jednotek procent a vyřeší problém v mnohem kratším čase. [8] [7]

2.3 Ant system

Marco Dorigo popsal Ant System, zkráceně AS, ve článku Ant System: Optimization by a Colony of Cooperating Agents [9], ze kterého budu při popisu AS čerpat.

AS je první algoritmus, který se inspiroval chováním mravenčích kolonií. Algoritmus pracuje s umělými mravenci, nazývanými agenty, kteří cestují grafem a při cestě mezi dvěma

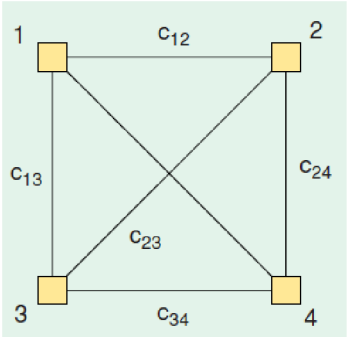
uzly vypouští feromonovou stopu. Kompletní cesta jednoho mravence je považována za jedno řešení. Umělí mravenci se od těch skutečných liší ve třech hlavních bodech:

- Umělí mravenci mají paměť,
- nejsou slepí, ale ví, jak vzdálené jsou okolní uzly,
- žijí v prostředí s diskrétním časem.

AS je určen hlavně k řešení problému obchodního cestujícího, proto si jeho funkci lépe popíšeme v následující podsekci.

2.3.1 Využití algoritmu Ant System v problému obchodního cestujícího

Pro řešení problému TSP pomocí AS si musíme vytvořit konstrukční graf, kde jsou města reprezentována uzly grafu a přechody mezi městy jsou reprezentovány hranami grafu, jak je uvedeno na obrázku 2.4.



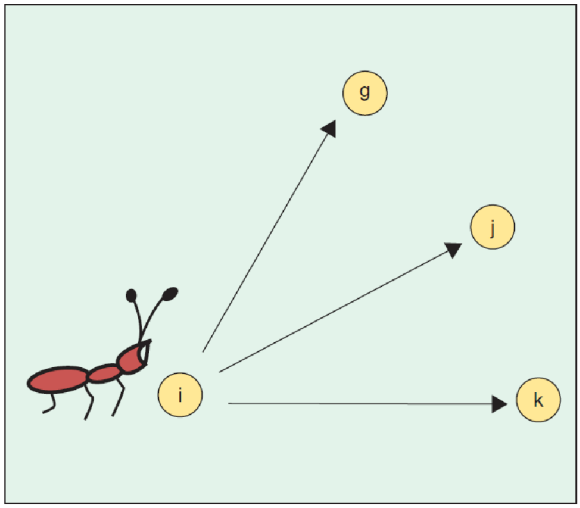
Obrázek 2.4: Konstrukční graf pro řešení TSP pomocí AS se čtyřmi městy [6].

Sestavení kandidátního řešení každým mravencem probíhá tak, že mravenec vychází z počátečního města, musí projít celým grafem a hlídá se, aby bylo každé město navštíveno daným mravencem právě jednou. Každý mravenec se v rámci realizace jednoho každého kroku rozhoduje, které další město navštívit z města aktuálního, přičemž jeho rozhodování má pravděpodobnostní charakter a je dáno vzdálenostmi (ohodnocením hran) těch měst, která vidí, jak je znázorněno na obrázku 2.5. Výpočet pravděpodobností se během konstrukce cesty každého mravence si rozebere v této sekci.

Při řešení TSP pomocí mravenčích algoritmů se každý mravenec řídí určitými pravidly:

- Mravenec si vybere další město s pravděpodobností založené na vzdálenosti cílového města od aktuálního a množstvím feromonu na oné cestě.
- Abychom zajistili, že mravenec nenavštíví jedno město vícekrát, využijeme tzv. tabu list, na kterém jsou všechna již navštívená města daného mravence.
- Po navštívení všech měst, zvýší mravenec množství feromonů na každé hraně, kterou prošel.

Každý mravenec si v čase t vybere město, do kterého dojde v čase $t + 1$. Jestliže m je rovno počtu mravenců, pak pohybu m mravenců v intervalu $(t, t+1)$ říkáme *krok*. Při m městech, po n krocích každý mravenec navštíví každé město, čemuž říkáme *cyklus*. V tento



Obrázek 2.5: Mravenec ve městě i si vybírá město, do kterého půjde. Pokud město j ještě nebylo navštíveno, mravenec ho navštíví s pravděpodobností úměrné množství feromonů na hraně (i, j) [6].

moment aktualizujeme množství feromonů τ na každé hraně (i, j) podle (2.1)

$$\tau_{ij} = (1 - \rho)\tau_{ij} + \Delta\tau_{ij}, \tag{2.1}$$

kde ρ je koeficient evaporace feromonu a platí, že $\rho < 1$ a $\Delta\tau_{ij}$ reprezentuje přírůstek feromonu na hraně (i, j) a je definován (2.2)

$$\Delta\tau_{ij} = \sum_{k=1}^{m} \Delta\tau_{ij}^{k}, \tag{2.2}$$

kde $\Delta\tau_{ij}^{k}$ je množství feromonu zanechaném mravencem k na hraně (i, j) a je definován (2.3)

$$\Delta\tau_{ij}^{k} = \begin{cases} \dfrac{Q}{L_k}, & \text{pokud } k\text{-tý mravenec použil hranu } (i, j) \\ 0, & \text{jinak,} \end{cases} \tag{2.3}$$

kde Q je konstanta a L_k je celková délka cesty k-tého mravence.

Abychom zajistili, že žádný mravenec nenavštíví jedno město během jednoho cyklu vícekrát, definujeme si pro každého mravence *tabu list*, do kterého si budeme ukládat všechna navštívená města. Tabu list využijeme i na konci každého cyklu, ze kterého jsme schopni sestavit trasu mravence, která je zároveň i řešením. Poté tabu list opět vyprázdníme.

Dále si definujme *viditelnost* η_{ij}, kterou vypočítáme jako $1/d_{ij}$, kde d_{ij} je vzdálenost města j od města i. Viditelnost se v průběhu AS nemění.

Pravděpodobnost, že k-tý mravenec půjde z města i do města j je definovaná (2.4)

$$p_{ij}^{k}(t) = \begin{cases} \dfrac{[\tau_{ij}(t)]^{\alpha} \cdot [\eta_{ij}]^{\beta}}{\sum_{k \in \text{allowed}_k} [\tau_{ik}(t)]^{\alpha} \cdot [\eta_{ik}]^{\beta}}, & \text{pokud } j \in \text{allowed}_k \\ 0, & \text{jinak,} \end{cases} \tag{2.4}$$

kde $\text{allowed}_k = \{\text{Množina měst} - \text{tabu}_k\}$ a α s β jsou parametry, které kontrolují relativní vztah mezi feromony a viditelností.

2.4 Ant Colony System

Jako další algoritmus inspirovaný mravenčí kolonií si představíme Ant Colony System, zkráceně ACS. ACS vychází z AS a liší se od něj ve třech hlavních bodech.

1. Při volbě, zda se mravenec vydá po feromonové stopě, nebo zda bude prohledávat okolí, je aplikováno mnohem agresivnější pravidlo, které podporuje nejlepší možnou cestu. (Volba cesty)

2. Evaporace a přírůstek feromonů se aplikuje pouze na hrany, kterými při své cestě prošel nejúspěšnější mravenec v daném cyklu. (Globální aktualizace feromonů)

3. Po každém průchodu mravence hranou (i, j) pro pohyb z města i do města j mravenec odebere část feromonů z hrany, aby zvýšil pravděpodobnost prohledání ostatních cest. (Lokální aktualizace feromonů)

Tyto úpravy si detailněji probereme později v kapitole.

2.4.1 Volba cesty

Při volbě cesty z města i zvolí mravenec k město j podle (2.5).

$$j = \begin{cases} \text{argmax}_{l \in \text{allowed}_k} \{\tau_{il}[\eta_{il}]^{\beta}\}, & \text{pokud } q \leq q_0 \\ J, & \text{jinak,} \end{cases} \tag{2.5}$$

kde q je náhodně číslo vygenerované rovnoměrným rozložením na intervalu $\langle 0, 1\rangle$, q_0 je parametr a platí, že $q0 \in \langle 0, 1\rangle$ a J je vypočítáno pomocí (2.4) s parametrem $\alpha = 1$. Jinými slovy, pravděpodobnost, že si mravenec vybere nejlepší cestu vypočítanou pomocí množství feromonů na cestě a heuristickou informací, je rovna q_0 a pravděpodobnost, že mravenec bude vyhledávat nové cesty je rovna $1-q_0$. Z toho vyplývá, že úpravou parametru q_0 ovlivňujeme, zda chceme aby mravenci spíše následovali feromonovou stopu, nebo aby se zaměřili spíše na cesty jiné.

2.4.2 Globální aktualizace feromonů

Po každém cyklu se vybere nejlepší mravenec a aktualizují se feromony na všech hranách, kterými mravenec prošel podle (2.6), převzatého z [8].

$$\tau_{ij} \leftarrow (1 - \rho)\tau_{ij} + \rho\Delta\tau_{ij} \tag{2.6}$$

kde je $\Delta\tau_{ij}$ dána (2.7)

$$\Delta\tau_{ij} = \begin{cases} 1/L_{gb}, & \text{pokud } (i, j) \in \text{nejlépe ohodnocená cesta} \\ 0, & \text{jinak.} \end{cases} \tag{2.7}$$

Jelikož nově neaktualizujeme feromony na všech cestách, ale jen na té nejlepší, snížila se nám náročnost výpočtu aktualizace feromonů z $\mathcal{O}(n^2)$ na $\mathcal{O}(n)$ [10].

2.4.3 Lokální aktualizace feromonů

Společně s globální aktualizací feromonů, mravenci v algoritmu ACS používají i pravidlo pro lokální aktualizaci feromonů, popsané v předpisu 2.8, které využívají okamžitě při průchodu hranou (i, j).

$$\tau_{ij} \leftarrow (1 - \xi) \cdot \tau_{ij} + \xi\tau_0, \tag{2.8}$$

kde ξ a τ_0 jsou parametry. τ_0 se rovná počáteční hodnotě feromonů a $\xi \in (0, 1)$. Experimenty ukázaly, že doporučená hodnota pro ξ je 0,1 a doporučená hodnota pro τ_0 je $1/nC^{nn}$, kde n je počet měst a C^{nn} je vzdálenost nejbližších dvou měst [10].

Smysl lokální aktualizace feromonů je ten, že se každým průchodem hranou (i, j) snižuje intenzita feromonů na dané hraně, což zvyšuje pravděpodobnost, že další mravenec půjde jinou cestou. Díky tomu se mravenci mnohem více zaměřují na procházení hran, které v daný moment nejsou považovány za optimální cestu.

Oproti AS, při kterém nezáleželo na pořadí, ve kterém mravenci prochází grafem, v ACS, kvůli lokální aktualizaci feromonů, toto rozhodnutí značně ovlivňuje chování mravenců. Ve většině scénářích se mravenci střídají po každém kroku, ale zatím žádný experiment nepotvrdil, že by měl některý z přístupů lepší výsledky [10].

2.5 Další varianty mravenčích algoritmů

V této podkapitole si uděláme stručný přehled o ostatních variantách mravenčích algoritmů.

2.5.1 Max-Min Ant system

Max-Min Ant system, zkráceně MMAS, byl představen T. Stutzlem a H. Hoosem v roce 1997 v MAX-MIN Ant System and local search for the traveling salesman problem [19] a v roce 2000 v Future Generation Computer Systems [20].

MMAS vychází z AS, do kterého vnesl čtyři modifikace:

1. Úprava feromonů proběhne jen na hranách, kterými prošel při své cestě nejúspěšnější mravenec. Tento přístup může vést ke stagnaci, při které se budou feromony zvyšovat pouze na jediné cestě. Proto byla zavedena druhá modifikace.
2. Množství feromonů na hraně může být pouze v intervalu $\langle \tau_{min}, \tau_{max} \rangle$.
3. Množství feromonů na všech hranách je na začátku nastaveno na τ_{max}, což společně s nízkou hodnotou evaporace feromonů způsobí na začátku větší míru prohledávání.
4. V případě stagnace nebo dlouhodobého nezlepšování výsledků dojde k nové inicializaci feromonů.

Aktualizace feromonové cestičky probíhá ve dvou fázích. V první fázi se feromony vypaří podle (2.9)

$$\tau_{ij} \leftarrow (1 - \rho)\tau_{ij} \tag{2.9}$$

a následně zvýší přírůstkem feromonů podle (2.10)

$$\tau_{ij} \leftarrow \tau_{ij} + \Delta\tau_{ij}^{best}, \tag{2.10}$$

kde $\Delta\tau_{ij}^{best}$ se vypočítá jako $1/C^{best}$, kde C^{best} je délka trasy nejlepšího mravence, která může být buď nejlepší v daném cyklu, nebo nejlepší celkově. Experimenty ukázaly, že pro jednoduché TSP je lepší variantou nejlepší cesta v iteraci a pro složité TSP je lepší použít nejlepší nalezenou [10].

2.5.2 Elitist Ant System

Elitist Ant System, zkráceně EAS, je první algoritmus odvozený z AS [10]. Hlavní myšlenkou je obrovské posílení hran, které jsou součástí nejlepší cesty T^{bs}.

Posílení cesty T^{bs} je zaručeno přidáním množství feromonů e/C^{bs} na hrany té cesty, kde e ke parametr, který definuje váhu danou nejlepší cestě T^{bs} a C^{bs} je délka cesty. Z toho je patrné, že se z (2.1) stane (2.11).

$$\Delta\tau_{ij} = \sum_{k=1}^{m} \Delta\tau_{ij}^{k} + e\Delta\tau_{ij}^{bs}, \tag{2.11}$$

kde $\Delta\tau_{ij}^{k}$ je definováno (2.3) a $\Delta\tau_{ij}^{bs}$ je definována jako (2.12)

$$\Delta\tau_{ij}^{bs} = \begin{cases} 1/C^{bs}, & \text{pokud } (i,j) \in T^{bs} \\ 0, & \text{jinak.} \end{cases} \tag{2.12}$$

Výsledky experimentů dokazují, že EAS se správnou hodnotou e najde lepší výsledky za menší dobu, než je tomu u AS [10].

Kapitola 3

Zpracování obrazu pomocí mravenčích algoritmů

Variant zpracování obrazu je mnoho. Mezi nejrozsáhlejší patří například filtrace, pod kterou si můžeme představit například redukci šumu, detekci hran, opravy poruch a retušování, různé afinní transformace, jako jsou například rotace, škálování a zkosení, transformace barev a různé techniky počítačové vidění.

V článku [14] byly mravenčí algoritmy použity pro segmentaci rentgenového snímků mozku, kde byla vyfiltrována šedá hmota, bílá hmota a mozkomíšní mok. Díky tomu bylo možné určit poměr jejich zastoupení a pochopit lépe strukturu mozku. Tato informace je také užitečná k objevení některých neurotických chorob, jako je třeba Alzheimerova choroba, Parkinsonova choroba apod.

V článku [25] byla porovnána segmentace pomocí mravenčích algoritmů a algoritmu K-means. Proběhly dva experimenty. V prvním experimentu byl použit šedotónový obrázek a porovnán výsledek obou algoritmů. Ve druhém experimentu byl barevný obrázek převedený na šedotónový s velkou váhou zelené barvy a výsledný obrázek byl opět segmentován pomocí obou algoritmů. Ve druhém experimentu byl výsledek pomocí algoritmu K-means téměř nepoužitelný, ale mravenčí algoritmy stále dodaly přijatelný výsledek.

Ve článku [26] se mravenčí algoritmy využili na určení prahu pro převod na černobílý obraz a výsledek se porovnával s Otsuho metodou [17] a s určováním pomocí genetických algoritmů [2]. Výsledek ukázal, že na rozdíl od obou metod, které dokázaly oddělit pozadí od popředí, pomocí mravenčích algoritmů bylo možné zobrazit i různé detaily, jako je obličej, klobouk i vzor na obleku. Dále se ukázalo, že výběr správného prahu zvládli mravenčí algoritmy rychleji, než zbylé dvě metody. Nevýhodou mravenčích algoritmů je ovšem množství parametrů, které je třeba správně nastavit pro každý obrázek, abychom dostali požadovaný výsledek.

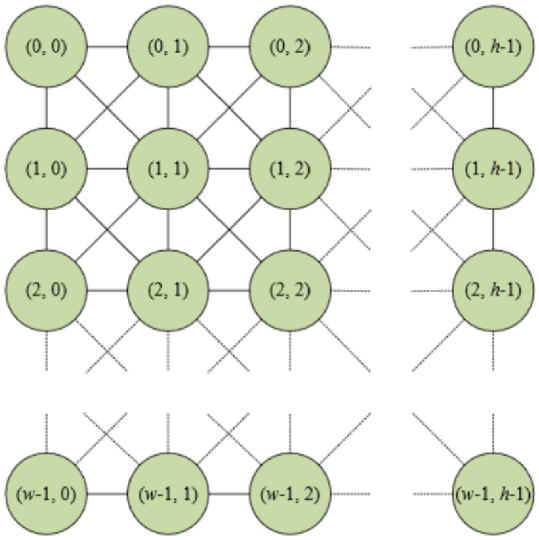
Ve článku [22] byly mravenčí algoritmy použity na rozpoznávání obličejů. Detekce probíhala ve třech krocích. Prvním krokem bylo předzpracování daného obrazu, aby bylo možné lépe detekovat jednotlivé obličejové rysy. Provádělo se například odstranění šumu nebo ořezání pozadí. Ve druhém kroku se vyhledávaly obličejové rysy pomocí algoritmu ACO, které se pak ve třetím kroku zpracovávaly pomocí genetických algoritmů.

Ve článku [3] byly mravenčí algoritmy použity společně s přizpůsobeným filtrem pro detekci žil z oftalmoskopických snímků, které mohou hrát významnou roli v některých závažných patologií, jako jsou například cukrovka nebo zvýšený krevní tlak.

Tato bakalářská práce se zaměřuje na detekci hran v obrazu pomocí mravenčích algoritmů. V této kapitole se dále zaměříme na různé přístupy detekce hran v obrazu pomocí mravenčích algoritmů a vysvětlíme si, jak pomocí ACO detekovat hrany v obrazu.

3.1 Interpretace obrazu pro detekci hran pomocí mravenčích algoritmů

Na začátku kapitoly bylo ukázáno, že je možné použít mravenčí algoritmy v metodách zpracování obrazu. Je tomu tak, protože obraz lze chápat jako graf, jehož uzly jsou reprezentovány jednotlivými pixely a hrany jsou reprezentovány přechody mezi sousedícími pixely horizontálně, vertikálně a diagonálně, jak je zobrazeno na obrázku 3.1.



Obrázek 3.1: Graf zobrazující obrázek o velikosti w x h. Každý pixel je představován uzlem v grafu a je propojen s jeho sousedícími pixely horizontálně vertikálně a diagonálně, čímž tvoří hrany grafu. [1] [12]

V této sekci se budeme specificky zabývat problémem detekce hran v obrazu pomocí mravenčích algoritmů. Pro Tyto potřeby použijeme algoritmus 1 převzatý z [1] a popíšeme úpravy algoritmu ACS tak, aby jej bylo možné použít pro detekci hran. K mravenců je

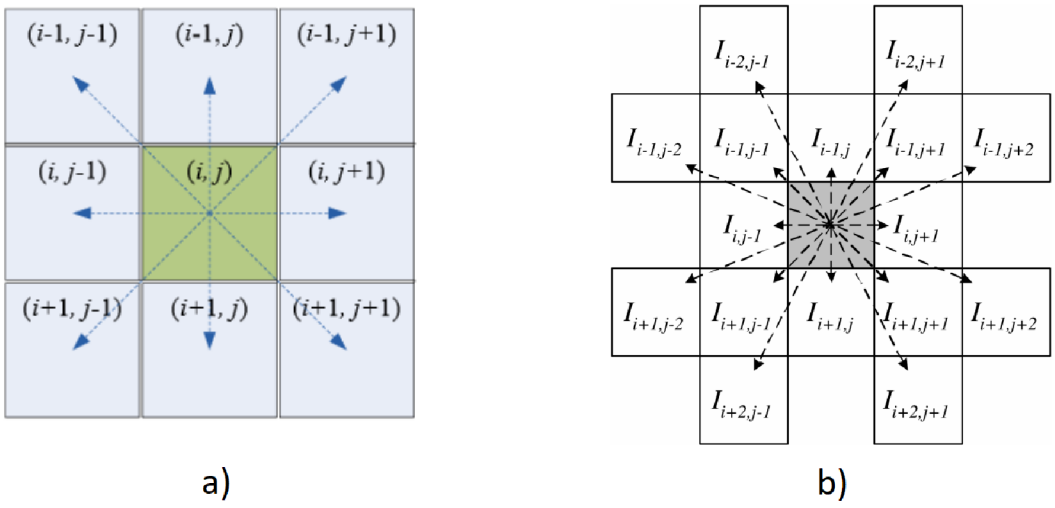
náhodně rozmístěno do obrazu a jejich následný pohyb je řízen intenzitou feromonů na jednotlivých pixelech a viditelností η_{ij}.

Algorithm 1: Pseudokód pro detekci hran pomocí ACS [1]

```
Inicializace obrázku, mravenců, feromonů a heuristiky;
foreach iterace n = 1:N do
    foreach konstrukční krok l = 1:L do
        foreach mravenec k = 1:K do
            Přejdi na další pixel;
            Aktualizuj feromony podle lokálního aktualizačního pravidla;
        end
    end
    Aktualizuj feromony podle globálního aktualizačního pravidla;
end
```

Na začátku simulace je hodnota intenzity feromonů na každém pixelu nastavena na hodnotu τ_0 a jedná se obvykle o velmi malou nenulovou hodnotu. Každý konstrukční krok se každý mravenec přesune na další pixel podle (2.5). Rovnice (2.4) je upravena tak, že rozdíl intenzit daných pixelů v obrazu podle obrázku 3.2a, rovnice (3.1).

$$V_c(I_{i,j}) = |I_{i-1,j-1} - I_{i+1,j+1}| + |I_{i-1,j+1} - I_{i+1,j-1}| + |I_{i,j-1} - I_{i,j+1}| + |I_{i-1,j} - I_{i+1,j}| \quad (3.1)$$



Obrázek 3.2: Okolí pro výpočet heuristiky intenzity za účelem detekce hran (a) osmiokolí [12], (b) šestnáctiokolí [21].

Na základě veličiny V_c se následně vypočítá viditelnost uzlů (pixelů) z aktuálního uzlu, ve kterém se mravenec nachází podle (3.2).

$$\eta_{ij} = \frac{V_c(I_{ij})}{V_{max}} \quad (3.2)$$

Další změna se týká rovnice (2.7), kde je $\Delta\tau_{ij}$ vypočítána jako součet průměrné intenzity všech pixelů, kterými každý z mravenců, kteří navštívili pixel (i, j), tento cyklus prošel. Dále v původním algoritmu ACS pro TSP probíhá globální aktualizace feromonů jen na

cestě nejlepšího mravence podle (2.6). Pro případ detekce hran je globální aktualizační pravidlo provedeno na cestách všech mravenců, protože spojení všech těchto cest vede ke kompletnímu řešení.

Průběžné řešení je sestaveno po každém cyklu pomocí (3.3), čím získáme intenzitu daného pixelu a pro správné zobrazení výsledku nastavíme všechny pixely s $I > 0$ na černou barvu a pixely s $I == 0$ na barvu bílou. Výsledné řešení je sestaveno na konci posledního cyklu stejným způsobem.

$$I \doteq (\tau_{ij} - \min)/\text{diff} \cdot 255, \tag{3.3}$$

kde *min* je minimální hodnota intenzity feromonů v obrázku a *diff* je rozdíl mezi maximální a minimální hodnotou intenzity feromonů v obrázku.

3.2 Nenáhodné rozmístění mravenců a jejich reinicializace

S. Gullipalli se ve článku [12] pokusil redukovat šum způsobený inicializací mravenců mimo hrany. Podle jeho tvrzení odebírá prahování kromě šumu i část výsledku a zároveň je zahozena práce několika mravenců, kterou by bylo možno využít i jinak. Jeho nápad byl umístit mravence na místa s vysokou hodnotou $V_c(I_{ij})$ vypočítanou pomocí (3.1).

Výsledkem tohoto přístupu je čistý obrázek bez šumu a bez prahování, na druhou stranu je ve výsledku pouze část hran a především izolované hrany není možné nalézt. Z toho důvodu se pokusil tuto metodu ještě vylepšit přidáním inicializačního kroku po každém cyklu.

V průběhu cyklu je zaznamenáváno množství feromonů umístěného každým mravencem a po globální aktualizaci feromonů je vypočítaný průměrný přírůstek feromonu mravencem τ_{avg}. Před začátkem dalšího cyklu jsou všichni mravenci, kteří umístili méně feromonů než je hodnota τ_{avg}, přemístěni na pixel s nejvyšší hodnotou $V_c(I_{ij})$, na který ještě žádný mravenec nebyl umístěn a je vymazána jejich paměť. Tato metoda pomohla najít více oddělených hran, ale je třeba si dávat pozor na počet cyklů, protože by v tomto případě nadbytek cyklů uškodil stejně tak, jako jejich nedostatek.

Reinicializací se také částečně zabývali ve článku [16], kde byl použit obyčejný AS. V momentě, kdy se měl mravenec rozhodovat, na který pixel přejde, proběhlo ověření, zda nestojí na pixelu, na kterém je část pozadí a pokud stál, ověřily se i všechny okolní pixely. Pokud i na všech okolních pixelech bylo vyobrazeno pozadí, mravenec byl reinicializován na nový náhodně zvolený pixel a již se dále nepohyboval.

3.3 Alternativa k výpočtu $V_c(I_{ij})$

J. Tian, W. Yu a S. Xie ve článku [21] přišli s alternativním výpočtem intenzity jednotlivých pixelů v obrázku.

Jako první si upravíme předpis (3.2), který nám určuje viditelnost pixelu (i, j), podle které určujeme pravděpodobnost, že se mravenec na tento pixel přesune. Nově vypočítáme η_{ij} pomocí (3.4).

$$\eta_{ij} = \frac{V_c(I_{ij})}{Z} \tag{3.4}$$

Všimněme si, že se změna týká jmenovatele zlomku, kde V_{max} bylo nahrazeno Z a $Z = \sum_{i=1:M_1} \sum_{j=1:M_2} V_c(I_{ij})$, kde M_1 je šířka obrázku a M_2 je výška obrázku.

Dále upravíme výpočet $V_c(I_{ij})$. Původně se $V_c(I_{ij})$ počítala pomocí (3.1). Výpočet vycházel z osmiokolí pixelu podle obrázku 3.2a. Nově budeme vycházet z obrázku 3.2b, ze kterého vyvodíme (3.5).

$$\begin{aligned} V_c(I_{i,j}) = f(&|I_{i-2,j-1} - I_{i+2,j+1}| + |I_{i-2,j+1} - I_{i+2,j-1}| \\ &+ |I_{i-1,j-2} - I_{i+1,j+2}| + |I_{i-1,j-1} - I_{i+1,j+1}| \\ &+ |I_{i-1,j} - I_{i+1,j}| + |I_{i-1,j+1} - I_{i-1,j-1}| \\ &+ |I_{i-1,j+2} - I_{i-1,j-2}| + |I_{i,j-1} - I_{i,j+1}|) \end{aligned} \tag{3.5}$$

Zároveň si můžeme všimnout, že je použita funkce $f(\cdot)$, která nám umožní lepší rozložení intenzity. V článku ji definovali jako 3.6.

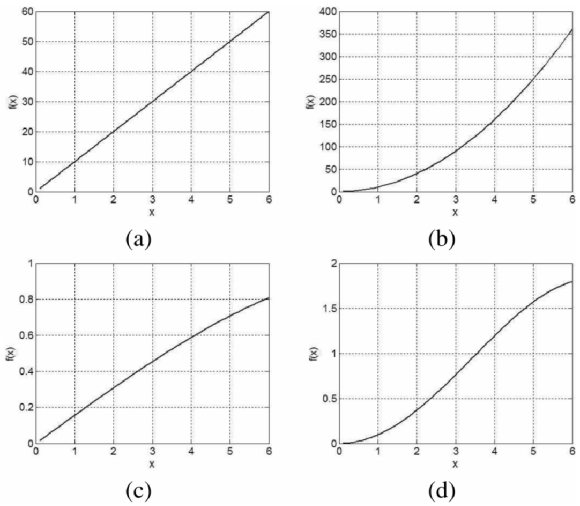
$$f(x) = \lambda x, \quad pro \quad x \geq 0 \tag{3.6a}$$

$$f(x) = \lambda x^2, \quad pro \quad x \geq 0 \tag{3.6b}$$

$$f(x) = \begin{cases} \sin\left(\frac{\pi x}{\lambda}\right) & \text{pokud } 0 \leq x \leq \lambda \\ 0 & \text{jinak} \end{cases} \tag{3.6c}$$

$$f(x) = \begin{cases} \frac{\pi x \sin\left(\frac{\pi x}{\lambda}\right)}{\lambda} & \text{pokud } 0 \leq x \leq \lambda \\ 0 & \text{jinak} \end{cases} \tag{3.6d}$$

kde λ je parametr, který určuje zakřivení funkce $f(\cdot)$, které je zobrazeno na obrázku 3.3.



Obrázek 3.3: Zakřivení funkce $f(\cdot)$ pro $\lambda = 10$. (a) (3.6), (b) (3.6b), (c) (3.6c), (d) (3.6d) [21].

3.4 Optimalizace náročnosti výpočtu

De-Sian Lu a Chien-Chang Chen přišli ve svém článku [15] na několik optimalizačních vylepšení. Hlavní myšlenka spočívá v tom najít koncové body hran a na každém konci inicializovat jednoho mravence. Jelikož se může jednat o stovky mravenců a většina kroků

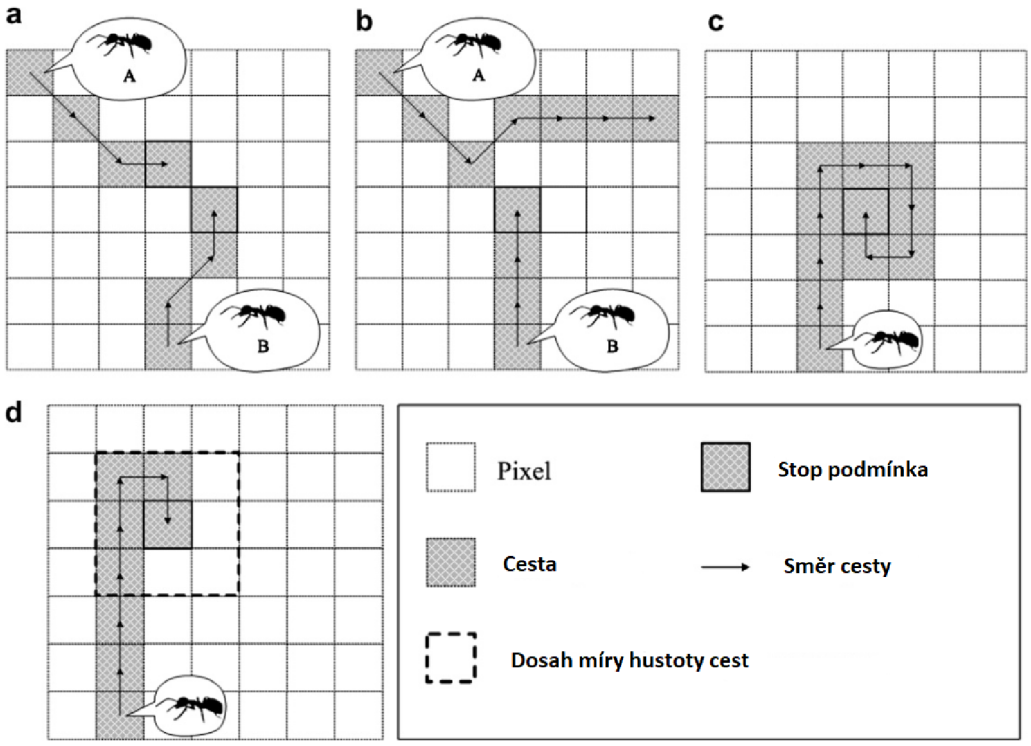
by byla redundantní, přišli s několika kroky, jak hledání optimalizovat a postupně počet mravenců snižovat.

1. *Přímé setkání.* Pokud se setkají dva mravenci, jako na obrázku 3.4a, oba zastaví v momentě, kdy by do sebe dalším krokem narazili.

2. *Dotyk cesty.* Pokud mravenec narazí na cestu, kterou prošel jiný mravenec, jako na obrázku 3.4b, zastaví se.

3. *Přizpůsobivý tabu list.* Pokud mravenec dojde do slepé uličky, protože už na všech okolních pixelech byl, jako na obrázku 3.4c, jeho tabu list se přizpůsobí a dovolí mu projít, aby mohl pokračovat dále v hledání.

4. *Míra hustoty cesty.* Pokud je okolo mravence více cest, než je námi definovaný práh, jako na obrázku 3.4d, zastaví se.



Obrázek 3.4: Optimalizační přístupy pro snížení výpočetní náročnosti. (a)Setkání dvou mravenců. (b) Mravenec došel k cestě vytvořené jiným mravencem. (c) Slepá ulička. (d) Míra hustoty cesty. [15]

Při aplikaci těchto čtyř přístupů byl výsledek zřetelnější a byla dosažena nižší výpočetní náročnost.

Kapitola 4

Detekce hran pomocí Ant Colony System

V této kapitole se zaměříme na můj přístup k řešení problému. Zvolil jsem implementaci v C++ především kvůli rychlosti výpočtu. Při implementaci jsem vycházel z verze od S. Gullipalliho[1], která byla napsána v prostředí Python.

V sekci 3.1 jsme si popsali, jak jsou obecně použity mravenčí algoritmy k detekci hran v obrazu, což tvoří základ mojí implementace, který se dále upravoval pro potřeby jednotlivých experimentů.

4.1 Přiblížení se výsledkům ze článku [12]

Cílem prvního experimentu bylo se přiblížit výsledkům z článku, proto byla použita Gullipalliho implementace tak, jak ji implementoval bez dalších úprav s výjimkou přidání měření času simulace pro porovnání výsledků. Měření začíná před spuštěním simulace a končí po jejím dokončení.

V tomto experimentu jsou porovnány výsledky z mé implementace a z implementace z [12]. Pro porovnání jsem vybral tři sady parametrů vypsané v tabulce 4.1. První (lichý) pokus se sadou parametrů jsou mravenci rozmístěni náhodně a druhý (sudý) pokus s tou samou sadou jsou mravenci rozmístěni na pozice s vysokým $V_c(I_{ij})$.

Parametry jsem volil tak aby byl ve výsledcích vidět rozdíl s větším počtem mravenců, konstrukčních kroků a cyklech. První sada parametrů je převzata z [12].

Jelikož Gullipalli při implementaci (2.4) vybíral vždy pixel s největší pravděpodobností, pro účely experimentu 1 jsem nechal implementaci stejnou. V ostatních experimentech se už mravenci budou řídit pravděpodobností zvolení pixelu danou předpisem.

4.2 Vynechání počátečních iterací při aktualizaci feromonů

Ve druhém experimentu se zaměříme na odstranění šumu v případě, že jsou mravenci rozmístěni náhodně. V tomto případě je šum způsoben mravenci, kteří jsou inicializováni daleko od hrany a musejí ji hledat. Protože za sebou zanechávají i tito mravenci feromonovou stopu, ostatní mravenci jsou touto stoupou ovlivňováni a jejich cesty jsou vidět ve výsledném obrazu. V tomto experimentu si zavedeme nový parametr --skip, který nám dovolí

[1] https://github.com/syamgullipalli/edge-detection-aco

Sada	Sada 1		Sada 2		Sada 3	
Experiment	1.1	1.2	1.3	1.4	1.5	1.6
Počet mravenců	512	512	100	100	1500	1500
Počet kroků	40	40	10	10	40	40
Počet cyklů	10	10	20	20	5	5
Paměť mravenců	8	8	8	8	8	8
α	1	1	1	1	2,5	2,5
β	1	1	1	1	2	2
ρ	0,1	0,1	0,1	0,1	0,1	0,1
ϕ	0,05	0,05	0,05	0,05	0,05	0,05
q0	0,7	0,7	0,7	0,7	0,7	0,7
τ_0	0,1	0,1	0,0001	0,0001	0,0001	0,0001
Náhodné rozmístění	ano	ne	ano	ne	ano	ne

Tabulka 4.1: Nastavení parametrů v experimentu 1

několik prvních iterací neaktualizovat feromonovou stopu žádným mravencem. V případě těchto několika iterací jsou veškeré hodnoty feromonů rovny inicializační hodnotě, proto se mravenci při své cestě řídí pouze funkcí $V_c(I_{ij})$, která by je v ideálním případě měla dovést k hraně, na které by měli zůstat. Po uplynutí daného počtu iterací začnou mravenci vypouštět feromony jako obvykle a stejně tak se jimi budou řídit.

Rozdíl mezi tímto přístupem a přístupem s inicializací mravenců přímo na hrany je ten, že se nám mravenci neshlukují v řadě na jedné hraně a je velká šance, že budou odhaleny i izolované hrany.

Experiment probíhal s parametry popsanými v tabulce 4.2.

Počet mravenců	512
Počet kroků	40
Počet cyklů	15
Paměť mravenců	8
α	1
β	1
ρ	0,1
ϕ	0,05
q0	0,7
τ_0	0,0001
skip	5

Tabulka 4.2: Nastavení parametrů v experimentu 2

Při volbě parametrů jsem převážně vycházel z experimentu 1. Upravil jsem akorát τ_0 na 0,0001, protože se ve většině ostatních experimentů pracovalo s touto hodnotou. Počet cyklů jsem zvýšil o pět, protože nyní prvních pět cyklů využijeme jen pro přesun mravenců na hrany.

4.3 Zavedení prahu pro zobrazení výsledných hran

Ve třetím experimentu budeme také vycházet z náhodného rozmístění mravenců, na jejichž výsledky aplikujeme prahování k redukci šumu. Prahování se provede těsně před zobrazováním výsledků, kde zobrazíme černě jen ty pixely, které mají výslednou hodnotu větší, než je hodnota prahu.

Ve třetím experimentu jsem zvolil parametry podle tabulky 4.3.

Počet mravenců	1200
Počet kroků	40
Počet cyklů	30
Paměť mravenců	8
α	1
β	1,5
ρ	0,15
ϕ	0,5
q0	0,4
τ_0	0,0001
práh	7

Tabulka 4.3: Nastavení parametrů v experimentu 3

Počáteční hladinu feromonů je třeba mít co nejmenší nenulovou hodnotu, aby všechny vzorečky fungovaly správně. Parametr q0 nám zaručí, že se mravenci budou pohybovat spíše podle pravděpodobnostní funkce (2.4), než aby šli po momentálně nejlepší možné cestě. Poměrem parametrů a β jsem dosáhl toho, že mravenci při volbě dalšího pixelu dávají větší váhu viditelnosti oproti intenzitě feromonů. Díky parametru ρ budou rychleji vyprchávat feromony z cestiček mezi hranou a počáteční pozicí mravenců. Velkou změnu udělalo nastavení parametru ϕ, které způsobilo drastické snižování feromonu na hranách, kterými mravenci právě prošli a ve výsledku to způsobilo celkové zvýraznění hran, protože mravenci chodí ve více řadách. Pokud by ale parametr ϕ zůstal na nízké hodnotě, hrany by byly stejně tlustší kvůli nízké hodnotě parametru q0. Parametr ϕ spolu s parametrem q0 navíc vyřešili problém s některými mravenci, kteří se točili kolem místa, kde začali, vytvářeli tak šum a nepřispívali k výsledku. S pamětí mravenců jsem nehýbal, protože jsem neobjevil příznivější nastavení. Práh pro prahování s tímto nastavením zobrazoval nejlepší výsledky v poměru odstranění šumu a neodstranění správných hran. Díky rychlosti výpočtu jsem byl schopen pracovat s 1200 mravenci, se 40 konstrukčními kroky a 30 cykly. V případě jednoduššího obrázku v experimentu 3.2 jsem snížil počet cyklů na 20.

4.4 Změna výpočtu $V_c(I_{ij})$

Ve čtvrtém experimentu vyzkouším vylepšení ze článku [21], které jsou popsány v sekci 3.3. Cílem je porovnat výpočet pomocí klasického osmiokolí a výpočet pomocí šestnáctiokolí znázorněného na obrázku 3.2b. Upravil jsem proto výpočet $V_c(I_{ij})$ podle (3.5).

Experiment jsem pouštěl se stejnými parametry a stejnými obrázky, jako experiment 3, aby bylo jednoduché určit, zda došlo ke zlepšení, či nikoliv. Parametr λ jsem nastavil na hodnotu 10.

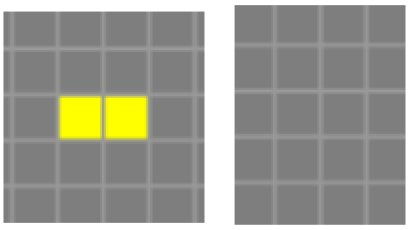
4.5 Zúžení hran pomocí principů Hry Život

Výsledky z experimentu 3 byly velmi přesné. Jedna z mála věcí, co by se dala vytknout, jsou tlusté hrany. V pátém experimentu se na ně zaměříme pomocí principu, který je inspirovaný Conwayovou hrou Život (Conway's Game of Life).

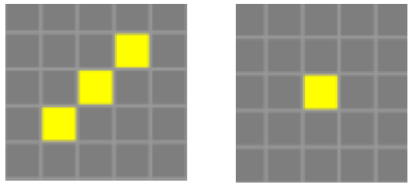
Hra Život byla představena roku 1970 [11]. Jedná se o celulární automat, zkráceně CA, pod kterým si můžeme představit nekonečnou dvourozměrnou mřížku buněk, kde každá buňka může nabývat dvou stavů, žívá či mrtvá. Celá hra se řídí čtyřmi pravidly:

1. Živá buňka, která má méně než dvě sousedící živé buňky zemře.
2. Živá buňka, která má dva nebo tři sousedící živé buňky přežije.
3. Živá buňka, která má více než tři sousedící živé buňky zemře.
4. Mrtvá buňka, která má přesně tři sousedící živé buňky ožije.

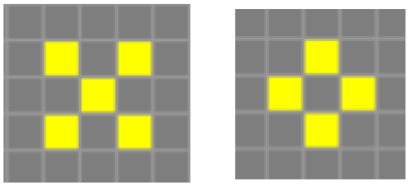
Tato čtyři jednoduchá pravidla nám dokáží zaručit Turingovu úplnost [18], což znamená, že jsme schopni pomocí hry Život simulovat Turingův stroj. Grafické znázornění pravidel můžeme vidět na obrázku 4.1.



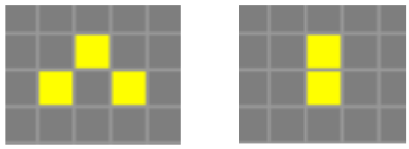
(a) Živá buňka, která má méně než dvě sousedící živé buňky zemře.



(b) Živá buňka, která má dva nebo tři sousedící živé buňky přežije.



(c) Živá buňka, která má více než tři sousedící živé buňky zemře.



(d) Mrtvá buňka, která má přesně tři sousedící živé buňky ožije.

Obrázek 4.1: Ukázka pravidel hry Život

Hru Život zmiňuji, protože jsem se v tomto experimentu inspiroval jejími pravidly, konkrétně pravidlem 1 a 3. Výsledný obrázek je také dvojrozměrná mřížka buněk, kde každá nabývá dvou stavů, buď 0 (černá), nebo 255 (bílá).

S podmínkami úmrtí jsem různě experimentoval a ukázalo se, že jakékoliv nízké číslo zvyšuje šum kolem hran, proto jsem nakonec zvolil pravidlo takové, že přežije jen buňka, která má více než šest živých sousedících buněk.

4.6 Technická specifikace implementace

Můj program jako první zpracuje parametry zadané pomocí příkazové řádky. Parametry mohou být zadané v dlouhé (--image) či krátké (-i) formě. K parsování parametrů využívám knihovnu cxxopts[2].

```
1  void Parse(int argc, char** argv, Parameters &parameters)
2  {
3    cxxopts::Options options("ACO_edge_detection",
4      "ACO for edge detection - bachelor thesis");
5    options.add_options()
6      ("i,image", "Path to an image to be processed",
7        cxxopts::value<std::string>()->default_value(
8        "./standard_test_images/256/lena_color.tif"))
9      ...
10     ("h,threshold", "Threshold for image rendering",
11       cxxopts::value<int>()->default_value("0"))
12     ;
13   auto parsed = options.parse(argc, argv);
14   parameters.imagePath = parsed["image"].as<std::string>();
15   ...
16   parameters.threshold = parsed["threshold"].as<int>();
17 }
```

Při zpracování parametrů si nejdříve vytvoříme třídu Options, do které vložíme všechny požadované parametry s výchozí hodnotou. Poté zavoláme na třídě Options metodu parse, která v parametru bere počet argumentů a jejich hodnoty (argc, argv), které jsou k dispozici od spuštění programu. Metoda parse nám vrátí třídu ParseResult, která obsahuje mapu všech požadovaných parametrů, ke kterým lze přistoupit pomocí jména a lze je uložit do proměnných.

Seznam parametrů, se kterými program pracuje:

- *--image, -i.* Datový typ string, kde je uložena relativní či absolutní cesta k obrázku. Výchozí hodnota je nastavena na ./standard_test_images/256/lena_color.tif.
- *--alpha, -a.* Datový typ float, kde je uložen parametr α pro pozdější výpočty. Výchozí hodnota je nastavena na 1.
- *--beta, -b.* Datový typ float, kde je uložen parametr β pro pozdější výpočty. Výchozí hodnota je nastavena na 2.
- *--ants, -n.* Datový typ int, kde je uložen počet mravenců, kteří budou inicializování. Výchozí hodnota je nastavena na 250.
- *--rho, -r.* Datový typ float, kde je uložen koeficient evaporace feromonů, který je použit při každé globální aktualizaci. Jeho hodnota by měla být v intervalu $\rho \in (0, 1\rangle$. Výchozí hodnota je nastavena na 0.1.

[2] https://github.com/jarro2783/cxxopts

- *--phi, -p.* Datový typ float, kde je uložen koeficient úbytku feromonů při průchodu mravence hranou a lokální aktualizaci feromonů. Jeho hodnota by měla být v intervalu $\phi \in (0, 1\rangle$. Výchozí hodnota je nastavena na 0.05.
- *--cons, -k.* Datový typ int, kde je uložen počet počet konstrukčních kroků v každé iteraci. Výchozí hodnota je nastavena na 40.
- *--iter, -l.* Datový typ int, kde je uložen počet iterací. Výchozí hodnota je nastavena na 10.
- *--q0, -q.* Datový typ float, kde je uložen koeficient průzkumu. Jeho hodnota by měla být v intervalu $\rho \in (0, 1\rangle$. Výchozí hodnota je nastavena na 0.4.
- *--tauini, -t.* Datový typ float, kde je uložena počáteční hodnota feromonů, která bude aplikována na každý pixel. Výchozí hodnota je nastavena na 0.01.
- *--mem, -m.* Datový typ int, kde je uložen počet předchozích kroků, které si každý mravenec bude pamatovat. Výchozí hodnota je nastavena na 8.
- *--debug, -d.* Datový typ bool. Při jeho nastavení na konci simulace program čeká na stisknutí klávesy pro ukončení simulace a seed pro generování pseudonáhodných čísel nastaví na 1, čímž je zaručeno, že opakování pokusu se stejnými parametry dosáhneme stejného výsledku.
- *--skip, -s.* Datový typ int, kde je uložen počet iterací. Číslo nám určuje, kolik prvních iterací se nebudou aktualizovat feromony. Výchozí hodnota je nastavena na 0.
- *--random.* Datový typ bool. Při jeho nastavení se mravenci inicializují na náhodné pozice místo na pozice určené heuristikou. Všimněme si, že tento parametr jako jediný nemá krátkou formu.
- *--threshold, -h.* Datový typ int, kde je uložen práh, který je použit při prahování při vykreslování výsledků. Výchozí hodnota je nastavena na 0.
- *--lambda, -f.* Datový typ int, kde je uložen parametr λ, který je použit při výpočtu heuristické hodnoty pomocí předpisu 3.6. Výchozí hodnota je nastavena na 10.

Po zpracování parametrů přijde na řadu zpracování obrázku. K práci s obrázky využívám knihovnu OpenCV2[3]. Obrázek nejdřív načtu z cesty zadané parametrem, poté ho převedu na šedotónový. Dále si vytvořím dva 2D vektory, do jednoho si uložím intenzitu každého pixelu a do druhého heuristiku, kterou počítám pomocí předpisu 3.1.

```
1 void ParseIntensity(Parameters &parameters)
2 {
3   Mat RGBImage = imread(parameters.imagePath);
4   Mat grayScaleImage(RGBImage.size(), CV_8UC1);
5   cvtColor(RGBImage, grayScaleImage, COLOR_RGB2GRAY);
6   parameters.intensity = vector<vector<int>>(grayScaleImage.cols,
7     vector<int>(grayScaleImage.rows));
8   parameters.heuristic = vector<vector<int>>(grayScaleImage.cols,
```

[3] https://opencv.org/

```
    vector<int>(grayScaleImage.rows));
  for (int i = 0; i < grayScaleImage.cols; i++)
  {
    for (int j = 0; j < grayScaleImage.rows; j++)
    {
      parameters.intensity[i][j] =
        int(grayScaleImage.at<uchar>(j, i));
      parameters.heuristic[i][j] =
        CalculateHeuristic(grayScaleImage, j, i);
    }
  }
}
```

CV_8UC1 značí, že se obrázek načte jako osmibitový jednokanálový neznaménkový char. Šedotónový obraz získáme metodou cvtColor s parametrem COLOR_RGB2GRAY.

Následně se inicializuje simulace, při které proběhne několik kroků.

1. Inicializace 2D vektoru feromonů, který je naplněn hodnotou danou parametrem tauini a který nám slouží k uchování feromonových cestiček.

2. Inicializace 2D vektoru, do kterého ukládáme intenzitu pixelů, abychom mohli zobrazit výsledek.

3. Inicializujeme mravence buď na náhodné pozice, nebo na pozice s největší heuristickou hodnotou.

4. Označíme startovní pozice mravenců na obrázku pro lepší představu o výsledku.

Nyní přichází na řadu samotná simulace, která probíhá podle algoritmu popsaném v algoritmu 1. Pro lepší představu přikládám i zdrojový kód níže.

```
void AntColonySystem::Run()
{
  std::cout << m_parameters.imagePath << std::endl;
  for (int i = 0; i < m_parameters.iterations; i++)
  {
    for (int j = 0; j < m_parameters.constructionSteps; j++)
    {
      for (auto& ant : ants)
      {
        Move(*ant);
        UpdateLocalPheromone(*ant);
      }
    }
    UpdateGlobalPheromone();
    DisplayResults(i+1);
    ResetAntsPheromone();
    cout << "Iteration " << i + 1 << " ended." << endl;
  }
  if(m_parameters.debug) cv::waitKey(0);
```

```
20    else cv::waitKey(500);
21  }
```

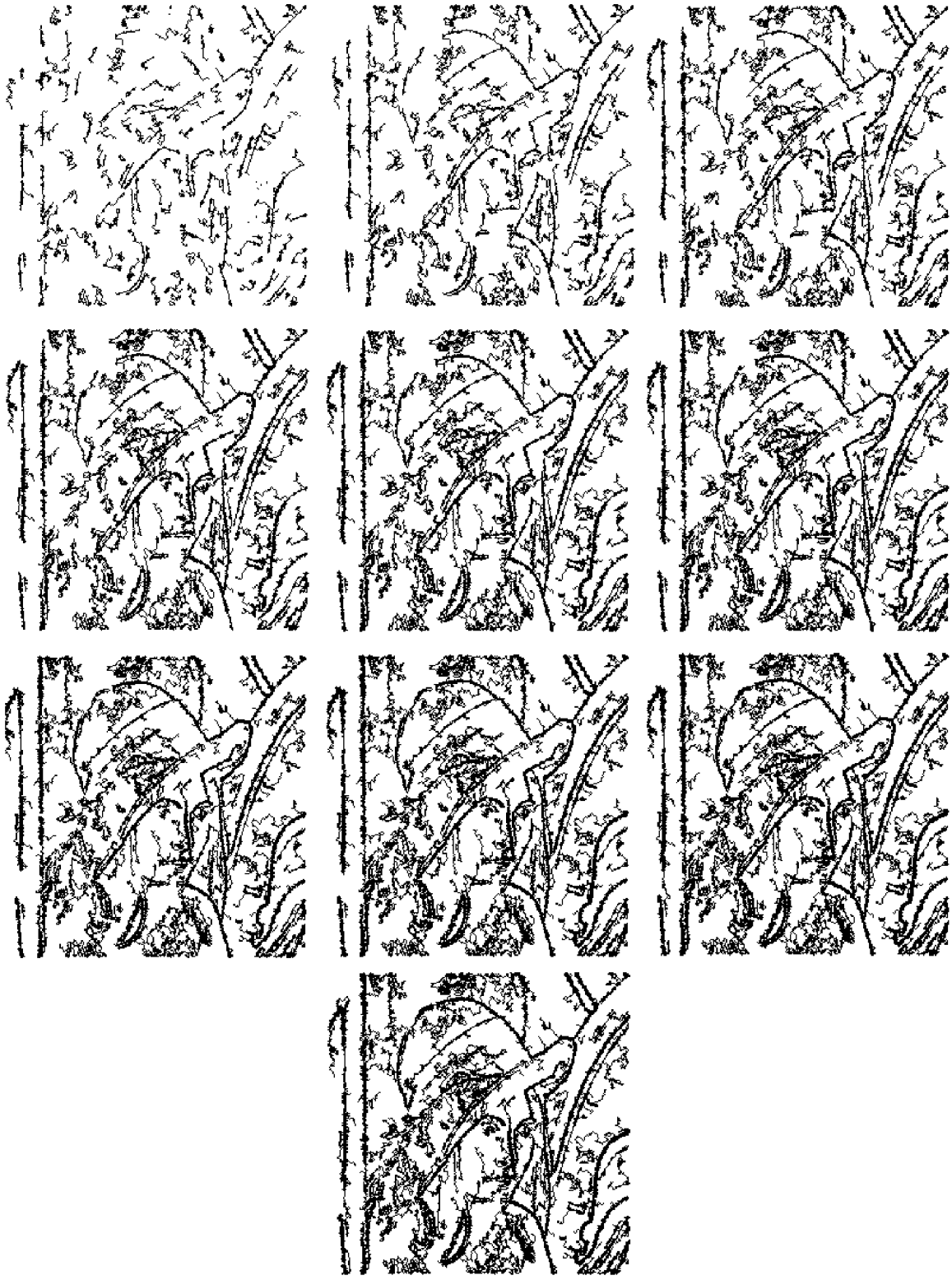
Při pohybu mravence si nejdříve uložíme všechny sousední pixely, které mravenec může navštívit, tj. nejsou v jeho paměti a nenachází se mimo hranice obrázku, u kterých si poznamenáme heuristickou hodnotu a hodnotu feromonů. Poté vygenerujeme pseudonáhodné číslo q, které nám určí kritéria pro volbu cesty podle 2.5. Po zvolení cílového města se do něj mravenec přesune a uloží si jeho souřadnice do paměti. Poté se aplikuje lokální aktualizační pravidlo 2.8 na pixel, na kterém mravenec momentálně stojí.

Po všech konstrukčních krocích, ve kterých se pohnou všichni mravenci, následuje globální aktualizace feromonů. Nejdříve se pro každého mravence vypočítá průměrná heuristika na jeho cestě a poté se na všech navštívených pixelech mravenci v této iteraci aplikuje globální aktualizační pravidlo podle předpisu 2.6. Nakonec se na všech pixelech, kde danou iteraci žádný mravenec neprošel, aplikuje vypařování feromonů.

Jako poslední akce v iteraci je zobrazení průběžných výsledků, která je znázorněna kódem níže. Nejdříve si zjistíme minimální *min* a maximální *max* hodnotu feromonů, kterou od sebe odečteme a získáme rozdíl *diff*. Poté už jen přidělíme každému pixelu intenzitu podle vzorce (3.3) a výsledek zobrazíme.

```
1   void AntColonySystem::DisplayResults(int iter)
2   {
3     ...
4     for (tuple<int, int> position : alreadyVisitedPositions)
5     {
6       m_parameters.edges[std::get<0>(position)][std::get<1>(position)] =
7             int(round(abs((GetPheromone(position) - min) / diff) * 255));
8     }
9     cv::Mat img = cv::Mat(maxHeight, maxWidth, CV_8UC1);
10    for (int i = 0; i < maxWidth; i++)
11    {
12      for (int j = 0; j < maxHeight; j++)
13          {
14            if(m_parameters.edges[i][j] > float(m_parameters.threshold))
15              img.at<uchar>(j, i) = 0;
16            else
17              img.at<uchar>(j, i) = 255;
18          }
19    }
20    cv::namedWindow("Edges", cv::WINDOW_AUTOSIZE);
21    imshow("Edges", img);
22    cv::imwrite( "RESULTS/Iteration" + std::to_string(iter) + ".png", img);
23    cv::waitKey(500);
24  }
```

Celá simulace tedy probíhá v několika konstrukčních krocích, které každý cyklus vytvoří část řešení a až poslední iterace ukáže výsledek. Jak vypadají průběžné výsledky je zobrazeno na obrázku 4.2



Obrázek 4.2: Postupné získávání výsledků každým cyklem. Každý obrázek reprezentuje průběžný výsledek po každém cyklu, poslední obrázek je konečný výsledek.

Kapitola 5

Experimentální výsledky

V této kapitole se zaměříme na experimenty, které jsem dělal v průběhu této bakalářské práce. Experimenty jsem prováděl na sadě čtyř obrázků zobrazených v obrázku 5.1.

(a) Lena

(b) Radiace

(c) Papriky

(d) Mandril

Obrázek 5.1: Testovací sada obrázků

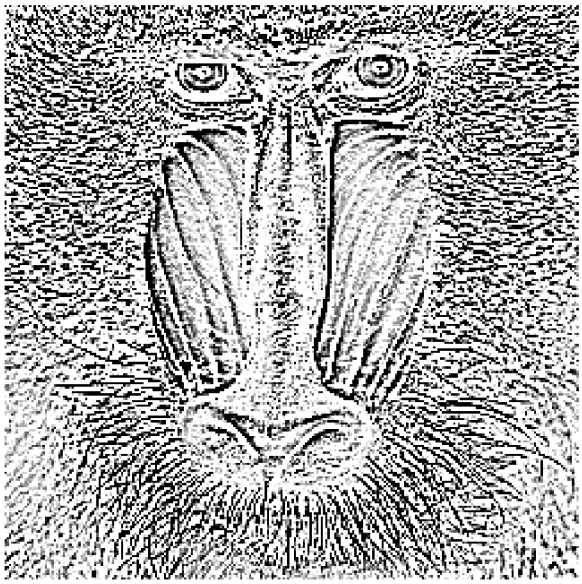
Obrázek Leny se nachází téměř ve všech článcích. Do své testovací sady jsem ho přidal, abych mohl porovnávat své výsledky s výsledky provedenými ostatními.

Na obrázku radiace se jednoduše hledaly správné parametry. Protože se zde nachází jen malé množství hran, mohl jsem na něm krásně odpozorovat, jak se mravenci chovají

v místě, kde se jsou všechny sousední uzly stejné a jak se chovají v momentě, kdy už narazí na hranu. Přestože by se mohlo zdát, že u tak jednoduchého obrázku není problém hrany detekovat, ve svých experimentech jsem se ve většině případů nakonec potýkal s nejvíce problémy právě u tohoto obrázku.

Obrázek paprik jsem vybral z toho důvodu, že se v něm hrany všemožně větví a spojují, což mohlo mravencům způsobit problémy z hlediska preference jedné či druhé cesty. Stejně tak můžeme na paprikách vidět několik odlesků světla, které tvoří další izolované cyklické hrany, které by také mohl být problém mravenci odhalit.

Jako poslední máme obrázek opice z rodu Mandril, kterou jsem vybral kvůli její srsti. Jak můžeme vidět na obrázku 5.2, nachází se na ní obrovské množství hran, které jsou pro mravence obrovskou výzvou. Zároveň má izolované oči a velmi výrazný obličej.



Obrázek 5.2: Čtvrtý testovací obrázek po detekci hran pomocí Laplaceovi metody s maticí 5x5

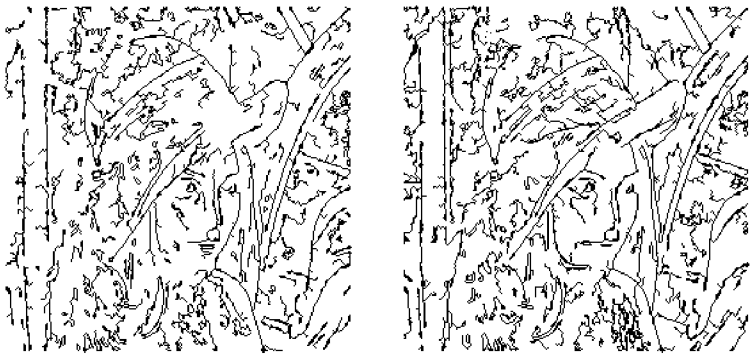
5.1 Přiblížení se výsledkům ze článku [12]

Tato sekce shrnuje výsledky experimentu 1, popsaném v sekci 4.1. Výsledky experimentů jsou zobrazeny na obrázku 5.3 a rychlost výpočtu v tabulce 5.1.

Sada	Sada 1		Sada 2		Sada 3	
Experiment	1.1	1.2	1.3	1.4	1.5	1.6
Implementace z článku [s]	1827	314	40	19	1761	461
Moje implementace [s]	8	7	11	11	8	5

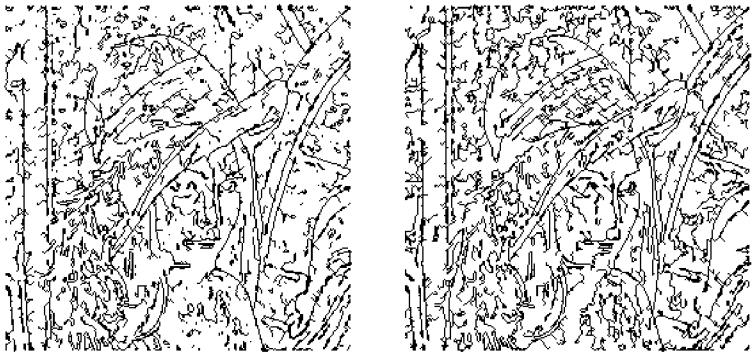
Tabulka 5.1: Čas běhu simulace v experimentu 1

Jak můžeme vidět, grafické výsledky jsou velmi podobné výsledkům z článku. Neuvedl jsem zde druhou sadu parametrů, protože z důvodu nízkého počtu mravenců a konstrukčních kroků nebyl výsledek nijak uspokojivý. Když se podíváme na tabulku 5.1, vidíme, že je má implementace značně rychlejší, což mi umožní efektivně pracovat s více mravenci, nebo je



(a) Sada 1, experiment 1.1

(b) Sada 1, experiment 1.2



(c) Sada 3, experiment 1.5

(d) Sada 4, experiment 1.6

Obrázek 5.3: Experiment 1. Porovnání mých výsledků (vlevo) s výsledky z článku (vpravo)

nechat pracovat delší dobu. Dosažené výsledky byly měřeny s implementací v C++, která vznikla v rámci této práce vůči implementaci v prostředí Python [12].

Pokud se zaměříme na experimenty z druhé sady, zjistíme, že čas potřebný k výpočtu mojí implementací vzrostl, ale čas který byl potřeba při implementaci z článku značně poklesl. Nárůst je způsoben větším počtem iterací a faktem, že se po každé generaci vykresluje průběžný výsledek, který zůstane zobrazený 500ms, což nám při 20 generacích dohromady zabere 10 vteřin. Drastický pokles časové náročnosti algoritmu z článku je způsoben menším počtem mravenců a konstrukčních kroků.

Z prvního experimentu byl patrný rozdíl mezi inicializací mravenců na náhodné pozice a na pozice s vysokou mírou heuristiky. Pokud jsou mravenci inicializováni na náhodné pozice, vyskytuje se v obrázku velké množství šumu v podobě „falešných hran“, které vznikají mezi místem, kde mravenec začínal a první skutečnou hranou, ke které dojde. Pokud jsou ovšem mravenci inicializováni na místa, kde je vysoká heuristická hodnota. Zbavíme se šumu, ale neodhalíme izolované hrany v obrazu.

5.2 Vynechání počátečních iterací při aktualizaci feromonů

Tato sekce shrnuje výsledky experimentu 2, popsaném v sekci 4.2. Výsledky experimentu jsou viditelné na obrázku 5.4.

Z výsledků je patrné, že došlo k výraznému zlepšení. Na obrázcích není tolik falešných hran, ale stále se tam nějaké nachází. Na výsledných obrázcích chybí plno detailů nebo nejsou zcela zřejmé. Na výsledku je z obrázku Leny vidět, že oproti výsledkům z experimentů 1.1 a 1.5 je šum značně redukován, ale zároveň oproti experimentům 1.2 a 1.6 je na obrázku mnohem více nalezených hran, jako je trám vlevo, celý obličej včetně brady a stuha na klobouku.

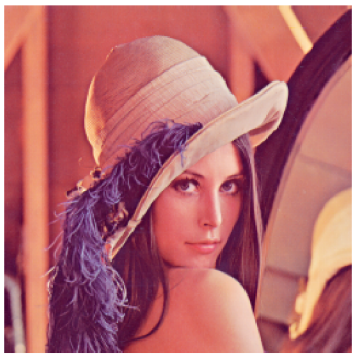
Jelikož se v experimentu 2.2, kde je vyobrazen obrázek radiace, jednalo o jednoduchý obrázek, je na něm zřetelný zbývající šum kolem hran. Ten je pravděpodobně způsobený parametrem q0, který je nastavený tak, že ve 30 % případů dojde k určení cesty mravence podle pravděpodobnosti, která já vypočítána předpisem (2.4). Z výsledku můžeme vidět, že v případě, kdy se vybere nesprávná cesta několikrát v řadě, může dojít k vychýlení mravence od hrany, kterou už pak nemusí být schopen znovu najít. Pokud by se ale mravenec řídil čistě jen aktuálně nejlepší cestou, mohl by se začít točit v kruhu kdekoliv po obrázku, proto je potřeba nějakou náhodnost zanechat.

V experimentu 2.4, kde je vyobrazen obrázek Mandrila, se nám ukázalo, že tento přístup není ideální v obrázku s velkým počtem hran, kde jsou mravenci pravděpodobně již od začátku poblíž nějaké hrany a je zde potřeba klást větší důraz na nalezení všech hran, než na redukci šumu.

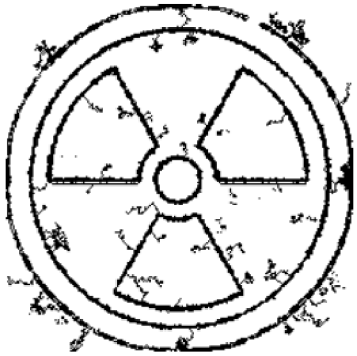
5.3 Zavedení prahu pro zobrazení výsledných hran

Tato sekce shrnuje výsledky experimentu 3, popsaném v sekci 4.3. Výsledky třetího experimentu jsou znázorněny na obrázku 5.5.

Z výsledků je zřejmé, že prahování odvedlo skvělou práci v redukci šumu. Obrázek je téměř čistý, bez stop mezi místem, kde mravenci započali cestu a hranou. Pokud se zaměříme na izolované hrany, můžeme si všimnout, že i ty byly nalezeny. Zřejmé je to na obrázku paprik z experimentu 3.3, ve kterém se nachází množství odlesků, které byly mravenci zaznamenány a korektně zobrazeny.

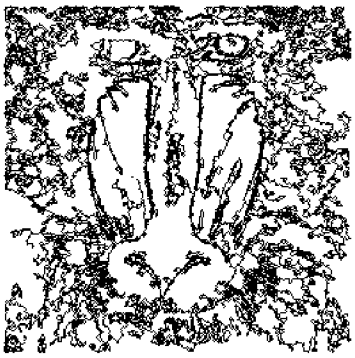



(a) Experiment 2.1 - Lena

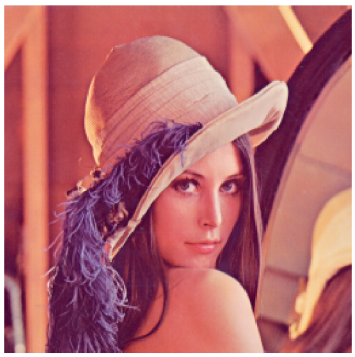
(b) Experiment 2.2 - Radiace


(c) Experiment 2.3 - Papriky

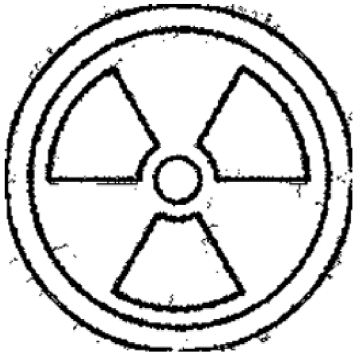



(d) Experiment 2.4 - Mandril

Obrázek 5.4: Experiment 2. Původní obrázek vlevo, výsledky experimentu vpravo.




(a) Experiment 3.1 - Lena

(b) Experiment 3.2 - Radiace


(c) Experiment 3.3 - Papriky


(d) Experiment 3.4 - Mandril

Obrázek 5.5: Experiment 3. Původní obrázek vlevo, výsledky po prahování vpravo.

Velké množství mravenců a iterací si ovšem vyžádalo svou daň z hlediska časové náročnosti, kterou můžeme vidět v tabulce 5.2. Můžeme v ní krásně vidět závislost počtu hran na náročnosti. V jednoduchém obrázku radiace využitém v experimentu 3.2 bylo jen pár hran a o třetinu iterací méně a běh simulace trval více než o polovinu času méně. Naopak, když se podíváme na výsledky z obrázku Mandrila z experimentu 3.4, kde byl obrovský počet hran kvůli srsti, časová náročnost byla značně větší.

Experiment	3.1	3.2	3.3	3.4
Potřebný čas [s]	67	36	64	79

Tabulka 5.2: Čas běhu simulace v experimentu 3

Také bych chtěl upozornit na fakt, že jsem na všechny obrázky použil stejné parametry, které nalezly řešení na všech obrázcích. Troufnu si proto tvrdit, že by se mohlo jednat o optimální parametry v případě tohoto přístupu.

5.4 Změna výpočtu $V_c(I_{ij})$

Tato sekce shrnuje výsledky experimentu 4, popsaném v sekci 4.4. Výsledky si můžeme prohlédnout na obrázku 5.6.

Výsledky čtvrtého experimentu jsou velmi podobné výsledkům z experimentu třetího. Při porovnání výsledků s výsledky ze třetího experimentu je patrné, že došlo k rozšíření hran a k redukci šumu.

5.5 Zúžení hran pomocí principů Hry Život

Tato sekce shrnuje výsledky experimentu 5, popsaném v sekci 4.5. Výsledky experimentu jsou zobrazeny na obrázku 5.7.

Cílem toho experimentu bylo zúžit detekované hrany, což se povedlo. Byl také redukován veškerý šum, což je znát především na obrázku radiace z experimentu 5.2. Budeme-li výsledky posuzovat z pohledu zachování detailů a celkové kvality, můžeme konstatovat, že v tomto případě došlo ke zhoršení, jelikož méně výrazné hrany nebyly ve většině případů zachovány.

5.6 Celkové zhodnocení

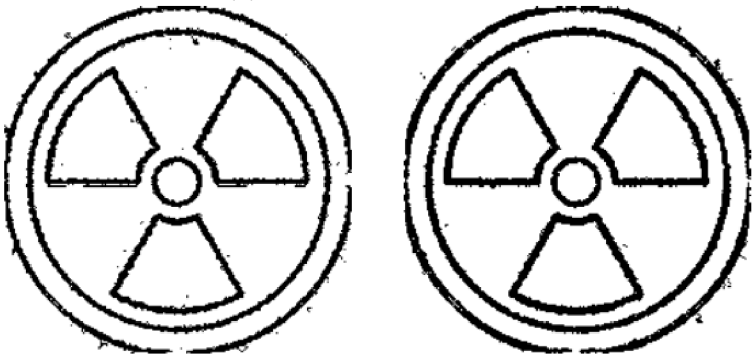
Jak je patrno z výše uvedených výsledků, mravenčí algoritmy poskytují platformu pro možnost detekce hran v obraze s širokým spektrem nastavení. Výše uvedené výsledky ukázaly, že různá nastavení do značné míry ovlivňují celkovou kvalitu výsledků.

V prvním experimentu, který uvažoval pouze základní nastavení ACS jak bylo uvedeno v původní publikaci [12], lze pozorovat mnoho falešných hran. Další rozšíření, která byla navržena v této práci, umožňují kvalitu detekce hran výrazně zlepšit. Obrázek 5.4, kde bylo vynecháním počátečních cyklů mravenců z výsledku dosaženo značné eliminace falešných hran.

Další rozšíření, kterými bylo zavedení prahování a alternativní výpočty inzenzit v obraze, přispěly ke zkvalitnění výsledků zejména v podobě viditelného rozšíření hran se zachováním jejich kvality. Toto je především vidět na obrázcích 5.6 vůči obrázkům 5.4.

Snahou redukce tloušťky hran byl poslední experiment využívající vybraných principů Hry Život, který potvrdil, že tloušťka hran může být touto technikou značně zredukována, jak je vidět na obrázku 5.7. Celková kvalita těchto výsledků závisí také na úhlu pohledu (např. je zde velmi nízký počet falešných hran, naopak některé nevýrazné hrany zmizely).

(a) Experiment 4.1 - Lena



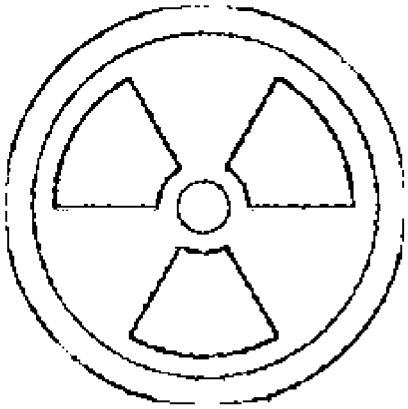
(b) Experiment 4.2 - Radiace

(c) Experiment 4.3 - Papriky

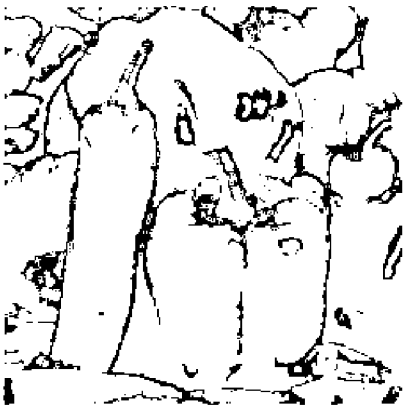
(d) Experiment 4.4 - Mandril

Obrázek 5.6: Experiment 4. Výsledky z experimentu 3 vlevo, výsledky po úpravě výpočtu heuristiky vpravo

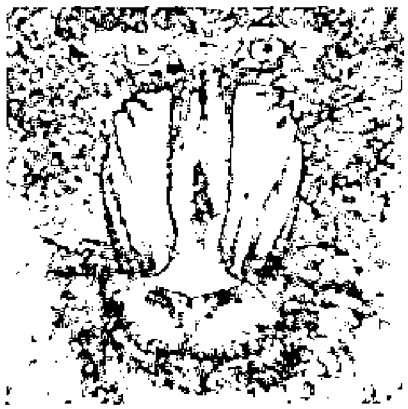
(a) Experiment 5.1 - Lena



(b) Experiment 5.2 - Radiace



(c) Experiment 5.3 - Papriky



(d) Experiment 5.4 - Mandril

Obrázek 5.7: Experiment 5.

Kapitola 6

Závěr

Cílem této bakalářské práce bylo implementovat program na detekci hran pomocí mravenčích algoritmů, vylepšit stávající řešení z práce S. A. Gullipalli [12] a přijít na univerzální přístup, který by bylo možné využít pro detekci hran ve většině obrazech.

Ve snaze přijít s co nejlepšími výsledky bylo potřeba se vcítit do problematiky, odhalit nedokonalosti v současných přístupech a přijít s nápady, co by se dalo zlepšit bylo třeba přečíst několik publikací, které se touto problematikou také zabývaly.

Z počátku jsem v této práci obecně nastínil, co jsou vlastně mravenčí algoritmy, jak vznikly, jak fungují a na co jsou původně určeny. Dále jsem podrobněji rozebral několik variant mravenčích algoritmů, především algoritmus Ant Colony System, který byl základem pro moji implementaci při řešení problému detekce hran.

Dále jsem vyzdvihl některé publikace, ze kterých jsem se inspiroval, krátce zmínil jak jejich autoři k danému problému přistupovali a čím konkrétně mě právě tyto publikace zaujaly. Později jsem v kapitole rozebral i jinou možnost využití mravenčích algoritmů v metodách rozpoznání obrazu, konkrétně segmentaci obrazu, což je také zajímavé téma s různými možnostmi využití.

V další kapitole jsem vysvětlil, jak jsem problém detekce hran v obrazu řešil ve své implementaci já. Celý algoritmus je tam podrobně popsán, jsou zde vyzdviženy zajímavé části kódu a popsána funkce jednotlivých parametrů, které se v mých experimentech vyskytují.

Poslední kapitola je zaměřena na samotné experimenty. Jsou zde popsány myšlenky, které mě ke každému z nich vedly, podrobný popis toho, na co se daný experiment zaměřuje a jakým způsobem se toho snažím dosáhnout. Dále jsou zobrazeny výsledky experimentu spolu s jejich rozborem. V prvním experimentu jsem se pokusil napodobit výsledky z implementace [12]. Ve druhém a třetím experimentu byla snaha tyto výsledky zlepšit, což se povedlo v obou případech a ve třetím experimentu jsem dokonce přišel s metodou a parametry, které se zdají být optimální pro většinu obrazů za cenou vyšší výpočetní náročnosti. Ve čtvrtém a pátém experimentu jsem se pak snažil vylepšit už tak dobré výsledky ze třetího experimentu. Při mém experimentování jsem vyzkoušel více různých nastavení a metod, ale těchto pět mi přišlo nejzajímavějších, proto jsem je vybral a zde popsal.

Přestože jsou výsledky práce velmi pěkné, stále je zde místo pro zlepšení. Z uvedených výsledků je patrné, kde se hrany v obraze nachází a jsou vyobrazeny i některé detaily. Bohužel jsou hrany tlustší, než by mohly být a při obrazech s vyšším rozlišením je třeba použít více mravenců, což by mělo dopad na výslednou rychlost výpočtu. Co se týče rychlosti, bylo by možné aplikaci optimalizovat rozdělením na více vláken například rozdělením obrazu na více částí, poté by bylo pochopitelně možné použít větší množství mravenců.

Tato práce je důkazem toho, že detekce hran pomocí mravenčích algoritmů je jedno z možných řešení tohoto problému, které dokáže přijít se správným přístupem k velice zajímavým výsledkům. Nevýhodou tohoto přístupu je náročnost správného nastavení všech parametrů, které se navíc mění s typem a velikostí obrazu samotného. Další nevýhodou je nedeterministický výsledek, který se při každém spuštění se stejnými parametry mění, protože v této metodě hraje velkou roli generování pseudonáhodných čísel.

Literatura

[1] Baterina, A. V. a Oppus, C. Image edge detection using ant colony optimization. *WSEAS Transactions on Signal Processing*. World Scientific and Engineering Academy and Society (WSEAS) Stevens Point .... 2010, roč. 6, č. 2, s. 58–67.

[2] Bosco, G. L. A genetic algorithm for image segmentation. In: IEEE. *Proceedings 11th international conference on image analysis and processing*. 2001, s. 262–266.

[3] Cinsdikici, M. G. a Aydin, D. Detection of blood vessels in ophthalmoscope images using MF/ant (matched filter/ant colony) algorithm. *Computer methods and programs in biomedicine*. Elsevier. 2009, roč. 96, č. 2, s. 85–95.

[4] Deneubourg, J.-L., Aron, S., Goss, S. a Pasteels, J. M. The self-organizing exploratory pattern of the argentine ant. *Journal of insect behavior*. Springer. 1990, roč. 3, č. 2, s. 159–168.

[5] Dorigo, M. Optimization, learning and natural algorithms [Ph. D. thesis]. *Politecnico di Milano, Italy*. 1992.

[6] Dorigo, M., Birattari, M. a Stutzle, T. Ant colony optimization. *IEEE computational intelligence magazine*. IEEE. 2006, roč. 1, č. 4, s. 28–39.

[7] Dorigo, M. a Gambardella, L. M. Ant colonies for the travelling salesman problem. *Biosystems*. Elsevier. 1997, roč. 43, č. 2, s. 73–81.

[8] Dorigo, M. a Gambardella, L. M. Ant colony system: a cooperative learning approach to the traveling salesman problem. *IEEE Transactions on evolutionary computation*. IEEE. 1997, roč. 1, č. 1, s. 53–66.

[9] Dorigo, M., Maniezzo, V. a Colorni, A. Ant system: optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*. IEEE. 1996, roč. 26, č. 1, s. 29–41.

[10] Dorigo, M. a Thomas, S. *Ant colony optimization*. MIT Press, 2004.

[11] Gardener, M. MATHEMATICAL GAMES: The fantastic combinations of John Conway's new solitaire game"life,". *Scientific American*. 1970, roč. 223, s. 120–123.

[12] Gullipalli, S. Search Improvement Based on Ants Performance in Image Edge Detection using ACO. 2015. Dostupné z: https://github.com/syamgullipalli/edge-detection-aco.

[13] Kumar, S., Parhi, D. R., Muni, M. K. a Pandey, K. K. Optimal path search and control of mobile robot using hybridized sine-cosine algorithm and ant colony optimization technique. *Industrial Robot: the international journal of robotics research and application*. Emerald Publishing Limited. 2020.

[14] Lee, M.-E., Kim, S.-H., Cho, W.-H., Park, S.-Y. a Lim, J.-S. Segmentation of brain MR images using an ant colony optimization algorithm. In: IEEE. *2009 Ninth IEEE International Conference on Bioinformatics and Bioengineering*. 2009, s. 366–369.

[15] Lu, D.-S. a Chen, C.-C. Edge detection improvement by ant colony optimization. *Pattern Recognition Letters*. Elsevier. 2008, roč. 29, č. 4, s. 416–425.

[16] Nezamabadi Pour, H., Saryazdi, S. a Rashedi, E. Edge detection using ant algorithms. *Soft Computing*. Springer. 2006, roč. 10, č. 7, s. 623–628.

[17] Otsu, N. A threshold selection method from gray-level histograms. *IEEE transactions on systems, man, and cybernetics*. IEEE. 1979, roč. 9, č. 1, s. 62–66.

[18] Rendell, P. A universal turing machine in conway's game of life. In: IEEE. *2011 International Conference on High Performance Computing & Simulation*. 2011, s. 764–772.

[19] Stutzle, T. a Hoos, H. MAX-MIN ant system and local search for the traveling salesman problem. In: IEEE. *Proceedings of 1997 IEEE international conference on evolutionary computation (ICEC'97)*. 1997, s. 309–314.

[20] Stutzle, T. a Hoos, H. H. MAX–MIN ant system. *Future generation computer systems*. Elsevier. 2000, roč. 16, č. 8, s. 889–914.

[21] Tian, J., Yu, W. a Xie, S. An ant colony optimization algorithm for image edge detection. In: IEEE. *2008 IEEE congress on evolutionary computation (IEEE world congress on computational intelligence)*. 2008, s. 751–756.

[22] Venkatesan, S. a Madane, S. S. R. Face recognition system with genetic algorithm and ANT colony optimization. *International Journal of Innovation, Management and Technology*. IACSIT Press. 2010, roč. 1, č. 5, s. 469.

[23] Wang, X., Shi, H. a Zhang, C. Path Planning for Intelligent Parking System Based on Improved Ant Colony Optimization. *IEEE Access*. IEEE. 2020, roč. 8, s. 65267–65273.

[24] Zhang, H., Tu, Y. a Guo, Y. Fidelity-Based Ant Colony Optimization for Control of Quantum System. In: *2017 International Conference on Smart Grid and Electrical Automation (ICSGEA)*. 2017, s. 239–242.

[25] Zhao, B., Zhu, Z., Mao, E. a Song, Z. Image segmentation based on ant colony optimization and K-means clustering. In: IEEE. *2007 IEEE International Conference on Automation and Logistics*. 2007, s. 459–463.

[26] Zhao, X., Lee, M.-E. a Kim, S.-H. Improved image thresholding using ant colony optimization algorithm. In: IEEE. *2008 International Conference on Advanced Language Processing and Web Information Technology*. 2008, s. 210–215.

Příloha A

Struktura adresářů na přiloženém CD

- **xprase07.pdf** - Bakalářská práce ve formátu pdf
- **src** - Složka s praktickou částí bakalářské práce
- **src/EXP1** - Složka se zdrojovými kódy k experimentu 1
- **src/EXP2** - Složka se zdrojovými kódy k experimentu 2
- **src/EXP3** - Složka se zdrojovými kódy k experimentu 3
- **src/EXP4** - Složka se zdrojovými kódy k experimentu 4
- **src/EXP5** - Složka se zdrojovými kódy k experimentu 5
- **src/standard_test_images** - Složka s obrázky pro testování
- **text** - Složka se zdrojovými texty pro vytvoření bakalářské práce
- **text/bib-styles** - Složka se styly pro správný překlad
- **text/obrazky-figures** - Složka s obrázky obsaženými v této práci
- **text/template-fig** - Složka s logem VUT na titulní stránku

Příloha B

Jak pracovat s programem

Pro sestavení a spuštění programu je třeba mít nainstalovanou knihovnu OpenCV pomocí návodu pro Windows[1] nebo pro Linux[2].

Práce s Makefile:

- Program lze přeložit příkazem `make` v adresáři src.
- Experimenty lze spustit po sestavení třemi různými způsoby, opět z adresáře src.
  1. `make test` - spustí veškeré testy.
  2. `make exp_` - kde lze dosadit číslo libovolného experimentu (1-5), který spustí. Např.: `make exp1`.
  3. `make exp_._` - kde lze dosadit čísla libovolného testu v experimentu (experiment 1 má rozmezí 1-6, ostatní mají rozmezí 1-4), který spustí. Např.: `make exp2.3`
- Adresář lze vyčistit od binárních souborů a výsledků příkazem `make clean`

Během provádění výpočtu se v konzoli zobrazuje číslo cyklu a minimální a maximální hodnota feromonů. Zároveň se zobrazí okna, v jednom jsou označeny počáteční pozice mravenců a ve druhám je zobrazován průběžný výsledek. Po doběhnutí experimentu se výsledky uloží do složky RESULTS/EXPx.y, kde x je číslo experimentu a y je konkrétní test. Pokud se program spouští ručně, jsou výsledky uloženy ve složce RESULTS, pokud existuje. Mezi výsledky nalezneme průběžný výsledek po každém cyklu, čas simulace, intenzity jednotlivých pixelů ze vstupního obrázku a vypočítané $V_c(I_{ij})$ pro každý pixel.

[1] https://docs.opencv.org/2.4/doc/tutorials/introduction/windows_install/windows_install.html

[2] https://docs.opencv.org/master/d7/d9f/tutorial_linux_install.html