VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INTELIGENTNÍCH SYSTÉMŮ
DEPARTMENT OF INTELLIGENT SYSTEMS

PLÁNOVÁNÍ CESTY PRO AUTONOMNÍ ZEMĚDĚLSKÉ STROJE

PATH PLANNING FOR AUTONOMOUS AGRICULTURAL MACHINES

SEMESTRÁLNÍ PROJEKT
TERM PROJECT

AUTOR PRÁCE **MARTINA CHRIPKOVÁ**
AUTHOR

VEDOUCÍ PRÁCE **Ing. JAROSLAV ROZMAN, Ph.D.**
SUPERVISOR

BRNO 2021

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav inteligentních systémů (UITS) Akademický rok 2020/2021

Zadání bakalářské práce

Studentka: **Chripková Martina**
Program: Informační technologie
Název: **Plánování cesty pro autonomní zemědělské stroje**
Path Planning for Autonomous Agricultural Machines
Kategorie: Umělá inteligence

Zadání:

1. Nastudujte algoritmy pro plánování cesty. Zaměřte se na algoritmy používané pro kompletní pokrytí daného prostoru (tzv. coverage path planning). Konkrétně se pak zaměřte na metody, jak správně naplánovat cestu pro zemědělský stroj s ohledem na sklon pole a vstupní a výstupní body.
2. Navrhněte program, který umožní vytvořit nebo z externího souboru nahrát tvar oblasti a pro ni vytvořit optimální plán cesty.
3. Navržený program implementujte a otestujte jeho funkčnost. Program otestujte i pro více oblastí, mezi kterými bude nutné přejet.

Literatura:

- Howie Choset et al., Principles of Robot Motion, 2005, ISN 0-262-03327-5.

Pro udělení zápočtu za první semestr je požadováno:

- První dva body zadání.

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

Vedoucí práce: **Rozman Jaroslav, Ing., Ph.D.**
Vedoucí ústavu: Hanáček Petr, doc. Dr. Ing.
Datum zadání: 1. listopadu 2020
Datum odevzdání: 30. července 2021
Datum schválení: 11. listopadu 2020

Abstrakt

Táto práca predstavuje návrh algoritmu pre trajektóriu pohybu autonómnych poľnohospodárskych strojov. Hlavným cieľom práce bolo navrhnúť algoritmus, ktorý dokáže minimalizovať trasu Tým pádom šetrí čas a náklady spojené s obhospodarovaním poľnohospodárskych plôch. Algoritmus dbá aj na potreby ochrany pôdy proti erózií.

Abstract

This thesis presents the design of a coverage path planning algorithm for autonomous agricultural machines. The main goal of this work was to design an algorithm that can minimize the route. As a result, it saves time and costs associated with the management of agricultural land. Algorithm also takes care of the needs of soil protection against erosion.

Klíčová slova

plánovanie cesty, plánovanie cesty pokrytia povrchu, python, matplotlib, autonómne poľnohospodárske stroje

Keywords

path planning, coverage path planning, python, matplotlib, autonomous agricultural machines

Citace

CHRIPKOVÁ, Martina. *Plánování cesty pro autonomní zemědělské stroje*. Brno, 2021. Semestrální projekt. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Jaroslav Rozman, Ph.D.

Plánování cesty pro autonomní zemědělské stroje

Prohlášení

Prehlasujem, že som túto bakalársku prácu vypracovala samostatne pod vedením pána Ing. Jaroslava Rozmana, Ph.D.. Uviedla som všetky literárne pramene, publikácie a dalšie zdroje, z kterých som čerpala.

.....
Martina Chripková
2. srpna 2021

Poděkování

V prvom rade by som chcela poďakovať svojmu vedúcemu práce, pánovi Ing. Jaroslavovi Rozmanovi, PhD., za nasmerovanie a pripomienky. Ďalej by som somra poďakovala svojej rodine a priateľom, ktorý mi pomáhali.

Obsah

Kapitola 1

Úvod

Už od nepamäti je poľhospodárstvo hlavnou činnosťou človeka pre získanie obživy. Otázkou správneho a najefektívnejšieho obrábania pôdy sa už zaberali naši predkovia, a nie je to inak ani dnes. Problém ako obrobiť pôdu čo najrýchlejšie, najšetrnejšie a s minimálnym ľudským zásahom je dôležitou témou dnešnej spoločnosti. Poľnohospodárstvo je dodnes považované za veľmi špecifický sektor, ktorý má strategický význam pre štát a jeho občanov [18]. Potreba precízneho poľnohospodárstva je silná v systémoch, kde sa uchytilo pestovanie veľkých plôch monokultúr [4]. Česká republika si osvojila tento systém po druhej svetovej vojne, kedy štátna politika preferovala formu obrovských poľnohospodárskych plôch. Takto to funguje až do dnešného dňa, kedy vlastníci pozemkov argumentujú výnosnosťou a konkurencieschopnosťou poľnohospodárov ako nutné podmienky na prežitie [14].

Cieľom práce bolo vytvoriť algoritmus, ktorý dokáže nájsť najlepšiu trasu, ktorá berie ohľad na eróziu pôdy, dĺžku trajektórie a taktiež zakrivenie povrchu. Môže sa zdať, že optimálna trasa je tá najkratšia, no náklady na pohonné hmoty môžu byť zanedbateľné oproti nákladom na revitalizáciu pôdy, ktorá bola spôsobená neekologickým obrábaním.

Táto práca sa zaoberá návrhom a implementáciou optimálneho plánovacieho algoritmu, ktorý hľadá trasu pokrývajúcu celý povrch parcely.

V prvej kapitole sú približené algoritmy na plánovanie trasy. Následuje analýza nákladových funkcií, ktoré sú použité pri návrhu. V ďalšej kapitole je predstavený samotný návrh algoritmu aj s cieľovými požiadavkami. V poslednom rade je možné nájsť informácie ohľadom implementácie a testovaní algoritmu. Práca zahŕňa aj diskusiu výsledkov, kde je možné nájsť návrhy na prípadne pokračovanie.

Kapitola 2

Plánovanie cesty

Plánovať cestu, aby sme pokryli povrch je potrebné vo viacerých prípadoch ako napríklad umývanie podlahy, kreslenie, kosenie trávy, skúmanie morského dna. Cao, Huang a Hall definovali v ich štúdií [6] kritéria na trasu pokrytia povrchu pre robota následovne:

- Povrch musí byť pokrytý celý
- Pokrytie povrchu bez prekrývania
- Nepretržitá prevádzka bez opakovania ciest
- Robot sa musí vyhnúť všetkým prekážkam
- Jednoduché trajektórie pohybu (priamky)
- Podľa možnosti je k dispozícií optimálna cesta

Pri takomto komplexnom probléme ako je plánovanie cesty pokrytia povrchu nie je stále možné splniť všetky kritériá a preto je kľúčové určiť si prioritu.

Trasa pokrytia povrchu je úzko zviazaná s problémom obchodného cestujúceho (nazývaným TSP), kedy máme N miest a z každého mesta je možné sa dostať do všetkých ostatných. Každá cesta medzi dvoma mestami je ohodnotená číslom, ktoré môže vyznačuje vzdialenosť, cenu, čas a podobne. Cieľom je navštíviť každé mesto práve raz a vrátiť sa naspäť do počiatočného mesta tak, aby výsledné číslo bolo čo najmenšie. Problém obchodného cestujúceho je klasifikovaný ako ťažký NP[1] problém. V tejto kapitole budú predstavené niektoré z algoritmov, ktoré riešia danú problematiku.

2.1 Metóda Lin-Kernighan

Metóda Lin-Kernighan(LK) je jednou z heuristických stratégií, ktoré riešia TSP. Základným princípom je postupné zlepšovanie dosadenej hamiltonovskej kružnice, pomocou výmeny k hrán pri každej iterácií algoritmu a následnej kontroly zlepšenia riešenia [8]. S rastúcim k sa zvyšuje presnosť výsledého riešenia, no výpočetná zložitosť sa zväčšuje. Je možné dosiahnúť lineárnu zložitosť algoritmu na úkor výslednej trajektórie, ktorá v takom prípade nebude optimalizovaná. Kombinácia zhlukovania, generických algoritmov a metodou LK vznikla nová trojstupňová metóda, ktorá nájde kvalitnú trasu s takmer lineárnou časovou zložitosťou.

[1] prevediteľný v polynomickom čase na Turingovom stroji

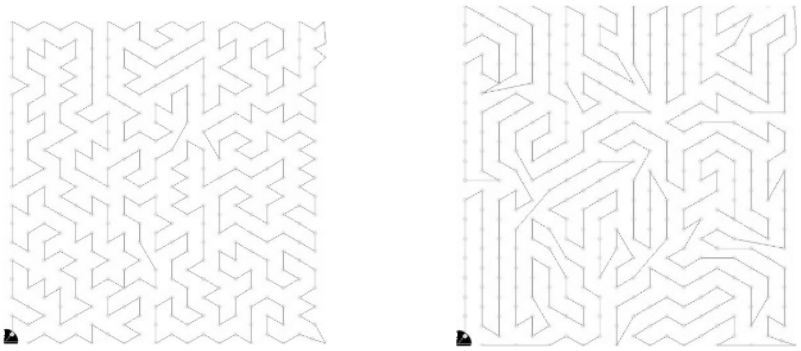
2.2 Evolučná metóda Lin-Kernighan

Jednotlivé dráhy, z ktorých sa skladá výsledná trajektória sú plánované v rámci každého clusteru pomocou metódy LK obohatený o prvok nazývaný Evolutionary Lin-Kernighan (ďalej len ELK). Táto schéma vychádza z genetických operátorov s podobnými architektúrami [15].

Využitie metódy clusterov k udržaniu priemernej veľkosti bunky udržuje náročnosť každej inštancie metódy ELK konštantnú. Ak sa algoritmu podarí udržať bunky podobných veľkostí iba s minimálnymi odchýlkami, tak vypočítané sub trajektórie budú taktiež nabývať podobné rozmery. Týmto je zaistená lineárna časová náročnosť a zároveň algoritmus stále zachováva dobrú kvalitu prejazdu. Zároveň sa tým umožní paralelný výpočet, kedže každá dielčia trajektória môže byť vypočítaná individuálne a môže byť nezávisle optimalizovaná. [15]

Kvalita trasy podľa algoritmu ELK v porovnaní s algoritmom LK je lepšia. Metódou ELK bola vytvorená cesta v menej iteráciách a mala vyššiu kvalitu. Ďalším zistením je, že výpočet trasy trval podstatne kratšiu dobu.

Výsledky štúdie Meutha a Wunscha ukazujú, že evolučná metóda ELK je schopná výrazne zlepšiť dobu dokončenia pre vozidlo s obmedzenou pohyblivosťou [15]. No v prípadoch, kde musí vozidlo v reálnom čase plánovať efektívnu cestu veľkým a zložitým prostredím, nie je ani táto metóda dostatočná.



(a) Optimalizácia podľa euklidovskej vzdialenosti

(b) Optimalizácia podľa rovnice pre heuristické fyzické vozidlo

Obrázek 2.1: Porovnanie výsledných trajektórií podľa euklidovskej vzdialenosti a vytvorenej rovnice pre heuristické fyzické vozidlo. Prevzaté z [15]

2.3 Plánovanie trasy na základe obmedzenej Delaunyho triangulácie (CDT)

Delaunyho triangulácia spočíva vo vytvorení záchytných bodov v priestore, od ktorých budú vypočítané bunky s maximalizovaným minimálnym uhlom tak, aby žiaden z bodov nezostal vo vnútri trojuholníkových buniek [19]. Tieto body, inak nazývané aj uzly, znamenajú v poľnohospodárskom priemysle prekážky v poli. Metóda CDT je dvojrozmerná. Jedná sa o zobecnený prístup ku konkávnemu prostrediu, kde uzly smerujú k okrajom a

ich počet je minimálny. Bránami sú dve geometrické informácie, čím vzniká dvojitý graf - plán cesty. Výslednou trasou sa tým pádom stávajú spojnice stredov, ťažísk, buniek.

Konštrukcia dvojitého grafu

Aby sa uzly a hrany grafu dostali na okraje polygónov, využíva sa algoritmus A*. Aj napriek tomu nemusí byť trasa optimálna, obzvlášť, ak sa v grafe objavujú príliš veľké bunky. V praxi by to znamenalo, že by vozidlo prešlo danú oblasť len jeden krát, bez ohľadu nato, že nezobralo celú úrodu. V algoritme je zvolený počiatočný bod A, cieľový bod G a ďalšie body ako uzly. Správna poloha uzlov je definovaná podľa siedmych podmienok, čím vzniká 6 rôznych prípadov [19].

- prípad C1: Cieľový bod je umiestnený v následujúcom trojuholníku
- prípad C2: Aspoň 1 z priesečníkov AG a druhého trojuholníka je na spoločnej strane trojuholníkov
- prípad C3: Počet priesečníkov krivky AG je menší ako 2
- prípad C4: Priesečník AG druhého trojuholníka je na spoločnom okraji, nie je tam prekážka
- prípad C5: Existuje tretí trojuholník, ktorý zdieľa s druhým trojuholník iba vrchol, tympádom by sa robot okolo druhého trojuholníku len otočil
- prípad C6: Tretí uzol má bližšie ku krajným bodom A a G ako k iným bodom
- prípad C7: Neexistuje 3. trojuholník, ktorý by zdieľal stranu s druhým trojuholníkom

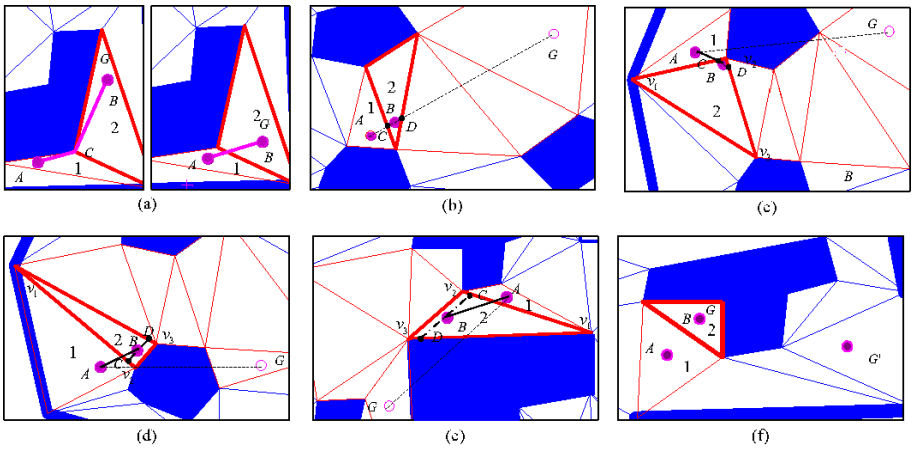
Nasleduje tabuľka, ktorá opisuje kombináciu podmienok, ktoré sa môžu vyskytnúť v reálnom svete [19]

Prípad	Zoznam podmienok
1.	$C_1 \cap \neg C_2$(a).left; $C_1 \cap \neg C_2$(a).right
2.	$C_1 \cap C_2 \cap \neg C_3 \cap C_4$
3.	$\neg C_1 \cap (\neg C_2 \cup C_3 \cup \neg C_4) \cap C5 \cap \neg C_6$
4.	$\neg C_1 \cap (\neg C_1 \cup C_3 \cup \neg C_4) \cap \neg C_5$
5.	$\neg C_1 \cap (\neg C_2 \cup C_3 \cup \neg C_4) \cap C_5 \cap C_6$
6.	C_7

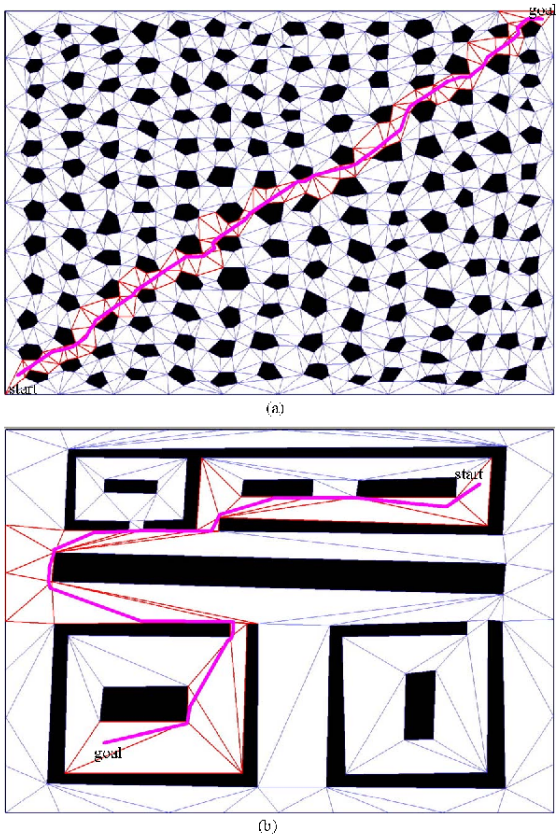
Náklady sú úmerné euklidovskej vzdialenosti bodov AB, ak im od cesty nie je postavená prekážka.Môže nastať situácia, kedy robot prejde bez prekážok do ďalšieho trojuholníka k cieľovému bodu B (B=G). 2.2 Ak sa cieľový bod G nachádza v inom trojuholníku a trojuholníkom 2 iba prechádza, tak vznikajú 2 body: bod C na spoločnom okraji počiatočného trojuholníka a trojuholníka 2, a bod D na spoločnom okraji trojuholníku 2 a cieľového trojuholníku. Bod je je stredom úsečky CD [19].

Existujú štyri prípady, kedy sa robot nemôže pohybovať rovno. Prvý prípad je vyobrazený na obrázku 2.2 , kedy sa robot musí vyhýbať prekážke. Potom je algoritmus vytvorený na náklade podmienok uvedených v tabuľke 2.3. Jedná sa o 2 trojuholníky, ktoré algoritmus vyberá, aby sa vyhol prekážke.

Posledný prípad je, keď je cesta zablokovaná konkávnou prekážkou (obr. 2.2 f). Táto situácia nastane, keď všetky tri vrcholy trojuholníka 2 sú zároveň aj vrcholy ten istej prekážky. Ak je cieľový bod v trojuholníku 2, berie sa cieľový bod G ako uzol B, viď prípad 1. Ak nie je cieľový bod v tomto trojuholníku, vráti sa bod B ako prázdna hodnota [19].



Obrázek 2.2: Ilustrácia možnosti prejazdu v poli. Prevzaté z [19].



Obrázek 2.3: Ilustrácia možnosti prejazdu v poli. Prevzaté z [19].

Stanovenie optimálnych ciest

Pole je tvorené nekonzistentným polygónom s dierami, ktoré predstavujú geometrické vyjadrenie prekážok. Tento dvojitý graf na trianguláciu prostredia, predstavuje konečný graf. Ak existuje optimálna cesta od počiatočného bodu k cieľu, algoritmus ju nájde. Graf je vytvorený až po triangulácií prostredia, v ktorom sa pohybuje. Vertexy a následné náklady na sú v súlade s optimalizačným algoritmom A*.

```
1  procedure ComputeShortestPath()
2         while(OPEN != 0)
3                 t = OPEN.Pop()
4                 if(t = t_goal)
5                         return true;
6                 for all t' in Succ(t)
7                         if( t' in NEW)
8                                 t'.node = FindNode(t, t');
9                                 g(t') = g(t)+c(t,t');
10                                t'.parent = t;
11                        OPEN.Insert(t',g(t') + h(t', t_goal));
12                else if(t' in OPEN)
13                        t_tmp = t';
14                        t_tmp.node=FindNode(t, t_tmp);
15                        if(g(t') > g(t)+c(t,t_tmp) and g(t') > g(t)+c(t, t'))
16                                t'.node=t_tmp.node;
17                                g(t')=g(t)+c(t,t');
18                                t'.parent=t;
19                                OPEN.Update(t',g(t')+h(t',t_goal));
20        return false
21 procedure Main()
22        OPEN is empty;
23        g(t_start) = 0;
24        OPEN.Insert(t_start,(g(t_start)+h(t_start,t_goal)));
25        if(ComputeShortestPath())
26                return GOALREACHED;
27        else
28                return NOPATH;
29 End of algorithm
```

Výsledky simulácie

Vo väčšine prípadov sa vypočítajú náklady na optimálnu cestu, ak prekážka nespôsobí, že robot zmení svoje správanie počas behu v priamke smerom k cieľovému bodu k pohybu po hranici prekážky [19]. Ak existujú viac ako dve prekážky ($q \geq 2$), potom by vznikla križovatka s dvoma cestami.

Vyššie uvedené pravidlá boli nasimulované a použité pri plánovaní cesty pre robota v prostredí, kde sú známe globálne informácie. V tomto prípade prostredie obsahuje 904 vrcholov a 1236 trojuholníkov s pravidelnými prekážkami. V príklade na obrázku 2.2f tvorí prostredie mnoho konkávnych prekážok a je vyjadrené 98 vrcholmi a 114 trojuholníkmi [19].

2.4 Optimálne plánovanie pokrytia na 2D povrchoch

V poľnohospodárskom priemysle je použitie bustrofédonských[2] ciest najjednoduchšou možnosťou. V štúdií podľa Jin a Tang je prioritou určenie najlepšieho smeru cesty. Existuje niekoľko metód ako vyhľadať optimálny smer [12]. Najjednoduchšou metódou je sledovať najdlhší okraj poľa. Takto by bolo možné získať optimálnu trasu len pre pole s jednoduchým konvexným tvarom, napríklad obdĺžnikovým. Optimálny algoritmus na plánovanie pokrytia cesty pre poľnohospodárstvo spočíva v niekoľkých krokoch. Najskôr je geometricky znázornený tvar poľa. Následovne je vyhľadávaný optimálny smer cesty a taktiež optimálne rozloženie poľa. Je definovaná nákladová funkcia uhlových otočení 3.2 , na základe ktorej je vypočítaný smer.

Cieľom bolo získať presný odhad premennej C_{turn} (náklady na obrábanie). Vzhľadom k obmedzenému minimálnemu polomeru otáčania, definovanej šírke riadku a obmedzenému priestoru, nemusí byť vždy vhodné použiť obrat typu „U“. Typy otočení sú podrobnejšie popísané v kapitole 3.2.

Zložitosť algoritmu je definovaná ako $O[n^3 log(n)]$, kde n je počet hrán poľa [12]. Výsledné testovanie ukázalo, že nákladny na otočenie sú o 16% menšie. Autori podotýkajú na možnosť ďalšie vylepšenia, ako napríklad možnosť voľby štartu a cieľa.

2.5 Metóda split-and-merge

V prípade, že je pole konvexné a nie sú na ňom žiadne prekážky, plánovanie cesty je pomerne jednoduché. Metóda split-and-merge rieši problém nájdenia optimálne trasy ak je pole konkávne, čo znamená, že obsahuje prekážky.

Hlavnou myšlienkou štúdie [17] je delenie poľa na menšie objekty, ktoré je možné obrábať samostatne. Algoritmus funguje na princípe hladného algoritmu[3]. V každom kroku rozdelenia na menšie regióny, hľadá blok s najlepšími nákladmi. Bloky sú vytvárané pomocou lichobežníkového rozkladu a zlučovania. Tento postup sa opakuje z rôznych uhlov smeru jazdy. V každom kroku sa vyberie blok s najlepšími nákladmi a následne sa z poľa odstráni. Postup sa opakuje aj pre zvyšnú oblasť. Algoritmus pokračuje, dokým sa celé pole nerozdelí na podpolia. Jazdná dráha sa skladná z priamok, čo znamená, že vyhľadávanie je rýchlejšie. Nevýhodou tohto postupu je, že sa neprispôsobí zakrivenej hrane poľa. V ďalšom kroku sa vyhľadáva optimálny prvý smer jazdy, ktorý je opísaný následujúcim algoritmom [17]:

1. Cena je vypočítaná pre šesť smerov: *0°, 30°, 60°, 90°, 120°, 150°*
2. Vyberú sa tri najlepšie smery a ostatné sa zahodia.
3. Veľkosť kroku v smere uhlu vyhľadávania sa zmenší na polovicu.
4. Pridajú sa nové smery vyhľadávania na obe strany troch najlepších smerov.
5. Vypočíta sa cena pre smery, ak ešte nebola vypočítaná.
6. Ak je dosiahnutý cieľ, koniec algoritmu, inak sa vráť na krok 2.

[2] rovné paralelné cesty so striedavým smerom

[3] `https://en.wikipedia.org/wiki/Greedy_algorithm`

Po piatich cykloch je rozlišovacia schopnosť menšia ako 1°, čo autori vyhodnotili ako dostatočné[17]. Zvyšná časť poľa je vyriešená bustrofédonskou metódou.
Algoritmus je schopný nájsť riešenie pre každé pole, čo bolo overené aj manuálnou kontrolou. Niektoré z výsledných trás avšak nie sú uskutočniteľné. Problémy vznikli, ak pole obsahovalo veľa malých prekážok. V takom prípade vznikali príliš komplikované trasy. Taktiež sa zistilo, že ak je pole konvexné a má definovanú najdlhšiu hranu, tak smer jazdy bol zhodný s najdlhším okrajom poľa.

2.6 Optimálne plánovanie cesty pokrytia na 3D povrchoch

Veľké množstvo obrábanej pôdy na Slovensku a Česku má sklon medzi dvomi až desiatimi percentami. Z tohto hľadiska má optimalizácia 3D plánovania veľký potenciál. Zobrazenie poľa v 2D a nebranie do úvahy sklon poľa môže viesť k tomu, že sa neobrobí celé územie, alebo trasa sa bude prekrývať. Následkom toho môže byť veľká ekonomická ujma. Ďalším problémom, ktorý je ignorovaný pri 2D modelovaní, je erózia pôdy. V štúdií Wendt uviedol, že spôsob obrábania pôdy vykonávané po vrstevniciach účinne znižujú eróziu pôdy, ktorá je spôsobená najmä búrkami nízkej až strednej intenzity[11].

Pri plánovaní na 3D povrchoch v poľnohospodárskom priemysle sa v praxi osvedčili heuristické metódy ako napríklad orba po vrstevnici, čím sa minimalizuje erózia pôdy. Môže byť využitý „seed curve“ algoritmus. Ten zvolí jednu vrstevnicu alebo okraj okraj poľa ako vzorovú trasu, z ktorej sa spravia rovnobežky tak, aby sa pokrylo celé územie. Algoritmus je opísaný v kapitole 6.2.

2.7 Cesta plánovania poľnohospodárskeho mobilného robota pomocou neurónovej siete a genetického algoritmu

Pri spracovaní optimalizovaného pohybu na poľnohospodársky využívaných plochách bola predstavená aj možnosť kombinácie neuronovej siete (ďalej NN) a genetického algoritmu (ďalej GA). Princíp NN sa aplikuje na popis pohybu robota. Ten má vysokú schopnosť učiť sa. Ak bola cesta vytvorená pomocou simulátoru NN, je potrebná následná optimalizácia. V temto prípade bola použitá metóda optimalizácie metoda GA, ktorá je inšpirovaná biologickým vývojom. Používa proces variácie a výberu pre hľadanie riešenia.

Robot bol navrhutý tak, aby špecifikáci odpovedala špecifikácii konvekčných traktorov s malým rozmerom. Má pohon na zadných kolesách a je poháňaný benzínovým motorom. Robot je schopný merať uhol riadenia pomocou potenciometrov a otáčanie zadných kolies pomocou rotačného pohonu kódermi. Na rozdiel od pohybu poľnohospodárskych vozidiel, ktoré používajú väčšie uhly riadenia (podobne ako automobily), vykazuje vysokú nelinearitu. Z toho dôvodu nemôže byť modelovaná s vysokou presnosťou s konvenčnými technikami. [16]

Prvý výpočet zahrňuje vynásobenie všetkých vstupov hmotnosti w_ij a následným sčítaním výsledkov ako:

$s_j = \sum w_{ij} a_{ij}$

Pre meranie polohy robota bolo potrebné nastaviť uhol riadenia merania.

Cesty boli následne pomocou GA optimalizované. Trasy sú definované ako reťazce, ktoré predstavujú miesta vo vyhľadávacom priestore. Reťazec s konečnou dĺžkou je definovaný ako individuálne riešenie s objektívnou funkčnou hodnotou bodu vo vyhľadávacom priestore.

Bolo zistené, že chyby v konečnom stave dokázal robot znížiť. Postupom generácie sa približoval aj cieľový stav robota. Z výsledkov je jasné, že vyvinutá metóda používajúca GA a NN bola efektívna. Cesta, ktorá používala kormidelné riadenie pripomínala kruhový obrat a cesta vytvorená objektívnou funkciou cesty bola podobná priamke. Kvalita vytvorenej cesty bola vylepšená každou generáciou GA.

Kapitola 3

Analýza nákladových funkcií

Kapitola sa zaoberá analýzou nákladových funkcií, ktoré majú podiel na výslednej trajektórií.

3.1 Analýza ceny erózie pôdy

RUSLE[1] je model, ktorý predpovedá dlhodobú priemernú ročnú stratu pôdy, ktorá odteká zo svahov v špecifikovaných systémoch obrábania. Model je definovaný rovnicou:

$$A = R * K * L * S * C * P$$

A označuje priemernú stratu erózie pôdy v tonách za rok.
R je faktor zrážkovej erozivity[2]
K je faktor pôdnej erodovateľnosti[3]
S určuje strmosť svahu
L je dĺžka hrany C je faktor, vyjadrujúci účinok postupov pestovania a obhospodarovania na rýchlosť erózie
P značí pomer straty pôdy

Trasa pokrytia ovplyvňuje len faktor P, tým pádom sa hodnota P môže použiť na označenie nákladom na eróziu pôdy. D je rozsah poľa, S_f je známka smeru cesty a S_l zastupuje lokálnu strmosť svahu [11].

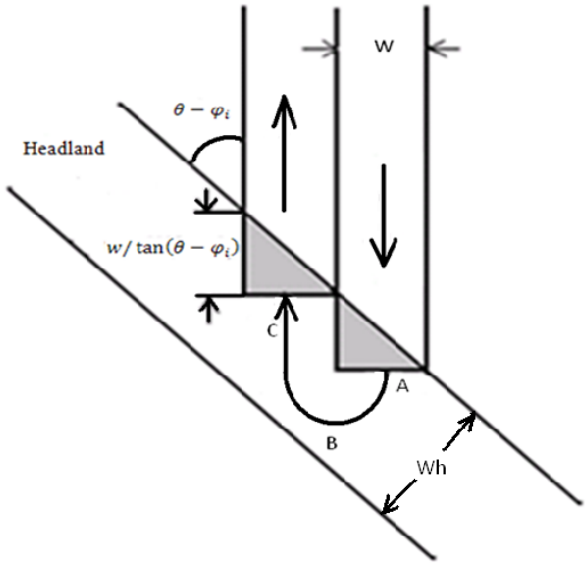
$$P_{off-grade} = \frac{\int\int_D [P_o + (1-P_o)(\frac{S_f}{S_1})^{\frac{1}{2}}] \sqrt{1+(\frac{dz}{dx})^2+(\frac{dz}{dy})^2}\mathrm{d}x\mathrm{d}y}{\int\int_D \sqrt{1+(\frac{dz}{dx})^2+(\frac{dz}{dy})^2}\mathrm{d}x\mathrm{d}y}$$

3.2 Analýza ceny otočenia

Na obrázku 3.1 sa predpokladá, že je možné vykonať obrat typu „U“. Vzhľadom na obmedzený minimálny polomer otáčania poľnohospodárskeho zariadenia, definovanú šírku riadkov a obmedzený priestor nemusí byť takýto obrat v niektorých situáciach použiteľný. V iných prípadoch sa môže obrat typu „U“ aplikovať, ale nie je to nákladovo najefektívnejšie riešenie. Namiesto toho sa používajú aj ďalšie typy obratov, napríklad ploché obraty, zákruty typu

[1] Revised Universal Soil Loss Equation
[2] kinetická energia a intenzita zrážok na opísanie účinku zrážok na eróziu pôdy
[3] miera citlivosti pôdnych častíc na oddelenie a transport zrážkami a odtokom

žiarovka (inak nazývaná aj kľúčová dierka), háčik (alebo aj asymetrická žiarovka) alebo zákruta typu rybí chvost. Podrobnosti o jednotlivých typoch a ich porovnania je možné nájsť v následujúcich podkapitolách.
Predpokladá sa, že rýchlosť v je v každom type rovnaká.

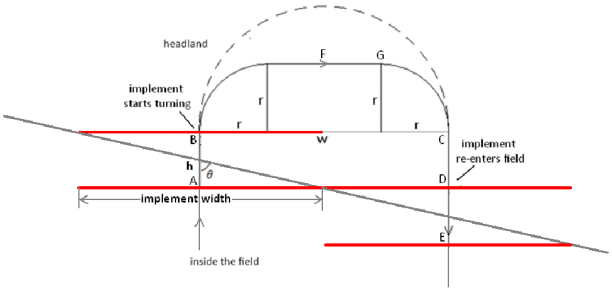


Obrázek 3.1: Ilustrácia uhlovej zákruty. w je šírka riadku, θ znamená smer riadku φ je smer hrany. Prevzaté z [11]

Plochá zákruta

Keď je polomer otáčania vozidla menší ako polovica šírky riadku ($r < w/2$) môže byť vykonaná plochá zákruta namiesto klasickej „U“, ktorá má väčši polomer otáčania. Tento typ otočenia znižuje dĺžku celkovej trajektórie, čím znižuje aj časové náklady na otáčanie. Za predpokladu, že v je rýchlosť otáčania, náklady na otočenie sú:

$$C_{turn} = \frac{w(1+\cot\theta)+r(\pi-2)}{v}$$

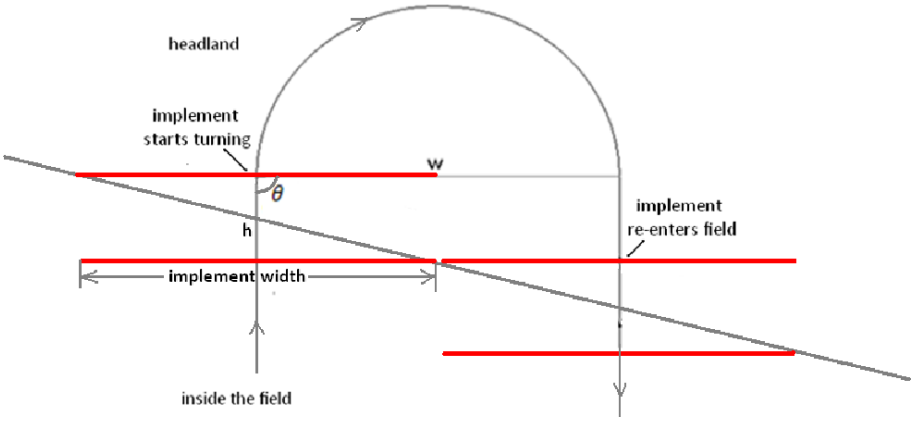


Obrázek 3.2: Plochá zákruta. Prerušovaná čiara znázorňuje zákrutu typu „U“. Prevzaté z [11]

Zákruta typu „U“

Otočenie tohto typu je vhodné uskutočniť, keď je dosiahnutý kritický bod plochej zákruty $r = w/2$. [11] Nákladová cena otočenia je:

$$C_{turn} = \frac{(\pi+2cot\theta)w}{2v}$$

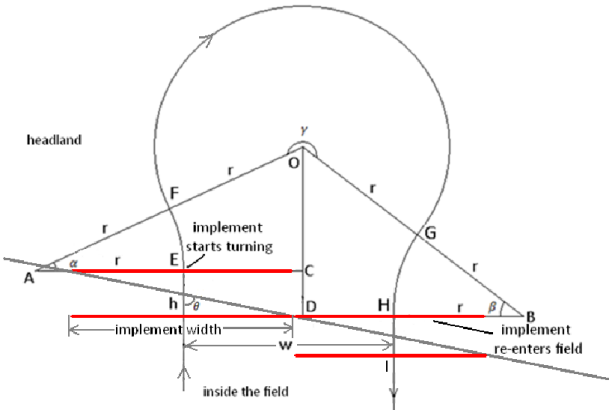


Obrázek 3.3: Zákruta typu „U“. Prevzaté z [11]

Zákruta typu „žiarovka“

V prípade, že $r > w/2$, miesto na vykonanie plochej zákruty alebo zákruty typu „U“ nie je dostatočné. Pohyb vozidla začína otočením do protismeru, aby vozidlo získalo dostatočný priestor na otočenie(E-F). Potom sa otočí späť (F-G) a na záver opäť otočí smer (G-H), aby sa dostalo na ďalší riadok. Na obrázku 3.4 je možné vidieť štartovací bod E, ktorý je zároveň aj koncovým bodom poľa. Obrat končí, keď sa vozidlo opäť vráti na pole, čo značí aj bod H. V tomto bode musí byť nasmerované v smere riadku. Vozidlo by sa malo otáča s minimálnym polomerom otočenia r. [11] Cena otočenia je:

$$C_{turn} = r(\pi + 2\arccos(\frac{w}{2r} + \frac{w^2(1+\tan^2\theta)}{8r^2\tan^2\theta} - 1/2))/v$$

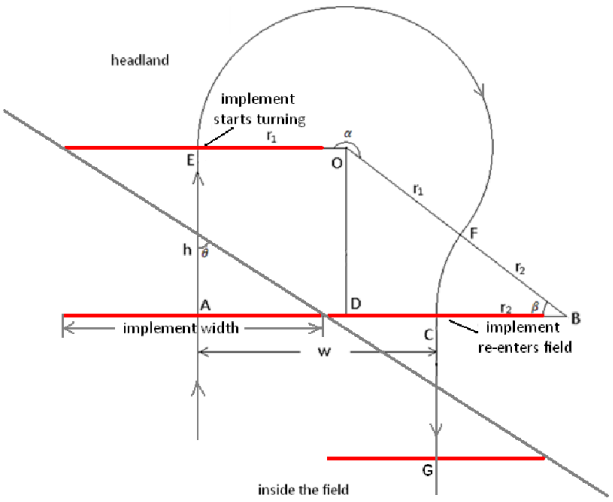


Obrázek 3.4: Zákruta typu žiarovka $r > w/2$. Prevzaté z [11]

Háčik alebo asymetrická „žiarovka“

Ďalším typom otočenia sa je háčik. Ten je možné vykonať, ak $r > w/2$. Namiesto toho, aby začiatok zákruty smeroval do opačného smeru, ako to je pri „žiarovke“, tento obrat začína rovnako ako obrat typu „U“. Po dosiahnutí bodu F sa zmení smer otočenia a vráti sa na ďalší riadok. 3.5
Nákladová cena na otočenie je definovaná ako:

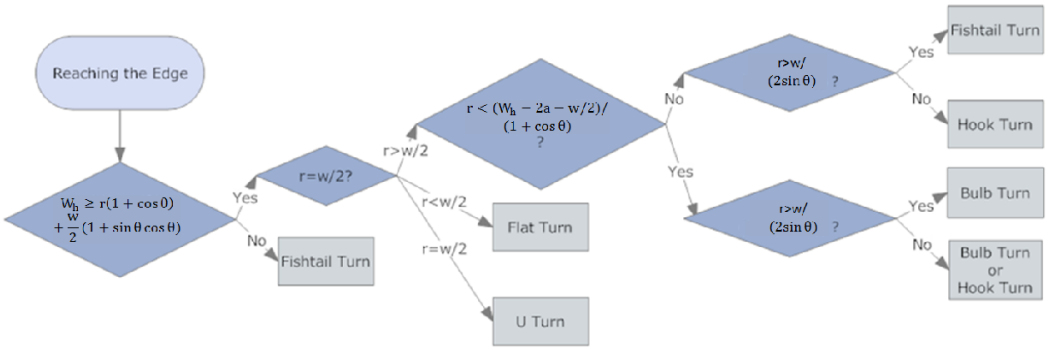
$$C_{turn} = (r\pi + \frac{4r^2-4wr+w^2\cot^2\theta+w^2}{4r-2w}\sin^{-1}\frac{4rw\cot\theta-2w^2\cot\theta}{4r^2-4rw+w^2\cot^2\theta+w^2})/v$$



Obrázek 3.5: Zákruta typu háčik. Prevzaté z [11]

3.3 Výber vhodného otočenia

Výber najvhodnejšieho typu obratu je dôležitý. Správny typ otočenia dokáže minimalizovať trajektóriu. Voľba správneho typu záleží od šírky riadku, voľného priestoru, minimálneho polomeru otočenia a uhla medzi riadkom a okrajom. Obmedzenia a podmienky pre každý typ sú zhrnuté v rozhodovacom strome na obrázku 6.2 znázornenom nižšie. Keď platia obe nerovnice $W_h > r(1 + 2\ \sin\theta\sin\alpha + 2\cos\theta\cos\alpha - \cos\theta) + \frac{w}{2}$ a zároveň $r \leq \frac{w}{2\sin\theta}$, tak je možné použiť nie len „žiarovku“, ale aj „háčik“. Aby sme vybrali najvhodnejší typ, je potrebné porovnať nákladové funkcie týchto dvoch otočení. Zistilo sa, že „žiarovka“ má stále menšiu trajektóriu. Avšak „háčik“ má taktiež svoje výhody. Jednou z nich je, že šírka potrebná k obratu môže byť menšia. Väčší polomer otáčania v FC (viď. obrázok 3.5) spôsobuje ľahšiu adaptáciu vozidla na susedný riadok pred opätovným vstupom. Z toho vyplýva, že voľba medzi týmito dvoma typmi obratov závisí od preferencií poľnohospodára.



Obrázek 3.6: Rozhodovací strom pre výber druhu otočenia. Prevzaté z [11]

Kapitola 4

Charakteristika strojov

Traktor je univerzálny poľnohospodársky stroj, ktorý ma väčšinou náhon na zadných kolesách[1]. Je schopný utiahnuť ďalšie zariadenie, ktoré zaisťuje prácu na poli ako napríklad orbu. Väčšinou nie je problém pripevniť na záves ľubovoľné zariadenie.

4.1 Autonómne poľnohospodárske stroje v súčasnosti

Pred niekoľkými rokmi boli autonómne poľnohospodárske roboty len v myšlienkách vizionárov. Jednou z firiem, ktoré ponúkajú autonómne systémy ponúka firma PrecisionHawk vo forme kombinovaných balíkov. Tie zahŕňajú robotický hardvér a softvér na analýzu. Poľnohospodár môže vo forme služby presunúť drony na role, inicializovať softvér a zobraziť informácie o plodinách v reálnom čase.

Cieľom viacerých viacerých vývojových inštitúcii je vyhotoviť autonomného mobilého robota, ktorý bude v reálnom teréne schopný samostatne a bez kolízií prejsť medzi všetky riadky. Roboti môžu poáhať poľnohospodárom nie len na zber informácií o pôde, ale aj na vykonávanie mechanickej práce.

V súčastnosti je snaha o modernizáciu poľnohospodárskeho priemyslu, avšak úplne autonómny poľnohospodársky stroj je zatiaľ niečím úplne výnimočným. Aktuálne sa využíva satelitná navigácia, ktorá zabezpečuje pohyb podľa GPS. Takýto systém nevypočítava optimálnu trasu, ale v reálnom čase ukazuje ako ma farmár držať líniu aby neprechádzal viac krát cez to isté miesto.

Obrázek 4.1: Základná navigácia Topcon XD

[1] dôležité vzhľadom na problematiku obratu

4.2 Parametre stroja ovplyvňujúce trajektóriu

Pri algoritmickom obrábaní pôdy je dôležité poznať parametre vozidla. V tejto práci zohráva najväčšiu úlohu šírka záberu. Táto premenná zásadne ovplyvňuje výslednú trasu. Šírka záberu sa líši podľa vykonávanej činnosti. Aj napriek tomu som je zvolená defaultna hodnota. Taktiež pomáha definovať spôsob otočenia sa v poli.

Ďalším parametrom je správna výška podvozku vozidla, ktorý umožňuje spracovávať plodiny bez poškodenia. Každá plodina má špecifické nároky na výšku podvozku stroja, ktorý ju obrába. V prípade, že je terén príliš členitý, môže byť klasický traktor nahradený pásovým. Ich výhoda spočíva v stabilite, no problematická býva práve výška podvozku. Výhodou pásových traktorov je aj nižšia váha a navyše je lepšie rozložená, tympádom sa znižuje tlak na pôdu. Pásove vozidla sú vhodné na polia veľkým sklonom, kedže pásy znižujú riziko bočného posunu.

Ak stroj vezie nebezpečné chemikálie, najmä agresívnejšie formy pesticídov, tak potom podlieha prísnejším pravidlám pri ich aplikácií. Pri použití v poľnohospodárstve je potrebné dbať na zvýšenú opatrnosť, aby sa nedostali účinné látky do vodných zdrojov. Stroj s chemikáliami by sa mal vyhnúť miestam, de by bolo možné ľahko chemikálie rozšíriť. [10]

V tejto práci je riešený pohyb stroja v uzatvorenom priestore, ktorý má presné hranice. V prípade, že je potrebné prejsť viacerými poliami, tak ne nepredpokladá, výskyt rizikovej oblasti.

Kapitola 5

Cieľe a požiadavky na výsledný algoritmus

Táto kapitola opisuje hlavné myšlienky pre vznik algoritmu. Uvážené sú taktiež enviromentálne opatrenia, ktoré majú svoje pevné miesto aj v Štátnej politike ochrany prírodného prostredia. Ide o legislavívne opatrenie, ktoré podporuje udržateľnosť a inovácie.

5.1 Zber celej úrody

Jeden z hlavných požiadavkov na algoritmus, je zber 100 % zasiatej úrody. To znamená, že ak je to možné, malo by byť pokrytie 100 percentné. Tento požiadavok kladie aj Európská únia v prípade poľnohospodárskych dotácií za účelom minimalizovať plytvanie potravinami. Jeden zo zaužívaných spôsobov je použitie pesticídov, najmä herbicidov a insekticidov Tieto látky majú negatívny vplyv nielen na zdravie človeka, ale aj na ekosystém krajiny. Táto problematika je rozoberaná na mnohých konferenciách a preto k nej existuje aj veľa štúdií. Minimalizácia ich použitia môže byť dosiahnutá aj vďaka vhodnému algoritmu, ktorý optimalizuje trasu tak, aby vozidlo striekajúce pesticídy, neprejde jedným úsekom viac krát, ale práve iba raz.

5.2 Vstupné a výstupné body

Ďalším požiadavkom na výsledný algoritmus je možnosť nastavenia vstupných a výstupných bodov. Toto nastavenie je dôležité, aby vozidlo zbytočne nejazdilo dlhé trate. Náklady na ľudské zdroje nie sú často krát malé a ani cena za stratené palivo. Tieto náklady by mohli byť na veľkých poliach markantné.

5.3 Zamedzenie erózie

Zamedziť eróziu je možné pridaní 3D informácie. Od riešenia v tretej dimenzii sa očakáva vyššia efektivita pohybu stroja. V prípade silného sklonu terénu je možné ušetriť palivo vhodne zvoleným smerom prejazdu. Avšak je stále potrebné dodržiavať empiricky overené postupy práce. Jedným z týchto príkladov je orba po vrstevnici. V štúdiach je veľmi často riešený iba najkratší prejazd a nezohľadňuje sa potreba pokrytia poľa[19].

5.4 Minimalizácia dĺžky trajektórie

Najväčšou výhodou pri minimalízacií dĺžky trajektórie je zníženie spotreby paliva, ktoré by bolo spotrebované pri prejazde. Okrem paliva môžu byť znížené aj ostatné náklady, ako napríklad prejazdový čas alebo ľudská práca.

Skrátenie dĺžky trajektórie na možné minimum avšak nesmie kolidovať s princípmi ochrany pôdy proti erózií. V závislosti od situácie nemusí byť stále najkrajšia trasa vyhodnotená ako vhodná.

5.5 Vyhnutie sa prekážkam v poli

Ďalším dôležitým bodom je vyhnutie sa stabilným prekážkam v poli. Často krát nie je pole homogénne a jeho súčasťou bývajú aj stabilné pevné objekty. Tieto objekty sú vnímané ako prekážky.

Ako prekážku si môžeme predstaviť prírodný objekt, ktorý nie je pominuteľný, alebo ide o objekt antropogenného pôvodu (napríklad stĺp vysokého napätia) a nie je možné ho presunúť. Vo vstupných dátach by mali byť definované ako diery vyčlenené z plochy.

Kapitola 6

Návrh

Táto kapitola sa zaoberá návrhom algoritmom, ktorý bude implementovaný. Vychádza z algoritmov autorov spomínaných v predchádzajúcich kapitolách. Základným vstupným parametrom je súbor v GeoJSON formáte[1]. Tento formát definuje niekoľko typov JSON objektov, ktoré sú definované takým spôsobom, aby reprezentovali geografické dáta a ich vlastnosti [5].

6.1 Dekompozícia terénu

Často krát je obtiažne nájsť jeden optimalizovaný vzor plánovania trasy pre pokrytie celého poľa. Preto je potrebné najskôr územie rozdeliť na menšie celky, pre ktoré bude trajektória vypočítavaná zvlášť. Rozdelenie poľa prebieha na základe atribútov ako napríklad vlastnosti terénu (sklon povrchu, výškový rozptyl) alebo pôdne podmienky. Pole by sa malo rozdeliť tak, aby vznikli jednoduché hranice kvôli eliminácii ďalších nákladov. Niekedy je potrebná rekombinácia susedných oblastí. Následne by sa mala pre každý región použiť najhodnejšia stratégia plánovania trasy, aby sa dosiahli minimálne náklady na pokrytie [11].

```
1  Algorithm TERRAINDECOMPOSITION(f)
2  Input: Planar subdivision f representing a field
3  Output: A~list of planar subdivisions, L, representing the sub-regions.
4         initialize L as containing f as the only item;
5         for i <- to n
6                 for each item R in L
7                         if (R should be divided according to the ith criterion)
8                                 then remove R from L,
9                                 and add the newly divided sub-regions into L;
10                        end if
11                end for
12        end for
13        search for all pairs of adjacent sub-regions
14        for each pair of adjacent sub-regions, R1 abd R2
15                if (combining the pair reduce the coverage cost)
16                        then remove R1 and R2 from L, and add the newly combined
17                        sub-region into L;
18                else go to line 13 to start over again;
```

[1] https://www.rfc-editor.org/info/rfc7946

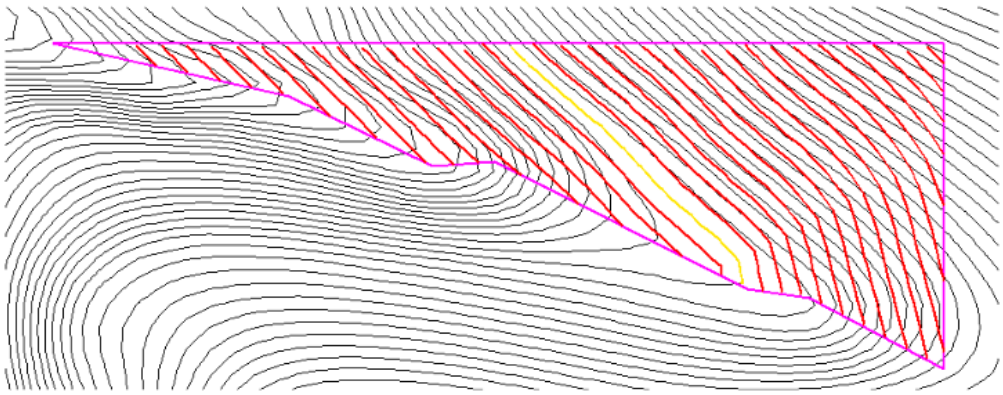
```
19                end if
20          end for
21          return L;
22
23  End of algorithm
```

6.2 „Seed curve“ vyhľadávací algoritmus

Pre každú časť poľa je konečným výstupom plánovania sada zakrivených ciest, ktoré sú vedľa seba. Skupina ciest môže byť určená jednou krivkou „seed curve“. Po jej nájdení môžu byť vygenerované zvyšné cesty tak, aby bol požadovaný povrch pokrytý celý. Pri plánovaní cesty je kľúčovým nájsť optimálnu *„seed curve“*, ktorá bude mať minimálne náklady. Vyhľadávací priestor akýchkoľvek „seed curve“ môže byť obrovský, čo znemožňuje vyhľadávanie. V prípade, že krivka je aproximovaná ôsmimi spojenými cestami, výpočtová zložitosť s dĺžkou n by bola $O(8^n)$. Preto sú potrebné heuristické metódy na zmenšenie vyhľadávacieho priestoru. V praxi sa najčastejšie používajú dve kategórie pre vyhľadanie optimálnej krivky:

- Vrstevnica
- Okraj poľa

Pre každú možnú konečnú krivku sa vypočíta cena pokrytia. Výsledná krivka má najlepšiu cenu. Z výslednej krivky sú spravené rovnobežky tak, aby bolo celé územie pokryté [11].



Obrázek 6.1: Príklad výslednej kviky. Na základe zlatej krivky sú spravené rovnobežky (červené), aby bolo celé územie pokryté. Prevzaté z [11]

6.3 Obchádzanie prekážok

Ako prekážku je možné si predstaviť prírodný objekt, ktorý má takú veľkosť, že nemôže byť zanedbateľný. Motivácia pre tento bod návrhu je opísaná v sekcii 5.5.

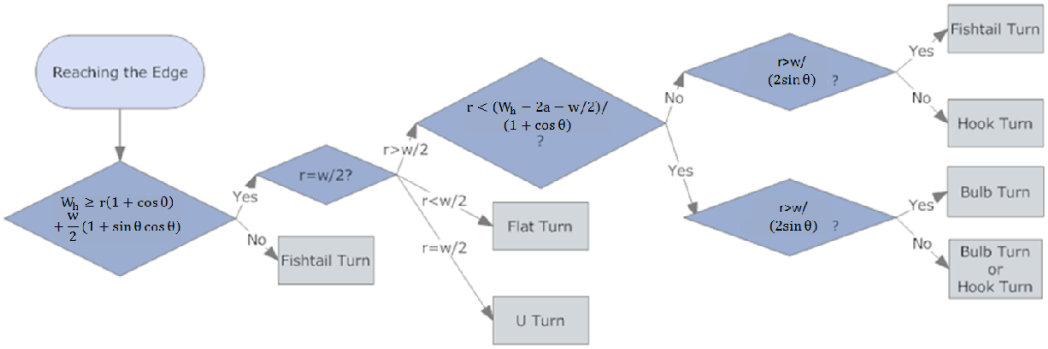
Prekážku je možné spracovať ako presne definovaný objekt. Prax ukázala, že najlepšou možnosťou je prekážku definovať ako kruhový objekt, kedy radián bude rovný priemeru objektu v jeho najširšom rozpätí [3]. K tomu je nutné pripočítať bezpečný odstup, ktorý bude predom definovaný. Navrhnutý bol následovný algoritmus:

```
Algorithm around_obstacle()
Input: obstacle
       divide obstacle into 2 trajectories like no obstacle in path
       if any side is not cultivated:
              go around to closer side of cultivated field
       else:
              go around tofurther side of cultivated field
End of algorithm
```

6.4 Algoritmus výberu otočenia

Výber správneho typu otočenia je dôležitý nie len kvôli úspore paliva a minimalizácie trajektórie. Dôležité sú aj faktory, ktoré nevieme ovplyvniť. Medzi tieto faktory patrí šírka riadku, minimálny polomer otočenia vozidla, voľný priestor a uhol medzi riadkom a okrajom.

Návrh rozhodovacieho stromu:



Obrázek 6.2: Rozhodovací strom pre výber druhu otočenia. Prevzaté z [11]

Kapitola 7

Implementácia

Táto kapitola sa zaoberá implementáciou plánovacieho algoritmu, ktorý vychádza z návrhu riešenia. Pri riešení problému, vzniklo viacero komplikácií. Jednou z nich je nevyužitie robotického operačného systému.

7.1 Použité technológie

Program je implementovaný v jazyku python3.8 s využitím knižníc shapely, geopandas a matplotlib.

Geopandas

Geopandas je open-source projekt, ktorý slúži na zjednodušenie práce s geopriestorovými údajmi v jazyku python. Rozširuje dátové typý používané v knižnici pandas tak, aby boli umožnené priestorové operácie na geometrických typoch. Taktiež využíva knižnicu shapely. Zjendodušuje využívanie takých operácii v pythone, ktoré by inak na ich uskutočnenie potrebovali priestorovú databázu, napríklad PostGIS.

Hlavnými dátovými typmi, ktoré implementuje sú GeoSeries a GeaDataFrame, ktoré sú vlastne podtriedami pandas.Series a pandas.DataFrame. Taktiež umožňuje jednoduchú prácu s geometrickými objektami, nad ktorými vie prevádzať geometrické operácie. GeoSeries je v podstate vektor, kde každá položka vo vektore je súprava tvarov zodpovedajúca jednému pozorovaniu. Záznam môže pozostávať iba z jedného tvaru alebo viacerých tvarov, ktoré sa majú považovať za jedno pozorovanie. Vhodným príkladom je mnoho polygónov ktoré tvoria štát.

GeodataFrame je tabuľková dátová štruktúra, ktorá obsahuje GeoSeries. Najdôležitejšou vlastnosou je stĺpec označovaný ako „geometry“, ktorý má špeciálny stav. V prípade, že sa zavolá nejaká priestorová metóda, tak sa aplikuje práve na tento stĺpec. [13]

Matplotlib

Matplotlib je knižnica, ktorá slúži na vizualizáciu dát v jazyku python. Aj keď má svoj pôvod v emulácií príkazov v prostredí MATLAB, je od tohto prostredia nezávislá. Je možné vytvárať statické animované aj interaktívne vizualizácie aj po zadaní malého počtu príkazov. [9]

Hlavné výhody knižnice:

- Multiplatformné riešienie, spustiteľné na operačných systémoch Windows a Linux

- Jednoduchá syntax
- Rýchlosť a flexibilita
- Open-source knižnica

Shapely

Deterministická priestorová analýza je dôležitou súčasťou výpočtových prístupov k problémom v poľnohospodárstve, ekológii, epidemiológií a mnohých ďalších oblastiach. Python balík shapely poskytuje funkcie na analýzu a manipuláciu s rovinnými prvkami. Využíva funkcie zo známej knižnice GEOS, ktorý je motorom rozšírenia PostGIS [1] pre PostgresSQL. Knižnica shapely je využívaná najmä v geografických informačných systémoch. V knižnici sú zachované štandardné triedy a operácie.

Základné typy geometrických objektov implementovaných balíkom sú:

- Bod - implementovaný triedou `Point`
- Krivka - implementovaný triedami `LineString` a `LinearRing`
- Povrch - implementovaný triedou `Polygon`

Shapely nepodporuje transformácie súradnicových systémov. Všetky operácie predpokladajú že objekty sa nachádzajú v rovnakom súradnicovom systéme. [7]

7.2 Vstupné dáta

Je dôležité, aby si používateľ mnoho stiahnúť vstupné dáta v potrebnom formáte, resp. potrebný formát si vytvoriť. Tvorba vstupného súboru by mala byť čo najjednoduchšia a najrýchlejšia. Jednotlivé open source nástroje neobsahujú všetky potrebné dáta a z tohto dôvodu som zvolila open-source nástroj JOSM, ktorý tvorbu značne uľahčí.

Nástroj JSOM

Open-source nástroj JOSM slúži na úpravu pracovnej plochy pre geografické dáta OpenStreetMap vytvorený v prostredí Java8+. Je podporovaný na všetkých operaćných systémoch a má GPL licenciu [2]. Umožňuje funkcie, ktoré nie sú v online editore OpenStreetMap dostupné. Na území Českej republiky sú jednotlivé poľnohospodárske parcely importované z verejného registra pôdy - LPIS. Pre zobrazenie týchto dát priamo V OpenStreetMap by musel byť použitý doplnok, ale JSOM automaticky tieto dáta importuje. Editor umožňuje vytvorenie a definovanie objektov. [1]

Editor umožňuje nie len vložiť vlastný *.osm súbor, ale aj stiahnuť dáta priamo z OpenStreetMap. Dáta môžu byť následne uložené v rôznych formátoch ako napríklad .json, .geojson alebo .osm.

Členenie dát v rámci verejného registra pôdy:

1. Geografické - na základe katastrálneho územia.
2. Obsahové

[1] `http://www.refractions.net/products/postgis/`
[2] `https://www.gnu.org/licenses/gpl-3.0.html`

- entita - jedná sa o varianty: Diel pôdneho bloku, ekologicky významný prvok, pôdny blok.
- typ - XML súbor, historické dáta, súbor obsahujúci geopriestorovú definíciu entity

3. Časové - dátum ku ktorému sú dáta vytvorené

[2]

Chýbajúcim, no celkom potrebným údajom, je informácia o vrstevniciach. Tá sa získava pomocou `phyghtmap`, ktorá ich vygeneruje z serveru NASA SRTM [3].

Ďalším chýbajúcim údajom sú prekážky. Pole je definované ako polygón s množinou bodov v súradnicovom systéme EPSG:3857. Vnútorný objekt nie je definovaný v rámci daného polygónu. Vstupný súbor je vygenerovaný pomocou JOSM.

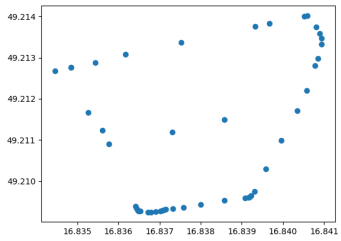
Ukážka vstupného .json súboru:

```
{
    "type": "FeatureCollection",
    "generator": "JOSM",
    "features":[
            {
            "type": "Feature",
            "properties": {
                "landuse": "farmland",
                "ref": "9810140",
                "source": "lpis"
            },
            "geometry": {
                "type": "Polygon",
                "coordinates": [
                    [
                        [
                            16.83674540000,
                            49.21556090000
                        ],
                        [
                            16.83731550000,
                            49.21590720000
                        ],
                        [
                            16.83674540000,
                            49.21556090000
                        ]
                    ]
                ]
            }
        },
    ]
}
```

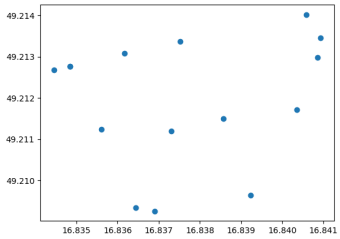
[3] https://www2.jpl.nasa.gov/srtm/

7.3 Zjednodušenie hraníc poľa

Vstupné údaje definujú pole ako polygón, ktorý je tvorený viacerými bodmi. Pre účely algoritmu je potrebné hranice zjednodušiť, aby vznikli čo najdlhšie úsečky, z ktorých vznikne výsledná trasa. Na zjednodušenie bola použitá funkcia `object.simplify(tolerance, preserve_topology=` z knižnice shapely s toleranciou 0.0001. Využíva sa rýchly Douglas-Peucker algoritmus, ktorý redukuje počet bodov potrebných na reprezentáciu objektu, resp. jeho karikatúry. [7]



(a) Znázornenie vertexov pred zjednodušením.



(b) Znázornenie vertexov po zjednodušení.

Obrázek 7.1: Porovnanie zjednodušenia hranice pola

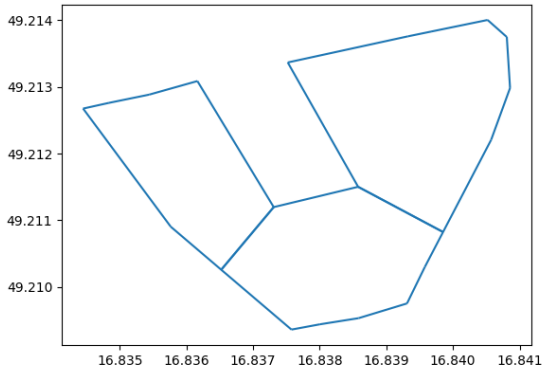
7.4 Dekompozícia terénu

Algoritmus na dekompozíciu terénu, ktorý je spomínaný v predchádzajúcej kapitole musel byť pozmenený z dôvodu neodstatočných informácií o teréne. Verejný register pôdy neobsahuje podrobné informácie o výškovom rozptyle a sklone povrchu. Tieto dáta by bolo možné získať pomocou digitálneho modelu terénu, ktorý popisuje reálny povrch. Jedným zo spôsobov ako by sme mohli získať podrobnejšie informácie o sklone je pomocou lineárne interpolačného algoritmu. Ten vychádza zo skutočnosti, že spád terénu medzi dvoma bodmi, medzi ktorými je interpolácia prevádzaná, je konštantný. Síce je táto metóda jednoduchá a výpočetne rýchla, no reálny priebeh vrstevníc vystihuje vo väčšine prípadov nevhodne. Z toho dôvodu je implementovaný následovný algoritmus:

```
1  Algorithm TERRAINDECOMPOSITION(f)
2  Input: Planar subdivision f representing a field
3  Output: A~list of planar subdivisions, L, representing the sub-regions.
4         for every 3 following vertexes
5                 if angle > 180:
6                         find nearest edge line
7                         make vertical line to this edge line
8                         create f1
9                         create f2
10
11                        return TERRAINDECOMPOSITION(f1) + TERRAINDECOMPOSITION(f2)
12
13                end if
14        end for
15        L.append(f)
```

```
16          return L
17          End of algorithm
```

Ak vnútorný uhol medzi troma po sebe idúcimi bodmi je väčší ako 180 tak potom je potrebné pole dekomponovať, tak aby vznikol konvexný tvar. Nájde sa najbližšia hrana poľa, a spraví sa kolmica. Na obrázku 7.2 je možné vidieť rozdelenie polygónu na tri menšie polygóny, ktoré sa budú obrábať samostatne.



Obrázek 7.2: Znázornenie dekompozície terénu.

7.5 Voľba štartu

Používateľ si môže zvoliť štartovaciu pozíciu. Od tej je nájdená najbližšia vrstevnica, ktorá symbolizuje najvyšší bod v rámci areálu. Následovne je vyhľadaný najbližší vertex polygónu, ktorý je výslednou štartovacou pozíciou pre vozidlo.

7.6 Obchádzanie prekážok

Vzhľadom na nedostatočné vstupné dáta nebola táto funkcionalita implementovaná. Chýbajúcim údajom sú prekážky, ktoré nie sú znázornené na poli. Vrámci práce bol vymyslený pseudokód algoritmu spomínaný v kapitole 6.3.

7.7 Výber vhodného otočenia

O dôležitosti výberu vhodného otočenia je písané v kapitole 3.3. Rozhodovací strom výberu vhodného typu obratu je znázornený na obrázku 6.2 Odporúčané otočenie je vypísané na štandarný výstup, v prípade, že uživateľ použije prepínač `-turns`.

7.8 Výsledná cesta

Uživateľ musí nastaviť pole resp. polia, pre ktoré sa vygeneruje cesta. Vrámci Českej republiky má každé pole definované referenčné číslo, ktoré slúži ako identifikátor poľa. Je možné nastaviť viacero polí, pre ktoré bude vygenerovaná cesta. Na toto nastavenie slúži argument

`-fields="ref_num1 ref_num2"`, resp. je možné použiť skrátenú verziu `-fie="ref_num1 ref_num2"`

Výsledný algoritmus pozostáva z algoritmov vo vyššie spomínaných kapitolách. Po spustení programu sa najskôr spracujú argumenty. Následne prebehne zjednodušenie hraníc a dekompozícia. Potom sa uživateľovi zobrazí graf, kde musí zvoliť štartovaciu pozíciu, pre ktoré má byť vygenerovaná výsledná cesta. Pre každé dekomponované pole je cesta naplánovaná zvlášť. Koncový bod predchádzajúceho poľa je zároveň aj začiatočným bodom pre nasledujúce pole, resp. pre nasledujúcu časť poľa.

Pomocou prepínača `-min_length` je možné zmeniť pomery nákladových funkcií. V tomto pripade sa bude minimalizovať dĺžka trajektórie a nie počet obratov. Defaultne je minimalizovaný počet potrebných obratov.

Kapitola 8

Testovanie a diskusia výsledkov

Algoritmus bol testovaný na viacerých poliach. Testy spočívali v porovnani výslednej trajektórie a reality. Ako je pole obrábané v realite je možno vidieť na satelitných snímok máp. Niektoré výsledky exporimentov sa líšia. Odchylku zapríčiňuje viacero faktorov. Jedným z faktorov je nedostatoćná informácia o výškovom rozptyle poľa. Istú ochylku spôsobuje aj neobchádzanie prekážok, vzhľadom na nedokonalé vstupné dáta. Ďalšou možnou príčinou je štartovací bod, ktorý ovplyvňuje výslednú trajektóriu.

Test algoritmu na jednoduchom poli

Ako prvý testovací objekt bolo zvolené pole v okolí Rousinova s referenčným číslom 1558492. Dané pole má jednoduchý konvexný tvar, takže nie je potrebná dekompozícia. Ako štartovací bod bol zvolený bod ktorý leží najbližšie pri ceste. Najskôr bol test prevedený bez ohľadu na vrstevnice. Výsledok testu sa nezmenil, ani keď boli brané vrstevnice do úvahy, vzhľadom na to, že výškový profil je minimálny.

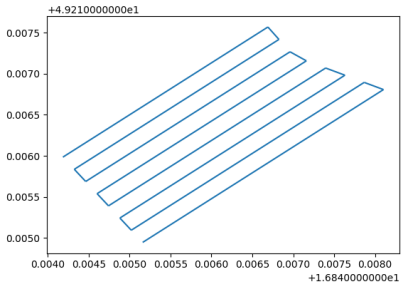
(a) Satelitná snímka daného poľa. (b) Výsledná trajektória algoritmu.

Obrázek 8.1: Porovnanie trasy na jednoduchom poli v okolí Rousinova.

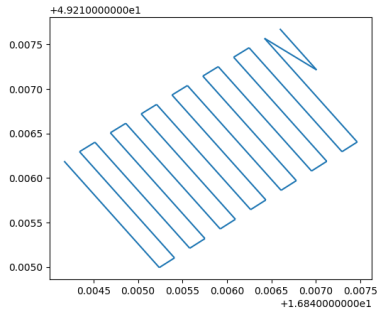
Test minimalizácie nákladov

Test je zameraný na minimalizáciu nákladov. V tomto prípade sú náklady chápané ako počet pootrebných otočení sa a dĺžky trajektórie. Na obrázku 8.2a algoritmus minimalizuje počet obratov. Na obrázku 8.2b algoritmus minimalizuje dĺžku trajektórie. Tento test bol

prevedení na viacerých poliach. Kvôli poznatkom z praxe bola a aj na základe týchto testov bola nastavená defaultna hodnota nákladovych funkcií ako 0:1, pričom väčšiu váhu ma minimalizácia počtu obratov.

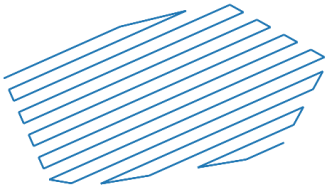


(a) Minimalizácia počtu obratov

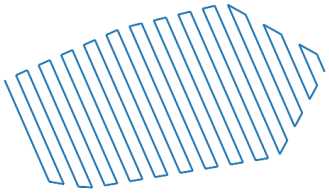


(b) Minimalizácia dĺžky trajektórie

(c) Satelitná snímka



(d) Trasa s minimálnym počtom otočení

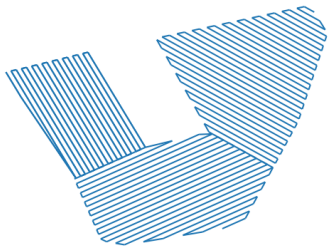


(e) Trasa s minimálnou dĺžkou trajektórie

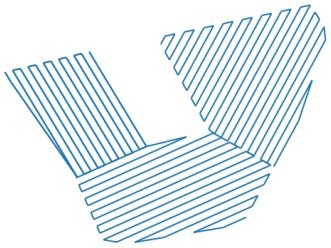
Obrázek 8.2: Porovnanie výslednej trasy pri zmene váhy nákladových funkcií

Test algoritmu zložitejšieho tvaru poľa

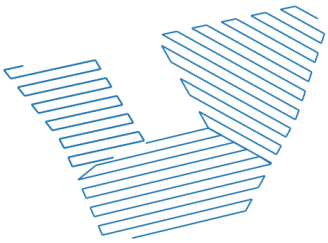
Algortimus je otestovaný aj pre koplexnejšie pole, ktoré je potrebné rozdeliť na menšie celky, pre ktoré bude trasa vypočítaná samostatne. Na obrázku 8.3c je vidieť podobnú trasu ako sa používa v praxi, čo je znázornené na obrázku 8.3d Výsledná trajektória závisí najmä od štartovacieho bodu.



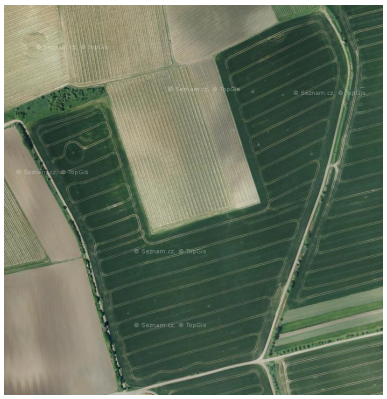
(a) Trajektória po dekompozícií, šírka 2.0



(b) Trajektória po dekompozícií, šírka 3.0



(c) Trajektória po dekompozícií, šírka 3.0, iný začiatočný bod

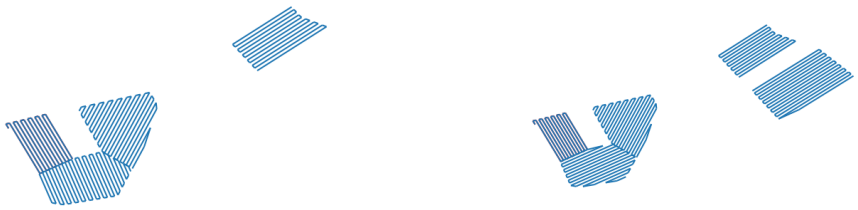


(d) Satelitná snímka

Obrázek 8.3: Plánovanie trasy pokrytia na zložitejšom poli

Test priechodu medzi viacerými poľami

Test je zameraný na plánovaní trasy pokrytia povrchu, v prípade, že užívateľ zadá viacero polí. Test bol prevedený pre dve a tri polia. Algortimus neslúži ako navigácia medzi poľami, takže nie je možné nájsť vo výslednom výstupe cestu medzi nimi. Motiváciou nenaplánovať cestu pre autonómny poľnohospodársky stroj medzi poliami je zákon, ktorý neumožňuje pohyb plne autonómnych vozidiel na verejnej komunikácií. Ďalším dôvodom je, že vo vstupných dátach nie sú definované poľné cesty, po ktorých sa bežne traktory presúvajú. Pre každé z polí sa vypočíta trasa pokrytia samostatne a následne je vykreslená na spoločnom grafe. Cieľový bod naposledy obrobeného poľa slúži ako štartovacia pozícia pre nasledujúce pole. Od tohto bodu je vyhľadaný najbližší hraničný bod.



(a) Plánovanie trasy na dvoch poliach (b) Plánovanie trasy pre tri polia

Obrázek 8.4: Plánovanie trasy pokrytia viacerých poli

8.1 Diskusia výsledkov

Optimalizovaný pilotný algoritmus je schopný spracovať trajektóriu z verejne dostupných dát, zároveň však behom testovania vznikli isté nedostatky algoritmu. Jedným z nedostatkom je dekompozícia objektu, ktorý v istých prípadoch rozdeľuje objekt na príliš malé časti, ktoré by v praxi nebolo možné obrobiť. Tento problém by mohol byť v ďalšej práci vyriešený pomocou rekombinácie susedných regiónov.

Ďalším problémom je chýbajúca informácia o prekážkach. Tým by sa trajektória zmenila a spresnila. Najlepšou možnosťou by bolo geodeticky zmerať prekážku, čím by dáta boli presné.

Podmienkou pre úspešný vývoj finálnej podoby algoritmu je testovanie na skutočných fyzických strojoch na poli. Pre takéto testovanie je nezbytné algoritmus upraviť aby, trasa mohla byť plánovaná v reálnom čase.

Ako nadstavba algoritmu by mohli byť vytvorené váhy pre jednotlivé funkcie a parametre v programe. koncový užívateľ by mohol zadávať vlastné váhy k funkciám a tým by sa trajektórie líšili. takýmto spôsobom by bolo momžné do algoritmu zakomponovať napríklad typ plodiny, ktorú chce poľnohospodár na danom mieste pestovať.

Zamedzenie erózií je v práci riešené len na základe informácie o vrstevniciach. V prípade, žeby boli k dispozícií presnejšie informácie o pôde, môže byť implementovaná funkcia na zamedzenie erózie. Podrobnejšie informácie o zamedzení erózie pôdy je možné nájsť v kapitole 3.1.

Kapitola 9

Záver

Cieľom tejto bakalárskej práce bolo naštudovať si plánovanie cesty a algoritmy pre pokrytie priestoru zamerané pre poľnohospodársky priemysel. Ďalej navrhnúť a implementovať program, ktorý umožní z externého súboru nahrať tvar oblasti a naplánuje cestu na pokrytie. Návrh a implementácia vychádzajú z algoritmov a štúdií predstavených v kapitolách 6 a 7. Implementácia vychádza z praktických poznatkov nie len overených matematický postupov pre optimalizovaný pohyb, ale aj z poľnohospodárskej praxe

Niektoré cieľové požiadavky na algoritmus neboli implementované vzhľadom na neodstatočné vstupné dáta. Medzi tieto cieľové požiadavky patrí obchádzanie prekážok.

Vrámci práce bol algoritmus porovnávaný s praxou využívanou v poľnohospodárskom priemysle. Porovnávala sa výsledná cesta implementovaného algoritmu a satelitné snímky, na ktorých je vidieť trasu vozidla. Trasy môžu byť rozdielne z viacerých dôvodov.

Algortimus má podobu pilotného návrhu, možnosti úpravy a rozšírení sú popísane v diskusií výsledkov 8.1.

Literatura

[1] *Introduction to JOSM* [online]. Dostupné z: https://josm.openstreetmap.de/wiki/Introduction.

[2] *Veřejný export dat LPIS* [online]. Dostupné z: http://eagri.cz/public/web/mze/farmar/LPIS/uzivatelske-prirucky/prirucky-pro-verejny-lpis/export-dat-lpis.html.

[3] Biglarbegian, M. a Al-Turjman, F. *Path planning for data collectors in Precision Agriculture WSNs.* 2014, s. 483–487. DOI: 10.1109/IWCMC.2014.6906404.

[4] Bičík, I. a Jančák, V. *Transformační procesy v českém zemědelství po roce 1990.* Atlantis Press, 2005. ISBN 80-86561-19-4.

[5] Butler, H., Daly, M., Doyle, A., Gillies, S., Hagen, S. et al. *The GeoJSON Format.* RFC 7946. RFC Editor, August 2016. Dostupné z: https://www.rfc-editor.org/info/rfc7946.

[6] Cao, Z. L., Huang, Y. a Hall, E. Region filling operations with random obstacle avoidance for mobile robots. *Journal of Robotic Systems 5(2):87 - 102.* April 1988. DOI: 10.1002/rob.4620050202. Dostupné z: https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.4620050202.

[7] Gillies, S. et al. *Shapely: manipulation and analysis of geometric objects.* 2007–. Dostupné z: https://github.com/Toblerity/Shapely.

[8] Helsgaun, K. *An Effective Implementation of K-opt Moves for the Lin-Kernighan TSP Heuristic.* Roskilde Universitet, 2006. Roskilde Universitet. Computer Science. Computer Science Research Report.

[9] Hunter, J. D. Matplotlib: A 2D graphics environment. *Computing in Science & Engineering.* IEEE COMPUTER SOC. 2007, sv. 9, č. 3, s. 90–95. DOI: 10.1109/MCSE.2007.55.

[10] Ing. Karel Pepperný, P. *Požadavky na pokusy s přípravky na ochranu rostlin - rezidua.* 2017.

[11] Jin, J. *Optimal field coverage path planning on 2D and 3D surfaces.(2009).* Disertační práce.

[12] Jin, J. a Tang, L. Optimal Coverage Path Planning for Arable Farming on 2D Surfaces. *Transactions of the ASABE.* 2010, sv. 53, s. 283–295. Dostupné z: https://lib.dr.iastate.edu/abe_eng_pubs/336.

[13] Jordahl, K., Bossche, J. V. den, Fleischmann, M., Wasserman, J., McBride, J. et al. *Geopandas/geopandas: v0.8.1.* 2020. Dostupné z: https://doi.org/10.5281/zenodo.3946761.

[14] Marada, P., Havlíček, Z. a Skládanka, J. *Ochrana přírody a krajiny - Ekosystémové služby, nový trend zemědělského podnikání.* 1. vyd. Mendelova univerzita v Brně, 2010, s. 45. ISBN 978-80-7375-416-7.

[15] Meuth, R. J. a Wunsch, D. C. *Divide and conquer evolutionary TSP solution for vehicle path planning.* 2008, s. 676–681. DOI: 10.1109/CEC.2008.4630868.

[16] Noguchi, N. a Terao, H. Path planning of an agricultural mobile robot by neural network and genetic algorithm. *Computers and Electronics in Agriculture.* 1997, sv. 18, s. 187–204.

[17] Oksanen, T. a Visala, A. *Coverage Path Planning Algorithms for Agricultural Field Machines.* 2009, sv. 26. DOI: 10.1002/rob.20300.

[18] Rezník, T., Charvát, K., Lukas, V., Jr., K. C., Horáková Šárka et al. *Open Data Model for (Precision) Agriculture Applications and Agricultural Pollution Monitoring.* Atlantis Press, 2015/09, s. 97–107. ISBN 978-94-62520-92-9. Dostupné z: https://doi.org/10.2991/ict4s-env-15.2015.12.

[19] Yan, H., Wang, H., Chen, Y. a Dai, G. Path planning based on Constrained Delaunay Triangulation. *2008 7th World Congress on Intelligent Control and Automation.* 2008, s. 5168–5173.

Přílohy

Příloha A

Manuál

- Vytvorenie vstupného súboru pomocou nástroja JOSM, ktorý nájdete voľne dostupný na stránke `https://josm.openstreetmap.de/`. V prípade, že má byť braný ohľad na sklon poľa, resp. vrstevnice je potrebné ich vygenerovať a pripojiť do *.json súboru. Možné využiť JOSM. Na generovanie vrstevníc môže byť použitý nástroj `pyhghtmap`, ktorý vezme vrstevnice z NASA SRTM a vráti ich ako OSM dáta. Podrobnosti môžete nájsť na stránke `https://wiki.openstreetmap.org/wiki/Phyghtmap#Using_phyghtmap`
- Inštalácia preprekvizít. `pip install -r requirements.txt`
- Spustenie programu: `python3.8 python3.8 main.py -file="*.json"-field="ref_num1 ref_num2"[-width=] [-radius=] [-contours] [-turns]`
  - `-file=` – cesta k súboru so vstupnými dátami. *.json
  - `-fields=` – identifikátor poľa, resp. viacero polí
  - `-width=` – určuje šírku vozidla, defaultna hodnota je 15.0
  - `-radius=` – určuje minimálny polomer otočenia vozidla, defaultna hodnota je 20.0
  - `-contours` – v prípade použitia prepínača bude výsledná trajektória braný zreteľ na sklon pola, to znamená, že sa zamedzí erózií pôdy
  - `-turns` – na štandarný výstup sú vypísané odporúčané typy obratov
  - `-min_length` – bude sa minimalizovať dĺžka výslednej trajektórie a nie počet obratov.
- Po vykreslení je potrebné kliknutím na grafe zvoliť štartový pozíciu, pre ktorú bude vygenerovaná optimálna cesta.
- Program sa ukončí zatvorením grafov.

Příloha B

Obsah CD

Priložené CD obsahuje:

- `xchrip01.pdf` – táto práca
- `tex` – adresár so zdrojovými kódmi
- `oop` – adresár so zdrojovými kódmi programu na plánovanie trasy
  - `main.py` – hlavný python súbor
  - `parser.py` – python súbor, ktorý načíta uživateľský vstup.
  - `field.py` – python súbor, s triedou `FieldDecomposition`.
  - `seed_curve.py` – python súbor s triedou `Seed`.
  - `fnc.py` – python súbor pomocnými funkciami.
  - `requirements.tx` – textový súbor prerekvizít
  - `vinicne_sumice.json` – ukážkový vstuný súbor