VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ

FACULTY OF MECHANICAL ENGINEERING

ÚSTAV MECHANIKY TĚLES, MECHATRONIKY A BIOMECHANIKY

INSTITUTE OF SOLID MECHANICS, MECHATRONICS AND BIOMECHANICS

SEGMENTACE 2D POINT-CLOUDU PRO PROLOŽENÍ KŘIVKAMI

2D POINT-CLOUD SEGMENTATION FOR CURVE FITTING

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. Marek Šooš**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. Jan Králík**
SUPERVISOR

BRNO 2021

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA STROJNÍHO INŽENÝRSTVÍ

Zadání diplomové práce

Ústav:	Ústav mechaniky těles, mechatroniky a biomechaniky
Student:	**Bc. Marek Šooš**
Studijní program:	Aplikované vědy v inženýrství
Studijní obor:	Mechatronika
Vedoucí práce:	**Ing. Jan Králík**
Akademický rok:	2020/21

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

Segmentace 2D Point–cloudu pro proložení křivkami

Stručná charakteristika problematiky úkolu:

Při mapování oblasti pomocí 2D lidaru je základním výstupem point cloud (mrak bodů), z hlediska velikosti a zpracování je však místy výhodnější pracovat s křivkami, než s mrakem bodů. V současnosti existuje několik způsobů, jak lze point cloud rozdělit na jednotlivé segmenty. Každé řešení má však jiné výkonnostní nároky, úspěšnost a možnosti segmentace.

Cíle diplomové práce:

1. Proveďte rešerši v oblasti segmentace point–cloudu.
2. Vybraná řešení naprogramujte.
3. Naprogramované řešení otestujte na reálných datech.
4. Porovnejte výsledky jednotlivých řešení.
5. Porovnejte výsledky i pro proložené křivky a jak moc metoda segmentace ovlivňuje výsledné proložení.

Seznam doporučené literatury:

BLANCHET, Gerard, CHARBIT, Maurice. Digital signal and image processing using Matlab. NewportBeach, CA: ISTE, 2006. ISBN 978-1-905209-13-2.

ECKART, Benjamin, Kihwan KIM a Jan KAUTZ, 2018. HGMR: Hierarchical Gaussian Mixtures for Adaptive 3D Registration. Computer Vision – ECCV 2018. Cham: Springer International Publishing, 2018-10-07, 730-746. Lecture Notes in Computer Science. ISBN 978-3-030-01266-3. Dostupné z: doi:10.1007/978-3-030-01267-0_43.

Termín odevzdání diplomové práce je stanoven časovým plánem akademického roku 2020/21

V Brně, dne

L. S.

prof. Ing. Jindřich Petruška, CSc. ředitel ústavu	doc. Ing. Jaroslav Katolický, Ph.D. děkan fakulty

Abstrakt

Predložená diplomová práca sa zaoberá rozdeľovaním súradnicových bodov do homogénnych skupín. Práca poskytuje široký prehľad súčasného stavu v danej téme vo forme analýzy článkov ako aj stručné vysvetlenie princípov hlavných segmentačných metód. Z analýzy článkov je vybraná pätica algoritmov, ktorá je naprogramovaná. V práci sú opísané princípy vybraných algoritmov a vysvetlené ich matematické modely. Pre každý algoritmus je uvedený aj popis návrhu ich kódového spracovania. Diplomová práca taktiež obsahuje vzájomné porovnania schopností segmentácie jednotlivých algoritmov na vytvorených ako aj nameraných dátach. Výsledky porovnávania preloženia jednotlivých kriviek sú medzi sebou porovnané grafickou aj číselnou formou. Na konci práce sa nachádza porovnávací graf časovej závislosti výpočtového času od počtu bodov point cloudu a súčasne tabuľka zahŕňajúca vzájomné porovnania algoritmov v jednotlivých oblastiach.

Summary

The presented diploma thesis deals with the division of points into homogeneous groups. The work provides a broad overview of the current state in this topic and a brief explanation of the main segmentation methods principles. From the analysis of the articles are selected and programmed five algorithms. The work defines the principles of selected algorithms and explains their mathematical models. For each algorithm is also given a code design description. The diploma thesis also contains a cross comparison of segmentation capabilities of individual algorithms on created as well as on measured data. The results of the curves extraction are compared with each other graphically and numerically. At the end of the work is a comparison graph of time dependence on the number of points and the table that includes a mutual comparison of algorithms in specific areas.

Kľúčové slová

point cloud, segmentácia, preloženie kriviek, Lidar, rýchlosť algoritmu

Keywords

point cloud, segmentation, curve extrapolation, Lidar, algorithm performance

Bibliografická Citácia

ŠOOŠ,Marek. *Segmentace 2D Point-cloudu pro proložení křivkami*. Brno, 2021. Dostupné také z: `https://www.vutbr.cz/studenti/zav-prace/detail/132911`. Diplomová práce. Vysoké učení technické v Brně, Fakulta strojního inženýrství, Ústav mechaniky těles, mechatroniky a biomechaniky. Vedoucí práce: Ing. Jan Králik.

Prehlásenie

Prehlasujem, že táto práca je mojím pôvodným dielom, spracoval som ho samostatne pod vedením Ing. Jána Králika a s použitím literatúry uvedenej v zozname.

Marek Šooš

Brno . . . .

Poďakovanie

Pri tejto príležitosti sa chcem poďakovať môjmu vedúcemu diplomovej práce pánovi Ing. Jánovi Králikovi, za cenné rady a pripomienky v priebehu riešenia mojej diplomovej práce.

Marek Šooš

Obsah

Úvod **9**

1 Rešerš **10**
1.1 Lidar . . . 10
1.1.1 Všeobecný popis technológie . . . 10
1.1.2 Popis konkrétneho zariadenia . . . 13
1.2 Segmentácia . . . 14
1.3 Analýza súčasného stavu . . . 15
1.3.1 Metóda hrán . . . 15
1.3.2 Metóda postupného rastu . . . 17
1.3.3 Metóda geometrickej podobnosti . . . 20
1.3.4 Metóda grafovej segmentácie . . . 21
1.3.5 Metóda strojového učenia . . . 22
1.4 Záver analýzy článkov . . . 24

2 Použité algoritmy **25**
2.1 SEF algoritmus . . . 25
2.1.1 Matematický princíp algoritmu . . . 25
2.1.2 Navrhnutá štruktúra algoritmu . . . 26
2.1.3 Časová zložitosť algoritmu . . . 27
2.1.4 Vývojový diagram . . . 27
2.2 Line tracking algoritmus . . . 28
2.2.1 Matematický princíp algoritmu . . . 28
2.2.2 Navrhnutá štruktúra algoritmu . . . 29
2.2.3 Časová zložitosť algoritmu . . . 29
2.2.4 Vývojový diagram . . . 30
2.3 Split and Merge algoritmus . . . 31
2.3.1 Matematický princíp algoritmu . . . 31
2.3.2 Navrhnutá štruktúra algoritmu . . . 32
2.3.3 Časová zložitosť algoritmu . . . 33
2.3.4 Vývojový diagram . . . 34
2.4 Ransac algoritmus . . . 36
2.4.1 Matematický princíp algoritmu . . . 36
2.4.2 Navrhnutá štruktúra algoritmu . . . 38
2.4.3 Časová zložitosť algoritmu . . . 39
2.4.4 Vývojový diagram . . . 40
2.5 Hough algoritmus . . . 41
2.5.1 Matematický princíp algoritmu . . . 41

2.5.2 Navrhnutá štruktúra algoritmu 43
2.5.3 Časová zložitosť algoritmu 44
2.5.4 Vývojový diagram 45
2.6 Spracovanie dát 46
2.6.1 Preprocessing obdržaných dát 46
2.6.2 Post-processing segmentovaných dát 46

3 Výsledky **51**
3.1 Porovnanie algoritmov na vytvorených dátach 51
3.1.1 Porovnanie segmentácie bodov 51
3.1.2 Porovnanie preložených kriviek 56
3.1.3 Zhodnotenie k vytvoreným dátam 58
3.2 Porovnanie algoritmov na nameraných dátach 59
3.2.1 Porovnanie segmentácie 59
3.2.2 Porovnanie preložených kriviek 64
3.2.3 Zhodnotenie k experimentálnym dátam 66
3.3 Porovnanie časovej závislosti 66

Záver **68**

Literatúra **73**

Úvod

Ľudstvo sa od nepamäti snaží postupne vytvárať lepšie podmienky pre svoju existenciu. Kľúčom k zvyšovaniu životnej úrovne spoločnosti bolo od nepamäti získavanie nových poznatkov a ich následná technická aplikácia do skutočných produktov. V súčasnej dobe zastáva pozíciu najrozsiahlejšieho aplikátora nových vedeckých a technických poznatkov zrejme automobilový priemysel. Automobily sa vskutku považujú za najkomplexnejšie zariadenia našej doby, ku ktorým má neobmedzený prístup v podstate každý jedinec. Nakoľko sa jedná o jeden z najväčších priemyselných výrobkov vôbec, výsledky jeho pôsobenia sú výrazne reflektované a preto aj diskutované širokou verejnosťou.
Aktuálna snaha pri prechode na autonómne jazdiace vozidlá vyžaduje ostražité a spoľahlivé nasadenie nových technológií tak, aby boli pre zdravie človeka bezpečné. Pre ich správne výsledné fungovanie je nevyhnutný súlad viacerých hardvérových či softvérových aspektov.
Jedným z najčastejšie používaných senzorových zariadení v danej oblasti je technológia umožňujúca získavanie okolitej informácie zvaná Lidar. Častejšou aplikáciou daného zariadenia, a tiež jeho postupne sa znižujúce výrobné náklady tak prispievajú k rozšíreniu tejto technológie. Dôsledkom toho je zvyšujúci sa aj univerzitný výskum v oblasti samočinného riadenia a navigácie mobilných zariadení. Táto technológia poskytuje priestorový 2D obraz okolia pomocou súradníc. V súčasnosti je v tejto oblasti aplikovaný vedecký a projektový výskum komplexnej stavby autonómneho robota. Segmentácia, alebo spracovanie množiny bodov, zohráva dôležitú úlohu v procese spracovania dát a zastáva podstatný medzistupeň medzi obdržaným poľom bodov point cloudu a jeho následným vyhodnotením nevyhnutným pre navigáciu robota.
Cieľom predkladanej diplomovej práce je prehľad súčasného stavu algoritmických riešení pre problém segmentácie point cloudu. Priblížením jednotlivých metód k prístupu rozdeľovania bodov point cloudu prezentujem rôznorodosť výsledkov, ako aj výhody či nevýhody pre jednotlivé algoritmy. Z analýzy článkov je zrejmé, že v stave súčasného poznania neexistuje postup, ktorý by problém segmentácie riešil optimálne vo všetkých aspektoch.
Jadro mojej práce je sústredené na návrh a naprogramovanie vybranej pätice algoritmov. Pri svojich návrhoch spracovaných algoritmov som vychádzal z analýz súčasného poznania. V prezentovanej práci je každý navrhnutý algoritmus podrobne opísaný. Súčasne je opísaný jeho matematický model, či časové zhodnotenie daného algoritmu.
Výsledkom mojej práce je vzájomné porovnanie opisovaných algoritmov na vytvorených, ako aj skutočne nameraných bodov point cloudu. Z experimentálnej analýzy, vykonanej na rôznych úrovniach náročnosti bodov, boli vyhodnotené základné výhody a nevýhody jednotlivých navrhovaných algoritmov.
Vypracovaná práca poskytuje čitateľovi nie len vyhodnotenie v oblasti segmentácie bodov a preloženia kriviek, ale aj časovú závislosť spojenú so súhrnným prehľadom charakteristických vlastností navrhovaných algoritmov.

1 Rešerš

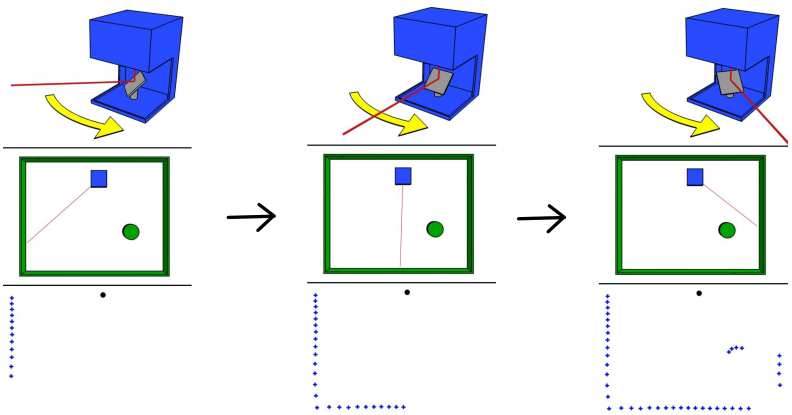
Kapitola rešerš je štrukturálne rozdelená do troch časti. V sekcií Lidar je vysvetlené fungovanie takéhoto zariadenia, jeho rôzne prevedenia či jeho využitie. Taktiež je v nej priblížené konkrétne používané zariadenie. V sekcií segmentácia je predstavená všeobecná myšlienka rozdeľovania bodov s popisom hlavných vlastností. Sekcia analýza súčasného stavu je štruktúrovaná do 5 predstavených segmentačných smerov. Každý z nich je príkladne vysvetlený, a pre každý z nich je zanalyzovaných niekoľko príkladoch algoritmických prístupov, použitých v rôznych prácach.

1.1 Lidar

Lidar, je zariadenie na mobilné snímanie vzdialeností a okolitého priestoru pomocou svetla vo forme laserového lúča. Získané informácie sa následne využívajú na vytvorenie obrazu okolia. Z Lidaru sa stala technológia určená na získavanie informácie vo forme hustého poľa dát o obraze okolia.

1.1.1 Všeobecný popis technológie

Lidar ("Laser imaging, detection, and ranging") môžeme nazvať aj senzorické zariadenie založené na princípe ToF (čas letu). Pri meraní sa využíva výpočtová metóda laserového zobrazovania a detekcie získavanie vzdialenosti senzoru od objektu [5], [11], [47].
Technológia podobná radaru využíva namiesto rádiových vĺn laserové pulzy, viď obr. 1.1. Radary dokážu oproti Lidarom prechádzať oblakmi, dažďom či hustou hmlou, [47]. Na základe oveľa väčšej vlnovej dĺžke rádiových vĺn ($10^3 m$) oproti svetelným vlnám ($10^{-9} m$), dokáže na druhú stranu Lidar zariadenie detegovať oveľa menšie objekty a s vyššou presnosťou, [45], [22]. Uvedená technológia bola v počiatkoch vytvorená a používaná najmä na skenovanie povrchu pomocou lietadiel. V súčasnosti je táto technológia veľmi používaná, kvôli jej schopnosti vytvárania vysoko presných a hustých bodov pre 3 rozmerný obraz okolia, [11], [5].



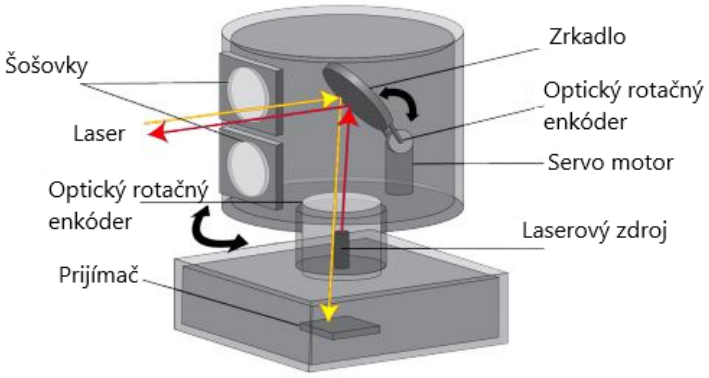
Obr. 1.1: Proces získavania point cloudu, prevzaté z [24]

Lidar vysiela lúč laserového svetla pričom si uchováva informácie o vlastnostiach každého lúča, ktorý vyšle. Vysoko intenzívny laserový lúč emitujúci Lidarom putuje priestorom, odráža sa od povrchu zachyteného telesa a vracia sa späť do Lidaru kde je zaznamenávaný. Za jednoduchú analógiu systému sa dá považovať loptička, odrážajúca sa predmetov a vracajúcu sa späť k zdroju, [42]. Meraním dĺžky časového rozdielu je následne vypočítavaná vzdialenosť, ktorú daný lúč prekonal a teda dvojnásobná vzdialenosť od objektu rovnica č. 1.1. Zo známej polohy a natočenia senzoru dokážu zistiť súradnice x,y,z, [18], [11], [47].
Konštrukciu príkladného Lidar zariadenia môžeme vidieť na obr. 1.2.

$$d = c * \frac{t}{2} \tag{1.1}$$

Kde

$d\ [m]$ je vzdialenosť od objektu
$c\ [\frac{m}{s}]$ je rýchlosť svetla
$t\ [s]$ je čas letu



Obr. 1.2: Konštrukcia Lidaru, prevzaté a upravené z [18]

Využitie tejto technológie sa nachádza v širokom spektre vedeckých disciplín. Optická šošovka nachádzajúca sa vo vnútri LiDAR zariadenia, má podobnú funkciu ako pri teleskope. LiDAR vysiela bilióny fotónov svetla pre zachytenie odrazených pár tisícov. Pre každých bilión generovaných fotónov, sa vráti tisíc odrazených späť do Lidar zariadenia. S narastajúcim počtom lúčov sa zvyšuje množstvo informácie, ktoré dokáže lidar zachytiť, [11].
Podľa základnej schopnosti vytvárania jedného a viac rozmerných priestorových obrazov, môžeme Lidary rozdeliť do 3 skupín :

1. Jednorozmerné 1D
2. Dvojrozmerné 2D
3. Trojrozmerné 3D

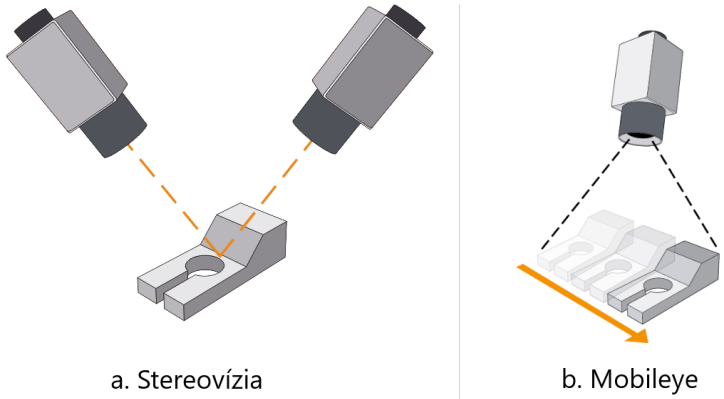
1D technológia Lidar zariadenia je najjednoduchšia, a v súčasnosti je využívaná na jednoduché meranie vzdialenosti v jednom smere - meter, [47], [48].

2D technológia Lidar senzoru, využíva vo vnútri rotujúci senzor v 360° stupňovom uhle. Odrazové sklíčko nachádzajúce sa vnútri Lidaru, zaisťuje neustále meranie, závislé na rotačnej frekvencií väčšinou v rozmedzí 1-100 Hz. Objekty sú definovateľné, iba pokiaľ sa nachádzajú v rovnakej rovinnej výške ako Lidar zariadenie. Vďaka takému to princípu, dokáže Lidar zariadenie vytvárať dáta o obraze horizontálnej plochy a následnú interpretáciu o súradniciach x,y, [47], [48].
Lidar vytvárajúci 3D priestorové vyobrazenie, funguje na viacerých typoch princípoch. Prvým z nich je využitie natáčacieho zariadenia s laserom z 2D technológie, pridaním vertikálneho pohybu v určitom rozsahu. Takto vytvorený Lidar dokáže snímať priestor a vytvárať z neho 3D priestorový obraz. Nevýhodou takéhoto vyhotovenia je najmä pomalšia rýchlosť snímania. Druhý princíp je založený na dvojici microelektrických zrkadlách (MEMS). Ich rotáciou dokáže Lidar zaznamenávať priestor okolia okolo seba. Nevýhodou sú vzájomné rušenia vibráciami lúčov, či možnosť nepresného „nastrelenia" laserového lúča do zrkadla, [25]. Najdrahší typ z trojice taktiež vychádza zo stavby 2D Lidaru. Na priestorové meranie využíva väčšie množstvo lúčom rozprestrených do vertikálnej roviny. 16-64 laserových lúčov je rozprestretých s konštantným uhlom delta a vďaka 360° rotácií okolo svojej osi vie zaznamenávať body objektov a vytvárať z nich priestorový obraz, [11], [48], [47].
Lidar má rozsiahle pole uplatnenia pozdĺž širokým spektrom vedných odborov. Technológia prvotne určená pre mapové monitorovanie povrchu, sa rozšírila aj do odborov ako robotika, automobilový priemysel či archeológia. Lidar emituje laserové lúče s vysokým rozpätím(vlnová dĺžka $600 - 1000nm$). Od voľby vlnovej dĺžky, závisí aj správna voľba vzdialeností, ktoré sa Lidarom dajú merať. Vo všeobecnosti teda sú pre rôzne aplikácie vyhovujúce rôzne vlnové dĺžky : [47], [22].

- Infračervené - meteorológia
- Blízko infračerveného - mapovanie povrchov
- Modré - batymetria(štúdium hĺbkových oceánov)
- Ultrafialové - Automobilový priemysel

V rámci diaľkového merania, rozlišujeme dva typy technológií podľa zdroju energie využitej na detekciu objektu. Pasívne systémy využívajú emitovanú energiu generovanú externým zdrojom, napríklad slnkom. Aktívne zariadenia, využívajú k detekcií vlastné zdroje energie. K aktívnym systémom zaraďujeme teda aj Lidar. Tento fakt prináša nesporné výhody v porovnaní s pasívnymi systémami. Táto vlastnosť umožňuje napríklad Lidarovým systémom pracovať v nočných podmienkach bez akýchkoľvek kompromisov. Dokonca sa to často javí ako výhodnejšia alternatíva. Taktiež priveľká presaturovanosť okolia nie je pre Lidar zariadenia obzvlášť problematická, [5], [11], [47].
Kamery, ktoré sa často používajú pre detekciu okolia, sa v súčasnosti taktiež vyznačujú veľmi vysokým rozlíšením, a dobrou rozoznávacou schopnosťou. Takéto zariadenia považujeme za pasívne. Problémy s oslňujúcim slnkom alebo slabou viditeľnosťou, teda priamo vyplývajú z daného zaradenia. Kamery sa primárne používajú pre obrazovú analýzu. Obrazové vstupy majú slabú schopnosť vytvárania priestorového obrazu v požadovanom rozlíšení. Pre dosiahnutie prijateľnej informácie sú potrebné komplexnejšie algoritmy, často spojené s vyššou výpočtovou náročnosťou, [11].
Vďaka kamerovým systémom existuje viac možností ako vytvárať point cloud. Jedna

možnosť je pomocou stereo vízie, viď obr. 1.3. Použitím dvojice kamier, sa získava dvojica fotiek so známou vzdialenosťou medzi sebou. Pomocou triangulácie je možné následne spočítať vzdialenosť k objektu, [18].



Obr. 1.3: Porovnanie stereovízie a technológie Mobileye, prevzaté z [39]

Druhou možnosťou je použitie fotografického riešenia, napríklad "Mobileye". Analýzou 2D obrázkov vytvorených v pohybe dokáže táto metóda vytvárať 3D informácie o objektoch, viď obr. 1.3. Metóda je ale plne závislá od pohyblivého zariadenia, preto pri zastavení nedokáže byť táto informácia získavaná, [11].

1.1.2 Popis konkrétneho zariadenia

Používané Lidar zariadenie, nachádzajúce sa na navigovanom robotovi je vyrábané firmou SLAMTEC. Naše konkrétne zariadenie RPLidar A1, viď obr. 1.4, umožňuje snímať okolie v 360° uhle a tak produkuje dáta 2D point cloudu. Princíp takéhoto zariadenia je opísaný v predošlej sekcií. Používané Lidar zariadenie dosahuje rýchlosť merania vzdialeností až 8000 krát za sekundu. Opisovaný Lidar dokáže automaticky zistiť, kedy môže meranie prebehnúť rýchlejšie, a kedy sa má spomaliť pre získanie dostatočne hustého poľa bodov. Frekvencia skenovania je nastaviteľná v rozmedzí 2 až 10 Hz. Zariadenia dokáže snímať objekty až do vzdialenosti 12 metrov pri uhlovom rozlíšení 1°. Cena používaného Lidaru sa pohybuje okolo 110 eur, [21], [20].

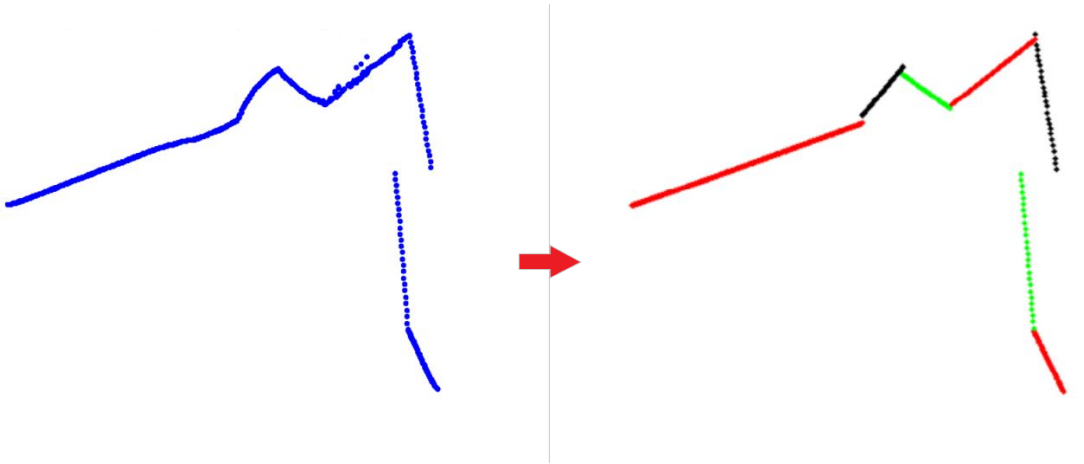
Obr. 1.4: Lidar RP A1, prevzaté z [20]

1.2 Segmentácia

Segmentácia je prvý krok pri analýze nameraných bodov. Po nameraní bodov point cloudu, je hlavná úloha segmentačného procesu rozdelenie bodov s podobnými charakteristikami do homogénnych častí, [18].
Vytvorené jednotlivé skupiny musia byť rozdelené tak, že budú mať určitý zmysel, viď obr. 1.5. Segmentácia je teda proces rozdeľovania vstupných bodov do disjunktných množín. Tieto oblasti sú zvonku úplne ohraničené. Vnútri ohraničenej skupiny zdieľajú dané body podobné vlastnosti, pričom žiadne dve množiny nemajú spoločné vlastnosti, [28], [6], [40]. Nech F je množina všetkých bodov vstupu, P je homogenita, tak segmentácia je taký proces rozdelenia F do podmnožín $S_1 - S_n$, že pre ne platí : [28]

$$\bigcup_{i=1}^{n} S_i = F, \qquad S_i \bigcap S_j = \oslash, \qquad i \neq j \tag{1.2}$$

Segmentácia point cloudu je obor, ktorý profituje najmä z rozvinutej oblasti obrazovej analýzy. V počítačovom videní je segmentácia 2D obrázkov bežný problém, na ktorý bolo vytvorených nespočetne veľa štúdií za desiatky rokov. Nápady a hlavné myšlienky týchto štúdií, boli často využité aj pri point cloud segmentácií, [6], [40].



Obr. 1.5: Segmentacia, prevzaté z [19]

Pri rozdeľovaní 3D dát sú často implementované metódy delenia založené na grafových algoritmoch. Zvyčajne sa používajú z vytvoreného grafu metódy delenia ako "normalized cutä "min cut". Zatiaľ pretrváva primárny záujem pre rovinnú 2D segmentáciu bodov point clodu, pre jej uplatnenie pri množstve úloh. Segmentácia rovinných bodov má štruktúru podobnú obrázkom, a preto je v nej vysoko praktické aplikovať postupy z oboru počítačového videnia, [26].
Rozdelené pole bodov, má veľmi široké rozpätie aplikácií. V literatúre a najmä článkoch sa nachádzajú 100-ky segmentačných techník point cloudu. Vo všeobecnosti sa aj napriek širokému výskumu v tejto oblasti, nenachádza jediná metóda, vyhovujúca všetkým typom point cloudu. Algoritmy vytvorené pre jednu skupinu obrázkov, nie sú často aplikovateľné pre ostatné. Pre výber určitej metódy nad inou, je často potrebné prizeranie na konkrétne pole bodov, jej aplikáciu a charakteristiku problému. Selekcia správnej techniky je teda komplikovaný problém, pričom do týchto chvíľ neexistuje univerzálne akceptovaná metóda

pre segmentáciu point cloudu, čo vytvára výzvy na jej ďalšie hľadanie v oblasti vedeckého výskumu, [6].
Práca Anguelova [1], predkladá predikcie pre point cloud segmentačné algoritmy, ktoré by mali mať podľa nej nevyhnutné vlastnosti určité vlastnosti.
Výhoda hĺbkovej informácie pri point cloude, by mala byt oproti plne využitá.
Algoritmus by mal byť schopný správne označiť body, ktoré ležia v riedko zahustených plochách podľa informácií o ich susedoch.
Segmentačné algoritmy by mali byť priamo adaptované na daný lidar skener, na ktorého dátach pracuje. Rôzne lidar zariadenia produkujú kvantitatívne odlišné point cloud dáta, a teda môžu mať rôzne vlastnosti.
Segmentácia môže byť často nápomocná na analýzu scény z rôznych smerov pohľadu, napríklad lokalizácia, rozoznávanie objektov, klasifikácia alebo vyňatie charakteristických čŕt. Pomocou point cloudu vieme zistiť tvar, veľkosť či vlastnosti meraných objektov, [26]. Segmentácia point cloudu avšak nie je úplne triviálna úloha. Vstupné dáta bývajú často zašumené, riedke a neorganizované. Hustota rozloženia dát býva taktiež nerovnomerná, kvôli meniacim sa podmienkam lineárnych posuvov a uhlovým rýchlostiam rotácie skeneru. Množstvo segmentačných techník často vyžaduje "ručné"doladenie. Pre potreby ľudského zásahu sa ale v aplikovanej praxi vytvára časová strata. Preto existuje domnienka vo vedeckých kruhoch, že zvyšujúca rýchlosť je aj napriek malým chybám užitočná, [43].

1.3 Analýza súčasného stavu

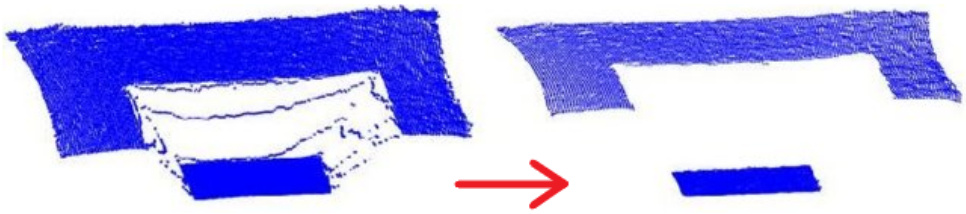
Segmentácia sa vykonáva na určitej množine dát, pričom sú použité rôzne postupy založené na rôznych kritériách hľadania podobných vlastností. V závislosti na nich sa vytvárajú následne rozhodnutia, a tvoria podmnožiny podobných vlastností. Rôzne postupy spôsobujú zhlukovanie množiny bodov do rôznych podskupín. Podobnosť je často meraná prekročením nejakej stanovenej hranice či meraním priestorového či rovinného prepojenia bodov. Typy algoritmov zameriavajúcich sa na segmentáciu môžeme rozdeliť do 4 skupín podľa metodiky ich riešenia, [26].

1. Metóda Edge Based metóda - metóda hrán
2. Metóda Surface growing metóda - postupného rastu
3. Metóda Model based metóda - geometrickej podobnosti
4. Metóda Graph theory - metóda grafovej segmentácie
5. Metóda Machine learning - metóda strojového učenia

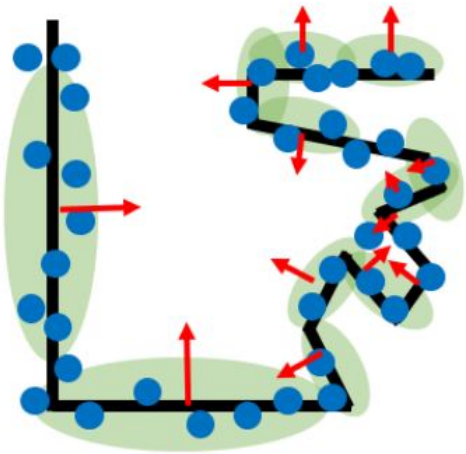
1.3.1 Metóda hrán

Metóda založená na vyhľadávaní hrán v mraku oblakov je metóda prevzatá a veľmi používaná pri obrazovej analýze. Napriek rokom od prvého uvedenia, sa táto metóda považuje v súčasnosti za najviac podrobovanú výskumu. Hrany, ktoré obkolesujú telesá obsahujú vlastnosti, ktoré sú danou metódou skúmané, [26].
Existujú dva typy hrán: skokové ("jump edge")(obr. 1.6) a záhybové hrany ("crease edge"),(obr. 1.7). Skokové hrany sú definované veľkou zmenou hĺbky alebo výšky. Záhybové hrany sa nachádzajú na rozhraní dvoch rozdielnych povrchov. Takéto hrany majú vlastnosti zmeny

normálových vektorov. Normály prislúchajúce bodom na hranách majú uhlové zmeny väčšie ako je prahová hodnota, [26].
Postupy algoritmov založených na detekcií hrán, sa dajú rozdeliť do dvoch častí. Prvotne nastáva detekcia takýchto hrán, a následne zoskupenie daných bodov nachádzajúcich sa vnútri množiny ohraničenej danými hranami. Tieto zmeny vlastností sú väčšinou detekované rôznymi nástrojmi ako gradientami, normálami, zmenou hladkosti kriviek či vyššími deriváciami, [15].



Obr. 1.6: Skokové hrany, prevzaté z [32]



Obr. 1.7: Záhybové hrany, prevzaté z [7]

Keďže tieto nástroje hľadajú náhle zmeny v množine bodov, sú náchylné na zašumené dáta. Tie vykazujú vlastnosti ako skokových tak aj záhybových hrán. Preto často robia problém pri ich správnej lokalizácií, [35].
Pri takýchto metódach často nastáva aj znehodnotenie pôvodnej informácie, nakoľko tú dostávame iba z oblastí jasných zmien - na hraniciach množín. Nastávajú aj problémy priveľkej či primalej segmentácie v dôsledku často nedokonalých, nepresných či neuzatvorených hrán objektov, [33].

Analýza vybraných článkov

Základným kameňom v tejto oblasti bola práca Bhanu [2] a spol. z roku 1985. Táto práca načrtla prvé možnosti riešenia segmentácie 3D bodov, za použitia hľadania vlastností o hranách. V tejto práci boli rovno predstavené hneď tri rôzne prístupy hľadania hrán. Práca sa zameriavala na rozsahové obrázky("range image"). V práci boli všetky prístupy

postavené na matematických nástrojoch vyskúšané na dodaný obrázkoch. Gradientné metód, napasovávanie priamok, či detekcia zmien v normálach boli použité v lokálnej mierke pre všetkých susedov daného bodu. Gradientný prístup bol založený na použití druhej derivácie v diskrétnom tvare. Cieľ pri takomto prístupe bolo hľadanie lokálneho maxima určeného nejakou hranicou. Technike napasovávania priamok predchádzalo vytvorenie binárneho stromu ktorý bol rozdelený podľa mediánu hodnôt jednej zo súradníc. Takto rozčlenený strom, je podľa smerom vektorov, získaných z dvoch listov smerujúcich do spoločného uzlu rozdelený, čím nastáva segmentácia. Pri tretej metóde sa využíva informácia o normálových vektoroch. Pri vypočítaní normálových vektorov, sa hľadá signifikantná zmena. Jej nájdenie identifikuje prítomnosť hrany. Dáta boli následne použité a vyskúšané na počítačovo vytvorených dátach a dátach získaných z laserového skenera. Získané výsledky práce boli indikované ako použiteľné pre ďalšie spracovanie ako napr. rozpoznávanie.

V práci od Angel D. Sappa [35] a spol., bol vytvorený algoritmus založený na princípe hľadania hrán. Algoritmus je rozdelený do časti vytvárania bitovej mapy ("bitmap"), a následná detekcia obrysov.
Vytváranie bitmapy nastáva skenovaním a pomocou ortogonálnych úsečiek, čoho výsledkom sú dve bitové mapy. Úsečky sú definované bodmi ktoré sa nachádzajú v daných na seba kolmých smeroch. Algoritmus je založený na myšlienke že všetky hrany vedia byť zachytené dvomi ortogonálnymi vektormi. Stĺpce a riadky daného obrázka identifikujú skokové hrany. Tie sú následne zmenené za krivky, ktorých analýza dokáže identifikovať záhybové. Filtráciou sa odstraňujú zašumené dáta. Dané krivky sú následne napasované pozdĺž bodov na ich aproximáciu. Pri prekročení určitej definovanej dĺžky alebo chyby k bodu vopred určenej hranice, je daná krivka predelená.
Pomocou koncových bodov kriviek a triangulácie týchto bodov je vytvorený graf minimálnej dĺžky stromu. Vrcholy daného grafu predstavujú segmenty aproximovaných kriviek. Aplikáciou algoritmu sa odstraňujú z grafu krátke vetvy. Následne je graf spätne pospájaný podľa vzdialenosti koncov jednotlivých segmentov. Po oddelení nespojených častí vzniká nájdená hrana.
Daná technika používa informáciu len o hranách, a preto sa vyznačuje vysokou výpočtovou rýchlosťou. Algoritmus bol primárne použitý pre rozsahové obrázky("range image"), na ktorých bol odskúšaný a porovnávaný.

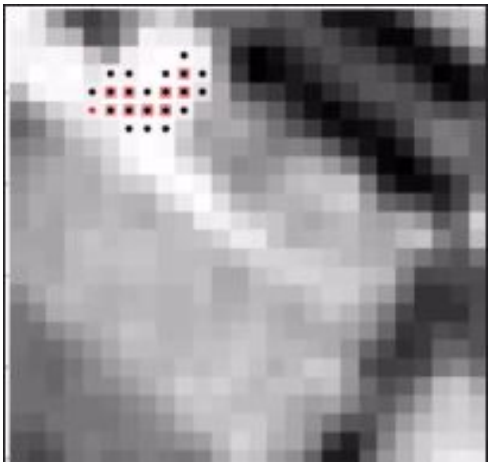
Metódy odvodené od analýzy hrán, zvyknú mať vysokú senzitivitu na zašumené dáta a nepravidelné rozloženie bodov čo sú často vyskytujúce sa situácie v point cloude. Na druhú stranu, sa vyznačujú vysokou výpočtovou rýchlosťou, [26].

1.3.2 Metóda postupného rastu

Metóda postupného rastu vytvára algoritmy založené na homogenite jednotlivých častí. Je to taktiež postup vychádzajúci z obrazovej analýzy známy pod názvom rast oblastí("region growing"). Metódu rozrastania môžeme rozdeliť podľa spôsobu fungovania do dvoch častí:

- Bottom-up prístup
- Top-down prístup

Metóda rastu "bottom-up"(obr. 1.8) začína buď náhodným alebo deterministickým vybraním malého množstva bodov, ktoré sú použité ako počiatočné body rozrastania.Vlastnosti ako zaoblenosť hrán, normálové vektory, sklon susedných hrán či vzájomné vzdialenosti susedných bodov sú často posudzované pri hodnotení príslušnosti nového bodu k stávajúcej množine. Podobnosť vlastností je porovnávaná medzi susednými bodmi a pri zhode zväčšuje svoju oblasť pôsobnosti až po hranice v ktorých sa už takéto spoločné vlastnosti nevyskytujú, [26], [15], [33].



Obr. 1.8: Metóda postupného rozrastania, prevzaté z [16]

Metóda "top-down"je metóda opačná. Začiatok takéhoto algoritmu vyberá daný point cloud ako jeden zhluk bodov. Nastáva delenie takejto skupiny bodov až do dosiahnutia dostatočne malej chyby určitej vlastnosti vrámci danej množiny. Postupným delením teda zaisťuje homogénne vlastnosti jednotlivých podmnožín, [26], [15].

Analýza vybraných článkov

Článok od autorov K. Koster [17] a M. Spann ukazuje algoritmus založený na metóde rastu, založenej na štatistickom teste. Spájajúcim kritériom nových bodov, je vlastnosť ktorú nazvali MIR(pomer vzájomného vkladania). Pomenovaná vlastnosť sa získava robustnou regressiou pomocou novej metódy výpočtu miery gausovej distribúcie. Algoritmus prvotne rozdelí rozsahové obrázky("range image") na väčšie entity, pozostávajúce z pixelov mierky 5×5. Klasické mozajkovanie ("teselácia") by produkovalo chybné hrany na okrajoch týchto blokov, preto medzi nimi nastáva posunutie jeden do druhého o max 2 pixely. Vrámci bloku sú vyrátané vnútorné parametre LMedS (metóda najmenších mediánov štvorcov) metódy pomocou množiny každých 3 bodov. Prvotné rozrastanie nastáva iteratívne od bloku s najmenšou hodnotou rozptylu, vypočítanou pomocou LMedS. Blok je prvotne rozrastaný o tých svojich susedov, ktorí spĺňajú podmienku rastu (MIR) pomeru. Na základe hodnoty MIR sú hodnotené "sily väzieb" medzi jednotlivými časťami. Jednotlivé oblasti sú teda postupne spájané na základe veľkosti MIR väzby pričom musí byť spojená minimálna požiadavka. Vytvorenie ohraničujúcich hrán nastáva na pixel úrovni z 5×5 mriežky. Hrany sú počítané zo šírky hrany medzi oblasťami. Pri prekročení určitej hodnoty, nastáva identifikácia hrany. Nakoniec bol použitý 3×3 medián filter na dodatočné odstránenie odľahlých bodov. Daný algoritmus bol porovnávaný na 60 skutočných

rozsahové obrázky("range image") s existujúcimi metódami, pričom je použiteľný na široké množstvo vstupných dátových typov.

Práca T. Rabbani a spol. [31] sa sústredila práve na analýzu 3D point cloudu. Metóda ma základ v zjemňovaní napätia("smoothnes constraint") myšlienke, ktorá hľadá a spája jemne spojené oblasti v point cloude. Segmentácia má dve časti. Najskôr prebieha delenie bodov na základe normál, a potom sa jednotlivé oblasti spájajú. Normály sa vytvárajú na základe aproximovanej úsečky medzi 2 bodmi. Pre výpočet algoritmus triedenia susedných bodov načrtli v práci dva postupy, KNN(k najbližších susedov) a FDN(konštantnej vzdialenosti k susedom). KNN vyberá body konštantne daného počtu K bodov, oproti FDN kde sa vyberajú body do určitej vzdialenosti. Využíva sa klasická euklidovská metrika pre oba postupy. V závislosti na pravidelnosti hustoty bodov, je jedna či druhá metóda lepšia. Prispôsobovanie plôch("plane fitting") normál rozdelených častí je aplikované pomocou metódy najmenších štvorcov. Body vrámci jednej oblasti, majú vytvoriť normály ktoré sa moc nelíšia. Pri prekročení určitej hranice nastáva ďalšie delenie. Následné spájanie jednotlivých rozdelených oblastí, je produkované pomocou porovnávania normál týchto podmnožín aj so zvyškami. Iteratívne sa volia oblasti s najmenším množstvom zvyškových bodov. Algoritmus vyžaduje malé množstvo nastavovacích parametrov, čím sa zjednodušuje jeho použitie na širšiu škálu modelov.

Práca T. Pavlidisa a L. Horowitza [29] priniesla pravdepodobne najväčšiu popularitu algoritmu v súčasnosti známeho ako Split and Merge("rozdeľ a spoj"). Predstavený algoritmus prináša, vďaka binárnemu deleniu poľa bodov, rýchlu prácu a segmentáciu poľa. Zložením oboch častí „Bottom-upä „Top-down"metódy postupného rastu, je vytvorený v práci komplexný algoritmus na segmentáciu bodov. Čisto geometrický princíp rozrastania je založený na postupnom delení priamok vytvorených spojením dvoch náhodných bodov v miestach najväčšej odchýlky bodu poľa k vytvorenej priamke. V práci boli predstavené dva prístupy výpočtu chyby. Následne spájanie množín je uskutočnené vtedy, keď dve spojené množiny majú menšiu chybu obohatenú o určitú konštantu z, ako množiny rozdelené. Takéto spájanie sa buď môže diať na dvoch susedných segmentov, alebo pre všetky segmenty vzájomne, do bodu kedy už nebude možné spojiť ďalšie segmenty. Algoritmus bol v práci otestovaný s rôznymi kombináciami nastavení na lineárnu aproximáciu polygónu.

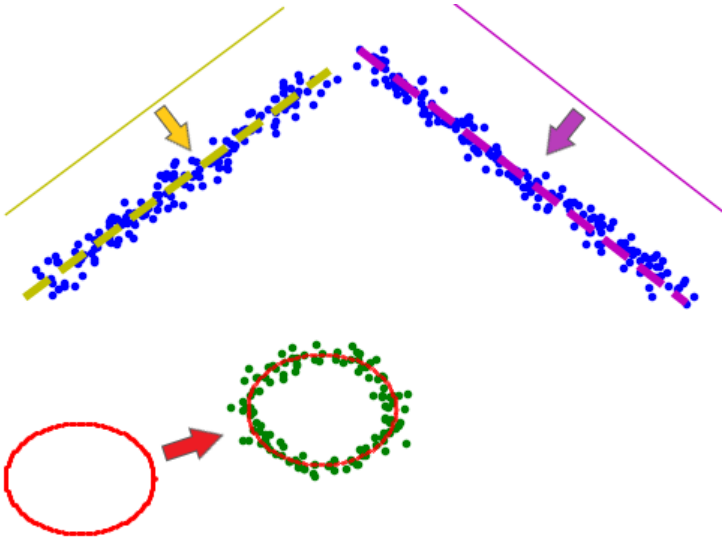
V práci z roku 1966 J. Vandorpe a spol. [41], bol predstavená matematický algoritmus na dynamické vyhľadávanie jednoduchých matematických útvarov pre mobilného robota. Segmentácia podľa priamok je vytvorená základe kritérií testovania blízkych bodov. Merané dáta bodov do sú prevedené karteziánskej súradnicovej sústavy. Extrakcia priamkových segmentov nastáva postupným pridávaním bodov k aktuálne vytvorenej priamka definovanej skupinou bodov. Pridanie bodov nastáva po splnení hlavných troch kritérií.
Vzdialenosť pridávaného bodu od posledne pridaného musí byť dostatočne malá. Vzdialenosť bodu k vytvorenej priamke ako uhlová zmena vykresľovanej priamky musia byť menšie ako stanovená hranica. Na konci sú takto vytvorené priamky spätne pomocou lineárnej regresie prevedené do polárnych súradníc reprezentujúcich (ρ, θ). Extrakcia kružníc je vykonávaná v prípade, že aspoň dva body v postupnosti nesplnia kritérium vytvárania priamky. Následovné body sú kruhovo extrahované, pri splnení podmienky, že daný pridávaný bod sa nachádza vo vzdialenosti k stredu kruhu menšej ako je jeho polomer. V

následnom testovaní metódy na nameraných dátach dosahuje algoritmus správne hodnoty segmentácie pre väčšinu testovaných bodov.

Algoritmy založené na tejto myšlienke sú náchylné na rozmiestnenie a počet počiatočných bodov rozrastania. Na druhej strane je táto metóda ale celkom odolná na zašumené dáta. Často nastáva priveľké alebo primalé segmentovanie a metóda je z výpočtového hľadiska časovo náročná. Metóda kvôli lepšej rezistencií na zašumené dáta, zvykne mať lepšie výsledky ako metóda hrán, [33].

1.3.3 Metóda geometrickej podobnosti

Takýto prístup je založený na myšlienke, že veľká časť analyzovaných objektov je vytvorená človekom, a teda jej základné črty sa dajú aproximovať jednoduchými geometrickými útvarmi ako je kruh, rovná plocha či valec, viď obr. 1.8. Z matematických modelov sú vytvorené útvary, pomocou ktorých sa vytvára príslušnosť jednotlivých bodov navzájom k sebe, pričom nastáva ich zoskupovanie.



Obr. 1.9: Metóda modelovania

Analýza vybraných článkov

Opačný princíp fungovania ako je zjemňovacia technika, bol popísaný v práci Fischler [9] a spol. v roku 1981. Technika pracuje pri inicializácií čo s najmenším množstvom dát, a rozširuje ho pokiaľ vybrané dáta spĺňajú kritéria. Ransac ("Random sample consensus") algoritmus začína prvotne s náhodným vybraním bodov. Pri prekladaní úsečkou, vytvára začiatočnú predikciu len z dvoch bodov. Pri dosiahnutí minimálneho počtu bodov, potrebných na vytvorenie úsečky, zjemňuje Ransac svoj odhad napríklad pomocou techniky najmenších štvorcov. Takýto Ransac algoritmus je založený na 3 voľbách parametrov, od ktorých sú ovplyvňované následné výpočty. Tolerancia chyby, určujúca maximálnu vzdialenosť bodu od modelu, počet iteratívnych cyklov a minimálny počet bodov prislúchajúcich k danej úsečku. Vzdialenosť chyby od bodu je často nastavovaná cez priemerné vzdialenosti alebo experimentom. Počet maximálnych iterácií založený na pravdepodobnosti, že každý vybraný bod je v dostatočnej vzdialenosti od modelu. Ransac algoritmus

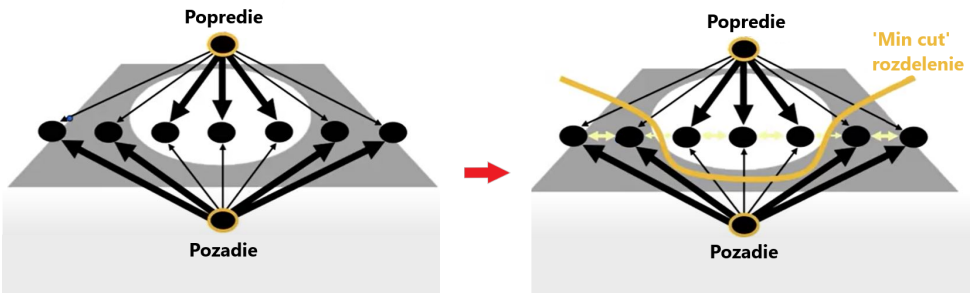
má oproti klasickejším metódam ako LNS(metóda najmenších štvorcov), výhodu efektívneho vyraďovania očividne nepatriacich bodov do danej množiny. Taktiež je výpočtovo málo náročný.

Algoritmus vytvorený Paulom Houghom [13] bol vytvorený v roku 1960 a následné aj podaný ako americký patent o dva roky neskôr. Algoritmus je vytvorený ako postup, ktorý dokáže nachádzať rôzne komplexné schémy v dodanom obrázku. Pointou algoritmu je prevedenie problému nachádzania útvarov (prevažne priamok) do Hough priestoru. V ňom sa v takzvanej akumulačnej matici vytvárajú obrazce tak, že pre všetky body sú vytvorené možné priamky nimi prechádzajúcu. Body v obrazovom priestore, sa v Hough priestore zobrazia ako úsečky alebo sínusoidy, podľa postupu parametrizácie. Hľadaním najväčšieho množstva prienikov takto vytvorených kriviek v Hough priestore, sa prevedie úloha hľadania priamok priamok v obrázku na úlohu hľadania lokálnych maxím v akumulačnej matici. Varianty postavené na základe tejto techniky sa s vysokou obľubou používajú v obrazovej analýze dodnes.

Selektujúce algoritmy založené na tejto myšlienke aproximácie tvarov majú čisto matematický základ. Vyznačujú sa výpočtovou rýchlosťou a najmä schopnosťou sa vysporiadať so zašumenými dátami. Nevýhodou je ich slabá aproximačná schopnosť nachádzať nepravidelne tvarované útvary, [26], [33].

1.3.4 Metóda grafovej segmentácie

Segmentácia bodov založená na štruktúrach z teórie grafov vytvára z bodu mrakov graf uzlov pospájaný hranami tak, že to vykresľuje vzájomnú naviazanosť jednotlivých bodov, viď obr. 1.10. Hlavná myšlienka rozdeľovania je založená na tom, že body v totožnej štruktúre sú navzájom ďaleko viac prepojené ako tie ktoré do nej nepatria. Hranice teda vznikajú na najslabších miestach hrán a podľa toho nastáva aj delenie. Segmentácia point cloudu založená na grafových algoritmoch bola prvýkrát predstavená v spojení s normovaným delením("normalized cut"), prebratým z obrazovej analýzy. Algoritmy založené na takejto štruktúre vykazujú efektivitu a často sa využívajú v robotike. Metóda meria ako vzájomné rozdielnosti medzi skupinami tak aj spoločné vlastnosti vrámci množiny. V porovnaní s ostatnými metódami, nevykazujú tieto algoritmy vysokú senzitivitu na zašumené dáta ale sú náchylnejšie na presegmentovanie, [26], [33], [44].



Obr. 1.10: Model najkratšieho stromu, prevzaté a upravené z [34]

Analýza vybraných článkov

Článok pod vedením F. Felzenszwalb [8] a spol., rieši problém rozdeľovania a nachádzania hraníc oblastí pomocou grafovej štruktúry. Algoritmus, rozhodujúci na základe vytvoreného grafu, bol použitý na obyčajné obrázky. Využitie daného postupu je ale jednoducho nastaviteľné na rozsahové obrázky("range image") či prípadne point cloud. Prvotne je z obrazového vstupu vytvorená grafová štruktúra. Každý uzol grafu predstavuje bod (pixel) obrázka, a každá hrana odráža podobnosť zvolených riadiacich vlastností medzi dvomi bodmi, napr. intenzita, farba, poloha atď. Uzli sú v prvom rade zoradené do množiny podľa stúpajúcej hodnoty váhy. Segmentácia v takomto grafe predstavuje nájdenie uzlov E' ktoré sú podmnožinou uzlov celého grafu E. Segmentácia a spájanie je založené na porovnávaní váh medzi jednotlivými uzlami. Spojenie dvoch bodov vyžaduje podobnosť váh menšiu ako zadefinovaná hodnota, a ich rozdelenie naopak hodnotu väčšiu. Obidva postupy sú usmerňované určitým prahovou hranicou, ktorého správnym nastavením sa zabezpečuje že množstvo segmentačných množín, nebude ani príliš veľké, ani malé. Algoritmus ma menšie problémy so spájaním oblastí, ak sa medzi nimi vyskytuje aspoň jedna malá hodnota váhy. Algoritmus bol v práci predstavený v dvoch podobách, a to pri obrazových vstupných dátach, a priestorových.

Autori v článku J. Shi a spol. v článku [36], predstavujú ďalší príklad algoritmu vytvoreného na grafovej štruktúre. V články predstavujú metódu používajúcu normované delenie("normalized cut") ako primárnu rozhodovaciu entitu. Takto vytvorený graf je viac zameraný na konzistenciu obrázka, ako na lokálny prístup často používaný v ostatných metódach. Grafová štruktúra je ako v predošlom prípade vytvorená zo vstupných dát tak, že body (pixeli) predstavujú uzli a hrany grafu reprezentujú vlastnosti medzi susednými bodmi. Normované delenie("Normalized cut") je myšlienka odvodená od známejšieho algoritmu "minimum cut". Oproti používanej technike ("minimum cut") kde rez grafu nastáva v závislosti na hodnote váh medzi uzlami, normované delenie("Normalized cut") pracuje s podielom súčtu všetkých hrán grafu. Zhodne sú definované aj kritéria na delenie alebo spájanie bodov grafu do určitých oblastí pomocou normovaného delenia("Normalized cut"). Riešenie daného NP problému (trieda komplexnosti výpočtu), predstavuje použitie vlastných vektorov matice hodnôt váh a matice sume hodnôt váh jednotlivý hrán. Algoritmus v prácu je vytvorený rekurzívnou formou, kedy pri každom vytvorení novej oblasti, sa daná oblasť kontroluje. Pri splnení podmienok nastáva ďalšie volanie, inak sa pokračuje na nasledujúcu časť grafu. Nastavovanie hodnôt váh grafu, môže mať vo forme obrázku odrážať pozíciu pixelov a ich intenzitu. Daný algoritmus bol v práci odskúšaný na počítačovo vytvorených dátach, ako aj na statických a pohyblivých reálnych obrázkoch.

Grafové algoritmy zvyčajne vykazujú efektivitu a často sa využívajú v robotike. V porovnaní s ostatnými metódami, nevykazujú tieto algoritmy vysokú senzitivitu na zašumené dáta. Niektoré z nich vyžadujú „offline" tréning a rýchlosť výpočtu často nedosahujú real time použitie, [26], [33], [44].

1.3.5 Metóda strojového učenia

Všeobecne sú algoritmy strojového učenia rozdelené podľa prístupu k učiteľovi na podporované a nepodporované. Základom podporovaného učenia sú označené, popísané dáta

ktoré sa využívajú na tréning. Proces oštítkovania tréningových dát je zvyčajne konaný človekom. Výsledky sú priamo závislé od kvality tréningových dát ako hustota atď. Druhou možnosťou je použitie tréningových dát generovaných počítačom, ktoré sú už priamo pri vytváraní aj označené. Takéto riešenie avšak nie vždy kopíruje presné vlastnosti vyskytujúce sa pri reálnych experimentálnych dátach. Vďaka rôznym chybám, často vyplývajúcich z ľudskej nepozornosti pri označovaní dát, nastáva zhoršenie interpretácie výsledkov pri takýchto metódach. Úloha označovania dát je náročná najmä kvôli veľkej diverzite meraných objektov, ako aj z časového hľadiska. Pre každý typ algoritmu strojového učenia, patria rôzne rozlišovacie zákonitosti. Skupiny dát sa vytvárajú pred a v priebehu označovacieho procesu. Z toho vyplýva, že po natrénovaní, ostávajú už výsledné skupiny nemenné. Vzniká pri každej novej úlohe nutnosť vytvárať a zároveň trénovať nový model systému odznova. Totožne je to aj s novou trénovanou množinou. Nepodporované učenie je trieda problémov, ktoré hľadajú zákonitosti o organizácií dát. Podporované učenie sa učí ako sú dodané tréningové dáta na seba závislé, a podľa vykonávajú zadanie, [12]. Učenie tréningového modelu vo všeobecnosti trvá veľké množstvo času. Zistilo sa, že kvalita dát z ktorých sa vytvára tréningová množina ďaleko prevažuje nad ich kvantitou, [26], [15].

Segmentácia riešená ako problém klasifikácie, bola pointa pri práci W. Bichen [46] a jeho týmu. Využitím konvolučnej neurónovej siete, presnejšie SqueezeSeg, ktorej vstupom sú transformované dáta z point cloudu a výstupom označené dáta. Pri preprocessingu externých dát, boli dáta transformované na trojrozmerný tenzor. Point cloud dáta boli zdiskretizované tak, že boli premietnuté do hustej mriežky tvaru gule. Pri 64 vertikálnych lúčoch a 90° záberu Lidar zariadenia nastalo zdiskretizovanie aj šírky uhlového záberu na 512 bodov. Následkom toho mohli byť vstupom do siete dáta v tvare ($šírka \times výška \times vlastnosť$). Ako vlastnosť boli použité súradnice x, y, z, miera intenzity a vzdialenosť od Lidar zariadenia r. Štruktúra siete vychádza zo siete využívanej na detekciu v obrazovej analýze. Kvôli nepomeru veľkostí medzi šírkou a výškou vstupujúcej do siete, boli dáta prevedené cez konvolučné jadrá na jej zníženie pred vstupom do siete a po výstupe spätne dekonvolonované. Výstupné dáta boli ošetrené od chybných označení totožných dát do rôznych skupín pomocou CRF (podmienené náhodné pole). Minimalizácia energetickej funkcie CRF, vedie k zlepšeniu štítkovania bodov. Výstupom modelu je pravdepodobnostná mapa, pri ktorej je nastáva zlepšenie ďalšími iteráciami. Trénovanie bolo tvorené na množine „KITTI"(voľne stiahnuteľná množina point cloud bodov, použitá v práci), ktorej označenia trénovacích dát, boli stiahnuté z 3D modelov. Práca taktiež vytvorili ďalšie trénovacie dáta pomocou známej hry GTA V. Algoritmus vo výsledkoch vykazoval vysoké rýchlosti pri dostatočných vyhovujúcich presnostiach. Algoritmus bol experimentálne overený na skutočných a vytvorených point cloud dátach, pričom boli hodnotené kategórie presnosti, rozpoznania a ostrosti hrán na rozmedzí tried.

Článok L. Xiaohu [23] a spol., prezentuje rozdeľovací algoritmus ktorý dokáže byť s menšími zmenami aplikovateľný na 2D alebo 3D vstupné dáta. Rozdeľovací algoritmus prvotne zanalyzuje požadované kritéria, následkom ktorých nastane delenie bodov do skupiny tried. Myšlienka princípu spájania bodov, vychádza z algoritmov "K-means", prípadne "Mean shift". Segmentácia založená čisto na týchto algoritmov často zvykne byť nedostatočná. Princíp prezentovaného algoritmu prepojenia párov("pairwise Linkage") je založený na hľadaní bodov s najväčšou hustotou susedných bodov. Lokálne maximum sa predáva od náhodného bodu. Za predpokladu, že niektorý zo susedných bodov v určí-

tom rozmedzí vzdialeností (definovanej ako medián najmenších vzdialeností), dosahuje vyššiu hustotu bodov, predáva sa pozícia lokálneho maxima danému bodu. Takto vytvorená metóda by bola veľmi náchylná na presegmentovanie poľa dát. Vlastný algoritmus na spájanie dvoch príslušných množín, funguje na základe porovnávania hustoty bodov, priemernej hustoty, a odchýlky príslušiacej pre každú triedu. Pri splnení dvoch podmienok kritérií, nastáva spojenie. Pri 3D dátach point clodu, ostáva myšlienka a práca algoritmu nezmenená. Mení sa iba spôsob delenia kritérií v 3 bodoch.
Namiesto susedných bodov v dopredu definovanej vzdialenosti sa pracuje s algoritmom KNN (K-nearest neighboors). Hustota poľa je zamenená za rovinnosť plochy. Vzdialenosť medzi plochami neurčuje eukludianska geometria ale odchýlka ich normál. Robustnosť danej metódy, je evidentne závislá od správneho nastavenia veľkosti vzdialenosti susedných bodov. Daný algoritmus bol v práci odskúšaný na menšom, aj väčšom množstve bodov generovaných bodov. 3D point cloud segmentácia bola pomocou neho overovaná taktiež aj na skutočne meraných dátach.

Výhody algoritmov strojového učenia, je ich schopnosť automatickej klasifikácie a vysokej presnosti. Metóda vyžaduje dlhý proces učenia, ktorý vyžaduje na svoju kvalitnú prácu veľké množstvo vstupných dát. Je náročné poukázať v trénovacích modeloch na vzťahy medzi dátami tak, aby algoritmus bol komplexne propagovateľný, [33].

1.4 Záver analýzy článkov

Ako už bolo vyššie spomínané, segmentácia point cloudu nie je triviálny problém. Segmentačné algoritmy sa všeobecne používajú a boli primárne vytvárané pre potreby obrazovej analýzy. Napriek množstvu publikovaných článkov a vydaných kníh, neexistuje ani tam zatiaľ univerzálny segmentačný algoritmus. Preto je pri vyberaní algoritmu potrebné poznať jeho vlastnosti, ako aj charakteristiku aplikácie na ktorú bude použitý. Problém point cloud segmentácie je riešený prevažne v publikovaných článkoch a existuje len minimum knižných zdrojov zameraných konkrétne na túto problematiku.
Z analýzy môžeme pohodlne usúdiť, že každá skupina či konkrétny algoritmus má kladné ako aj záporné stránky v porovnaní s ostatnými, či už v rýchlosti, presnosti segmentácie, senzitivite na zašumených dátach, alebo robustnosti celého algoritmu.
Keďže cieľom mojej práce je porovnanie jednotlivých algoritmov, vybral som si také, ktoré prechádzajú celým spektrom vlastností. Vo vybranej pätici algoritmov teda nájdeme veľmi rýchle, ktorých výsledky sú často jednoducho ovplyvniteľné napríklad zašumenými bodmi alebo vyžadujú dodanie dát v chronologickom meraní. Naprogramované boli aj robustné algoritmy, ktorých vstup môže byť po jednoduchých úpravách nie len pole meraných bodov point cloudu, ale aj obrazová štruktúra. Kvôli ďaleko výraznejšiemu množstvu riadkov v naprogramovanom kóde, a teda aj množstve operácií ktoré sa od nich žiada zaostávajú za ostatnými rýchlosťou a jednoduchosťou myšlienky.

2 Použité algoritmy

Nasledujúca kapitola bude pozostávať z konkrétneho opisu princípu vybraných algoritmov. V každej podkapitole predstavíme jeden z pätice vybraných algoritmov. Štruktúra prestavenia bude členená na matematický popis princípov algoritmu, kde je vysvetlený podrobný princíp fungovania algoritmu, navrhnutú štruktúru algoritmu, ktorá zahŕňa dôkladný popis nami vytvoreného kódu. V každej podkapitole sa nachádza taktiež teoretické zhodnotenie časovej zložitosti opisovaného algoritmu a prislúchajúci vývojový diagram. Poslednej podkapitola opíšeme spracovanie dát pred a po aplikovaní navrhovaných algoritmov. To je rozdelené na časť preprocessingu, kde opisujeme spracovanie dát z Lidar zariadenia a post-processingu, kde opisujeme novovytvorenú funkciu na zlepšenie výsledkov algoritmov.

2.1 SEF algoritmus

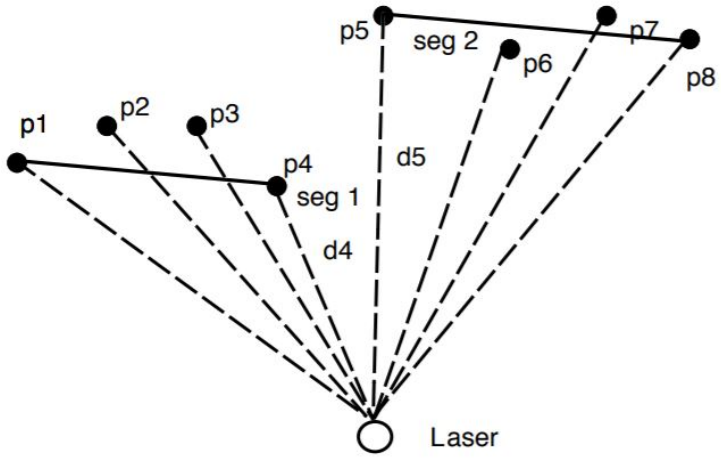
Segmentácia založená na myšlienke SEF algoritmu ("successive edge following") využíva vlastnosti získavania dát pomocou Lidar zariadenia. Popisovaný algoritmus obzvlášť vyniká vo výpočtovej rýchlosti aj pri veľkom množstve dátových bodov. Algoritmus považuje plný scan 360°, za sekvenciu bodov P_c tak, že každý bod je predstavený v polárnej sústave dvojicou súradníc (r_n, q_n), [30], [38].

2.1.1 Matematický princíp algoritmu

Hlavná rozhodovacia časť algoritmu je podľa podmienky :
Ak $D(\rho_i, \rho_{i-1} > D_{max})$, *tak potom segmenty rozdel*, *Inak nerob nic*
Vzdialenosť $D(r_i, r_{i-1})$ sa určuje ako rozdiel dvoch po sebe idúcich naskenovaných polomerov, viď rovnica 2.1.

$$D(\rho_i, \rho_{i-1}) = \rho_i - \rho_{i-1} \tag{2.1}$$

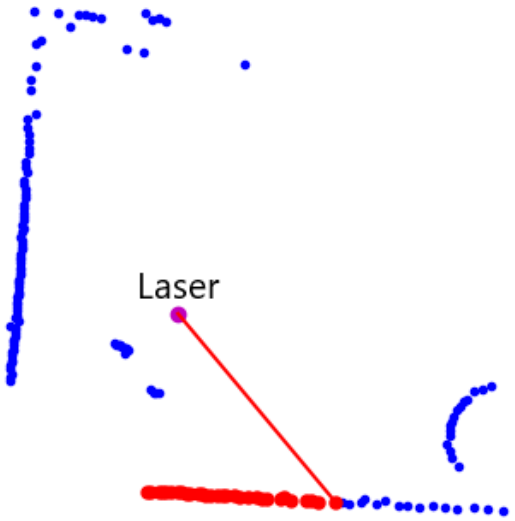
Uvádzaný princíp je veľmi náchylný na stanovanie správnej hodnoty D_{max}, teda maximálnej dovolenej dĺžky zmeny polomeru. Na zvolenie výpočtu medznej hodnoty D_{max}, existuje viacero spôsobov. Princíp funkcie popisovaného algoritmu, môžeme vidieť na obr. 2.1.



Obr. 2.1: Rozdelenie pomocou SEF algoritmu, prevzaté z [38]

2.1.2 Navrhnutá štruktúra algoritmu

SEF algoritmus sme napísali procedurálnou formou. Funkcia sef_alg() iteratívnym prejdením poľa bodov si inicializuje hraničné hodnoty pre rozdelenie polí. Vstupom do funkcie je pole bodov v polárnom súradnicovom systéme. Následným prejdením poľa algoritmus pridáva do výstupného poľa postupne už rozdelené skupiny bodov. Tie sú predelené podľa podmienky delenia $D(\rho_i, \rho_{i-1} > D_{max})$. Inicializácia tejto hodnoty je počítaná v závislosti na najmenšiu nenulovú zmenu vzdialenosti medzi bodmi. Táto hodnota sa ukázala v priebehu všetkých experimentov ako v podstate konštantná v závislosti na podmienkach. Vykresľovanie ako aj výstup polí je prepočítavaný do karteziánskych súradníc. Koniec funkcie je doplnení o spätné spojenie prvej a posledne meranej množiny bodov. Keďže tieto susediace množiny, by mohli byť nesprávne rozdelené, nastáva prekontrolovanie zmeny polomeru aj v danom mieste. Ak je podmienka splnená, dané dve množiny sú spojené. V opačnom prípade funkcia vyhodí nezmenené množiny bodov. Na obr. 2.2, môžeme vidieť prácu takto fungujúceho algoritmu. Červeno vyznačená množina bodov, symbolizuje neukončenú skupinu bodov jednej triedy, ktorú algoritmus vyhodnotil počas postupnej rotácie v protismere hodinových ručičiek.

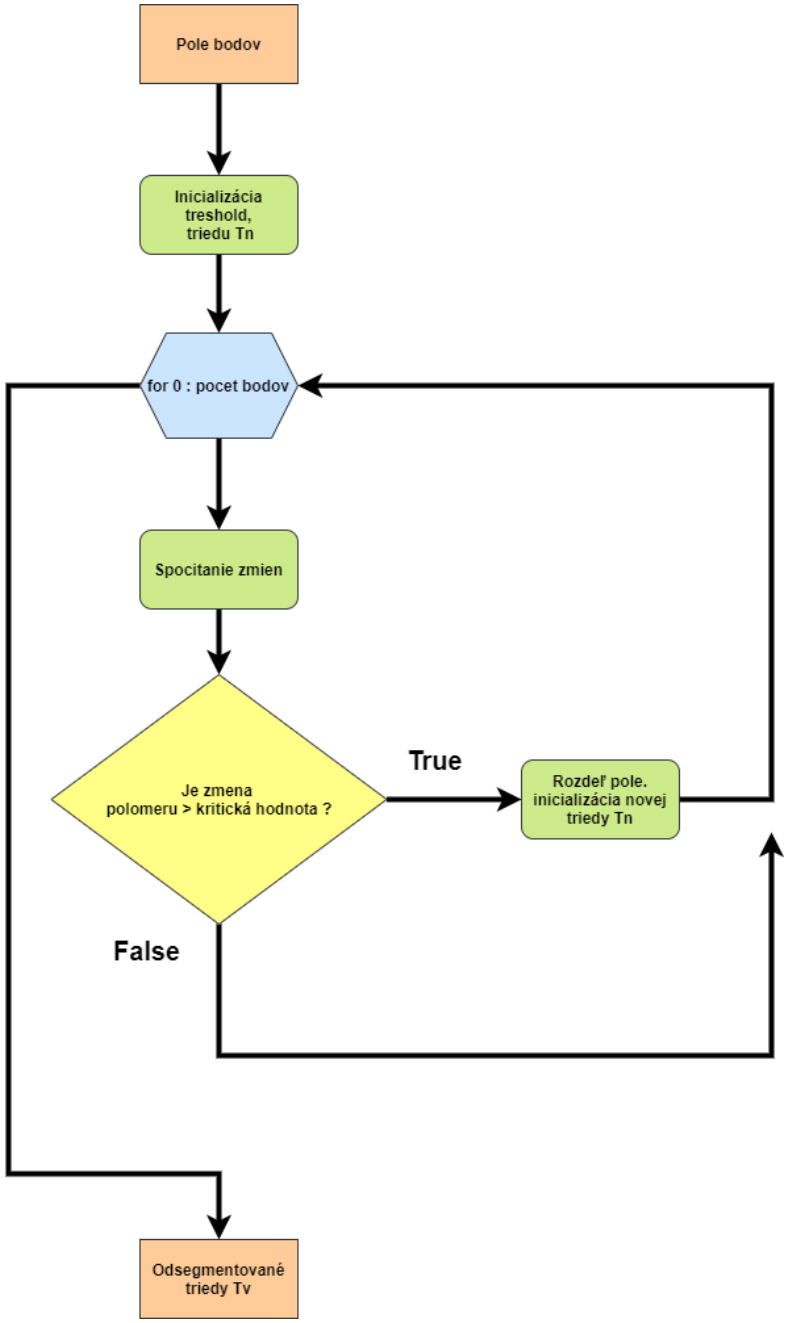


Obr. 2.2: Proces SEF algoritmu

2.1.3 Časová zložitosť algoritmu

Opísaný algoritmus má problémy pri husto naskenovaných dátach okolia, v prípadoch rozlišovania ostrých rohov. Tie nedokáže v rôznych situáciach správne identifikovať. Vyniká však už spomenutou vysokou výpočtovou rýchlosťou. Výpočtová zložitosť je vzhľadom na pole bodov n lineárna $O(n)$. Algoritmus je možné pri správnych vstupných parametroch používať samostatne. Súčasne má potenciál byť používaný aj ako hrubá segmentácia, respektíve prvá časť nejakého zložitejšieho algoritmu. Vývojový diagram vidíme na obr. 2.3.

2.1.4 Vývojový diagram



Obr. 2.3: Vývojový diagram SEF algoritmu

2.2 Line tracking algoritmus

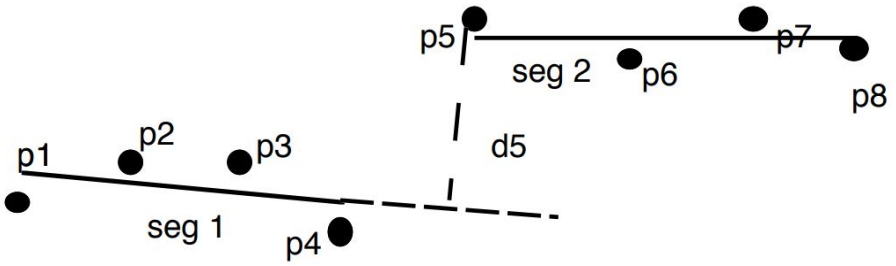
Tento algoritmus vyniká svojou jednoduchosťou. Linear Tracking algoritmus ("nasledovania priamky") je zo skupiny iteračných algoritmov, ktorý taktiež vyžadujú pre svoju správnu činnosť chronologický vstup dát.

2.2.1 Matematický princíp algoritmu

Algoritmus začína výberom prvých dvoch bodov poľa p_n, p_{n+1}, z ktorých vytvára priamku p. Následne iteračným postupom pridáva ďalšie body ktoré spĺňajú ktoré splnia kritérium pridelenia. Jedná sa o „Bottom-up" metódu, pričom sa začína jedným rozrastajúcim sa bodom. Podmienkou pridania ďalšieho bodu je vzdialenosť nového bodu p_{n+2} k priamke p podľa rovnice 2.2, [3], [27], [38].

$$T_{n_{i+2}} < T_{max} \tag{2.2}$$

V prípade že vzdialenosť je menšia, pridá sa daný bod k rozrastajúcej sa skupine. Nastáva prepočítanie priamky pomocou metódou najmenších štvorcov. Tá vytvára aproximáciu lineárnej krivky z poľa bodov. V prípade že nasledujúci bod dosiahne väčšiu kolmú vzdialenosť, nastáva delenie množiny, viď obr. 2.4. Daný bod je pritom považovaný už ako začiatok novej množiny. Princíp pridávania bodov pozdĺž priamky vidíme na obrázku číslo 2.4.



Obr. 2.4: Rozdelenie množiny bodov v mieste prekročenia kritickej hodnoty(p5), prevzaté z [3]

Aproximácie priamky sme riešili metódou najmenších štvorcov. Každým pridaním nového bodu nastáva pomocou tejto techniky prepočítanie danej priamky.
Všeobecná rovnica priamky má v dvojrozmernom priestore tvar $a * x + b * y + d = 0$. Konečná priamka vzniká minimalizáciou funkcie E, v tvare 2.3, [10].

$$E(a, b, c) = \sum_{i=1}^{n} (a * x_i + b * y_i - d)^2 \tag{2.3}$$

Keďže minimalizáciu funkcie 2.3, dostávame pomocou deriváciou podľa vzdialenosti rovnej k nule, dostávame rovnicu 2.4, pri určení ostatných parametrov 2.5.

$$0 = \frac{\partial E}{\partial d} = -2 * \sum_{i=1}^{n} (a * x_i + b * y_i - d) \tag{2.4}$$

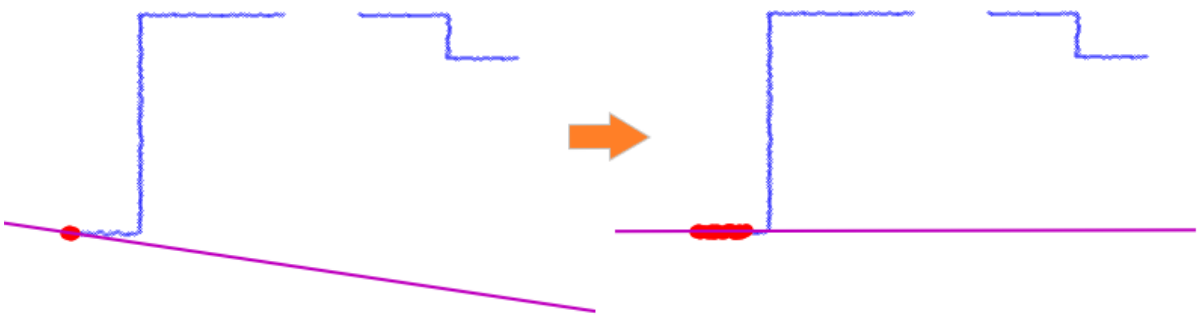
$$d = a * \bar{x} + b * \bar{y}, \quad \bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i \;\; a \;\; \bar{y} = \frac{1}{n} \sum_{i=1}^{n} y_i \tag{2.5}$$

Rovnica energie po našich úpravách má konečný tvar 2.6.

$$E = \sum_{i=1}^{n} [a * (x_i - \bar{x}) + b * (y_i - \bar{y})]^2 \tag{2.6}$$

2.2.2 Navrhnutá štruktúra algoritmu

Algoritmus je postavený na cyklickom prejdení bodov v karteziánskej súradnicovej sústave. Pri prvom prejdení sa inicializujú hraničné hodnoty delenia. Tie sú vytvorené na v závislosti na minimálnu vzdialenosť vstupných bodov. Druhý prechod poľa bodov už v rámci funkcie lt_funkcia() sprostredkúva samotné delenie bodov a ich segmentáciu do rôznych tried. V prípade nedostatočného počtu bodov v pracovnej triede T_n, sa dokladajú body potrebné na vytvorenie priamky, ktorá je následne definovaná práve nimi. Algoritmus potom rozhoduje či nasledujúci bod poľa, spĺňa podmienku pridania do pracovnej triedy T_n. Ak je odpoveď kladná, pracovná trieda sa rozrastie a je kontrolovaný ďalší bod. V prípade, že bod nie je v dostatočnej vzdialenosti, nastane prepočítanie priamky pomocou metódy najmenších štvorcov. Po jej vytvorení nastáva znova overenie kritéria rozdeľovania. Až v prípade že daný bod nesplní danú podmienku po prepočítaní priamky, je pole rozdelené a daný bod je začiatok novej množiny. Takýto dodatok dokáže daný algoritmus výrazne zrýchliť, nakoľko prepočítanie priamky pomocou metódy najmenších štvorcov nie je nutné robiť v každej iterácií, ale len keď je to evidentne nevyhnutné. Funkcia na konci vyhodnotí taktiež prvú a poslednú rozdelenú množinu, podobne ako v prípade SEF algoritmu (podkapitola 2.1). Takýmto ukončením sa zamedzí možnému nesprávnemu rozdeleniu susedných množín, ku ktorému by došlo kvôli zlému štartovaciemu bodu na začiatku funkcie. Popísaný princíp môžeme vidieť na obr. 2.5, kde je zobrazené postupné pridávanie nasledujúcich bodov množiny do triedy ako aj úprava natočenia priamky.



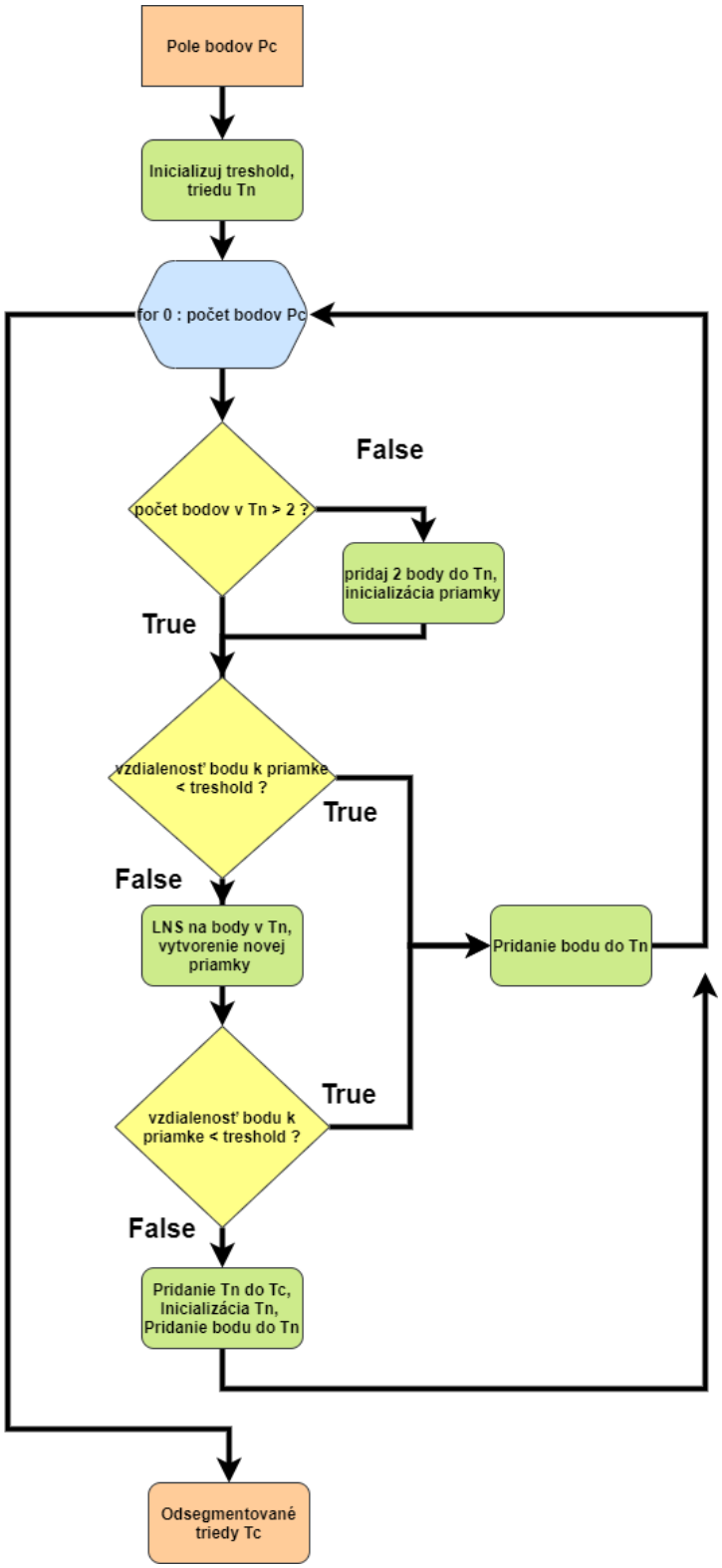
Obr. 2.5: Proces Line Tracking algoritmu

2.2.3 Časová zložitosť algoritmu

Veľkou výhodou vyššie popísaného algoritmu je rýchlosť a taktiež prispôsobovací charakter pre extrakciu priamok z poľa bodov. Algoritmus má niekdy problémy rozlišovať medzi zvyškovými bodmi, a bodmi podporujúcimi danú priamku. Pridaním opisovaného dodatku prepočítavania priamky iba v nevyhnutných prípadoch, sa zrýchli časová zložitosť z kvadratickej $O(n^2)$, na asymptoticky lineárnu $O(n)$. Pre najhorší prípad, kedy každý bod by tvoril vlastnú triedu, pracuje algoritmus s kvadratickou zložitosťou $O(n^2)$.

Vývojový diagram môžeme vidieť na obr. 2.6.

2.2.4 Vývojový diagram



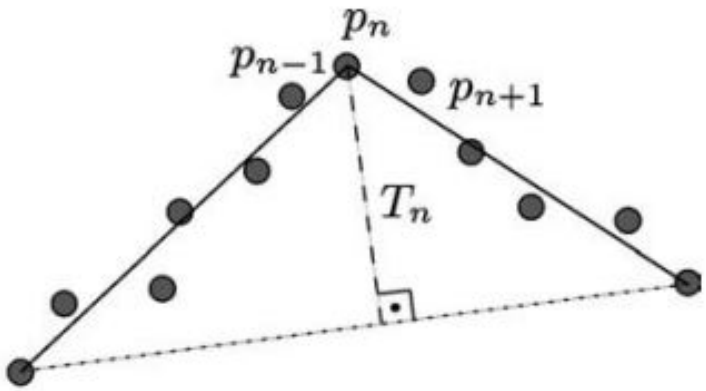
Obr. 2.6: Vývojový diagram Line Tracking algoritmus

2.3 Split and Merge algoritmus

Rekurzívna metóda je založená na „Top-down" prístupe segmentácie oblastí. Metóde sekunduje taktiež postup "Bottom-up" tak, že spolu vytvárajú plne funkčnú kombináciu algoritmov na segmentovanie množiny modov. Metóda je založená na čisto geometrickom prístupe. Základná myšlienka danej metódy je delenie poľa bodov až do chvíle, pokiaľ chyba v rámci každej triedy nie je dostatočne malá. Následné spájanie tried je založené na myšlienke, že ak má dvojica spojených rozdielnych tried normalizovanú chybu nižšiu ako je aspoň jedna z chýb tried osamotených, považujú sa tieto triedy za podobnú množinu a nastáva ich spojenie. Jedná sa o jednu z najpoužívanejších metód segmentácie kriviek v počítačovom videní.

2.3.1 Matematický princíp algoritmu

Metóda je vysoko závislá na počiatočných bodoch vytvárania priamok. Tento problém zdieľa spolu s ostatnými algoritmami v skupine rozrastajúcich oblastí(surface growing), z ktorých sú všetky algoritmy viac či menej ovplyvnené týmto faktom. Princíp samotného algoritmu, ako už bolo spomenuté vyššie, je rekurzívne delenie poľa bodov $P_n = \{pn_0, pn_1, ..., pn_n\}$ na triedy $P_n1 = \{pn1_0, pn1_1, ..., pn1_m\}$ a $P_n2 = \{pn2_0, pn2_1, ..., pn2_l\}$, tak že $|P_n| = |P_n1| + |P_n2|$, pri nedosiahnutí kritéria rozdelenia. Priamka p, vzniká teda ako prienik bodov a, b; *kde* $a, b \in P_n$. Rozdelenie daného poľa bodov nastáva pri splnení podmienky delenia pozdĺž priamky p, [3], [14], [38]. Kritériom rozdelenia množiny bodov pri tomto algoritme je najväčšia kolmá vzdialenosť bodu k vytvorenej priamke p prechádzajúcej dvomi bodmi. Na zistenie najväčšej vzdialenosti bodu k priamke sa vytvára priamka q taká, že $q \perp p$ a bod $E \in q$; keď E je najvzdialenejší bod pola od priamky p. V prípade že vzdialenosť prekročí stanovenú hodnotu, rozdeľuje priamka q množinu bodov na dve časti. Koncové body novo vytvorených skupín sú zároveň body definujúce priamky pre ďalšiu segmentáciu. Takto prebieha proces až dokým maximálna vzdialenosť e_max (max. chyba) nie je menšia ako hraničná vzdialenosť. Tá je stanovená na začiatku lineárnym prejdením poľa bodov a určená v závislosti na stanovení najmenšej vzdialenosti medzi bodmi. Na obr. 2.7 vidíme, že rozdelenie pomocnej priamky (čiarkovaná), nastalo v najvzdialenejšom bode p_n. Bod p_n teda rozdeľuje množinu vykreslených bodov na dve časti.

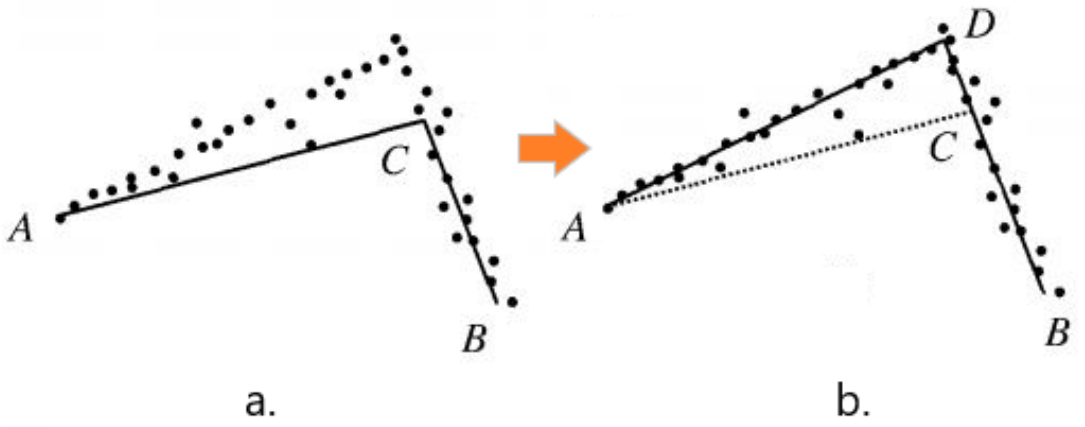


Obr. 2.7: Princíp Split algoritmus, prevzaté z [3]

Druhá časť algoritmu je definovaná ako Bottom-up metóda. Keďže samotné delenie

pomocou rozdeľovania priamok, vytvára presegmentované pole bodov, pracuje tento algoritmus v kombinácií so spájacou časťou „Merge". Myšlienka „merge" algoritmu je založená na postupnom spájaní jednotlivých množín bodov, pri splnení kritéria. Kritériom spájania je normalizovaná chyba spojeného poľa porovnávaná s jednotlivými dvomi triedami. V prípade že chyba spojeného poľa je menšia alebo rovná väčšej z dvoch chýb tried, nastáva spojenie daných tried v jednu, čoho dôsledkom je poľa zväčšenie bodov v množine alebo oprava pôvodnej množiny, viď obr. 2.8. Normalizovaná chyba je rátaná pre samostatnú aj spojenú skupinu podľa vzorca 2.7.

$$chyba = \frac{e_{max}}{D_{pola}} \tag{2.7}$$

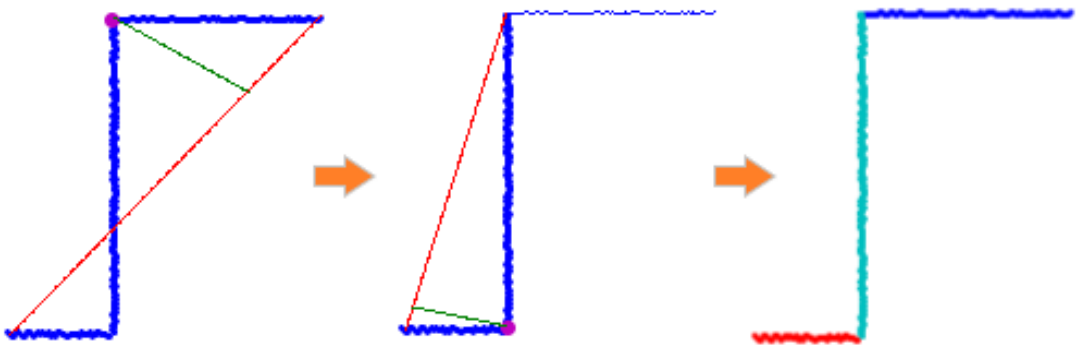


Obr. 2.8: Spätné spájanie bodov, prevzaté z [14]

2.3.2 Navrhnutá štruktúra algoritmu

Ako už bolo vyššie spomenuté, algoritmus je vytvorený rekurzívnou formou, a teda nastáva postupné volanie funkcie split() až dokým kritérium deliteľnosti nevyhodí obdržané pole. Prvým bodom algoritmu je rozhodnutie vybratia počiatočných bodov. Tie sú zvolené ako najvzdialenejšie body, a predstavujú najdlhšiu možnú priamku zo vstupného poľa. Vrámci funkcie split() sa vytvorí priamka prechádzajúca danými bodmi. V cykle sú prebehnuté všetky body poľa, pričom je nájdená a zapamätaná najväčšia kolmá vzdialenosť bodu k priamke. Tá je v našom prípade počítaná analyticky, pomocou funkcie max_vzdialenost(). Nastavenie tejto vzdialenosti je závislé od minimálne vzdialenosti medzi bodmi pri prvom lineárnom prebehnutí poľa bodov. V prípade prekročenia hranice nastáva rekurzívne delenie pre body vyskytujúce sa do bodu s maximálnou chybou a následne pre ostávajúcu časť množiny. Graficky ilustrovaný postup môžeme vidieť na jednoduchom prípade na obr. 2.9. Na obrázku sa pôvodná množina bodov (modrá) rozdelí postupne na 3 časti v miestach, ktoré sú označené fialovým bodom.

Pre maximálnu vzdialenosť menšiu alebo rovnú, ako je inicializovaná hranica, vracia funkcia vstupné pole bodov. V opačnom prípade nastáva ďalšie postupné delenie pola bodov. Body sú rozdelené na 2 časti, podľa polohy od priamky q kolmej na pôvodnú priamku p. Prvý a posledný bod nových rozdelených polí, sú zároveň najvzdialenejšie body pozdĺž



Obr. 2.9: Proces delenia poľa bodov

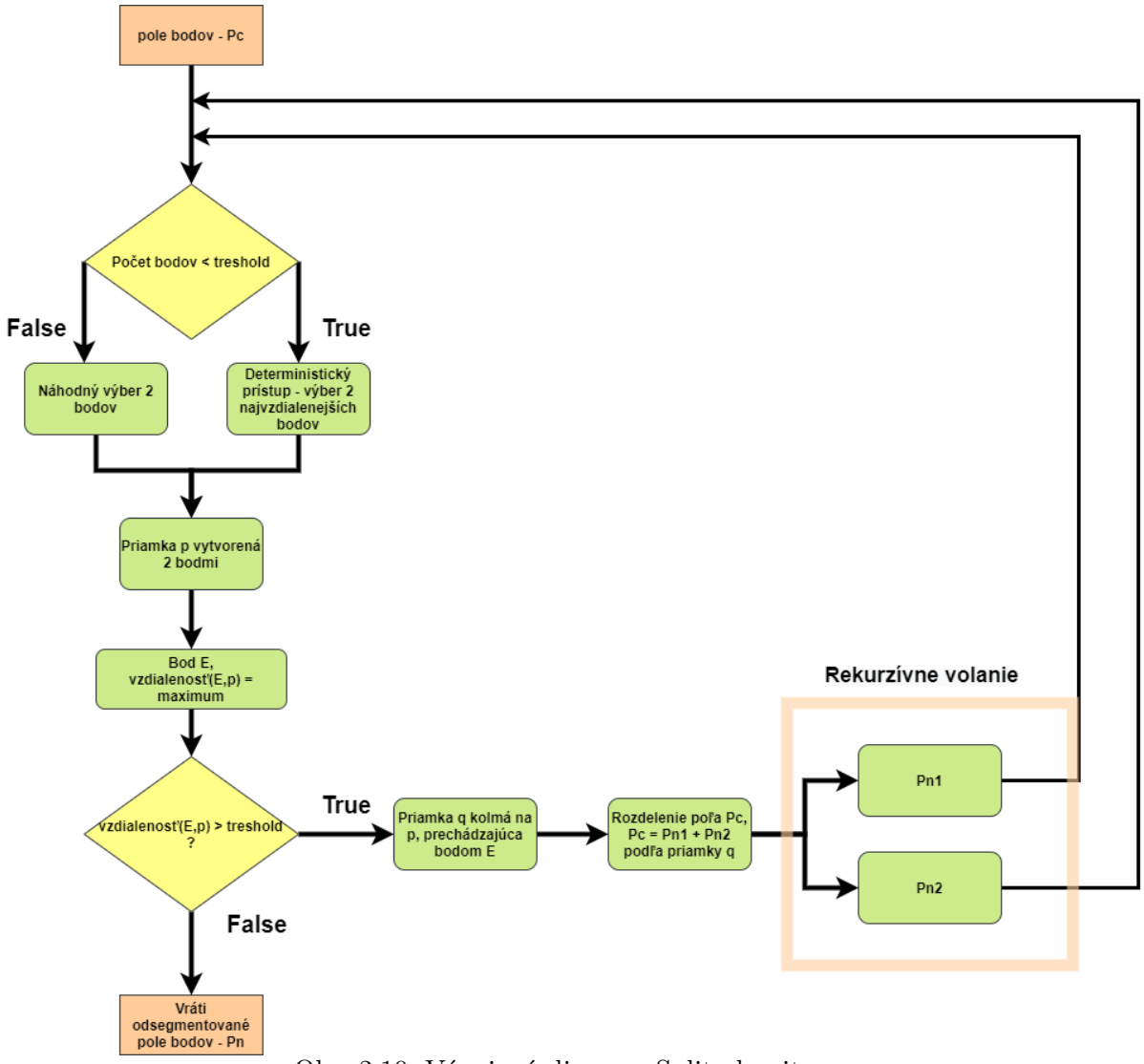
priamky p, kvôli iteratívnemu ukladaniu pre následné spracovanie. Pre takto vytvorené nové polia sú postupne volané funkcie split() ktoré fungujú až dokým chyba v rámci množiny nepresiahne hraničnú hodnotu.

Funkcia merging() je ako už bolo vyššie spomenuté vytvorená tak, že porovnáva rozdelenú a spoločnú normalizovanú chybu. Potreba použitia normalizovanej chyby je podstatná, nakoľko viac segmentov by aproximovali body s nižšou chybou ako jeden veľký segment. Algoritmus íteratívne prebehne rozdelené triedy zo segmentácie pomocou delenia priamok. Najskôr algoritmus počíta chybu jednotlivých polí bodov. Následne túto chybu porovnáva s polom bodov, ktorý vznikne spojením daných tried. V prípade že je spoločná chyba menšia, spojí tieto dve množiny a rekurzívne zavolá funkciu merging(). Takýmto postupom dosiahneme porovnávanie až do kým sa nevyčerpajú všetky, aj novovzniknuté množiny.

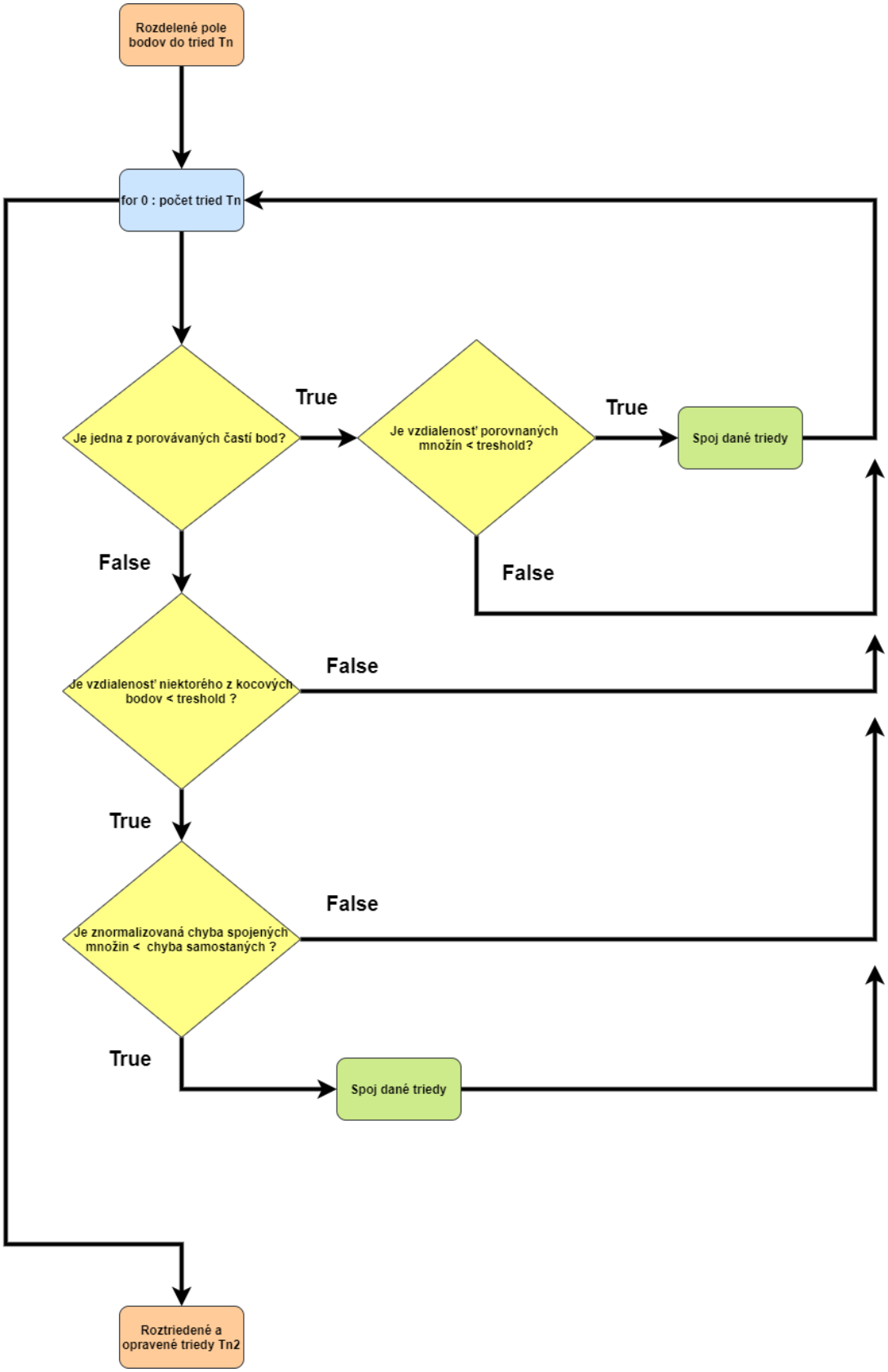
2.3.3 Časová zložitosť algoritmu

Daný algoritmus rozdeľovania má kvôli deleniu poľa na dve časti, časovú zložitosť $O(n * \log n)$, kde n je počet bodov. Tá zdieľa podobnosť so štruktúrami binárnych stromov. V najnepriaznivejšom prípade, algoritmus pracuje s časovou zložitosťou $O(n^2)$, kde n je počet bodov. Takýto prípad nastáva, keď každý bod bude tvoriť svoju vlastnú množinu. V prípade spájania množín, ktoré sa deje už po rozdelení množín pomocou delenia priamok, je časová zložitosť algoritmu $O(n^2)$, kde n je počet tried segmentácie.
Vývojový diagram nami vytvoreného algoritmu pre funkciu split() sa nachádza na obr. 2.10, a pre funkciu merging() sa nachádza na obrázku 2.11.

2.3.4 Vývojový diagram



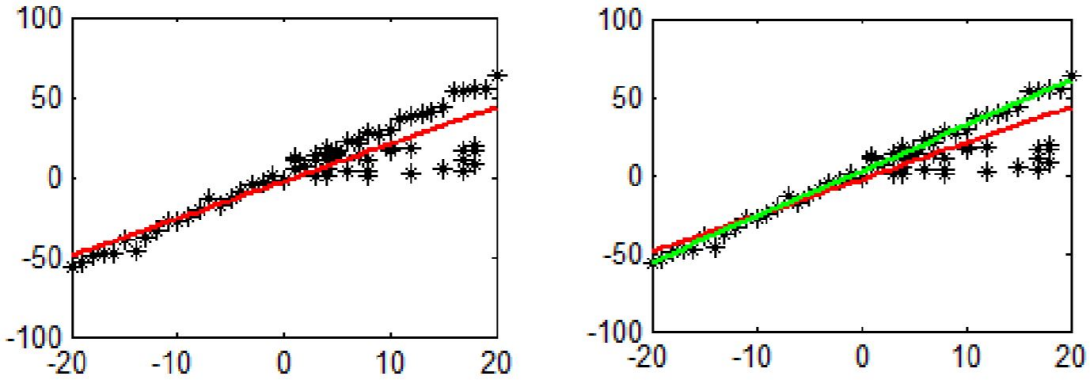
Obr. 2.10: Vývojový diagram Split algoritmus



Obr. 2.11: Vývojový diagram Merge algoritmus

2.4 Ransac algoritmus

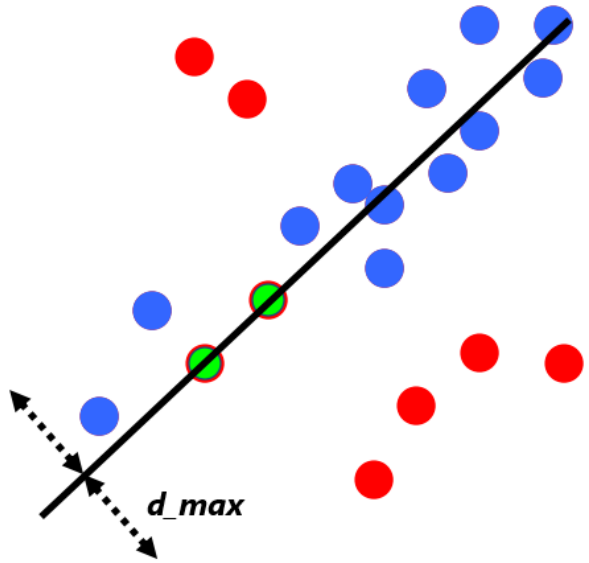
Ransac celým názvom („random sample consensus"), algoritmus patrí opäť ku konvekčným vyhľadávacím technikám v oblasti spracovania obrazu. Technika často využívajúca sa na aproximáciu bodov priamkou, zvyčajne najmä porovnávaná so stálicou v tejto oblasti, metódou najmenších štvorcov. Ransac ale založený na opačnom princípe oproti opísanej metóde najmenších štvorcov. Zatiaľ čo tie, sa snažia zohľadňovať pri aproximácií čo najväčší počet bodov, a postupnými iteráciami zmenšuje pole bodov ktoré použije pre aproximáciu. Algoritmus Ransac začína čo s najmenším množstvom bodov pola (pre priamku dva body, pre kružnicu 3 body), a postupne zväčšuje počet bodov pola prislúchajúcich k danej priamke, obr. 2.12 . V porovnaní je dôležité si všimnúť ako Ransac pracuje s nevhodnými, mimo stojacími bodmi, ktoré metóda najmenších štvorcov nekorektne do aproximácie započíta.



Obr. 2.12: Porovnanie LNS vs Ransac, prevzaté a upravené z [33]

2.4.1 Matematický princíp algoritmu

Ransac algoritmus je charakterizovaný trojicou špecifických parametrov. Jeho výsledky a správne fungovanie sú závislé na hodnote vzdialenosti k modelu priamky, celkovému počtu pokusov, a hodnote najmenšieho počtu bodov potrebného na vytvorenie oddelenej množiny. Samotný Ransac algoritmus je po získaní daných parametrov pomerne jednoduchý. Výberom dvoch náhodných bodov, minimálne potrebných pre vytvorenie daného útvaru (pre priamku 2 body), zostrojí priamku spojením týchto bodov. Následné nastáva hľadanie bodov poľa, ktoré sa nachádzajú v tolerancií určenej ako d_max, najväčšej dovolenej kolmej vzdialenosti bodu k vytvorenej priamke. Takýto princíp je ukázaný na obr. 2.13. V prípade že počet „vyhovujúcich bodov" je menší ako minimálny počet bodov b_min na zoskupenie daných do jednej množiny, pokračuje proces pre ďalšie náhodné body až do vopred vypočítaného množstva pokusov k. V prípade, že nájde minimálny počet bodov, vybrané body odseparuje a pokračuje ďalej, [9].



Obr. 2.13: Priradenie bodov, nachádzajúcich sa v tolerancií, prevzaté a upravené z [37]

Hodnota maximálneho počtu pokusov k, je založená na predpoklade, že na nájdenie podmnožiny z celkového počtu bodov n, je pravdepodobnosť w, že ľubovoľne zvolené číslo sa nachádza v tolerancií k modelu. V našom prípade je počet bodov n potrebných zostrojenie priamky rovných dvom $n = 2$. Očakávanú hodnotu získame $E(k)$ získame zo vzorca 2.8.

$$E(k) = b + 2 * (1 - b) * b + 3 * (1 - b)^2 * b + \ ..... \ + i * (1 - b)^{i-1} * b \tag{2.8}$$

Po zavedení vzorca pre geometrickú postupnosť a ďalšej algebraickej úprave dostávame 2.9.

$$E(k) = \frac{1}{b} = w^{-n}, \quad ak\ b = w^n \tag{2.9}$$

Tabuľka prepočtu hodnôt očakávanej hodnoty $E(k)$, pre príslušnú pravdepodobnosť w a počet bodov n (v našom prípade $n = 2$) nájdeme v tabuľke 2.1.

w	n=1	2	3	4	5
0.9	1.1	1.2	1.4	1.5	1.7
0.8	1.3	1.6	2.0	2.4	3.0
0.6	1.7	2.8	4.6	7.7	13
0.4	2.5	6.3	16	39	98
0.2	5	25	125	625	-

Tabuľka 2.1: Ransac tabuľka očakávaných hodnôt

Pravdepodobnosť teda, že nenájdeme korektnú priamku pri jednom pokuse získame ako $(1 - w^2)$. Z opačnej strany, ak chceme vedieť že aspoň jeden z náhodne vybraných bodov, bude vybratá množina bez chýb s pravdepodobnosťou z. Takúto rovnicu postavíme pravdepodobnosti, že pri k pokusoch nenájdeme ani jeden bod prislúchajúci modelu, dostávame rovnicu ľahko riešiteľnú rovnicu 2.10.

$$(1 - z) = (1 - w^2)^k \tag{2.10}$$

Z danej rovnosti je vhodné vyňať k, z čoho po ďalších algebraických úpravách vieme odvodiť pravdepodobnostnú rovnicu určujúcu celkový počet pokusov 2.11.

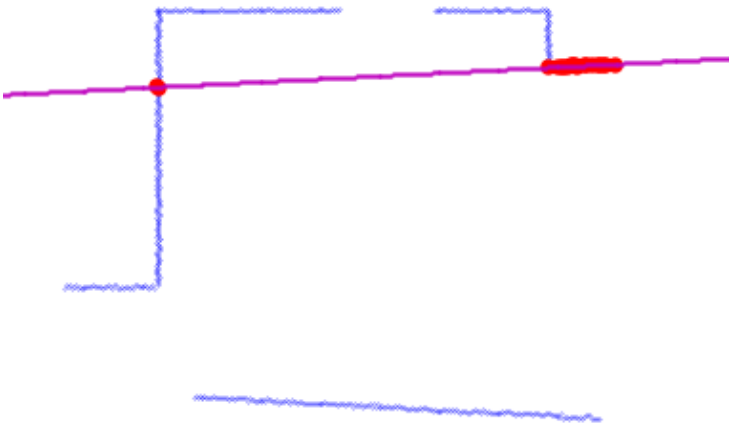
$$k = \frac{\log(1-z)}{\log(1-w^2)} \tag{2.11}$$

V prípade zaistenia napríklad 90% pravdepodobnosti, že dosiahneme aspoň jednu bezchybnú selekciu, z rovnice 2.12 dostávame že budeme potrebovať 25 pokusov.

$$k = \frac{\log(0.9)}{\log(1-\frac{1}{11})} = 24.16 \tag{2.12}$$

2.4.2 Navrhnutá štruktúra algoritmu

Nami vytvorený algoritmus začína inicializáciou maximálneho počtu iterácií k pre hodnoty $w = 0,2$ a $z = 0.9$. V cykle sa vytvára priamka z náhodných bodov, a zisťuje či počet bodov P_n ktorých vzdialenosť ku priamke je menšia ako d_max, a zároveň je počet bodov väčší ako minimálna hodnota b_min. Hodnota d_max je závislá od minimálnej vzdialenosti bodov. Počiatočná hodnota b_min sa počíta percenta vstupných bodov, a je postupne znižovaná. V prípade kladnej odpovede, vytvára z daných bodov priamku získanú pomocou metódy najmenších štvorcov, a následne k nej segmentuje pole bodov podľa ich vzdialenosti d_max. Metóda najmenších štvorcov má nastavenú zväčšenú hodnotu d_max, pre získanie väčšieho počtu bodov pozdĺž priamky. Nakoľko metóda najmenších štvorcov už po takejto pred úprave nepracuje s náhodne vybranými bodmi, je jej aproximácia priamky danými bodmi lepšia. Po zoskupení vytvorenej triedy, ktoré môžeme vidieť na obr. 2.14, nastáva jej odobrania z poľa bodov. Funkcia v ukázanom prípade našla pozdĺž priamky aj nesprávnu množinu bodov na ľavej strane obrázka. Takáto charakteristika je pre daný algoritmus typická, a riešenie opíšeme v podkapitole 2.6. Odobranie nájdenej časti bodov sme riešili ako rekurzívne volanie funkcie ransac(), už zo zníženým celkovým počtov bodov poľa, ako aj počtom pokusov. Iteratívne hľadanie nastáva až do bodu, že sa vyčerpajú možné pokusy alebo zostávajúci počet bodov je menší ako minimálny. Táto metóda je naprogramovaná tak, že nastáva postupné znižovanie minimálneho počtu bodov b_min, v závislosti od maximálnej získavanej po ukončení funkcie ransac() funkcie vyrátaj_chybu(). Funkcia vyrátaj_chybu() vracia maximálnu chybu zoradeného poľa bodov, ako maximálnu vzdialenosť k priamke prechádzajúcej medzi prvým a posledným prvkom poľa.

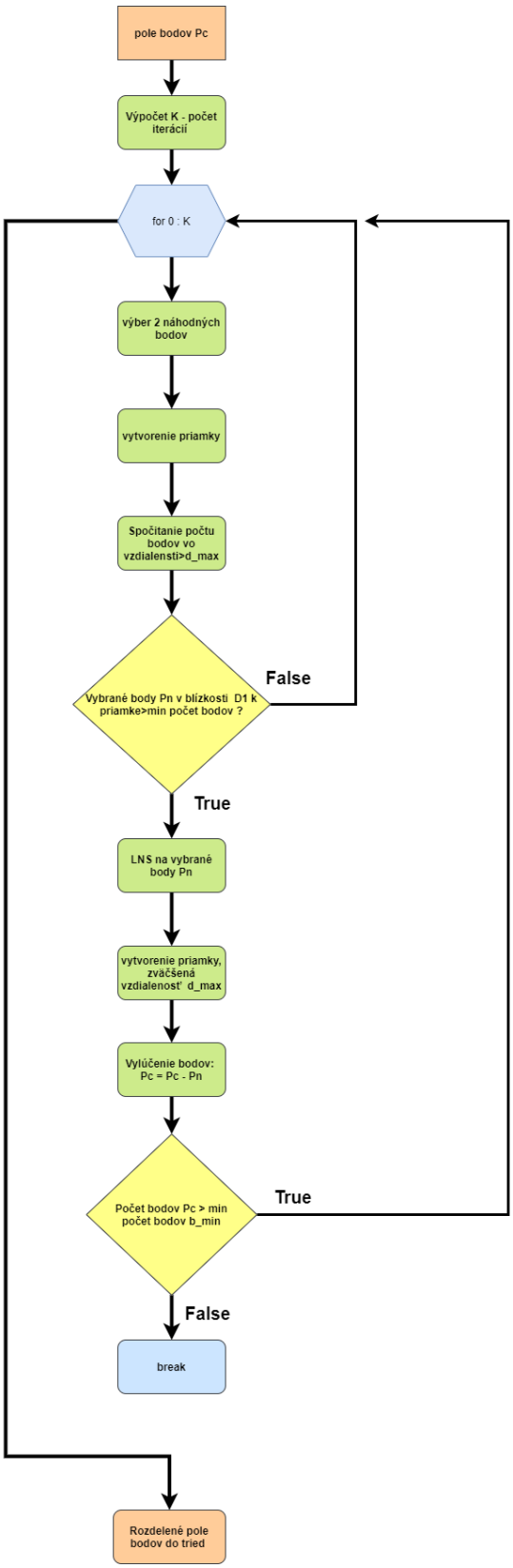


Obr. 2.14: Proces priradenia bodov pozdĺž priamky

2.4.3 Časová zložitosť algoritmu

Po správnom určení parametrov, funguje uvedený algoritmus pomerne rýchlo. Aj pri väčšej množine vstupných bodoch nenastáva veľké spomalenie, nakoľko algoritmus nepotrebuje žiadne zoradenie alebo deterministický výber počiatočných hodnôt. Na druhej strane môže byť výsledok zriedka ovplyvnený náhodným vybraním počiatočných bodov. Algoritmus má časovú náročnosť závislú na počte pokusov, a preto dosahuje konštantnú závislosť výpočtu $O(1)$. Na obr. 2.15 môžeme vidieť vývojový diagram daného algoritmu.

2.4.4 Vývojový diagram



Obr. 2.15: Ransac vývojový diagram

2.5 Hough algoritmus

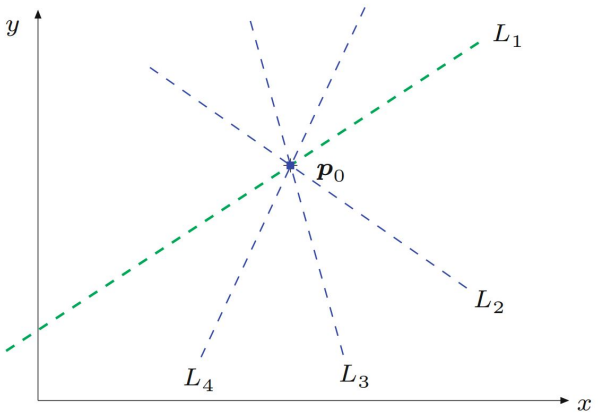
Hough transformácia je metóda segmentácia, prvotne vynájdená Paulom Houghom, a bola podaná ako americký patent. Je to robustná metóda, ktorá je využívaná najmä v oblasti počítačového videnia. Je založená na prevedení problému segmentácie na problém hľadania extrémov v Hough priestore. Metóda dokáže segmentovať primárne body formujúce sa do jednoduchých matematických útvarov ako úsečka, kruh atď, [4].

2.5.1 Matematický princíp algoritmu

Všeobecný prístup daného algoritmu je založený na parametrizovaný 2D priamky do všeobecnej formy 2.13.

$$y = a * x + b \tag{2.13}$$

Cieľ je nájsť také koeficienty a, b, ktoré definujú vyhovujúcu úsečku veľkým počtom bodov point cloudu. Kreslenie každej možnej priamky vyhovujúcej daným vstupným bodom, je aj pri zdiskretizovanom stave považované za metódu hrubej sily ("brute force") a časovo veľmi náročné. Takýmto prístupom nastáva skoro "nekonečne" veľa možností priamok v danom rozložení poľa bodov obrázku.
Hough transformácia rieši tento problém z opačnej strany. Kreslením priamok dané len pre dané body point clodu, ukázané na obr. 2.16 a nájdenie vyhovujúcich je len prevedenie problému na hľadanie bodov extrémov v akumulačnej matici. [4]



Obr. 2.16: Vytváranie priamok prechádzajúcich daným bodom, prevzaté z [4]

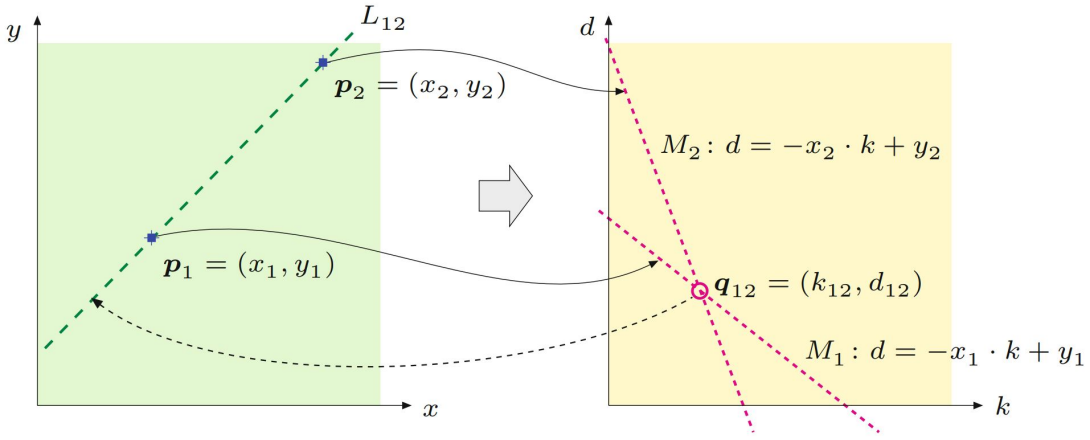
Daný algoritmus prevádza každý bod v (euklidovskom priestore) do parametrického Hough priestoru. Bod obrazu je zobrazený v Hough priestore ako úsečka, viď obr. 2.17. Priamka je vyobrazená ako bod v Hough priestore, viď tabuľka 2.2.

Euklidovský priestor			Hough priestor	
Bod	$p_i = (x_i, y_i)$	$\leftrightarrow$	$M_i: \; d = -x_i * k + y_i$	Úsečka
Úsečka	$L_j : y = k_j * x + d_j$	$\leftrightarrow$	$q_j = (k_j * d_j)$	Bod

Tabuľka 2.2: Vzťah medzi Hough a Euklidovským priestorom

Algoritmus vytvára akumulačnú maticu, ktorej prvky sa iteračne zvyšujú pri zapisovaní jednotlivých priamok. Akumulačná matica má rozmery $[polomer \times uhol(0 - \pi)]$. Pre prípady vertikálnych priamok, kedy koeficient $a = \infty$, počítame s Hessian formou reprezentácie priamky 2.14.

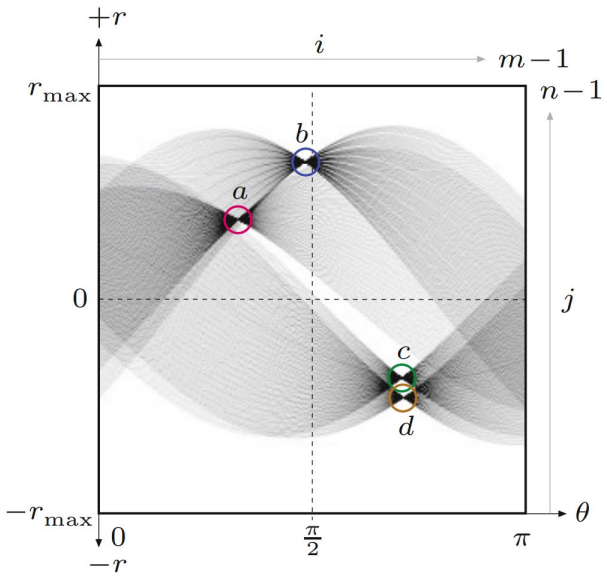
$$x * \cos\theta + y * \sin\theta = r \tag{2.14}$$



(a) x/y Obrazový priestor (b) k/d Hough priestor

Obr. 2.17: Prevedenie bodu do Hough priestoru, prevzaté a upravené z [4]

Pri akumulačnej matici, nastáva posunutie bodov do stredu „obrázka". Funkcia akumulačnej matice je teda hľadanie najviac označených miest medzi jej prvkami.
Prípad pretnutia N úsečiek v Hough priestore znamená, že N bodov v euklidovskom priestore leží na priamke $y = k' * x + d'$. Čím väčšia je hodnota bodu v akumulačnej matici, tým väčšia je pravdepodobnosť výskytu priamky v euklidovskom priestore. V Hough priestore sa body nezobrazujú ako úsečky. Body vytvárajú v parametrickom priestore sinusoidy, kvôli danému parametrickému opisu, viď obr. 2.18.
Správna detekcia a nájdenie vrcholov extrémov akumulačnej matice, nie je triviálna úloha. Daný problém znásobuje aj fakt, že aj pri rovných priamkach v euklidovskom priestore, nenastáva kvôli diskretizácií ich pretnutie v akumulačnej matici práve v jednom mieste. Ich pretnutie je rozložené do malej oblasti v okolí ideálneho bodu. Problém hľadania len extrémnych bodov, teda nie je možné riešiť len lineárnym prejdením 2 rozmerným polom, a nachádzaním extrémnych bodov. Daný krok je možné riešiť viacerými postupmi, pričom v našom prípade bolo použité prvotné odfiltrovanie nevhodných bodov a následná detekcia správnych pomocou výberu z maxím vrámci konvolučnej matice.



Obr. 2.18: Akumulačnej matica, prevzaté z [4]

2.5.2 Navrhnutá štruktúra algoritmu

Algoritmus je písaný v procedurálnej forme programovania. Funkcia houghTransfor() sa stará o vytvorenie akumulačnej matice ako aj zhodnotenie a nájdenie najpravdepodobnejších priamok. Inicializácia akumulačnej matice je závislá na veľkosti kroku d_ρ ad_p. So zväčšujúcimi sa hodnotami d_ρ ad_p, sa zvyšuje nepresnosť vykreslení v akumulačnej matici, ale zvyšuje výpočtová náročnosť algoritmu. V našom prípade sme inkrementy pridávali do akumulačnej matice do miest vhodných miest vyrátaných podľa vzorcov 2.15.

$$d_\theta = \frac{\pi}{m} \quad a \quad d_r = \frac{\sqrt{M^2 + N^2}}{n} \tag{2.15}$$

Pričom m a n je počet hodnôt polomeru a uhlu v poradí. Pre každý bod poľa, je v akumulačnej vytvorená krivka ním prechádzajúca v uhlovom rozpätí $\theta_i = (0, \pi)$ s krokom d_θ a výpočtom konkrétneho uhlu 2.16.

$$r(\theta_i) = (u - x_r) * \cos(\theta_i) + (v - y_r) * \sin(\theta_i), \quad kde\ u, v\ je\ stred\ pola \tag{2.16}$$

Diskrétny index pre daný bod v akumulačnej matici je získaný pomocou vzorca 2.17.

$$j = j_0 + round(\frac{r(\theta_i)}{d_r}) \tag{2.17}$$

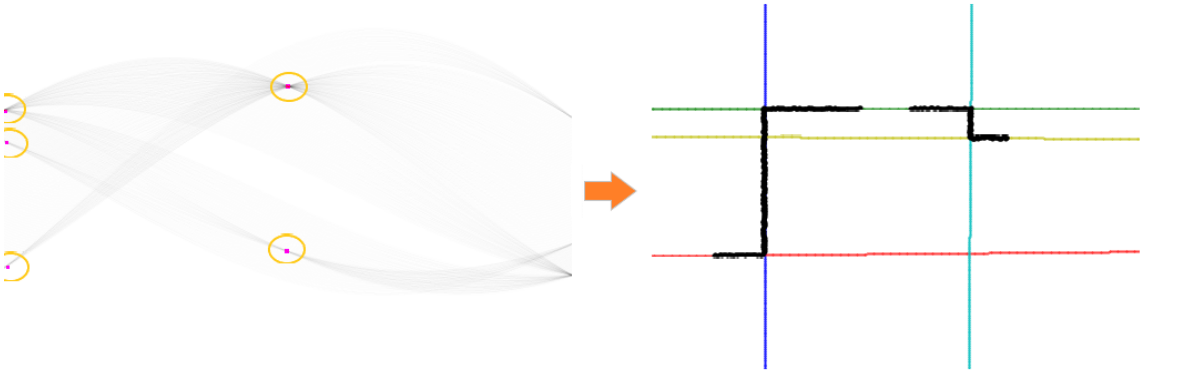
V dvojitom cykle sa takto naplní akumulačná matica. Nájdenie extrémov spočíva v potlačení ne extrémnych hodnôt v okolí každého prvky matice. V prípade že pre prvok matice $M(m \times n)$, existuje sused v určitom okolí, ktorého hodnota je väčšia, je hodnota prvku $M(m \times n)$ znížená. Veľkosť okolia je definovaná ako časť akumulačnej matice. V opačnom prípade sa pokračuje na ďalší prvok a hodnota sa nemení.
Po nájdení extrémov nastáva prevedenie bodov akumulačnej matice z Hough priestoru späť do euklidovského priestoru. Pred prevedením sú body zoradené podľa zostupného

čísla počtu hlasov pre daný bod akumulačnej matice. Takto zoradené priamky sú výstupom vyššie spomenutej $houghTransfor()$ funkcie. Popísanou rovnicou v tvare 2.18, a spätná parametrizácia nastáva pomocou rovnice v tvare 2.19.

$$L_k = \langle\, \theta_k, r_k, \alpha_k \,\rangle \tag{2.18}$$

$$\theta_i = i * d_\theta \;, \quad r_j = (j - j_0) * d_r \tag{2.19}$$

Delenie bodov sme naprogramovali ako pridelenie jednotlivých bodov najviac pravdepodobným úsečkám. Po zoradení sú vo funkcií delenie_bodov() v cykle porovnávané body pola s najviac pravdepodobnými priamkami. Kritérium porovnávania predstavuje vzdialenosť bodu od priamky. Tá je inicializovaná podľa minimálnej vzdialenosti vstupnej množiny bodov. Pri každom správnom priradení bodov k priamke, sú tieto body z pola odstránené a priradené do výslednej segmentovanej triedy. Podobné odoberanie bodov z poľa, bolo vysvetlené v Ransac algoritme. Takto nastáva rozdelenie bodov pola point cloudu. Ukončujúcim kritériom je buď vyčerpanie všetkých použiteľných priamok z Hough algoritmu, alebo nedostatočný počet ostávajúcich bodov. Na obr. 2.19, môžeme vidieť lokalizáciu extrémov v akumulačnej matici a ich prevedenie do priamok v pôvodnom obrázku.



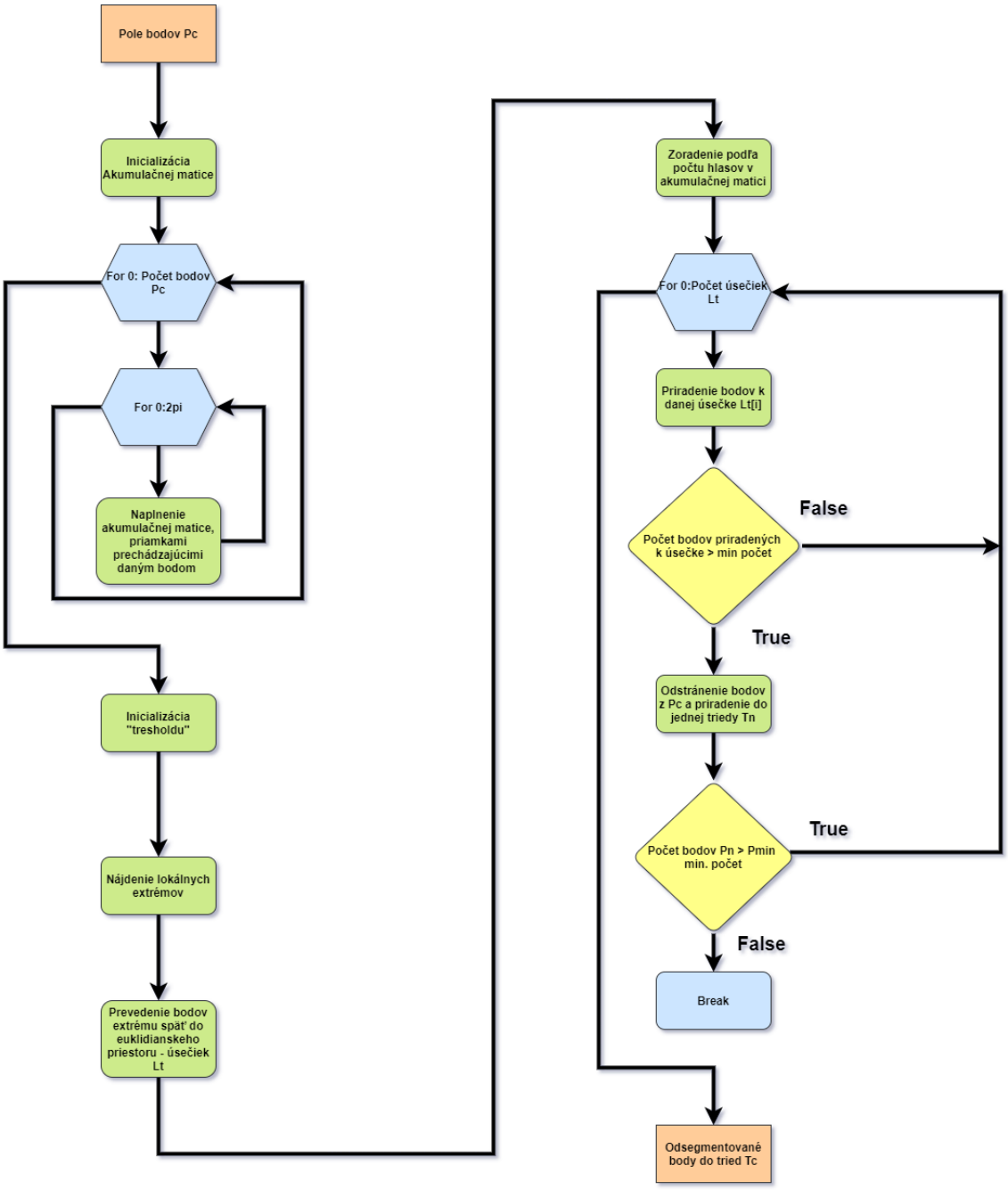
a. Lokalizácia extrémov v akumulačnej matici b. Prevedenie extrémov do pôvodného obrázka

Obr. 2.19: Proces Hough algoritmu

2.5.3 Časová zložitosť algoritmu

Rýchlosť výpočtu daného algoritmu závisí jednak o veľkosti vstupného poľa na ktorom sú vytvárané možné priamky, tak aj od veľkosti akumulačnej matice. Veľkosť akumulačnej matice je primárne závislá od rozlíšenia uhlového delenia d_θ. Ak teda použijeme šírku akumulačnej matice m, tak môžeme tvrdiť, že Hough algoritmus má časovú zložitosť $O(m \times n)$, kde n je celkový počet prvok poľa. Vývojový diagram názorne ukazujúci priebeh algoritmu sa nachádza na obr. 2.20.

2.5.4 Vývojový diagram



Obr. 2.20: Vývojový diagram Hough algoritmus

2.6 Spracovanie dát

V nasledujúcej podkapitole popisujeme spracovanie dát pred a po aplikácií algoritmov opisovaných vyššie v tejto kapitole. V časti preprocessing sme uviedli spracovanie obdržaných dát z Lidaru, ich filtrovanie pre potreby Hough alebo Ransac algoritmu či sekvenčné radenia poľa bodov pre potreby viacerých meraní v prípade SEF a Line tracking algoritmu. Post-processing prináša nami vytvorenú funkciu na spracovanie výsledkov z algoritmov. Funkcia je vytvorená na základe potreby úpravy segmentácie v niektorých prípadoch výsledkov prezentovaných algoritmov.

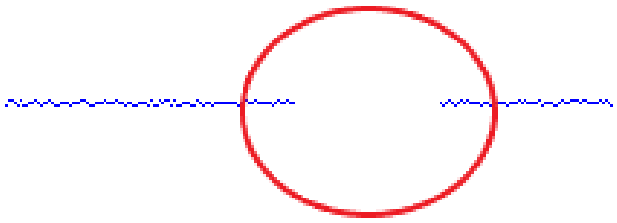
2.6.1 Preprocessing obdržaných dát

Priamo z Lidaru obdržané dáta, vychádzajú vo forme matice $M(m \times n)$. Po odstránení hlavičky, sú jednotlivé merania uložené v stĺpcoch matice M tak, že matica obsahuje v našom prípade 360 stĺpcov. Každý stĺpec teda indexuje 1° uhlového posuvu. Počet stĺpcov ako aj jemnosť delenia je vlastnosť nášho používaného Lidaru, popísaného v podkapitole 1.1. Vytvorená funkcia dokáže pracovať aj s jemnejším uhlovým, teda s delením uhla nižším ako 1°. Hodnoty na danom indexe označujú nameranú vzdialenosť objektu od meracieho zariadenia. V prípade, že meraná časť sa nachádza v príliš veľkej vzdialenosti od Lidaru, je reprezentujúci index matice vyplnený hodnotou inf. Riadky matice označujú jednotlivé merania vykonávané Lidar zariadením, teda každé otočenie zariadenia o celú otáčku sa zapisuje do nového riadku.
Preprocessing dát dodaných vo vyššie opísanej štruktúre bol prevedený pre požiadavky jednotlivých algoritmov.
Algoritmus SEF, pracuje priamo s dátami vo forme polárnych súradníc. Podstatnou podmienkou je chronologickosť radených dát. Preto v prípadoch pre segmentáciu viacerých meraní naraz, bolo nevyhnutné vytvorenenie vstupného poľa pomocou postupného radenia indexov jednotlivých stĺpcov za sebou.
Linear tracking algoritmus, bol vytvorený vo forme spracovania dát v karteziánskych súradniciach. Preto boli dáta z Lidar zariadenia prevedené na takúto formu. LT algoritmus je taktiež závislí od sekvenčne radených bodov merania. Taktiež sa tu teda vyskytoval preprocessing sekvenčného radenia dát, popísaný v SEF preprocessingu dát.
Robustnejšie tvorené algoritmy Split & Merge, Ransac a Hough transformácia pracujú s poľom dát v karteziánskom súradnom systéme. Výsledky týchto algoritmov sú nezávislé od usporiadania poľa vstupných dát. Algoritmy boli vytvorené tak, že dokážu segmentovať aj dáta vo forme obrázkov, prípadne vo forme „bitmapy". Prezentované algoritmy sú teda veľmi vhodné na segmentáciu viacerých meraní naraz, nakoľko nepotrebujú špeciálne radenia bodov.
Najmä pre potreby hlasovacích algoritmov (Hough, Ransac), bolo potrebné pridať aj algoritmus na separáciu viacnásobne sa vyskytujúcich totožných bodov. Nakoľko Ransac, a najmä Hough algoritmus pracujú s hodnotenými priamkami, vytvárali viacnásobne totožné body nesprávne predikcie pre separované priamky.

2.6.2 Post-processing segmentovaných dát

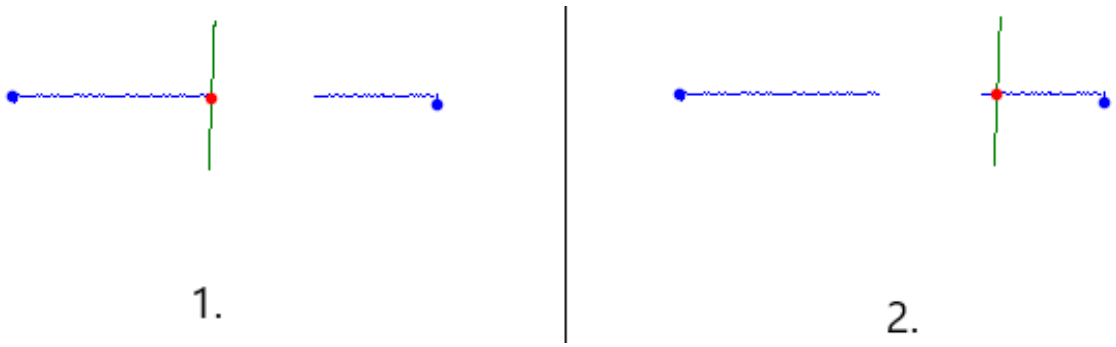
Nakoľko väčšina vytvorených algoritmov hľadá body aproximované nejakou priamkou, spája takýto prístup aj oblasti ktoré síce ležia na jednej priamke, ale syntakticky dané oblasti spolu nedávajú význam, viď obr. 2.21. Bolo pre to nutné vytvoriť algoritmus, ktorý by riešil takýto problém nastávajúci po hlavnej časti segmentácie. V prvej časti nastáva

teda delenie už odsegmentovaných oblastí. To dáva ešte ďalší priestor na ich následné spájanie, nakoľko takýmto prístupom vznikajú nové triedy ktoré môžu ako rozdelené vytvárať spolu väčší význam.



Obr. 2.21: Zobrazený problém segmentácie bodov ležiacich v priamke

Na riešenie takéhoto nedostatku bola použitá vlastná implementácia funkcií. Princíp rozdeľovania boľ poňatí ako iteratívne prejdenie segmentovanej triedy bodov, pričom vyhľadávané boli „diery" pozdĺž priamky definujúcu bodmi. Vstupom do funkcie delenie_poly_koniec(), je pole bodov obsahujúce roztriedené body v rámci jedného indexu triedy. V prípade že pole bodov nie je roztriedené (Hough, Ransac), roztriedi funkcia automaticky body v rámci skupiny pozdĺž najdlhšej priamky p. Tá vznikne spojením dvoch najvzdialenejších bodov triedy. Pomenovaná funkcia si vytvorí priamku q ako kolmicu na priamku p a iteratívne ju vytvára v každom bode poľa. Keďže mi potrebujeme zistiť presný bod, na ktorom nastáva nový koniec priamky, funkcia iteratívne kontroluje, či sa nachádzajú body na oboch stranách kolmej priamky q v určitom okolí. V prípade, že sa v určitom mieste poľa, nachádzajú body len na jednej strane, viď obr. 2.22 (1.), rozdelí funkcia dané pole v meranom bode(bod patrí do starého poľa). Funkcia následne pokračuje v prehľadávaní triedy ďalej obr. 2.22 (2.). Kontrola polohy bodov od priamky q, je riešená pomocou znamienka skalárneho súčinu. Pole ktoré po takomto iteratívnom prejdení vznikne, môžeme vidieť na obr. 2.23. Vývojový diagram celého procesu sa nachádza na obr. 2.24.
Takéto delenie, má lineárnu časovú zložitosť vnútri jednej triedy $O(n)$, kde n je celkový počet bodov.

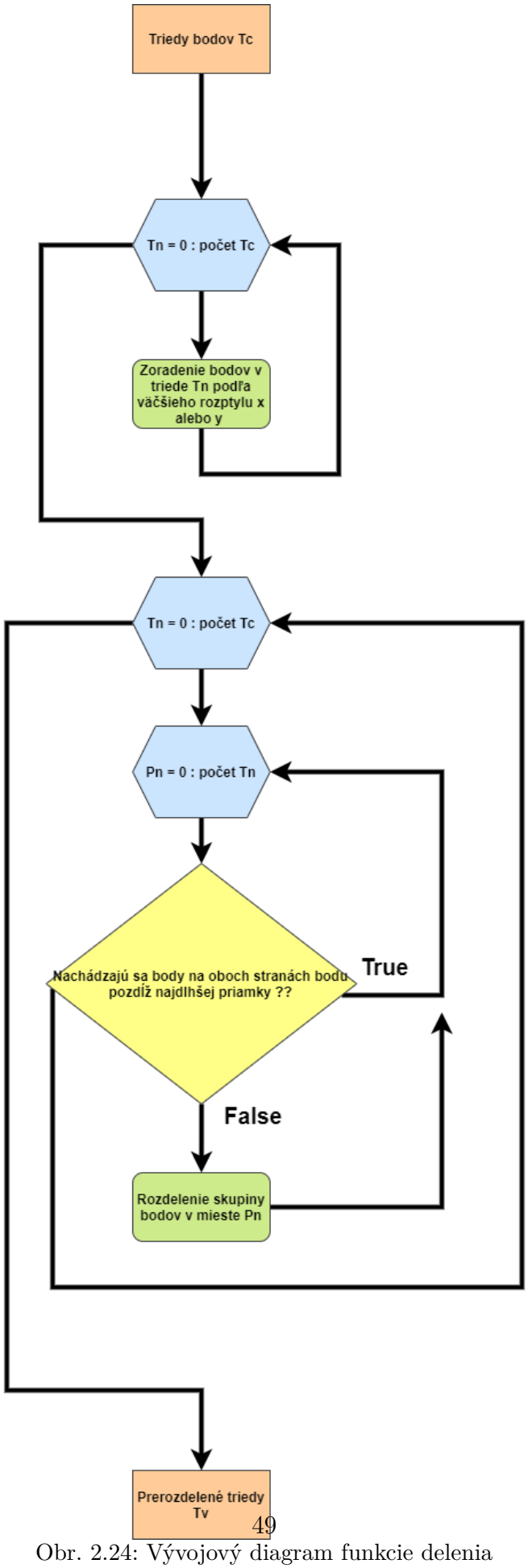


Obr. 2.22: Ukážka princípu delenia množiny pomocou normály

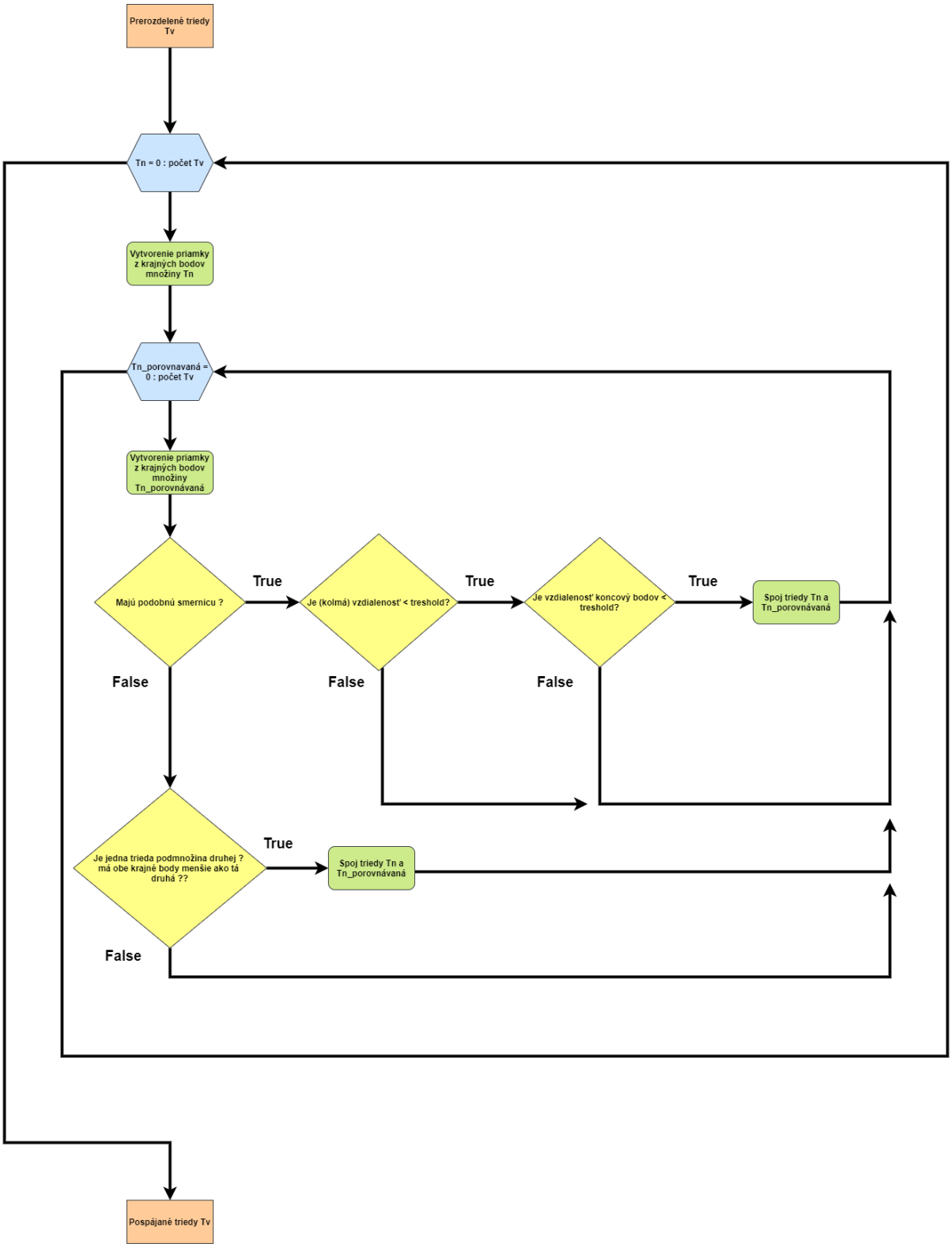
Keďže zvyklo nastávať po takejto úprave presegmentovanie výsledku, bola vytvorená ešte funkcia kolinear() na ich spätné, už správne pospájanie. Pri vytvorení nových menších skupín, môžu existovať segmenty, ktoré vykazujú podobné vlastnosti v dostatočnej blíz-

Obr. 2.23: Rozdelená priamka post-processing

kosti, a sú teda kandidátmi na ich spojenie. V závislosti na danú skutočnosť, bol ešte na predelené triedy vytvorený algoritmus, ktorý kontroluje kolinearitu vstupných odsegmentovaných tried bodov. Algoritmus prejde všetky triedy a porovnáva ich vzájomné vlastnosti s ostatnými. Kritéria spájania sú nastavené na podobnosť smernice daných bodov, vzájomnej kolmej vzdialenosti priamok ako aj, či sa neprekrývajú koncové body jednej priamky s „telom"druhej. V prípade, že porovnávané priamky, nevykazujú podobnú hodnotu smernice, kontroluje sa, či sa jedna priamka nenachádza celou svojou dĺžkou vnútri druhej. V prípade že sa nejaké dve množiny vyhodnotia ako „podobné", určené k spájaniu, pokračuje následné iteratívne hľadanie už s novými, spojenými triedami. Diagram popísaného procesu nájdeme na obr. 2.25.
Takto predstavený algoritmus kolinear(), má časovú zložitosť $O(n^2)$, kde n je počet vstupných tried.



Obr. 2.24: Vývojový diagram funkcie delenia



Obr. 2.25: Vývojový diagram korelačnej funkcie

3 Výsledky

Kapitola výsledky, bude rozdelená do 4 častí. Opisované budú obsahovať názorné ukážky segmentácie jednotlivými navrhnutými algoritmami. Tie sme vzájomne porovnávali na umelo vytvorených dátach z hľadiska segmentácie a následne aj z hľadiska preloženia kriviek. Príklady v každej nasledujúcej sekcií budú mať postupne sa sťažujúci charakter. Každý príklad v nasledujúcej podkapitole sme doplnili aj o výsledky, ktoré vzniknú po aplikácií post-processingového algoritmu uvádzaného v podkapitole 2.6. Na experimentálnych dátach takýto príklad taktiež uvedieme, ďalej budeme ale pokračovať už bez rozlišovania týchto dvoch funkcií. V poslednej podkapitole predstavíme porovnávací graf z hľadiska výpočtového času od počtu prvkov. Ten sme nakoniec doplnili o porovnávaciu tabuľku, kde je vzájomným poradím algoritmov určená ich kvalita v jednotlivých oblastiach.

3.1 Porovnanie algoritmov na vytvorených dátach

Porovnanie na vytvorených dátach bude prebiehať na 4 prezentovaných algoritmov Line tracking, Split and Merge algoritmus, Ransac a Hough algoritmus. SEF algoritmus uvedený v podkapitole 2.1, budeme prezentovať až v podkapitole experimentálnych dát 3.2. Ako sme v opise algoritmu spomenuli, algoritmus SEF nedosahuje komplexnosť ostatných algoritmov. SEF algoritmus by mal značný problém s rozdelením rohov príliš hustého poľa bodov. Algoritmus budeme teda testovať až nasledujúcej podkapitole pri porovnaní na experimentálnych dátach.
Segmentácia bodov je porovnávaná osobitne od porovnávania aproximovaných kriviek. Prvotne porovnávame výsledky na postupne sa zvyšujúcej zložitosti segmentovaných bodov. Budú zobrazené výsledky algoritmov, ako aj ich stav po aplikovaní funkcie post-processingu, opísanej v podkapitole 2.6. Takto predstavené výsledky algoritmov budú medzi sebou porovnávané a analyzované. V ďalšej sekcií sa zameriame na zmenu preloženia aproximovaných kriviek. Pre porovnanie ich, sme vybrali najdlhšiu segmentovaná časť, ktorá bude pomocou metódy najmenších štvorcov preložená prislúchajúcou priamkou. Takto vytvorené priamky pre každý z algoritmov, budú medzi sebou porovnávané so správnou formou segmentácie danej časti.

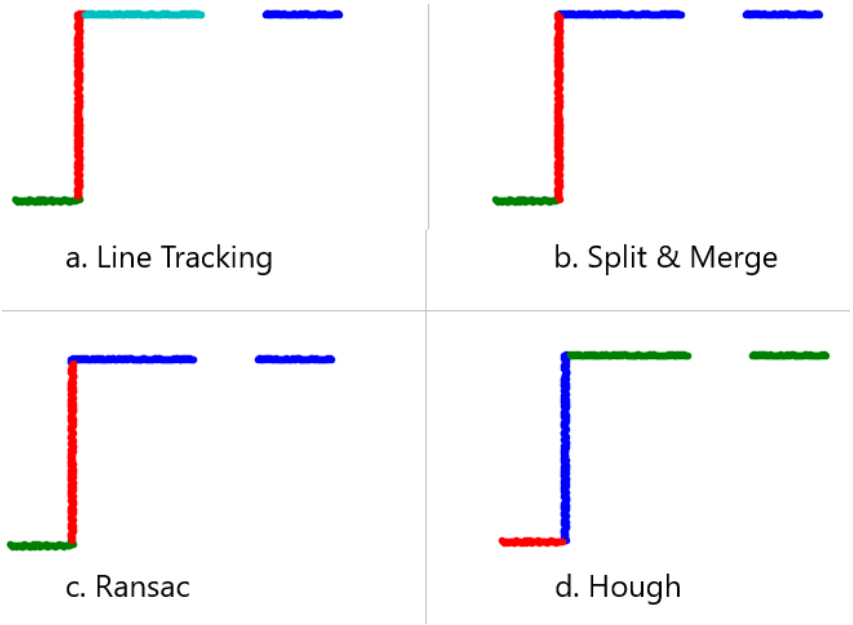
3.1.1 Porovnanie segmentácie bodov

Porovnanie výsledkov segmentácie algoritmov na jednotlivých úrovniach množiny bodov.

Úroveň 1

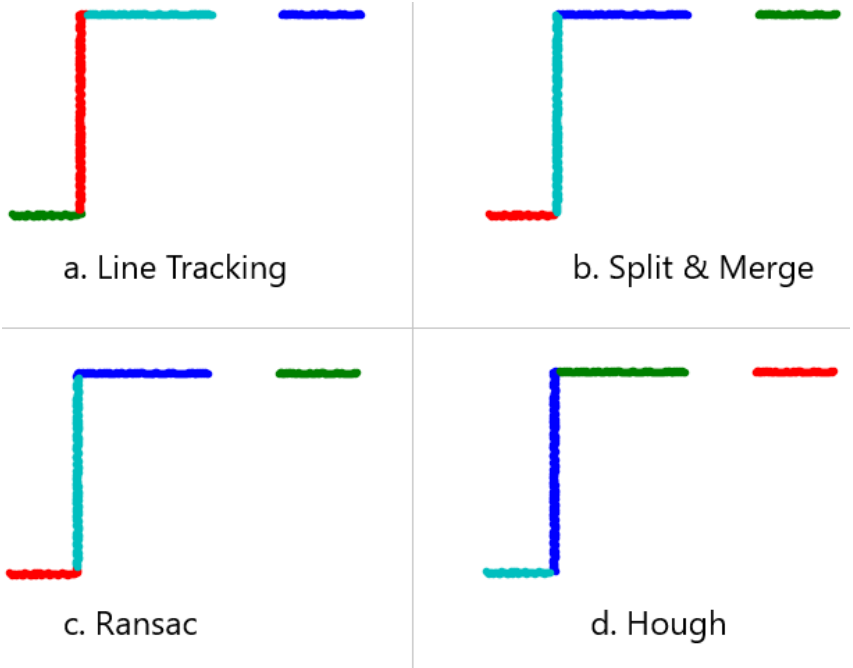
Na prvom porovnaní, nachádzajúcom sa na obr. 3.1, názorne ukazujeme výsledky segmentácie vytvorenými algoritmami. Algoritmy sú zoradené postupne Line tracking, Split and Merge algoritmus, Ransac, Hough algoritmus. Je zreteľné, že výsledky segmentácie pre algoritmy a. až d. sú pre najnižší stupeň náročnosti "1"dostatočné. Algoritmy dokážu

spoľahlivo identifikovať súvisiace body, ako presne nachádzať spojnice dvoch susedných množín / tried. Pri trojici algoritmov b. až d., nastáva problém so spojením hornej horizontálnej časti bodov. Takýto problém sme opisovali v podkapitole 2.6.



Obr. 3.1: Segmentácia vytvorených bodov, stupeň náročnosti 1

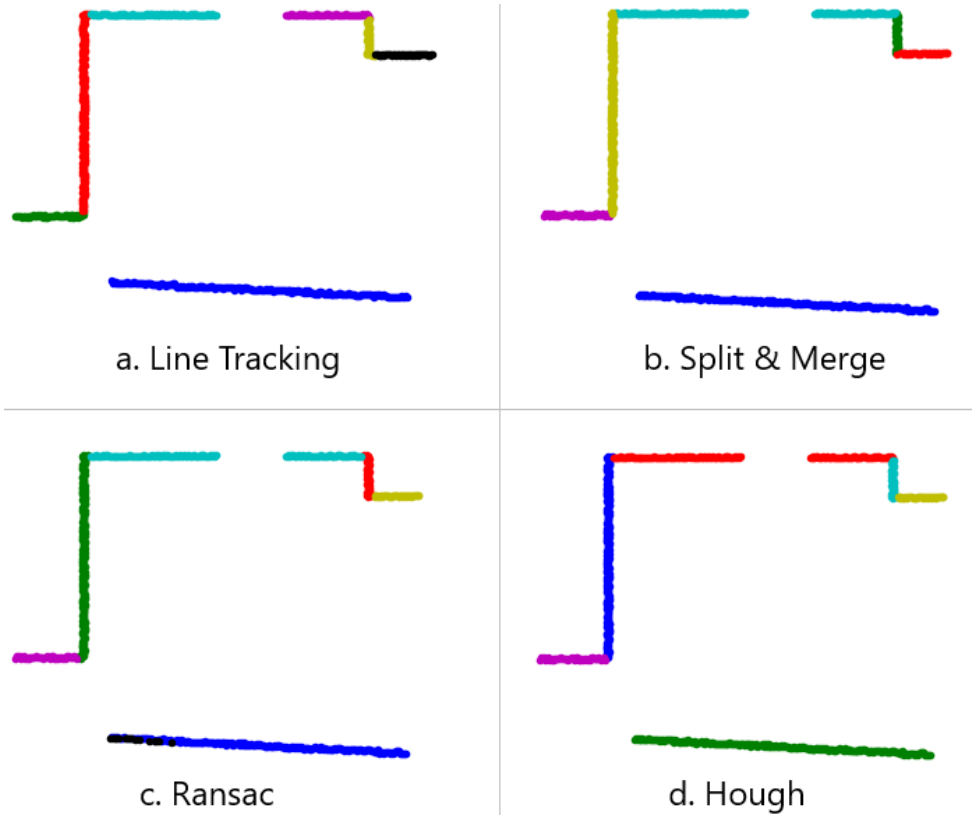
Aplikovaním post-processingových funkcií na výsledky zo segmentácie bodov stupňa náročnosti "1", dostávame výsledky zobrazené na obr. 3.2. Evidentne dokáže tento dodatok, správne pracovať zo vstupnými výsledkami jednotlivých algoritmov b. až d., ako aj správne nachádzať prázdne miesta a tie patrične rozdeliť.



Obr. 3.2: Segmentácia vytvorených bodov, post-processing, stupeň náročnosti 1

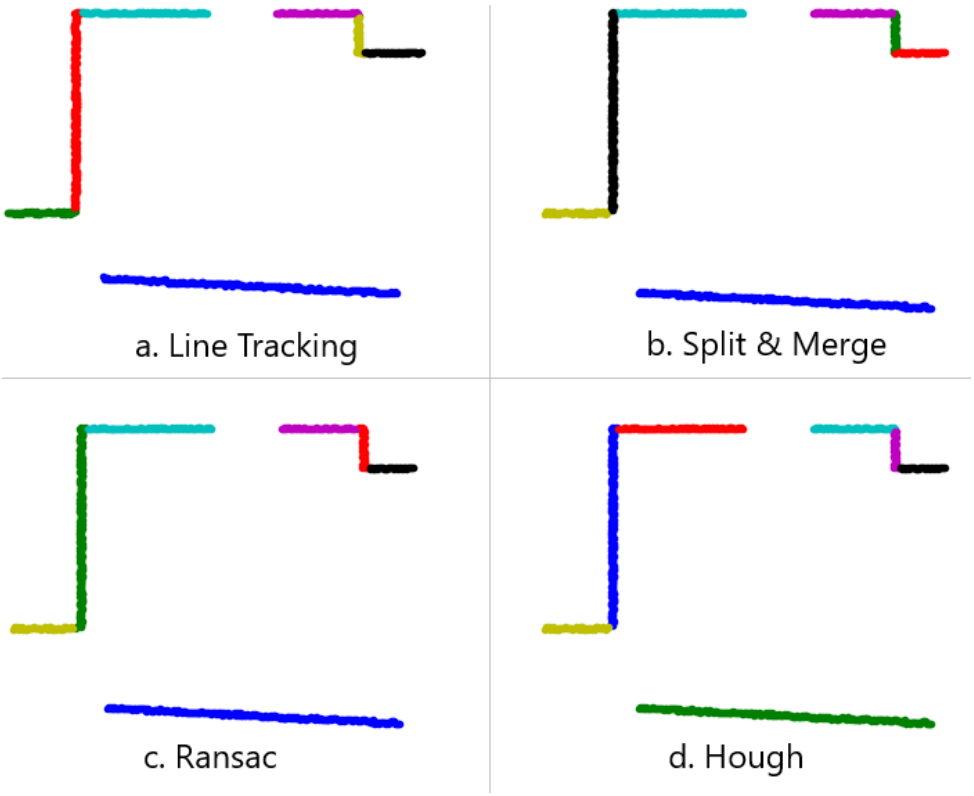
Úroveň 2

Porovnanie segmentácie komplexnejšie rozdelených bodov stupňa náročnosti "2", môžeme vidieť na obr. 3.3. Pole bodov sme vytvorili obohatením predošlého obrázka o ďalšie úsečky. Z prezentovaných výsledkov je očividné, že všetky algoritmy dokážu body takto definujúce útvary čisto segmentovať. Môžeme si všimnúť, že algoritmus Ransac (c.) segmentuje dolnú horizontálnu krivku viac ako by mal. Takéto správanie je pre daný algoritmus typické a nastáva pri nájdení dostatočne veľkej množiny bodov. Na ďalšom obrázku uvidíme, ako si s týmto neduhom poradí post-processing.



Obr. 3.3: Segmentácia vytvorených bodov, stupeň náročnosti 2

Po aplikácií post-processingovej funkcie delenia a spájania, dostávame výsledky zobrazené na obr. 3.4. Podstatné je najmä odstránenie chyby pri algoritme Ransac (c.) v dolnej horizontálnej priamke. Tá už po post-processing nie je rozdelená na dve množiny, čím nastáva správne opravenie segmentácie.

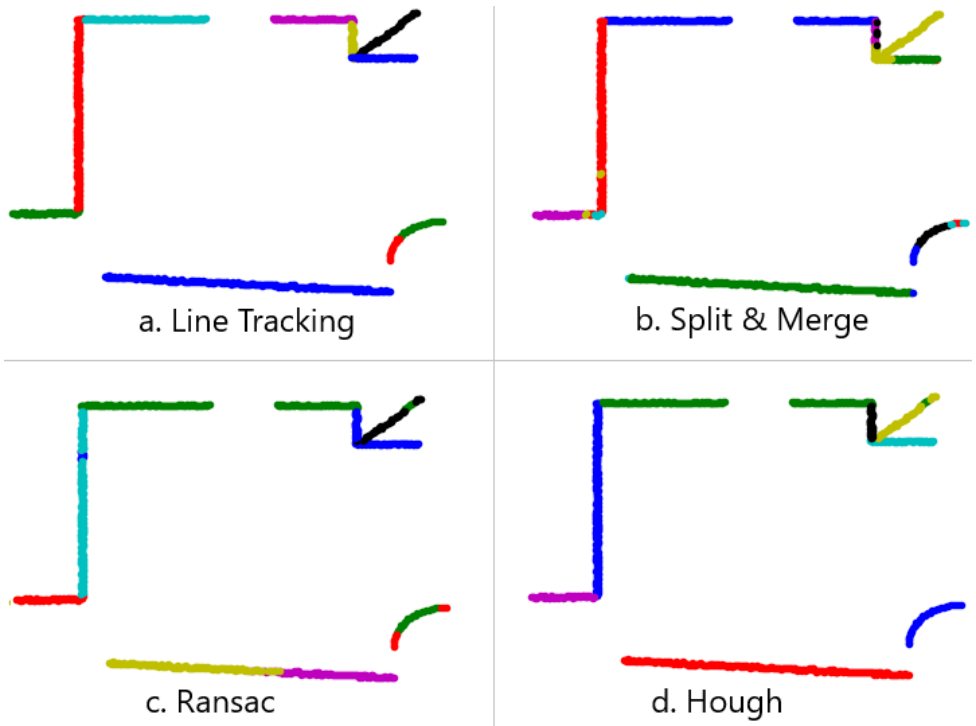


Obr. 3.4: Segmentácia vytvorených bodov, post-processing, stupeň náročnosti 2

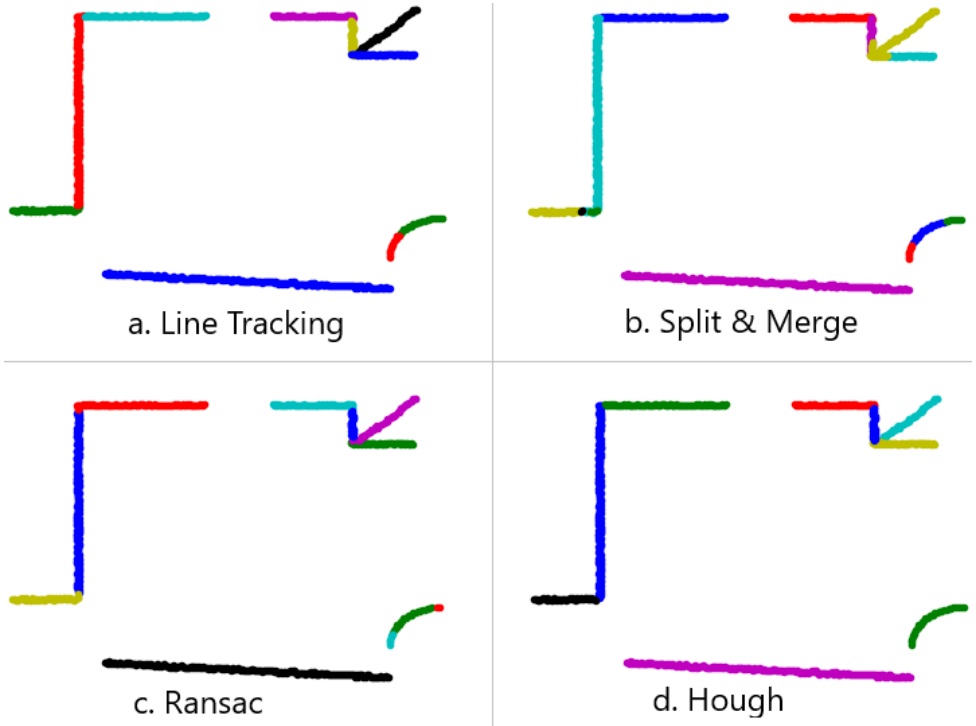
Úroveň 3

Príklad množiny bodov stupňa náročnosti "3", v sebe zahrňuje pridanie časti kruhu ako aj ďalšie rozšírenie počtu priamok potrebných na segmentáciu. Po pridaní ďalších bodov nastáva evidentné sťaženie správnej segmentácie. Ako vidíme na obr. 3.5, každý z algoritmov (a. až d.) obsahuje minimálne chyby v rámci svojich rozdeľovacích výsledkov. Viditeľne správne rozdelenie oblúku nastalo iba v prípade Hough algoritmu (d.). Ten správnou identifikáciou všetkých priamok, označil túto (oblúkovú) množinu dát ako neidentifikovateľnú, a priradil jej jednu segmentačnú triedu. Algoritmus Split and Merge (b.) nesprávne segmentuje vrchnú časť bodov, taktiež chybuje v dolnej časti najdlhšej vertikálnej priamky. Ransac (c.) rozdeľuje najmä nesprávne spodnú horizontálnu priamku na dve časti. Taktiež nesprávne zaradí malé množstvo bodov v najdlhšej horizontálnej priamke do inej skupiny. Ako bolo spomenuté, tak správne označenie polkruh nastalo iba v jednom zo štyroch prípadov (d.). Keďže primárnou funkciou tejto diplomovej práce bola segmentácie a nachádzanie úsečiek, považujeme výsledky segmentácie polkruhu, nachádzajúce sa v príkladoch (a. až c.) za postačujúce.

Aplikácia postprocesových funkcií na analyzovaný obr. 3.5, výrazne zlepšuje výslednú segmentáciu jednotlivých algoritmov, viď obr. 3.6. Znateľne nastáva zlepšenie výsledku v prípade Ransac algoritmu (c.), kedy postprocesing dokáže jednak opraviť pôvodne rozdelené množiny v dolnej oblasti horizontálnej priamky. Taktiež nastáva aj správne priradenie avizovanej malej množiny bodov vyskytujúcich sa v oblasti najdlhšej vertikálnej priamky u zmieneného algoritmu (c.). Takáto korekcia malej množiny bodov nastáva aj v prípade Hough algoritmu (d.), kde sa tento problém nachádza v oblasti pravého horného rohu v obr. 3.5 (d.). Napriek výraznému zlepšeniu pri algoritme Split and Merge (b.), ostáva viditeľná zvýšená segmentácia v oblasti najdlhšej vertikálnej priamky.



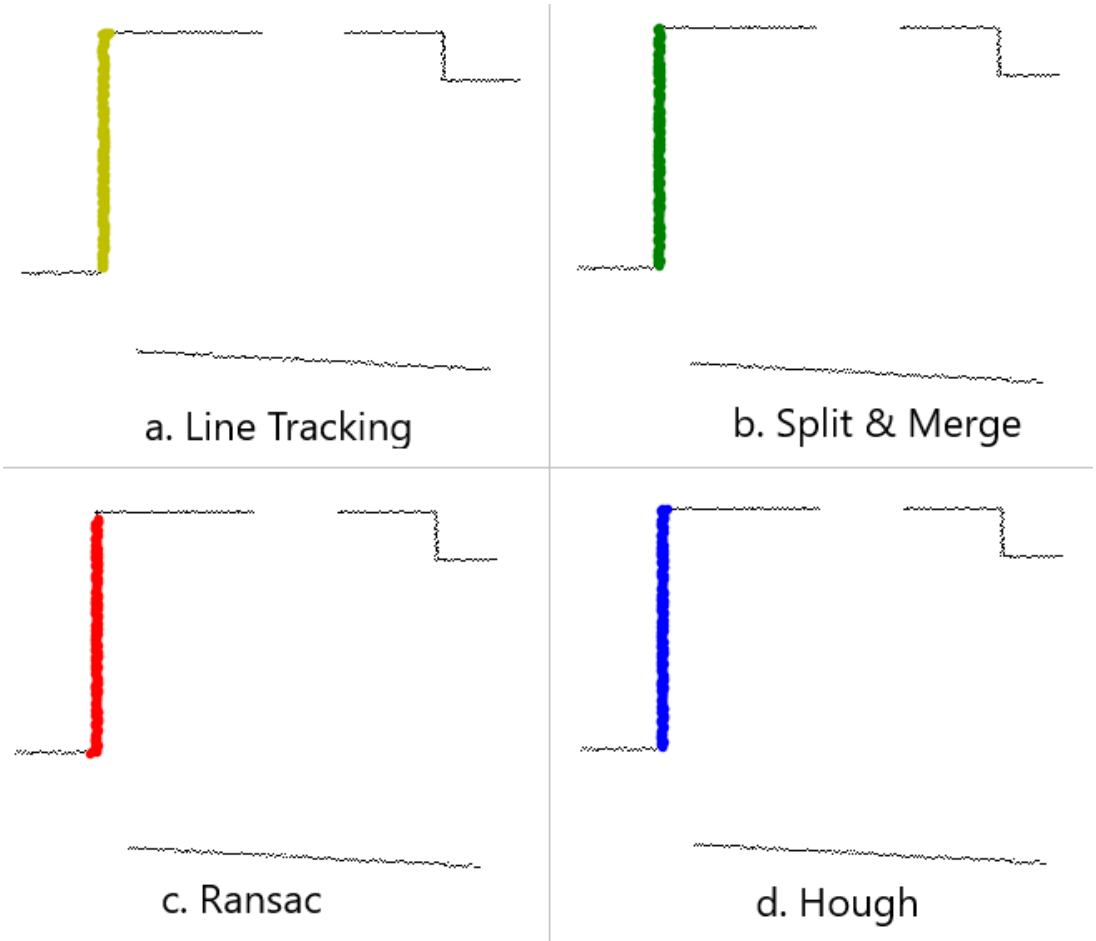
Obr. 3.5: Segmentácia vytvorených bodov stupňa náročnosti 3



Obr. 3.6: Segmentácia vytvorených bodov, post-processing, stupeň náročnosti 3

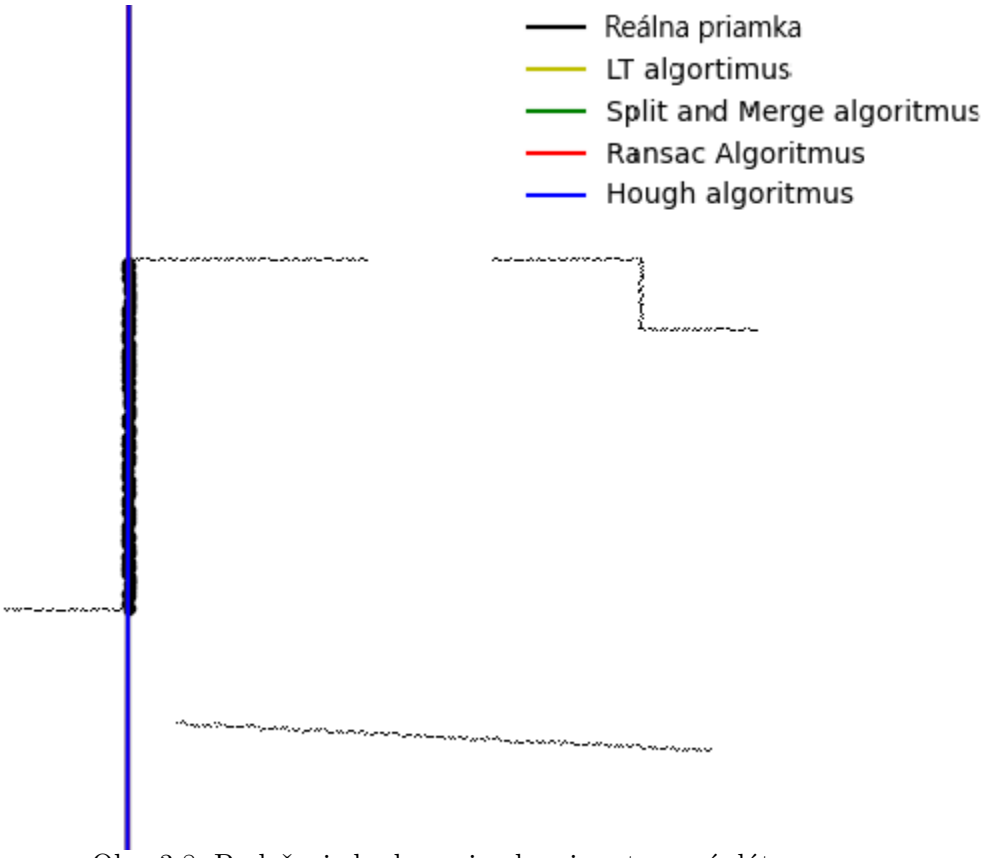
3.1.2 Porovnanie preložených kriviek

Porovnanie preložených kriviek, uskutočňujeme na odsegmentovanej časti, nami vytvorenej množine bodov stupňa náročnosti "2". Pre názornú ukážku budeme porovnávať najväčšiu rozdelenú množinu, s ktorou zároveň môžu mať niektoré algoritmy menší problém, nakoľko je od nich vyžadovaná lokalizácia kolmých úsečiek ako aj správne predelenie množín v danom mieste. Na obr. 3.7, vidieť výslednú segmentáciu jednotlivých algoritmov pre túto časť. Je zjavné, že algoritmy zahrnuli do tejto oblasti iné skupiny bodov. Očividne je koniec aj začiatok rozdelenej množiny, pre jednotlivé postupy(a. až d.) rozdelený inak. Algoritmy Line tracking a Hough (a. až d.) ukončili množinu neskôr ako „by mali". Opačný efekt, teda príliš skoré začatie rozdelenej množiny nastalo pri algoritme Ransac (c.).



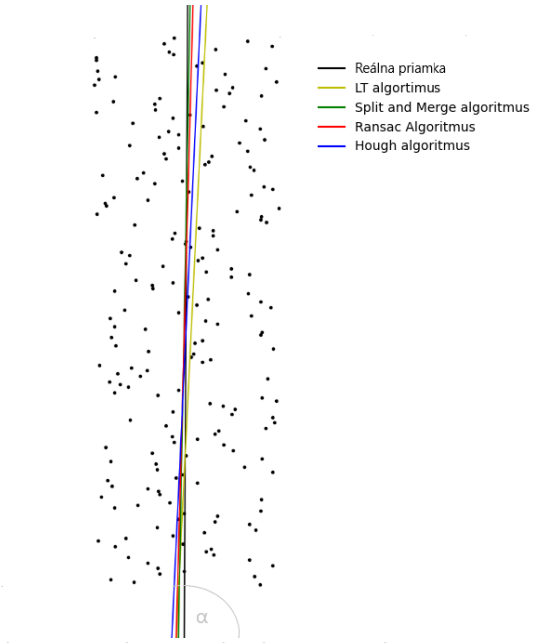
Obr. 3.7: Segmentácia vybranej množiny

Takto rozdelenými výslednými bodmi, bola aproximovaná priamka metódou najmenších štvorcov. Výsledné priamky prekladajúce danú zobrazenú množinu čiernych bodov môžeme vidieť na obr. 3.8. Z prvotného pohľadu, môžeme usúdiť, že z globálneho hľadiska sa priamky pre daný prípad prekrývajú. Zvýraznené čierne body, avizujú správne oddelenie množiny. Priamku nimi aproximovanú (čiernu), budeme považovať za referenčnú.



Obr. 3.8: Preloženie bodov priamkami, vytvorené dáta

Priblíženie obr. 3.8 na skúmanu časť, vidíme na obr. 3.9. Jednotlivé priamky, vzniknuté preložením bodov, pre každú z metód sú farebne označené podľa obr. 3.7. Z obr. 3.9, dokážeme usúdiť dvojicu skutočností. Priamky sa líšia iba v zanedbateľnom priblížení. Pozoruhodné je natočenie všetkých aproximovaných priamok oproti „správne vykreslenej priamke". Tento jav pripisujeme skutočnosti, že na začiatku a konci segmentácie sa nachádzajú ďalšie body. Pridaním týchto bodov do segmentačnej množiny z jedného či druhého konca, nastáva jemné natočenie priamky práve týmto smerom. V tabuľke 3.1, môžeme vidieť číselné vyjadrenie natočenia jednotlivých priamok. Jedná sa o hodnotu uhlu α, vytvorenú medzi priamkou a horizontálou, načrtnutú v dolnej časti obrázka.



Obr. 3.9: Priblíženie preloženia bodov priamkami, vytvorené dáta

Algoritmus	LT	S&M	Ransac	Hough	Reálna
Uhol [°]	89.91	89.97	89.96	89.92	90
Rozdiel uhlu [°]	-0.09	-0.03	-0.04	-0.08	-

Tabuľka 3.1: Uhlové vyjadrenie natočenia priamok, vytvorené dáta

3.1.3 Zhodnotenie k vytvoreným dátam

V aktuálnej podkapitole, sme testovali nami navrhnuté algoritmy na vytvorených dátach. Algoritmy sme postupne aplikovali na množiny bodov rôznych stupňoch náročností. Z preukázaných grafických výsledkov v jednotlivých stupňoch náročnosti (1 až 3), je evidentné, že navrhnuté algoritmy dokážu dané vytvorené množiny bodov rozdeľovať do správnych množín. Pridanie polkruhu v poslednej, "3"stupni náročnosti, viď obr. 3.6, a jeho následné označenie jednotlivými algoritmami sa úspešne podarilo iba v prípade Hough algoritmu (d.). Výsledky najťažšej množiny bodov stupňa náročnosti "3", už v sebe niesli niekoľko drobných chýb vo výsledkoch algoritmov. Je ale podstatné poznamenať, že daný útvar jemne prevyšuje potreby segmentácie dát z Lidar zariadenia, nakoľko, sa tam nachádzajú časti, ktoré Lidar zariadenie nie je schopné nasnímať, ako napr. objekty stojace za sebou(úsečka v pravom hornom rohu).
Ako sme patrične ukázali v úrovni analyzovanej množiny bodov, tak pridanie postprocessingových funkcií badateľne zlepšilo výsledky daných algoritmov. Daný doplnok vo dokázal korektne opravovať chyby, na ktoré bolo poukazované v opise. Podstatné je všimnúť si, že opravy nastávali prevažne na správnych miestach a teda nedochádzalo k chybnej detekcií už správne rozdelených množín bodov.
V prípade porovnania preloženia kriviek, som deklaroval, že pri nami vytvorených dátach, nezáležalo na type použitého algoritmu. Pretože nás pri preložení bodov zaujímajú najpravdepodobnejšie nachádzajúce sa krivky, bola preložená najdlhšia krivka z vytvoreného modelu dát. Uhly natočenia preložených kriviek vytvorené danými algoritmami sa líšili len zanedbateľne.

3.2 Porovnanie algoritmov na nameraných dátach

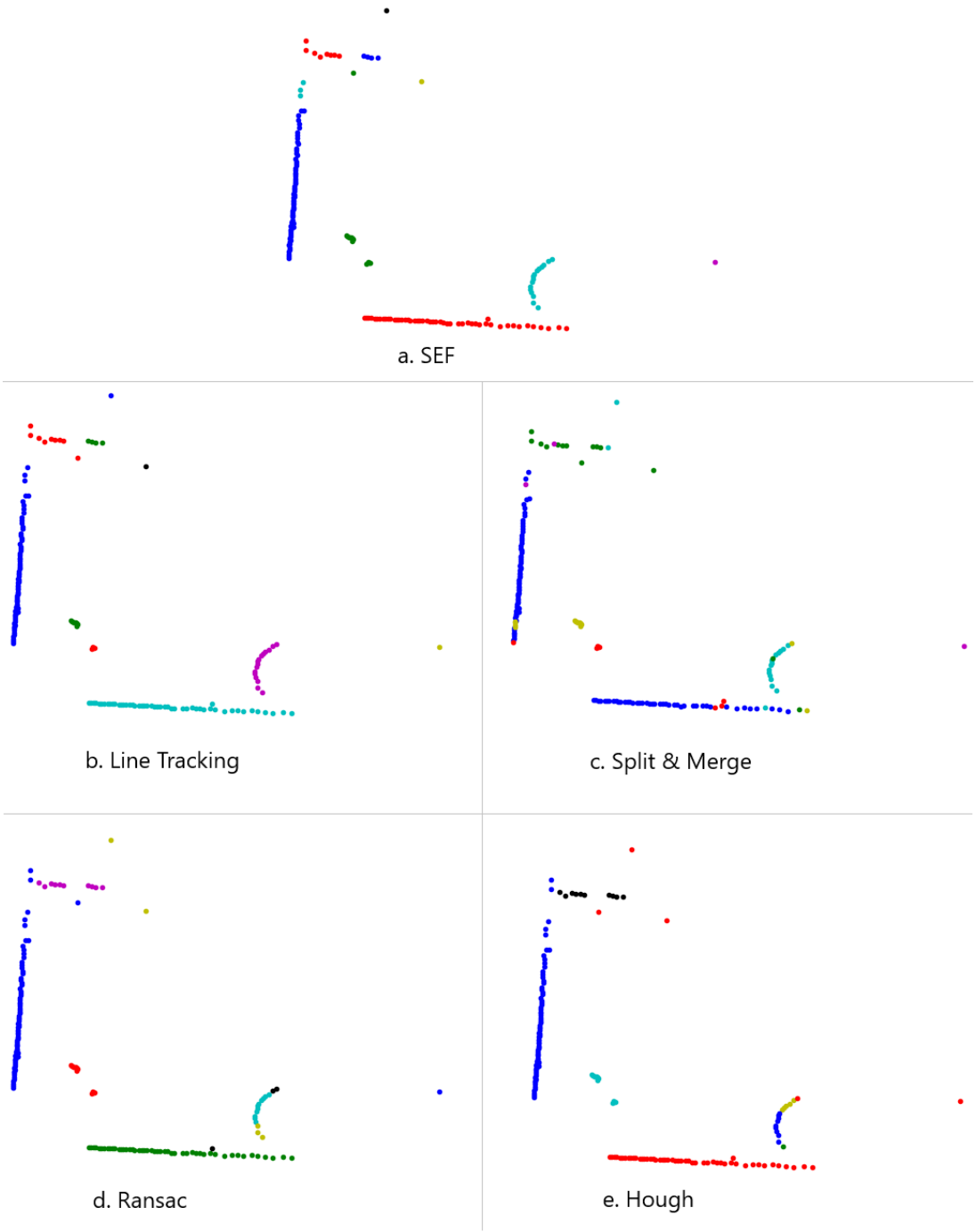
Aktuálnu podkapitolu sme štruktúrovali podobne ako predošlú 3.1. Nami navrhnuté algoritmy v nej testujeme na experimentálne nameraných dátach z Lidar zariadenia. Celkovo bolo vytvorených 107 meraní. V nasledujúcej sekcií porovnáme výsledky algoritmov na dvoch rozličných meraniach. Taktiež v nej prezentujeme porovnanie výsledkov pred a po použití funkcií post-processingu podkapitoly 2.6. V danej podkapitole budeme pracovať aj s viacerými spojenými meraniami, nakoľko aj takáto aplikácia môže byť v rámci ich použitia žiadúca. V závere aktuálnej podkapitoly porovnáme preloženie kriviek, kde ako porovnávacie dáta zvolíme množinu vytvorenú z viacerých meraní, pre vytvorenie väčšej diverzity uhlov priamok. Množina experimentálnych bude rozdielna od použitej v analýze segmentácie.
V nasledujúcich podkapitolách už zahrnieme do porovnávania aj algoritmus SEF popísaný v podkapitole 2.1. Zastúpenie algoritmov nachádzajúcich sa na prezentovaných obrázkoch v tejto podkapitole bude v poradí SEF, Line Tracking, Split and Merge, Ransac a Hough algoritmus (a. až e.).

3.2.1 Porovnanie segmentácie

Porovnanie algoritmov z hľadiska segmentácie uskutočníme najskôr na dvojici rozličných meraní. V prvom príklade ukážeme rozdiel výsledkov pred a po použití funkcií post-processingu. Nakoľko tento vplyv bol podrobne popisovaný v predošlej podkapitole 3.2, budeme už ďalej od daného príkladu ukazovať výsledky vzniknuté po segmentácií aj s použitím post-processingu. Na konci porovnania segmentácie ukážeme aj príklad rozdelenia množiny bodov, ktorý vznikol spojením viacerých meraní.

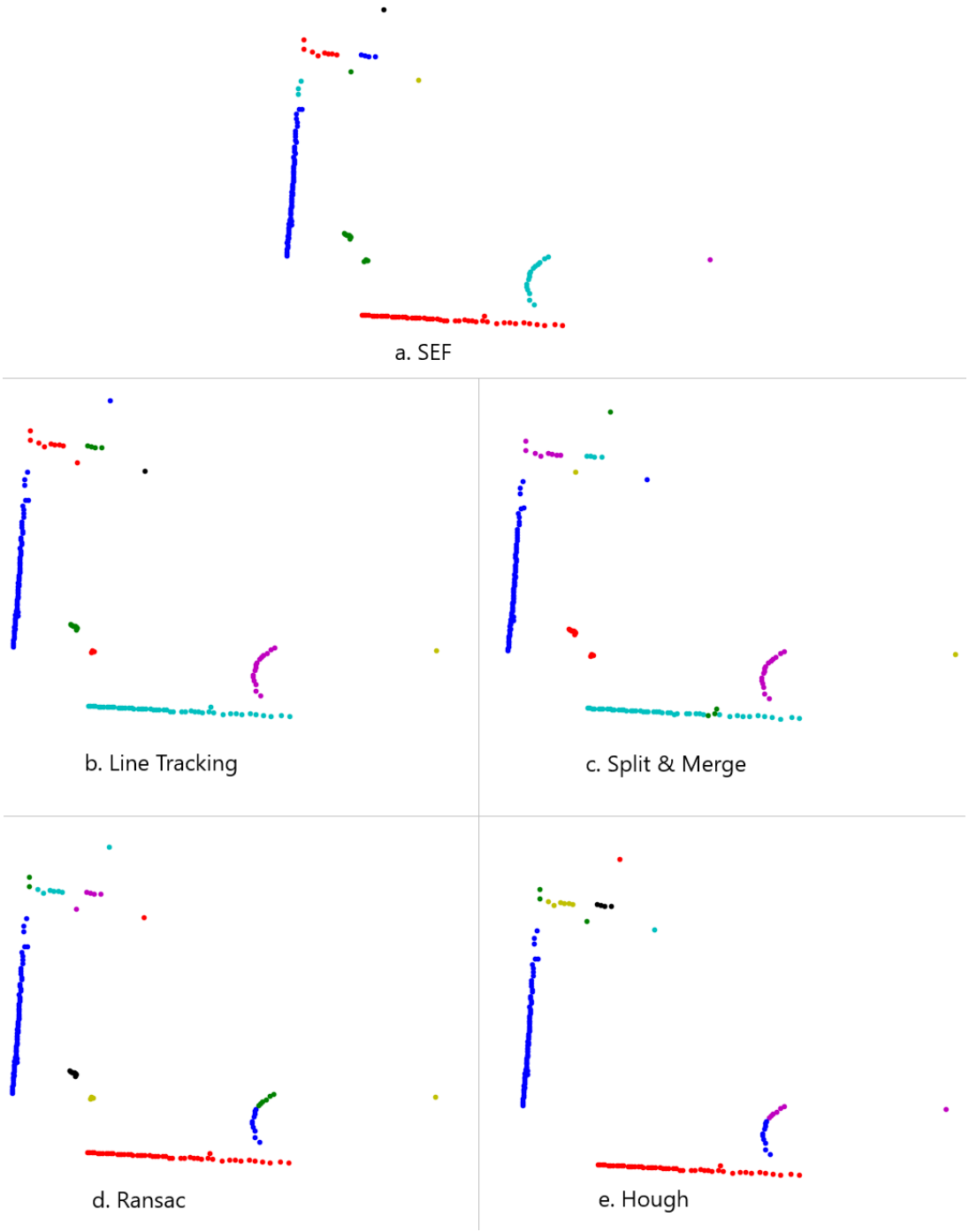
Meranie číslo 1

Segmentáciu bodov uskutočnenú na prvom meraní môžeme vidieť na obr. 3.10. Na obrázku sú jasne vidieť rozdiely jednotlivých algoritmov. Najhoršie v danom porovnaní skončila metóda Split and Merge (c.), v ktorej nastala nadmerná segmentácia rôznych častí bodov. Veľmi dobre môžeme označiť výsledky algoritmov Ransac (d.) a Hough (e.), ktoré dokázali veľmi presne odčleniť mimo stojace body pozdĺž najdlhších kriviek. V prípade SEF algoritmu (a.) dochádza zjavne k presegmentovaniu v ľavej hornej časti.



Obr. 3.10: Segmentácia experimentálnych bodov, meranie č. 1

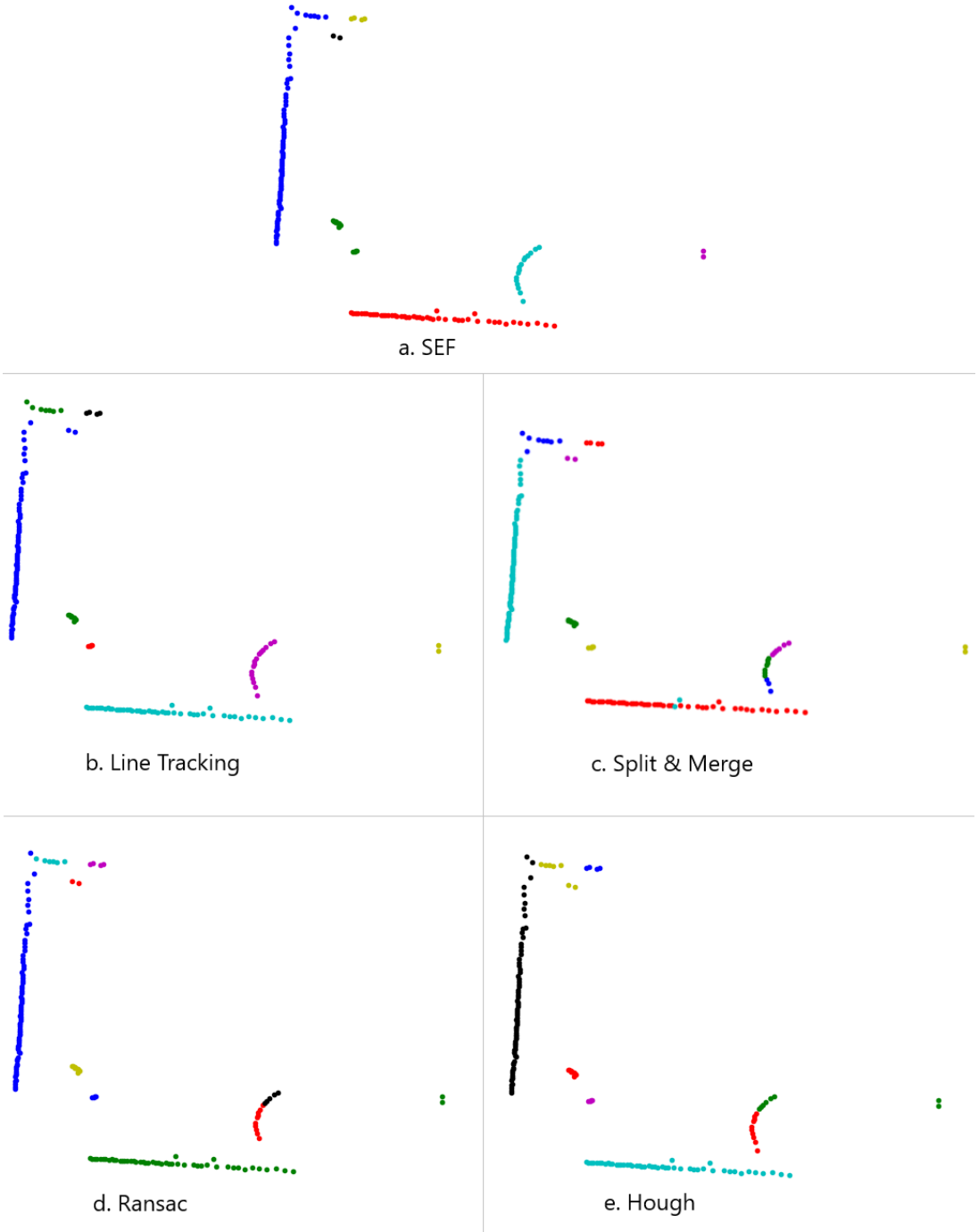
Výsledky algoritmov na obr.3.10, po opravení niektorých oblastí pomocou funkcie post-processingu, môžeme vidieť na obr. 3.11. Najvýraznejšie zlepšenie sa udialo pri algoritme Split and Merge (c.), v ktorom nastala oprava a následné správne spojenie väčšiny zle označených bodov. Post-processing taktiež zvládol správne rozdeliť dva osamotené body v ľavom dolnom rohu, pre všetky robustnejšie algoritmy na ktoré bol aplikovaný (c. až e.). Označenie oblúka bolo úspešne u dvojici algoritmov SEF (a.) a Line tracking (b.). Ako už ale bolo vyššie uvedené, nadmerná segmentácia daného útvaru je v rámci našej práce vyhovujúca.



Obr. 3.11: Segmentácia experimentálnych bodov, post-processing, meranie č. 1

Meranie číslo 2

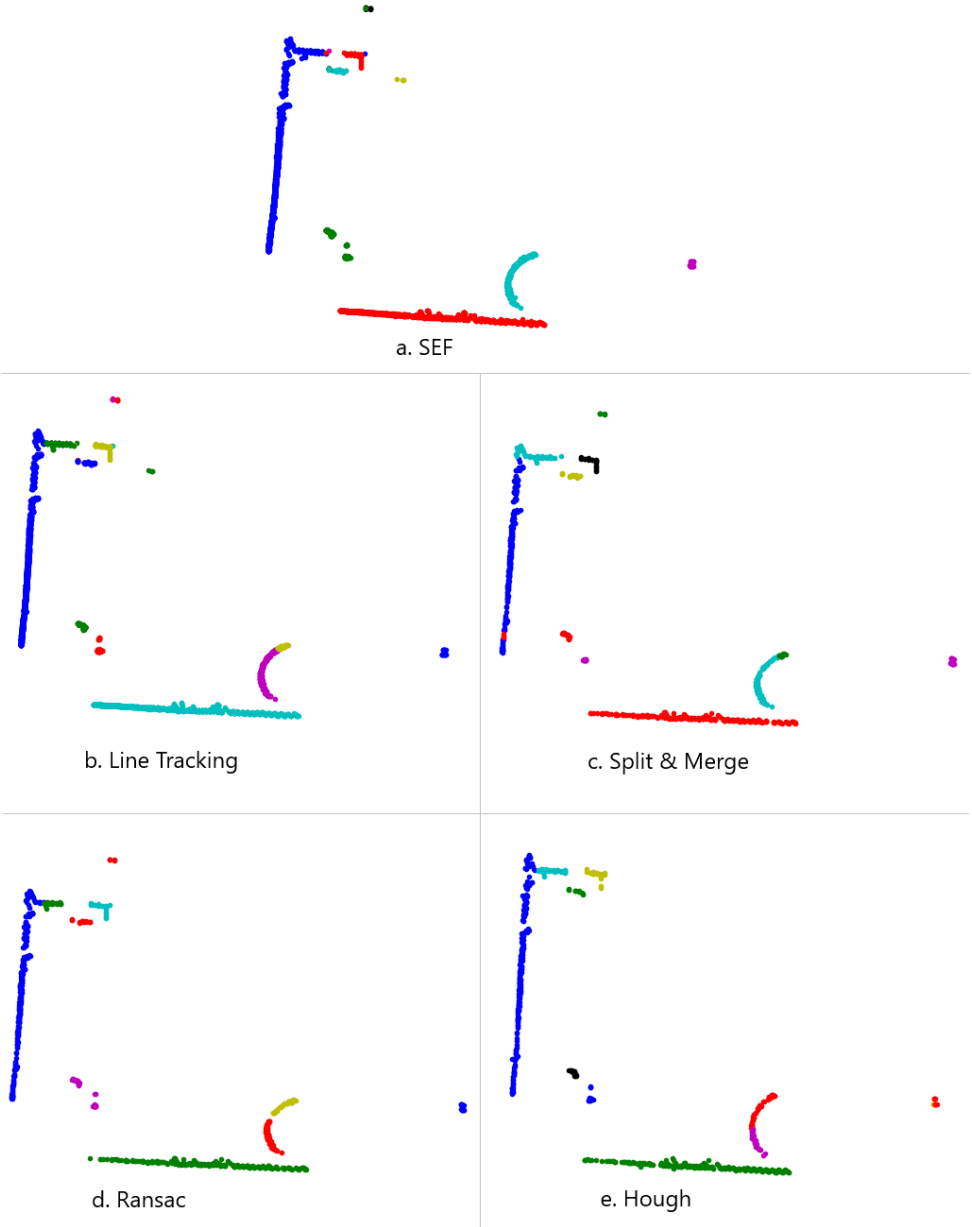
V ďalšom porovnaní, pracujeme s ďalším z experimentov. Na obr. 3.12 sa nachádzajú výsledky všetkých algoritmov (a. až e.) už aj po aplikácií post-processingovej funkcie. Mimo algoritmu SEF (a.), ktorý zjavne nedokázal presne určiť detekovaný roh, vykazujú všetky algoritmy dostatočne kvalitné výsledky pre daný typ merania.



Obr. 3.12: Segmentácia experimentálnych bodov, post-processing, meranie č. 2

Porovnanie viacerých meraní naraz

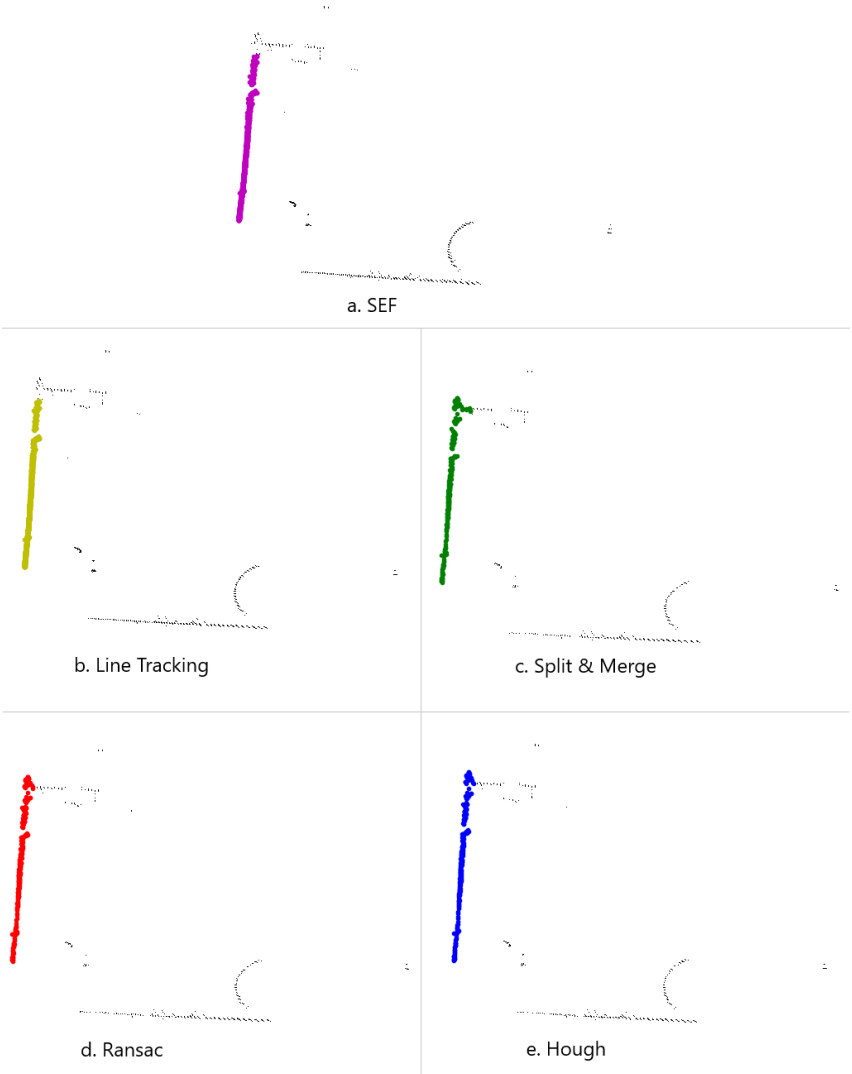
Algoritmy budeme testovať aj na 20, po sebe opakujúcich sa meraniach. Merania vznikli pri pohybe Lidaru priestorom, a preto sa môže pozícia detekovaných bodov javiť ako posunutá. Takýmto postupom sme vytvorili hustejšie pole dát. Tým sa aj podstatne zvýšil celkový počet vstupných dát. Výsledky segmentácie takto vytvorených dát, vidíme na obr. 3.13. Všetky algoritmy evidentne dokázali nájsť najpravdepodobnejšie (najdlhšie) krivky, ako aj správne rozdeliť väčšinu ostatných skupín bodov. Napriek nepresnej polohe rohu v ľavej hornej časti pri algoritme SEF (a.), ako aj nadmernej segmentácií Split and Merge algoritmu (c.), môžeme tvrdiť, že výsledky sú uspokojivé. Výsledky segmentácie algoritmov Line Tracking (b.), Ransac (d.) a Hough (e.) môžeme v tomto prípade označiť za výborné.



Obr. 3.13: Segmentácia spojených experimentálnych meraní

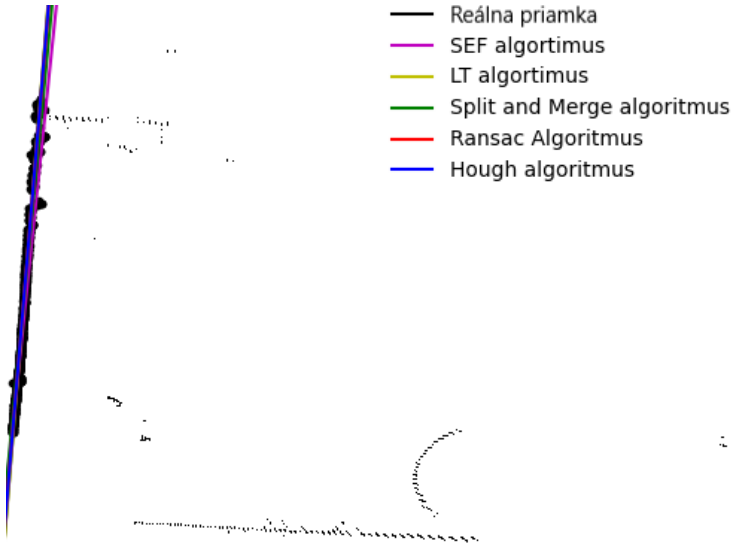
3.2.2 Porovnanie preložených kriviek

Dané algoritmy budeme pre preloženie kriviek testovať aj na väčšom počte dát. Ako sme už vyššie spomenuli, je to z dôvodu vytvorenia väčšej rozmanitosti dátových bodov ako aj rozličnejších výsledkov preložených kriviek. Na vytvorenie bodov použijeme taktiež 20 meraní, nie však totožných s predchádzajúcim prípadom porovnávania segmentácie. Na obr. 3.14 vidíme rozdelenú časť bodov pre každú z predstavených algoritmov. Rozdelené časti sú zafarbené patrične, podľa príslušnosti k algoritmu. Ako je z obrázka zrejmé, každý algoritmus rozdelil najdlhšiu časť bodov rozdielne. Z toho môžeme usúdiť, že aj preloženie daných kriviek nebude identické.



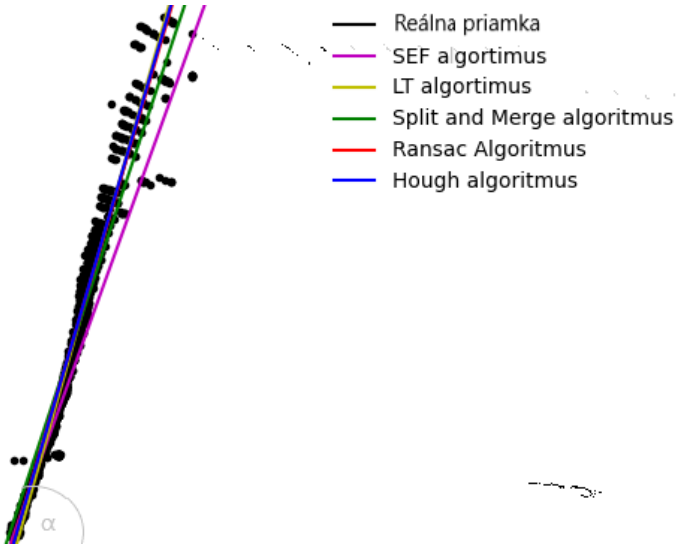
Obr. 3.14: Segmentácia bodov vybranej množiny

Na obr. 3.15, vidíme jemné rozdiely preložených priamok. Z globálneho hľadiska sa rozdiely medzi priamkami zdajú byť nepodstatné. Čiernou farbou zvýraznené body predstavujú reálne rozdelenú triedu bodov. Jednotlivé farebne odlíšené priamky, predstavujú preložené krivky pre danú rozdelenú triedu podľa predchádzajúceho obr. 3.14 pre prípade (a. až e.).



Obr. 3.15: Preloženie bodov priamkami, experimentálne dáta

Zväčšenie riešenej časti obr. 3.15, vidíme na obr. 3.16. Z obrázka sú evidentné, najlepšie výsledky pre prípady Line tracking (b.), Ransac (d.) a Hough (e.) algoritmu. Algoritmy SEF a Split and Merge vykazujú jemnú odchýlku od správnosti preloženia. Z číselného porovnania údajov v tabuľke 3.2 je zrejmé, že hodnota zmeny natočenia priamok nie je dramatická. Najväčšia uhlová zmena nastala pri SEF (a.) algoritme, pričom uhlový rozdiel je do 1° (stupňa). Hodnota uhlu α, je načrtnutá v dolnej časti obr. 3.16 a jedná sa o uhol medzi priamkou a horizontálnou čiarou.



Obr. 3.16: Priblíženie preloženia bodov priamkami, experimentálne dáta

Algoritmus	SEF	LT	S&M	Ransac	Hough	Reálna
Uhol [°]	85.57	86.52	85.93	86.36	86.41	86.49
Rozdiel uhlu [°]	-0.92	+0.03	-0.56	-0.13	-0.08	-

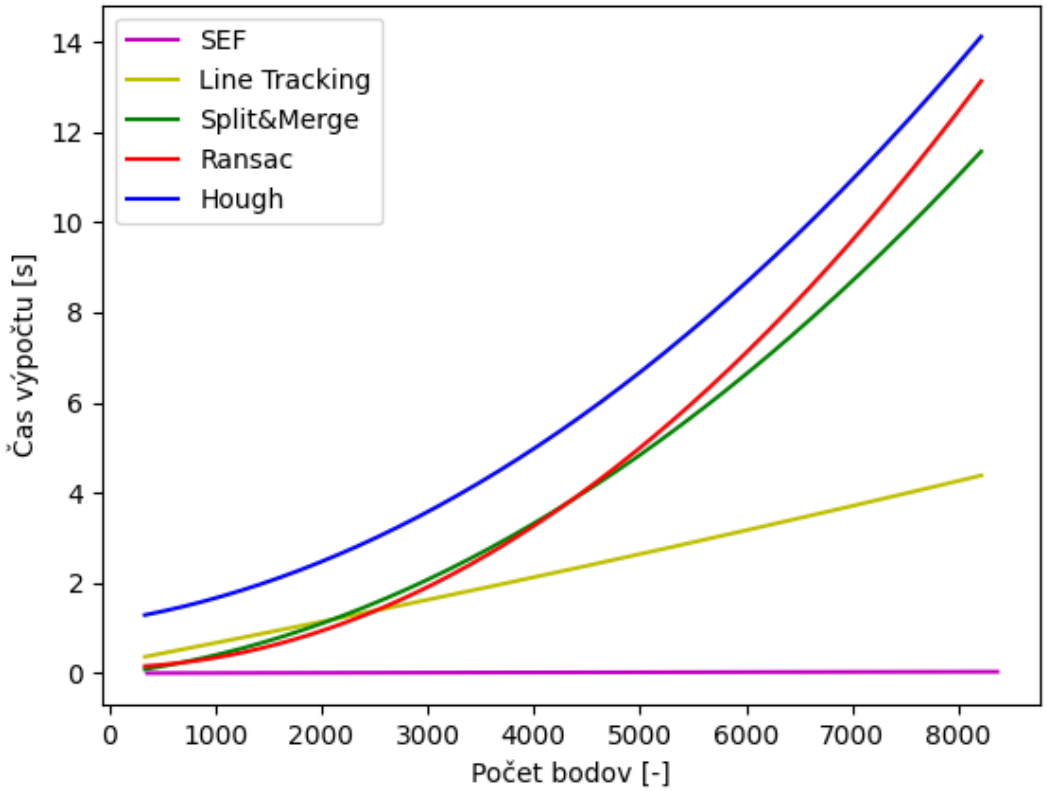
Tabuľka 3.2: Uhlové vyjadrenie natočenia priamok, experimentálne dáta

3.2.3 Zhodnotenie k experimentálnym dátam

V popisovanej podkapitole sme analyzovali výsledky algoritmov na experimentálne nameraných dátach z Lidaru, čím sme overili funkčnosť algoritmov aj na takto získaných bodov. Výsledky algoritmov boli analyzované na dvojici rozličných meraniach. Pri analýze prvého merania, sme ukázali aj vplyv post-processingu na výsledných dátach. Od merania č. 2, sme prezentovali výsledky zahrnuté aj s použitím funkcií post-processingu. Algoritmy sme testovali aj na veľkej množine dát, kde dokázali svoju schopnosť rozdeľovať aj husto nameranú množinu bodov.
Preloženie kriviek bolo ako v predošlej podkapitole analyzované na najpravdepodobnejšej (najdlhšej) rozdelenej množine. Výsledky z daného porovnania ukázali rozdielnosť preloženia priamok odsegmentovanými bodmi v lokálnom merítku. Z globálneho hľadiska boli rozdiely medzi pozíciou a natočeniami jednotlivých priamok minimálne.

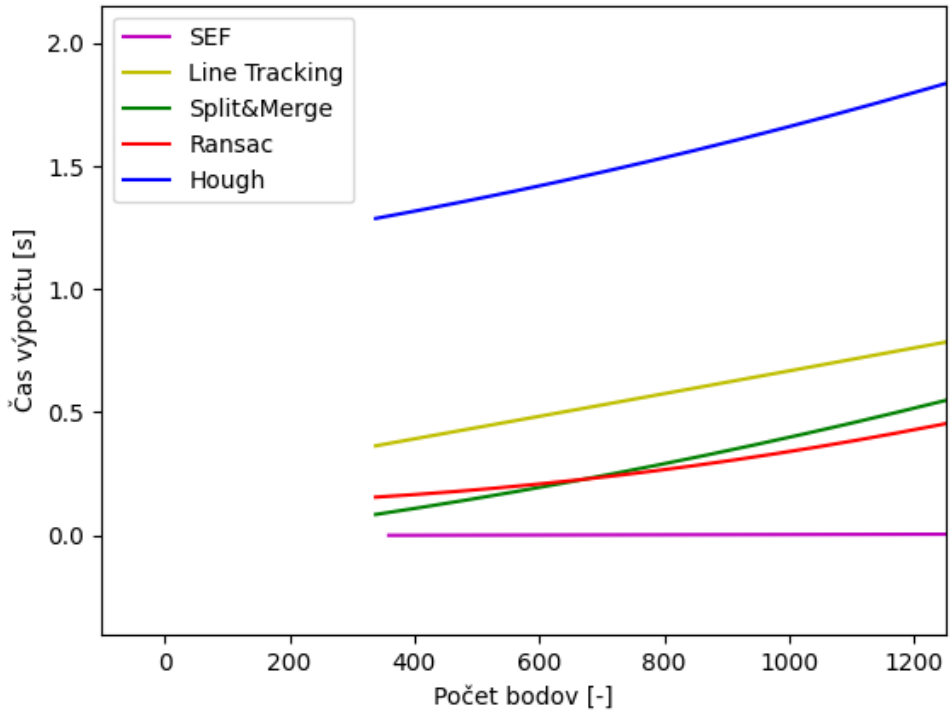
3.3 Porovnanie časovej závislosti

Výpočtový čas, hrá veľmi dôležitú úlohu pri segmentácií point cloudu. Segmentačné algoritmy sa používajú najmä na rozdeľovanie bodov v reálnom čase, kedy je počet bodov síce menší, ale je potrebná dostatočne vysoká výpočtová rýchlosť. Taktiež sa ale môžu používať na selekciu veľkého množstva bodov až po nameraní. Kvôli možnému použitiu navrhovaných algoritmov v oboch zmienených prípadoch, boli všetky algoritmy testované na rozličných veľkostiach bodov point cloudu. Pri časovej analýze je nevyhnutná teoretická časová zložitosť algoritmov, popísaná v teoretickej časti pre každý z algoritmov v kapitole 2. Pre prax je ale často viac potrebná experimentálne nameraná závislosť. Na obrázku číslo 3.17, vidíme časovú závislosť jednotlivých algoritmov od celkového počtu bodov point cloudu. Z časovej závislosti je evidentné, že rýchlosť segmentácie SEF algoritmu, je bezkonkurenčne najvyššia. Ostatné algoritmy sa pohybujú v podobných medziach, pričom ich vzájomná rýchlosť sa s pribúdajúcim počtom prvkov point cloudu mení. Okrem algoritmu SEF, vyčnieva taktiež zo skupiny algoritmus Line tracking. Algoritmy Split and Merge, Ransac a Hough vykazujú podobnú časovú závislosť. Deklarovaná konštantná závislosť Ransac algoritmu nie je dosiahnutá, nakoľko bola funkcia spustená viacnásobne pri znižovaní kritérií pre potreby správnej segmentácie.
Proces výpočtov bol vykonaný na počítači s operačným systémom Windows 10, procesorom Inter(R) Core(TM) i7-6700HQ (2.6 GHz), veľkosťou pamäte RAM 16GB.



Obr. 3.17: Graf závislosti výpočtového času od počtu bodov pre jednotlivé algoritmy

Po zobrazení začiatku grafu z obrázku č. 3.17, viď obrázok číslo 3.18, vidíme časovú závislosť algoritmov pri segmentácií jedného merania (360 prvkov v našom prípade). Jediný Hough algoritmus nedokáže spracovať výsledky pri malom množstve prvkov pod 1 sekundu. Veľmi dôležité je taktiež vzájomné relatívne postavenie algoritmov. Zatiaľ čo na začiatku, s malým počtom bodov, výrazne zaostáva Hough algoritmus, je z porovnania zjavné, že so zvyšujúcim sa počtom prvkov sa rozdiely pre algoritmy Hough, Ransac a Split and Merge vyrovnávajú. Algoritmus Line Tracking vykazuje najmenší vplyv zvyšovania výpočtového času oproti zmienenej trojice algoritmov. Tento jav pripisujeme ako dôsledok lineárne asymptotickej časovej zložitosti $O(n)$, kde n je počet prvkov. Tá bola dosiahnutá našim dodatkom k pôvodnému algoritmu, zahrňujúci prepočítanie priamky iba v nevyhnutných prípadoch, podrobne popísané v podkapitole 2.2.



Obr. 3.18: Priblížený graf závislosti výpočtového času od počtu bodov

Porovnanie kvality rôznych vlastností pre všetky nami navrhnuté algoritmy, vidíme v tabuľke 3.3. Stupnica hodnotenia 1 (najlepší) až 5 (najhorší). Ako môžeme vidieť, každý z algoritmov má v celkovom porovnaní určité prednosti, pričom nevyniká v iných oblastiach. Tabuľka slúži na celkové porovnanie algoritmov v rozličných oblastiach. Porovnávacia tabuľka sa dá taktiež použiť na vyhodnotenie algoritmov v rôznych oblastiach naraz. Výsledný algoritmus s najnižším súčtom bodov, by bol najoptimálnejší pri výbere určitých charakteristických vlastností.(r. je skratka pre rýchlosť algoritmu do určitého množstva bodov)

Algoritmus	Segmentácia	Preloženie	r. do 1000	r. nad 1000	Komplikovanosť
SEF	5	5	1	1	1
Line Tracking	3	1	2	4	2
Split & Merge	4	4	4	3	4
Ransac	2	3	4	2	4
Hough	1	2	5	5	5

Tabuľka 3.3: Vzájomné porovanie algoritmov

Záver

Predložená diplomová práca je štruktúrovaná do 3 hlavných kapitol, ktoré sú doplnené o úvod a záver.
Prvá kapitola je tvorená ako rešeršná štúdia získavania a následného spracovania dát bodov point cloudu. Okrem všeobecného opisu princípov fungovania lidar zariadenia, som v nej priblížil aj konkrétne používané zariadenie. Vysvetlená segmentácia bodov point cloudu, je doplnená o hlavné segmentačné metódy charakterizované podobnými vlastnosťami. V analýze súčasného stavu som uviedol najdôležitejšie kladné a záporné stránky prezentovaných skupín v odbornej literatúre. Z rozboru jednotlivých článkov je zjavné, že v rovine súčasného poznania, nejestvuje ideálny algoritmus, ktorý by mal optimálne výsledky pre väčšinu aplikácií. Výsledkom vykonaného rozboru je výber najdôležitejších algoritmov.
V druhej kapitole sa zaoberám podrobným opisom vybranej pätice algoritmov. Pre každý algoritmus je uvedený podrobný matematický popis jeho princípov s následným vlastným návrhom naprogramovaného algoritmu doplneného o zlepšujúce dodatky. Každá podkapitola prezentujúca jeden opisovaný algoritmus obsahuje aj zhodnotenie prezentovanej metódy z hľadiska časovej zložitosti a je doplnená o vývojový diagram algoritmu. V podkapitole spracovanie dát sú vysvetlené základné rozdiely spracovania vstupných dát pre jednotlivé algoritmy. Súčasne by som zdôraznil časť post-processingu, kde som dokázal vytvoriť dvojicu vlastných funkcií opravujúcich výstupy z jednotlivých algoritmov.
Najpodstatnejšia časť mojej práce je obsiahnutá v poslednej kapitole. V nej sa nachádza porovnanie výsledkov algoritmov na postupne sa zvyšujúcich stupňoch náročnosti (vstupnej množiny bodov). Na získaných výsledkoch je zreteľne ukázané, že algoritmy dokážu rozdeľovať analyzovanú množinu bodov do správnych tried, s nepatrnými chybami pri zložitých obrazcoch. Súčasne som na výsledkoch explicitne ukázal správne fungovanie funkcií post-processingu. Ten je schopný chybné rozdelenie množiny správne lokalizovať a opraviť, pričom nedochádza k nesprávnemu rozdeľovaniu či spájaniu už správne rozdelených množín. Z výsledkov preloženia kriviek na vytvorených dátach evidentne vyplýva, že použitá metóda nemá vplyv na tvorbu priamok.
Algoritmy boli taktiež použité na experimentálne meraných dátach z lidar zariadenia. Dostatočne kvalitné výsledky segmentácie dosiahli algoritmy nie len pri ich aplikácií na dvojicu meraní, ale aj na množine 20 po sebe sa opakujúcich meraniach. Nakoľko takáto množina obsahuje ďaleko hustejšie pole dát ako body jedného merania, oceňujem schopnosť algoritmov adaptácie na hustejšie pole bodov. Tú pripisujem zaradeniu minimálnej vzdialenosti bodov ako hlavnej metriky pri inicializácií hraničných hodnôt algoritmov. Výsledky preloženia priamok na experimentálne namerané body priniesli väčšiu diverzitu ako v prípade vytvorených dát. Výsledky natočení priamok sa z globálneho hľadiska líšia len minimálne. Pri číselnom zhodnotení dosahuje najvzdialenejšia priamka rozdiel oproti reálnej menší ako 1°.
V záverečnej kapitole je uvedené porovnanie časovej závislosti výpočtu algoritmov na

počte vstupných bodov point cloudu. Výsledkom takéhoto porovnania je skutočnosť, že vzájomná rýchlosť algoritmov sa voči sebe v smere zvyšovania bodov mení. Ďalší dôležitý výsledok je rýchlosť SEF algoritmu, ktorá je o dva rády nižšia ako u ostatných algoritmov a fakt lineárnej závislosti Line Tracking algoritmu dosiahnutej našim dodatkom. Sumárna tabuľka zobrazuje jednotlivé podstatné vlastnosti navrhnutých algoritmov. Ponúka nielen ich vzájomné porovnanie v popísaných oblastiach, ale aj prípadné nájdenie algoritmu dominujúceho v určitých aspektoch. Z tabuľky je taktiež možné najnižším súčtom číslic pre jednotlivý algoritmus, určiť optimálny algoritmus pre konkrétne vybranú skupinu funkcií.

Zoznam skratiek a symbolov

c Rýchlosť svetla $[\frac{m}{s}]$

d Vzdialenosť $[\frac{m}{s}]$

t Čas $[s]$

1D Jednorozmerný priestor

2D Dvojrozmerný priestor

3D Trojrozmerný priestor

FDN Fixed distance neighbors

KNN K-nearest neighbors

Lidar Laser imaging, detection, and ranging

LT Line Tracking

MEMS Micro electro mechanical systems

Obr. Obrázok

r. Rýchlosť

Ransac Random sample consensus

S&M Split and Merge

SEF Successive edge following

ToF Time of Flight

Zoznam príloh

Príloha 1 2021_DP_MarekSoos_192398_zdrojovyKod.zip

Literatúra

[1] ANGUELOV, D, B TASKARF, V CHATALBASHEV, D KOLLER, D GUPTA, G HEITZ a A NG. Discriminative learning of Markov random fields for segmentation of 3D scan data. *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*. IEEE, 2005, **2**, 169-176 vol. 2. ISBN 0769523722. ISSN 10636919. Dostupné z: doi:10.1109/CVPR.2005.133

[2] BHANU, Bir, Singkee LEE, Chih-Cheng HO a Tom HENDERSON. Range Data Processing: Representation of Surfaces by Edges. *International Association for Pattern Recognition*. Utah, 1986, **8**, 1-15.

[3] BORGES, Geovany, Marie-jose ALDON a Geovany BORGES. Line Extraction in 2D Range Images for Mobile Robotics. *Journal of Intelligent and Robotic Systems*. 2004, **40**(3), 267-297. ISSN 0921-0296. Dostupné také z: http://search.proquest.com/docview/28220436/

[4] BURGER, Wilhelm a Mark J. BURGE. *Digital Image Processing*. 2. London: Springer London, 2016. ISBN 978-1-4471-6683-2. Dostupné z: doi:10.1007/978-1-4471-6684-9

[5] CARTER, Jamie, Keil SCHMID a Kirk WATERS. *Lidar 101: An Introduction to Lidar Technology, Data, and Applications.: National Oceanic and Atmospheric Administration (NOAA) Coastal Services Center*. Charleston, 2012. Dostupné také z: https://coast.noaa.gov/data/digitalcoast/pdf/lidar-101.pdf

[6] DASS, Rajeshwar, PRIYANKA a Swapna DEVI. Image Segmentation Techniques. *International Journal of Electronics and Communication Technology* [online]. India, 2012, **3**(1), 66-70 [cit. 2021-5-11]. Dostupné z: http://www.iject.org/vol3issue1/rajeshwar.pdf

[7] ECKART, B., K. KIM a J. KAUTZ. HGMR: Hierarchical gaussian mixtures for adaptive 3D registration. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Springer Verlag, 2018, **11219**, 730-746. ISBN 9783030012663. ISSN 03029743. Dostupné z: doi:10.1007/978-3-030-01267-0_43

[8] FELZENSZWALB, Pedro a Daniel HUTTENLOCHER. Efficient Graph-Based Image Segmentation. *International Journal of Computer Vision*. Boston: Kluwer Academic Publishers, 2004, **59**(2), 167-181. ISSN 0920-5691. Dostupné z: doi:10.1023/B:VISI.0000022288.19776.77

[9] FISCHLER, Martin A a Robert C BOLLES. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Communications of the ACM*. California, 1981, **24**(6), 381-395.

[10] FORSYTH DAVID, A. PONCE JEAN. *Computer vision a modern approach*. New Jersey: Prentice-Hall, 2003, 693 s. ISBN 0-13-191193-7.

[11] *LiDAR: Driving the Future of Autonomous Navigation: Analysis of LiDAR technology for advanced safety*. 1. California: Frost and Sullivan, 2016.

[12] GRILLI, E, F MENNA a F REMONDINO. A REVIEW OF POINT CLOUDS SEGMENTATION AND CLASSIFICATION ALGORITHMS. *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences* [online]. Gottingen: Copernicus, 2017, **XLII-2/W3**(2), 339-344 [cit. 2021-5-21]. ISSN 16821750. Dostupné z: doi:10.5194/isprs-archives-XLII-2-W3-339-2017

[13] HOUGH, Paul V. C. *METHOD AND MEANS FOR REC0SGNIZENG COMPLEX PATTTERNS*. Spojené štáty americké. Uděleno 18.12.1962. Zapsáno 25.3.1960.

[14] JAIN, Ramesh a Rangachar KASTURI. *Machine vision*. Boston: McGraw-Hill, 1995, 549 s. ISBN 0-07-032018-7.

[15] JUNZHE, Shen. *A SIMULATED POINT CLOUD IMPLEMENTATION OF A MACHINE LEARNING SEGMENTATION AND CLASSIFICATION ALGORITHM*. Indiana, 2020. Diplomová práca. Purdue University.

[16] KÖRTING, Thales Sehn. How region growing image segmentation works. *Youtube* [online]. California: Google, 2014, 31. 7. 2014 [cit. 2021-5-21]. Dostupné z: https://www.youtube.com/watch?v=VaR21S8ewCQ

[17] KOSTER, Klaus a Michael SPANN. MIR: an approach to robust clustering-application to range image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. IEEE, 2000, **22**(5), 430-444. ISSN 0162-8828. Dostupné z: doi:10.1109/34.857001

[18] LARSSON, Christian a Erik NORÉN. *3D Object Classification using Point Clouds and Deep Neural Network for Automotive Applications*. Gothenburg, 2019. Diplomová práca. Chalmers University of Technology. Vedoucí práce Knut Åkesson.

[19] LI, Xinzhao, Yuehu LIU, Zhenning NIU a Zhichao CUI. A line segments extraction based undirected graph from 2D laser scans. *2015 IEEE 17th International Workshop on Multimedia Signal Processing (MMSP)*. IEEE, 2015, , 1-6. Dostupné z: doi:10.1109/MMSP.2015.7340851

[20] RPLiDAR A1M8. *Rpishop* [online]. Roudné, Česká republika [cit. 2021-5-21]. Dostupné z: https://rpishop.cz/gravirky-scannery-tiskarny/1631-rplidar-a1m8-360stupnovy-laserovy-scanner-kit-dosah-12m.html

[21] *RPLIDAR A1* [online]. Shanghai: SLAMTEC [cit. 2021-5-21]. Dostupné z: http://www.slamtec.com/en/lidar/a1

[22] *Light Source Measurement* [online]. Eckhardt Optics [cit. 2021-5-10]. Dostupné z: https://www.eckop.com/applications/light-source-measurement/

[23] LU, Xiaohu, Jian YAO, Jinge TU, Kai LI, Li LI a Yahui LIU. PAIRWISE LINKAGE for POINT CLOUD SEGMENTATION. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*. Copernicus, 2016, **3**, 201-208. ISSN 21949042. Dostupné z: doi:10.5194/isprs-annals-III-3-201-2016

[24] MIKE. LIDAR scanned SICK LMS animation. *Wikimedia Commons* [online]. Wikipedia, 2008 [cit. 2021-5-21]. Dostupné z: https://upload.wikimedia.org/wikipedia/commons/c/c0/LIDAR-scanned-SICK-LMS-animation.gif

[25] MOCKEY, Nick. A self-driving car in every driveway? Solid-state lidar is the key. *Digitaltrends* [online]. Portland: Digital Trends Media Group, 2018, 15.3.2018 [cit. 2021-5-21]. Dostupné z: https://www.digitaltrends.com/cars/solid-state-lidar-for-self-driving-cars/

[26] NGUYEN, Anh a Bac LE. 3D point cloud segmentation: A survey. *2013 6th IEEE Conference on Robotics, Automation and Mechatronics (RAM)*. IEEE, 2013, , 225-230. ISBN 9781479911981. ISSN 21582181. Dostupné z: doi:10.1109/RAM.2013.6758588

[27] NGUYEN, Viet, Stefan GÄCHTER, Agostino MARTINELLI, Nicola TOMATIS a Roland SIEGWART. A comparison of line extraction algorithms using 2D range data for indoor mobile robotics. *Autonomous Robots* [online]. Boston: Springer US, 2007, **23**(2), 97-111 [cit. 2021-5-10]. ISSN 0929-5593. Dostupné z: doi:10.1007/s10514-007-9034-y

[28] PAL, Nikhil R a Sankar K PAL. A review on image segmentation techniques. *Pattern recognition*. Elsevier, 1993, **26**(9), 1277-1294. ISSN 0031-3203. Dostupné z: doi:10.1016/0031-3203(93)90135-J

[29] PAVLIDIS, T a S.L HOROWITZ. Segmentation of Plane Curves. *IEEE Transactions on Computers*. IEEE, 1974, **C-23**(8), 860-870. ISSN 0018-9340. Dostupné z: doi:10.1109/T-C.1974.224041

[30] PREMEBIDA, Cristiano a Urbano NUNES. *Segmentation and geometric primitives extraction from 2D laser range data for mobile robot applications*. Portugalsko, 2005. Technický report. Universidade de Coimbra.

[31] RABBANI, T., F.A. VAN DEN HEUVEL, G. VOSSELMAN, H.G. MAAS a D. SCHNEIDER. Segmentation of point clouds using smoothness constraints. *ISPRS 2006: Proceedings of the ISPRS commission V symposium Vol. 35, part 6*. 2006, **35**, 248-253.

[32] RATSHIDAHO, T T, J R TAPAMO, J CLAASSENS a N GOVENDER. AN INVESTIGATION INTO TRAJECTORY ESTIMATION IN UNDERGROUND MINING ENVIRONMENTS USING A TIME-OF-FLIGHT CAMERA AND AN INERTIAL MEASUREMENT UNIT. *South African Journal of Industrial Engineering*. Bedfordview: South African Institute for Industrial Engineering, 2014, **25**(1), 145-161. ISSN 1012277X. Dostupné z: doi:10.7166/25-1-736

[33] RUAN, Xiaoyi a Baolong LIU. Review of 3D Point Cloud Data Segmentation Methods. *International journal of advanced network, monitoring, and controls*. Exeley, 2020, **5**(1), 66-71. Dostupné z: doi:10.21307/ijanmc-2020-010

[34] SABERI, Alan. Digital image processing: p043 Graph Cuts. *Youtube* [online]. California: Google [cit. 2021-5-21]. Dostupné z: https://www.youtube.com/watch?v=HMGX8HXskKk

[35] SAPPA, A.D a M DEVY. Fast range image segmentation by an edge detection strategy. *Proceedings Third International Conference on 3-D Digital Imaging and Modeling*. IEEE, 2001, , 292-299. ISBN 0769509843. ISSN 15506185. Dostupné z: doi:10.1109/IM.2001.924460

[36] SHI, Jianbo a J MALIK. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. IEEE, 2000, **22**(8), 888-905. ISSN 0162-8828. Dostupné z: doi:10.1109/34.868688

[37] SCHOLZ, Benjamin. Object Reconstruction. *Technische Aspekte Multimodaler Systeme* [online]. Hamburg: University of Hamburg, 2017 [cit. 2021-5-10]. Dostupné z: https://tams.informatik.uni-hamburg.de/lehre/2017ss/seminar/ir/doc/Benjamin_Scholz_object_reconstruction.pdf

[38] SIADAT, Ali, Axel KASKE, Siegfried KLAUSMANN, Michel DUFAUT a René HUSSON. An Optimized Segmentation Method for a 2D Laser-Scanner Applied to Mobile Robot Navigation. *IFAC Proceedings Volumes*. 1997, **30**(7), 149-154. ISSN 1474-6670. Dostupné z: doi:10.1016/S1474-6670(17)43255-1

[39] *3D VISION TECHNOLOGY - STEREO VISION* [online]. Mníchov, Nemecko: MVTec [cit. 2021-5-21]. Dostupné z: https://www.mvtec.com/technologies/3d-vision/3d-vision-technology-stereo-vision

[40] TREVOR, Alexander J. B., Suat GEDIKLI, Radu B. RUSU a Henrik I. CHRISTENSEN. *Efficient Organized Point Cloud Segmentation with Connected Components*. Georgia, 2013. Georgia Institute of Technology.

[41] VANDORPE, J., H. VAN BRUSSEL a H. XU. Exact dynamic map building for a mobile robot using geometrical primitives produced by a 2D range finder. *Proceedings - IEEE International Conference on Robotics and Automation*. IEEE, 1996, **1**, 901-908. ISSN 10504729.

[42] VARSHNEY, Vedaant. *LiDAR: The Eyes of an Autonomous Vehicle* [online]. San Francisco: Medium, 2019 [cit. 2021-5-10]. Dostupné z: https://medium.com/swlh/lidar-the-eyes-of-an-autonomous-vehicle-82c6252d1101

[43] VO, Anh-vu, Linh TRUONG-HONG, Debra F LAEFER a Michela BERTOLOTTO. Octree-based region growing for point cloud segmentation. *ISPRS journal of photogrammetry and remote sensing*. Ireland: Elsevier B.V, 2015, **104**, 88-100. ISSN 0924-2716. Dostupné z: doi:10.1016/j.isprsjprs.2015.01.011

[44] WANG, Jun a Jie SHAN. Segmentation of LiDAR point clouds for building extraction. *ASPRS 2009 Annual Conference Baltimore*. Maryland, 2009, , 1-12.

[45] WILLCOX, Eric N. *Forward Perception Using a 2D LiDAR on the Highway for Intelligent Transportation*. Worcester, 2016. Diplomová práca. Worcester Polytechnic Institute. Vedoucí práce Xinming Huang.

[46] WU, Bichen, Alvin WAN, Xiangyu YUE a Kurt KEUTZER. SqueezeSeg: Convolutional Neural Nets with Recurrent CRF for Real-Time Road-Object Segmentation from 3D LiDAR Point Cloud. *Proceedings - IEEE International Conference on Robotics and Automation*. Institute of Electrical and Electronics Engineers, 2018, , 1887-1893. ISBN 9781538630815. ISSN 10504729. Dostupné z: doi:10.1109/ICRA.2018.8462926

[47] *What is LiDAR technology?* [online]. Generation Robots, 2019 [cit. 2021-5-10]. Dostupné z: https://blog.generationrobots.com/en/what-is-lidar-technology/

[48] YALCIN, O, A SAYAR, O.F ARAR, S AKPINAR a S KOSUNALP. Approaches of Road Boundary and Obstacle Detection Using LIDAR. *IFAC Proceedings Volumes*. 2013, **46**(25), 211-215. ISSN 1474-6670. Dostupné z: doi:10.3182/20130916-2-TR-4042.00025