UNIVERZITA KOMENSKÉHO V BRATISLAVE

FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

Detekcia jednoduchých 3D objektov v RGB-D zázname

Bakalárska práca

2022 Andrej Paluch

UNIVERZITA KOMENSKÉHO V BRATISLAVE

FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

Detekcia jednoduchých 3D objektov v RGB-D zázname

Bakalárska práca

Študijný program: Aplikovaná informatika

Študijný odbor: Informatika

Školiace pracovisko: Katedra aplikovanej informatiky

Školiteľ: Ing. Viktor Kocur

Bratislava, 2022 Andrej Paluch

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Andrej Paluch
Študijný program: aplikovaná informatika (Jednoodborové štúdium, bakalársky I. st., denná forma)
Študijný odbor: informatika
Typ záverečnej práce: bakalárska
Jazyk záverečnej práce: slovenský
Sekundárny jazyk: anglický

Názov: Detekcia jednoduchých 3D objektov v RGB-D zázname
Detection of simple 3D objects in RGB-D footage

Anotácia: Toto zadanie je súčasťou projektu interakcie ľudského učiteľa s robotom. Robot pri tejto interakcii manipuluje jednoduchými objektmi na základe pokynov od ľudského učiteľa. Pre tento účel je tak vhodné aby robot dokázal správne detegovať pozíciu daných objektov pomocou svojej RGB-D kamery.

Cieľ: Cieľom tejto práce je navrhnúť, implementovať a otestovať algoritmus na detekciu jednoduchých 3D objektov v RGB-D zázname. Algoritmus bude navrhnutý a testovaný v kontexte interakcie ľudského učiteľa s robotom.

Vedúci: Ing. Viktor Kocur
Katedra: FMFI.KAI - Katedra aplikovanej informatiky
Vedúci katedry: prof. Ing. Igor Farkaš, Dr.

Dátum zadania: 30.09.2020

Dátum schválenia: 06.10.2020

doc. RNDr. Damas Gruska, PhD.
garant študijného programu

.....
študent

.....
vedúci práce

Čestné prehlásenie: Čestne prehlasujem, že som túto bakalársku prácu vypracoval samostatne s použitím uvedených zdrojov.

V Bratislave dňa: ....

Andrej Paluch

Abstrakt

Cieľom tejto bakalárskej práce je navrhnúť, implementovať a otestovať algoritmus na detekciu jednoduchých 3D objektov v RGB-D zázname. Práca je rozdelená na úvodnú časť popisujúcu prehlaď problematiky a možné prístupy rozpoznávania objektov z 3D dát, druhá časť práce je venovaná návrhu riešenia. V tretej a štvrtej časti práce si opíšeme použité technológie a implementáciu riešenia.

Kľúčové slová: detekcia jednoduchých 3D objektov, RANSAC

Abstract

The aim of this bachelor thesis is to design, implement and test an algorithm for detecting simple 3D objects in RGB-D recording. The work is divided into an introductory part describing an overview of the issue and possible approaches of recognizing objects from 3D data, the second part of the work is devoted to the design of the solution. Third and fourth part of the work will be dedicated to used technologies and implementation of the solution.

Keywords: simple 3D geometric primitives detection, RANSAC

Obsah

Zoznam obrázkov

Úvod

V dôsledku neustáleho vývoja 3D skenovacích zariadení a s komercializáciou hĺbkových kamier v posledných rokoch zaznamenávame obrovsky nárast zachytených 3D dát. Ich spracovanie a analýza je veľkou výzvou pre odvetvia počítačového videnia a počítačovej grafiky. Detekcia jednoduchých objektov z 3D dát v súčasnej dobe nájde uplatnenie v mnohých odvetviach od robotiky a modelovania až po architektúru. Rôzne aplikácie sa líšia na základe od odvetvia a problému, pre ktorý sú navrhnuté. Zadanie našej práce je zasadené do kontextu interakcie učiteľa s robotom. Na základe inštrukcií robot manipuluje s jednoduchými 3D objektami a na to potrebuje vedieť detegovať ich polohu. Na projekt je použitá cenovo dostupná hĺbková kamera a preto budeme musieť rátať so šumom a menej presnými dátami.

Kapitola 1

1 Prehľad problematiky

V tejto kapitole si uvedieme základné teoretické východiská, ktoré sú nevyhnutné pre pochopenie a neskoršie riešenie danej problematiky. Na úvod si vysvetlíme čo sú to jednoduché 3D objekty. Ďalej si popíšeme princípy na základe ktorých dokážu hĺbkové kamery zachytiť 3D dáta, základné reprezentácie 3D dát a možné konverzie medzi nimi. Tiež si uvedieme niečo o farebných modeloch a priestoroch. Hlavnou častou tejto kapitoly bude vysvetlenie algoritmu RANSAC a náhľad aj do ostatných metód detekcie jednoduchých 3D objektov.

1.1 Detegované objekty

V mojej práci sa zaoberám detekciou jednoduchých 3D objektov. To sú také, pre ktoré platí nasledujúca charakteristika:

- majú fixne daný a konečný počet globálnych vlastných parametrov, t.j. takých, ktoré definujú iba veľkosť, orientáciu a polohu daného objektu.
- sú konvexné (okrem kruhového prstenca).
- sú symetrické.
- majú jednoduchý tvar, a ich spájaním navzájom môžu vznikať iné komplexné objekty.

Jednoduché objekty spĺňajúce túto definíciu delíme v oblasti detekcie objektov z 3D dát na štyri kategórie:

- roviny a rovinne útvary
- kvádre a kocky
- gule, valce a kužele
- ostatné

Väčšina prístupov na detekciu 3D objektov sa snaží nájsť dané objekty skôr ako záplaty rovinných útvarov [1]. Roviny sú najzákladnejšie a najbežnejšie sa vyskytujúce trojrozmerné útvary v prostrediach vytvorených človekom. Rovinu vieme jednoducho definovať normálovým vektorom a vzdialenosťou od bodu počiatku súradnicovej sústavy. Kvádre a kocky vieme pomenovať ako súbory zostavených ortogonálnych rovín. Definovať ich môžeme stredom, vektorom orientácie a troma dĺžkami strán. Môžeme ich tiež definovať ich ôsmimi vrcholmi, alebo tiež parametrami jednotlivých rovín, ktoré ich tvoria.

Gúľa je izotropný objekt a možno ju tak jednoducho definovať stredom a polomerom. Aby sme mohli parametrizovať valec, potrebujeme poznať bod ležiaci v strede osi, polomer, výšku a taktiež vektor orientácie. Kužeľ vieme definovať podobne ako valec výškou, polomerom základne, orientačným vektorom a jeho vrcholom.

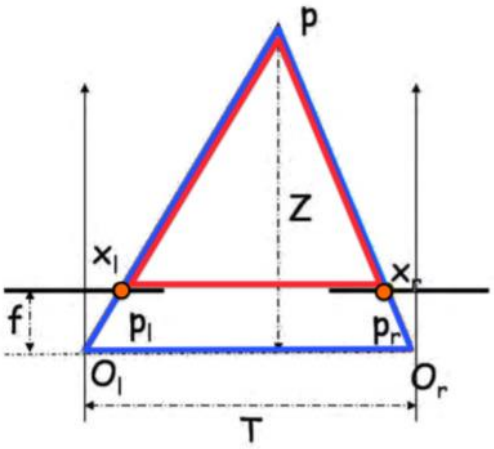
1.2 Range imaging

Range imaging je súhrnný názov pre techniky ktoré sa využívajú na zachytenie hĺbkovej informácie o scéne v podobe 2D obrázku, kde každý pixel reprezentuje vzdialenosť kamery od konkrétneho bodu v scéne. Príkladom jednej takejto techniky je stereoskopia. Princíp stereoskopie spočíva v tom že máme dve kamery, ktoré nám poskytnú dva mierne odlišné obrazy. Princíp stereoskopie je v zásade triviálny, keďže každý bod v obraze predstavuje priamku v 3D svete. Aby sme dokázali vypočítať hĺbku nejakého bodu v scéne potrebujeme poznať parametre kamery (ohnisková vzdialenosť, vzájomná vzdialenosť oboch kamier) a zobrazenia tohto bodu v oboch obrazoch. Dvojicu pixelov z jedného a druhého obrazu, ktoré vyobrazujú ten istý bod v 3D svete nazývame korešpondujúca dvojica. Každý z pixelov však predstavuje priamku možných bodov. Tie sa pretínajú práve v mieste kde daný bod leží na základe čoho vieme vypočítať jeho hĺbku. Ak platí, že sú kamery koplanárne, potom môžeme vyjadriť nasledujúci vzťah:

$$\frac{T + x_l - x_r}{Z - f} = \frac{T}{Z}$$

Z toho ľahko odvodíme hĺbku bodu:

$$Z = f\frac{T}{x_r - x_l} = f\frac{T}{D}$$



Obrázok 1: princíp stereo triangulácie

Práve nájdenie dvojice korešpondujúcich bodov je najväčší problém pri tejto technike. Napríklad pre pixle v obraze, ktoré sa nachádzajú v oblasti s homogénnou farbou alebo intenzitou je ťažké nájsť korešpondujúce body. Preto použitím tejto metódy dokážeme získať spoľahlivé odhady hĺbky iba pre podmnožinu všetkých viditeľných bodov. Veľkou výhodou stereoskopie však je, že na rozdiel od väčšiny ostatných techník, ktoré síce nemusia riešiť problém korešpondencie, nevyžaduje špeciálne podmienky týkajúce sa osvetlenia scény.

1.3 Vstupne dáta

Ako výstup z kamery dostaneme RGB-D záznam. RGB-D obraz je kombináciou normálneho RGB obrázku a jemu prislúchajúcej hĺbkovej mapy. Hĺbková mapa je obraz alebo obrazový kanál, ktorý obsahuje informácie týkajúce sa vzdialenosti povrchov objektov scény od kamery. Spojením s RGB obrázkom tak okrem farebnej informácie o scéne dostávame aj informáciu o tvare. Takýto RGB-D obraz vieme premietnuť do 3D svetových súradníc a získať tak farebný point cloud.

Point cloud, po slovensky mračno bodov, je množina bodov, ktorá reprezentuje scénu. Ideou point cloudu je navzorkovanie skutočného sveta, ktorý je spojitý, do jednotlivých bodov, čím dostaneme diskrétne dáta. Každý bod je reprezentovaný ako trojica XYZ súradníc s možnými dodatočnými informáciami ako napr. farba alebo normála. Point cloud môže byť štruktúrovaný, to znamená, že jeho body sú organizované v jednej, alebo viacerých 2D mriežkach, čo umožňuje rýchle prehľadávanie.

RGB-D obrazy a farebné point cloudy sú dve najčastejšie používane reprezentácie 3D dát z informáciou o farbe [2]. RGB-D obrazy sú používane hlavne v oblasti počítačového videnia, keďže majú rovnakú topológiu ako obrázky, zatiaľ čo v oblasti počítačovej grafiky sú využívane hlavne point cloudy [2]. Čo je dôležite je, že obe reprezentácie poskytujú rovnakú informáciu o scéne. So správnymi parametrami kamery je teda možná konverzia medzi oboma reprezentáciami. Point cloudy získane konverziou z RGB-D obraz sú navyše organizované, keďže je korešpondencia jedna k jednej medzi bodom v point cloude a pixelom v RGB-D obraze.

1.4 Pinhole camera model

Ako sme si uviedli vyššie, medzi RGB-D obrazom a point cloudom je možná konverzia, a práve konverzia z RGB-D obrazu do point cloudu je potrebná, keďže väčšina algoritmov na detekciu objektov pracuje práve s údajmi vo forme point cloudu.

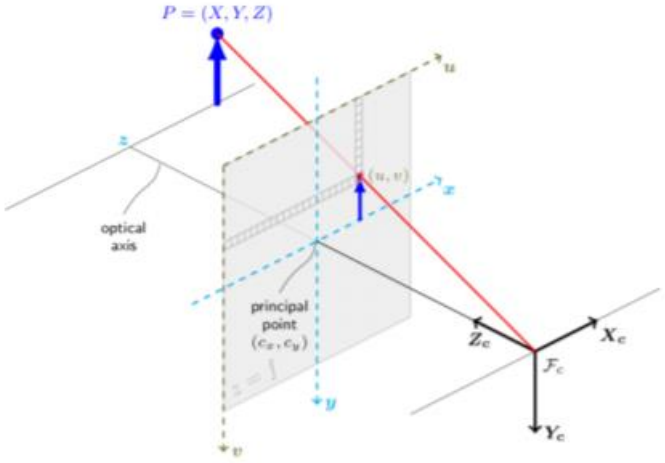
Pinhole camera model je matematický model, ktorý popisuje vzťah medzi súradnicami bodu v trojrozmernom priestore, a jeho premietnutím na rovinu obrazu. Je založený na princípe fungovania ideálnej dierkovej komory, čo je jednoduchá kamera bez objektívu. Hlavnou ideou tohto modelu je získanie 2D súradníc premietacej roviny z 3D súradníc sveta, keďže ale ku každému bodu z premietacej roviny (obrázku) poznáme aj jeho hĺbku, dokážeme tak naopak zo súradníc premietacej roviny získať súradnice sveta, A to na základe nasledujúcich pomerne jednoduchých rovníc.

XYZ sú súradnice sveta, xy sú súradnice premietacej roviny, z informácia o hĺbke a f je ohnisková vzdialenosť.

$$Z = z, \qquad X = \frac{Z}{f}x, \qquad Y = \frac{Z}{f}y$$

Štandardne je však nutné tieto rovnice ešte upraviť, keďže bežne je začiatok súradnicového systému obrazu v jeho ľavom hornom rohu a nie v bode prieniku optickej osi.

$$Z = z, \qquad X = \frac{Z}{f}(u - c_x), \qquad Y = \frac{Z}{f}(v - c_y)$$



Obrázok 2: pinhole camera model

1.5 Farebné priestory

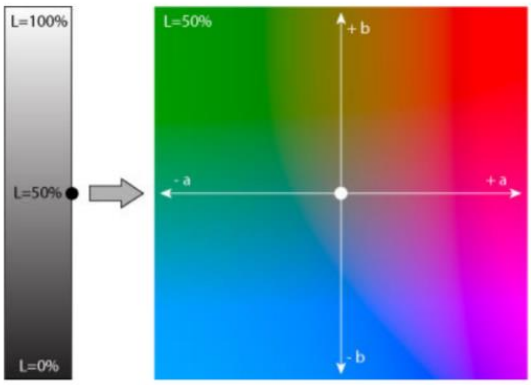
Farebný priestor je špecifická organizácia farieb, väčšinou založený na farebnom modely, čo je matematické vyjadrenie farby na základe podielu základných zložiek, ktorými je definovaný. Medzi základné farebné modely patrí RGB. Jeho základnými zložkami sú červená (Red), zelená (Green) a modrá (Blue). Pre tieto farby je príznačné, že ľudské oko má najlepšiu citlivosť pre ich vlnové dĺžky [3]. Model vieme reprezentovať ako jednotkovú kocku, kde každá z troch osi predstavuje jednu zložku. RGB je z pomedzi ostatných modelov asi najviac technický orientovaný, no pri práci s obrazmi, predovšetkým segmentácií, nie je veľmi efektívny. V hardvérovo orientovaných modeloch môže byť ťažké vyjadriť niektoré farby a preto vznikli používateľsky orientované modely, ktoré sú navrhnuté pre vnímanie farieb bližšie vnímaniu človeka.

Medzi takéto modely patrí napríklad Lab (často tiež L*a*b), ktorý bol navrhnutý tak, aby bol úplne nezávislý na zariadení. To ho umožňuje používať ako referenčný farebný model, čo znamená že pri prevode medzi dvoma modelmi ho môžeme použiť ako pomocný model. Je definovaný troma zložkami:

- svetlosť (L) – popisuje svetlosť bodu v rozsahu 0 až 100.
- zložka a – popisuje farbu na zeleno(-a) – červenej(+a) osi
- zložka b – popisuje farbu na modro(-b) – žltej(+b) osi

Najväčšími výhodami modelu Lab okrem samotnej strojovej nezávislosti je aj to, že má najširší rozsah zaznamenateľných farieb z pomedzi všetkých farebných modelov. Ďalšou veľkou výhodou je úplne oddelenie jasovej zložky od farebných zložiek, čo viac

korešponduje s tým ako vníma farby ľudské oko, ktoré je na zmeny jasu oveľa citlivejšie, ako na zmeny farieb.

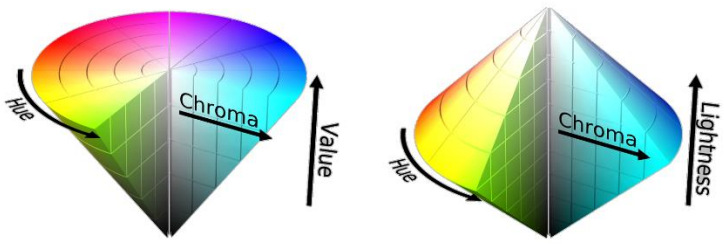


Obrázok 3: vizualizácia modelu Lab [3]

Ďalším používateľsky orientovaným modelom je HSV, ktorý asi najlepšie odpovedá ľudskému vnímaniu farieb [3]. Tiež je definovaný troma základnými zložkami:

- odtieň farby (Hue) – udáva sa v stupňoch od 0 po 360 a popisuje vlastnú čistú farbu.
- sýtosť (Saturation) – udáva sa v percentách a popisuje, ako veľmi je farba čistá, teda bez prímesi bielej.
- jas (Value) – popisuje jas farby v percentách

Podobným modelom ako je HSV je HSL, ale odstraňuje niektoré nedostatky HSV. Je definovaný odtieňom farby, sýtosťou a svetlosťou (Lightness). Jednou z jeho výhod je, že kontrastné farby čierna a biela tu majú svoju vlastnú zložku.



Obrázok 4: vľavo vizualizácia modelu HSV, vpravo vizualizácia modelu HSL

1.6 Metódy detekcie jednoduchých objektov

Keď už máme dáta v potrebnom tvare prichádza hlavná časť, ktorou je samotná metóda na detekciu objektov. Existuje viacero možných prístupov, z ktorých každý má svoje výhody aj nevýhody. Dané prístupy rozdeľujeme do troch skupín:

- stochastic – RANSAC, local statistic
- parameter spaces – Hough transform, Clustering parameter space
- clustering techniques – region growing, automatic clustering, segmentation then fitting

V praxi platí že mnoho konkrétnych aplikácií na detekciu jednoduchých 3D objektov nepoužíva čisto iba jednu z týchto metód, ale nejako vhodne ich kombinuje [1]. Dané metódy môžeme popísať na základe určitých charakteristík, na základe ktorých sa môžeme inšpirovať pri výbere algoritmu pre našu vlastnú aplikáciu. Niektoré z nich sú napr.:

- detected objects – prvá základná charakteristika podľa ktorej môžeme jednotlivé metódy rozlíšiť, a to podľa objektov, na ktorých detekciu sa zameriavajú.
- aplication context – kontext, v akom sú poskytnuté vstupne dáta. Teda či ide o interiéry, exteriéry alebo samostatné objekty.
- timing – podľa veľkosti vstupných dát môžeme rozlišovať metódy na tie, ktoré sú real-time, a tie ktoré nie.
- scalability – to ako sa ktorá metóda zachová pri zmene objemu dát
- user assistance – či metóda vyžaduje, alebo ponúka možnosť používateľovi zapojiť sa do procesu.
- needs extra information – či sa vyžaduje nejaká informácia navyše, aby sme mohli úspešne detegovať objekt.
- robustness to noise – či dokáže metóda správne detegovať objekt aj zo zašumených dát.
- Robustness to incomplete data – či dokáže metóda správne detegovať objekt aj z neúplných dát

1.7 Random sample consensus

Random sample consensus, skrátene RANSAC, je veľmi populárny iteratívny algoritmus určený na odhadovanie parametrov matematického modelu, prvý krát ho predstavili Fisher a Bolles v roku 1981 [4]. Je známy predovšetkým tým, že je obzvlášť odolný voči odľahlým hodnotám (tzv. outliers), to sú hodnoty ktoré buď nepatria modelu ktorý chceme detegovať, alebo sú to hodnoty, ktoré vďaka chybe pri meraní, alebo nesprávnou interpretáciou vstupných dát nadobudli nesprávnu hodnotu, a preto tiež nezapadajú do modelu.

Základnou myšlienkou RANSACu je vyskúšať veľa možných náhodných modelov, ktoré by mohli zodpovedať pravým nezašumeným dátam. Je to nedeterministický algoritmus, v zmysle, že poskytuje zmysluplný výsledok iba s určitou pravdepodobnosťou, tá však môže byť vysoká, ak zvolíme dostatočný počet iterácií. Algoritmus pozostáva z dvoch hlavných krokov:

1.7.1 Vzorkovanie vstupných údajov

Štruktúra RANSACu je pomerne jednoduchá. V prvom kroku je zo vstupnej množiny dát vybraná náhodná vzorka. Každý bod má rovnakú šancu, že bude vybraný do vzorky. Na základe tejto vzorky sa vypočíta pasujúci model a jemu zodpovedajúce parametre. Veľkosť vzorky závisí od objektu, ktorý detegujeme. V zásade ale platí, že veľkosť vzorky je najmenšia možná, ale taká aby bolo z nej možné určiť parametre modelu. Napríklad, ak by sme chceli detegovať kružnicu, potrebujeme do vzorky vybrať tri body, keďže kružnica je jednoznačne určená troma bodmi.

1.7.2 Ohodnotenie potenciálneho modelu

V druhom kroku sa ohodnotí kvalita potenciálneho modelu na celej vstupnej množine. Bežná funkcia na ohodnotenie jednoducho spočíta, ktoré body z celej množiny sú z potencionálnym modelom konzistentné. Tieto dva kroky sa stále opakujú, pričom si pamätáme najlepšie doposiaľ ohodnotený model. Opakujú sa dovtedy, kým buď nevyčerpáme počet iterácií, alebo natrafíme na taký model, ktorý dostatočne dobre vyhovuje vstupným dátam. Čo znamená dostatočne dobre je na konkrétnej aplikácií algoritmu. Existuje viacero variantov RANSACu, ktoré zachovávajú prvú časť kde sa náhodne vzorkuje, ale používajú zložitejšie metódy na ohodnotenie potenciálneho modelu.

1.7.3 Premenné

RANSAC používa na riadenie procesu detekcie tri premenné. Prvá určuje, s akou toleranciou môžeme o nejakom bode prehlásiť, že patrí potenciálnemu modelu. Druhá premenná je maximálny počet iterácií, ktoré musíme vykonať. Ten závisí od toho, koľko odľahlých hodnôt nepatriacich modelu sa nachádza vo vstupnej množine a od veľkosti vzoriek. Viac odľahlých hodnôt vo vstupných dátach a väčšia veľkosť vzoriek znamená viac iterácií ktoré musí algoritmus vykonať. Tretia premenná sa tiež týka tolerancie, a určuje ako dobre musí byť ohodnotený potencionálny model, aby sme ho mohli prehlásiť za detegovaný objekt.

```
while iterations < k do
    maybeInliers := n randomly selected values from data
    maybeModel := model parameters fitted to maybeInliers
    alsoInliers := empty set
    for every point in data not in maybeInliers do
        if point fits maybeModel with an error smaller than t
             add point to alsoInliers
    end for
    if the number of elements in alsoInliers is > d then
        // This implies that we may have found a good model
        // now test how good it is.
        betterModel := model parameters fitted to all points in maybeInliers and alsoInliers
        thisErr := a measure of how well betterModel fits these points
        if thisErr < bestErr then
            bestFit := betterModel
            bestErr := thisErr
        end if
    end if
    increment iterations
end while

return bestFit
```

Obrázok 5: algoritmus RANSAC - pseudokód [5]

1.7.4 vylepšenia

Výpočtová zložitosť RANSACu sa dá vylepšiť niekoľkými spôsobmi. Rýchlosť algoritmu závisí od dvoch faktorov. Prvým je počet iterácií, ktoré je potrebné vykonať, aby sme mali zaručené, že dostaneme dobrý odhad. Druhým je čas, ktorý nám zaberie ohodnotenie každého potenciálneho modelu, ten je úmerný veľkosti vstupnej množiny.

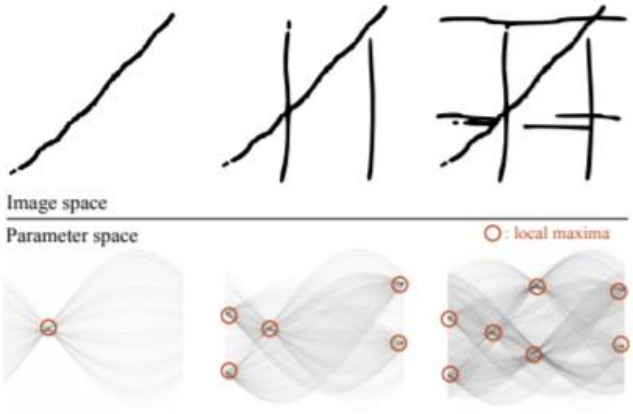
Platí, že veľké množstvo potenciálnych modelov je vytvorených z kontaminovaných vzoriek (t.j. vzoriek obsahujúcich body nepatriace modelu). Takéto modely sú konzistentné iba s malým množstvom údajov. Ohodnotenie modelov možno zrýchliť tým že . Každý potenciálny model najprv otestujeme iba náhodných bodov zo vstupnej množiny. Ak daný model nedostane dostatočnú podporu od tejto malej množiny náhodných bodov, potom môžeme z vysokou pravdepodobnosťou usúdiť, že nie je dobrým odhadom a ďalej nepokračovať s jeho ohodnocovaním. Modely, ktoré prejdú cez náhodné ohodnotenie sa potom vyhodnotia aj na celej množine. Takto vylepšený algoritmus sa volá Randomized RANSAC [6].

Výkonnosť algoritmu sa zhoršuje z narastajúcou veľkosťou vzorky, keďže je menšia šanca že nebude obsahovať žiadne hodnoty nepatriace modelu (outliers). Bežné pozorovanie je také, že hodnoty nepatriace modelu majú difúzne rozdelenie, na rozdiel od hodnôt patriacich modelu, ktoré majú tendenciu byť umiestnené blízko seba [7]. Preto, pôvodné vzorkovanie bodov môže byť nahradené vzorkovaním kde berieme do úvahy priestorové vzťahy bodov [8]. Počiatočný bod vzorky je vybraný náhodne. Ostatné body sú náhodne zvolené body, ktoré ležia v hypersfére so stredom v prvom bode. Výber vzorky, ktorá obsahuje susedné body môže významne zvýšiť pravdepodobnosť, že všetky body danej vzorky patria modelu, a tým pádom znížime počet potrebných iterácií algoritmu. Takto vylepšený algoritmus poznáme ako NAPSAC [8].

1.8 Ostatné prístupy

1.8.1 Hough transform

Princíp fungovania metódy Hough transform je založený na vytvorení parametrického priestoru. Parametrický priestor je toľko rozmerný, koľko parametrov definuje detegovaný objekt. Každý bod v ňom tak predstavuje jeden objekt. Po nakvantovaní parametrického priestoru dostaneme tzv. akumulačný priestor. Všetky body zo vstupných dát zahlasujú za všetky tie modely, ktorých sú súčasťou. Ako rozpoznaný objekt bude vybraný z akumulačného priestoru ten, ktorý ma najviac hlasov. Pôvodne sa tento algoritmus používal na detekciu hrán v obraze, no neskôr bol zovšeobecnený na detekciu 2D aj 3D objektov. Veľkou výhodou tohto algoritmu je odolnosť voči neúplným dátam, zato jeho hlavnou nevýhodou sú nedostatočne definované hranice rozmerov parametrického priestoru, čo môže byť problém z hľadiska pamäti a času.



Obrázok 6: príklad Houghovej transformácie na detekciu hrán [1]

1.8.2 Parameter space clustering

Metódy založené na tomto princípe priamo využívajú a analyzujú parametrický priestor aby detegovali jednoduché objekty. Predovšetkým sú využívane na detekciu rovín, keďže parametrický priestor pre roviny je dvojrozmerný a vieme rozdeliť do dvoch jednoduchých disjunktných priestorov. Sú to parametrický priestor normálových vektorov a parametrický priestor euklidovských vzdialeností od bodu počiatku súradnicového systému. V prvom kroku použijeme klusterizáciu vstupných bodov na základe ich normálovej orientácie. Dospejeme tak ku zoznamu lokálnych maxím priestoru normál. V tejto fáze náš výstup ešte nepredstavuje detegované roviny ako skôr množiny bodov s rovnakou orientáciou. V druhom kroku zoberieme na vstup výstup z prvého kroku, čiže zoznam lokálnych maxím priestoru normál, a pre každé takéto maximum použijeme klusterizáciu bodov na základe vzdialenosti od bodu počiatku. Ako výsledok tak dostaneme jednu alebo viac dvojíc normálových vektorov so vzdialenosťami od počiatku. Každá takáto dvojica predstavuje jednu detegovanú rovinu.

1.8.3 Clustering

Existuje niekoľko metód ktoré využívajú segmentáciu a zhlukovanie na detekciu jednoduchých objektov. V zásade rozdeľujeme tri hlavné typy zhlukovania [1]:

- primitive growing
- automatic clustering
- segmentatin then fitting

Algoritmus jednoduchého narastania regiónov sa používa na detekciu poprepájaných objektov buď v RGB-D obrazoch, alebo v štruktúrovaných point cloudoch, kde máme informáciu o susednosti. Na začiatku máme jedno alebo viac tzv. semienok, to sú body ktoré patria objektom. Potom iteratívne prechádzame ich susedné body, a tie môžeme k ním priradiť ak sú ich vlastnosti dostatočné podobné. Tieto vlastnosti ako farba, hĺbka alebo orientácia normály sú vyhodnocované ako podobné s určitou toleranciou, ktorá zvyčajne závisí od konkrétnej aplikácie algoritmu.

Algoritmy automatického zhlukovania postupujú iteratívne za cieľom rozdelenia bodov do rovnomerne distribuovaných oblastí. Dva najbežnejšie algoritmy automatického zhlukovania sú: K-Means a Mean-Shift. Algoritmus K-Means berie na vstupe fixné číslo k počtu zhlukov, ktoré sa majú detegovať. Najprv sa výbere k náhodných bodov, ktoré budeme považovať za ťažiská zhlukov. Zvyšné body potom priradíme k tým zhlukom, ktorých ťažiská sú najbližšie. Potom vypočítame nové ťažiská jednotlivých zhlukov a celý proces opakujeme až kým neskonvergujeme k výsledku.

Algoritmus Mean-Shift je metóda, ktorá hľadá ťažiská zhlukov ako lokálne maximá funkcie hustoty bodov. Na začiatku sa vyrobí kópia bodov. Iteratívne potom každý bod v kópie posúvame v smere ťažiska oblastí okolo aktuálnej pozície bodu. Veľkosť tejto oblasti určuje šírka pásma, čo je vstupným argumentom algoritmu. V krajných prípadoch, pri zvolení príliš malej šírky pásma by každý bod dostal svoj vlastný zhluk. Naopak pre príliš veľkú šírku pásma by algoritmus vrátil iba jeden zhluk, do ktorého by patrili všetky body. Výhodou Mean-Shiftu je hlavne to, že nepotrebujeme dopredu poznať počet zhlukov. Za výhodu môžeme tiež považovať fakt, že zhluky ktoré dostaneme nie sú sférické. Hlavnou nevýhodou algoritmu je jeho kvadratická zložitosť.

Pri metóde segmentation then fitting najprv segmentujeme vstupne dáta, aby sme sa čo najlepšie zbavili bodov nepatriacich modelu. Segmentovať môžeme napr. metódami ako region growing. V tomto prípade však dané segmentačné algoritmy nevyhľadávajú objekty, namiesto toho používajú heuristiky ako farba, vzdialenosť alebo nejaké iné špecifické pre konkrétnu aplikáciu. Nasleduje fáza samotnej detekcie objektov v zhluku alebo zhlukoch, ktoré sme dostali segmentáciou. Na to je vhodné použiť napríklad algoritmus RANSAC. Jeho použitím vieme dostať veľmi rýchlo presný model, keďže jednotlivé zhluky, ktoré sme dostali segmentáciou často zodpovedajú jednému objektu. Detekciu v zhluku však možno sformulovať aj ako optimalizačný problém, ten vieme riešiť napr. minimalizáciou energie, ktorá predstavuje vzdialenosť modelu od vstupných dát.

Kapitola 2

2 Návrh postupu

V tejto kapitole sa budeme venovať podrobnému návrhu algoritmu na detekciu jednoduchých 3D objektov z RGB-D záznamu. Ako sme si uviedli v prvej kapitole, existuje viacero metód, ktoré sú členené do kategórií. Taktiež sme si uviedli, že v praxi väčšina prístupov nepoužíva výhradne jednu metódu ale kombináciu viacerých [1]. Rovnako sa aj my v našom návrhu dotkneme viacerých metód, no ako hlavnú metódu, na ktorej bude náš algoritmus založený som si zvolil metódu RANSAC. Jeho hlavnou výhodou je robustnosť voči neúplným alebo nepresným dátam a preto je ideálny pre náš algoritmus, keďže snímané objekty uvidíme iba z jednej strany a musíme rátať aj z nie úplne presnými hĺbkovými dátami.

Postup riešenia si vieme rozdeliť do niekoľkých samostatných krokov:

- nasnímanie videa
- čítanie dát
- výpočet pointcloudu
- segmentácia podľa farby
- detekcia algoritmom RANSAC
- zobrazenie detegovaných objektov

2.1 Nasnímanie videa

Detekcia sa bude odohrávať v interiéri so stálymi svetelnými podmienkami. Scéna bude snímaná hĺbkovou kamerou IntelRealSense d435i [9], pričom dáta z kamery sa buď nahrajú do súboru alebo sa budú rovno posielať programu.

2.2 Čítanie dát

Čítanie dát z kamery alebo už nahratého súboru bude prebiehať v cykle, kde v každej iterácií dostaneme dvojicu farebný obraz a hĺbková mapa v podobe šedoúrovňového obrazu. Toto bude zároveň aj hlavný cyklus programu. S farebným obrazom budeme pracovať ako s dvojrozmerným poľom kde každý prvok poľa je trojica čísel reprezentujúca zložky rgb. S hĺbkovou mapou budeme taktiež pracovať ako s dvojrozmerným poľom s tým rozdielom, že prvok poľa je len jedno číslo popisujúce hĺbku bodu.

2.3 Výpočet point cloudu

Pre ďalší postup potrebujeme prekonvertovať hĺbkovú mapu na point cloud. To vieme urobiť využitím modelu dierkovej komory. Potrebujeme na to poznať rozmery

vstupného obrazu a ohniskové vzdialenosti kamery. Po konverzií dostávame point cloud ako dvojrozmerné pole trojíc. Ide teda o štruktúrovaný point cloud. Samotný point cloud nám pri ďalšom postupe nestačí a budeme potrebovať aj farebný obraz. Keďže oba majú rovnaké rozmery, môžeme ich kombináciu považovať za ofarbený point cloud.

Veľkosť point cloudu priamo závisí na rozlíšení vstupnej snímky, čiže môže ísť rádovo o státisíce bodov. Náš program by mal pracovať v reálnom čase. Do úvahy preto pripadá optimalizácia v podobe redukcie bodov. Zredukovať môžeme point cloud o také body, o ktorých môžeme už na základe ich hĺbky rozhodnúť, že určite nepatria žiadnemu z objektov, ktoré chceme detegovať. Ide predovšetkým o body pozadia, ktoré sú za určitou hranicou, ktorú si sami nastavíme tak aby najlepšie vyhovovala scéne ktorú kamera sníma.

2.4 Segmentácia podľa farby

Na získanie bodov patriacich detegovaným objektom využijeme znalosť ich farby. Je dôležite vybrať si vhodný farebný priestor, v ktorom budeme reprezentovať farby. Zvolil som si prácu s farbami v priestore CIE Lab. Hlavným dôvodom je nezávislosť farebných zložiek od zložky jasovej, vďaka čomu je tento priestor vhodný na segmentáciu.

Najprv potrebujeme zistiť farby patriace objektom. To spravíme tak, že si manuálne vyberieme z jedného alebo viacerých obrázkov časti, ktoré patria objektom. Z týchto častí potom spočítame pre každú zložku Lab priemernú hodnotu a priemernú odchylku. Toto je krok ktorý vykonáme iba raz, aby sme zistili farby objektov.

Na hľadanie bodov patriacich objektom použijeme dve prahovania. Pri prvom porovnávaní použijeme relatívne prísny prah, aby sme našli len zopár bodov o ktorých ale s veľkou pravdepodobnosťou budeme vedieť, že patria želaným objektom. Pre každú farbu pre ktorú sa algoritmus snaží nájsť objekty dostaneme pole nájdených bodov. Tie však môžu patriť viacerým objektom rovnakej farby. Body ktoré sme dostali preto potrebujeme zhlukovať. Na to použijeme algoritmus MeanShift, ktorý nájde ťažiská zhlukov. Všetky body tak budeme môcť roztriediť do samostatných polí.

Pri druhom porovnávaní sa už využijú priemerne pozície nájdených bodov, vďaka čomu sa môže použiť menej prísny prah. Výhodou takéhoto prístupu je, že sa minimalizuje početnosť zhlukov bodov, ktoré nepatria želaným objektom a ušetrí sa tým výpočtový čas, ktorý by zabrala ich následná detekcia.

2.5 Detekcia algoritmom RANSAC

Keď už máme dáta v potrebnom tvare na rad prichádza samotná detekcia. Cieľom je nájsť parametre objektu, tak aby najlepšie vyhovovali dátam. Všeobecné fungovanie algoritmu vieme zhrnúť do nasledujúcich bodov:

- Vyberieme náhodnú podmnožinu bodov

- Vybraná množina bude pozostávať len z toľko bodov, koľko je na definovanie objektu potrebné
- Ohodnotíme ako zistený model pasuje zo všetkými bodmi
- Opakujeme celý postup kým nie je splnené nejaké kritérium

Pri detekcií sa zameriame hlavne na jednoduchšie objekty, na ktorých definíciu nám stačí len pár bodov:

- Priamka – 2 body
- Rovina – 3 body
- Guľová plocha – 4 body
- Valec – 4 body

Ohodnotenie modelu

Model ohodnotíme počtom bodov, ktoré s ním súhlasia. Pre objekty ako rovina alebo priamka si môžeme určiť toleranciu, ktorá nám určí do akej vzdialenosti môžeme ešte bod zarátať medzi také, ktoré sedia s modelom.

Kritérium ukončenia

Základným kritériom ukončenia bude maximálny počet iterácií ktoré sa môžu vykonať. Vyhneme sa tak možnému zacykleniu. Ďalšie kritérium, ktoré môžeme pridať je ohodnotenie modelu ktoré nám stačí na to aby sme mohli zvoliť daný model ako detegovaný objekt.

Detekcia kvádru

Kváder nevieme jednoducho definovať niekoľkými bodmi. Môžeme sa však zamerať na steny, z ktorých sa skladá. Použitím algoritmu pre rovinu vieme nájsť parametre tej roviny, na ktorej leží najviditeľnejšia stena kvádru. Podľa vzdialenosti od roviny vieme získať body patriace tejto stene a z nich následne rohy kvádru podľa minimálnych a maximálnych hodnôt x-ových a y-ových súradnic.

2.6 Zobrazenie detegovaných objektov

Detegované objekty získame ako zoznam parametrov, ktoré ich definujú spolu s ďalšími parametrami ako napríklad farba pod ktorou sme ich detegovali. Prirodzene by sme však chceli detegované objekty vizualizovať. Tak ako z farebného obrazu a hĺbkovej mapy vieme získať súradnice bodov, vieme aj opačne zo súradníc bodov získať súradnice pixelu. Pomocou úsečiek a kružníc môžeme potom priamo do vstupného farebného obrazu zakresliť detegované objekty.

2.7 Vytvorenie datasetu

Aby sme mohli ohodnotiť kvalitu detekcie aj inak ako len od oka podľa vizualizácie je nevyhnutné vytvorenie datasetu s anotovanými objektami na ktorom by sme mohli

algoritmus otestovať. Bude sa jednať o menší dataset obsahujúci niekoľko anotovaných dvojíc farebný obraz – hĺbková mapa. Tieto dvojice manuálne vytiahneme z ľubovoľnej časovej stopy z rôznych dostupných nahratých záznamov. Potrebujeme však implementovať nástroj ktorým by sme vedeli ich vedeli anotovať.

2.8 Nástroj na anotáciu

Bude sa jednať o jednoduchý nástroj, ktorý nám umožní do farebného obrazu zakresliť jednoduché 3D objekty. Pre každý objekt, ktorý zakreslíme manuálne určíme o aký typ objektu sa jedná, a pod akou farbou ho evidujeme. Po zakreslení všetkých objektov nachádzajúcich sa na obrázku získame ich 3d súradnice rovnakým spôsobom ako pri konverzií farebného obrazu a hĺbkovej mapy do point cloudu. Výsledok zapíšeme do vhodného formátu, akým môže byť napr. json.

Kapitola 3

3 Technológie

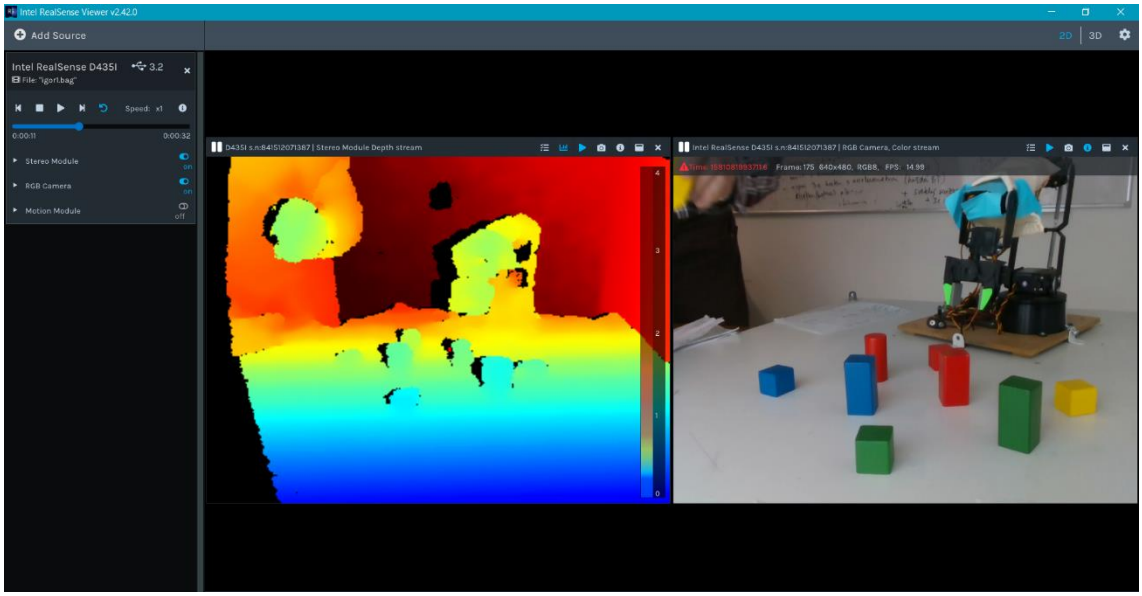
V tejto kapitole si rozoberieme technológie a softvérové riešenia, ktoré sú celosvetovo rozšírene a ideálne na použitie pri práci v danej problematike.

3.1 Kamera IntelRealSense d435i

Na snímanie objektov v 3D je použitá kamera Intel RealSense Depth Camera D435i [9], ktorá na výpočet hĺbky využíva stereoskopické videnie. Implementácia stereoskopického videnia pozostáva z ľavej a pravej kamery, a voliteľného infračerveného projektora, vďaka ktorému je presnosť hĺbky v snímke lepšia, a to hlavne v scénach s nevýraznou textúrou. Ľavá a pravá kamera zachytávajú scénu a posielajú údaje do procesora na spracovanie hĺbky (depth vision processor), ktorý vypočíta hĺbkové hodnoty pre každý pixel v obrázku, pomocou porovnania bodov na obrázku z ľavej kamery a obrázku z pravej kamery, a ich vzájomným posunom. Následne sa vytvorí hĺbková mapa, ktorá sa pridá ku RGB obrázku ako ďalší kanál. Kamera dokáže nasnímať 30 RGB snímok za sekundu v maximálnom rozlíšení 1920x1080 a až 90 hĺbkových máp za sekundu s rozlíšením 1280x720. Pri najvyššom rozlíšení je minimálna hĺbka, ktorú dokáže s veľkou presnosťou určiť 28cm.

Inter RealSense Viewer

Výrobca poskytuje ku kamere aj voľne dostupný softvér Inter RealSense Viewer, ktorý umožňuje prehrávať nahraté záznamy alebo naživo streamovať z pripojenej kamery a ukladať nahraté dáta do nového bag súboru. Program dokáže aj prepnúť do 3D módu vďaka čomu si môžeme vizualizovať point cloud a priamo aj vidieť ako sa mení v čase.



Obrázok 7: Intel RealSense Viewer

Intel RealSense SDK

Intel RealSense SDK je multi-platformová knižnica vytvorená pre prácu s dátami z hĺbkových kamier IntelrealSense. Je vytvorená pre programovacie jazyky C/C++, C#, Matlab, Node.js a Python a mnohé ďalšie frameworky ako napríklad Unity alebo OpenCV. Knižnica poskytuje predovšetkým funkcionalitu pre streamovanie farebného obrazu a hĺbky zo záznamu.

3.2 Python

Python je moderný objektovo orientovaný jazyk, ktorý patri k najpopulárnejším programovacím jazykom na svete a jeho popularita stále rastie. Na rozdiel od iných celosvetovo populárnych jazykom akými sú napr. Java alebo C++, ktoré sú kompilačné Python je interpretovaný jazyk. Python je taktiež netypový a pomerne vysokoúrovňový jazyk vďaka čomu je vývoj v ňom jednoduchší a tým pádom aj rýchlejší. Nevýhodou je jeho pomalší beh, čo je dôsledkom vyššie spomenutých výhod. Python môže byť až 5-krát pomalší ako Java preto pri vývoji musíme dbať aj na optimálnosť [10].

3.3 Knižnice

Pyrealsense2

Pyrealsense je pythonovský wrapper na multi-platformovú knižnicu Intel RealSense SDK, ktorá je určená pre prácu s dátami z kamier InterRealSense [11].

Numpy

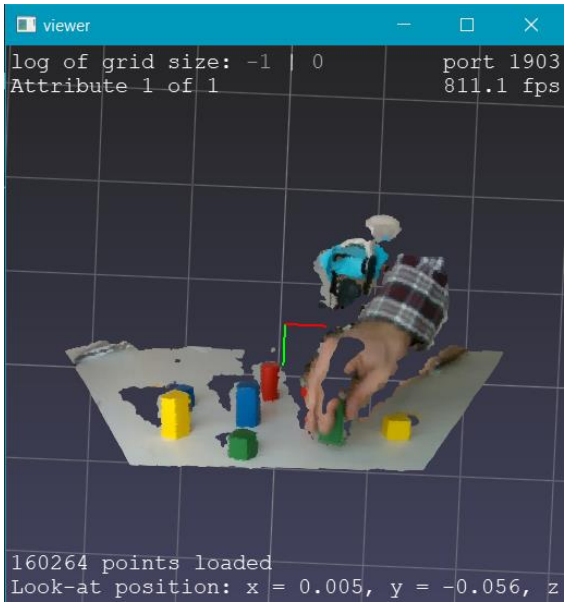
Numpy je voľne dostupná pythonovska knižnica určená pre efektívnu prácu s vektormi, maticami a predovšetkým viacrozmernými poliami. Okrem toho poskytuje aj celu radu matematických funkcií, ktoré môžeme použiť pri práci s poliami [12].

OpenCV

OpenCV je voľne dostupná knižnica, ktorá je určená primárne na prácu z obrazom v oblasti počítačového videnia a strojového učenia. Knižnica je vytvorená pre programovacie jazyky C++,Java a Python a je podporovaná operačnými systémami Windows, Linux, MacOS, iOS a Android [13].

PPTK viewer

Point cloud viewer je pythonovska knižnica určená pre vizualizáciu a spracovanie 2D a 3D point cloudov. Podporuje Numpy polia ako vstup a je preto vhodná najmä pri debugovaní.



Obrázok 8: point cloud viewer

Scikit-learn

Scikit-learn je voľne dostupná pythonovská knižnica zameraná na využitie v oblasti strojového učenia. Obsahuje rôzne algoritmy určené na klasifikáciu, regresiu alebo klusterizáciu. Je navrhnutá tak, aby bola kompatibilná s knižnicami ako NumPy alebo SciPy [14].

Kapitola 4

4 Implementácia

4.1 Detekcia kvádrov

V implementácií sme využili existujúce riešenie, ktoré ma implementované metódy pre detekciu kvádrov a kociek. Funkcionalita programu je rozdelená do nasledujúcich súborov:

- camera_utils.py
- collect.py
- detection.py
- geometry.py
- gesture_dataset.py
- record_bag.py
- run.py
- run_serial.py

Základné funkcie pre získavanie dát zo záznamu sa nachádzajú v súbore camera_utils.py. Funkcia ***retrive_aligned_pipeline*** prečíta vstupný súbor a vráti objekt, z ktorého potom funkcia ***get_images_from_pipeline*** získava korešpondujúce dvojice farebný obraz a hĺbka až kým neskončí záznam.

V súbore geometry.py sa nachádzajú funkcie zamerané na prevody medzi 2D súradnicami v obraze a 3D súradnicami v point cloude. Hneď po získaní farebného obrazu a hĺbkovej mapy funkcia ***create_xyz*** vytvorí point cloud. Ten sa následne zredukuje o body pozadia funkcia ***reduce_pts***.

Pre každú farbu, ktorá je zadefinovaná funkcia ***find_pts*** nájde zodpovedajúce body patriace objektom danej farby. Využitím algoritmu MeanShift z knižnice sklearn sa body patriace viacerým objektom rovnakej farby rozdelia do samostatných polí.

```
colors = {'red': [79.46608003, 173.12259944, 159.88620013],
          'green': [49.37390089, 114.06434519, 131.26390264],
          'yellow': [147.95821667, 131.98808242, 189.42594587],
          'blue': [48.36433141, 129.38893223, 107.09408644]
         }

color_devs = {'red': [12.45986815, 5.20190592, 4.17891806],
              'green': [5+9.3610654, 6+2.21609464, 3+1.59495349],
              'yellow': [28.13293378, 4.28111556, 9.63940965],
              'blue': [5+9.6500446, 1+1.58701702, 1+3.78901351]
             }
```

Obrázok 9: farby a dovolené odchýlky v lab

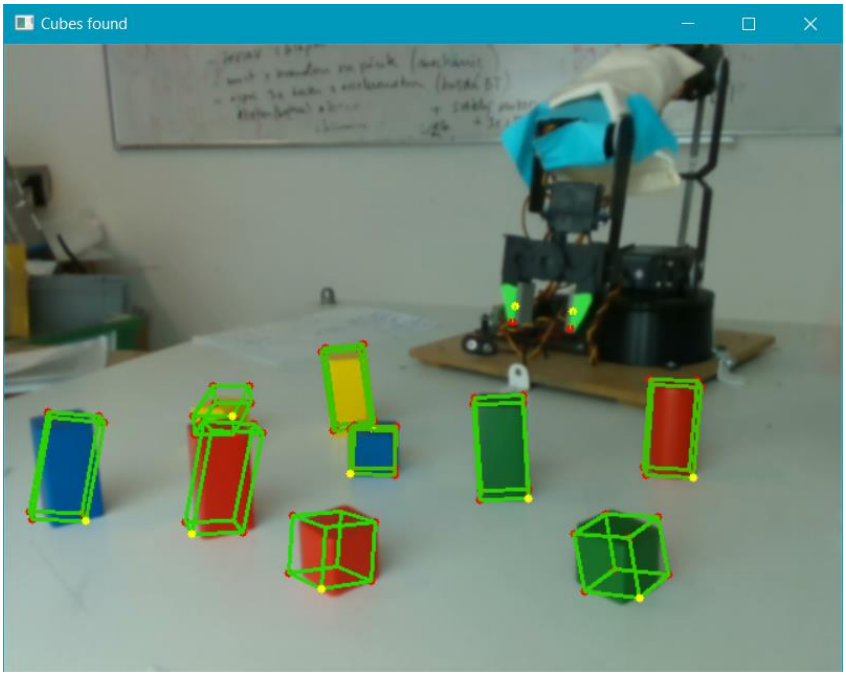
Funkcia ***find_object_ransac*** sa pokúsi detegovať kocku alebo kváder. Výsledok detekcie zapíše do premennej typu dict a obsahuje nasledujúce parametre:

- type – typ objektu (kocka, kváder atď..)
- corners – súradnice rohov
- color_name – názov farby objektu
- color_lab – farba objektu v priestore lab
- conf – hodnota v rozsahu 0-1 popisujúca ako veľmi sme presvedčený, že to čo sme detegovali naozaj patrí želanému objektu.

```
{'type': 'cube', 'corners': array([[-0.02829713,  0.09357697,  0.30243829],
       [-0.02919432,  0.06963836,  0.3009762 ],
       [-0.00521291,  0.06876089,  0.30062686],
       [-0.00431572,  0.09269951,  0.30208895],
       [-0.02800214,  0.09210295,  0.32639117],
       [-0.02889934,  0.06816434,  0.32492907],
       [-0.00491793,  0.06728687,  0.32457974],
       [-0.00402073,  0.09122549,  0.32604183]]),
 'color': 'green', 'conf': 0.8253943298652149,
 'color_name': 'green', 'color_lab': [104.55, 87, 156]
}
```

Obrázok 10: detegovaný objekt zapísaný do premennej typu dict

Pre všetky detegované objekty sa vykoná funkcia ***draw_object***, ktorá ich vizualizuje.



Obrázok 11: vizualizácia detegovaných objektov

4.2 Chýbajúca implementácia

Detekcia valcov

V aktuálnom riešení vieme detegovať len kvádre a kocky. V dostupných záznamoch sa však častokrát nachádzajú aj valce. Detekcia algoritmom RANSAC nájde najlepšie pasujúce riešenie, a tak aj pre iné objekty nájde nejaké riešenie, vieme však posúdiť, že sa jedná o objekt iného typu podľa parametru conf.

Vytvorenie datasetu

Čo nám tiež chýba je vytvorenie datasetu s anotovanými objektami, ktoré by nám slúžili ako referenčne dáta a vedeli by sme na nich otestovať algoritmus. Anotovaný dataset bude súbor vo formáte json, ktorý bude pre každú korešpondujúcu dvojicu farebného obrazu a hĺbkovej mapy z datasetu obsahovať cesty k nim na disku a zoznam objektov.

```
{
    "igor1r": {
        "rgb": "C:/Users/13and/Desktop/rs-vision-master\\annotation\\colors\\igor1r_color_45.jpg",
        "depth": "C:/Users/13and/Desktop/rs-vision-master\\annotation\\depths\\igor1r_depth_45.png",
        "objects": {
            "red": [ ... ],
            "blue": [ ... ],
                .
                .
                .
    },
        .
        .
        .
}
```

Obrázok 12: json s anotovanými objektami

Záver

Cieľom tejto bakalárskej práce bolo navrhnutie, implementácia a otestovanie algoritmu na detekciu jednoduchých 3D objektov z RGB-D záznamu, čo sa nám podarilo iba z časti. V prvej časti práce sme sa venovali úvodu do problematiky. Ukázali sme si princípy na ktorých fungujú zariadenia zachytávajúce 3D dáta, rôzne druhy reprezentácie 3D dát a tiež existujúce metódy detekcie jednoduchých objektov z 3D dát. V druhej časti sme načrtli návrh toho ako by mal algoritmus fungovať, a tiež čo je potrebné aby sme mohli algoritmus otestovať. Na základne návrhu sme postupovali pri implementácií algoritmu. Algoritmus, ktorý sme implementovali kombinuje dve možne prístupy detekcie jednoduchých objektov. Ide o algoritmy MeanShift ktorým najprv segmentujeme vstupne dáta a algoritmus RANSAC ktorým sa snažíme detegovať objekty. Implementovaná je zatiaľ iba detekcia kvádrov a kociek pričom v záznamoch sú aj iné jednoduché objekty, ktoré by sa mali detegovať. V práci tiež chýba vytvorenie anotovaného datasetu na ktorom by sme mohli otestovať správnosť algoritmu.

Literatúra

[1] Adrien Kaiser, Jose Alonso Ybanez Zepeda, Tamy Boubekeur, A Survey of Simple Geometric Primitives Detection Methods for Captured 3D Data, LTCI, Telecom ParisTech, Paris-Saclay University.

[2] Paul L. Rosin, Yu-Kun Lai, Ling Shao, Yonghuai Liu, RGB-D Image Analysis and Processing, Springer Nature, 2019.

[3] J. Chudý, Farebné modely a práca s nimi, bakalárska práca, Univerzita Tomaše Bati ve Zlíně, 2007.

[4] M.A. Fischler and R.C. Bolles, Random sample consensus: A paradigm for model fitting with application to image analysis and automated cartography, Communications of the ACM, 1981.

[5] „Random sample consensus," [Online]. Available: https://en.wikipedia.org/wiki/Random_sample_consensus.

[6] O. Chum and J. Matas, Randomized ransac with t(d,d) test, Brittish Machine Vision Conference, Cardiff, UK, 2002.

[7] H. Cantzler, Random Sample Consensus (RANSAC), Institute for Perception, Action and Behaviour, Division of Informatics, University of Edinburgh.

[8] D.R. Myatt, P.H.S. Torr, S.J. Nasuto, J.M. Bishop, and R. Craddock., Napsac: High noise, high dimensional robust estimation - it's in the bag..

[9] „Intel® RealSenseTM Product Family D400 Series datasheet," 2020. [Online]. Available: URL: <https://www.intelrealsense.com/wp-content/uploads/2020/06/Intel-RealSense-D400-Series-Datasheet-June-2020.pdf>.

[10] Python3, „https://www.python.org," [2022].

[11] pyrealsense2, „https://dev.intelrealsense.com/docs/python2," [2022].

[12] NumPy, „https://numpy.org," [2022].

[13] OpenCV, „https://opencv.org," [2022].

[14] Scikit-learn, „https://scikit-learn.org/stable/," [2022].

[15] Dibya Jyoti Bora, Anil Kumar Gupta, Fayaz Ahmad Khan, Comparing the Performance of L*A*B* and HSV Color Spaces with Respect to Color Image Segmentation, Department of Computer Science & Applications, Barkatullah University, Bhopal, India, 2015.

[16] I. R. SDK, „https://www.intelrealsense.com/sdk-2/," [2022].