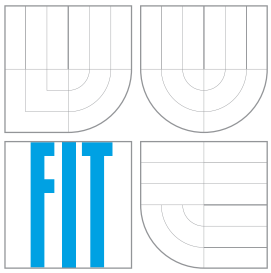
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

DETEKCE OBJEKTŮ POMOCÍ KINECTU

OBJECT DETECTION USING KINECT

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — JÁN ŠVEC
AUTHOR

VEDOUCÍ PRÁCE — Ing. MICHAL ŠPANEL, PhD.
SUPERVISOR

BRNO 2012

Abstrakt

Tato bakalářská práce se zabývá detekcí objektů pomocí Kinectu. Kinect je pohyb snímající zařízeni od společnosti Microsoft pro hrací konzoli Xbox 360. Cílem práce bylo zhodnotit existující přístupy a postupy používající pro detekci nejen obraz z kamery, ale i hloubkovou mapu (RGB–D senzor). Blíže se práce zabývá nástrojem RoboEarth, jeho instalací, nastavením, tvorbou 3D modelu a detekcí. Samotná detekce byla v práci zkoumána experimentálně a také vyhodnocena.

Abstract

This bachelor's thesis deals with the object detection using Kinect. Kinect is a motion sensing input device by Microsoft company for the Xbox 360 video game console. The objective of this work was to evaluate the existing approaches and practices for the detection not only using the image from the camera but also depth map (RGB–D sensor). The work deals in details with RoboEarth tool, its installation, settings, creation of 3D model and the detection. Performance of the RoboEarth tool was in the work investigated experimentally and also evaluated.

Klíčová slova

detekce, Kinect, SIFT, SURF, RoboEarth, ROS, Mračno bodov, DOT

Keywords

detection, Kinect, SIFT, SURF, RoboEarth, ROS, Point cloud, DOT

Citace

Ján Švec: Detekce objektů pomocí Kinectu, bakalářská práce, Brno, FIT VUT v Brně, 2012

Detekce objektů pomocí Kinectu

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Michala Španela PhD. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Ján Švec
15. května 2012

Poděkování

Chtěl bych poděkovat vedoucímu práce Ing. Michalu Španělovi PhD. za užitečné rady a odbornou pomoc při řešení práce. Dále bych rád poděkoval své rodině a přátelům za podporu.

Zoznam obrázkov

Obsah

Kapitola 1

Úvod

Na detekciu objektov je možné nahliadať ako na klasifikačný problém, kde potrebujeme rozlišovať v scéne hľadaný objekt od iných objektov. Človek tento problém prirodzene rieši bez väčších problémov. Počítač ako stroj túto schopnosť nemá, takže sa ho človek snaží naučiť čo najlepšie túto vlastnosť napodobiť. Disponuje lepšími predpokladmi ako súčasné počítače. Oči bežného človeka majú podstatne lepšiu rozlišovaciu schopnosť ako akýkoľvek fotoaparát či kamera. Taktiež trénovanie prebieha u človeka na neporovnateľne väčšej sade dát, počítače ich majú zväčša len hŕstku. Tomu však ale nemusí byť vždy tak.

Táto práca pojednáva o detekcii pomocou Kinectu a to tak, že sú tu uvedené metódy a postupy spájajúce pri detekcii obrazovú a hĺbkovú informáciu. Jej cieľom bolo vytvoriť a predviesť takýto robustný detektor v akcii. Ako referenčný systém tejto práce bol zvolený RoboEarth. Predstavuje sa tu jeho získanie, vytvorenie 3D modelu pre trénovanie detektoru, samotná detekcia a samozrejme aj vyhodnotenie jeho úspešnosti. RoboEarth pracuje s obrazom ako s množinou kľúčových bodov získaných a opísaných metódou SURF, ktoré si na základe informácie z hĺbkovej mapy prevedené do 3D priestoru. Korešpondujúce body sa vyberú algoritmom najbližšieho suseda (k-ANN) a na základe ich geometrických vlastností.

Práca je delená do kapitol, kde po Úvode 1 nasleduje kapitola 2 predstavujúca Kinect a niektoré lokálne príznaky. Ďalšia kapitola 3 predstavuje stručný popis metód a postupov pre detekciu. Kapitola 4 opisuje nástroj RoboEarth. Návrh systému, opis experimentov kapitola 5. Predposledná kapitola 6 predstavuje vyhodnotenie experimentov. A v závere 7 sú zhrnuté získané poznatky.

Kapitola 2

Detekcia objektov s použitím Kinectu

V tejto kapitole bude predstavený samotný Kinect, princípy detekcie a opisu významných bodov v obraze použitím SIFT a SURF.

2.1 Detekcia objektov

Detekcia objektov všeobecne spadá do oboru *počítačového videnia*. Pod pojmom objekt sa dá chápať napríklad ľudská tvár, auto, pohár, atď.,čiže čokoľvek zachytené v obraze. Ja sa v mojej práci budem venovať detekcii objektov s ktorými sa môžeme stretnúť bežne doma, ako sú šálky a rôzne krabičky. Takýto druh detekcie sa môže uplatňovať napríklad v robotike, kde sa robot pokúša nájsť objekt ktorý má priniesť.

Nedefinoval som ale, čo vlastne pod detekciou máme rozumieť. Detekciou môžeme chápať činnosť, kedy sa snažíme v obraze nájsť hľadaný objekt. Ak sa nám to podarí, je tiež nutné v tomto obraze tento objekt alebo objekty označiť (obr. 2.1). Takto by sme mohli chápať všeobecne detekciu, ale existujú aj iné prípady, kedy môžeme hovoriť o detekcii. Jedným z nich je keď vieme, že v danom obraze je hľadaný objekt a musíme určiť len pozíciu prípadne natočenie. Tomu sa hovori *lokalizácia* prípadne odhadnutie pozície. Môže to byť sekvencia po sebe idúcich snímkov s objektom a my máme tento objekt sledovať. Ja sa zameriam na prípady, kedy nevieme či sa objekt v obraze skutočne nachádza.

S detekciou a jej vyhodnocovaním sa spája niekoľko špecifických pojmov[14]. *True positive rate*(TPR), tiež nazývaný *recall* je pomer medzi počtom správnych detekcií(True positive) a počtom všetkých detekcií, ktoré mohli nastať. Môžeme to interpretovať ako mieru správnosti detekcie. *Positive predictive value*(PPV) viac nazývaný *precision* je pomer správnych detekcií k všetkým detekciám ktoré nastali. Pre zhodnotenie týchto parametrov je možné zostaviť krivku *ROC (Receiver–Operating Charakteristic)*.

2.2 Kinect

Kinect[16] je pohyb snímajúce zariadenie od spoločnosti Microsoft, pôvodne určené len pre ich hraciu konzolu Xbox 360. Nová verzia predpokladá aj možné pripojenie k počítaču s operačným systémom Windows. Jeho cieľom je, aby bolo možné ovládať veci intuitívne, pohybom. Kinect sa dokonca dostal do Guinessovej knihy rekordov, keď sa za prvých 60 dní predalo celkovo 8 miliónov kusov. Hlavné časti sú RGB kamera, hĺbkový senzor

Obrázok 2.1: Ukážka detekcie z prác [5] a [7].

a všesmerový mikrofón. Hĺbkový senzor sa skladá z infračerveného projekčného laseru kombinovaného s monochromatickým CMOS senzorom, ktorý zachytáva 3D dáta v akýchkoľvek svetelných podmienkach. Táto konštrukcia (obr. 2.3) umožňuje snímanie farebného obrazu scény synchronizovaného s obrazom hĺbkovej mapy. Mapa má rovnaké rozlíšenia v pixloch ako farebný obraz, kde hodnota pixlu predstavuje vzdialenosť od kamery, čím väčšia tým je objekt bližšie. Vhodná reprezentácia takýchto dát sa ukázalo ako mračno bodov, podrobnejšie opísané v kap. 2.4.

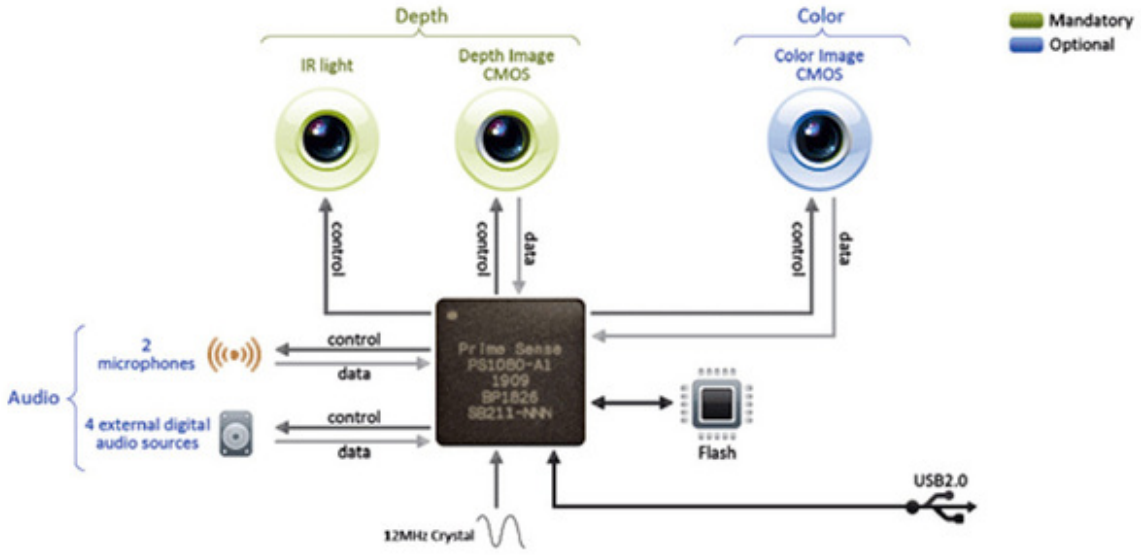
Obrázok 2.2: Kinect. Zdroj:
<http://blog.robotiq.com/Portals/13401/images/Kinect-resized-600.jpg>.

Technické informácie:

- **RGB kamera** : farebný, 8 bit hĺbka, rozlíšenie 640x480
- **Hĺbkový senzor** : monochromatický, 11 bit hĺbka, rozlíšenie 640x480, 2048 úrovní citlivosti
- **Frame rate** : 30 snímkov/s
- **Rozsah snímania hĺbkovej mapy** : 40cm – 2,4m[1]
- **Rozhranie** : USB

Po uvedení tohoto zariadenia na trh zo strany open source komunity, presnejšie firmou Adafruit, zazneli hlasy o vytvorenie slobodných ovládačov, dokonca táto firma vypísala aj

[1] Určené mojimi experimentami



Obrázok 2.3: Konštrukcia Kinectu. Zdroj:
<http://vipultaneja.com/wp-content/uploads/2010/11/kinect-3.jpg>.

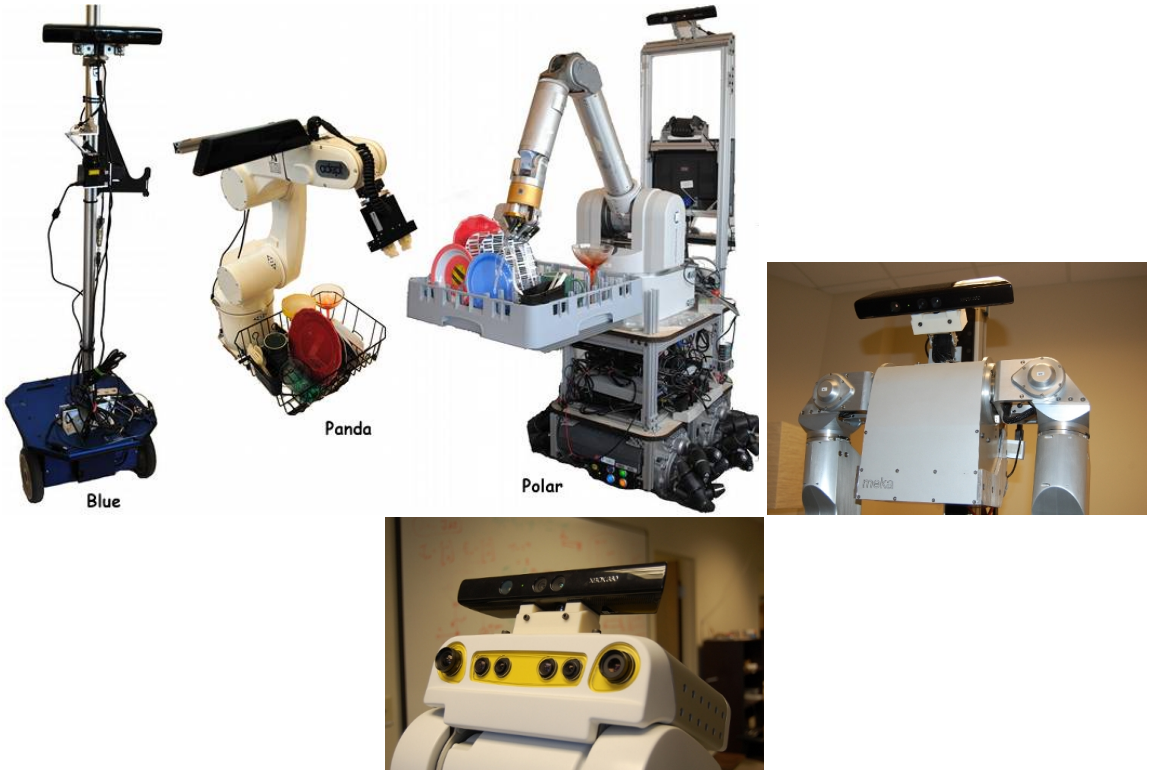
peňažnú odmenu pre toho, komu sa to prvému podarí. Tento počin bol zo začiatku chápaný zo strany Microsoftu ako hacknutie, to sa však vysvetlilo ako nedorozumenie. V novembri 2010 bola odmena vyplatená a v júni 2011 Microsoft vydal vývojovú sadu pre nekomerčné účely pre svoj operačný systém Windows.

Najväčšie uplatnenie mimo sveta počítačových konzol našiel Kinect ako 3D snímací senzor v robotike. Svedčia o tom fotky robotov (obr. 2.4) a články, ktoré boli na túto tému za posledný rok vydané. Dá sa povedať, že Kinect spôsobil na tomto poli vďaka svojej použiteľnosti a cene revolúciu. O to viac je možné sa tešiť na uvedenie Kinectu2, ktorý sľubuje vyššie rozlíšenie, vyšší frame-rate snímania, možno dokonca aj snímanie pohybu pier.

Firma Asus pod vedením PrimeSense, táto firma je zodpovedná za software interpretujúc gestá pre ovládanie bez dotykov pomocou Kinectu, vydala podobné zariadenie kompatibilné s PC názvom *WAVI Xtion*.

2.3 SIFT a SURF

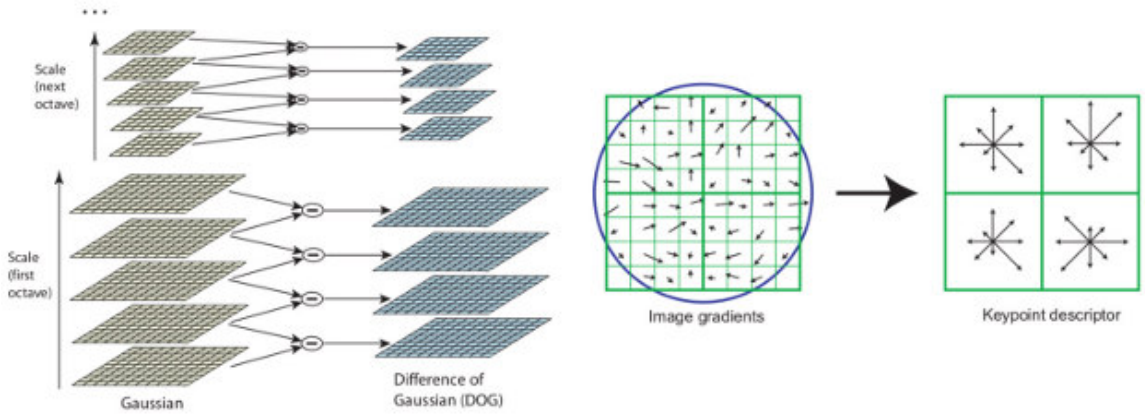
Scale Invariant Feature Tranform skratkou SIFT[9], je metóda pre detekciu a popis významných bodov(InP) v obraze. Touto metódou získané body sa vyznačujú invariantnosťou voči mierke, rotácii a čiastočne i osvetlení. Ako jadro detekcie bodu používa rozdiel gaussia-



Obrázok 2.4: Roboti s Kinectom. Zdroj:<http://www.cornellsun.com/node/47587> a <http://www.hsi.gatech.edu/hrl-wiki/index.php/Kinect_Sensor_Mount>

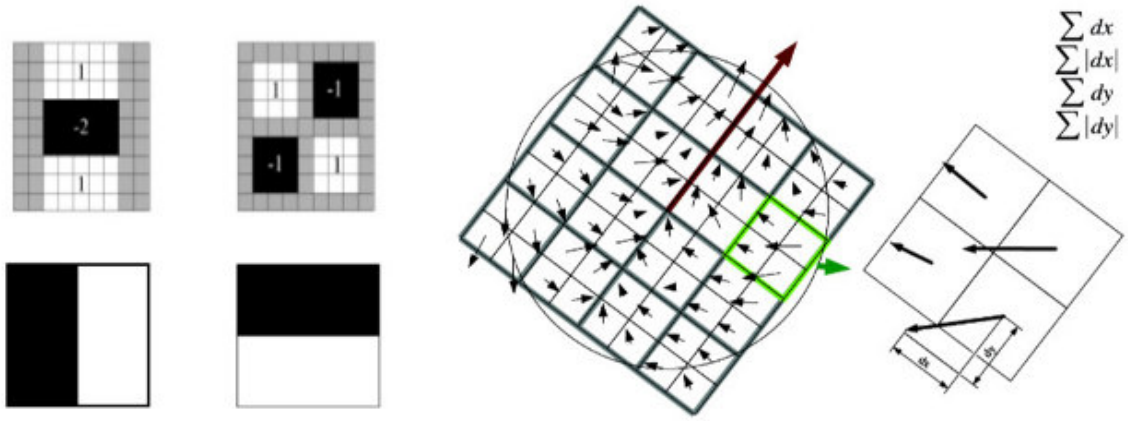
nov(Diffrence of Gaussian - DoG) na scale-space. Každému InP je priradená orientácia a mierka. Naznačenie detekcie a výpočtu deskriptorov InP je možné vidieť na obrázku 2.5. V každom bode okolia je vypočítaný gradient určený veľkosťou a orientáciou. Toto okolie sa rozdelí na podoblasti, v ktorých sa vypočíta histogram gradientov ktorý ma 8 binov. Gradient sa priradí práve jednému z binov podľa najmenšieho rozdielu uhlu. Autori SIFTu uvádzajú ako najlepšiu variantu použiť 16x16 pole vzorkov rozdelené do 4x4 poľa deskriptoru.

Speeded–Up Robust Features skratkou SURF[1], je metóda pre detekciu a popis významných bodov(InP) v obraze inšpirovaná SIFTom, taktiež invariantná voči mierke a rotácii. Ich hlavnou prioritou pre úpravu bolo urýchlenie výpočtu. Preto ako základ detekcie kľúčových bodov, realizovali pomocou aproximácie Hessianovej matice na integrálnom obraze umožňujúc rýchly výpočet konvolúcii s box-filtrov. Integrálny obraz je špeciálna reprezentácia obrazu, ktorá umožňuje vypočítať sumu intenzít vo výreze len štyrmi prístupmi do pamäte a troma operáciami nad nimi. Zachovanie nezávislosti na mierky, riešia použitím postupne zväčšujúceho sa filtra(9x9, 15x15, 21x21, 27x27) na rovnaký obrázok, čím dostavajú scale-space na ktorý, použijú rozdiel gaussianov. Kľúčové body sa nájdu ako pri SIFTe hľadaním lokálnych maxím. Orientácia sa urči výpočtom najlepšej odozvy Haarových vlniek smere x a y v kruhovej oblasti okolo bodu s polomerom $6s$ kde s je mierka v ktorej bol bod detekovaný. Výpočet deskriptoru zahajujeme vytvorením štvorcovej oblasti o hrane $20s$ okolo kľúčového bodu a jej natočení podľa orientácie. Región rozdelíme na 4x4 podregióny a pre každý určime odozvu Haarových vlniek oblasti 5x5. Každý podregión má 4-D vektor $\mathbf{v}$ získany $\mathbf{v} = (\sum d_x, \sum d_y, \sum |d_x|, \sum |d_y|)$ spojením týchto 4x4 podregiónov získame



Obrázok 2.5: SIFT: Tvorba scale–space, Tvorba deskriptoru[9].

výsledný deskriptor o 64 dimenziách. Ukážka Hessianovej matice a Haarových vlniek je na obrázku 2.6.

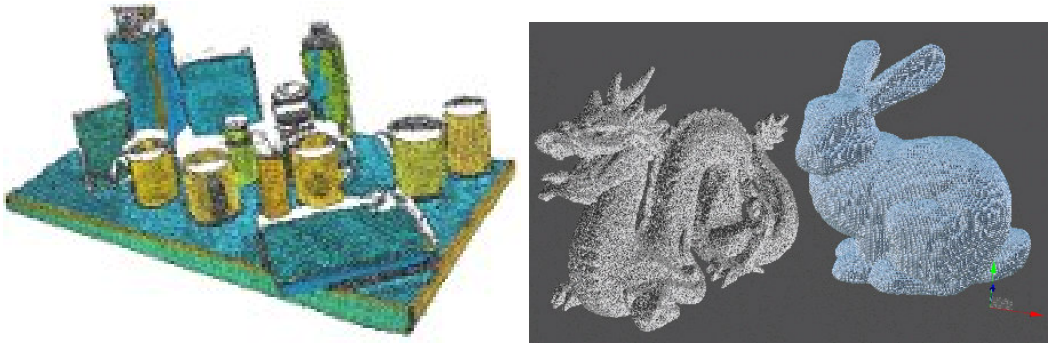


Obrázok 2.6: Aproximácia druhej derivácie Gaussovej funkcie v smere $y(Lyy)$ a $xy(Lxy)$,
Haarove vlnky pre výpočet odozvy v smere x a y,
Tvorba deskriptoru[1].

2.4 Mračno bodov

Mračno bodov, anglicky Point cloud[17] je množina bodov v 3D priestore, kde každý bod nesie informáciu o svojej polohe (súradnicu X, Y, Z) a typicky sa používa pre definíciu povrchu objektu. Pre vytvorenie takéhoto mračna sa najčastejšie používajú 3D skenery, ktoré dokážu vytvoriť veľmi rozsiahle a presné mračná bodov. Uplatňujú sa pri tvorbe 3D CAD modelov, v medicíne, v metrológii/kontroly kvality, vizualizácii, animácii a vo vykresľovaní počítačovej grafiky. Mračná bodov sa ale priamo nepoužívajú, prevedú sa na polygonálnu alebo trojuholníkovú sieť. Jediné odvetvie kde sa priamo používajú je kontrola kvality v priemysle a to tak, že mračno bodov vyrobenej časti sa porovná s jeho CAD modelom a vyhodnotia sa rozdiely.

Point Cloud Libery(PCL)[19] je open source multiplatformová knihovňa pre prácu s mračnami bodov. Zahrňuje algoritmy filtrovania, extrakciu príznakov, rekonštrukcie po-



Obrázok 2.7: Ukážka mračna bodov[19],
<http://www.3dvia.com>.

vrchu, registráciu, porovnávanie modelu a segmentáciu. Je vydávaná pod licenciou BSD, čo znamená že je voľne dostupná pre komerčné a vedecké účely. Implementačný jazyk je C++.

Kapitola 3

Existujúce metódy detekcie objektov s Kinectom

S príchodom Kinectu sa objavilo niekoľko metód a postupov využívajúcich pre detekciu obrazovú a hĺbkovú informáciu. Niekoľko z nich bude predstavených v tejto kapitole.

3.1 Metóda používajúca HOG

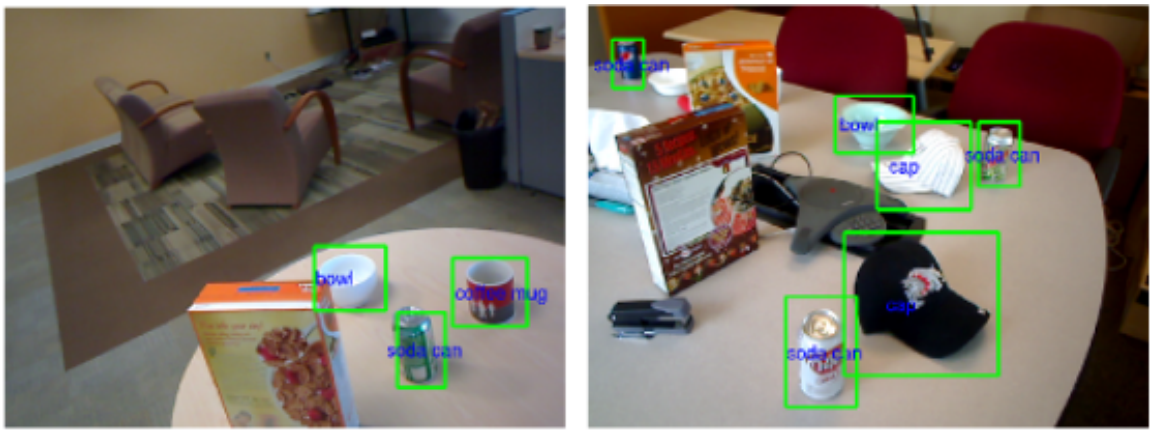
Metóda[8][7] popísaná v publikácii washingtonskej univerzite používa ako základnú myšlienku Histogram of Oriented Gradient[2], kde si túto metódu upravujú na základe článku[4]. Táto verzia zohľadňuje ako na kontrast citlivé tak necitlivé príznaky, kde orientácia gradientov v každej bunke(mriežka 8x8 pixlov) sú zakódované použitím 9 pri $0° - 180°$ a 18 pri $0° - 360°$ kvantizačných úrovní. Tým získajú 108–D vektor príznakov. Z toho sa však používa len 31–D. Prvých 27–D zodpovedá rozdielnej orientácii kanálov(18 citlivých a 9 necitlivých príznakov na kontrast). Posledne 4–D predstavujú celkovú energiu gradientu v štyroch blokoch o veľkosti 2x2 bunky. Taktiež ako v pôvodnej metóde sa použije pre klasifikáciu support vector machine(SVM).

Ako jadro klasifikátoru používajú *klasické detekčné okno*[2][4], kde systém vyhodnocuje hodnotiacu funkciu pre každú pozíciu a škálu obrazu a prahuje skóre pre obdržanie ohraničujúceho rámčeka. Detekčné okno má konštantnú veľkosť a prehľadáva 20 úrovni škálovej obrazovej pyramídy. Pre efektivitu je hodnotiaca funkcia lineárna.

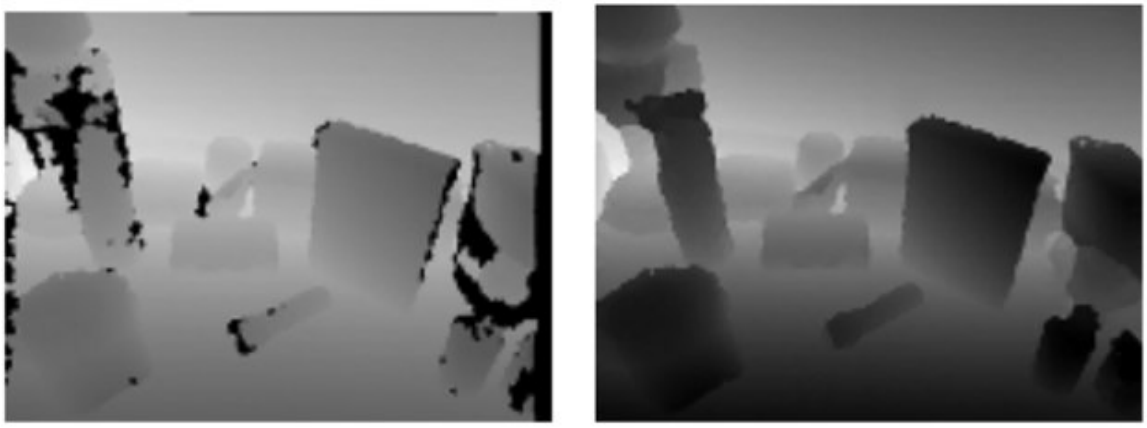
Výkonnosť klasifikátoru veľmi zavisí na dátach na ktorých bol trénovaný. S jedného obrázku sa da vygenerovať 10^5 negatívnych príkladov pre detekčné okno. Preto bol aplikovaný postup *hard negative mining*. Spočíva v tom, že pozitívne príklady je hľadaný objekt. Počiatočné negatívne príklady sú náhodne vybrané z množiny obrazov pozadia, alebo objektu patriacemu do inej triedy. Natrénovaný klasifikátor je použitý pre detekciu a sú vybrané falošne pozitívne(false positiv) prípady s najvyšším skóre(hard negatives). Hard negatives su zvolené ako negatívne príklady a klasifikátor sa natrénuje znovu. To sa opakuje 5x až obdržíme konečný klasifikátor. Ukážku detekcie je možné vidieť na obrázku 3.1.

Na hĺbkovej mape sa taktiež počítajú HOGi, ale je nutná predpríprava. Pretože hĺbková mapa nemá niektoré hodnoty definované, musí sa aplikovať rekurzívny medianový filter s mriežkou 5x5, aby tieto hodnoty dopočítal. Výsledok je možné vidieť na obrázku 3.2.

Dokážu taktiež určiť skutočné rozmery objektu. Ak d je vzdialenosť od kamery ktorá je nepriamo úmerná mierke o pre obraz v určitej mierke s môžeme napísať $c = \frac{o}{s}d$, kde c je konštanta. Pri konštantnom detekčnom okne môžeme taktiež o prehlásiť za konštantné. Tak



Obrázok 3.1: Ukážka detekcie
Vľavo: Viac tried, Vpravo: Jednej triedy s viacerými výskytmi[8].



Obrázok 3.2: Originálny obraz(vľavo) a upravený obraz použitím rekurzívneho medianového filtra(vpravo). Čierne pixle v ľavom obrázku sú chýbajúce hodnoty hĺbkovej mapy[8].

potom normalizovanú hĺbku môžeme vyjadriť ako $\frac{d}{s}$. Pre každý pixel v obraze je hodnota d konštantná, tak normalizovaný hĺbkový histogram môže vybrať správnu škálu obrázku z obrazovej pyramídy.

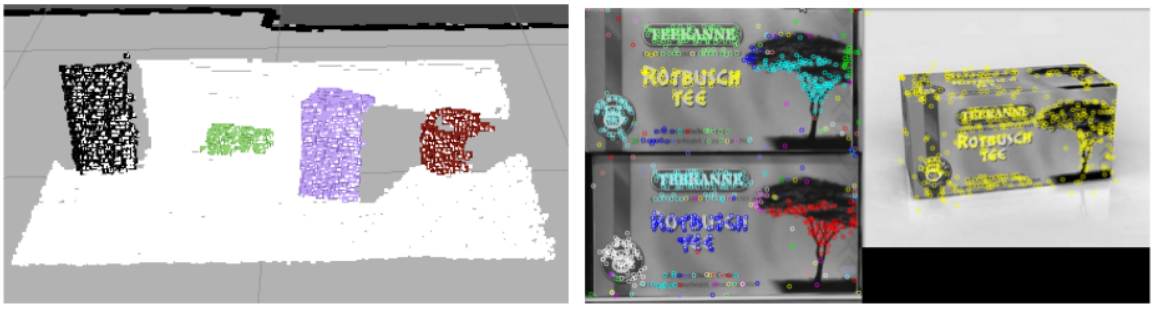
3.2 Metóda používajúca kľučové body

Metóda popísaná v publikácii[13] používa lokálne deskriptory bodov SIFT[9] zo slovníkovým stromom. Toto rozširuje o segmentáciu zaujímavých oblastií (obr. 3.3), v ktorých sa budú hľadať kandidáti detekcie:

Kombinácia 2D-3D: Detekuje horizontálne rovné plochy z dát Kinectu a určia sa zhluky bodov v tomto mračne bodov. Na tieto zaujímavé miesta sa pomocou back-projection nanesie uložený snímok.

Over-Segmentation-Based: Na začiatku sú nájdené SIFT kľúčové body na ktoré sa aplikuje algoritmus pre rast oblasti pre získanie odhadu zhlukov. Tento algoritmus začína

na bode, ktorý nepatrí žiadnemu zhluku a pridáva body, ktoré sú v preddefinovanom rozsahu r okolo originálneho bodu. Toto sa opakuje pre všetky novo pridané body. Výsledkom su silne texturované „ostrovy“ v obraze.



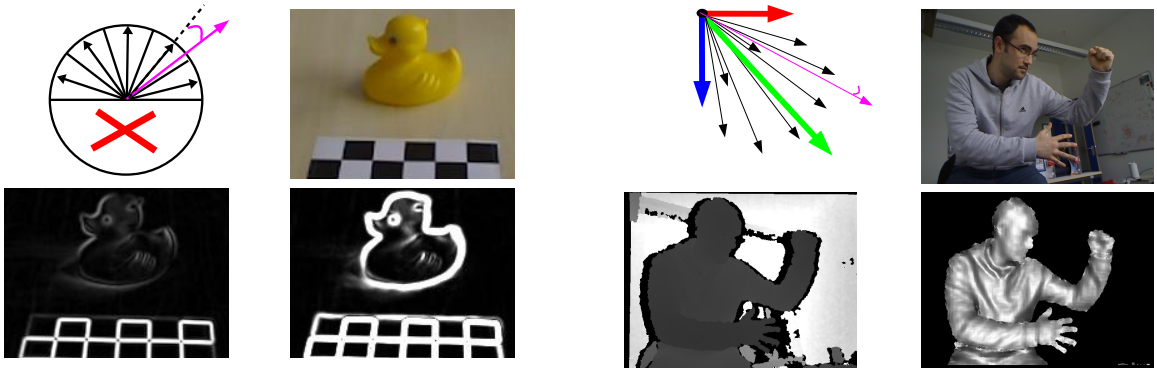
Obrázok 3.3: Vľavo: Zaujímavé oblasti s použitím back-projection z 3D bodov, Vpravo: Over-segmentation založenej na rastúcej oblasti[13].

V tomto prípade sa k detekcii pristupuje ako k problému hľadania dokumentu, čo podľa nich zlepšuje výsledky pri rôznych podmienkach osvetlenia a preplnenosti scény. Strom je zostavený tak, že jeho koreň je zložený zo všetkých príznakov v databáze. Listy su konkrétne objekty. Pre príznak *sd* ktorý chceme klasifikovať pomocou databázy môžeme veľmi rýchlo nájsť podobnú množinu deskriptorov, ktoré sú od seba minimálne vzdialené. Pre efektivitu sa neporovnáva deskriptor s celou množinou deskriptorov, ale len s jej ťažiskom.

3.3 Metóda používajúca Template matching

DOT(Dominant Orientation Templates) [6] je metóda vytvorená pre 3D detekciu objektov. Ako príznaky sa používajú gradienty, pretože majú vyššiu rozlišovaciu schopnosť ako iné príznaky a sú dostatočne robustné voči zmenám osvetlenia a šumu. Taktiež sú to príznaky, ktoré umožňujú detekovať objekty bez textúry.

Ich šablóna je zložená s n regiónov. Šablóna a región majú štvorcový tvar. Experimentálne určili ako najlepší región o hrane 7 pixlov.



Obrázok 3.4: **Ľavá 4-ica:** Ukážka práce s obrazom. **Pravá 4-ica:** Ukážka práce s hĺbkovou mapou[5].

V každom regióne určia najväčší gradient a ten podľa jeho orientácie priradia do jedného určitého smeru, čím rozdelia priestor do m zodpovedajúcich si oblastí. Taktiež si ukladajú

informáciu o regióne čo predstavuje uniformnú(jednotnú) oblasť. Ako m používajú 7, tak si môžu tieto informácie uložiť do 1 bytu: Jeden bit z $i \in 0 \ldots 6$ je nastavený na jedničku a $i = 7$ je nastavený na jedničku ak región predstavuje uniformnú oblasť. Takto uložené šablóny je možné veľmi rýchlo porovnávať s rovnako predpočítanými regiónmi zo vstupného obrazu pomocou logickej operácie AND. V článku taktiež uvádzajú, že pri použití SSE-inštrukcií dokáže systém porovnať 16 regiónov v 3 SSE operáciách a jedným prístupom do look-up tabuľky s 16 vstupmi.

Pri práci kde máme k dispozícií aj hĺbkovú mapu[5] sa pre zvýšenie robustnosti počítajú gradienty v každom farebnom kanáli obrázku samostatne a ako normalizovaný gradient sa použije ten s najväčšou hodnotou. Pre samotnú prácu s hĺbkovou mapou používajú kvantované normály povrchu. Tie umožňujú reprezentovať objekty umiestnené blízko aj ďaleko od kamery, zatiaľ čo ich štruktúra je zachovaná. Kvantovanie prebieha porovnávaním uhlov medzi získanou normálou a predpočitanými vektormi, ktoré sú usporiadané do kužeľa vrcholom smerujúcim ku kamere. Ukážku je možné vidieť na obrázku 3.4.

Z metódy DOT vychádza aj metóda BiGG[12].

Kapitola 4

RoboEarth

V tejto kapitole bude predstavený nástroj RoboEarth a to jeho inštalácia a konfigurácia. Taktiež tu bude predstavená časť RoboEarthu, ktorá umožňuje vytvárať 3D model objektu. S jeho pomocou dokážeme natrénovať detektor RoboEarthu, čo bude taktiež v tejto kapitole predstavené.

Obrázok 4.1: Logo projektu RoboEarth[3].

4.1 O projekte

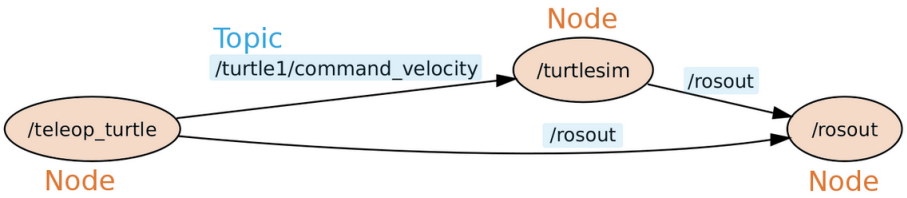
RoboEarth[15] ako projekt, ktorého hlavnou myšlienkou je postaviť „internet pre robotov“, cez ktorý budú medzi sebou zdielať informácie a učiť sa o chovaní a prostredí iných robotov, pod heslom „Skúsenosť je najlepší učiteľ“. RoboEarth poskytuje kompletnú infraštruktúru[3] pre cloud robotiku, ako databázu dostupnú z internetu a nástroje pre prácu s ňou. V tejto databáze je možné udržovať informácie pre rozpoznávanie objektov (obrázky, modely objektov), navigáciu (mapy, modely sveta), úlohy (popis akcií a stratégií pre manipuláciu) a prídavné informácie (anotácie obrazu, štruktúry z offline učenia). Projekt zatiaľ prechádza búrlivým vývojom. Systém je implementovaný v jazyku C++. RoboEarth bol prispôsobený aj pre ROS[10], ktorý je opísaný v kap. 4.2.

4.2 ROS.org

ROS (Robot Operating System)[18] je sada nástrojov a knižníc pre tvorbu robotických aplikácií od spoločnosti Willow Garage. Umožňuje simulácie hardware, ovládačov pre zariadenia, knižnice, vizualizácie a mnoho viac. Hlavné programovacie jazyky sú C++ a Python. ROS je vydávaný ako open source pod licenciou BSD.

Základnou organizačnou jednotkou ROS je *balíček* (package), ten môže obsahovať: knižnicu, nástroj, spustiteľný binárny súbor atď. Každý balíček musí obsahovať súbor `manifest.xml` ,ktorý obsahuje popis balíčku a jeho závislosti. Množina balíčkov ROS sa nazýva *stack*. Stack taktiež obsahuje svoj súbor pre popis ako balíček, ale s názvom `stack.xml`.

Koncept chovania ROS obsahuje *uzly* (Nodes), *témy*(Topics) a *správy*(Messages). Uzol môžeme chápať ako jeden spustený program, ktorý vykonáva požadovanú činnosť. Témy sú ako zbernice dát. Uzol môže z nich prijímať alebo do nich odosielať dáta vo forme správ. Grafické vyjadrenie komunikácie je na obrázku 4.2.



Obrázok 4.2: Komunikácia v ROS[18].

ROS obsahuje aj niekoľko balíčkov pre rozpoznávanie v obraze. Bohužiaľ sa ale tieto balíčky dnes už neudržujú, ale spoločnosť Willow Garage vyvíja nový systém s názvom Ecto, v ktorom sa nachádza sekcia pre rozpoznávanie.

4.3 Inštalácia RoboEarth

Samotnú inštaláciu zahájime kontrolou svojho operačného systému a ROSu. ROS oficiálne podporuje len operačný systém Ubuntu a to v rozsahu jeho verzii 10.04 do 11.10[1]. Na ostatné systémy je ho možné taktiež nainštalovať, ale len zo zdrojových kódov. ROS má aktuálnu verziu s názvom „ROS Fuerte“ vydanú 23. apríla 2012, ktorá je vývojármi doporučená inštalovať a používať, avšak momentálne ešte nie sú pripravené podporné baličky používané RoboEarth. Preto je nutné instalovať nižšiu verziu „ROS Electric Emys“ vydanú 30. augusta 2011. RoboEarth je oficiálne podporovaný len pre Ubuntu 10.10. Ja som ale exterimentálne zistil, že systém funguje aj pre Ubuntu 11.04 a 11.10 bez rozdielu. Návod pre inštaláciu ROS je možné najsť na jeho stránke[2].

Pre stiahnutie RoboEarth je potrebné mať inštalovaný nástroj `rosinstall`, ktorý slúži ako verzovací systém pre ROSu.

Kód 4.1: Stiahnutie RoboEarth

```
$ rosinstall ~/ros_workspace/roboearth /opt/ros/electric 'http
   ://www.ros.org/wiki/roboearth?action=AttachFile&do=get&
   target=roboearth.rosinstall'
$ source ~/ros_workspace/roboearth/setup.bash
```

Cesta `/opt/ros/electric` určuje adresár, kde sa nachádza ROS. Cesta `~/ros_workspace/roboearth` určuje miesto, kde chceme nainštalovať RoboEarth. Príkaz `source` sa použije pre nainicializovanie premenných v aktuálnom termináli. Tento príkaz je vhodné vložiť do `~/.bashrc` alebo `~/.bash_login`, aby sme nemuseli inicializovať ručne pre každý novo otvorený terminál. Predtým je ešte nutné nainštalovať niekoľko podporných balíčkov ROS: Taktiež je nutné skontrolovať, či máme nainštalovaný balíček predstavujúci ovládače pre Kinect s názvom `ros-electric-openni-kinect`.

[1] Údaj zo dňa 24.3.2011

[2] <www.ros.org/wiki/electric/Installation/Ubuntu>

Kód 4.2: Inštalácia podpory pre RoboEarth

```
sudo apt-get install ros-electric-ias-common ros-electric-
   perception-pcl-addons ros-electric-simulator-gazebo ros-
   electric-vision-opencv ros-electric-octomap-mapping
```

U mňa nastala chyba „zamrznutia“ pri sťahovaní. Na tento problém existuje riešenie na stránkach RoboEarth. Prijať certifikát od *ipvs.informatik.uni-stuttgart.de*

Kód 4.3: Stiahnutie certifikátu

```
$ svn info https://ipvs.informatik.uni-stuttgart.de/roboearth/repos/
   public/tags/latest
Press 'p'
```

Ak všetko prebehlo v poriadku prejdeme k samotnej kompilácii RoboEarth a jeho závislosti. Použijeme na to nástroj ROSu `rosmake`, pri ktorom je nutné aby všetky závislé balíčky boli umiestnené v adresároch ktoré ROS používa. Príkaz je jednoduchý:

```
$ rosmake roboearth
```

Aj pri kompilácii nastala chyba, ktorá taktiež mala svoje riešenie na stránke RoboEarth:

Kód 4.4: Chyba v balíčku Artoolkit

```
../../../include/AR/sys/videoLinuxV4L.h:24:28: fatal error:
   linux/videodev.h: No such file or directory

$ roscd artoolkit
$ sed -i 's,#include␣<linux/videodev.h>,#include␣<libv4l1-
   videodev.h>,g' /build/artoolkit/include/AR/sys/
   videoLinuxV4L.h
$ sed -i 's,#include␣<linux/videodev.h>,#include␣<libv4l1-
   videodev.h>,g' /build/artoolkit/lib/SRC/VideoLinuxV4L/video
   .c
```

Ak všetko prešlo bez problémov, máme nainštalovaný RoboEarth.

4.4 Tvorba 3D modelu v RoboEarth

Model sa v RoboEarth skladá zo sady mračien bodov. Na začiatku je nutné spustiť všetky uzly potrebné pre tvorbu 3D modelu. Ovládač pre Kinect, uzol pre zostavenie súradnicového priestoru a GUI. Je treba poznamenať, že RoboEarth obsahuje aj uzol pre nahrávanie a sťahovanie modelu on-line, ale v mojom prípade budem uvažovať len o modeloch uložených a ukladaných lokálne. **Príprava:** Vytlačíme si podložku so značkami, ktorú je možné nájsť

na stránkach RoboEarth[3] a skompletujeme ju. Je nutné, aby hrana čierneho štvorca mala 80 mm a logo RoboEarth bolo v strede. Položíme objekt na logo. Objekt nesmie zakrývať žiadnu časť čiernych štvorcov. Kinect umiestnime asi 60 cm od objektu, asi s náklonom 30° smerom dole od horizontu. Kinect by mal byť umiestnený staticky, aby sa nemenili svetelné podmienky. Moja konfigurácia pri nahrávaní je na obrázku 4.3. Model samotný sa skladá zo sady binárnych súborov s príponou .pcd, z nich každý obsahuje jedno mračno bodov a taktiež farby jednotlivých bodov a súboru `meta.xml`, ktorý obsahuje meno modelu, jeho typ a počet súborov s mračnami bodov z ktorých model pozostáva. Obsahuje aj položku merítka, ale tá sa v súčasnej implementácii nepoužíva.

Obrázok 4.3: Nahrávacia platforma.

Ovládač pre Kinect:

```
$ roslaunch openni_launch openni.launch
```

Slúži pre komunikáciu s Kinectom a prevádza surové RGB/IR prúdy na RGB a hĺbkový obraz, rozdielový obraz a mračno bodov. Toto mračno bodov sa používa na vstupe RoboEarth.

Ar_bounding_box:

```
$ rosrun ar_bounding_box ar_kinect
```

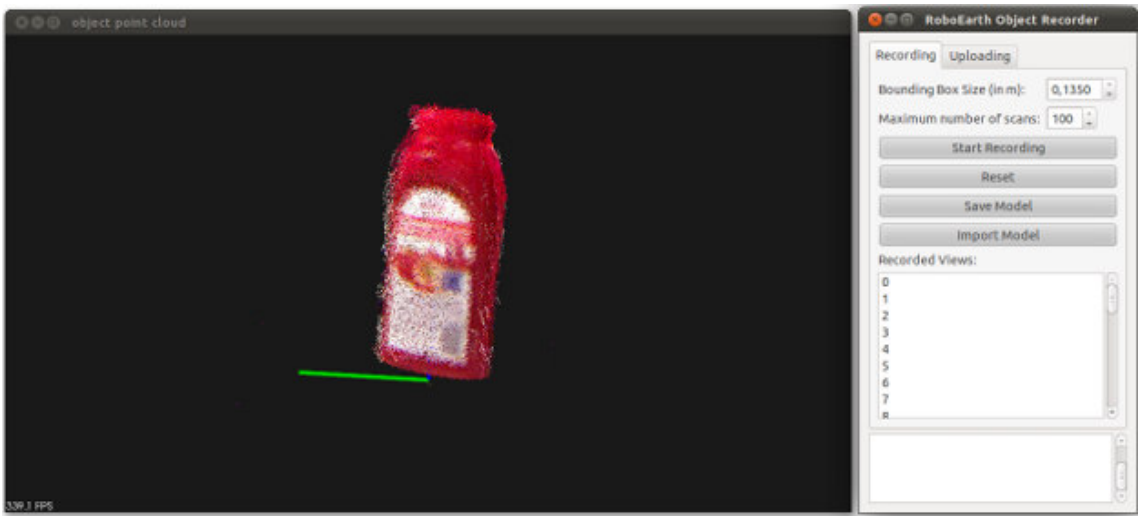
Ar_bounding_box slúži na detekciu sady vzorov a nasledné vytvorenie súradnicového systému a jeho ohraničenie. S jeho pomocou je možné ohraničiť snímané mračno bodov, určiť jeho umiestnenie v súradnicovom systéme a nakoniec túto informáciu poslať samotnému recordéru. Táto časť je založená na knižnici ARToolKit, ktorá sa používa pre tvorbu aplikácie rozšírenej reality. Rieši problém určenia pozície a orientácie kamery pomocou markerov v reálnom čase.

GUI:

```
$ rosrun re_object_recorder record_gui
```

[3] Podložka zo značkami <http://www.ros.org/wiki/re_object_recorder?action=AttachFile&do=get&target=marker_template.pdf>

S jeho pomocou je možné vytvoriť, upraviť, uložiť alebo odoslať na server RoboEarth model objektu. Po spustení a načítaní modelu uvidíme okno vizualizácie a ovládací panel (obr. 4.4). *Poznámka:* Pri spustení môžeme dostať okno zo správou: „*Depth registration*



Obrázok 4.4: GUI rekordéru.

is disabled in openi driver. This will probably lead to badly aligned pointcloud“. To sa dá vyriešiť tak, že si spustíme:

```
$ rosrun dynamic_reconfigure reconfigure_gui
```

a nájdeme položku `/camera/driver` a tam zaškrtneme checkbox *depth_registration*. Po reštarte GUI recorderu by malo byť všetko v poriadku.

Tlačidlá na karte *Recording* :

Start Recording Spustí nahrávanie mračien bodov. Každé mračno má svoje číslo v *Recorder Views* a je vizualizovaný v okne „object point cloud“. Týmto tlačídlom sa taktiež nahrávanie zastavuje.

Reset Vymaže doteraz nahrané mračná.

Save Model Vytvorí z nahraných mračien model a ten uloží na disk. Tu je ale nutné podotknúť, že aby bolo možné model uložiť, musia byť na karte *Uploading* vyplnené náležitosti *Object Name* a *Object Class*. Napríklad pre hrnček, máme triedu „*cup*“ a jeho názov „*RedCup*“. Názvy musia byť bez medzier (bielych znakov).

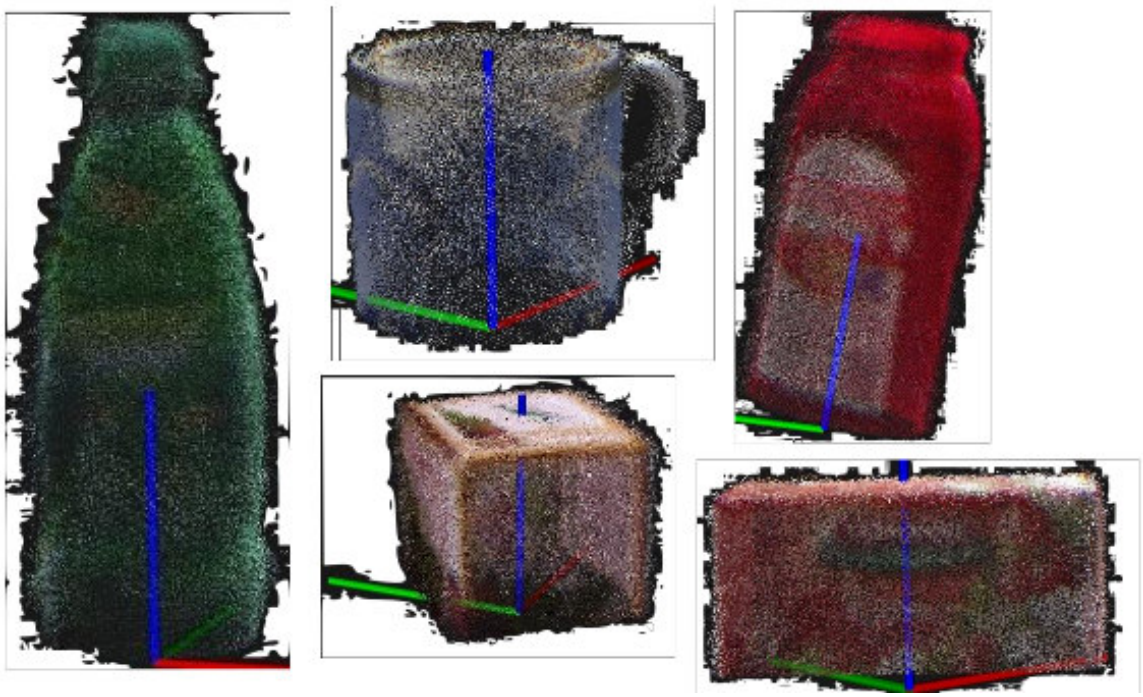
Import Model Importuje model z lokálneho disku do recordera. Je treba zvoliť adresár v ktorom sa nachadzajú súbory s mračnami a `meta.xml`.

Hodnotu položky *Bounding Box Size(in m)* nie je vhodné meniť, pretože parameter je štandardne nastavený pre podložku z RoboEarth. Parameter *Maximum number of scans*. je štandardne nastavený na 100, čo je dostačujúce. Podľa mojich experimetov je vhodné voliť hodnotu v rozsahu od 30 do 70, podľa toho či skenujeme celý objekt alebo len jeho časť.

Nahrávanie prebieha tak, že podložku pomaly otáčame s objektom v jej strede (okolo osy kolmej k povrchu podložky). Zároveň môžeme sledovať vizualizované mračná bodov a ako

sa z nich postupne rysuje objekt. To je vhodné priebežne sledovať z dôvodu, že nahraný povrch nepatrí vždy len objektu ale aj podložke. To je možné skorigovať tak, že v poli *Recorded Views* nájdeme chybné mračno bodov. Po kliknutí na akékoľvek mračno sa jeho body zvýraznia na červeno a klávesou *Delete* je možné toto mračno bodov odstrániť. Po odhadnutí z vizualizácie, že model je už pre naše účely dostačujúci zastavíme nahrávanie. Ak máme vyplnené údaje o modele, uložíme si ho. Tak sa vytvoria všetky pcd–súbory s mračnami bodov a `meta.xml`.

Pri nahrávaní sa vyskytli taktiež chyby. Najčastejšia „*distance X/X too large: XXXX* “, čo znamená, že Ar_Kinect nebol schopný nájsť 3 štvorce pre určenie súradnicového systému. Tým, že som zvýšil osvetlenie scény sa mi podarilo uvedenú chybu odstrániť.



Obrázok 4.5: Výsledky tvorby 3D modelov.

4.5 Detekcia objektov v RoboEarth

Detekcia v RoboEarth je prevádzkovaná pomocou 3 uzlov, ktoré je treba spustiť. Samotnú detekciu ale prevádza len jeden, ostatné sprostredkovávajú vstupné dáta a vykresľujú výsledky detekcie. Názov tohoto uzlu v ROSe je `re_kinect_object`. Skladá sa z 3 zdrojových súborov.

slam_main obsahuje hlavnú triedu `ROSCom` s metódami `model_path_cb` a `kinect_cb`, ktoré reagujú na prijatie ROS_správy.

recognitionmodel je zložený z hlavičkového a zdrojového súboru. Nachádzajú sa tu metódy pre zabezpečenie priebehu samotnej detekcie, úlohu tu zohráva `matchAspects` pre model a samotný aspekt[4], `findCorrespondences`, `naiveNearestNeighbor`, `compareSURFDescriptors` a ďalšie

[4] Aspekt je v RoboEarth-e označenie, pre jeden pohľad na objekt, obsahuje jedno mračno bodov.

`kinect_cb` je metóda, ktorá sa zavolá pri obdržaní mračna bodov scény. Metóda volá funkcie pre výpočet zhody načítaných modelov a výsledky o nich posiela uzlu `re_object_detector_gui`, ktorý vizualizuje výsledky.

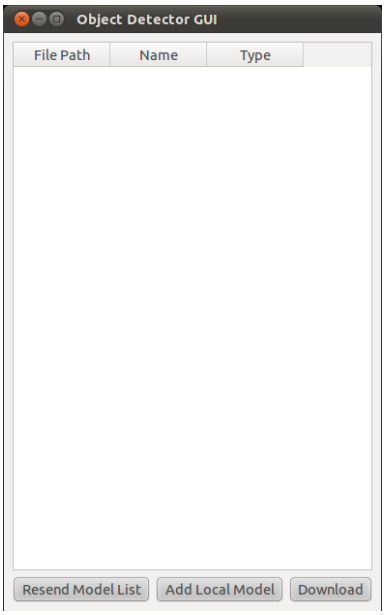
Tlačidlá okna *Object Detector GUI* :

Resend Model List slúži pre opätovné zaslanie informácie o modeloch, ktoré boli cez užívateľské rozhranie zadané.

Add Local Model sa používa pre načítanie nového modelu pre natrénovanie z lokálneho disku. Po zvolení sa jeho meno objaví v zozname a získané informácie sa pošlú uzlu `re_kinect`.

Download pomocou neho je možné stiahnuť a načítať modely zo serverov RoboEarth.

Po spustení uzlu `re_object_detector_gui` sa nám zobrazí okno na obrázku 4.6. Po vybratí zodpovedajúceho 3D modelu sa začnú načítať .pcd súbory podľa `meta.xml`. U každého pohľadu prebehne detekcia a popis kľúčových bodov. Tie sa uložia a súčasne namapujú pomocou 3D informácie do priestoru z 2D. Načítaný model je uložený v štruktúre `RecognitionModel` a obsahuje vektor štruktúr `ObjectAspect`, ktorého položky sú jednotlivé pohľady. V ďalšej kapitole popíšeme jadro detekcie.



Obrázok 4.6: GUI detektoru.

Samotná detekcia začína v metóde `kinect_cb`, ktorá sa spustí pri obdržaní obrazu scény mračnom bodov. Na začiatok vykoná extrakciu 2D snímku z mračna, na ktorom sú vypočítané kľúčové body SURFu. Tieto body sa namapujú z 2D priestoru do 3D priestoru, podľa umiestenia kľúčových bodov. Potom sa vyberie prvý uložený model, ktorý použije svoju metódu `matchAspects` na zistenie či jeden z jeho aspektov nie je zhodný s aspektom scény. Samotný aspekt modelu má tiež metódu `matchAspects`, ktorá tieto aspekty porovná. A to tak, že pomocou metódy `findCorrespondences` zistíme sadu párov

odpovedajúcich si bodov. Korešpondencie sa hľadajú algoritmom najbližšieho suseda (k-ANN) $k = 2$, ktorý fakticky použije len toho lepšieho z dvojice. Implementácia je v metóde `naiveNearestNeighbor` a tá fyzicky porovnáva deskriptory pomocou `compareSURFDescriptors`. Ak máme zistené tieto dvojice korešpondujúcich si deskriptorov, začnú sa hľadať 3 zodpovedajúce si korešpondencie, ktoré musia spĺnať pravidlo, že vzdialenosti korešpondencií v aspekte modelu musia byť väčšie ako hodnota parametra `DISTANCE_TRASHOLD` a vzdialenosť medzi korešpondenciami menšia ako hodnota `EPSILON`. Vzdialenosť je počítaná ako Euklidová. Ak všetko prebehlo správne nakoniec sa určí transformácia aspektu scény oproti aspektu modelu.

Kapitola 5

Návrh riešenia a experimentov

V návrhu môjho riešenia som uvažoval o detektore, ktorý by používal lokálne príznaky popisujúce významné body v obraze. Zvažoval som SIFT a SURF. Pretože autori SURFu sľubujú vyššiu rýchlosť rozhodol som sa pre neho. Otázku ako reprezentovať dáta z Kinectu za mňa vhodne vyriešil on sám. Presnejšie knihovňa PCL, ktorá poskytuje všetko potrebné. Ako trénovacie dáta by sa mohol zostaviť 3D model objektu, reprezentovaný množinou kľúčových bodov. Detekcia by prebiehala ako hľadanie odpovedajúcich si príznakov model vs. scéna, ktoré ležia rovnako geometricky rozmiestnené.

Pri hľadaní ďalších informácií som narazil na systém pripomínajúci navrhovaný. Jeho meno bolo **RoboEarth**. Model objektu nie je zložitý a jeho získanie zahŕňa rýchlu a jednoduchú procedúru. Vďaka tomu je možne prevádzať detekciu zo všetkých strán bez rozdielu. Preto som sa rozhodol použiť ho ako základ pre moju prácu, overiť jeho vlastnosti pomocou experimentov.

5.1 Voľba objektov

Pre moje experimenty ako testovacie objekty som použil 5 objektov, ktoré je možné vidieť na obrázku 5.1. Zvolil som si ich náhodne, aby som pokryl triedu objektov krabička, fľaška a šálka, pretože vývojári RoboEartha doporučujú jeho použitie na detekciu objektov bežného života. Každý z týchto objektov bol samostatne nahrávaný pomocou rekordéra z RoboEarthu.

5.2 Parametre detektoru

Dva parametre, ktoré je možné v detektore meniť sú `EPSILON` a `DISTANCE_TRASHOLD`. V mojich experimentoch som menil len parameter `EPSILON`, pretože som chcel zistiť citlivosť korešpondencií medzi modelom a scénou. V základnej inštalácii je parameter nastavený na hodnotu *0.001*, ktorú som pri mojich experimentoch len zvyšoval v rozsahu *0.001* do *0.1*.

5.3 Scény

Experimentálne scény boli dve. Jedna predstavovala jednoduchú scénu, v ktorej neboli rušivé efekty ako nerovnosti a iné objekty. Budem ju nazývať *scéna1*. Druhá bola preplnená, nachádzali sa v nej aj iné objekty. V tejto scéne sa muselo skúmať viac kľúčových bodov, na rozdiel od prvej, kde kľúčové body vznikali hlavne na povrchu sledovaného objektu. Budem

Obrázok 5.1: Objekty. Zľava: balzam, hrnček, čaj, kečup, ľadový čaj.

ju nazývať *scéna2*. U oboch scén bol Kinect umiestnený 60 cm od objektu a vo výške 10 cm od vodorovnej plochy, na ktorej bol objekt umiestnený. S Kinectom sa pri experimentovaní nehýbalo. Pohybovalo sa len s objektami, ktoré som skúmal. Obe scény boli taktiež umelo prisvetľované. Je treba podotknúť, že na všetkých snímkoch sa natáčaný objekt nachádzal, neodstraňoval som ho zo scény ani som ho vôbec nezakrýval.

Detektor bol otestovaný ako si poradí s jedným z objektov v každej zo scén, dvojicou objektov a zo všetkými piatimi objektami v *scéne1*.

Obrázok 5.2: Scéna1, Scéna2.

5.4 Prostredie

Všetky experimenty som realizoval na konfigurácii: CPU Intel Core i5-2500K, RAM 16GB, GPU Sapphire Radeon HD 6870 1GB, OS Ubuntu 11.04(64bit) s ROS Electric.

5.5 Nahrávanie scény

Nahrávanie prebiehalo pomocou nástroja v ROSe s názvom `rosbag`. Tento program produkuje bag-súbor, ktorý je vlastne sada správ ROSu. Nahrávanie prebiehalo v komprimovanej podobe takže 70 sekúnd bežiaci bag-súbor má približne 221 MB, namiesto 1 GB bez komprimácie. Nahrával som len rostopic `camera/depth_registered/points`, ten predstavuje spojenie RGB snímku a mračna bodov. Počet nahraných snímkov bolo od 72 do 890 na jeden bag-súbor podľa toho, ako rýchlo sa mi podarilo objektom otočiť. Počítal som 4 sekundy na jedno natočenie. Otáčanie som vykonával manuálne. Pomocou `rosbag` bola aj spätne spúšťaná sada snímkov ako vstup pre detektor.

```
$ rosbag record -j -O <bag_súbor> "camera/depth_registered/points"
$ rosbag play <bag_súbor>
```

5.6 Anotácia vstupných dát

Pre vyhodnotenie správnosti detekcie bolo nutné vstupné dáta anotovať. Pre tento účel som si zvolil `viperGT`[1]. Najskôr som však spustil detektor na „nečisto“ pre získanie snímkov ktoré do neho vstupujú. Tieto snímky som pomocou programu `mencoder` previedol na video súbor. Ten však nebol ešte vhodný pre anotáciu, preto bol následne konvertovaný v programe `Avidemux`. Po anotácii `viperGT` vracia súbor v XML formáte, z ktorého pre moje účely som použil len výčet súradníc ohraničujúcich rámčekov. Tie som uložil do súborov formátu názvu `objN_<meno_objektu>.map`, kde N je číslovanie objektov v scéne.

5.7 Úpravy Roboearth

Aby bolo možné vyhodnotiť tieto experimenty, bolo nutné upraviť alebo pridať do RoboEarthu niektoré časti.

Základná inštalácia RoboEarth informáciu o dobe detekcie neposkytuje, preto ju bolo nutné doplniť. V mojom prípade je to meraný čas, doba od obdržania mračna bodov scény do poslania výsledkov uzlu, zabezpečujúceho zobrazenie výsledkov. Použil som triedu `ros::Time` ktorá sa nachádza v ROSe. Po behu detekcie je vytvorený súbor `duration.txt` ktorý na každom riadku obsahuje dobu detekcie pre jeden snímok v sekundách. Podobná konštrukcia bola zvolaná aj pri meraní času výpočtu SURFu a porovnávaní modelu.

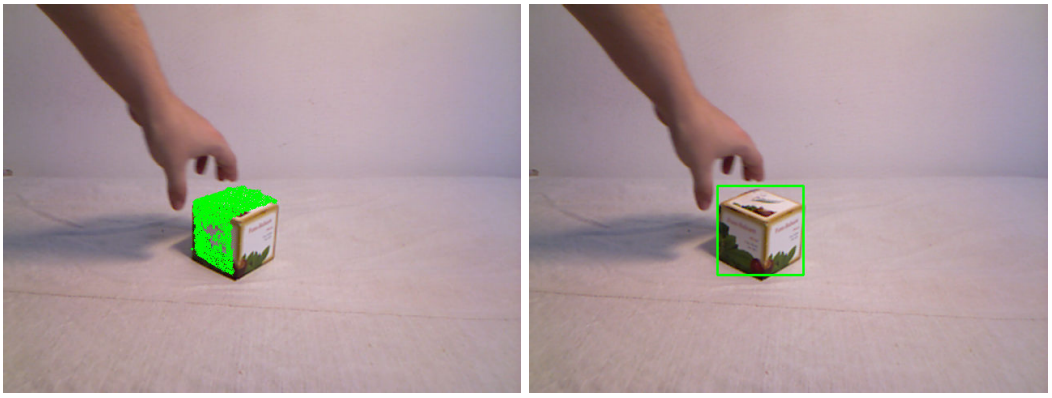
Pozíciu detekcie RoboEarth v základnej inštalácii určuje len vizuálne, čo je vlastne len ofarbenie časti scény zodpovedajúce aspektu detekovaného modelu, ktorý je transformovaný do scény. RoboEarth obsahuje časť, ktorá výsledný snímok ofarbí. Túto časť som využil tak, že si vypočítavam minimum a maximum z množiny bodov pre ofarbovanie a z týchto výsledkov pomocou funkcie z OpenCV `cv::line` zostavím línie ohraničujúceho rámčeka, ktorý nanesiem do obrazu scény. Viac obrázok 5.3. Ďalej si pre účely vizuálneho overenia ukladám obrázky výstupu detektoru. Po spracovaní jednej sady testovaných snímkov obdržíme súbor `det.txt`, ktorý obsahuje informácie o detekcii. Štruktúra každého riadku je:

cislo_snimku *meno_objektu* *x_suradnica* *y_suradnica* *dlzka* *sirka*

[1] ViperGT je dostupný na <http://viper-toolkit.sourceforge.net/>

x_suradnica,y_suradnica – je ľava horná súradnica ohraničujúceho rámčeka

Nakoniec v adresári `xout` je uložená sada výstupných snímkov z detektoru vo formáte názvu *OUT_NNN.png* kde *N* je čislica od 0 – 9.



Obrázok 5.3: Vľavo: Pôvodné označenie detekovaného objektu.
Vpravo: Označenie detekovaného objektu po mojej úprave.

O načítanie objektu sa stará uzol `re_kinect_object_detector` podľa informácie, ktorú mu pošle `re_object_detector_gui` pomocou ROS–správy. Pre moje účely som potreboval aby sa model načítaval automaticky po spustení uzlu, preto som metódu `model_path_cb` triedy `ROSCom`, ktorá slúži pre načítanie pozmenil tak, aby neprijímala ako parameter ROS–správu, ale ako obyčajný textový reťazec(`std::string`). Odstránil som jej registráciu ako odberateľa týchto správ a jej volanie umiestnil do konštruktoru `ROSCom`. Ako jej parameter som uviedol konštantný reťazec predstavujúci absolútnu cestu k modelu v mojom systéme.

5.8 Proces experimentu

Experiment začíname tak, že spustíme kolekciu ros-uzlov `roscore`. Zvolíme hodnotu parametru `EPSILON` a skompilujeme systém. Následne spustíme ROS–uzly pre detekciu okrem uzla predstavujúceho ovládač Kinectu. Počkáme až sa detektor natrénuje a spustíme `rosbag` s príslušním bag-súborom. Po jeho skončení máme súbory `det.txt` a `duration.txt` a zložku `xout` naplnenú snímkami z výstupu detektoru.

5.9 Proces vyhodnocovania experimentu

Vyhodnotenie som vykonával pomocou skriptov napísaných v jazyku Python a BASH. Pre získanie doby detekcie použijeme skript `countTime.py` takto:

```
$ ./countTime.py <subor_formátu_duration.txt>
```

Pretože v súbore `det.txt` sa môžu nachádzať výsledky pre viaceré objekty bolo prevedené rozdelenie na špecifické det-súbory `det_<meno objektu>` a to podľa mena objektu. Túto operáciu prevádza `separeDet.py`.

```
$ ./separeDet.py <súbor_formatu_det.txt>
```

Pre samotné vyhodnotenie úspešnosti detekcie použijeme `checkAnnotation.py`:

```
$ ./checkAnnotation.py <det_súbor_jedneho_objektu.txt> <xml subor anotacie>
```

Skript `checkAnnotation.py` načíta detekčný súbor a anotačný súbor. Hľadajú sa rovnaké čísla snímkov, v ktorých prebehla detekcia a boli anotované. Porovnajú sa ohraničujúceho rámčeky a ich percentuálne prekrytie. Ak prekrytie dosiahne viac ako 80% prehlásime, že detekcia je korektná a zvýšime počítadlo true positive. V prípadoch, že je prekrytie menšie zvýšime počítadlo false positiv. Skript nepočíta počet výskytov false negativ ale udržuje si celkový počet anotovaných snímkov, čo predstavuje súčet true positive a false negative. Hodnotiace parametre detekcie tj. *presnosť* a *odozva* vypočítame pomocou rovníc (5.1) a (5.2):

$$presnost = \frac{True\_positive}{True\_positive + False\_positive} \tag{5.1}$$

$$odozva = \frac{True\_positive}{True\_positive + False\_negative} \tag{5.2}$$

Aby som vykonávanie aspoň čiastočne automatizoval vytvoril som skripty `one_exp.sh` a `one_exp_multi.sh`, ktoré zahrňujú kompiláciu RoboEarthu, natrénovanie a spustenie detekcie. Na koniec, výpis vyhodnotenia tohoto behu detekcie. Pre jeden beh experimentu je nutné zvoliť parameter `EPSILON` skontrolovať *meno modelu*, *cestu k súboru s anotáciou objektu* a správnosť *bag-súboru*.

Po skončení skriptu mame adresár s menom parametru `EPSILON` v zložke spustenia skriptu, v ktorom sú všetky výsledky daného experimentu a taktiež nám do terminálu vypíše náležitosti: *true positive*, *false positive*, *celkový počet anotovaných snímkov*, *priemerná doba detekcie*, *odozvu* a *presnosť*.

Kapitola 6

Výsledky

Po vykonaní 20 cyklov experimentu každého modelu(modelov), bolo možné zostaviť grafy 6.1, 6.2 a 6.3. Ako môžeme vidieť detektor v *scéne1* dosahoval vysokú presnosť (až 80%). *Odozva* sa pohybovala okolo 50% zo všetkých možných prípadov detekcie.

U *scény2* je vidieť, že len objekt kečup a balzam dosiahli nad 80% *presnosti*, dokonca v prípade objektu *kečupu* bola lepšia ako v *scéne1*. Objekt *ľadový čaj* nedosiahol ani 10% presnosti a *odozvy*, dokonca objekt *šálka* nebol vôbec detekovaný. Všeobecne taktiež poklesla odozva u všetkých objektov.

Detekcia viacerých objektov dosahovala nadpolovičnú *presnosť* dokonca pri spojení objektov *kečup* s *čajom* až 80%.

Experimentálne som tiež určil, že každý objekt vykazoval najlepšie výsledky(*presnosť* a *odozvu*) pri inej hodnote parametru `EPSILON`.

Objekty[1]:

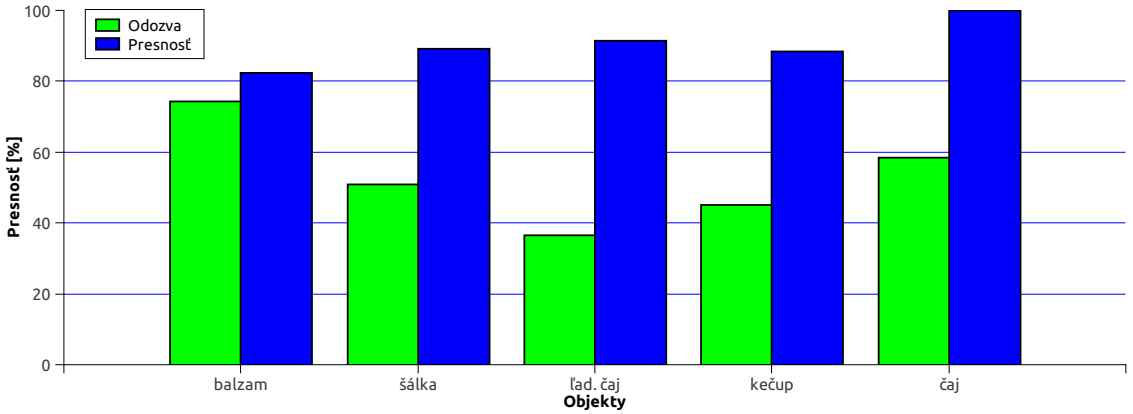
- **Balzam** : 0.02, 0.03
- **Kečup** : 0.009, 0.015
- **Čaj** : 0.04, 0.09
- **Ľad. čaj** 0.2 : 0.04
- **Šálka** : 0.1, X

Je potrebné podotknúť, že pri objekte *kečup* nastávala detekcia len z jeho prednej strany kde má etiketu, zadná strana a boky sú uniformné, takže detektor kľúčových bodov SURFu na týchto stranách nič nenachádzal. Celkovo môžeme povedať, že výsledky zo scény2 boli horšie. Podľa môjho názoru to malo za príčinu osvetlenie, keďže deskriptor SURFu nie je úplne invariantný na podmienky osvetlenia. Taktiež objekty ktoré majú kontrastnejší povrch dopadli lepšie. Príkladmi sú objekty *kečup* s bielou na červenej a *balzam* so zelenou na bielej.

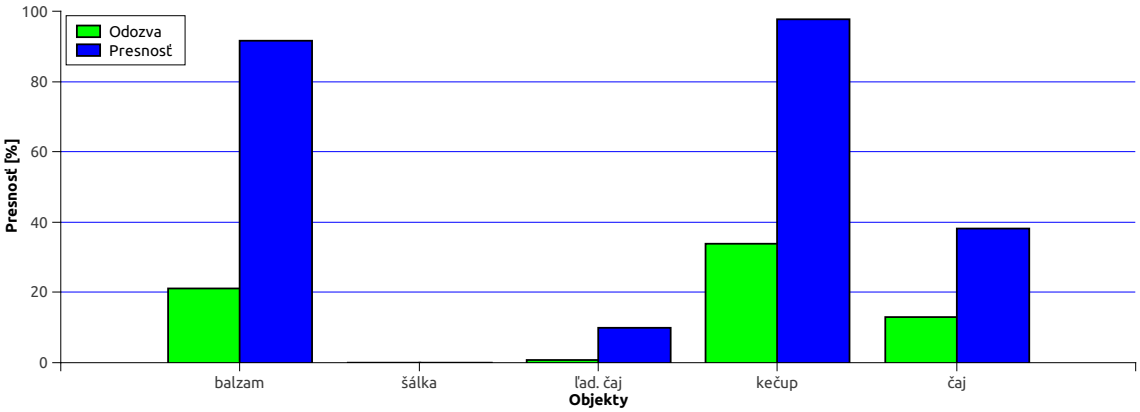
V tabuľke 6.1 sú uvedené priemerné hodnoty získané s doby behu jednej detekcie. Túto dobu ovplyvňuje hlavne počet aspektov, ktoré museli byť skúmané. Priemerná doba spracovávania jedného aspektu bola v scené1 *2 ms* a u scény *3 ms*. Rozdiel je spôsobený tým, že detektor musel skúmať vyšší počet kľučových bodov.

Zistil som, že výpočet SURFov trval v priemere u scény1 *66 ms* a *97 ms* v scéne2, čo znamená, že detektor venoval viac ako 80% času detekcii a popisu kľúčových bodov scény.

[1] Parameter pri scéne1 a scéne2.

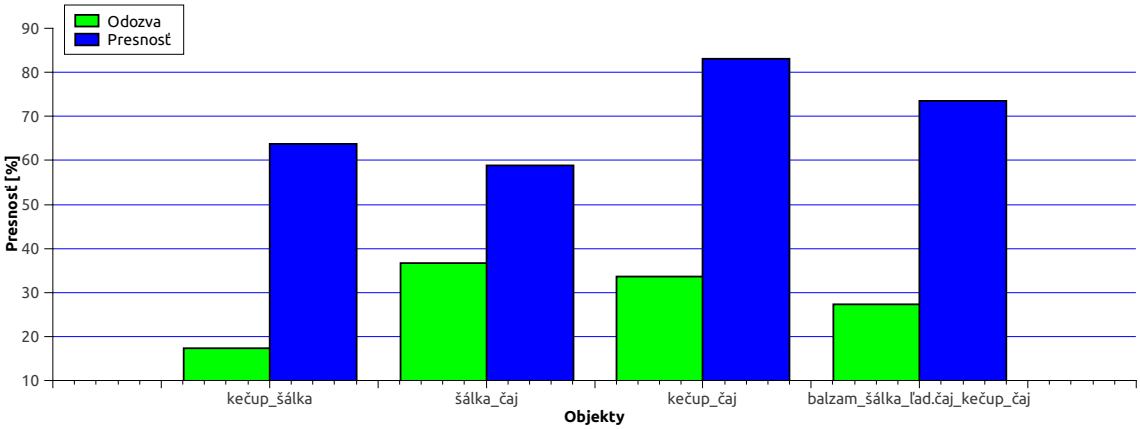


Obrázok 6.1: Výsledky detekcie v scéne1.



Obrázok 6.2: Výsledky detekcie v scéne2.

Celkovo ak môžem hodnotiť najlepšie fungovala detekcia u objektu *balzam* a *kečup*. Najhoršie u *šálky*. Pri nahrávaní aj detekcií robilo najväčšie problémy osvetlenie a textúra objektu. Výsledkami ma prekvapil objekt *ľadový čaj*, s *odozvou* 0.68% a *presnosťou* 9.93% v scéne2, pretože som si myslel, že nápis bude pre detekciu postačujúci aj v preplnenej scéne. Systém pri nahrávaní hlásil, že nemôže nájsť značky na podložke, preto bola prvá konfigurácia nahrávania ktorú som skúsil, doplnená o dve lampy, ktoré problém vyriešili. Boli prípady kedy objekty(*šálka*, *ľad. čaj*) v scéne neboli detekované a stačilo im len zvýšiť osvetlenie. Nahrávanie testovacích dát prebiehalo bez problémov. Ovládanie systému je podľa mňa príliš „skostnatelé“. Pri zmene parametru `EPSILON` bolo nutné prekompilovať celý systém, čo trvalo približne dve minúty. Taktiež si myslím, že systém takéhoto druhu by mal umožňovať „zabúdanie“ objektov, prípadne odloženie s hľadiska využitia zdrojov. Z týchto dôvodov si myslím, že systém nie je určený pre detekciu v scénach s dynamickým osvetlením.



Obrázok 6.3: Výsledky pri detekcii viacerých objektov v scéne1.

Objekt	**počet aspektov**	**scéna1 [ms]**	**scéna2 [ms]**
balzam	48	90	167
šálka	31	80	114
ľad.čaj	29	82	117
kečup	47	87	125
čaj	43	84	141

Objekty	**scéna1 [ms]**
kečup_šálka	102
šálka_čaj	108
kečup_čaj	110
Všetky obj.	222

Tabuľka 6.1: Doba detekcie

Kapitola 7

Záver

Cieľom tejto práce bolo zhrnúť do dnešnej doby prezentované metódy a postupy pri detekcii objektov využívajúce ako obrazovú tak hĺbkovú informáciu z obrazu. Taktiež natrénovať robustný detektor a vyhodnotiť jeho výkon. Bolo ukázané, že detektor zo systému RoboEarth si poradí s detekciou ako jedného tak viacerých objektov súčasne. Výrazne lepšieho výkonu dosahoval v jednoduchej scéne.

Bolo zistené, že najväčšie problémy pri detekcii robia systému textúra objektu a osvetlenie. Algoritmus SURF a použitie jeho deskriptoru pri čistom porovnaní bodov z modelu a scény ukázalo na jeho rezervy. Možno by pomohlo zhlukovanie SURF deskriptorov, kde by sa do významného bodu ukladal len index zhluku, do ktorého deskriptor spadá, prípadne priamo stred zhluku kde by veľkosť deskriptora bola zachovaná. To by znamenalo aj upraviť metriku porovnávania. Taktiež výpočet SURFov je zložitá operácia a bolo by vhodné aplikovať urýchlenie jeho výpočtu napríklad na grafickej karte[11].

Ďalej by som videl možnosti zlepšenia v rozčlenení detekčného uzla `re_kinect_object` na vlákna. RoboEarth všetko vykonáva sekvenčne pre každý obdržaný obraz scény.

Každý mnou skúmaný objekt vykazoval najvyššiu odozvu pri odlišnej hodnote parametra `EPSILON`. Tento parameter je umiestnený v kóde staticky a počas behu detekcie ho nie je možné meniť. Opätovná rekompilácia RoboEarth pri zmene parametru trvá okolo dvoch minút. Dalo by sa to riešiť napr. cez GUI, kde by sa zadala hodnota parametru a tá by sa pomocou ROS-spravy poslal detektoru, alebo by táto hodnota mohla byť súčasťou modelu. Ďalšiu nevýhodu vidím v tom, že systém nevie „zabúdať“ objekty za behu. Pre vymazanie je nutné celý systém reštartovať.

Aj keď prvá verzia systému RoboEarth vyšla len v septembri minulého roku, pracuje sa na ňom stále. Svedčí o tom fakt, že za posledný rok čo som s týmto systémom pracoval sa systém menil a opravoval. Budúcnosť nám ukáže na akú úroveň sa systém dostane.

Literatúra

[1] Bay, H.; Ess, A.; Tuytelaars, T.; aj.: Speeded-up robust features (SURF). *Computer Vision and Image Understanding*, ročník 110, č. 3, 2008: s. 346–359.
Dostupné z WWW:
<http://www.sciencedirect.com/science/article/pii/S1077314207001555>

[2] Dalal, N.; Triggs, B.: Histograms of Oriented Gradients for Human Detection. *Proc. of the IEEE Conf. on Comp. Vis. and Pat. Rec. (CVPR)*, 2005.
Dostupné z WWW:
<http://lear.inrialpes.fr/people/triggs/pubs/Dalal-cvpr05.pdf>

[3] Eindhoven University of Technology: RoboEarth.org. 2011, [online], [cit. 2012-02-14].
Dostupné z WWW: <http://www.roboearth.org/>

[4] Felzenszwalb, P.; McAllester, D.; Deva, R.: A discriminatively trained, multiscale, deformable part model. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008.
Dostupné z WWW:
<http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4587597>

[5] Hinterstoisser, S.; Holzer, S.; Cagniart, C.: Multimodal templates for real-time detection of texture-less objects in heavily cluttered scenes. *Vision (ICCV), 2011*, Listopad 2011: s. 858–865, doi:⟨10.1109/ICCV.2011.6126326⟩.
Dostupné z WWW:
<http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6126326http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6126326>

[6] Hinterstoisser, S.; Lepetit, V.; Ilic, S.; aj.: Dominant Orientation Templates for Real-Time Detection of Texture-Less Objects. *CVPR*, 2010.

[7] Lai, K.; Bo, L.; Ren, X.: Detection-based Object Labeling in 3D Scenes. *International Conference on Robotics and Automation*, 2012.
Dostupné z WWW:
<http://ai.cs.washington.edu/www/media/papers/3dobject-labeling.pdf>

[8] Lai, K.; Bo, L.; Ren, X.; aj.: A Large-Scale Hierarchical Multi-View RGB-D Object Dataset. *Proc. of International Conference on Robotics and Automation (ICRA)*, 2011: s. 1817–1824, [online].
Dostupné z WWW: <http://www.cs.washington.edu/ai/Mobile_Robotics/projects/postscripts/rgbd-dataset-icra-11.pdf>

[9] Lowe, D. G.: Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, ročník 60, č. 2, Listopad 2004: s. 91–110, ISSN 0920-5691.
Dostupné z WWW: <http://www.springerlink.com/openurl.asp?id=doi:10.1023/B:VISI.0000029664.99615.94>

[10] Marco, D. D.; Janssen, R.: Roboearth - ROS Wiki. 2011, [online], [cit. 2012-02-14].
Dostupné z WWW: <http://www.ros.org/wiki/roboearth>

[11] Martinez, M.; Collet, A.; Srinivasa, S. S.: MOPED: A scalable and low latency object recognition and pose estimation system. *2010 IEEE International Conference on Robotics and Automation*, Květen 2010: s. 2043–2049, [online].
Dostupné z WWW:
<http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5509801>

[12] Muja, M.; Rusu, R. B.; Bradski, G.; aj.: REIN - A Fast, Robust, Scalable REcognition INfrastructure. *Pattern Recognition*, 2011.

[13] Pangercic, D.; Haltakov, V.; Beetz, M.: Fast and Robust Object Detection in Household Environments Using Vocabulary Trees with SIFT Descriptors. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Workshop on Active Semantic Perception and Object Search in the Real World*, San Francisco, CA, USA, 2011, str. 8.
Dostupné z WWW:
<http://ias.cs.tum.edu/_media/spezial/bib/irosws11germandeli.pdf>

[14] Szeliski, R.: *Computer vision: algorithms and applications*. Springer, 2011, ISBN 9781848829343.

[15] Waibel, B. M.; Beetz, M.; Civera, J.; aj.: Roboearth - A World Wide Web for Robots. *Robotics & Automation Magazine, IEEE*, ročník 18,2, č. June, 2011: s. 69–82, [online].
Dostupné z WWW:
<http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=5876227>

[16] Wikipedia: Kinect. [online], [cit. 2012-2-14].
Dostupné z WWW: <http://en.wikipedia.org/wiki/Kinect>

[17] Wikipedia: Point cloud. [online], [cit. 2012-2-14].
Dostupné z WWW: <http://en.wikipedia.org/wiki/Point_cloud>

[18] Willow Garage: ROS.org. 2011, [online], [cit. 2012-02-14].
Dostupné z WWW: <http://www.ros.org/wiki/>

[19] Willow Garage; Google; Toyota; aj.: PCL - Point Cloud Library. 2011, [online].
Dostupné z WWW: <http://pointclouds.org/>

Príloha A

Obsah CD

Priložené CD obsahuje:

- upravené súbory detektoru RoboEarth
- skripty pre vyhodnotenie detekcie
- skript pre ukážku jedného experimentu
- jeden model objektu vytvorený pomocou RoboEarth
- jeden testovací bag-súbor slúžiaci ako vstup pre detektor
- PDF a LaTeXverzia tejto práce
- súbor *README* s popisom použitia
- stručný plagát prezentujúci prácu