VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INTELIGENTNÝCH SYSTÉMOV
DEPARTMENT OF INTELLIGENT SYSTEMS

VIRTUÁLNÍ BRÁNA PRO POČÍTÁNÍ POČTU PRŮCHODŮ OSOB

VIRTUAL GATE FOR COUNTING THE PASSING OF PERSONS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **ANDREJ OLIVER CHUDÝ**
AUTHOR

VEDOUCÍ PRÁCE **Doc. Ing. MARTIN DRAHANSKÝ, Ph.D.**
SUPERVISOR

BRNO 2017

Zadání bakalářské práce/20083/2016/xchudy03

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav inteligentních systémů Akademický rok 2016/2017

Zadání bakalářské práce

Řešitel: **Chudý Andrej Oliver**
Obor: Informační technologie
Téma: **Virtuální brána pro počítání počtu průchodů osob**
Virtual Gate for Counting the Passing of Persons
Kategorie: Umělá inteligence

Pokyny:

1. Prostudujte technologie počítačového vidění s využitím hloubkové kamery a běžné webové kamery.
2. Navrhněte a implementujte aplikaci pro počítání průchodů osob skrz virtuální bránu v dynamickém prostředí.
3. Otestujte více technologií snímání prostoru a porovnejte je mezi sebou.
4. Zhodnoťte dosažené výsledky a diskutujte další pokračování tohoto projektu.

Literatura:

- Del Pizoo, Luca, et al. Counting people by RGB or depth overhead cameras. *Pattern Recognition Letters*, 2016.
- Beymer, David. Person counting using stereo. In: *Proceedings of Workshop on Human Motion*. IEEE, 2000. p. 127-133.

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 a 2.

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Drahanský Martin, doc. Ing., Dipl.-Ing., Ph.D.**, UITS FIT VUT
Datum zadání: 1. listopadu 2016
Datum odevzdání: 17. května 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav inteligentních systémů
612 66 Brno, Božetěchova 2

L.S.

doc. Dr. Ing. Petr Hanáček
vedoucí ústavu

Abstrakt

Táto práca sa zaoberá návrhom a realizáciou aplikácie, ktorá je schopná detekovať prechod osôb cez virtuálnu bránu (bez akejkoľvek fyzickej prekážky). Za účelom dosiahnutia čo najväčšej presnosti, boli implementované **dva programy**. Jeden pre spracovávanie **2D** výstupu z web-kamery a druhý, určený pre spracovanie hĺbkových **3D** dát. Oba programy boli otestované aj v reálnych podmienkach farmaceutického skladu, kde bola dosiahnutá úspešnosť až **98%**. Aplikácia realizovaná v rámci tejto práce má tak potenciál stať sa základom systémov zabezpečujúcich objektovú bezpečnosť alebo jedinečným systémom umožňujúcim zber relevantných štatistických údajov o návštevnosti objektov v reálnom čase. Súčasťou práce je aj porovnanie troch používaných hĺbkomerov.

Abstract

The goal of this thesis is to propose a viable technique of detecting people passing through a virtual line (any opening without a turnstile or other form of the physical obstacle). **Two software solutions** were implemented to achieve high accuracy. First one processes **2D** data output from webcam and the other one processes depth data from the **3D** imaging sensor. Both software solutions were deployed and tested in a pharmaceutical storage room, where they achieved the accuracy of **98%**. The software proposed in this thesis is thus reliable enough to become the basis of access control and security systems or for real-time evaluation of visitor rate statistics of commercial properties and events. The thesis also contains a comparative analysis of 3 widely used depth sensors.

Kľúčové slová

kamera, hĺbkomer, počítačové videnie, opencv, 3D, 2D, počítanie, Raspberry Pi, Kinect 360, Orbbec Astra S, Intel RealSence, hĺbková mapa, objektová bezpečnosť, video, odčítavanie pozadia

Keywords

camera, depth sensor, opencv, python, 3D, 2D, counting, Raspberry Pi, Kinect 360, Orbbec Astra S, Intel RealSence, depth map, object safety, testing, background subtractor, video, computer vision

Citácia

CHUDÝ, Andrej Oliver. *Virtuální brána pro počítání počtu průchodů osob*. Brno, 2017. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Drahanský Martin.

Virtuální brána pro počítání počtu průchodů osob

Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne pod vedením pána Doc. Ing., Dipl.-Ing. Martina Drahanského, Ph.D. a uviedol som všetky literárne pramene a publikácie, z ktorých som čerpal.

.....
Andrej Oliver Chudý
15. mája 2017

Poďakovanie

Moje poďakovanie patrí pánovi Doc. Ing., Dipl.-Ing. Martinovi Drahanskému, Ph.D. za poskytnuté cenné rady a pripomienky. Ďalej by som rád poďakoval firme *Touchless Biometric Systems s.r.o.* za poskytnutie materiálu využitého pri tejto práci. Samostatné poďakovanie patrí aj firme *APIS, spol. s r.o.*, na ktorej námet vznikla celá táto práca a zaslúžila sa o vybavenie väčšiny testovacích miest aplikácie.

Obsah

Kapitola 1

Úvod

Väčšina systémov používaná pre zaznamenávanie počtu osôb prechádzajúcich virtuálnymi bránami je založená na technológií infračervenej závory, z dôvodu nízkej ceny. Tento systém je však veľmi nepresný, pretože nedokáže detekovať viac ľudí prechádzajúcich vedľa seba alebo oproti sebe, čo je veľmi častá situácia, ktorá zapríčiňuje značnú mieru chybovosti. Pri použití iných systémov je nutné prenášať gigabajty dát po sieti alebo sú veľmi drahé (stovky EUR). Preto je nutné vytvoriť systém, ktorý bude presnejší ale zároveň si zachová priaznivú cenu. Touto problematikou sa zaoberá táto práca. Spojením lacnej webkamery a nízkonákladového počítača Raspberry Pi je možné zrealizovať systém s vysokou mierou presnosti a veľmi priaznivou cenou.

Informácie vytvorené takýmto systémom môžu byť veľmi užitočné, napríklad pre obchodné reťazce, ktoré na základe dlhodobej štatistiky návštevnosti jednotlivých predajní dokážu pripraviť obchodné stratégie, vyhodnotiť dopady reklamnej kampane na návštevnosť či vyrátať úspešnosť predaja zamestnancov ako pomer zákazníkov a predaného tovaru. Všetky tieto dáta sú kľúčovým faktorom pre analýzu obchodných stratégii a ich využitie vedie k zefektívneniu prevádzky.

Ďalšie využitie takéhoto systému je na miestach s obmedzenou kapacitou ako napríklad kúpaliská, štadióny, divadlá alebo sklady, kde na základe informácie o počte zamestnancov na jednotlivých oddeleniach, je možné dynamicky prerozdeľovať pracovnú silu v závislosti na množstve požiadaviek na jednotlivé oddelenia.

1.1 Existujúce riešenia

Problematikou počítania (detekovania) prechodov ľudí sa zaoberá celá rada aplikácií. Jedna zo základných aplikácií je IR závora, ktorú bežne môžeme nájsť na akejkoľvek garážovej bráne. Je založená na princípe vysielača a prijímača. Vysielač vysiela infračervený lúč a keď niekto prejde, lúč sa zatieni a rele v prijímači sa zopne. Táto technológia má však veľmi veľa problémov a obmedzení. Najväčším obmedzením je, že môže byť inštalovaná len v zúžených priestoroch, kde môže v jednom okamihu prechádzať len jeden človek. Toto tvrdenie sa opiera o fakt, že ak by išli dvaja alebo viacerí ľudia vedľa seba, senzor to zaznamená ako jedno zopnutie a teda prechod len jednej osoby.

Firma riSys[1] ponúkajú vo svojom portfóliu aplikačné riešenie problému počítania osôb. Využívajú na to termálne kamery. Výrobca uvádza, že ich produkt dosahuje úspešnosť na úrovni 98%. Avšak cena za najlacnejší model sa pohybuje okolo 1000 EUR za kus. Táto firma na svojich stránkach ďalej uvádza, že infračervená technológia, ktorú používajú je v dnešnej dobe najpresnejšou metódou počítania prechodov osôb na trhu.

Ďalšou spoločnosťou, ktorá sa venuje problematike vytvárania štatistických dát z počítania prechodu osôb, odhad veku či spokojnosti zákazníkov je firma Brickstream[2]. Vytvorili malé zariadenie, ktoré na základe stereoskopickej technológie dokáže zaznamenávať jednotlivé transakcie (prechody). Výrobca uvádza presnosť na úrovni 97%.

Ostatné riešenia sa poväčšine opierajú o technológiu obyčajnej kamery. Tento prístup je veľmi lacný, lebo kamera nemusí byť ničím výnimočná dokonca množstvo objektov má kamery už predom inštalované. Postupom času rôzni výrobcovia, začali dokonca implementovať takéto technológie do firmwarov kamier za pomoci hardvérovej akcelerácie. Dokážu počítať ľudí, spustiť alarm pri narušení virtuálnej zóny či čítať poznávacie značky aut. Pri testoch však bolo dokázané, že presnosť takýchto služieb zo strany kamerových spoločností je skôr podpriemerná.

[1] Oficiálne zastúpenie spoločnosti iriSys na Slovensku: http://www.irisys.sk/people-counting
[2] Oficiálna stránka spoločnosti: http://www.brickstream.com

Kapitola 2

Počítačové videnie

2.1 Úvod

Počítačové videnie ako vedná a technologická disciplína sa zaoberá schopnosťou elektronických zariadení získať informáciu z digitálneho obrazu - pochopiť situáciu a na základe nej urobiť rozhodnutie či vykonať zadanú úlohu. Patria medzi technológie využívané vo všetkých oblastiach priemyslu a výskumu. Aplikácie počítačového videnia môžu riešiť jednoduché zadania, ako napríklad počítanie fliaš na plniacej linke alebo komplexné úlohy, ktoré vyžadujú doslova pochopenie okolitého sveta a využívajú najmodernejšie systémy umelej inteligencie. [9]

Základom je spracovanie 2D obrazu (*Image Processing*). Túto etapu väčšinou predstavujú algoritmy na nízkej úrovni abstrakcie (*low-level*). Algoritmy, ktoré extrahujú znalosti z nižšej úrovne, dávajú ich do súvislostí a snažia sa im porozumieť (*Image understanding, machine learning*) pracujú na vyššej úrovni (*hight-level*). To by nebolo možné bez znalosti fyzikálnej podstaty senzorov, optiky, okolitého prostredia a tiež matematických a štatistických metód na ich hodnotenie. [9]

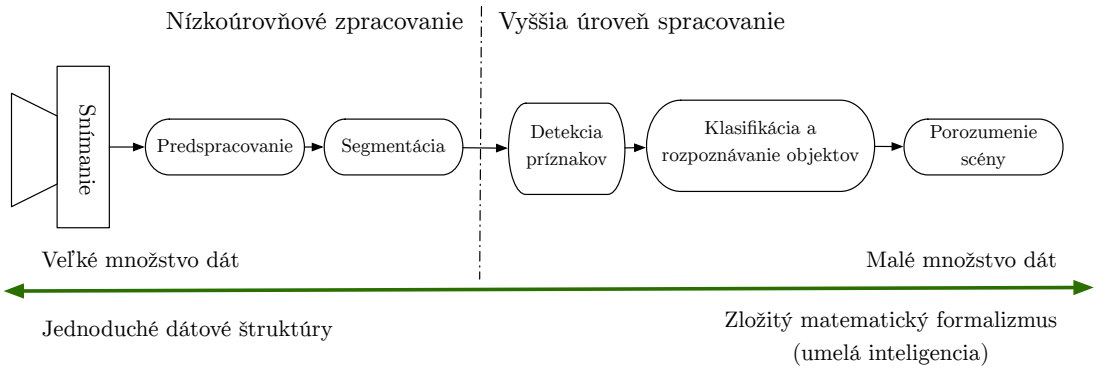
2.2 Najvýznamnejšie problémy počítačového videnia

Možno ich rozdeliť do nasledujúcich bodov [12]:

- Strata informácie pri premietaní z 3D do 2D.
- Interpretácia ľudského mozgu nie je správna. Sietnica ľudského oka je dvojrozmerná a trojrozmernú predstavou okolia vytvára mozog, na základe znalostí.
- Šum je prítomný v každom reálnom obraze a preto namiesto deterministických modelov je potrebné používať **stochastické modely**, ktoré predpokladajú určitý stupeň neurčitosti a dokážu túto neurčitosť zahrnúť do matematického modelu.
- Obrovský tok dát prijímaný z kamier, ktoré je nutné spracovať v reálnom čase.
- Nejednoznačná interpretácia jasu jednotlivých častí objektu. Bez poznania fyzikálnej povahy sledovaného objektu nie sme schopní jednoznačne určiť, či výrazné svetlé škvrny na obraze sú dôsledkom extrémne svetlých častí objektu alebo iba dôsledkom odrazu zdroja svetla v lesklej (ale nie svetlej) časti povrchu.

2.3 Etapy spracovania obrazu

Jednotlivé etapy spracovania obrazu možno vyjadriť grafom, ktorý znázorňuje postupnosť jednotlivých krokov v závislosti na množstve dát a veľkosti abstrakcie v jednotlivých krokoch. [12]



Obr. 2.1: Etapy spracovania obrazu.

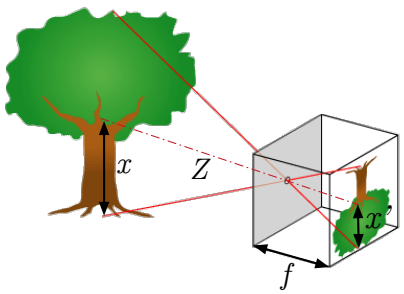
V nasledujúcom texte si postupne opíšeme každú fázu spracovania.

2.3.1 Snímanie

Základným obmedzením pri vnímaní okolitého 3D sveta, je skutočnosť, že človek dokáže vnímať okolie len v 2D, ktoré predstavuje zjednodušený model 3D okolia.

Dierková komora

Tento model kamery možno použiť ako prvú aproximáciu mapovania 3D scény do 2D obrazu. Je to analógia toho, čo vzniká na sietnici ľudského oka. Obraz sa premieta do obrazovej roviny vzdialenej od dierky o ohniskovej vzdialenosti f. V prípade, že sa zrkadlová rovina presunie pred rovinu projekcie z podobnosti trojuholníkov vyplýva nasledujúci vzťah [2][9]:



Obr. 2.2: Model dierkovej kamery

$$x = \frac{Xf}{Z} \tag{2.3.1}$$

Tento výraz predstavuje najjednoduchší model dierkovej kamery, ktorý je základom zložitejších modelov slúžiacich na výpočty napr. pri 3D projekcií.

Prevedenie obrazu do počítača (snímanie)

Je proces transformácie fyzikálneho obrazu do podoby vhodnej pre spracovanie za pomoci počítača (matica pixlov). Tento proces sa skladá z troch častí:

- **Snímanie** (*scanning*) - cieľom prvého kroku je prevod optických veličín na elektrické. Pri použití nekvalitného snímača sa na výstupe objaví veľké množstvo šumu, ktoré len ťažko možno korigovať. Preto je nutné použiť snímač čo najväčšej kvality. [12]

- **Vzorkovanie** [12] - Shanonova veta o vzorkovaní nám vraví, že vzorkovacia frekvencia musí byť aspoň 2x vyššia než je maximálna frekvencia, ktorú obsahuje vzorkovací signál. [12]

$$f_{vzor} \geq 2f_{max} \tag{2.3.2}$$

  V obrazovej terminológii nám táto veta vraví, že interval vzorkovania musí byť menší alebo rovný polovici rozmerov najmenších detailov na obraze.

- **Kvantovanie** je prevod hodnoty obrazovej funkcie do číselnej hodnoty. Vždy je to proces stratový a nevratný. Počet kvantovacích bodov musí byť dostatočne jemný. Najčastejšie sa v praxi používa 256 úrovní jasu, čo vieme zakódovať do 8 bitov. Výnimku tvoria binárne obrazy, ktoré si vystačia len s dvoma úrovňami 0 a 1. Často sa používa na vytvorenie masky, v ktorej možno ľahko nájsť kontúry. Najväčší problém kvantovania na nedostatočný počet úrovní je vnímanie falošných hrán. To je možné odstrániť nelineárnym kvantovaním alebo rozostrením obrazu. [12]

- **Dátové štruktúry** - najčastejšou reprezentáciou celého obrazu je **matica** hodnôt. Súradnice prvku v matici zodpovedajú súradniciam pixlu na obraze. Jeho hodnota zodpovedá napríklad jasu. Okrem nej sa však používajú aj formy reprezentácie obrazu, ako napríklad kódovanie kontúr segmentovaných objektov. Kontúra je uzavretá hranica objektu, ktorá môže reprezentovať objekt reálneho sveta na obraze a môže byť zdrojom informácií o ňom (tvar, farba, veľkosť). [12]

  Ďalšou možnosťou kódovania je **Run-length coding** forma reťazového kódu, použitá hlavne v binárnych obrazoch.

  Freemanov (reťazový) kód je určený počiatočným bodom a postupnosťou symbolov zodpovedajúcich úsečkám jednotkovej dĺžky v niekoľkých vopred stanovených orientáciách. [12]

  Polygonálna reprezentácia hranice je reprezentácia, ktorá aproximuje oblasť mnohouholníkom. Každá hranica je jednoznačne určená vrcholmi aproximovaného mnohouholníka. [12]

  Hierarchické dátové štruktúry obsahujú okrem originálneho obrazu aj jeho zjednodušené časti. Typicky tieto dátové štruktúry pomáhajú znižovať výpočtovú náročnosť a to tak, že algoritmus začne hrubšou aproximáciou obrazu, kde sa vytipujú záujmové oblasti a ďalej algoritmus pokračuje na vybrané časti originálneho obrazu. Základné hierarchické reprezentácie sú pyramídy a kvadrantové stromy. [12]

2.3.2 Predspracovanie obrazu

Vstupom aj výstupom je obraz na nízkej úrovni abstrakcie. Slúži na zlepšenie obrazu z hľadiska ďalšieho spracovania. Môže potlačiť alebo naopak zvýrazniť isté informácie v obraze. Metódy spracovania delíme na [9]:

- Bodové jasové transformácie
- Geometrické transformácie
- Lokálne predspracovanie (filtrácia, ostrenie a detekcia hrán)
- Obnovenie obrazu pri známej degradácii
- Matematická morfológia

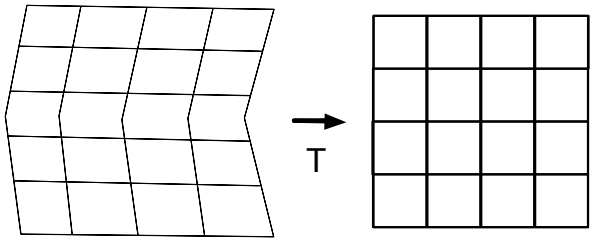
Bodové jasové transformácie

Jasová korekcia - používa sa pri nerovnomernom osvetlení obrazu. Pokiaľ je porucha osvetlenia systematická, vieme korigovať odchýlku každého bodu od ideálnej charakteristiky. [9]

Modifikácia jasovej stupnice - metóda nezávisí na polohe bodu v obraze. Táto modifikácia T prevedie vstupnú jasovú hodnotu na výstupnú pomocou tabuľky. [9]

Geometrické transformácie

Geometrická transformácia 2D obrazu je vektorová transformácia T, ktorá zobrazí bod (x,y) do bodu $(x_0,\ y_0)$. Jedná sa teda o transformáciu súradníc bodov obrazu. Vďaka tomu môžeme odstrániť geometrické skreslenie vzniknuté pri snímaní obrazu (napr. korekcie geometrických porúch objektívu kamery, oprava skreslenia družicového snímku spôsobená zakrivením zemegule). Z toho vyplýva, že pomocou referenčnej mriežky vieme vyrovnať geometricky deformovaný obraz. Transformačné rovnice môžu byť známe alebo odvodené na základe vstupného a transformačného obrazu. Pred začiatkom transformácie sa musia nájsť dvojice zodpovedajúcich si bodov (vlícovacie body), ktoré sa na obrázkoch ľahko hľadajú napr. priesečníky vláknitých štruktúr, rohy objektov a pod. Následne sa vykoná transformácia súradníc bodov do nových pozícií. A nakoniec sa musí vykonať aproximácia jasovej funkcie. [14]



Obr. 2.3: Príklad geometrickej transformácie.

Lokálne predspracovanie - filtrácia

Používa sa na odstránenie vysokofrekvenčných zložiek, čo predstavujú hrany a šum. Pri tejto operácií dochádza k strate detailov obrazu (rozmazanie). [13]

Spriemerovanie - ide o najjednoduchší prostriedok pre vyhladenie šumu v obraze. Pre každý bod obrazu sa jeho jas nahradí aritmetickým priemerom jasov jeho susedov. Filtrácia spriemerovaním je vlastne špeciálny prípad konvolúcie[1], kde konvolučná maska vyzerá nasledovne: [13]

$$h = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \tag{2.3.3}$$

Filtrovanie pomocou Gaussovej masky - Gaussovú masku dostaneme posilnením stredového bodu, prípadne jeho štyroch susedov tak, aby lepšie aproximoval šum s Gaussovým rozdelením. [13]

$$h = \frac{1}{10}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 1 \end{bmatrix} \tag{2.3.4}$$

$$h = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \tag{2.3.5}$$

Mediánový filter - medián alebo prostredná hodnota je hodnota, ktorá rozdeľuje postupnosť podľa veľkosti zoradených výsledkov na dve rovnako početné polovice. V štatistike patrí medzi stredné hodnoty. Usporiadame jasové úrovne z lokálneho okolia a vyberieme strednú hodnotu usporiadania. To zabráni, aby krajné extrémy ovplyvnili vybranú hodnotu. Filtrácia však nie je vhodná v obraze, kde sa nachádzajú tenké čiary a ostré rohy. [13]

Lokálne predspracovanie - detekcia hrán

Jedná sa o opak filtrácie, zvyšuje vysokofrekvenčné časti spektra (hrán) a odstraňuje nízkofrekvenčné časti. Sprievodným javom je aj zvýraznenie šumu v obraze. Hrana je vektorová veličina, určená veľkosťou a smerom. Pri hranovej detekcií sa využíva vlastnosť gradientu, ktorá hovorí, že hodnota gradientu funkcie dvoch premenných je v oblasti hrany najväčšia. [5]

Laplacov operátor - tento operátor aproximuje druhú deriváciu a predstavuje rýchlosť zmeny hodnôt jasu. Prejaví sa najmä na strmých alebo izolovaných hranách alebo ju možno použiť na detekciu izolovaných bodov. Bude zvýrazňovať aj šum. [13][5]

$$h_1 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \tag{2.3.6}$$

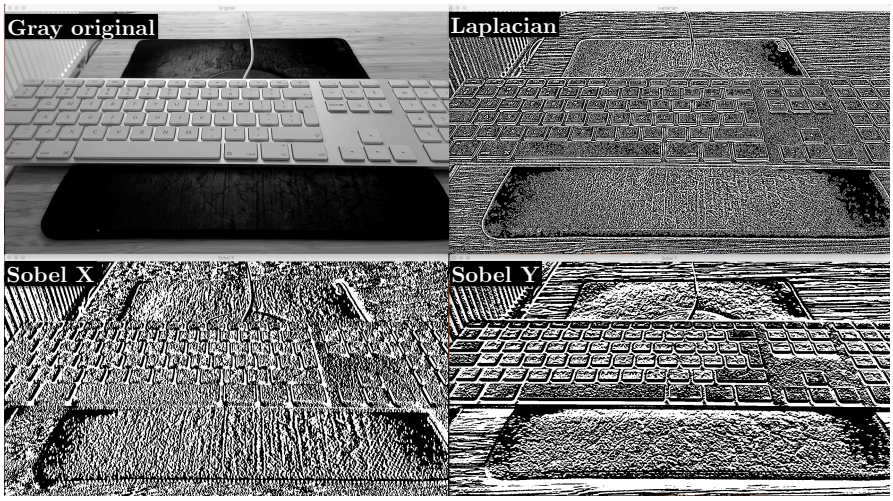
$$h = \begin{bmatrix} 0 & 0 & -1 & 0 & 0 \\ 0 & -1 & -2 & -1 & 0 \\ -1 & -2 & 16 & -2 & -1 \\ 0 & -1 & -2 & -1 & 0 \\ 0 & 0 & -1 & 0 & 0 \end{bmatrix} \tag{2.3.7}$$

[1] Diskrétna 2D konvolúcia http://bruxy.regnet.cz/fel/36ACS/konvoluce.pdf

Sobelov operátor - aproximuje prvé parciálne derivácie. Nakoľko samotná derivácia zvýrazňuje šum, vykonáva aj vyhladzovanie. Pre každý smer hrán existuje špeciálna maska. [13]

$$h_1 = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \tag{2.3.8}$$

$$h_2 = \begin{bmatrix} 0 & 1 & 2 \\ -1 & 0 & 1 \\ -2 & -1 & 0 \end{bmatrix} \tag{2.3.9}$$



Obr. 2.4: Vzájomné porovnanie algoritmou pre detekciu hrán. Demo program príloha E.

2.3.3 Segmentácia obrazu

Segmentácia je proces rozdelenia obrázku na časti, ktoré korelujú s objektami reálneho sveta. Pri čiastočnej segmentácií je cieľom rozdeliť obraz na časti, ktoré sú homogénne z hľadiska vybranej vlastnosti (farba, jas, odrazivosť atď.). Cieľom segmentácie v počítačovom videní je značná redukcia objemu dát. Nejednoznačnosť obrazových dát je hlavným problémom segmentačného procesu. Segmentačné metódy možno rozdeliť na tri skupiny [9]:

- Prahovanie
- Segmentácia založená na hranách (diskontinuita)
- Segmentácia založená na oblastiach (podobnosť)

Prahovanie

Jedná sa o najjednoduchšiu segmentačnú techniku. Je založená na predpoklade, že jednotlivé objekty majú konštantnú obrazivosť, či pohltivosť svetla na svojom povrchu. Je to transformácia, ktorá zobrazuje vstupný obraz $f(i,j)$ na výstupný obraz $g(i,j)$ nasledovne [8]:

$$\begin{aligned} g(i,j) &= 1, \quad akf(i,j) \geq T \\ g(i,j) &= 0, \quad akf(i,j) < T \end{aligned} \tag{2.3.10}$$

Z toho vyplýva, že na segmentovanie objektov od pozadia sa používa jasová konštanta, ktorá sa nazýva prah. [9]

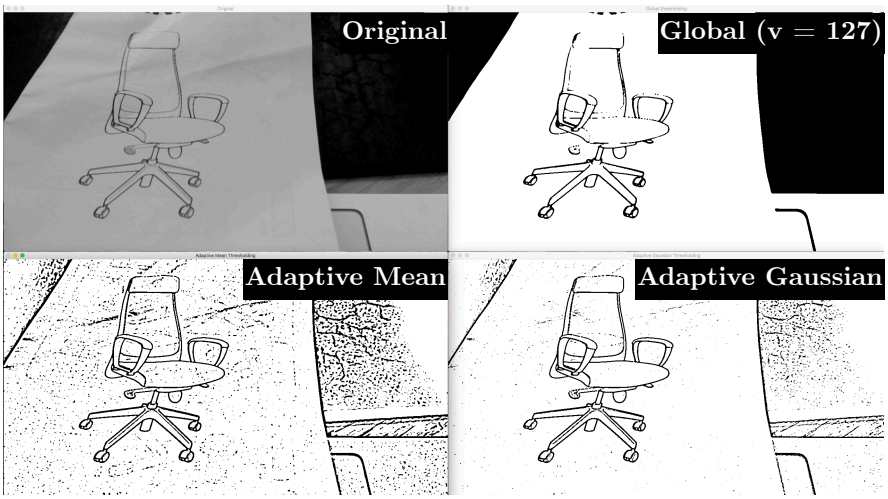
Globálne prahovanie - je to technika, ktorá je vhodná v situáciach, keď sa objekty na scéne diametrálne líšia svojimi charakteristikami. V takom prípade môžeme nastaviť prah ako interval platných hodnôt. Príklad použitia môžeme vidieť na obrázku s kačkou. Originálny obrázok sa konvertuje do HSV farebného priestoru, ktorý umožňuje lepšiu prácu s odtieňmi farieb spôsobené nehomogénnym osvetlením. Následne sa aplikuje filter, ktorý prepustí len hodnoty v danom intervale. [8][5]

Obr. 2.5: Detekcia tela kačky na základe prahovania. Demo program príloha E.

Lokálne prahovanie - málokedy je možné použiť jednu hodnotu prahu (alebo interval) na celý obraz, kvôli fyzikálnym podobnostiam objektov v scéne a nehomogénnemu osvetleniu scény. Preto sa často používajú metódy lokálneho prahovania. Algoritmy sa počítajú v malých regiónoch obrazu. Tak dostaneme rôzne prahové hodnoty pre rôzne regióny rovnakého obrazu a to nám dáva lepšie výsledky u snímok s rôznym osvetlením. Môžeme ho definovať takto [8][1]:

$$T = T(f, f_c), \tag{2.3.11}$$

kde f_c je časť obrazu, v ktorej sa určuje prah.



Obr. 2.6: Porovnanie výsledkov metód lokálneho prahovania. Demo program príloha E.

Segmentácia na základe detekcie hrán

Sú to techniky založené na informáciach, ktoré poskytujú hrany v obraze. Na začiatok je nutné si vysvetliť, aký rozdiel je medzi pojmom hrana a hranica. Hrana je miesto, kde sa skokovito mení hodnota obrazovej funkcie. Pre účely segmentácie sú však dôležitejšie hranice regiónov. Región je množina súvislých bodov. Hranica regiónu je množina bodov, ktoré sú súčasťou regiónu, ale zároveň aspoň jeden z ich susedov nepatrí regiónu. Takáto hranica tvorí vnútornú hranicu regiónu. [9]

Metóda sledovania hranice - metóda pracuje na binárnom obraze. Vnútornú hranicu získame tak, že prehľadávame obraz zľava doprava, zhora dole, pokiaľ nenarazíme na bod patriaci regiónu. Tento bod pridáme do hranice. Potom prehľadávame okolie tohto bodu v protismere hodinových ručičiek od posledne kontrolovaného bodu. Pokiaľ narazíme na bod, ktorý je patrí do rovnakého regiónu, pridáme ho do hranice. Tento postup opakujeme dokiaľ sa nedostaneme naspäť k počiatočnému bodu. [9]

Algoritmus nájde iba vnútorne hranice. Pokiaľ chceme nájsť vonkajšie hranice, môžeme ich získať tak, že nájdeme vnútorné. Potom vonkajšiu hranicu tvoria body, ktoré sme pri hľadaní vnútornej hranice testovali, ale nepatrili do regiónu.

Tento algoritmus funguje pre oblasti väčšie než 1 pixel. Ide o veľmi obľúbený a účinný algoritmus, ktorý sa veľmi často používa.

Cannyho detektor [6] - jeden z najlepších algoritmov pre detekciu hrán, ktorý je založený na hľadaní hodnoty gradientu a spoľahlivosti bodu na základe susedných bodoch. Požiadavky na úspešnosť sú presnosť, minimálny počet chýb a jednoznačnosť.

Prvý krok algoritmu je eliminácia šumu použitím *Gaussovho filtru*. Následne nájdeme miesto a smer gradientu použitím *Sobelového operátora*. Ďalej musíme odobrať z hodnôt gradientu body, ktoré nie sú lokálne maximá, napríklad ak máme pixel, ktorým prechádza zvislá hrana, musí byť jeho ľavý a pravý sused nižšej hodnoty, aby bol uznaný ako skutočná hrana. V poslednom kroku musíme uplatniť metódu *prahovania s hysteréziou*. Zvolíme si minimálny (T1) a maximálny (T2) prah, medzi ktorým môže gradient kolísať. Ak je hodnota gradientu pixlu vyššia než T2, pixel je označený ako hranový. Ak hodnota bodu leží medzi T1 a T2 bod je uznaný ako hranový, ak leží vedľa suseda označeného ako hranový. [6]

Hľadanie hraníc Houghovou transformáciou [10] - pomocou tejto metódy je možné nájsť v obraze objekt, ktorého tvar je možné popísať analytickým výrazom (priamka, kruh, elipsa a iné). Veľkou výhodou je invariantnosť metódy na zmenu mierky, pootočenie a veľká odolnosť voči pôsobeniu šumu v obraze. [10]

Segmentácia založená na spájaní a delení oblastí

Metódy tejto kategórie nehľadajú hranice jednotlivých regiónov ale snažia sa o nájdenie oblastí priamo. Hlavná myšlienka je založená na klasifikácií obrazu do niekoľkých spojitých homogénnych podoblastí, ktoré sú vzájomne disjunktné, pričom zjednotením podoblastí je celý obraz. [9]

$$R = \bigcup_{i=1}^{s} R_i, \quad R_i \bigcap R_j = 0, \quad pre \quad i \neq j \tag{2.3.12}$$

Medzi základné podmienky segmentácie patrí kritérium homogenity:

$$\begin{gathered} H(R_i) = THRUE, \quad i = 1,2,...,s, \\ H(R_i \cup R_j) = FALSE, \quad i \neq j, \quad R_i \; je \; susedne \; k \; R_j \end{gathered} \tag{2.3.13}$$

S je celkový počet regiónov v obraze a $H(R_i)$ je hodnota binárne homogenity regiónu R_i. Prvá podmienka sa týka vlastností, ktoré musia spĺňať pixle v segmentovanom regióne, druhá určuje, že susedné regióny R_i a R_j majú odlišné veľkosti. Existuje veľké množstvo metód založených na segmentácií regiónov, pričom sa z výkonnostného hľadiska veľmi nelíšia. Veľkou výhodou metód tejto kategórie je odolnosť voči šumu. Preto v prípadoch veľkého zašumenia obrazu sú lepšou variantu v porovnaní s metódami založenými na detekcií hrán avšak na úkor väčšej výpočtovej náročnosti. [9]

2.3.4 Príznaky a rozpoznávanie

Proces segmentácie zaručuje, že obraz bol rozdelený do vzájomne disjunktných častí. To však vo väčšine prípadov nestačí a je nutné rozdeliť jednotlivé regióny tak, aby korelovali s objektami reálneho sveta. To si však vyžaduje vytvoriť exaktný popis oblastí. Až na základe neho môže klasifikátor rozdeliť jednotlivé objekty do tried. Z toho vyplýva, že proces segmentácie bezprostredne predchádza procesu klasifikácie. Tento proces však nie je vždy nutný. Ak je cieľom odlíšenie objektov od pozadia, výsledkom takejto snahy je binárny obraz, kde objekty záujmu sú biele a nepodstatné okolie čierne. Rozpoznávanie je klasifikačná úloha. Existujú dve základné formy opisu objektu, podľa nich delíme aj metódy rozpoznávania na dve základné skupiny [12]:

- Príznakové - príznaky popisujúce objekt
- Syntaktické - pomocou primitív (formálne gramatiky, produkčné pravidlá, fuzzy logika)

Skalárne deskriptory

Je to výsledok merania, ktorý kvantifikuje nejakú vlastnosť. Medzi jednoduché skalárne deskriptory patria napríklad veľkosť, okrúhlosť, obvod, dĺžka hlavnej osi, uhol hlavnej osi, ťažisko, projekcia, výška, šírka, výstrednosť, pozdĺžnosť, pravouhlosť, smer, nekompaktnosť, Feretov priemer a iné. [12]

Tvarové invarianty - sú to deskriptory tvaru objektu, ktoré sú invariantné k určitej triede transformácií. Často používané sú aj viacrozmerné skalárne deskriptory, medzi ktoré patrí napríklad obraz, histogram alebo kombinácia skalárnych deskriptorov. [12]

Momenty

Jeho všeobecná definícia je takáto [9]:

$$m_{pq} = \sum_{i=\infty}^{\infty} \sum_{i=\infty}^{\infty} i^p j^q f(j,j) \tag{2.3.14}$$

kde $f(j,i)$ je hodnota jasu pixlu na pozícii j, i. Na základe momentov je možné vypočítať charakteristiky oblastí, ako napríklad ťažisko takto:

$$x_c = \frac{m_{10}}{m_{00}}\ , \quad y_c = \frac{m_{01}}{m_{00}} \tag{2.3.15}$$

Následne je možné definovať centrálny moment, ktorý je invariantný voči posunu.

Štatistické metódy rozoznávania

Základom týchto metód sú štatistické postupy a úzko súvisia s pojmom klasifikácia. Klasifikátor je nástroj, ktorý na základe príznakov zaradí objekty do tried. Funkcie, ktoré oddeľujú jednotlivé triedy sú diskriminačné funkcie. Pokiaľ ich vieme znázorniť ako priamku, ide o lineárne klasifikátory. Existujú tri typy klasifikátorov [9]:

- Deterministický – založený na diskriminačných funkciách, určitá vzorka je vždy zaradená do určitej triedy
- Stochastický - založený na pravdepodobnosti, že klasifikátor zaradí niektorý z objektov do nesprávnej triedy (každé rozhodnutie nesie riziko chyby). Napríklad Bazesov klasifikátor
- Heuristický - neurónové siete, k-najbližších susedov, logistická regresia, *Support Vector Machines* (SVM), rozhodovacie stromy (*Decision trees*)

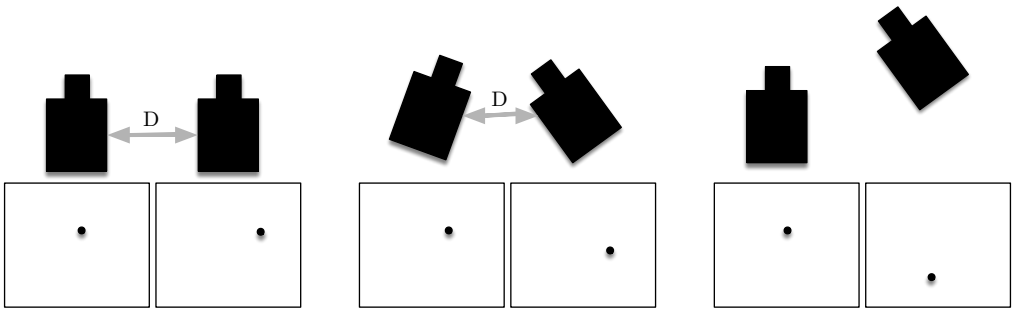
2.3.5 3D snímanie priestoru

Základným princípom je projekcia lúčov odrazených od 3D objektov sveta na 2D plochu snímača. Pri tomto procese však dochádza k rôznym efektom, pri ktorých sa stráca veľa informácií, napríklad premietanie viacerých bodov 3D scény do jedného bodu na 2D senzore, prekrývanie objektov alebo aj šum. Existujú však metódy, ktorými sa darí dané problémy eliminovať a dosiahnuť tak hlavný cieľ, ktorým je porozumieť 3D snímanej scéne. Metódy 3D snímania možno deliť do dvoch základných skupín [12]:

- Pasívne metódy
- Aktívne metódy

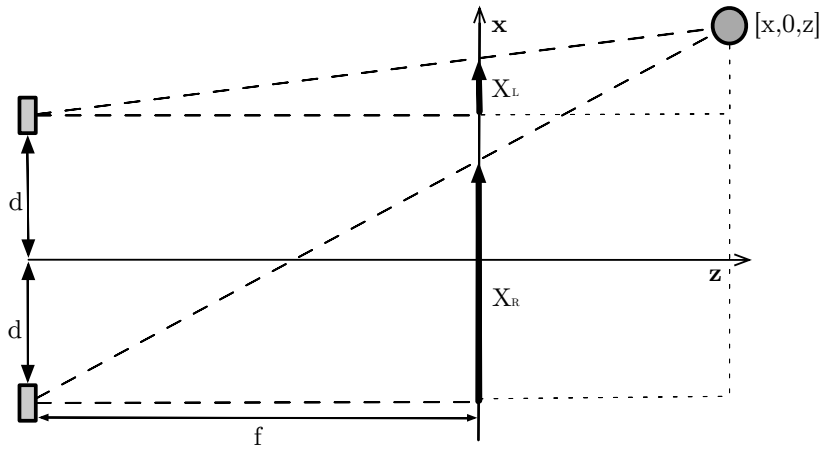
Pasívne metódy - stereo videnie

Tieto metódy potrebujú pre snímanie scény zisk z dvoch kamier so vzájomne rôznou pozíciou a ľubovolným uhlom.



Obr. 2.7: Možnosti vzájomných polôh dvoch kamier.

V nasledujúcom texte sa však obmedzíme na prvú situáciu na obrázku 2.7. Obe kamery snímajú scénu pod rovnakým uhlom a známou vzdialenostnou medzi nimi. Táto situácia je rovnaká ako v ľudskom videní, len namiesto očí pracujeme s kamerami. Táto metóda je založená na triangulácií, čo je orientácia snímkovej dvojice. [14]



Obr. 2.8: Výpočet vzdialenosti na základe triangulácie.

$$\frac{X_L}{f} = \frac{x-d}{z} \; ; \quad \frac{X_R}{f}\frac{x+d}{z} \qquad (2.3.16) \qquad z = \frac{-2df}{X_L - X_R} \qquad (2.3.17)$$

Z obrázka vyplýva, že pri známych veličinách f (ohnisková vzdialenosť) a d (vzdialenosť kamier od seba) vieme vypočítať pre každý pixel jeho hĺbku. Je to možné na základe disparity, čo je rozdiel medzi polohou bodu snímaného jednou kamerou oproti pozícií toho istého bodu snímaného druhou kamerou. Takže každý pixel scény pozorovanej dvoma kamerami má istú hodnotu disparity. Čím je vzdialenosť od kamier väčšia, tým je táto hodnota menšia a čím je vzdialenosť menšia, hodnota disparity je väčšia. Implementácia je veľmi náročná. Vyžaduje si presné technické špecifikácie kamier a algoritmy, ktoré dokážu nájsť korešpondujúce body sú zložité. Najpoužívanejším je RANSAC. [9]

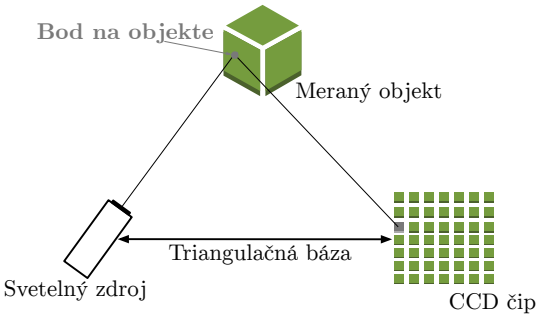
Pre kamery, ktoré nie sú umiestnené v rektifikovanej polohe sa používa epipolárna geometria.

Postupné ostrenie obrazu

Pokiaľ máme k dispozícii kvalitnú kameru, ktorá dokáže selektívne zaostrovať na určitú vzdialenosť, vieme postupným procesom získať hĺbku každého pixlu, podľa momentu, kedy bol správne zaostrený. To, či je obraz na určitom mieste zaostrený, je možné určiť pomocou rôznych lokálnych operátorov známych z detektorov hrán (sekcia 2.3.2). [9]

Aktívna triangulácia

Techniky aktívnej triangulácie spočívajú vo fotogrametrickej rekonštrukcií snímaného objektu osvietením jeho povrchu svetelným zdrojom a súčasným snímaním CCD snímačom. Zdroj svetla spolu so snímačom a osvetleným bodom na skúmanom objekte tvoria triangulačný trojuholník. Spojnicu svetelného zdroja a snímača nazývame triangulačná báza. Na strane zdroja je uhol zvieraný s triangulačnou bázou nemenný, avšak na strane snímača je uhol určený premennou pozíciou vysvieteného bodu CCD snímača. Z veľkosti tohto uhla a na základe znalosti triangulačnej bázy možno veľkosť súradnice z dopočítať. [7]

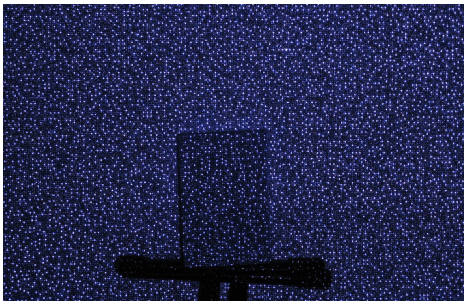


Obr. 2.9: Triangulačný trojuholník.

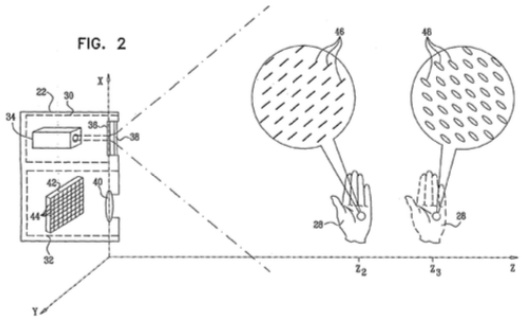
Svetelné zdroje, ktoré vnášajú dodatočnú informáciu do scény, možno rozdeliť nasledovným spôsobom:

- Svetelný lúč - 1D triangulácia
- Svetelný pruh - 2D triangulácia
- Štruktúrovaný svetelný zväzok - 3D triangulácia (technika moiré, technika svetelného vzoru, technika farebného kódu, technika fázového posunu)

Technológia od firmy PrimaSence, ktorú používa napríklad aj Kinect 360, je založená na špeciálnej astigmatickej šošovke s rôznou ohniskovou vzdialenosťou pre x-ový a Y-ový smer. Z premietaného kruhu sa potom stáva elipsa, ktorej orientácia je závislá na vzdialenosti od objektu. [11]



Obr. 2.10: Premietaný štruktúrovaný svetelný zväzok. [11]



Obr. 2.11: Technológia patentovaná spoločnosťou PrimaSence [11].

2.3.6 Analýza pohybu

Princíp väčšiny aplikácií, ktoré riešia počítačové videnie spočíva v nájdení, rozpoznaní a následnom sledovaní objektu, ktorý sa pohybuje po scéne.

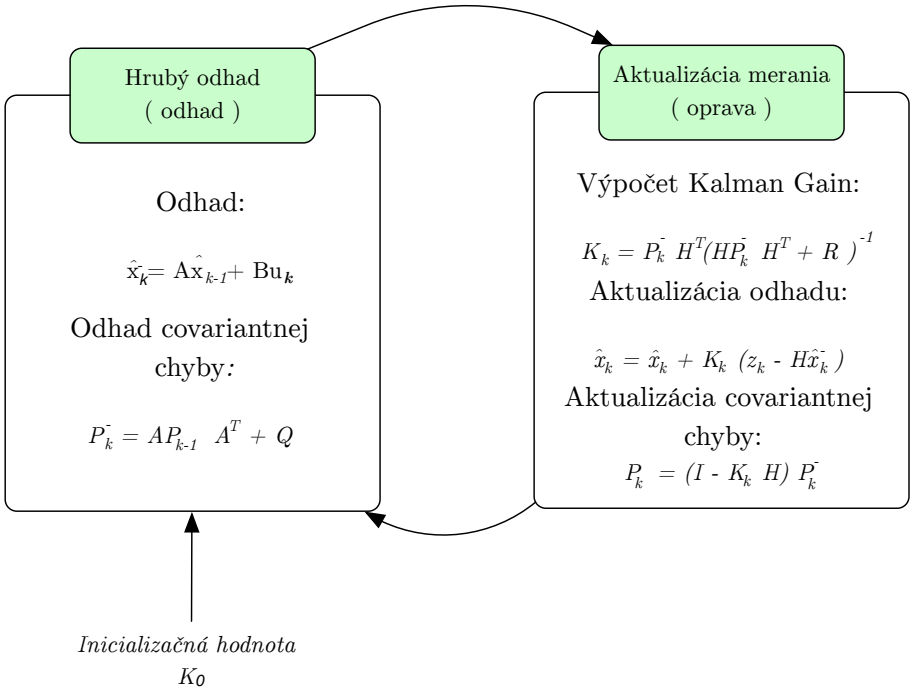
Veľmi používaná je detekcia samotného pohybu [9]. V tomto druhu aplikácií statická kamera sleduje určitú scénu a jej cieľom je zistiť nežiadaný pohyb na sledovanej scéne. Ide o veľmi jednoduché aplikácie implementované napríklad aj v bezpečnostných kamerách ako inteligentná spúšť nahrávania. Algoritmy nemajú za úlohu riešiť smer ani trajektóriu pohybu, sú schopné len vyrátať o koľko sa zmenila scéna oproti referenčnej vzorke. Jedna z

najpoužívanejších metód je **diferenčná metóda**, ktorá sa zaoberá detekciou, lokalizáciou a predikciou pohybujúceho sa predmetu. Okrem detekcie samotného pohybu, majú tieto metódy za úlohu zistiť aj trajektóriu a vyrátať predikciu ďalšej polohy objektu. Medzi najkomplikovanejšie situácie patria tie, pri ktorých sa naraz môže pohybovať kamera aj objekt súčasne. [9]

Kalmanov filter

Pri analýze pohybu objektu veľmi často dochádza k spájaniu, zatieneniu, či zmiznutiu sledovaného objektu na obraze. Preto musíme zabezpečiť, aby očakávaná hodnota odhadu bola rovná očakávanej hodnote stavu (potrebujeme priemernú hodnotu odhadovaného stavu). Je nežiadúce, aby odhadovaná hodnota bola posunutá nahor alebo nadol. Ďalšou požiadavkou je nájsť taký prostriedok pre odhad, ktorý má čo najmenšiu variáciu chyby, teda odhadovaná a následne nameraná hodnota by sa mala líšiť minimálne. [4]

Všetky tieto predpoklady spĺňa práve Kalmanov filter, avšak je možné ho použiť len vtedy, ak meraný systém je opísateľný iným lineárnym systémom. Je to z dôvodu, aby Kalmanov filter nebol ovplyvnený šumom. Kalmanov filter teda hľadá najoptimálnejším odhad budúceho stavu na základe stavov minulých a popisu systému. Existuje už vyše 50 rokov, no stále je jedným z najdôležitejším a najodporučanejším algoritmom. [4]



Obr. 2.12: Popis algoritmu Klmanového filtra. [3]

Hrubý odhad je predchádzajúci odhad polohy pred aktualizáciou merania. V časti aktualizácia merania už naozaj odhadneme nasledujúci krok. Z rovnice *Kalman Gain* alebo výpočet kalmanového zisku vyplýva, že ak je šum z merania R veľké, nebudeme dávať veľkú váhu meraniu pri výpočte ďalšieho kroku. Naopak ak je šum merania R malé, priradíme meranej hodnote veľkú váhu pri ďalšom kroku. [4]

Kapitola 3

Návrh riešenia problému

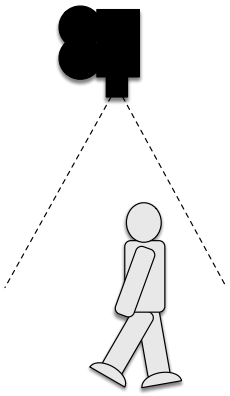
3.1 Existujúce problémy

Pred zahájením tvorby aplikácie je nutné oboznámiť sa s problémami. Na prvý pohľad sa môže zdať, že počítanie osôb je jednoduchý proces. Predpokladajme, že virtuálna brána je umiestnená pri vstupe do miestnosti. Proces prechodu osoby spočíva v zvýšení hodnoty počítadla vstupov alebo výstupov v závislosti na smere prechodu osoby. Systém musí byť navrhnutý tak, aby bol schopný detekovať a sledovať aj viacero ľudí, ktorí v jednom okamihu prechádzajú cez virtuálnu bránu rovnakým alebo rozličným smerom. Jeden z najväčších problémov je však dynamickosť prostredia, kde by aplikácia mala byť nasadená. Rýchle svetelné zmeny okolia, nepredvídateľné pohyby ľudí na scéne (osoba môže zastať, dotýkať sa, ťahať / tlačiť iný objekt) toto všetko je nutné riešiť tak, aby aplikácia dosiahla čo najväčšej presnosti.

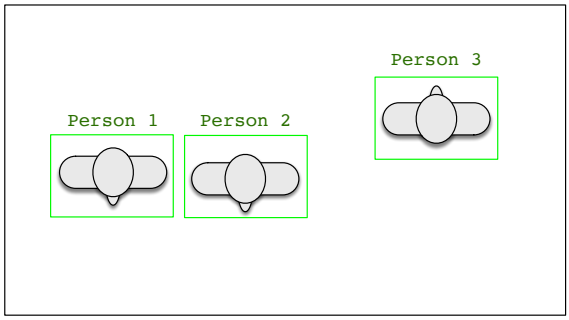
3.2 Konceptuálny návrh systému

Z problémov opísaných v sekcii 3.1 je jasné, že jednoduché systémy typu Infračervená závora, nebudú mať dostatočne veľkú presnosť pri zaznamenávaní prechodu viacerých osôb v jednom časovom okamihu. Je nutné oprieť sa o metódy, ktoré vedia zaznamenať viac informácii o objekte, ktorý prechádza vymedzeným priestorom (tvar, rýchlosť, veľkosť, smer, atď).

Jedným z najkomplexnejších prostriedkov, ako dostať dáta do počítača na spracovanie je kamera, ktorá nám poslúži ako senzor. Umiestnime ju na vrchnú priečku prechodu tak, aby snímala osoby z vrchu. Táto pozícia je najlepšia, lebo eliminuje možnosti zatienenia sledovaného objektu iným objektom na scéne.



Obr. 3.1: Poloha kamery.



Obr. 3.2: Obraz snímanej scény.

3.3 Výber hĺbkového snímača

Keďže koncept snímania priestoru sa opiera o snímanie z výšky, veľmi silným kandidátom sú aktívne 3D kamery. Veľká výhoda je meranie na základe tvorby hĺbkového obrazu (sekcia 2.3.5). Vďaka tomu, má meranie vysokú presnosť, nepotrebuje osvetlenie priestoru ani kontrastné prostredie. Ďalšou veľkou výhodou je informácia o výške osoby, ktorá prechádza. Táto informácie je použiteľná v mnohých aspektoch ďalšieho spracovania. Na druhej strane technológia aktívneho 3D hĺbkomeru je náchylná na priame slnečné lúče. Tie dokážu merania veľmi znehodnotiť. Táto práca sa zaoberá testovaním troch rôznych aktívnych hĺbkomerov:

- Kinect 360
- Orbbec Astra S
- Intel RealSence SR300

3.3.1 Základné informácie o testovaných hĺbkomeroch

Kinect 360

Kamera sníma frekvenciou 30Hz. Hĺbka je snímaná s presnosťou na 11 bitov, čiže hĺbka jedného pixlu môže nadobudnúť hodnoty od 0 do 2.047. Pracovný rozsah senzora je 1,2 - 3,5 metra. Využíva technológiu vyvinutú firmou PrimaSence (sekcia 2.3.5).

Keďže od spoločnosti Microsoft neexistuje oficiálna podpora operačných systémov Unixového typu, na čítanie obrazu z kamery bola použitá knižnica *libfreenect*[1], ktorá je vyvíjaná a podporovaná komunitným spôsobom. Bola testovaná na *Raspberry Pi verzie 3*[2] s operačným systémom Rasbian a MacOS Sierra. Použitá knižnica fungovala spoľahlivo. Na Raspberry Pi občasne vznikala menšia latencia spôsobená nižším výkonom (príloha C).

Výrazným problémom pri používaní tohto hĺbkomeru je jeho veľkosť, váha, nutnosť externého napájania. Ďalší problém sa týka dostupnosti senzora. Senzor je zastaralý a už sa nevyrába. Cena sa momentálne stále drží na úrovni 100 EUR.

[1] Knižnica dostupná na GitHub.com https://github.com/OpenKinect/libfreenect.git
[2] Malý počítač založený na ARM technológii https://www.raspberrypi.org

Orbbec Astra S

Ide o kameru, ktorá má veľmi podobný princíp funkčnosti ako kinect, čiže technológia je založená na vysielaní štrukturovaného svetla. Pracovný rozsah senzora je 0.4 – 2 metra. Kamera je pomerne malá a nepotrebuje externé napájanie. Používa USB 2.0 rozhranie.

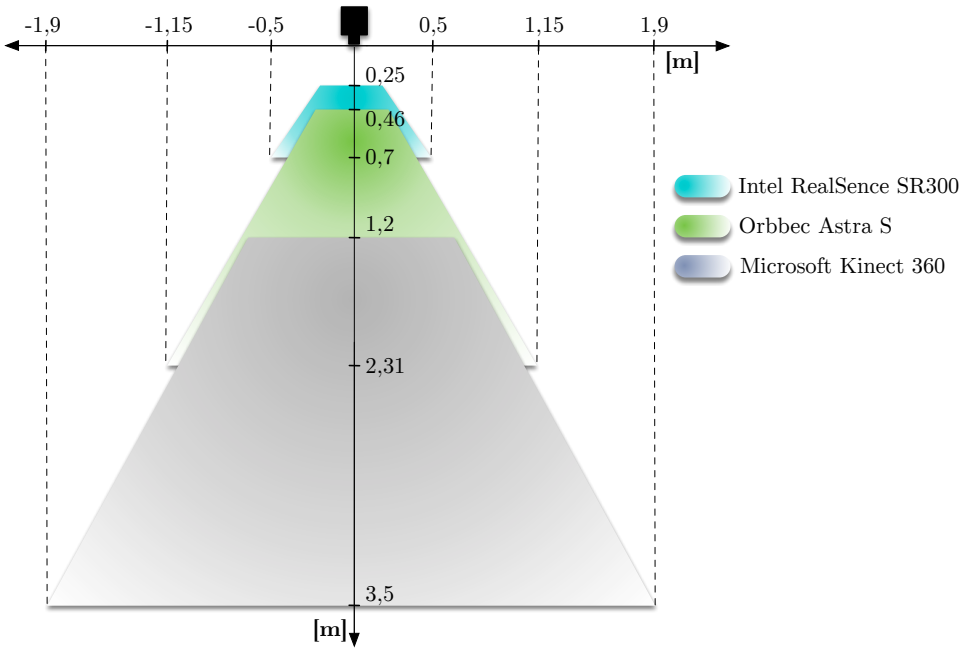
Knižnice od spoločnosť Orbbec má veľmi dobrú podporu všetkých operačných systémov (príloha A). Cena 150 EUR.

Intel RealSence SR300

Je kamera, ktorá sa pomerne dosť líši od predchádzajúcich dvoch. Hlavným rozdielom je, že používa USB 3.0, čo znamená, že posiela väčšie množstvo dát a nie je možné ho pripojiť na *RasperryPi*. Technológia snímania je založená na štrukturovanom svetle, podobne ako v predchádzajúcich dvoch prípadoch. Pracovný rozsah kamery je 25 až 70 cm. Cena kamery je tiež okolo 150 EUR. (príloha B)

3.3.2 Porovnanie zorného poľa kamier

Prvým parametrom porovnávania je **zorné pole kamier**. Každá hĺbková kamera, podobne ako klasické RGB kamery, má zorný uhol daný použitou optikou kamery. Hĺbkomery však majú navyše atribúty maximálnej a minimálnej vzdialenosti, na ktoré sú schopné vykonať meranie. Je to z dôvodu primárneho použitia kamery, pre ktoré ho výrobca vyrába. Napríklad *Kinect 360* je určený pre hernú konzolu a detekciu celého tela, preto ma zorné pole veľmi veľké, aby mal hráč aj istú dávku voľnosti v pohybe. Presný opak kamery Kinect predstavuje *Intel RealSence*, ktorá je určená na aplikácie ovládania PC rôznymi gestami. Z dôvodu takéhoto obmedzenia nie je možné postaviť aplikáciu sledovania prechodov ľudí na jednom type kamery.



Obr. 3.3: Vizualizácia vertikálneho zorného pola hĺbkomerov.

Fyzické okolie virtuálnej brány nemusí dovoľovať umiestniť hĺbkomer do požadovanej snímacej výšky. Preto najlepším spôsobom je vyberať senzor špeciálne pre každé miesto nasadenia. Vďaka tomu je možné realizovať meranie od 2m do 4,5m.

3.3.3 Testovanie kvality snímania

Kvalita snímaného hĺbkového obrazu je veľmi dôležitým aspektom pre presnosť samotnej aplikácie. Keďže všetky testované hĺbkomery sú založené na technológii aktívnej triangulácie (kap 2.3.5), v rôznych svetelných podmienkach môže kvalita vytvorených hĺbkových záznamov výrazne kolísať v závislosti na kamere. Z toho dôvodu boli vykonané testy, na základe ktorých je možné porovnanie jednotlivých snímačov.

Predpokladajme, že každý bod scény, ktorý sa nachádza v zornom poli kamery sa premietne do šedotónového obrazu, v ktorom intenzita každého pixlu nadobúda hodnoty patriace do intervalu <0, 254> v závislosti na vzdialenosti. Body, ktoré nebolo možné zmerať (nekonečne ďaleko, šum, materiál alebo tvar objektu atd...) nadobúdajú hodnotu 255 (biely pixel). Potom metódu, ktorá zráta priemerný počet pixlov s hodnotou 255 na snímku, je možné použiť na získanie **metriky** pre porovnanie. Avšak, *metóda pre výpočet hodnotiacej metriky konverguje k objektívnemu výsledku len vtedy, ak sa každý objekt scény nachádza v zornom poli kamery, ktoré udáva jej výrobca* (príloha E).

Priebeh testovania

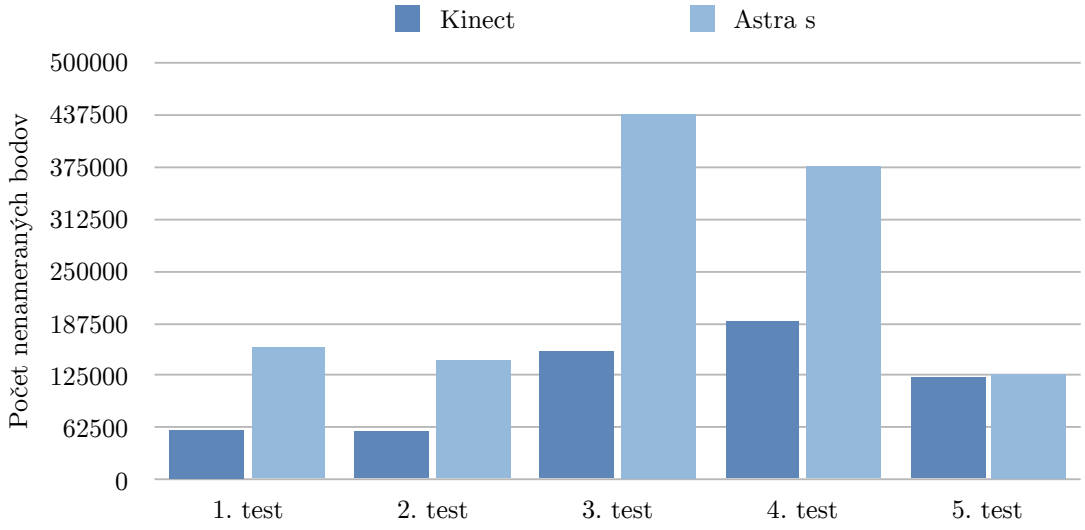
Testovanie spočívalo v umiestnení kamier na náhodné miesto, kde sa v jednom okamihu nahrával ten istý bod na všetkých testovaných kamerách. Vďaka tomu je zaručené, že všetky kamery boli vystavené rovnakým podmienkam merania. Následne sa na vzniknuté záznamy aplikovala metóda pre výpočet metriky. Meranie sa konalo v piatich opakovaniach:

1. test - pôda fakulty za denného svetla v zamračenom počasí
2. test - pôda fakulty za denného svetla s tlmeným slnečným žiarením
3. test - exteriér za denného svetla za slnečného počasia (v tieni)
4. test - exteriér za denného svetla za slnečného počasia s objektami v scéne
5. test - interiér za umelého osvetlenia

Výsledky testov

Pri testovaní kamery *Astra S* bola odhalená chyba. Kamera nie je schopná merať ľavý kraj o šírke 50 pixlov. Aby nedošlo k zkresleniu v meraní, súčasťou hodnotiacej funkcie bolo odrezanie chybnej časti obrazu zo všetkých testovaných záznamov (aj na záznamoch z kamery Kinect).

Ďalšia komplikácia nastala s kamerou *Intel RealSence SR300*. Vzhľadom na jej extrémne malé zorné pole, sa nepodarilo splniť podmienku konvergencie hodnotiacej metódy. Ďalším problémom tejto kamery je nekompatibilita s Raspberry PI z dôvodu absencie USB 3.0. Z uvedených dôvodov táto kamera nebola zaradená do výsledkov testov.



Obr. 3.4: Vizualizácia vertikálneho zorného pola hĺbkomerov.

Z výsledkov vyplýva, že *Kinect 360* je výrazne odolnejší v prípadoch interferencie denného svetla. Pri testoch za umelého alebo žiadneho osvetlenia je kvalita oboch kamier porovnateľná.

Kapitola 4

Implementácia

Nasledujúca kapitola sa bude zaoberať popisom implementácie dvoch programov pre snímanie prechodov ľudí cez virtuálnu bránu. Prvý program je založený na základe snímania hĺbkovej mapy pomocou hĺbkomeru a druhý na použití bežnej RGB web kamery.

4.1 Programovací jazyk a využité knižnice

Pri implementácii oboch programov bol zvolený programovací jazyk *Python 3*[1] v spojení s knižnicou *OpenCV*[2]. Je to knižnica vytvorená pre počítačové videnie a strojové učenie, distribuovaná pod BSD licenciou. Kladie veľký dôraz na aplikácie bežiace v reálnom čase.

4.2 Výpočetná základňa aplikácie

Z dôvodu minimalizácie ceny zariadenia bola zvolená ako primárna platforma **RasperryPi vo verzii 3**.

Ide o malý počítač, za ktorým stojí britská nadácia s rovnomenným názvom, Raspbery PI. Je postavený na integrovanom Broadcom SoC procesore BCM2837, ktorý má štyri 1.2 GHz 64-bitové ARM Cortex-A53 jadrá. Ďalšími parametrami sú 1 GB RAM, 100 Mbps Ethernet port, štyri USB 2.0 porty, HDMI výstup. Cena sa pohybuje okolo 35 EUR.

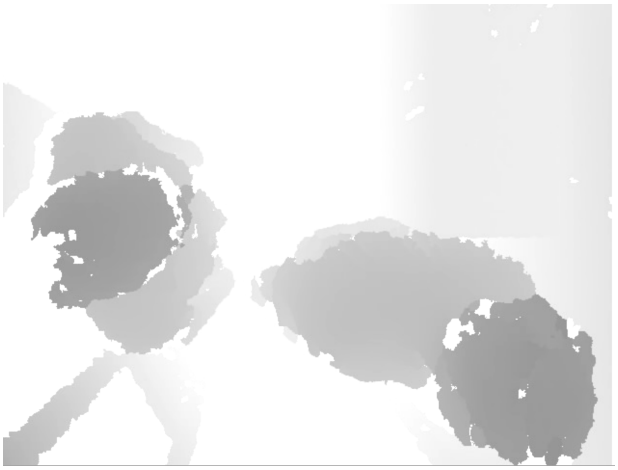
Avšak z dôvodu absencie USB 3.0 boli pri realizácií využité aj iné počítačové platformy.

4.3 Implementácia aplikácie s využitím hĺbkového snímania

Vstupom algoritmu je séria šedotónových hĺbkových obrazov, kde hodnota každého pixlu vyjadruje vzdialenosť od objektu v scéne. Takýto obraz sa nazýva *surová hĺbková mapa* (obrázok 4.1). Program tento obraz ďalej spracováva v nasledujúcich dôležitých etapách:

- Prahovanie
- Nájdenie kontúr
- Priradenie kontúr objektom
- Sledovanie objektu

[1] Skriptovací jazyk https://www.python.org
[2] Computer vision knižnica http://opencv.org



Obr. 4.1: Surová hĺbková mapa.

4.3.1 Prahovanie

Prahovanie je prvý krok algoritmu. Výstupom je binárna reprezentácia obrazu, kde záujmové body sú reprezentované log 1 a body, ktoré sú nezaujímavé (predmety nemmené, súčasť scény) reprezentované log 0 (obrázok 4.2). Na vstupe fázy algoritmu je surová hĺbková mapa, kde každý pixel predstavuje vzdialenosť od objektu scény. Vďaka tomu môžeme využiť jednoduchý algoritmus globálneho prahovania (kapitola 2.3.3), kde prahová hodnota reprezentuje minimálnu výšku pixlov, ktoré je možno označiť za záujmové. Táto hodnota je súčastou konfiguračných hodnôt, ktoré je nutné nastaviť pri inštalácií aplikácie na prevádzkové miesto. Na prahovanie program využíva funkciu z knižnice opencv *cv2.threshold(grayFrame, MIN_HEIGHT, 255, cv2.THRESH_BINARY_INV)* kde *grayFrame* je snímka hĺbkovej mapy *MIN_HEIGHT* je hodnota minimálnej výšky. Ďalší argument je hodnota logickej 1 vo výslednej binárnej maske a posledný argument je podmienka určenia $log1$ alebo $log0$.

Samostatným problémom metódy globálneho prahovania je, že v scéne môžu existovať predmety, ktoré sú umiestnené v podobnej výške a metóda nie je dostatočná na ich odfiltrovanie. Za týmto účelom sa pri štarte programu vytvorí referenčný snímok scény a pred samotným použitím prahovania sa vyráta absolútny rozdiel medzi referenčným a aktuálnym snímkom. Opísaná funkčnosť je zaistená volaním funkcie *cv2.absdiff(gray_frame, bg_reference)*.

4.3.2 Nájdenie kontúr

Cieľom tejto časti spracovania je nájsť kontúry, ktoré korelujú s osobami prechádzajúcimi cez virtuálnu bránu. Vstupom funkcie je binárny obraz vytvorený v 4.3.1 a výstupom sú kontúry a ich ťažiská. Činnosť tejto fázy možno rozdeliť do nasledujúcich bodov:

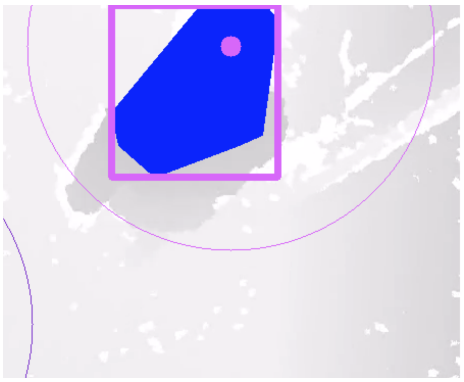
- Nájdenie kontúr na základe metódy sledovania hranice (sekcia 2.3.3)
- Spájanie roztrieštených kontúr
- Filtrácia nevyhovujúcich kontúr
- Výpočet najvyššieho bodu

Hľadanie a spájanie kontúr

Na základe binárneho obrazu, aplikujeme metódu sledovania hraníc *cv2.findContours*, ktorá je súčastou openCV knižnice. Jej výsledkom je zoznam všetkých uzavretých blobov (kontúr), ktoré sa v binárnom obraze nachádzajú. V dôsledku šumu a nedokonalého snímania hĺbkovej kamery však dochádza k vzniku falošných kontúr a rozpadu objektu na veľké množstvo malých kontúr, ktoré ho reprezentujú. Preto je nutné nájsť spôsob ako tieto chyby napraviť.

Spájanie kontúr - rekurzívny algoritmus, ktorý má za úlohu prehľadať dáta a spojiť kontúry, ktoré potencionálne reprezentujú jeden objekt.

Obr. 4.2: Roztrieštená binárna maska.



Obr. 4.3: Spojená kontúra.

Popis algoritmu, pričom všetky kontúry nájdené prostredníctvom funkcie metódy sledovania hraníc, sú uložené v zozname *allCont*:

1. Zo zoznamu *allCont* odstránime všetky kontúry, ktoré majú nulový obsah.
2. Vytvoríme nový zoznam *newContura*, vyberieme prvú kontúru z *allCont*, vložíme do zoznamu *newContura* a označíme ju ako *root*.
3. V zozname *allCont* nájdeme všetky kontúry vyhovujúce podmienke minimálnej absolútnej vzdialenosti ťažísk kontúr oproti kontúre označenej ako *root*.
4. Nájdené položky vložíme do zoznamu *newContura*, ostatné označíme ako nový zoznam *allCont*.
5. Pre každú najdenú kontúru označíme ako *root* a pokračujeme bodom 3.
6. Všetky kontúry v zozname *newContura* spojíme do jednej kontúry.
7. Pre každú konturu v zozname *allCont* pokračujeme bodom 2.

Inými slovami, algoritmus prechádza všetky kontúry a na základe vzdialenosti ťažiskových bodov ich spája do jedného celku.

Spájanie zoznamu kontúr je realizované prostredníctvom volania funkcie *cv2.convexHull*, ktorá pre množinou bodov, ktoré sú obrysové body oblasti, nájde inú množinu bodov reprezentujúcich jej konvexný obal.

Filtrácia nevyhovujúcich kontúr

Keďže poznáme približnú veľkosť kontúry, ktorej detekcia je cieľom záujmu, všetky ostatné sú nežiaduce a je nutné ich odstrániť. Na základe tejto filtrácie sa program zbaví všetkých náhodných oblastí, ktoré vznikli pôsobením šumu. Túto filtráciu je však možné použiť až po aplikovaní algoritmu pre spájanie kontúr. Je to posledná fáza segmentácie obrazu.

Výpočet najvyššieho bodu

Vďaka tomu, že technológia snímania vytvára hĺbkovú mapu, je možné v rámci každej kontúry definovať lokálne maximum každej oblasti. Pri zvolenej koncepcii snímania, bod patrí do množiny bodov nachádzajúcich sa na vrchnej časti hlavy. Tento bod je významný pri sledovaní pohybu objektu.

Proces nájdenia je definovaný postupom týchto krokov:

1. Vytvoríme prázdnu **masku** veľkosti snímaného obrazu *np.zeros*.
2. Do **masky** vložíme kontúru, ktorej maximálnu oblasť chceme nájsť *cv2.drawContours*.
3. Nájdeme minimálnu, respektíve maximálnu hodnotu reprezentujúcu najvyšší bod kontúry *cv2.minMaxLoc(originalFrame, mask=mask)*.
4. Nájdenú hodnotu použijeme ako prahovú hodnotu a aplikujeme funkciu prahovania s globálnym prahom na originálny obraz. Výsledkom je binárna maska *cv2.threshold*.
5. Medzi maskou vytvorenou v bode 2 a maskou vytvorenou v bode 4 vykonáme logickú operáciu AND, čoho výsledkom je binárna maska maximálnej oblasti skúmanej kontúry *cv2.bitwise_and(mask, thresh)*.
6. Následne môžeme znovu vyhľadať všetky uzavrené oblasti, ktoré vznikli na obraze *cv2.findContours*.
7. Vyberieme kontúru s najväčším obsahom, pre ktorú vyrátame ťažisko a nájdený bod prehlásime za maximálny bod oblasti.

4.3.3 Priradenie kontúr objektom

V prípade, že sme schopní v každom obraze, ktorý patrí do konštantného toku dát zo snímača vykonať spoľahlivú segmentáciu, je možné postúpiť na objektový level abstrakcie. Objekt je určený sériou rôznych kontúr, medzi ktorými existuje vzájomná závislosť.

Proces vytvárania a zanikania kontúr aplikácie je založený na dvoch hlavných vlastnostiach:

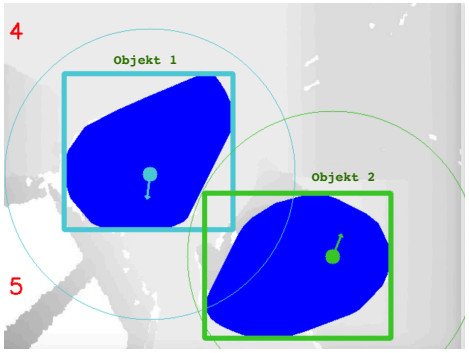
- Pozícia oblasti
- Hodnota výškového maxima oblasti

Pre každý existujúci objekt na scéne sa vypočíta absolútna vzdialenosť ku všetkým nájdeným oblastiam aktuálneho obrazu. Následne sa vzdialenosti usporiadajú od najmenšej po najväčšiu a odstránia všetky, ktoré sú väčšie ako hraničná hodnota stanovená pri konfigurácii. Priradenia, ktoré ostali v zozname sa aplikujú a aktualizuje sa poloha objektov. V prípade, že kontúre nebol priradený žiadny objekt, vytvorí sa nový a kontúra sa priradí ako inicializačná poloha nového objektu.

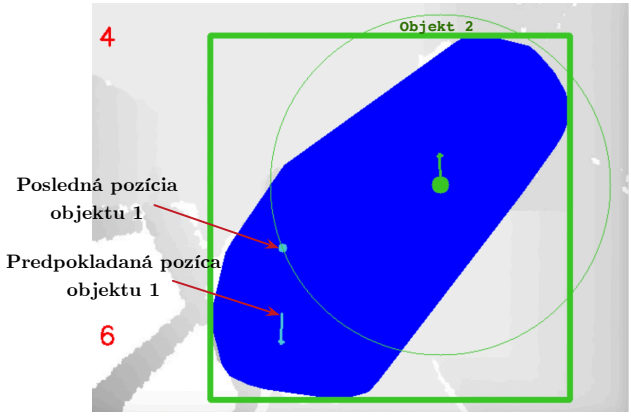
4.3.4 Sledovanie objektu

Pri sledovaní osôb v pohybe dochádza k rôznym problémom, ktorým je nutné venovať veľkú pozornosť. Najčastejšie riešeným problémom je jednoznačne určiť, ktorá **kontúra bude priradená ktorému objektu**. Vzhľadom na to, že priemerný tok dát z kamier je 15 - 20 snímkov za sekundu, zmeny polohy objektov medzi dvoma za sebou idúcimi snímkami môžu byť natoľko veľké, že sledovanie zlyhá. Druhým vážnym problémom je **spojenie dvoch kontúr do jednej**, čo spôsobuje vytlačenie jedného z objektov, ktorý stratí kontúru, ktorú by mohol sledovať. Táto situácia môže nastať fyzickým kontaktom dvoch osôb (veľmi blízky prechod, zrazenie sa a iné...). Oba problémy majú spoločné riešenie a tým je predikcia nasledujúcej polohy na základe smeru a rýchlosti, ktorú objekt nadobudol do momentu incidentu. V rámci implementácie som otestoval dva algoritmy.

- Kalmanov filter
- Výpočet na základe histórie a rýchlosti



Obr. 4.4: Moment pred kolíziou dvoch kontúr.



Obr. 4.5: Využitie predpovedania polohy.

Kalmanov filter - Teoretický základ funkčnosti tohto filtra bol opísaný v sekcii 2.3.6. Pri implementácii som využil knižnicu *openCV*, ktorá ho obsahuje. Pri testoch som však objavil veľký problém. Kalmanov filter potrebuje pomerne veľkú históriu korekcií na to, aby získal požadovanú presnosť. Priemerný prechod jedného človeka cez virtuálnu bránu, je zachytený na 10 - 20 snímkoch. Podľa testov Kalmanov filter potrebuje na získanie základnej presnosti 5 - 6 korekcií. Tento problém je samozrejme možné riešiť zmenou snímacieho zariadenia alebo umiestnením kamery na vyššie položené miesto, a tým maximalizovať zotrvanie objektov na scéne.

Výpočet na základe histórie a rýchlosti - Táto metóda je veľmi jednoduchá a opiera sa o vedomosti zo základnej školy. Pre každý objekt vytvoríme históriu, ktorá obsahuje pozíciu a časovú známku. Veľkosť histórie definujeme podľa toho, koľko bodov má ovplyvňovať predikciu nasledujúceho bodu. Proces výpočtu predikcie potom vyzerá tak, že najprv vyrátame moment rýchlosti pre x-ovú aj y-ovú os:

$$v_x = \frac{last_x - first_x}{t} \qquad (4.3.1)$$

$$v_y = \frac{last_y - first_y}{t} \qquad (4.3.2)$$

kde *last* je najstaršia známa poloha, *first* je najnovšia pozícia a t je celkový čas od najstaršieho po najnovší záznam v histórii. Keď máme vypočítané rýchlosti, máme všetko pre určenie predikcie na základe aktuálneho času takto:

$$predicate_x = last_x + v_x * \Delta t \quad (4.3.3) \qquad predicate_y = last_y + v_y * \Delta t \quad (4.3.4)$$

kde, Δt je rozdiel medzi časom hľadanej polohy a časom poslednej známej polohy. Tento výpočet funguje spoľahlivo už pri druhom snímku a pri testoch dosahoval lepšie výsledky ako Kalmanov filter. Z toho dôvodu bol na predikciu polohy využitý práve tento algoritmus.

4.4 Implementácia aplikácie s využitím RGB web kamery

Implementácia aplikácie s využitím RGB web kamery využíva veľmi podobné stratégie ako aplikácia založená na hĺbkovom snímaní, ktorá bola opísaná vyššie. Dva najvýznamnejšie rozdiely predstavujú:

- Spôsob vytvárania binárnej masky
- Určenie významného bodu oblasti - ťažisko oblasti

4.4.1 Binárna maska

Princíp snímania webkamery je založený na interakcii CCD čipu so svetlom prichádzajúcim od pozorovaného objektu. Odrazom svetla od povrchu sa môže odraziť v plnom spektre pod rovnakým uhlom (zrkadlo, lesklé objekty), alebo svetlo prenikne do objektu, kde sa časť vlnovej dĺžky sa pohltí a zvyšok sa odrazí. Senzor teda vníma spektrum odrazeného svetla ako určitú farbu.

Ak by sme aj v tomto prípade chceli využiť **metódu prahovania**, museli by sme zabezpečiť, aby priestor snímaný kamerou bol vždy farebne iný od objektov prechádzajúcich skrz. Tak by bolo možné odfiltrovať všetky pixle s vlnovou dĺžkou odrazeného svetla (farbou) zodpovedajúcou pozadiu. Ďalej by bolo nutné zabezpečiť homogénne osvetlenie scény, aby bol zabezpečený dostatok svetla. Vzhľadom na všetky tieto problémy je metóda nepoužiteľná pre daný účel.

Výhodnejšou stratégiou predstavuje použitie **metódy odčítavania pozadia** (*Background Subtractor*). Je to analytická metóda, pre postupné učenie sa pozadia a jeho odčítavanie od obrazu v popredí. Výsledkom je binárna maska obrazu, kde pixel, ktorý sa zmenil má logickú hodnotu 1 a pixel, ktorý je pozadím 0.

Požiadavky na výkon

Algoritmy pre odčítavanie pozadia sú veľmi náročné na výpočtový výkon a to priamoúmerné rozlíšeniu spracovávaného obrazu. V rámci implementácie boli testované MOG[3], MOG2[4], KNN.

Najlepší pomer kvality binárnej masky a výkonu potrebného na jej získanie má *KNN (K-nearest)*. Jeho implementáciu obsahuje knižnica *cv2.createBackgroundSubtractorKNN()*. Je veľmi efektívny, ak sa v popredí obrazu mení len jeho malá časť.

[3]Odkaz na publikáciu: http://link.springer.com/chapter/10.1007%2F978-1-4615-0913-4_11
[4]Odkaz na publikáciu: http://ieeexplore.ieee.org/document/1333992/

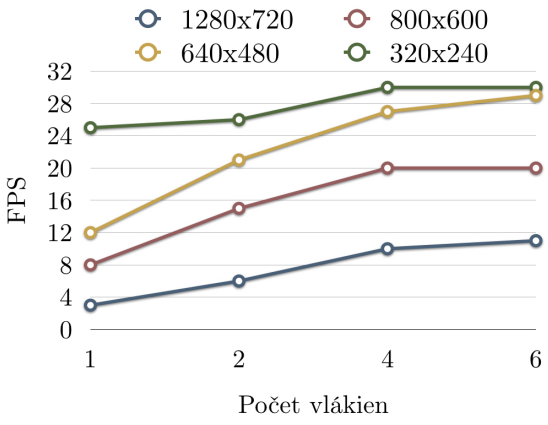
Paralelné spracovávanie

Pri implementácii ďalších testoch sa však ukázalo, že pri spustení aplikácie na Rasperry PI rýchlosť spracovávania jednotlivých obrazov bola nedostatočná (8-10 FPS). Pre zvýšenie priepustnosti spracovávania bolo nutné zlepšiť vyťaženie viacerých jadier procesora súčasne (asynchrónny prístup).

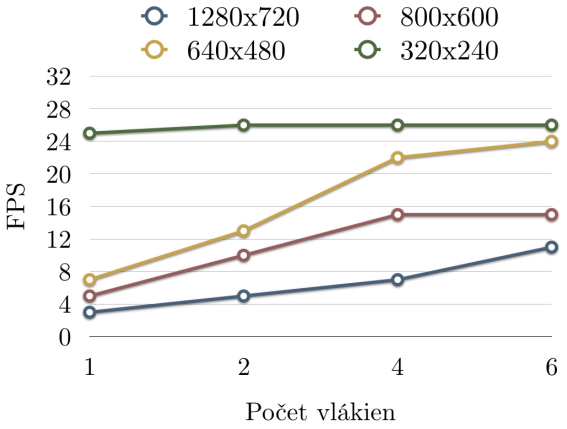
Implementácia asynchrónneho spracovania funguje na princípe generácie množstva vlákien, ktoré sú spolu synchronizované na základe zámkov. V prvom kroku algoritmu sa inicializuje *videostream* a *background substractor*. Následne sa vygeneruje N vlákien (*workers*). Každé vlákno obsahuje jeden zámok na čítanie ďalšieho snímku obrazu a druhý zámok pre zápis do zoznamu spracovaných snímkov. Tie sa pri vytváraní uzamknú. Každé vlákno v nekonečnom cykle vykonáva:

1. Uzamknutie zámku pre čítanie ďalšieho snímku
2. Čítanie ďalšieho snímku
3. Otvorenie zámku pre čítanie nasledujúceho vlákna *(i + 1)*
4. Zahájenie výpočtu Bacground substractoru
5. Uzamknutie zámku pre zápis výsledku do zoznamu výsledkov
6. Vloženie výsledku operácie do zoznamu výsledkov
7. Otvorenie zámku pre vloženie výsledku ďalšieho vlákna *(i + 1)*

Algoritmus bol testovaný na *Raspberry PI 3*. Nasledujúce grafy prezentujú závislosť rýchlosti spracovávania snímkov na veľkosti snímaného obrazu a počte vlákien algoritmu:



Obr. 4.6: Nemenná scéna.



Obr. 4.7: Priemerne veľký pohyb na scéne.

Z testov vyplýva, že algoritmus najlepšie pracuje pri snímacom rozlíšení **640*480 pixlov** s využitím **štyroch asynchrónnych vlákien**. Pri týchto atribútoch dosiahnutá hodnota snímkov za sekundu (FPS) aj veľkosť rozlíšenia dostatočná pre ďalšie spracovanie.

Kapitola 5

Výsledky práce a možnosti rozšírenia

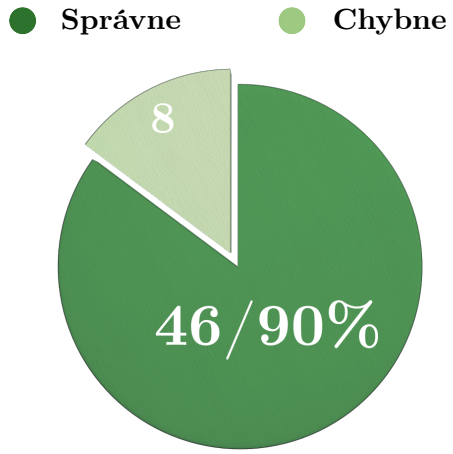
5.1 RGB kamera vs hĺbkový senzor

Jedným z hlavných cieľov tejto práce je porovnať využitie 3D a 2D technológie pri sledovaní prechodov osôb cez virtuálnu bránu. Za týmto účelom boli vytvorené dva programy, ktoré sa podrobili dôkladným testom, ktorých výsledky boli následne porovnané.

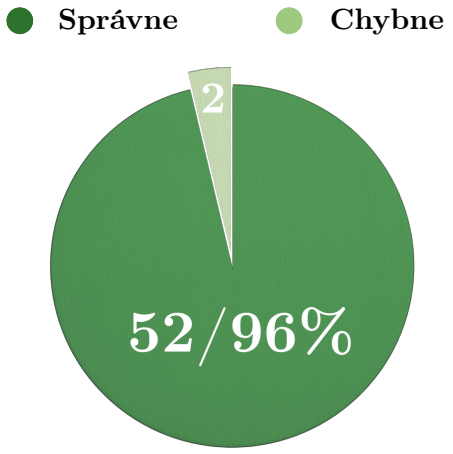
5.1.1 Testovanie v reálnom prostredí farmaceutickej firmy

Za účelom testovania aplikácie boli vytvorené dva 3 - 4 hodinové záznamy zo špedičnej prevádzky farmaceutickej firmy. Zariadenie bolo umiestnené na miesto frekventovaného prechodu, ktorý predstavoval spojovacie dvere medzi dvoma oddeleniami. Zamestnanci prechádzali osobitne, ale aj v skupinách a väčšina v rukách alebo na ručnom vozíku prenášala tovar, ktorý predstavoval kopy krabíc naukladaných na sebe.

Obe kamery boli umiestnené podľa konceptu (kapitola 3.2) tak, aby snímali približne rovnaký bod prechodu. Jednotlivé programy uspeli nasledovne:

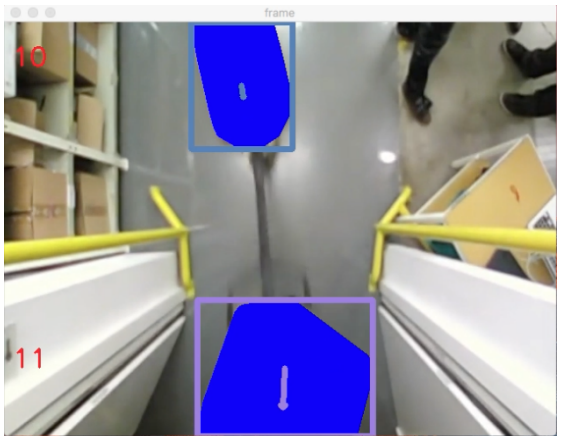


Obr. 5.1: 2D úspešnosť detekcie.



Obr. 5.2: 3D úspešnosť detekcie.

Všetky chyby spôsobené v programe používajúci **2D snímanie scény** boli spôsobené vozíkom, ktorý človek ťahal za sebou ako je znázornené na obrázku.

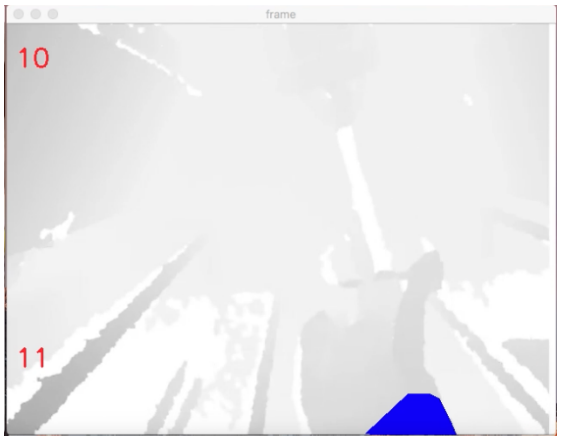


Obr. 5.3: Segmentácia snímku programom.

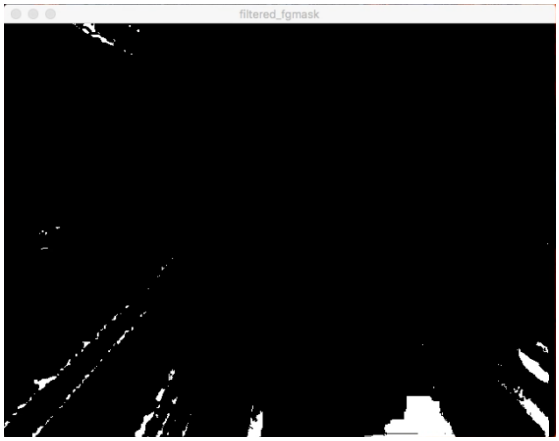
Obr. 5.4: Binárna maska zo substractora.

Na vyriešenie opísanej chyby je nutné disponovať väčším množstvom informácií z obrazu. S použitou technológiou nie je možné spoľahlivo rozoznať, či sa jedná o vozík, ktorý je ťahaný za človekom alebo o ďalšieho človeka kráčajúceho za ním. V prípade, že zamestnanec vozík tlačil pred sebou, program dokázal jednotlivé kontúry pospájať do jednej veľkej a tak človek aj vozík predstavovali jeden objekt. Ďalším zdrojom chýb bol extrémne naplnený vozík, ktorého náklad bol tak vysoký, že pri prechode zabral celý obraz snímaný kamerou. Tento prípad sa programu nepodarilo detekovať a odstrániť.

V rovnakých podmienkach si program, ktorý používa **3D metódu snímania** poradil o niečo lepšie. Na 54 veľmi náročných prechodoch, situáciu vyhodnotil zle len 2x. Obe chyby boli spôsobené už spomínaným vozíkom s veľmi vysoko naloženým nákladom. Avšak ten istý vozík prechádzal štyrikrát, čiže v polovici prípadov si program využívajúci 3D poradil aj s touto hraničnou situáciou na rozdiel od programu využívajúceho 2D snímanie, ktorý zlyhal vo všetkých štyroch prípadoch.



Obr. 5.5: Segmentácia snímku 3D programom v rovnakej situácií ako obrázok 5.3.



Obr. 5.6: Binárna maska zo substractora v rovnakej situácií ako obrázok 5.4.

5.1.2 Dlhodobé testovanie v simulovaných podmienkach

V priebehu vývoja oboch algoritmov, vzniklo veľké množstvo videí, na ktorých boli simulované rôzne situácie, ktoré v dynamickom prostredí virtuálnej brány môžu nastať. Ako napríklad prechod jednej, dvoch alebo viacerých osôb jedným alebo opačným smerom, zrazenie sa dvoch a viacerých ľudí, "motanie sa"v priestoroch virtuálnej brány, prebehnutie, skákanie a iné situácie. S využitím všetkých záznamov bola vypočítaná úspešnosť oboch algoritmov na vzorke stoviek prechodov.

- Program, ktorý na snímanie využíva **2D snímanie priestoru** dosiahol úspešnosť na hranici **94%**
- Program, ktorý na snímanie využíva **3D snímanie priestoru** dosiahol úspešnosť **98%**

Hlavným dôvodom výraznejšieho úspechu 3D technológie je stabilnejšie určenie primárneho bodu v kontúre (hlava). Vďaka tomu, predikčný algoritmus pracoval s presnejšími hodnotami a v prípade náhlej straty kontúry objektu, dokázal s veľkou úspešnosťou opätovne kontúru nájsť na základe predikcie nasledujúceho pohybu. Pri 2D technológií obrazu je veľmi náročné určiť stabilný bod. Preto sa algoritmus opiera o hodnotu ťažiska jednotlivých kontúr. Tvar a veľkosť kontúry je však veľmi menná, čo spôsobuje, že aj poloha ťažiska nemá lineárny priebeh (vysoký šum). Ďalšou výhodou 3D technológie, je možnosť odfiltrovať objekty, podľa ich výšky.

5.1.3 Bezpečnostné problémy

Porovnanie v tejto kategórií je veľmi prínosné vzhľadom na ďalšie využitie v bezpečnostných aplikáciach. Obe technológie snímania majú svoje silné a slabé stránky.

3D technológia

Prvý problém predstavuje situácia, ak je stanovený výškový prach detekcie objektov (prípad s vozíkmi) alebo hĺbkový snímač nemá dostatočne veľký dosah snímania (až po zem). V týchto prípadoch môže útočník podliezť tento prah snímania. Ďalším potenciálnym problémom je, že hĺbkové senzory nevedia vykonať meranie oproti istému druhu materiálov, ako napríklad sklo. Na druhej strane sú však mnohé výhody, ako napríklad nezávislosť na osvetlení priestoru. Snímanie funguje aj za úplnej tmy.

2D technológia

Slabinou tohto systému je samotný background substractor, ktorý pracuje na princípe učenia sa pozadia. To znamená, že ak sa potencionálny útočník postaví do obrazu a bude tam stáť dostatočne dlho, systém sa ho naučí ako súčasť pozadia. Takto môže postupne prejsť celý prechod. Ďalším všeobecným problémom technológie je svetlo. Prechod musí byť vždy dobre osvetlený (homogénne svetlo), inak detekcia nie je možná.

5.1.4 Zhodnotenie

Systém založený na 2D snímaní pomocou kamery je vhodný pre aplikácie, kde nie je požadovaná extrémna presnosť, ale nízka cena. Príkladom môže byť obchod, ktorý si vedie štatistiku návštev. Na druhej strane 3D snímanie pomocou hĺbkomerov je vhodné pre bezpečnostné aplikácie súvisiace s objektovým zabezpečením. Výhodou je pridaný údaj o hĺbke, vysoká presnosť snímania, spoľahlivosť.

5.2 Možnosti rozšírenia

Najvýznamnejšou možnosťou rozšírenia projektu je využitie aplikácie ako základ pre objektovú bezpečnosť. Ďalším krokom vpred, by malo byť priradenie identity jednotlivým objektom. To možno dosiahnuť rôznym spôsobom:

- *Vysoko-výkonového RFID* - jednalo by sa teda o identitu na základe vlastníctva (čip). Tento spôsob však prináša technologické problémy, keďže by bolo veľmi ťažké zabezpečiť, aby RFID antény príjmali signál len od čipov nachádzajúcich sa v priestore brány.
- *Biometrické systémy* - na základe tvárovej biometrie by bolo možné priradiť identitu danému človeku. V spojení s týmto projektom by bolo možné odhaliť potencionálneho útočníka, ktorý odmietne interakciu s takýmto biometrickým zariadením (musí sa pozrieť do kamery).

Ďalšia možnosť spočíva v dobudovaní serverovej infraštruktúry a vytvorení informačného systému pre správu štatistík o návštevnosti nejakého miesta. Cieľovým zákazníkom takéhoto systému sú všetky obchodné reťazce, ktorých zaujíma návštevnosť svojich predajní v závislosti na čase či mieste.

Kapitola 6

Záver

Cieľom tejto práce bolo vytvoriť aplikáciu, ktorá bude schopná počítať prechody osôb s čo najväčšou presnosťou a zachová si priaznivú cenu. V rámci práce boli vytvorené dve aplikácie. Jedna pre snímanie priestoru virtuálnej brány za pomoci 2D technológie a druhá, ktorá používa pre snímanie aktívne hĺbkomery (príloha E).

V prípade aplikácie využívajúcej 2D snímanie sa podarilo vytvoriť nasaditeľnú aplikáciu, ktorej úspešnosť detekcie sa pohybuje medzi **90 - 94%**. Cenu takéhoto zariadenia sa podarilo stlačiť na **70 EUR** vďaka použitiu malého a lacného počítača Rraspberry Pi a štandardnej širokouhlej webkamery. Nízka cena zariadenia dovoľuje použiť takýto produkt pre rôzne maloobchodné prevádzky, ktoré potrebujú informácie o počte návštev zo štatistických dôvodov a nižšia spoľahlivosť zariadenia im neprekáža. Takéto zariadenia majú potenciál nahradiť nepresné IR závory, ktoré sú stále veľmi často používané kvôli cenovej nedostupnosti presnejších riešení.

V prípade druhej aplikácie, ktorá detekuje prechod osôb za pomocí hĺbkového snímku priestoru brány, bolo na začiatok nutné vybrať najvhodnejší hĺbkomer. V rámci práce boli k dispozícii žiaľ len tri a všetky používali technológiu aktívnej triangulácie (sekcia 2.3.5). Napriek tomu sa podarilo vytvoriť systém pre porovnanie kvality hĺbkového snímku na základe priemerného počtu chybne nameraných pixlov. Práca ukázala, že Kinect 360 má väčšiu odolnosť voči interferencii denného svetla avšak nie je možné určiť, ktorý hĺbkový snímač je pre aplikáciu najlepší. Je to z dôvodu rozdielnych zorných polí kamier. Preto na výber najvhodnejšej kamery je nutné poznať presné miesto nasadenia systému. Nameraná úspešnosť aplikácie vytvorenej v rámci tejto práce bola **96 - 98%**. Cena sa pohybuje okolo **150 - 220 EUR** v závislosti na type použitého hĺbkomera, keďže predstavuje najdrahšiu položku celého produktu. Vďaka vysokej presnosti je tento typ aplikácie vhodný na príklad v aplikáciach objektovej bezpečnosti v spojení s ďalším systémom overujúcim identitu prechádzaného objektu alebo v aplikáciach, kde je potrebné zaznámenávať väčšie množstvo informácií, ako napríklad výšku človeka.

Literatúra

[1] *Image Thresholding*. [Online; navštíveno 25.04.2017].
URL http://docs.opencv.org/trunk/d7/d4d/tutorial_py_thresholding.html

[2] Balihar, D.: *WHAT IS A PINHOLE CAMERA?* [Online; navštíveno 25.04.2017].
URL http://www.pinhole.cz/en/pinholecameras/whatis.html

[3] Esme, B.: *Kalman Filter For Dummies*. [Online; navštíveno 13.05.2017].
URL http://bilgin.esme.org/BitsAndBytes/KalmanFilterforDummies

[4] Faragher, R.: *Understanding the Basis of the Kalman Filter Via a Simple and Intuitive Derivation*. Září 2012, [Online; navštíveno 25.04.2017].
URL https://www.cl.cam.ac.uk/~rmf25/papers/Understanding%20the%20Basis%20of%20the%20Kalman%20Filter.pdf

[5] Gray Bradski, A. K.: *Learning OpenCV*. O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, 2008, ISBN 978-0-596-51613-0.

[6] Hurtík, P.: *DETEKCE HRAN V OBRAZU*. [Online; navštíveno 25.04.2017].
URL http://konference.osu.cz/svk/sbornik2012/pdf/budoucnost/informatika/hurtik.pdf

[7] Ing. Kalová Ilona, I. K.: *Optické metody měření 3D objektů*. [Online; navštíveno 25.04.2017].
URL http://www.elektrorevue.cz/clanky/05023/index.html

[8] Ing. Michal Španěl, I. V. B.: *Obrazové segmentační techniky*. [Online; navštíveno 25.04.2017].
URL http://www.fit.vutbr.cz/~spanel/segmentace/

[9] doc. Ing. Zoltán Tomori, M. M. N., CSc.: *Počítačové videnie v praxi*. Ústav experimentálnej fyziky SAV, Watsonova 47, 04001 Košice, 2016, [Online; navštíveno 25.04.2017].
URL http://home.saske.sk/~tomori/Downloads/Poc_videnie/PV_2016.pdf

[10] Jakub, O.: *Houghova transformácia a jej použitie v oblasti spracovania obrazu*. [Online; navštíveno 25.04.2017].
URL http://www.posterus.sk/?p=18844

[11] MacCormick, J.: *How does the Kinect work?* [Online; navštíveno 25.04.2017].
URL https://users.dickinson.edu/~jmac/selected-talks/kinect.pdf

[12] Milan Sonka PhD, R. B. D., Vaclav Hlavac PhD: *Image Processing, Analysis and Machine Vision*. Timothy L. Anderson, 2008, ISBN 978-1-133-59360-7.

[13] RNDr. Šikudová, P. I. B. C. R. H. R. K., PhD. RNDr. Černeková: *Počítačové videnie. Detekcia a rozpoznávanie objektov*. AWikina, Livornská 445, 109 00 Praha 10, 2011, ISBN 978-80-87925-06-5.
URL http://sccg.sk/~cernekova/Pocitacove_videnie.pdf

[14] Szeliski, R.: *Computer Vision Algorithms and Applications*. 2010, ISBN 978-1-84882-934-3, [Online; navštívené 25.04.2017].
URL http://szeliski.org/Book/

Prílohy

Príloha A

Program pre čítanie kamery Orbbec Astra S

```
import cv2
import numpy as np
from primesense import openni2
from primesense import _openni2 as c_api
def initCapture():
    openni2.initialize("./lib/openni2") #Path to libOpenNI2.dylib
    dev = openni2.Device.open_any()
    depth_stream = dev.create_depth_stream()
    depth_stream.start()
    depth_stream.set_video_mode(c_api.OniVideoMode(pixelFormat =
            c_api.OniPixelFormat.ONI_PIXEL_FORMAT_DEPTH_100_UM,
        resolutionX = 640, resolutionY = 480, fps = 30))
    return depth_stream

def cap_read(depth_stream):
    raw_frame = depth_stream.read_frame()
    frame_data = raw_frame.get_buffer_as_uint16()
    img = np.frombuffer(frame_data, dtype=np.uint16)
    img.shape = (1, 480, 640)
    img = np.concatenate((img, img, img), axis=0)
    img = np.swapaxes(img, 0, 2)
    img = np.swapaxes(img, 0, 1)
    img8 = (img/256).astype('uint8')
    img8 = (255-img8)
    return img8
depth_stream = initCapture()
while True:
    frame = cap_read(depth_stream)
    grayFrame = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    cv2.imshow("image", frame)
    cv2.waitKey(1)
openni2.unload()
```

Príloha B

Program pre nahrávanie kamery Intel RealSence SR30

```
import time
import numpy as np
import cv2
import pyrealsense as pyrs
import sys

def cap_read(dev):
    dev.wait_for_frame()
    d = dev.depth * dev.depth_scale * 255
    d = d.astype('uint8')
    d = (255-d)
    return d

record = False
if "-r" in sys.argv:
    record = True
    fourcc = cv2.VideoWriter_fourcc(*'XVID')
    out = cv2.VideoWriter('intelRealSenceSR300.avi',
          fourcc, 30.0, (640,480))

pyrs.start()
dev = pyrs.Device()

while True:
    frame = cap_read(dev)
    frame = cv2.cvtColor(frame, cv2.COLOR_GRAY2BGR)
    if record:
        out.write(frame)

    cv2.imshow('frame', frame)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
```

Príloha C

Program pre nahrávanie kamery Kinect 360

```
import numpy as np
import cv2
import freenect

def getDepthMap():
    depth, timestamp = freenect.sync_get_depth()
    np.clip(depth, 0, 2**10 - 1, depth)
    depth >>= 2
    depth = depth.astype(np.uint8)
    return depth

fourcc = cv2.VideoWriter_fourcc(*'XVID')
record_cap = cv2.VideoWriter('output.avi',fourcc, 20.0, (640,480))
while(True):
        depth = getDepthMap()
        frame = cv2.cvtColor(depth,cv2.COLOR_GRAY2BGR)
        record_cap.write(frame)
        cv2.imshow('frame',frame)
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break
```

Príloha D

Program pre výpočet metriky hĺbkovej kamery

```
import numpy as np
import cv2
import sys

cap = cv2.VideoCapture("./4/orbbecAstraS.mov")
limint = 1000
count = 0
sumPixels = 0
while(True):
    # Capture frame-by-frame
    ret, frame = cap.read()

    if count == limint:
        print "Resolution: ", len(frame), "x", len(frame[0])
        print "AVG pixels for frame: ", sumPixels / count
        sys.exit(0)

    frame = frame[:,40:]
    sumPixels += np.count_nonzero(frame == 255)
    count += 1
    cv2.imshow('fgrayrame',frame)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

# When everything done, release the capture
cap.release()
cv2.destroyAllWindows()
```

Príloha E

Obsah DVD

DVD nosič obsahuje:

- Zdrojové kódy aplikácie založenej na snímaní 2D
- Zdrojové kódy aplikácie založenej na snímaní 3D
- Vzorka testovacích dát
- Kompletný obraz pamäťovej karty Rraspberry Pi
- Demo aplikáce hranového detektora, globálneho prahovania, metód lokálneho prahovania využitých v práci