ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
Fakulta elektrotechnická
Katedra radioelektroniky

Navigační pomůcka pro nevidomé

Navigation aid for visually impaired people

Diplomová práce

Studijní program: Komunikace, multimedia a elektronika
Studijní obor: Multimediální technika

Vedoucí práce: Ing. Vítek Stanislav Ph.D.

Bc. Václav Punda

Praha 2014

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

Poděkování

Děkuji především vedoucímu mé diplomové práce Ing. Stanislavu Vítkovi Ph.D. za ochotu a pomoc při vypracování.

Abstrakt

V uvedené práci je představena aplikace pro mobilní telefon na platformě Android, která slouží jako navigační pomůcka pro zrakově postižené. Cílem bylo sestavit algoritmus, který bude schopen detekovat překážky pomocí obrazového senzoru a převádět tuto informaci na signalizaci, která bude srozumitelná zrakově postiženému. Algoritmus využívá k detekci optického toku a v reálném čase převádí informaci na zvukový signál pomocí zvuků tónové volby. Aplikace byla otestována pomocí natočených videí na počítači a následně jako samostatná aplikace v mobilním telefonu. Výsledky ukazují, že zvolená metoda má nedostatky a je potřeba dalšího vývoje a testování aby byla aplikace použitelná v reálném prostředí. V závěru je navrženo jakým způsobem by se toho dalo dosáhnout.

Abstract

Aplication for mobile phones on Android platform is presented. Application serves as an aid for visually impaired. The aim was to develop algorithm that is able to detect obstacles. Application converts information from image sensor to audio signal. Optical flow is used as main method for detection obstacles. Dual-tone multi-frequency was used for giving information to visually impaired. Application was tested on computer with recorded videos, subsequently on mobile phone. Results shows that there is a lot of issues that must be solved with this method in real use. Conclusion describes some methods that can be use in future development.

Obsah

Seznam obrázků

Kapitola 1

Úvod

Tato práce má za cíl vytvořit a otestovat aplikaci pro mobilní telefony na platformě *Android*, která bude sloužit jako navigace pro nevidomé. Aplikace by měla reagovat v reálném čase na základě informace z obrazového snímače. Cest, jak dosáhnout tohoto cíle, je mnoho, proto bude jako základ sloužit rešerše, která pomůže zvolit metodu vhodnou pro mobilní telefon. V textu budu využívat volné překlady článků, které se zabývají touto problematikou. Po uvedení do problematiky pomůcek pro nevidomé popíšu vlastní vývoj aplikace a testování. V průběhu práce jsem se mírně odchýlil od zadání a to tím, že zvolenou metodu nebudu testovat v programovém prostředí Matlab, nýbrž v jazyce *C++* pomocí knihovny *OpenCV*. Toto je z důvodu snadnější implementace a rychlejšímu výpočtu.

Mobilní telefony dnes poskytují velmi zajímavé spojení výpočetního výkonu a kvalitních obrazových snímačů. V podstatě se dá říci, že původní účel telefonu je dnes upozaděný. Vývoj, jakým se mobilní telefony za poslední roky ubíraly je dán především cílem integrovat co nejvíce funkcí do jediného přístroje a zvyšovat výkon, kvůli multimediálním aplikacím a hrám. Tento vývoj, který je dán primárně trhem, ovšem přináší příležitosti využít tyto "přenosné počítače" i k jiným účelům, jako například k asistivní technologii. Podle odhadu Světové zdravotnické organizace (WHO) je na světě okolo 285 miliónů zrakově postižených, z toho 39 miliónů je úplně slepých a zbytek je slabozraký[1]. Tato organizace dělí zrakově postižené na:

- normální (nepostižené),
- mírně postižené,
- silně postižené,
- nevidomé.

Za posledních dvacet let se počet zrakově postižených snížil, nicméně potřeba pomůcek pro tuto skupinu je stálá. Zejména při pohybu v městském prostředí je stále požadavek na jakékoliv pomůcky, které by zrakově postiženým umožnily lepší orientaci a předcházení nebezpečí.
Nejčastější pomůcky, které zrakově postižení používají jsou bílá hůl a vodící pes. Jsou oblíbené zejména pro svojí jednoduchost a rozšířenost. Tyto historicky známé pomůcky ovšem nejsou schopny poskytnout úplnou informaci (hůl) o tom, co má nevidomý před sebou, nebo jsou náročné finančně a časově (cena vycvičeného psa je v současné době 250 000,- Kč)[2].
Moderní technologie nabízejí velmi široké možnosti tvorby pomůcek pro nevidomé. Využívá se mnoho fyzikálních a matematických principů pro stanovení vzdálenosti od překážky, navigace, orientace v prostoru atd.

1.1 Pohyb v prostředí

Velkou část našeho života strávíme přesunem z jednoho místa na druhé. Pro zrakově postižené přináší cestování řadu komplikací. Při denních úkonech se dostávají do situací, které vyžadují orientaci v prostoru a nalezení cíle. V dnešní době jsou města stále jen málo uzpůsobena pro pohyb zrakově postižených, ti proto musí využívat pomůcek, které jim ulehčují, a mnohdy i umožňují se dostat tam, kam potřebují. Problémy při cestování se dají rozdělit do dvou skupin [9]:

- mobilita - vyhýbání se překážkám, orientace a navigace,
- pohyb v prostředí - minimalizace rizika, informace a značení.

Pohyb zrakově postiženého je možné shrnout do dvou typů procesů. Je to zaprvé získání vjemu, tedy odhad toho, co je v okolí a vyhnutí se překážce. Za druhé je to orientace a navigace v prostoru.

Vjem a vyhnutí se překážky Jedná se o zaznamenání potenciálně rizikových objektů v prostředí před zrakově postiženým a umožnění se jim vyhnout. Tento proces je hlavním účelem pomůcek pro nevidomé.

Vjem a identifikace značení Jedná se o zaznamenání značení a předání informace. Nevidomý může využívat značení pro vidomé a orientovat se tak v prostoru, získávat informace o směru pohybu a vyhnout se nebezpečí.

Orientace v prostoru Tento proces zahrnuje bezprostřední okolí zrakově postiženého. Řeší orientaci na cestě a možnost sledování směru bez nechtěného zatáčení.

Zeměpisná orientace - Navigace Narozdíl od **Orientace v prostoru** se jedná o orientaci v širším měřítku a nalezení strategie cesty do vzdáleného cíle. Zahrnuje plánování trasy z bodu A do bodu B.

1.2 Asistenčních technologie pro pohyb v prostředí

Ve vývoji asistenčních technologií došlo k velkému posunu. Dlouhou dobu šel vývoj se zaměřením na technologický aspekt a uživatelská přívětivost se zanedbávala. Mnoho pomůcek nevyšlo ze stádia prototypu a jen málo se jich dostalo do užívání nevidomými. Proto stále zůstává nejoblíbenější pomůcka slepecká hůl a vodící pes, jakožto historicky ověřené a dostupné pomůcky. Elektronika má ale v tomto velký potenciál. Možné důvody malého rozšíření asistenčních pomůcek jsou:

- velká složitost zařízení a obtížné naučení se pomůcku používat,
- vysoká cena vývoje,
- míjení se potřeb nevidomých a možností pomůcky,
- ve výsledku horší funkcionalita než slepecká hůl.

Pro vývoj je vhodné definovat typické otázky k řešení při pohybu zrakově postiženého:

Vyhýbání se překážkám Lze rozdělit na prostor v úrovni nohou, hrudníku a hlavy. Dále je to potřeba volného prostoru po stranách. Například zeď, lešení, dveře nebo zužující se cesta. Také to jsou změny v úrovni cesty, například schody, obrubníky atd.

Orientace a navigace Tato otázka definuje: bezpečné sledování cesty, bezpečné použití přechodu, nalezení cesty a pohyb po této cestě.

Minimalizace rizika Bezpečné použití přechodu - vyhnutí se srážky s automobilem. Dále upozornění na konec chodníku, či přechod. Městský mobiliář umístěný tak aby nebyl překážkou pro chodce.

Informace a značení Přístupné značení, informace o veřejné dopravě a další veřejné značení

1.3 Rozdělení pomůcek

Většina pomůcek pracuje s jednoduchým principem a to:

- získání informace z okolí (informace o překážce, směru, trase),
- předání informace uživateli ve vhodné formě.

Předání informace uživateli je jedna z klíčových vlastností pomůcky pro zrakově postižené. Je potřeba se při vývoji zaměřit na to aby pomůcka neomezovala smysly, které jsou pro nevidomé důležité. Pomůcky můžeme dělit podle:

1. primární a sekundární pomůcka - zda-li je pomůcka použita jako hlavní zdroj informací nebo je jen doplňující. Sekundární pomůcky zajišťují doplňující informace o okolí, například o překážce,
2. dělení podle funkcionality - zařízení umožnující vyhnutí se překážce, pomůcky pro orientaci a navigaci dávající informaci o značení a umožňující najít nejlepší cestu, zařízení pro hledání objektů,
3. podle technologie kterou získáváme informaci z okolí - ultrazvuk, infračervené spektrum, obrazová informace z kamery, globální polohovací systéme (GPS), zjišťování pozice pomocí mobilního telefonu,
4. založené na způsobu předání informace uživateli - hmatový vjem, řeč, zvukové pulzy, hudební tóny,
5. podle způsobu používání - hůl, pomůcka do jedné ruky, pomůcka do kapsy, pomůcka do batohu, zavěšení na krk.

Kapitola 2

Teorie

2.1 Příklady pomůcek

2.1.1 Echolokace

Projekt zabývající se echolokací vznikl v devadesátých letech v Japonsku [12] a jeho cílem je vytvořit pomůcku pro nevidomé napodobující echolokaci netopýrů. Dva ultrazvukové senzory jsou připevněny na tradiční brýle a data z těchto senzorů jsou převedena pomocí mikroprocesoru na slyšitelný stereo signál, který je následně přehráván ve sluchátkách. Různá intenzita a časové rozdíly mezi signály indikují velikost a polohu překážky. Byly provedeny předběžné pokusy s vyhodnocením schopnosti uživatele rozlišit objekty pomocí různých ultrazvukových frekvencích. Výsledky ukazují, že je to možné pouze v omezené míře a více dalších testů a statistik je potřeba k vyhodnocení životaschopnosti projektu. Výhodou tohoto řešení je jednoduchost. Nevýhodou jistě je, nutnost uživatele se učit používat tuto pomůcku.

2.1.2 Navbelt

Navbelt vyvinul pan Borenstein a jeho spolupracovníci na universitě v Michiganu. Systém využívá ultrazvukové senzory, počítač a sluchátka. Počítač přijímá informaci z osmi ultrazuvkových senzorů a vytváří mapu úhlů (pro každý senzor jednu) a vzdáleností objektu v tomto úhlu. Poté algoritmus pro vyhnutí se překážce vytvoří zvuk odpovídající různým módům.
Navbelt má dva módy: průvodcovský mód a obrazový mód. Během průvodcovského módu počítač zná uživatelův cíl a s jednoduše se opakujícím pípáním provádí uživatele

optimální cestou. V praxi a reálné aplikaci by bylo pro tento mód potřeba více senzorů. V obrazovém módu je osm tónů s rozdílnou amplitudou zahráno v rychlém sledu z osmi různých směru (podobně jako posun radaru). Počítač, s ohledem na mód, převede tyto zvuky na formu, kterou lze přehrát na sluchátkách. Nevýhody tohoto systému jsou použití sluchátek a dlouhá doba potřebná pro zaučení nevidomého.

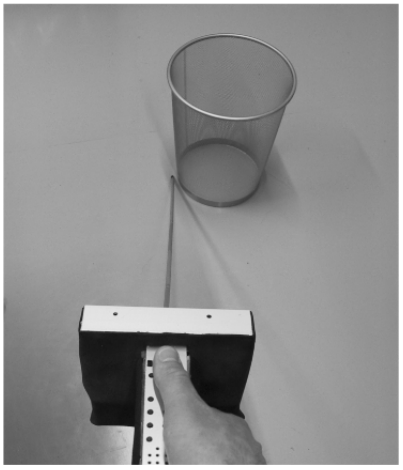
2.1.3 vOICe

Obrázek 2.1: Implementace systému vOICe - "vidět zvuk", brýle s integrovanou kamerou, sluchátka, počítač [18].

Tato pomůcky využívá předpokladu, že člověk dokáže rozeznat složité a rychle se měnící zvuky. Prototyp na obrázku č. 2.1 se skládá z brýlí s digitální kamerou, sluchátky a počítače. Získané obrázky z kamery převádí počítač na zvuk. Obrázky se převádí jedna ku jedné, jeden obrázek na jeden zvuk. Zvuk je pak zasílán do sluchátek. Při přehrávání nejsou použity žádné filtry, předpokládá se, že lidský mozek je natolik výkonný, že dokáže zpracovat tak komplexní informaci. K stávajícímu jednoduchému systému bylo přidáno zařízení na bázi sonaru pro větší bezpečnost. Systém vyzkoušelo mnoho jedinců s poměrně dobrou zpětnou vazbou, nicméně k lepšímu využití vyžaduje složitější učení [18].

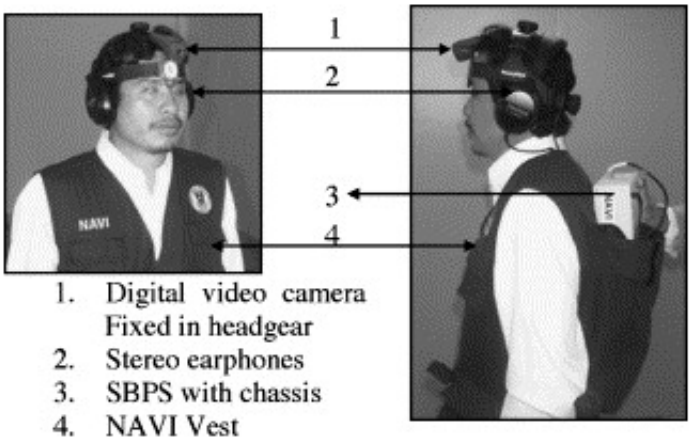
2.1.4 Projekt university ve Stuttgartu

Přenosný systém pro orientaci ve vnitřních prostorech (obr. 2.2) byl vyvinut na univerzitě ve Stuttgartu. Prototyp se skládá z modulu, na kterém jsou umístěny senzory, na tento modul je umístěna hůl podobná slepecké. Dále je k zařízení připojen přenosný počítač. Modul je vybaven dvěma kamerami, klávesnicí podobnou té na mobilním telefonu, digitálním kompasem a reproduktorem. Počítač obsahuje systém pro detekci vzdálenosti objektů na základě barev a přístup k bezdrátové síti. Zařízení funguje víceméně v reálném čase. Pro vylepšení zařízení byl vytvořen 3D model prostředí, takže informace ze senzoru mohou být sladěny s virtuálním 3D prostorem. Výhody tohoto zařízení jsou v robustnosti senzorů, uživatelské přívětivosti a v detekci v téměř reálném čase.



Obrázek 2.2: Zařízení university ve Stuttgartu je podobné klasické slepecké holi [11].

2.1.5 NAVI - Navigation assistance for Visually Impaired



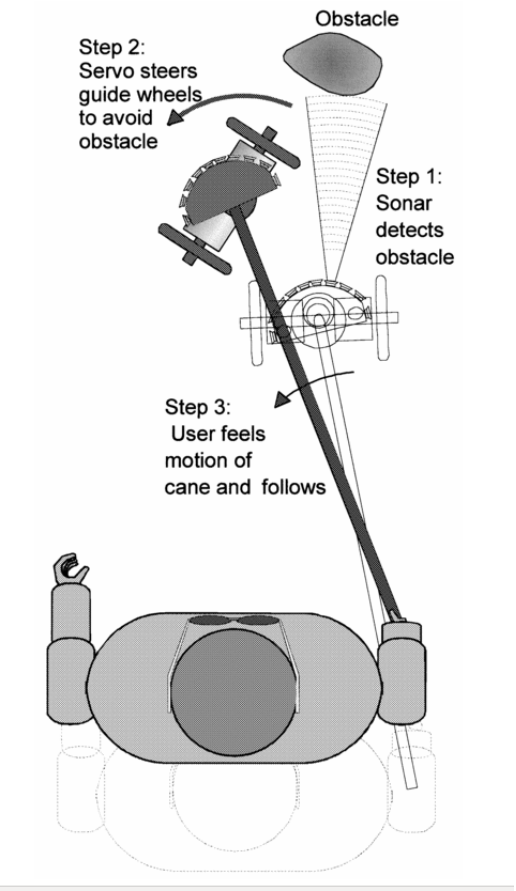
Obrázek 2.3: Jednotlivé části systému NAVI [19].

Tato pomůcka navržená výzkumníky z university v Malajsii (University Malaysia Sabah) [19] pomáhá zrakově postiženým lidem zaznamenat překážku pokud se nachází přímo před nimi. Prototyp navigační pomůcky na obr. 2.3 se skládá z digitální kamery upevněné na helmě, stereo sluchátek, SBPS - single board processing system, baterií a vesty, na které je připevněn SBPS a baterie.

Kamera snímá prostředí před zrakově postiženým a obraz je zpracován pomocí "fuzzy" shlukové analýzy v reálném čase. Zpracovaný obraz je převeden do speciálních strukturovaných stereo akustických vzorců a přehráván ve sluchátkách. Tento systém vyžaduje trénink, zrakově postižení se navíc musí naučit jednotlivé vzory, aby rozpoznali překážku. Testované osoby byly schopny po tréninku rozpoznat i pomalu pohybující se objekty.

Výhodou tohoto systému jsou fungování v reálném čase. Nevýhodou je pouze informace o přítomnosti překážky a ne o vzdálenosti.

2.1.6 GuideCane

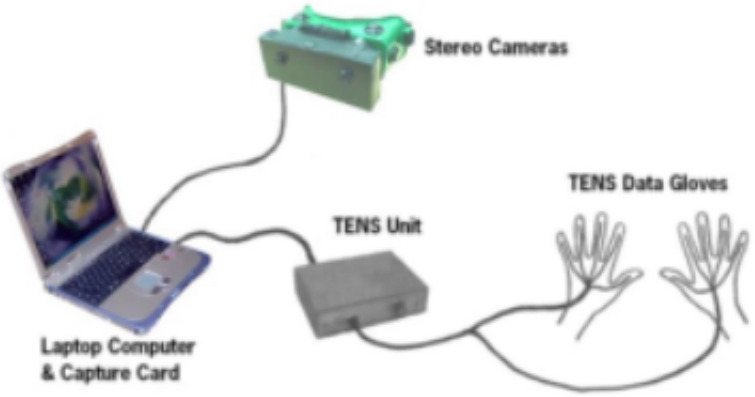


Obrázek 2.4: Princip použití systému [24].

GuideCane [24] je zařízení, které bylo vyvinuto s využitím stávající pomůcky a to slepecké hole. Je to projekt od stejného týmu jako Navtbelt viz 2.1.2. Na obr. 2.4 je prototyp, který se jak vidno skládá ze slepecké hole, na jejímž konci je umístěno hlavní zařízení. Toto zařízení má kolečka, také jsou na něm umístěny ultrazvukové senzory a počítač. Součástí je také mechanismus, který otáčí celým zařízením. Mechanismus vyhnutí se překážce je jednoduchý: uživatel před sebou tlačí zařízení pomocí hole. Jakmile je detekována překážka pomocí ultrazvukových senzorů, algoritmus v počítači vyhodnotí, jakým směrem by se měl uživatel překážce vyhnout. Mechanismus pak se zařízením zatočí v požadovaném směru. Uživatel má na konci hole také joystick, kterým může směr pohybu ovládat. Senzor dokáže detekovat malé překážky na cestě a překážky jako jsou zdi atd.

V porovnání s ostatními pomůckami vyčnívá tato pomůcka svou jednoduchostí a také tím, že neomezuje uživatelovo sluchové ústrojí, tudíž ho může využívat tak jak je zvyklý. Dále není nutný nijak zvláštní trénink pro používání této pomůcky.

2.1.7 ENVS - Electron-Neural Vision System

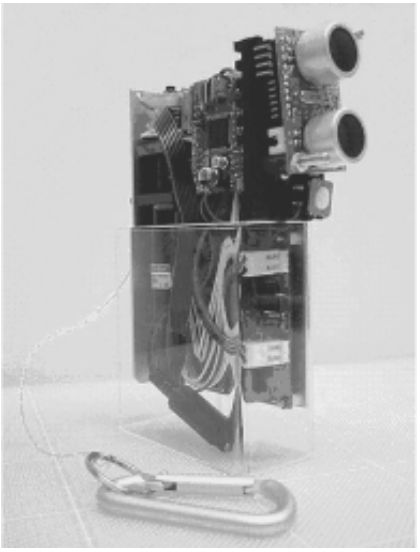


Obrázek 2.5: Jednotlivé části systému [17].

Na universitě v Wollongongu [17] v Austrálii vyvinuli systém pro navigaci a vyhýbání se překážkám ve venkovním prostředí. Pomůcka využívá obrazového senzoru, GPS navigace, taktilní simulace. Na obr. 2.5 je prototyp, obsahuje dvě kamery, digitální kompas a přenosný počítač s GPS, jednotku pro elektrickou nervovou stimulaci a rukavice přenášející taktilní vjemy. Základní koncept je ve využití stereoskopického snímání prostředí a výpočtu hloubkové mapy. Pomocí detekce překážky a GPS souřadnic systém dává informaci uživateli v podobě elektrických pulzů v rukavicích. Intenzita pulzů je úměrná vzdálenosti od překážky a směru, v jakém se překážka nachází.
Prototyp byl testován uživateli, kteří měli zakryté oči. V reálném čase byli schopni s minimem tréninku určit pozici překážky, vyhnout se jí a dorazit do definovaného místa. Tento systém je velmi komplexní, jelikož využívá jak detekci překážek, tak navigaci uživatele do určitého místa. Dále neomezuje sluchové ústrojí, důležitý to orgán pro zrakově postižené. Nevýhodou je omezení hmatu v rukavicích. Jelikož kamery jsou umístěny na hlavě, je riziko, že překážky blízko u země nejsou detekovány.

2.1.8 CyARM

CyARM [13] je pomůcka, usnadňující pohyb a orientaci, která využívá poměrně nestandardní rozhraní: ultrazvukový sensor detekuje překážku a vypočítá vzdálenost od uživatele. Uživatel je informován o vzdálenosti pomocí drátu, který má připevněný na



Obrázek 2.6: Prototyp zařízení CyARM s viditelným zapojením a sensory [13].

pásek. Pokud je překážka blízko, drát se více napne, zatímco pokud je ve směru senzorů volný průchod, drát se uvolní.
Prototyp je zařízení o váze asi půl kila, které se drží v ruce. Obsahuje mikrokontrolér, který zpracovává informaci ze senzorů a ovládá motorek, který napíná drát. Úspěšnost zařízení při detekci překážek byla otestována na poměrně malém počtu opakování. Nicméně při 90 % testů byli uživatelé schopni nalézt větší překážku a vyhnout se jí nebo projít mezi dvěma překážkami. Při pohybujících se cílech bohužel výsledky neodpovídaly použitelnosti. Hlavní výhodou tohoto systému je jednoduchá obsluha, kterou se uživatel nemusí nijak výrazně učit. Nevýhodou tohoto zařízení, je nutnost pomůcku neustále držet a prohlížet s ní okolí. Bohužel experimentálních výsledků se zrakově postiženými je málo.

2.2 Pomůcky na bázi získávání informace z obrazu

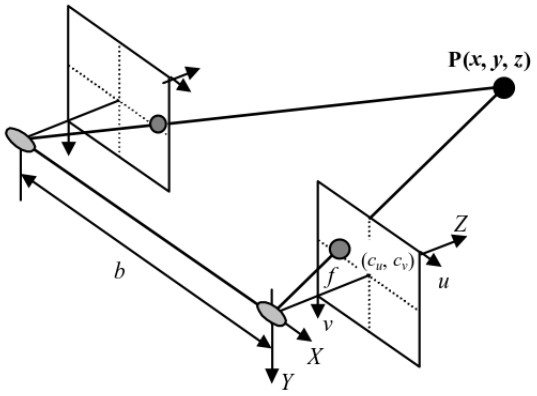
Metod detekce překážek je celá řada. Vývoj zařízení je hnán především automobilovým průmyslem, pro který je tento výzkum již léta zajímavý. Jelikož metod je nepřeberné množství omezíme se pouze na obrazovou informaci. Vzhledem k cíli práce, tedy implementaci detekce překážek do mobilního telefonu, je potřeba zvolit metodu, která bude brát ohled na parametry přístrojů. V dnešní době je výpočetní výkon v mobilních telefonech na takové úrovni, že umožňuje počítat náročné operace zpracování obrazové informace. Také obrazové snímače v mobilních telefonech mají dobré parametry. V zásadě se používají tři metody získání informace o překážce z obrazové informace [28].

- hledání vzorů v obraze
- metody založené na stereoskopii
- detekce pohybu v obraze

Metoda hledání vzorů v obraze je většinou určena pro specifické aplikace. Jako detekce chodců, automobilů atd. Tento přístup využívá specifických parametrů objektů v obraze, jako jsou obrys, barva, textura, vertikální hrany, symetrie atd. Pro výpočet se využívá neuronových sítí a algoritmů, které umožňují naučit se vzory v obraze. Tato metoda je využitelná pouze pro specifické typy objektů.
Metody založené na stereoskopii jsou velmi rozšířené. V zásadě se jedná o dva typy získání informace o překážce. Jedna na bázi výpočtu *rozdílové mapy* [21] a druhá na bázi *inverzní perspektivy* [22] více kapitola 2.2.1.
Metoda využívající pohybu v obraze je založena na myšlence, že pohyb v obraze je vykonáván především pohybujícím se objektem. Využito je rozdílových snímků a optického toku [15] viz kapitola 2.2.2.
Vyvstává nám otázka, zda využít pro detekci překážky dvou či jedné kamery.
S prosazováním 3D technologií se na trhu objevily mobilní přístroje, které mají dva obrazové snímače pro reprodukci stereoskopického obrazu. Proto tato možnost není irelevantní a stojí za to se jí zabývat.

2.2.1 Pomocí dvou obrazových snímačů

Rozdílové mapy



Obrázek 2.7: Princip výpočtu rozdílových map [21].

Princip detekce obrazu pomocí dvou snímačů vychází z fyziologie lidského zraku. Rozdíl v obraze z levého a pravého oka nám umožňuje získat informaci o prostoru.

Na obr. 2.7 příklad modelu pro stereoskopické vidění. Souřadnice (X, Y, Z) mají střed v levém obraze. Pro daný bod $P(X, Y, Z)$mají obrazy v pravém a levém snímači souřadnice $p_l(x_l, y_l)$ a $p_r(x_r, y_r)$ a jeho souřadnice mohou být vypočteny jako:

$$X = \frac{b \cdot x_l}{d}, Y = \frac{b \cdot y_l}{d}, Z = \frac{b \cdot f}{d} \tag{2.1}$$

kde b je délka mezi snímači, f je ohnisková vzdálenost čočky a d je rozdíl mezi souřadnicemi vypočtený jako $x_l - x_r$. Na základě těchto vztahů může být vytvořen 3D model.
Pro získání informace o překážce lze využít různých metod [21].

Inverzní perspektiva

Úhel pohledu (např. kamery) a vzdálenost objektu (překážky) komplikuje stanovení polohy objektu (prespektiva). Při zpracování obrazu musí být perspektiva brána v potaz a to zvyšuje výpočetní nároky na aplikace. Proto se zavedla geometrická transformace - *Inverzní perspektiva* (Inverse Perspective Mapping - IPM). Tato transformace umožňuje odstranit efekt perspektivy a převést obraz do nové dvoudimenzionální domény. Informační obsah v této doméně je rovnoměrně rozdělen mezi všechny pixely. IPM vyžaduje k výpočtu další informace jako pozice kamery, orientace, optika atd., také je zde potřeba jistý předpoklad o scéně. IPM může být tedy využito v aplikacích, kde je kamera připevněna v pevné pozici, nebo kde ke kalibraci systému můžeme využít další senzory, které nám dají potřebné informace.
IPM představuje vlastně trasformaci z třídimenzionálního Euklidovského reálného prostoru W na dvoudimenzionální Euklidovský prostor $I.$, kde:

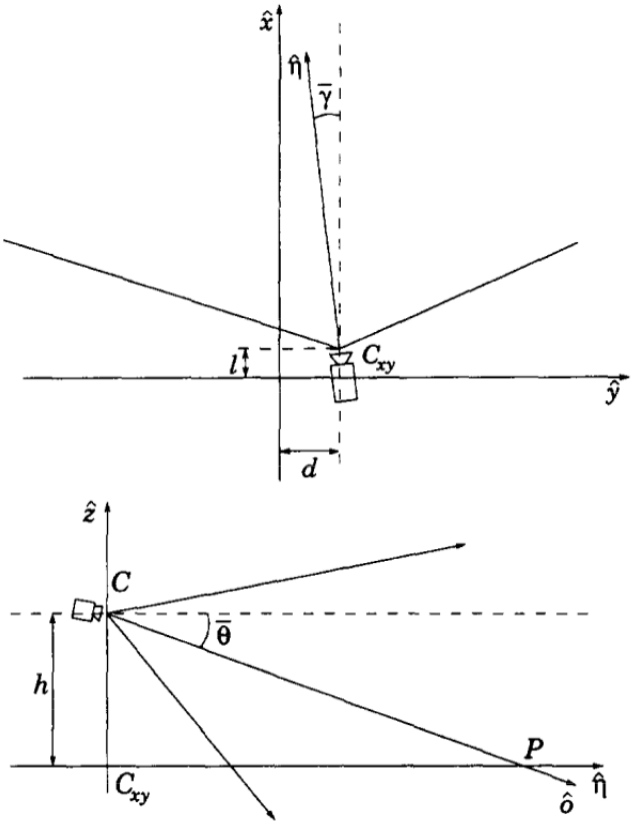
- $W = (x, y, z) \in E^3$ představuje třídimenzionální prostor,
- $I = (u, v) \in E^2$ představuje dvoudimenzionální obrázek kde je zobrazen prostor W. Prostor I odpovídá získanému obrázku. Za předpokladu rovného povrchu je převedený obrázek definován jako rovina v prostoru W nazvaná jako $S \triangleq (x, y, 0) \in W$.

Použití IPM vyžaduje tyto parametry:

- *poloha* - pozice kamery je $C = (l, d, h) \in W$,
- *směr pohledu* - je dáno úhlem ô,
- *úhel pohledu* $\bar{\gamma}$ daný vektorem $\tilde{\eta}$ v rovině $z = 0$ viz obr. 2.9,

Obrázek 2.8: Příklad přepočteného obrazu pomocí inverzní perspektivy [23]

- $\bar{\theta}$ - úhel daný optickou osou ô a vektorem $\tilde{\eta}$,
- apertura je 2α,
- rozlišení je $n \times n$.



Obrázek 2.9: Rovina xy v prostoru W představuje plochu S. Rovina $z\eta$ za předpokladu, že počátek je přesunut do projekce C_{xy} bodu C v S [3].

Výsledné souřadnice jsou:

$$\begin{cases} x(u,v) = h \times \cot\left[(\bar{\theta}-\alpha) + u\frac{2\alpha}{n-1}\right] \times \cos\left[(\bar{\gamma}-\alpha) + v\frac{2\alpha}{n-1}\right] + l \\ y(u,v) = h \times \cot\left[(\bar{\theta}-\alpha) + u\frac{2\alpha}{n-1}\right] \times \sin\left[(\bar{\gamma}-\alpha) + v\frac{2\alpha}{n-1}\right] + d \\ z(u,v) = 0, \end{cases} \tag{2.2}$$

rovnice 2.2 vrací bod $(x, y, z) \in S$ odpovídající bodu $(u, v) \in I$.

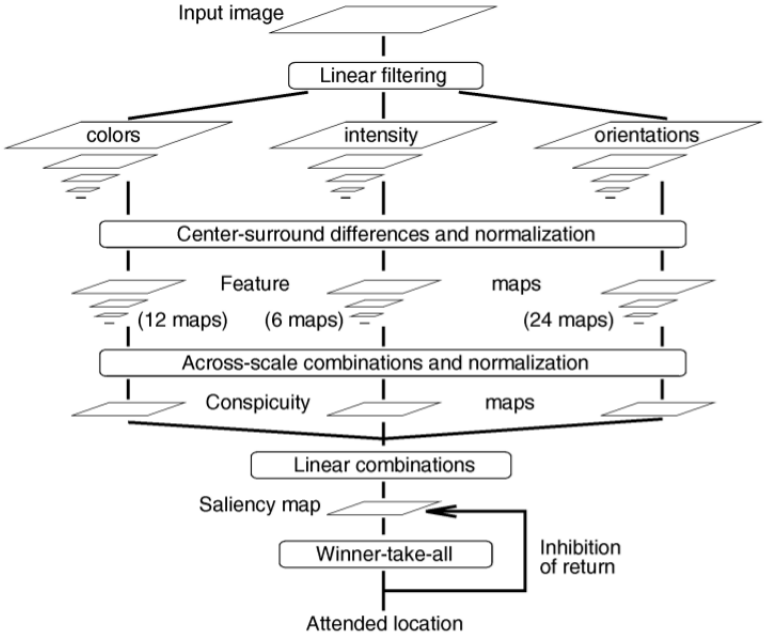
Pro detekci překážek lze tohoto nástroje využít např. pro nalezení pohledu z levé kamery v obraze v pravé kameře [23]. Z levého obrázku se promítne informace do třídimenzionálního prostoru a následně je zpětně promítnuta do pravého obrázku, kde je porovnána s pravým obrázkem. Tímto způsobem dokážeme zjistit obrysy překážky nad zemním povrchem. Lze také počítat rozdílové mapy (viz. Kapitola 2.2.1) přímo z obou přepočtených obrázků. Tato metoda není příliš vhodná pro aplikaci s mobilním telefonem, jelikož by se jen těžko zajišťovala přesná poloha zařízení a další parametry, které jsou potřeba ke zjištění inverzní perspektivy.

2.2.2 Pomocí jednoho obrazového snímače

Možností pro získání informace o překážce pomocí jednoho obrazového snímače není mnoho. Přesto lze některé s úspěchem použít. Metoda využívající pouze jedné kamery je také vhodnější pro dostupné zařízení jako je mobilní telefon. Na trhu je pouze pár modelů se dvěma kamerami, které by se daly využít pro zjištění polohy překážky pomocí metod založených na stereoskopii.

Model salience

Pro detekci překážek lze využít model salience založený na modelování vlastnosti zraku. Model je založený na detekci míst v obraze, na které jako první padne naše pozornost, pokud pozorujeme danou scénu. Myšlenka je založená na předpokladu, že lidský zrak je zaujat nejprve jasnými objekty, které se objeví v obraze. Tyto objekty jsou pak detekovány jako překážka.
Model je založen na architektuře odpovídající biologii vizuálního systému. Vstupní obraz je nejprve rozdělen na sadu topografických map. Každá mapa reaguje na jiné informace v obraze, které jsou odpovědné za upoutání naší pozornosti. Mezi jednotlivými mapami, kde je zachyceno prostorové uspořádání, je pak vybrána ta, jejíž hodnoty nejvíce vystupují. Při dalším kroku, hledání dalšího objektu, mezi sebou opět soutěží jednotlivé charakteristiky a je vybrána ta s nejvyššími hodnotami.



Obrázek 2.10: Model salience - vstupní obrázek je filtrován na pyramidu a dále rozdělen na barevné kanály, intensitu a mapu orientace [25].

Architektura modelu je následující viz obr. 2.10. Model vybírá určité oblasti pozornosti na základě tří charakteristik. Jas signálu, orientace a barva. Vstupní obrázek je podvzorkován do dyadické Gaussovy pyramidy pomocí konvoluce s lineární Gaussovým filtrem a následně decimován faktorem dva. Opakováním tohoto filtrování vznikne pyramida obrazů se snižující se kvalitou obrazové informace.
Pokud máme tři kanály r, g, b, jas obrázku se získá jako:

$$M_I = \frac{r + g + b}{3} \tag{2.3}$$

Výpočet je opakován pro každou úroveň vstupní pyramidy. Tím vznikne charakteristika jasu.
Dále jsou vypočítány barevné kanály červená-zelená (RG) a modrá-žlutá (BY):

$$M_{RG} = \frac{r - g}{max(r, g, b)}, M_{BY} = \frac{b - min(r, g)}{max(r, g, b)}. \tag{2.4}$$

Mapy barevných kanálů jsou dále prahovány.
Další charakteristikou je mapa orientace. Je vytvořena pomocí konvoluce map intensity s Gaborovým filtrem natočeným v různých úhlech. $O(\sigma, \theta)$ kde σ je počet pyramid a $\theta \in 0°, 45°, 90°, 135°$ jsou preferované orientace Gaborových filtrů (Gaborovy filtry aproximují impulsovou odezvu selektivně orientovaných neuronů v primární zrakové kůře).

Z těchto tří charakteristik se vypočítá celková mapa tzv. mapa salience. Tato mapa reprezentuje salienci pomocí skalární veličiny pro každé místo v obraze. Umožňuje výběr zúčastněných míst pomocí prostorového rozložení salience. Kombinace tří výše zmíněných charakteristik přináší vstup pro výpočet mapy salience, která je modelována jako dynamická neuronová síť.
Jeden problém při kombinování tří rozdílných charakteristik je, že reprezentují neporovnatelné modality s rozdílným způsobem získávání a rozsahem hodnot. Jelikož je kombinováno 42 charakteristik, jen několik málo map bude maskováno šumem nebo méně výrazným objektem. Aby bylo možné kombinovat tyto charakteristiky, zavádí se normalizační operátor. Aplikováním těchto normalizačních operátorů (více v [14]) vytvoříme tři mapy viditelnosti. Motivace vytvoření těchto nezávislých map je na základě předpokladu, že podobné mapy soutěží o výsledné místo pozornosti společně zatímco rozdílné nezávisle. Mapy přepočtené pomocí normalizačního operátoru jsou sečteny do výsledné mapy salience.

$$S = \frac{1}{3}\left(N(\overline{I}) + N(\overline{C}) + N(\overline{O})\right) \tag{2.5}$$

kde N je normalizační operátor.
Z této mapy lze jednoduše vybrat maximum, které definuje oblast na kterou nejdříve padne naše pozornost. V modelu se nicméně dále počítá s využitím neuronové sítě, která simuluje zrakového ústrojí a vybírá charakteristiky, která určí místo pozornosti.

Vhodné vzory v obraze pro trasování

Je mnoho bodů v obraze, které by bylo možno sledovat pomocí trasování. Je však poměrně náročné rozhodnout, který bod je zrovna vhodný. Je jasné, že pokud budeme mít bílou zeď a na ní jeden bod bude poměrně jednoduché nalézt tento bod i v dalším snímku videa. Pokud však budou v obraze body, které jsou stejné či velmi podobné bude mnohem náročnější nalézt požadovaný bod v následujícím snímku. Pokud se nám ovšem podaří nalézt v obraze bod, který bude nějakým způsobem unikátní, máme velkou šanci ho nalézt znovu. V praxi je něco takového možné a nalezený bod může mít takové parametry, že existuje možnost je porovnat s body v dalším obrázku.
Body které mají velkou derivaci mohou představovat vhodnou volbu. Je zřejmé, že tato podmínka nebude dostačující nicméně pomůže nám začít. Bod který má velkou změnu oproti jiným bodům, může být na hraně nějakého předmětu, ale může vypadat stejně jako všechny body na této hraně. Avšak pokud je derivace vypočtena ve dvou navzájem kolmých směrech, můžeme uvažovat, že tento bod je mnohem víc specifický než ostatní.

Z tohoto důvodu je mnoho vhodných bodů v obraze pro sledování nazýváno *rohy*. Intuitivně můžeme říci, že rohy - ne hrany - jsou body, které obsahují dostatek informace pro jejich nalezení v dalším obrázku.

Nejvíce používaná definice rohů v obraze je podle Harrise [8]. Tato definice závisí na matici druhých derivací $(\partial^2 x, \partial^2 y, \partial^2 x, \partial x \partial y)$ vypočítaných z jasu obrázku. Můžeme uvažovat derivace druhého řádu vypočítané pro všechny obrazové body, které nám dají obrázek druhých derivací nebo tzv. *Hessián* obrázek. Tato terminologie vznikla z Hessiánu, matice orientované okolo bodu, který je definován dvoudimenzionálně jako:

$$H(p) = \begin{bmatrix} \frac{\partial^2 I}{\partial x^2} & \frac{\partial^2 I}{\partial x \partial y} \\ \frac{\partial^2 I}{\partial y \partial x} & \frac{\partial^2 I}{\partial y^2} \end{bmatrix}_p . \tag{2.6}$$

Pro nalezení vhodných rohů uvažujeme *autokorelační matici* druhých derivací v malém okolí každého bodu. Matice může vypadat takto:

$$M(x,y) = \begin{bmatrix} \sum_{-K \le i,j \le K} w_{i,j} I_x^2(x+i, y+i) & \sum_{-K \le i,j \le K} w_{i,j} I_x(x+i, y+i) I_y(x+i, y+i) \\ \sum_{-K \le i,j \le K} w_{i,j} I_y(x+i, y+i) I_x(x+i, y+i) & \sum_{-K \le i,j \le K} w_{i,j} I_y^2(x+i, y+i) \end{bmatrix} \tag{2.7}$$

.

Kde $w_{i,j}$ je váhovací parametr, zpravidla kruhové okénko nebo Gaussovské váhování. Definice rohů podle Harrise, jsou místa v obraze kde autokorelační matice druhých derivací má dvě velká vlastní čísla. V podstatě to znamená, že je v tomto místě vzor (nebo hrana), který má alespoň dva rozdílné směry okolo středu bodu, stejně jako se roh potkává uprostřed obrazu ze dvou směrů. Tato definice má další výhody, že pokud uvažujeme pouze vlastní čísla autokorelační matice, uvažujeme hodnoty, které jsou invariantní k rotaci, což je důležité pokud se předměty, které se snažíme trasovat, otáčí. Dvě vlastní čísla jsou tedy také výhodné, ne jenom pro rozpoznání, zda-li je bod vhodný pro trasování, ale přináší nám také vodítko pro identifikaci bodu.

Harrisova původní definice vyžadovala výpočet determinantu $H(p)$, odečtení od matice (s váhovacími koeficienty) a poté porovnání tohoto rozdílu s předpokládaným prahem. Později Shi a Tomasi [20] přišli na to, že dobré body pro trasování jsou ty, které mají menší ze dvou vlastních čísel větší než minimální práh. Jejich metoda není pouze účinnější, ale v mnoha případech dává i lepší výsledky.

Optický tok

Další z metod využitelných pro detekci překážek je výpočet optického toku z obrazové informace. Výhoda jeho využití je opět v použití jedné kamery, dále v relativně snadné formě výpočtu a velkých možností optimalizace. Optický tok nám dává informaci o pohybu kamery vůči okolnímu světu nebo naopak objektů v okolním světě vůči kameře. Výpočtem získáme informaci o směru pohybu v obraze v závislosti na čase.
Existuje několik metod výpočtu optického toku. Tyto metody jsou rozděleny do tří hlavních částí:

- korelační,
- diferenční,
- blokové.

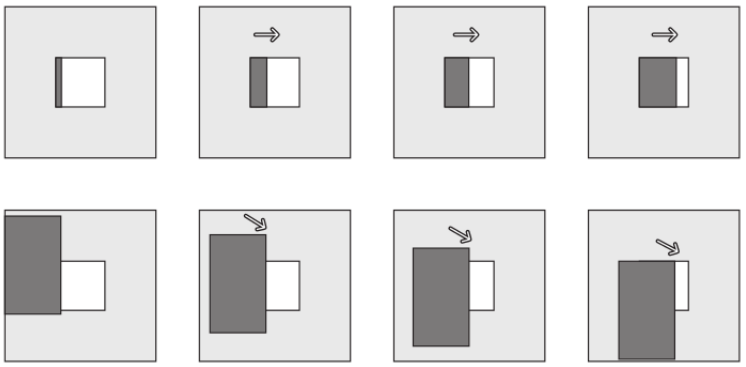
Metody založené na diferenci jsou nejčastěji používány a lépe se vypořádávají s texturou pozadí [5]. Hlavní metody a historicky jedny z prvních, které se pro výpočet optického toku používají jsou:

- Lucas-Kanade [16],
- Horn-Schunck [10].

Lucas-Kanade metoda výpočtu optického toku

Tato metoda vznikla v osmdesátých letech na universitě v Pittsburghu Bruce D. Lucasem a Takeo Kanadem [16] a je hojně využívaná v oblasti počítačového vidění. Tato diferenční metoda předpokládá, že optický tok v okolí pixelu, který bereme v úvahu je konstantní a řeší základní rovnice optického toku pro všechny pixely v okolí pomocí výpočtu nejmenších čtverců.
Nyní popíšeme základ výpočtu optického toku pomocí metody Lucas-Kanade. Uvažujme I a J jako dva šedotónové obrázky. Veličiny $I(\mathbf{x}) = I(x, y)$ a $J(\mathbf{x}) = J(x, y)$ jsou poté šedotónové intenzity ze dvou obrázků a můžeme psát $\mathbf{x} = [x\ y]^T$, kde x a y souřadnice pixelu obecného bodu v obraze $\mathbf{x}$. Obrázek I můžeme označit jako první a J jako druhý. Z praktických důvodů je obrázek uvažovat obrázky jako diskrétní funkce a levý horní roh bude mít souřadnice $[0\ 0]^T$. Nechť n_x a n_y jsou výška a šířka obou obrázků, poté pravý dolní roh je na souřadnici $[n_x - 1\ n_y - 1]^T$
Uvažujme v prvním obrázku bod $\mathbf{u} = [u_x\ u_y]^T$. Cíl odhadu optického toku a sledování bodů je nalézt pozici bodu $\mathbf{v} = \mathbf{u} + \mathbf{d} = [u_x + d_x\ u_y + d_y]^T$ v druhém obrázku $J(\mathbf{u})$,

který je "stejný" jako $I(\mathbf{v})$. Vektor $\mathbf{d} = [d_x\ d_y]^T$ je rychlost v bodě $\mathbf{x}$, také známá jako optický tok v bodě $\mathbf{x}$. Kvůli problému vnímání pohybu okolo apertury (*aperture problem*) je nezbytné definovat pojem podobnosti ve smyslu sousedních bodů ve 2D. Tento problém vzniká pokud měříme pohyb v malé apertuře. Pokud je pohyb detekován v malé apertuře, vidíme pouze pohyb hrany nikoli rohu. Pomocí hrany nicméně nemůžeme v případě malé apertury detekovat, jakým způsobem se ve skutečnosti předmět pohybuje viz obr. 2.11. Nechť ω_x a ω_y jsou dvě celá čísla. Definujeme rychlost změny v obraze $\mathbf{d}$ jako vektor, který minimalizuje funkci ϵ popsanou takto:



Obrázek 2.11: Problém apertury - pokud vidíme pouze část objektu skrze malou aperturu, může, pokud je objekt větší než apertura, dojít ke zkreslení interpretace pohybu [6].

$$\epsilon(\mathbf{d}) = \epsilon(d_x, d_y) = \sum_{x=u_x-\omega_x}^{u_x+\omega_x} \sum_{y=u_y-\omega_y}^{u_y+\omega_y} (I(x,y) - J(x+d_x, y+d_y))^2, \tag{2.8}$$

tato funkce hledá minimum v obraze v okolí $(2\omega_x + 1) \times (2\omega_y + 1)$. Toto okolí je také nazýváno integrační okno. Typické hodnoty $\omega_x a \omega_y$ jsou 2, 3, 4, 5, 6, 7 pixelů.

Popis trasovacího algoritmu Při výpočtu optického toku touto metodou jsou klíčové dva parametry a to přesnost a robustnost. Přesnost závisí na lokální přesnosti zúčastněných pixelů při výpočtu. Přirozeně, malé integrační okno bude zajišťovat aby se při výpočtu nezanedbaly detaily obsažené v obraze (malé hodnoty ω_x a ω_y). Toto je především nutné při překrývajících se částech obrazu, kde se dvě části pohybují s velmi rozdílnými rychlostmi [27].

Robustnost souvisí s citlivostí trasování vzhledem k změnám osvětlení, velikosti obrázku, pohybu atd. Zejména pokud se jedná o rozsáhlý pohyb, intuitivně volíme velké integrační okno. Tedy, ve vztahu k rovnici 2.8, upřednostňujeme mít $d_x \leq \omega_x$ a $d_y \leq \omega_y$. Je zde tedy přirozený rozpor mezi těmito parametry, přesností v detailu a robustností při výběru integračního okna.

Nyní popíšeme vlastní výpočet optického toku. Pro každý snímek definujme nově obrázky A a B takto:

$$\forall(x,y) \in [p_x - \omega_x - 1, p_x + \omega_x + 1] \times [p_y - \omega_y - 1, p_y + \omega_y + 1],$$
$$A(x,y) \doteq I(x,y), \tag{2.9}$$
$$\forall(x,y) \in [p_x - \omega_x, p_x + \omega_x] \times [p_y - \omega_y, p_y + \omega_y],$$
$$B(x,y) \doteq J(x + g_x, y + g_y). \tag{2.10}$$

Jak je vidět definice obrázku A a B je rozdílná. Souřadnice bodu $A(x,y)$ jsou definovány přes velikost okna $(2\omega_x+3)\times(2\omega_y+3)$, namísto $(2\omega_x+1)\times(2\omega_y+1)$. Budeme tedy hledat vektor posunutí $\mathbf{d}$, který minimalizuje následující funkci podobnou té již definované 2.8:

$$\epsilon(\mathbf{d}) = \epsilon(d_x, d_y) = \sum_{x=u_x-\omega_x}^{u_x+\omega_x} \sum_{y=u_y-\omega_y}^{u_y+\omega_y} (A(x,y) - B(x+d_x, y+d_y))^2. \tag{2.11}$$

Na tuto rovnici může být aplikován standardní iterativní Lucas-Kanade algoritmus. Ideálně je první derivace ϵ s ohledem na $\mathbf{d}$ rovna nule:

$$\left.\frac{\partial \epsilon(\mathbf{d})}{\partial \mathbf{d}}\right|_{\mathbf{d}=d_{opt}} = [0\ 0]. \tag{2.12}$$

Po derivaci celé funkce dostáváme:

$$\frac{\partial \epsilon(\mathbf{d})}{\partial \mathbf{d}} = -2 \sum_{x=u_x-\omega_x}^{u_x+\omega_x} \sum_{y=u_y-\omega_y}^{u_y+\omega_y} (A(x,y) - B(x+d_x, y+d_y)) \cdot \begin{bmatrix} \frac{\partial B}{\partial x} \frac{\partial B}{\partial y} \end{bmatrix}. \tag{2.13}$$

Pokud provedeme substituci za $B(x+d_x, y+d_y)$ pomocí Taylorovy řady prvního stupně v bodě $\mathbf{d} = [0\ 0]^T$:

$$\frac{\partial \epsilon(\mathbf{d})}{\partial \mathbf{d}} \approx -2 \sum_{x=u_x-\omega_x}^{u_x+\omega_x} \sum_{y=u_y-\omega_y}^{u_y+\omega_y} (A(x,y) - B(x,y) - \begin{bmatrix} \frac{\partial B}{\partial x} & \frac{\partial B}{\partial y} \end{bmatrix} \mathbf{d}) \cdot \begin{bmatrix} \frac{\partial B}{\partial x} & \frac{\partial B}{\partial y} \end{bmatrix}. \tag{2.14}$$

Povšimněme si, že hodnota $A(x,y) - B(x,y)$ může být interpretována jako časová derivace obrazu v bodě $[x\ y]^T$:

$$\forall(x,y) \in [p_x - \omega_x, p_x + \omega_x] \times [p_y - \omega_y, p_y + \omega_y],$$
$$\delta I(x,y) \doteq A(x,y) - B(x,y). \tag{2.15}$$

Matice $\left[\frac{\partial B}{\partial x}\frac{\partial B}{\partial y}\right]$ je pouze gradient obrazových bodů. Nyní přepíšeme zápis takto:

$$\nabla I = \begin{vmatrix} I_x \\ I_y \end{vmatrix} \doteq \left[\frac{\partial B}{\partial x}\ \frac{\partial B}{\partial y}\right]^T. \tag{2.16}$$

Derivaci obrazu I_x a I_y lze vypočítat přímo z prvního obrázku $A(x,y)$ v okolí $(2\omega_x + 1) \times (2\omega_y + 1)$ bodu $\mathbf{u}$ nezávisle na druhém obrázku $B(x,y)$. Důležitost tohoto zjištění bude zřejmá při popisu iterativní verze optického toku.

$$\forall(x,y) \in [p_x - \omega_x, p_x + \omega_x] \times [p_y - \omega_y, p_y + \omega_y]$$
$$I_x(x,y) = \frac{\partial A(x,y)}{\partial x} = \frac{A(x+1,y).A(x-1,y)}{2}, \tag{2.17}$$
$$I_y(x,y) = \frac{\partial A(x,y)}{\partial x} = \frac{A(x,y+1).A(x,y-1)}{2}. \tag{2.18}$$

Rovnici 2.13 můžeme díky tomu přepsat:

$$\frac{1}{2}\frac{\partial \epsilon(\mathbf{d})}{\partial \mathbf{d}} \approx \sum_{x=u_x-\omega_x}^{u_x+\omega_x} \sum_{y=u_y-\omega_y}^{u_y+\omega_y} (\nabla I^T \mathbf{d} - \delta I)\nabla I^T, \tag{2.19}$$

$$\frac{1}{2}\left[\frac{\partial \epsilon(\mathbf{d})}{\partial \mathbf{d}}\right]^T \approx \sum_{x=u_x-\omega_x}^{u_x+\omega_x} \sum_{y=u_y-\omega_y}^{u_y+\omega_y} \left(\begin{bmatrix} I_x^2 & I_xI_y \\ I_xI_y & I_y^2 \end{bmatrix} \mathbf{d} - \begin{bmatrix} \delta I & I_x \\ \delta I & I_y \end{bmatrix}\right), \tag{2.20}$$

pokud označíme

$$G \doteq \sum_{x=u_x-\omega_x}^{u_x+\omega_x} \sum_{y=u_y-\omega_y}^{u_y+\omega_y} \begin{bmatrix} I_x^2 & I_xI_y \\ I_xI_y & I_y^2 \end{bmatrix}, \tag{2.21}$$

a dále

$$\overline{b} \doteq \sum_{x=u_x-\omega_x}^{u_x+\omega_x} \sum_{y=u_y-\omega_y}^{u_y+\omega_y} \begin{bmatrix} \delta I & I_x \\ \delta I & I_y \end{bmatrix}, \tag{2.22}$$

může být rovnice 2.20 přepsána jako:

$$\frac{1}{2}\left[\frac{\partial\epsilon(\mathbf{d})}{\partial\mathbf{d}}\right]^T \approx G\mathbf{d} - \overline{b}. \tag{2.23}$$

Tedy podle rovnice 2.12, je optimální vektor optického toku:

$$\mathbf{d} = G^{-1}\overline{b}. \tag{2.24}$$

Tento výraz platí pouze pokud je matice G regulární. To je ekvivalentní tvrzení, že obrázek $A(x, y)$ má gradient v obou směrech x a y v okolí bodu $\mathbf{u}$.
Toto je standardní výpočet optického toku pomocí metody Lucas-Kanade. Tato metoda má několik omezení:

- platí pouze pokud je pohyb mezi snímky malý (stupeň Taylorova rozvoje je pouze první),
- mezi dvěma snímky se zásadně nemění jas,
- prostorová koherence - většina pixelů leží ve stejné rovině v jakém se vykonává pohyb.

Pro dosažení požadované přesnosti je využito iteračního mechanismu. Index k označuje počet iterací a $k \geq 1$. Předpokládejme, že předchozí výpočty od 1,2,...,k-1 přináší počáteční odhad vektoru $\mathbf{b}^{k-1} = [\mathbf{b}_x^{k-1}\ \mathbf{b}_y^{k-1}]^T$ pro posun pixelu v obraze $\mathbf{d}$. Nechť B_k je nový posunutý obraz odpovídající počátečnímu odhadu $\mathbf{b}^{k-1}$:

$$\begin{aligned} \forall(x, y) \in [p_x - \omega_x, p_x + \omega_x] \times [p_y - \omega_y, p_y + \omega_y], \\ B(x, y) = B(x + \mathbf{d}_x^{k-1}, y + \mathbf{d}_y^{k-1}). \end{aligned} \tag{2.25}$$

Cíl je tedy opět nalezení vektoru pohybu pixelu $\overline{\eta}^{-k} = [\eta_x^{-k}\ \eta_y^{-k}]$, který minimalizuje chybovou funkci.

$$\epsilon(\overline{\eta}^k) = \epsilon(\eta_x^k, \eta_y^k) = \sum_{x=u_x-\omega_x}^{u_x+\omega_x} \sum_{y=u_y-\omega_y}^{u_y+\omega_y} (A(x, y) - B(x + \eta_x^k, y + \eta_y^k))^2. \tag{2.26}$$

Řešení této minimalizace může být výpočet jednoho kroku Lucas-Kanade optického toku z rovnice 2.24:

$$\overline{\eta}^k = G^{-1}\overline{b}_k, \tag{2.27}$$

kde vektor $\overline{b}_k$ o velikosti 2×1 je definován takto:

$$\overline{b}_k = \sum_{x=u_x-\omega_x}^{u_x+\omega_x} \sum_{y=u_y-\omega_y}^{u_y+\omega_y} \begin{bmatrix} \delta I_k(x,y) I_x(x,y) \\ \delta I_k(x,y) I_y(x,y) \end{bmatrix}, \tag{2.28}$$

kde $k^{tý}$ rozdíl obrázků δI_k je definován takto:

$$\begin{aligned} \forall (x,y) \in [p_x - \omega_x, p_x + \omega_x] \times [p_y - \omega_y, p_y + \omega_y], \\ \delta I_k(x,y) = A(x,y) - B_k(x,y). \end{aligned} \tag{2.29}$$

Všimněme si, že prostorové derivace I_x a I_y (ve všech bodech okolí $\mathbf{u}$) jsou vypočteny pouze na začátku rekurze pomocí rovnic 2.17 a 2.18. Proto matice G 2×2 zůstává konstantní pro všechna opakování. Tím se sníží počet výpočtů. Jediná hodnota, která se musí opakovaně počítat pro každý krok k, je vektor $\overline{b}_k$, který zachycuje rozdíl v obraze po posunu o vektor $\mathbf{d}_k$. Pokud je vypočten vektor $\overline{\eta}^k$ podle rovnice 2.27, nový posun pixelu vypočteme pro každý krok k následovně:

$$\overline{\mathbf{d}}^k = \overline{\mathbf{d}}^{k-1} + \overline{\eta}_k. \tag{2.30}$$

pomocí rekurze počítáme vektor $\overline{\eta}_k$ dokud je menší než práh (např. 0.03 pixelu) nebo dokud není dosaženo maxima opakování. V průměru je potřeba pět opakování k dosažení konvergence. Pro první opakování (k=1) je počáteční odhad nastaven do nuly:

$$\overline{\mathbf{d}} = [0\ 0]^T \tag{2.31}$$

Po všech opakováních a dosažení konvergence je výsledný vektor $\mathbf{d}$ roven optickému toku:

$$\mathbf{d} = \sum_{k=1}^{K} \overline{\eta}^k \tag{2.32}$$

Tento vektor minimalizuje funkci popsanou v rovnici 2.8.

Horn-Schunk metoda výpočtu optického toku

Tato metoda je hojně využívaná pro svoji jednoduchost a účinnost. Jako jedna z prvních začala využívat k výpočtu optického toku rozdílu mezi dvěma snímky.

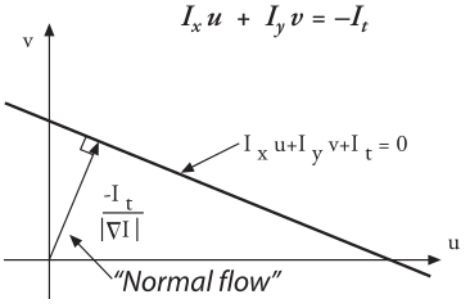
Předpoklad stálého jasu Sekvenci snímků lze matematicky popsat jako funkci $I(x, y, t)$, kde I je jas obrazu v čase t o prostorové souřadnici *(x,y)*. Úplná derivace změny obraze v čase je dána:

$$\frac{DI}{Dt} = \frac{\partial I}{\partial x}\frac{dx}{dt} + \frac{\partial I}{\partial y}\frac{dy}{dt} + \frac{\partial I}{dt}, \tag{2.33}$$

kde $\frac{\partial I}{\partial x}$, $\frac{\partial I}{\partial y}$ a $\frac{\partial I}{dt}$ může být vypočteno přímo z dvou po sobě následujících snímků $I(x, y, t)$ a $I(x, y, t\partial t)$. $\frac{dx}{dt}$ a $\frac{dy}{dt}$ jsou složky rychlosti u a v. Horn a Schunck svoji metodu založili na předpokladu, že intenzita se v čase mezi dvěma snímky příliš nemění, tudíž levou stranu rovnice 2.33 $\frac{DI}{Dt}$ položíme nule. V souvislosti s tímto omezením můžeme rovnici psát jako:

$$\frac{DI}{Dt} = I_x u + I_y v + I_t = 0, \tag{2.34}$$

kde $I_x = \frac{\partial I}{\partial x}$, $I_y = \frac{\partial I}{\partial y}$ a $I_t = \frac{\partial I}{\partial t}$. Je to tedy jedna rovnice pro dvě neznámé. Pokud rovnici opět přepíšeme získáme rovnici přímky, která je znázorněna na obr. 2.12.



Obrázek 2.12: Na obrázku je znázorněna rovnice přímky. "Normal flow" značí normálový vektor viz rovnice 2.36 [6].

$$u = -\frac{I_y}{I_x}v - \frac{I_t}{I_x}. \tag{2.35}$$

Na obrázku 2.12 je znázorněn normálový vektor d, který se vypočte ze souřadnic u, v jako:

$$d = -\frac{I_t}{\sqrt{I_x^2 + I_y^2}}. \tag{2.36}$$

Dále vypočteme pro u a v pro každý pixel obrázku minimalizací funkce:

$$E^2 = \int_\Omega [\alpha^2 E_c^2 + E_b^2] \mathrm{d}\Omega, \tag{2.37}$$

kde funkce znázorňují omezení. E_c je již zmiňované omezení konstantního jasu. E_b je vyhlazení obrazových bodů pomocí umocnění na druhou:

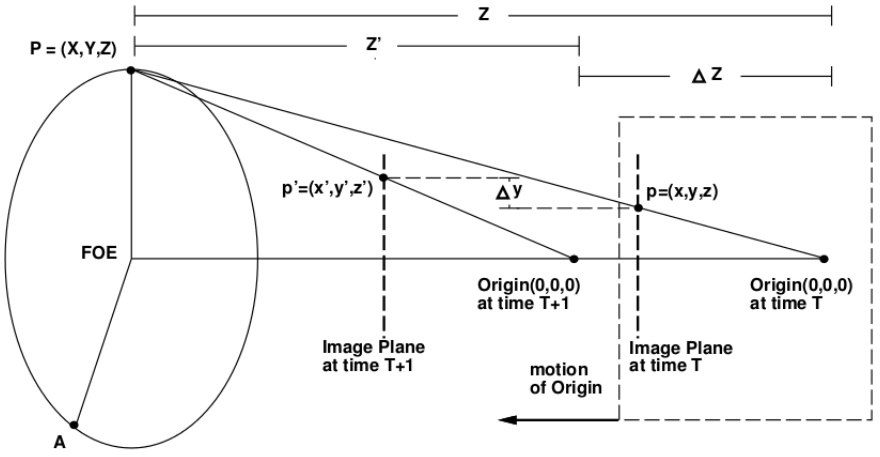
$$E_c^2 = (I_x u + I_y v + I_t)^2 \tag{2.38}$$

$$E_b^2 = (u_x)^2 + (u_y)^2 + (v_x)^2 + (u_y)^2 = \|\nabla \mathbf{v}\|_2^2 \tag{2.39}$$

Okamžik kontaktu s objektem

Optického toku můžeme využít k výpočtu tzv. okamžiku kontaktu nebo občas pesimističtěji nazývaného okamžiku kolize či srážky. Pomocí optického měření a bez známé rychlosti pohybu či vzdálenosti od překážky (obecně povrchu) je možné touto metodou vypočítat kdy dojde ke kontaktu.
Obrázek 2.13 popisuje geometrii a princip výpočtu okamžiku kontaktu. Bod v obraze P o souřadnicích (X,Y,Z) má projekci v bodě (0,0,0). Předpokládejme, že P je umístěn fyzicky v prostoru a nehýbe se. Počátek projekce se pohybuje spolu s obrazovou rovinou rychlostí $\frac{dZ}{dt}$. Jestliže je obrazová rovina kolmá na směr pohybu, pak je tento směr obecně znám jako ohnisko rozšíření (FOE - Focus of Expansion). Tedy bod, ze kterého optický tok diverguje. Obrazová rovina je umístěna ve vzdálenosti z před počátkem, zvolíme $z = 1$ (správná hodnota z záleží na různých faktorech, jako například ohnisková vzdálenost kamery). Tato obrazová rovina se pohybuje s počátkem. P se zobrazí do bodu p. Jak se tato rovina přibližuje k bodu P, tak pozice p se v obrazové rovině mění. S využitím podobnosti trojúhelníků můžeme psát:

$$\frac{y}{z} = \frac{y}{1} = \frac{Y}{Z}, \tag{2.40}$$



Obrázek 2.13: Princip výpočtu TTC. Postup získání FOE je uveden v kapitole 2.2.2 [7].

pokud derivujeme podle času [kde $\dot{a}$ představuje $\frac{dx}{dt}$]:

$$\dot{y} = \frac{\dot{Y}}{Z} - Y\left(\frac{\dot{Z}}{Z^2}\right), \tag{2.41}$$

Pokud P zůstává na místě, můžeme zvolit $\dot{Y} = 0$ a provést substituci za (yZ) za Y:

$$\dot{y} = -y\left(\frac{\dot{Z}}{Z}\right), \tag{2.42}$$

nakonec podělíme celý výraz y:

$$\frac{y}{\dot{y}} = \frac{-Z}{\dot{Z}} = \tau, \tag{2.43}$$

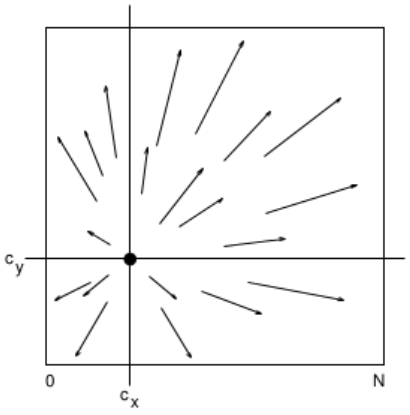
Hodnota τ se uvádí jako okamžik kontaktu. Levá strana rovnice se stává pouze z veličin vypočtených z obrazové informace, proto nám tato hodnota nedává žádnou informaci o rychlosti nebo vzdálenosti za čas, ale pouze o měřítku. Levá strana nám dává způsob, jak vypočítat okamžik kontaktu: pro kameru pohybující se ve stejném směru jako FOE, vezměme bod v obraze a vydělme vzdálenost od FOE časovou derivací vzdálenosti od FOE. Nicméně tento způsob se ukazuje jako nedostatečný [7], lepší výpočet je předveden v článku [4].

Výpočet ohniska rozšíření - Focus Of Expansion - FOE

Algoritmus výpočtu vychází z předpokladu, že derivace optického toku je v bodě FOE nulová a všechny vektory optického toku směřují do tohoto bodu.
Nechť $\mathbf{V} = (v_x(x,y), v_y(x,y)), 0 \leq x < N, 0 \leq y < N$ je vektor rychlosti, získaný ze sekvence dvou po sobě následujících snímcích o velikosti N. Toto vektorové pole obsahuje pouze radiální složky v okolí FOE $C = (c_x, c_y) = (\lambda_{foe}, \mu_{foe})$, tedy:

$$\mathbf{V}(x,y) = \begin{pmatrix} x \perp c_x \\ y \perp c_y \end{pmatrix}. \tag{2.44}$$

Výsledný obrázek může být rozdělen do čtyř sekcí okolo FOE podle souřadnic, jak je vidět na obrázku 2.14.



Obrázek 2.14: Bod o souřadnicích (c_x, c_y) je v divergenci pole vektorů rychlosti změny [4].

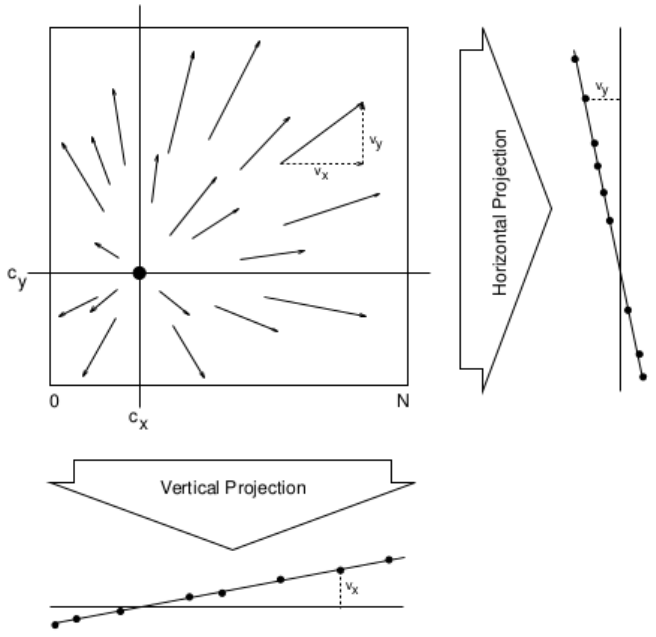
Nechť $P_{hy}(y)$ je normalizovaná horizontální projekce (h v indexu značí horizontální) y komponent V, podobně $P_{hx}(y)$, $P_{vx}(x)$ a $P_{vx}(x)$ další tři projekce. Tyto projekce se vypočítají jako:

$$P_{hy}(y) = \frac{1}{N} \sum_{x'=0}^{N-1} v_y(x', y) = \frac{1}{N} \cdot N(\perp c_y) = y \perp c_y \; y, \tag{2.45}$$

$$P_{hx}(y) = \frac{1}{N} \sum_{x'=0}^{N-1} v_x(x', y) = \frac{1}{N} \left(\frac{N(N \perp 1)}{2} \perp N \cdot c_x \right) = \frac{N \perp 1}{2} \perp c_x = const_x, \tag{2.46}$$

$$P_{vy}(x) = \frac{1}{N}\sum_{y'=0}^{N-1} v_y(x,y') = \frac{1}{N}\left(\frac{N(N \perp 1)}{2} \perp N \cdot c_y\right) = \frac{N \perp 1}{2} \perp c_y = const_y, \quad (2.47)$$

$$P_{vx}(x) = \frac{1}{N}\sum_{y'=0}^{N-1} v_x(x,y') = \frac{1}{N} \cdot N(\perp c_x) = x \perp c_x \; x, \quad (2.48)$$



Obrázek 2.15: Horizontální a vertikální projekce [4].

Rovnice 2.45 a 2.48 popisují přímky s hodnotami c_y, respektive c_x, kde počátek má souřadnice $y_0 = c_y$ resp. $x_0 = c_x$. Výrazy 2.46 a 2.47 jsou konstanty, které závisí pouze na velikosti obrázku N a souřadnicích C. To nám dává tři možnosti odhadnout pozici C.

- z kořenů x_0 a y_0 rovnic 2.48 a 2.45: $P_{hy}(y_0) = 0 \Leftrightarrow c_y = y_0$ a $P_{vx}(x_0) = 0 \Leftrightarrow c_x = x_0$
- z hodnot výrazů 2.48 a 2.45 v nule: $c_y = \perp P_{hy}(0)$ a $c_x = \perp P_{vx}(0)$
- z konstant2.46 a 2.47: $c_y \frac{N-1}{2} \perp const_y$ a $c_x \frac{N-1}{2} \perp const_x$

Kapitola 3

Praktická část

V této části je popsána realizace, testování a v neposlední řadě i popis pracovního postupu při vytváření aplikace pro mobilní telefony. Při vyvíjení aplikace byla využita knihovna OpenCV. Tato knihovna se velmi hodí na nejrůznější aplikace při zpracování obrazu, analýzu pohybu v obraze, trasovací algoritmy atd. Pro tuto práci se hodí i proto, že je multiplatformní. Je to knihovna v jazyce *C++*, nicméně velkou část je možno používat i v jazyce *Java* a funkce se stále rozšiřují, většinu je možné také používat v *Python*. Toto umožňuje použití v systému *Android* pro mobilní telefony, pro který byla tato navigace navrhována, jelikož tento systém je napsán v jazyce *Java*. Další výhodou této knihovny je, díky její multiplatformnosti, možnost testovat aplikaci přímo na desktopu počítače, pokud je napsána v *C++*. To umožňuje rychlejší testování a lehčí ověřování dat než při vývoji přímo na mobilním telefonu.

3.1 Pracovní postup při vytváření aplikace

Při vývoji aplikace byl zvolen postup napsání velké části kódu v jazce *C++*, díky výše zmíněným výhodám a také díky jistějšímu vývoji v OpenCV. Systém *Andriod* umožňuje volat funkce obsahující nativní kód. Je k tomu využit tzv. Native Development Kit - NDK, tedy sada nástrojů, která umožňuje implementovat části aplikace využívající nativních kódu jako *C* a *C++*. Níže bude uveden postup, jak tento nástroj implementovat a používat.

Vytvoření aplikace - Eclipse - OpenCV - Android

Nejprve je potřeba vytvořit aplikaci pro *Android*, využijeme zde postup na stránkách OpenCV. Pro práci je potřeba mít nainstalovaný tento software:

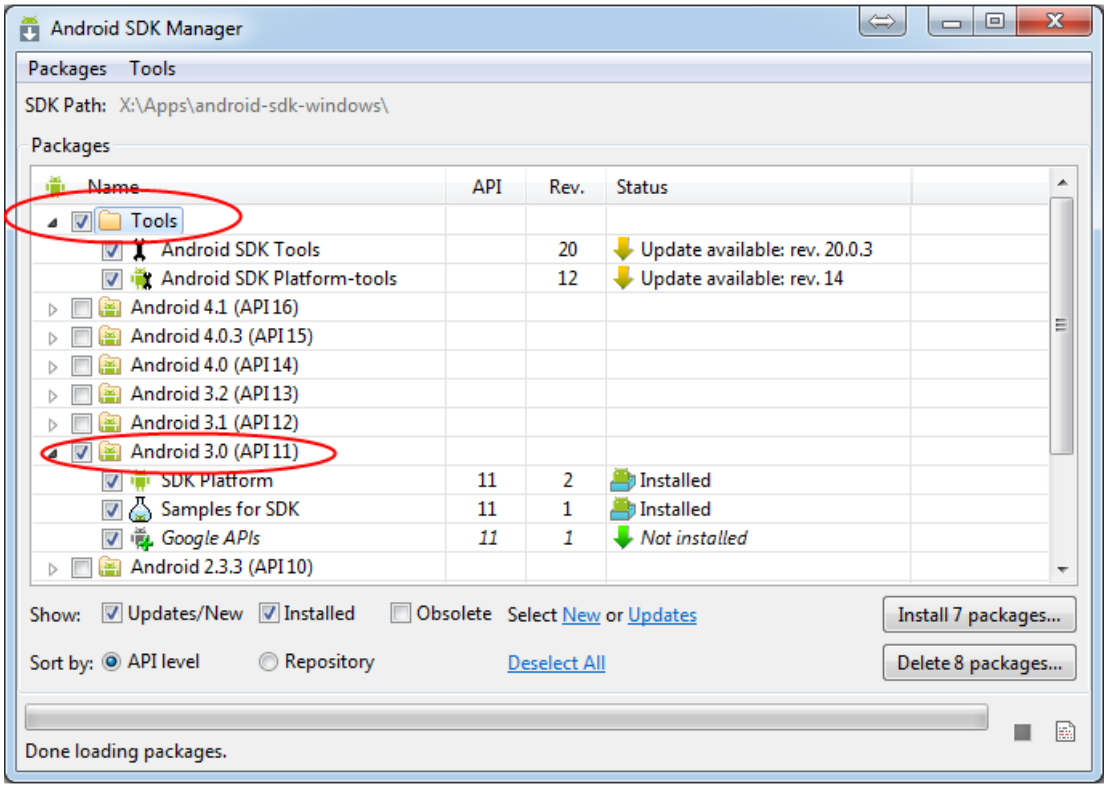
- Sun JDK 6 (možné i 7)
  Ke stažení na:
  http://www.oracle.com/technetwork/java/javase/downloads/index.html

- Android SDK
  Pro instalaci je potřeba nejnovější verze stažitelná na:
  http://developer.android.com/sdk/index.html.

- Android SDK - části nutné pro OpenCV

  – Android SDK Tools, verze 20 nebo vyšší.

  – SDK Platform Android 3.0 (API 11). Funguje i pro nižší API (minimum je 8) ale je doporučeno 11.



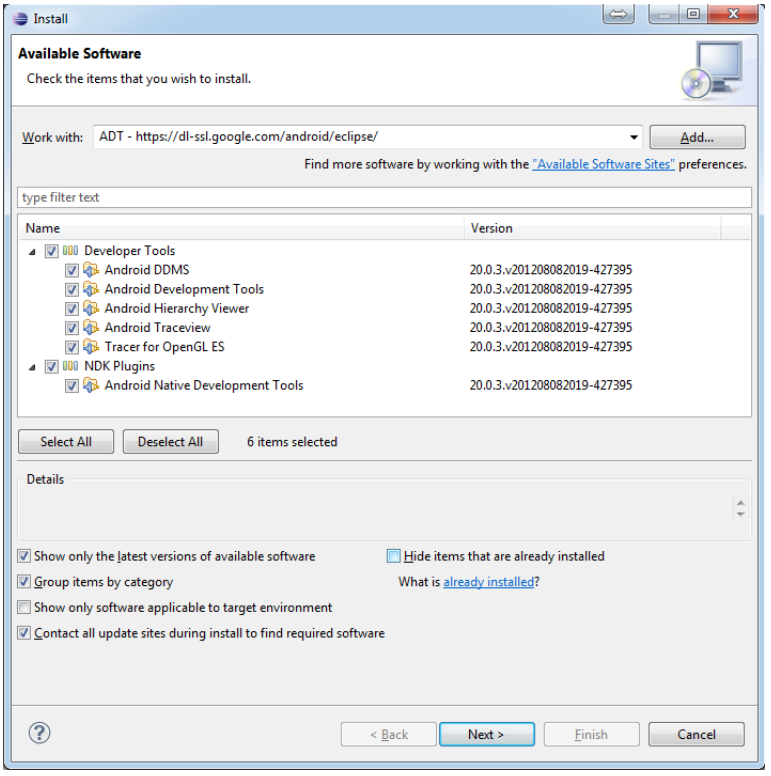
Obrázek 3.1: Potřebné balíky

- Eclipse IDE
  Oficiální stránky pro stažení http://www.eclipse.org/downloads/. Doporučena je verze **Eclipse 3.7 (Indigo)** nebo **Eclipse 4.2 (Juno)**.

- ADT Plugin pro Eclipse

  Postup je také popsán na stánkách Android Developers http://www.eclipse.org/downloads/:

  1. Pokud je již naistalovaný Eclipse zapněte ho a jděte na *Help* → *Install New Software...*
  2. Klikněte na *Add* v horní pravém rohu
  3. V *Add Respository*, který se zobrazí zadejte za název "ADT Plugin"a následující URL adresu:
     https://dl-ssl.google.com/android/eclipse/
  4. *OK*
  5. V dialogu *Available Software Dialog*, zaškrtněte *Developer Tools* a klikněte na *Next*
  6. V dalším okně uvidíte list nástrojů ke stažení, pro vývoj aplikace s NDK zatrhněte *NDK Plugins*. Klikněte na *Next*
  7. Přečtěte si licenční ujednání a klikněte na *Finish*
  8. Po kompletní instalaci restartujte Eclipse



Obrázek 3.2: Nastavení ADT pluginu

Instalace Pluginu NDK pro vývoj v C++

1. *Android NDK*
   Pro kompilování je potřeba nainstalovat NDK manažer stažitelný zde:
   http://developer.android.com/tools/sdk/ndk/index.html
   Pro instalaci je potřeba pouze rozbalit stažený archiv na nějaké místo na počítači.

2. Pro kompilování v Eclipse je dále potřeba doinstalovat CDT plugin. Nicméně pokud jste při instalaci *ADT* pluginu zvolili instalaci *NDK* pluginu, měl by již *Eclipse* tento balík potřebný ke kompilování kódu v *C++* obsahovat. Pokud ne postupujte jako při instalaci *ADT*, místo bodu 2. však zvolte "Available Software Sites" a napište pro filtrování *CDT* poté nainstalujte podobným postupem jako výše.

OPENCV4ANDROID SDK - Knihovny pro Android

Pro vývoj aplikace pro Android je potřeba knihovna OpenCV pro Android.

1. Na stránkách http://sourceforge.net/projects/opencvlibrary/files/opencv-android/ lze stáhnout poslední verzi.

2. Vytvořte novou složku, do které se rozbalí stažený balík. Pro použití s NDK by název neměl obsahovat mezery.

3. Rozbalte stažený balík do složky.

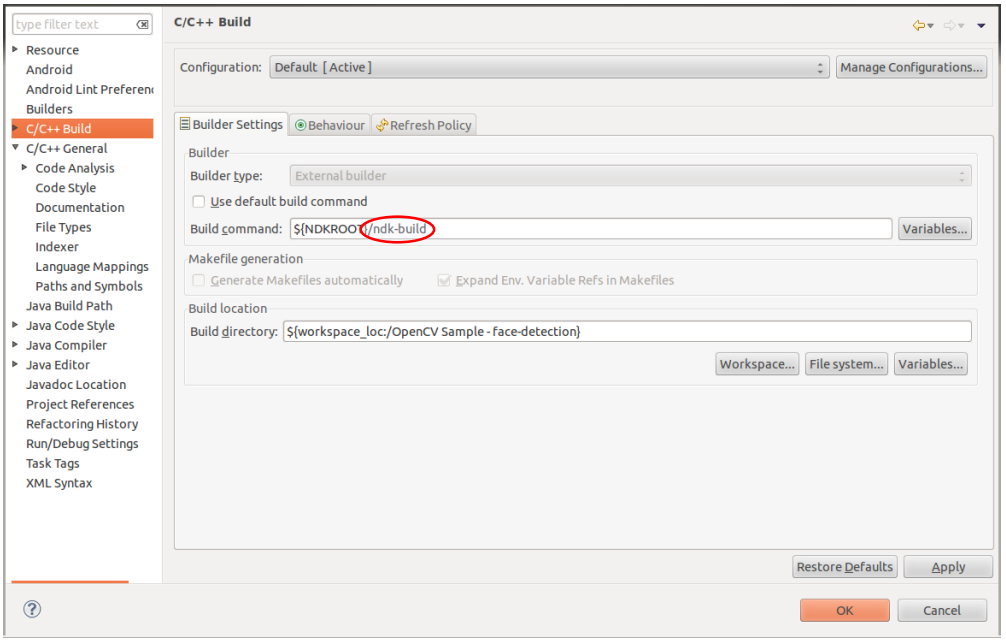
4. Importování knihovny do Eclipse
   Pro import knihoven a ukázek je následující postup.

   - Doporučuje se začínat s čistým prostředím v Eclipse.
   - Pro importování se v Eclipse v *Package Editor* klikne pravím tlačítkem a zvolí *Import* z kontextového menu.
   - Dále v *General* zvolíme *Existing Projects Into Workspace*.
   - V dalším okně v *Select root directory* vybreme složku s "OPENCV4ANDROID". Eclipse by měl automaticky najít OpenCV knihovnu a ukázky.
   - Klikněte na dlačítko *Finish*.

Práce s NDK

Pro aplikace využívající nativního kódu je potřeba upravit nastavení projektu.

- Pokud používáme jiný systém než Windows je potřeba změnit příkaz kompilující nativní kód viz obr. 3.3. To provedeme vymazáním přípony ".cmd" v nastavení: pravý klik na projekt, dále *Properties* → *C/C++ Build* → *Build command*.

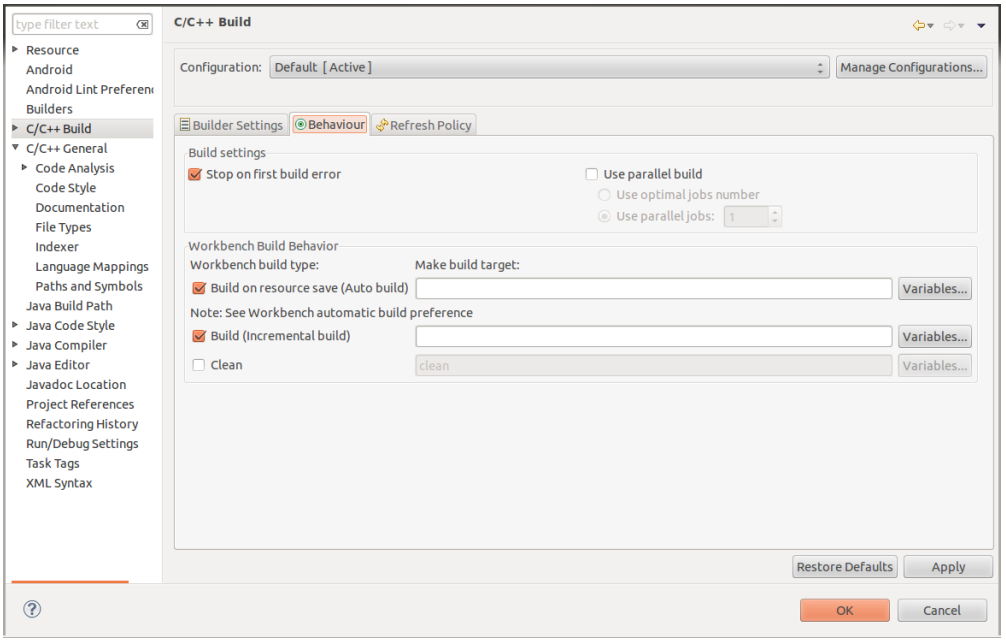


Obrázek 3.3: Příkaz pro kompilaci nativního kódu

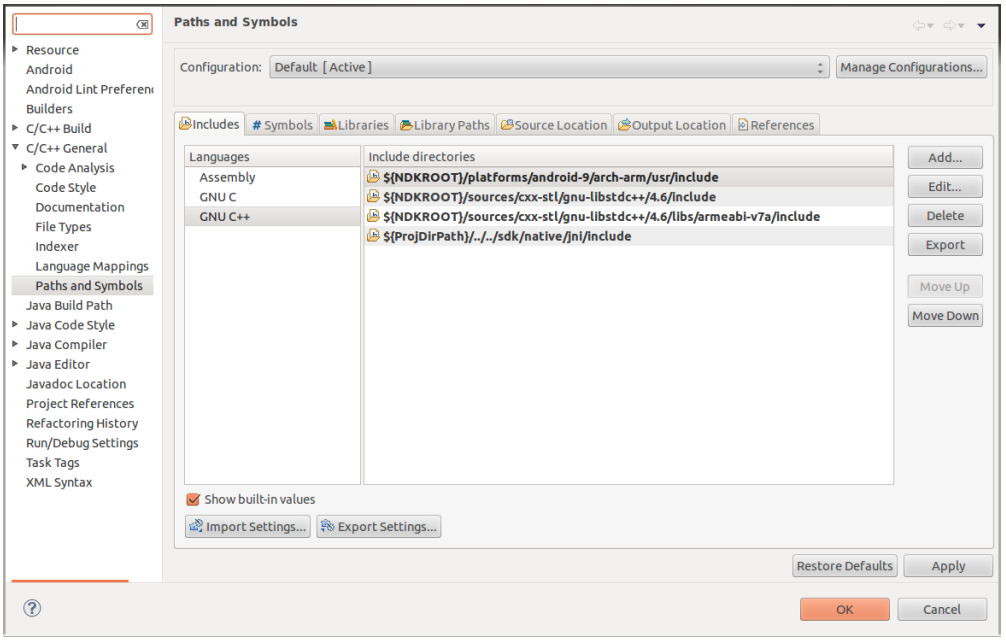
- V záložce *Behaviour* nastavíme chování podle obrázku 3.4
- Dále je potřeba nastavit knihovny. V *Properties* podle obr. 3.5 nastavíme podle verze NDK tyto knihovny:

```
# pro NDK r8 a vyšší:
${NDKROOT}/platforms/android-9/arch-arm/usr/include
${NDKROOT}/sources/cxx-stl/gnu-libstdc++/include
${NDKROOT}/sources/cxx-stl/gnu-libstdc++/libs/armeabi-v7a/include
${ProjDirPath}/../../sdk/native/jni/include

# pro NDK r8b a nižší:
${NDKROOT}/platforms/android-9/arch-arm/usr/include
${NDKROOT}/sources/cxx-stl/gnu-libstdc++/4.6/include
```



Obrázek 3.4: Nastavení chování



Obrázek 3.5: Nastavení knihoven

```
${NDKROOT}/sources/cxx-stl/gnu-libstdc++/4.6/libs/armeabi-v7a/include
${ProjDirPath}/../../sdk/native/jni/include
```

- V soubrou `Android.mk` ve složce `jni` je potřeba přidat cestu k `OpenCV.mk`. Tedy zapíšeme podle umístění např.:

  ```
  include /home/workspace/OPENCV4ANDROID/sdk/native/jni/OpenCV.mk
  ```

  pod příkaz

  ```
  include $(CLEAR_VARS)
  ```

- Dále v souboru `Aplication.mk` by měli existovat následující příkazy. Poslední dva řádky závisí na cílové platformě.

  ```
  APP_STL := gnustl_static
  APP_CPPFLAGS := -frtti -fexceptions
  APP_ABI := armeabi-v7a
  APP_PLATFORM := android-9
  ```

Ladění nativního kodu pomocí příkazového řádku

Pokud je v aplikaci obsažen nativní kód, není prakticky možné ladit kód, pomocí debugovacího nástroje v Eclipse. Nicméně můžeme jej vyvíjet pomocí standardních nástrojů pro vývoj v jazyce *C++* v příkazovém řádku, a to programu *gdb* resp. jeho modifikaci *cgdb*. Tento program umožňuje snadno vstoupit do běžícího programu v jazyce *Java*, a následně debugovat funkce, které jsou napsány v jazyce *C++*.
Program je ke stažení na http://cgdb.github.io/ nainstalujeme ho pomocí standardních nástrojů. Pro spouštění programu namísto *gdb* musíme provést změnu příkazu v souboru *ndk-gdb* ve staženém *NDK* (viz 3.1). V textu zaměníme tento příkaz:

```
GDBCLIENT=${TOOLCHAIN_PREFIX}gdb
GDBSETUP=$APP_OUT/gdb.setup
```

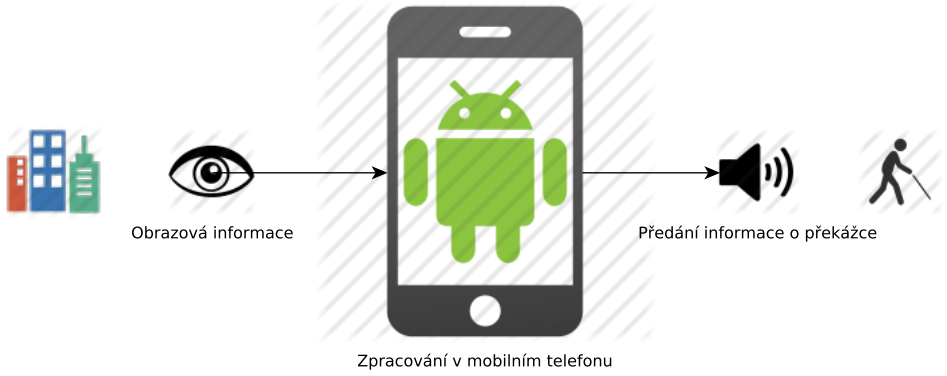
za

```
GDBCLIENT="cgdb -d ${TOOLCHAIN_PREFIX}gdb --"
GDBSETUP=$APP_OUT/gdb.setup
```

Po spuštění aplikace v *debug* módu v Eclipse, nebo pomocí jiných nástrojů, můžeme zadat v příkazové řádce `ndk-gdb` to následně spustí zmíněný program pomocí kterého, můžeme procházet běžící program.

3.2 Aplikace *NaviNevi - Navi*gace pro *Nevi*domé

3.2.1 Princip fungování



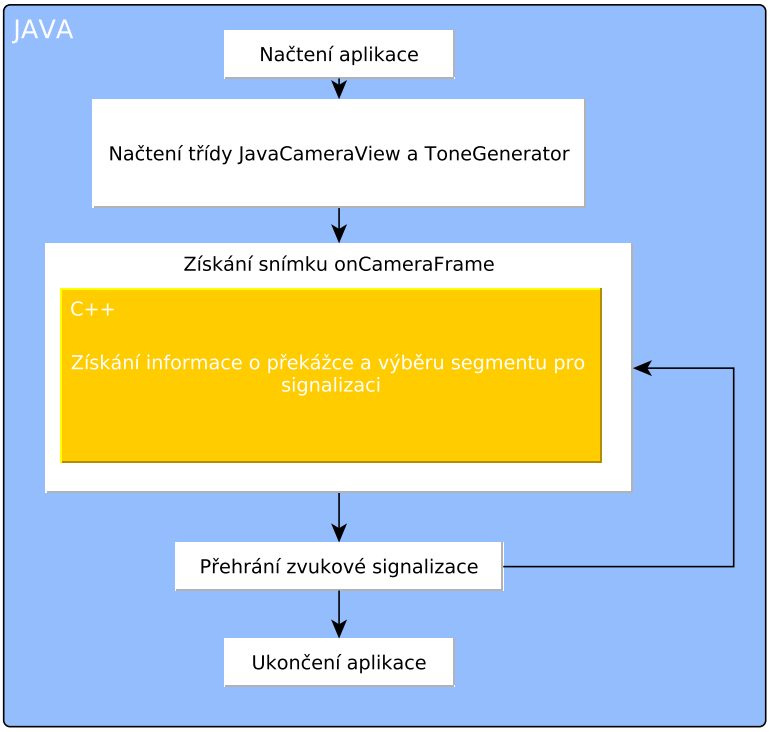
Obrázek 3.6: Princip použití pomůcky

Principem této navigace, je získávání informace pomocí obrazového senzoru na mobilním telefonu a převod této informace pomocí procesoru, na jiný vjem rozpoznatelný osobou se zrakovým postižením (dále uživatel) pokud možno v reálném čase (obr. 3.6). Uživatel tuto pomůcku používá jako nástroj k získání informace o překážce a směru, ve kterém se překážka nachází. To je mu signalizováno pomocí zvukového signálu. Uživatel si mobilní telefon upevní na hruď a kamera snímá v reálném čase prostředí před uživatelem. Pokud algoritmus vyhodnotí, že překážka je v určité vzdálenosti, přehraje mobilní telefon zvukový signál. Zvukové signály byly zvoleny z tónové volby používané v telefonii. Obraz je rozdělen na šest segmentů tři nahoře a tři dole viz 3.7.
K zjištění polohy překážky a vzdálenosti je využito několik principů popsaných v kapitole 4. práce. Chod aplikace v systému Android se řídí podle daného schematu, viz obr. 3.9

Obrázek 3.7: Rozdělení obrazu do segmentů

Jak již bylo uvedeno aplikace pracuje ve dvou jazycích. V jazyce *Java* systému *Android* jsou provedeny rutiny zajištující získání snímku z obrazového snímače, dále běh samotné aplikace, ovládání a také zajištění zvukového signálu pro nevidomého. V jazyce *C++* je zajištěno veškeré zpracování obrazové informace a předání informace pro zvukové upozornění.

Java Samotná aplikace pracuje podle schematu na obr. 3.8. Ve funkci `onCameraFrame` se získávají jednotlivé snímky, které se pak předávají do funkce s nativním kódem. Výstupem této funkce je informace pro přehrání zvuku, který informuje uživatele o tom kde je překážka. Rozhodovací rutina pak přehraje požadovaný zvuk. Na obrázku 3.10 vidíme rozmístěné zvuků pro signalizaci. Čísla představují zvuky které posílají jednotlivé klávesy na telefonu. Zvuky tónové volby byly zvoleny pro jejich snadnou implementaci pomocí třídy `ToneGenerator`. Rozmístění je dáno takové, aby byly jednotlivé segmenty snadno rozpoznatelné. Dále je zde zajištěn přenos předchozích snímků pro další výpočty.



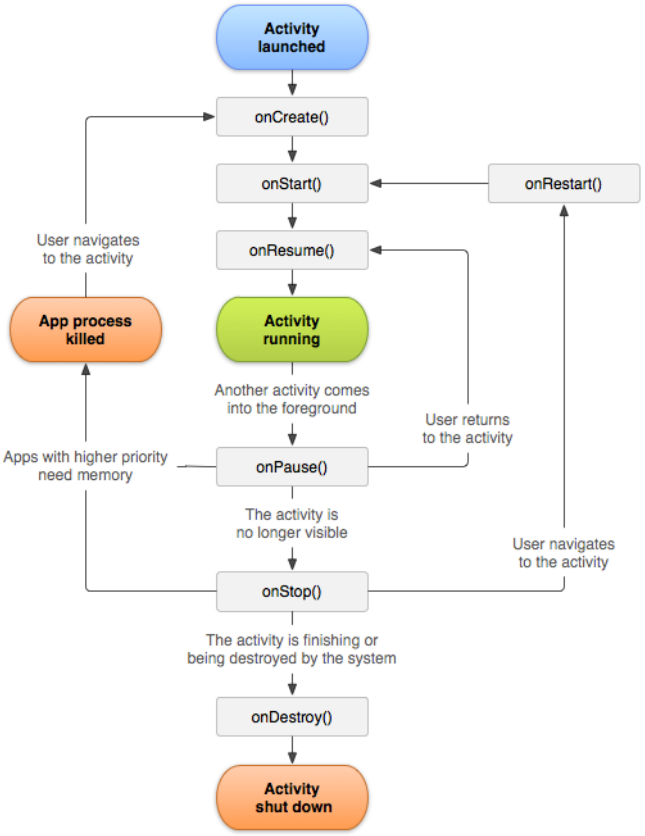
Obrázek 3.8: Chod aplikace

C++ V jazyce *C++* je napsaná podstatná část algoritmu. Jeho fungování popisuje vývojový diagram 3.11.

1. Do funkce vstupují dva snímky, aktuální a předchozí. Tyto snímky jsou předány z *Java* části aplikace.

2. Z obrázků, se jednou za daný počet snímků, vypočítají body v obraze vhodné pro trasování (funkce `goodFeaturesToTrack` implementovaná v OpenCV) viz část 2.2.2. Doba za jakou se počítají znovu body v obraze je důležitá pro trasování. Po nastavenou dobu totiž další funkce počítají pouze s pomocí předchozích napočítaných bodů z funkce `calcOpticalFlow`, která zajišťuje trasování. Zvolení doby je popsáno v části *Testování* 3.3.

3. Napočítané body `prevPoints` jsou buď získány z předchozí funkce, nebo pokud se zrovna nepočítají, použijí se z předchozího běhu funkce (snímku).



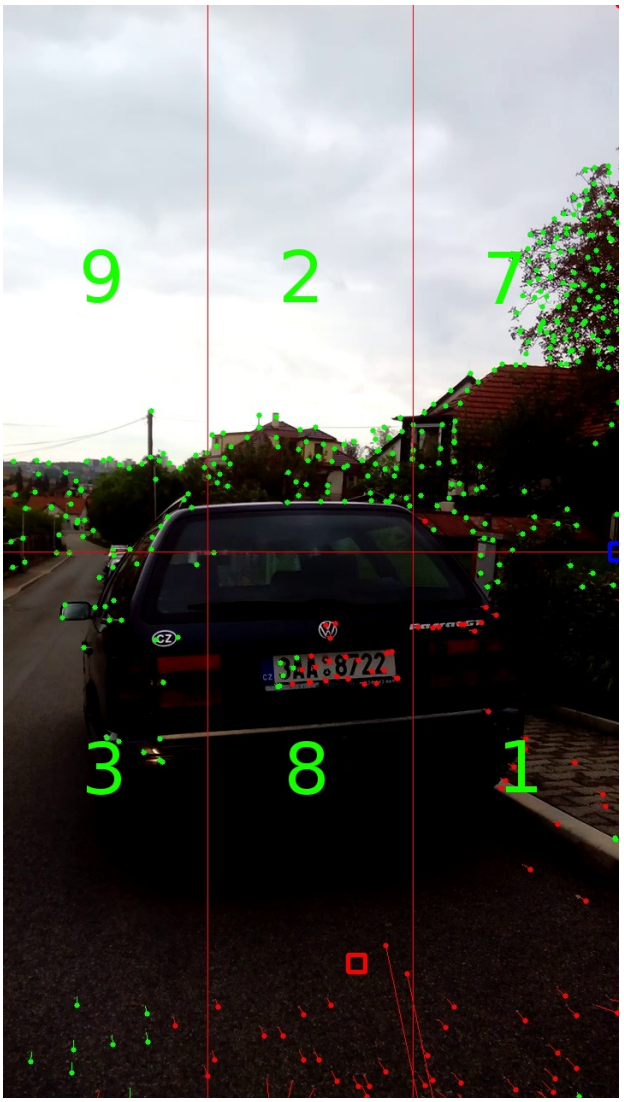
Obrázek 3.9: Standardní chod aplikace pro Android.

Funkce `calcOpticalFlowLK` používá tyto body k zjištění optického toku, resp. vypočte body v aktuálním snímku, které odpovídají pohybu v obraze.

4. V další části vstupují body do **for** cyklu, který prochází vektor bod po bodu. V tomto cyklu se vypočítá pro každý bod divergence - amplituda pohybu každého bodu (amplituda optického toku). Pro potřeby testování jsou zde také vykresleny body do obrazu.

5. Následující část určí minimum této amplitudy, což odpovídá *ohnisku rozšíření - FOE - Focus of Epansion*. Je to bod v obraze, kde je optický tok nejmenší (část 2.2.2. Na tento bod je aplikován Kalmanův filtr (popis viz například [26] původní článek zde [4]). Pomocí tohoto filtru je snaha eliminovat velké změny FOE v obraze. Filtr predikuje z naměřené polohy FOE budoucí polohu bodu tohoto bodu.

6. Následuje další cyklus, který prochází body a počítá tzv. okamžik kontaktu (TTC), tedy vzdálenost mezi FOE a jednotlivými body, viz kapitola 2.2.2. V tomto cyklu jsou přiřazeny body do svých segmentů, podle pozice v obraze.

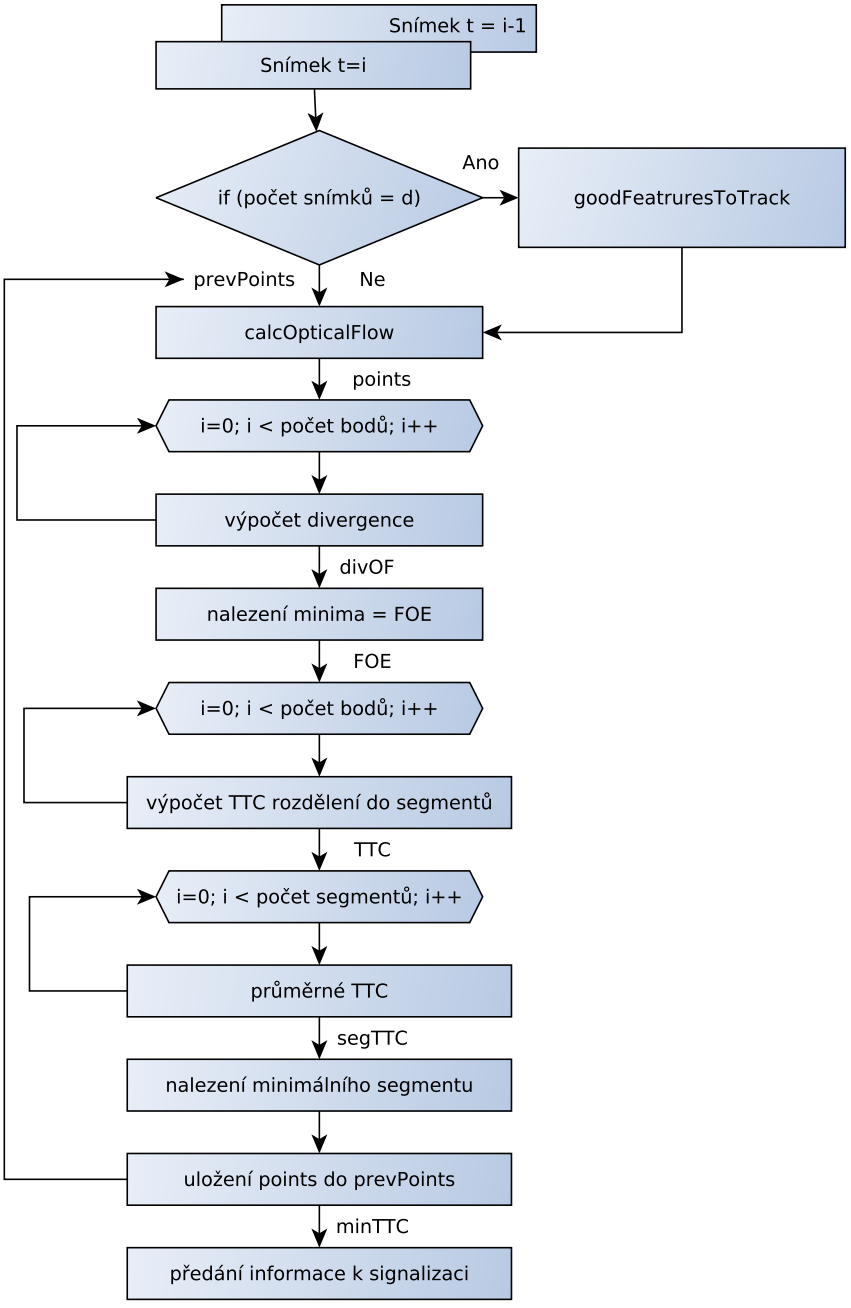


Obrázek 3.10: Rozmístění tónové volby DTMF.

7. V tomto kroku se vypočítá průměr TTC pro daný segment. Následně se najde v jakém segmentu je průměrné TTC nejmenší a podle toho se určí, který segment se bude signalizovat. Informace se uloží do proměnné, podle které se určuje zvuková signalizace.

8. Posledním krokem, je uložení aktuálních bodů jako předchozích, pro další běh funkce.

Předchozí body jsou uloženy v paměti jako statická proměnná. Při dalším běhu funkce se totiž program vrací do části v jazyce *JAVA*, aby načetl další snímek, přečetl uloženou informaci o signalizaci a přehrál požadovaný zvuk. Dále program opět vstoupí do funkce v *C++* a cyklus se opakuje.



Obrázek 3.11: Průchod funkcí pro výpočet optického toku, FOE, TTC a signalizace.

3.3 Testování aplikace

Při testování bylo využito přenositelnosti hlavní části aplikace, která je napsána v jazyce *C++* a hlavní test se provedl na desktopu počítače. K testům se využilo video natočené přímo mobilním telefonem. Důvod k testování na počítači byl snadná obsluha a rychlost testování. Kód doznal jen malých změn, kterých bylo zapotřebí ke spouštění a ladění programu na počítači, základ aplikace je stejný, jen se místo zvukové signalizace využilo obrazového upozornění přímo ve videu. Dále byli provedeny testy přímo s mobilním telefonem.

3.3.1 Testování v počítači

K nalezení výsledného segmentu vede celá řada výpočtů, u kterých je možné různé nastavení. Z toho důvodu byly provedeny testy, které mají za cíl nalezení nejvhodnějších konstant pro chod aplikace. Testy byli provedeny pro tyto stupně volnosti:

1. Počet snímků, za které se opět nastaví vzory pro trasování (funkce `goodFeaturesToTrack`). Počet snímků způsobuje, jak často se napočítají body v celém obraze, nejmenší možný počet je dva, maximum nelze jednoznačně stanovit. Nicméně jak obraz plyne, vzory v obraze, které byli detekovány se ztrácejí. Za čas se ztratí úplně, proto již není co trasovat. Výpočet bodů, je poměrně náročný na výkon zařízení a zpomaluje běh programu. Při počtu snímků dva, je například prodleva mezi těmito dvěma snímky poměrně velká, tudíž se ztrácí informace v obraze.
2. Počet bodů, které se napočítají pomocí funkce `goodFeaturesToTrack`. Tento parametr je důležitý, jelikož určuje kolik bodů se bude pomocí optického toku sledovat. Při nízkém počtu bodů je informace pouze u pár vzorů, tudíž nepostihne mnoho případných překážek. Při velkém počtu bodů je funkce náročnější na výpočetní výkon.

Základem testů jsou dvě videa. Jedno natočené při chůzi a jedno z plynule se pohybujícího vozíku. Videa byli natáčeny ve vnitřních prostorách. Během měření se zaznamenávaly detekované segmenty v obraze pro hodnoty volnosti, které jsou uvedeny v tabulce. Předpokládané využití aplikace je při chůzi. Nicméně během testování bylo zjištěno, že výpočet optického toku je velmi ovlivněn otřesy při chůzi, proto byl zvolen i test za pomocí plynulého pohybu.

Při testování bylo využito vizualizace vypočtených bodů přímo do obrazu. Na následujících obrázcích (3.12) jsou snímky při testování. Body v obraze jsou aktuální body vypočtené

Stupně volnosti	
Počet snímků	Napočítané body
2	10
10	100
20	200
30	500
50	100

pomocí funkce `calcOpticalFlow`, čáry vedoucí od bodů signalizují velikost a směr optického toku. Červený čtverec signalizuje vypočtené FOE. Modrý čtverec ukazuje do pravého horního rohu segmentu, který byl detekovaný jako potenciálně ohrožující.

Obrázek 3.12: Snímky z testování programu. Vpravo pro 10 napočítaných bodů, vlevo pro 1000.

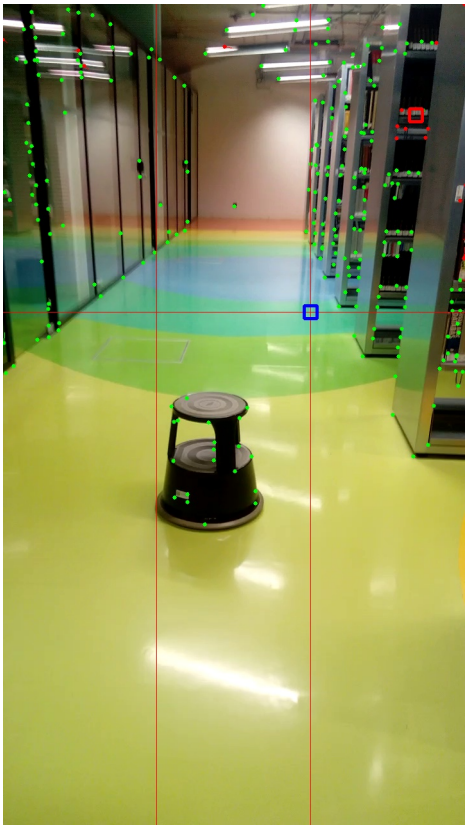
3.3.2 Výsledky

Jak již bylo zmíněno výpočet optického toku je ovlivněn prudkými změnami v obraze, které jsou pohybem kamery v jiném směru než je směr pohybu. Výpočet FOE by měl zajisti správnou interpretaci pokud kamera vykonává translační pohyb a nepohybuje se v ose ohniska kamery. Bohužel není schopen eliminovat pohyby kamery v krátkém časovém horizontu, tudíž otřesy během chůze silně narušují výpočet. Při pozorování

videa natočeného při chůzi je zřejmé, že okamžiky kdy se kamera pohybuje ve směru FOE, jsou v obraze přítomny jen velmi zřídka. Při pohybu pomocí vozíku je pohyb již plynulejší a proto se dá předpokládat, že bude lépe fungovat teorie, které je implementována.

Testovací videa byly zvoleny jako průchod chodbou s překážkou. Jelikož byly natočeny ve vnitřních prostorách, projevují se v daném algoritmu nedostatky. Zejména je to zahrnutí do detekování překážek i prostor stropu. U vnitřních prostor se dá předpokládat, že strop bude přibližně stejně vzdálen od kamery jako podlaha. Pokud je strop členitý, zahrnuje se do výpočtu jako překážky, které ovšem nejsou nijak potenciálně nebezpečné.

Z hodnot pro celou sekvenci snímků videa byly vypočtena hustota pravděpodobnosti, které určují počet výskytů signalizace pro daný segment. Tedy jak často byl segment zvolen jako potenciálně ohrožující. V příloze jsou tabulky pro jednotlivé měření. Je zde vidět, že nejčastěji jsou voleny segmenty v horní části obrazu, kde dochází k detekci členitého stropu. V prvních okamžicích videa, je překážka také z části v prostředním segmentu, viz obrázek 3.14 vlevo, proto detekce odpovídá. Jelikož dále se překážka ve videu objeví v prostředním segmentu dole (segment 4) viz obr. 3.14 vpravo, dala by se předpokládat větší četnost, nicméně jak je z tabulek zřejmé není tomu vždy tak.



Obrázek 3.13: Jiné rozdělení segmentů dolní segmenty jsou ve vertikálním směru větší

Pokud budeme brát jako parametr úspěšnosti četnost výskytu signalizace ve čtvrtém segmentu, kde je z hlediska potenciální srážky překážka nejzřejmější, aplikace v tomto

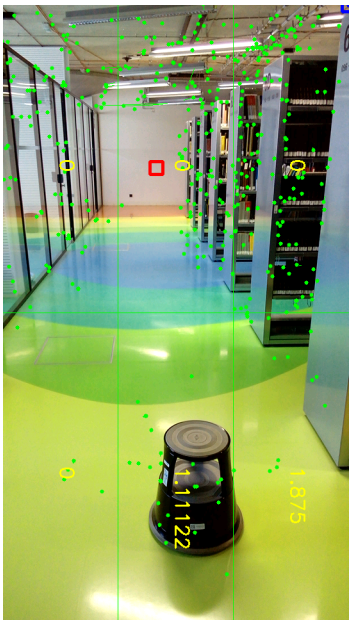
selhává. Nicméně pokud vezmeme v potaz komplexnost obrazové informace (členité prostředí na stropě i po stranách) algoritmus funguje tak, jak by se dalo předpokládat. Řešení tohoto problému, by bylo jiné rozmístění segmentů, jak je to naznačeno v obrázku 3.13 a stanovení menší váhy při výpočtu pro horní segmenty.

Obrázek 3.14: Umístění překážky v obraze

Některé z nastavení volnosti omezuje funkčnost, zejména v souvislosti výpočtu bodů vhodných pro trasování. Tato funkce je poměrně výpočetně náročná a proto úměrně tomu, jak často se body počítají, stoupá čas nutný k zpracování jednotlivých snímků. Dochází tedy k prodlevám, které způsobí, že se zvětší počet snímků získaných za vteřinu a tedy časové rozlišení.

3.3.3 Testování na mobilním telefonu

Testování proběhlo při stejném schematu a prostředí jako při nahrávání videa pro počítač. Jelikož, v mobilní aplikaci je stejná implementace jako na počítači, dali se očekávat stejné výsledky. Proběhly pokusy pohybovat se s cílem vyhnout se překážce.
Před samotným testováním je nejprve nutné se naučit způsob signalizace. Jelikož signalizace je pomocí tónové volby, uživatel musí vědět jaký tón je pro jaký segment. Tónová volba pro jednotlivé segmenty byla zvolena tak, aby jednotlivé tóny byly v co



Obrázek 3.15: Obrazovka při testování, čísla u jednotlivých segmentů znamenají průměrný okamžik kontaktu pro daný segment

nejvzdálenějším intervalu a tedy co nejlépe rozpoznatelné. Při testování bylo dobře patrné jaký segment je právě zvolen, tedy který obsahuje překážku. Výsledky jsou velmi podobné testování na počítači. Nejčastěji se ozývají horní segmenty, občas segment obsahující překážku, což odpovídá rozložení četnosti pro jednotlivé segmenty naměřené na počítači.

Kapitola 4

Závěr

V úvodu této práce jsem se snažil šířeji nastínit problematiku pomůcek pro zrakově postižené. Při dnešních technických a výpočetních možnostech se nabízí celá řada principů, které mohou být využity pro vývoj těchto pomůcek. Mnoho výzkumných center a universit se snaží vyvinout nějakou pomůcku, která by usnadnila zrakově postiženým a nevidomým orientaci v prostředí, v kterém žijí. Z průzkumu je zřejmé, že před náročnější vývojem takové pomůcky je potřeba dobře zvážit, zda jí budou nevidomí schopni opravdu využívat. Zdá se, že většina pomůcek končí ve stadiu vývoje a nedostane se do praktického užívání. Důvod proč nejsou elektronické pomůcky více využívány, je také zřejmě způsoben tím, že standardními pomůcky jsou poměrně dobře funkční a dostupné. V neposlední řadě je na ně komunita zrakově postižených zvyklá.
V teoretické části, je také uvedena rešerše na aktuální vývoj pomůcek, na univerzitách. Z tohoto průzkumu je dobře patrný široký záběr, který tato problematika má a množství fyzikálních principů, které lze využít. Konkrétněji je zde rozebrána problematika využití obrazové informace pro detekci překážek.
Při rozhodování, který princip detekce využít, se braly v úvahu možnosti mobilního telefonu. Zejména získání obrazové informace jedním obrazovým snímačem a výpočetní náročnost algoritmu. Ačkoliv mobilní telefony mají již dnes velmi velký výpočetní výkon, pro složitější výpočty je v některých případech stále nedostačující.
V počátcích rozhodování jakou metodu zvolit, připadala v úvahu také detekce překážek pomocí *moodelu salience*, který je v práci také popsán. Důvod, proč nebyla navigace zrealizována pomocí tohoto modelu, byl právě jeho velká výpočetní náročnost. Pro výběr detekce překážek pomocí optického toku, jsem se inspiroval využitím této metody k navigaci v robotice, kde se tento princip často objevuje. Dvě nejčastěji používané metody jsou popsány v teorii. Pro výběr metody *Lukas-Kanade* hovoří opět menší výpočetní náročnost.
Při vývoji aplikace jsem vyzkoušel několik metod pracovního prostředí a způsobu vývoje.

Jako problémový se jeví vývoj aplikace v systému *Android* s nativním kódem v *C++*. Nakonec se mi podařilo stanovit pracovní postup takový, že jsem mohl poměrně bez problémů ladit kód v obou jazycích. Postup nastavení jsem shrnul na začátku praktické části. Do budoucna by bylo vhodné nelpět na grafických vývojových prostředích, a vyvíjet aplikaci přes terminál operačního systému, což se jeví kupodivu jako rychlejší a bezproblémovější způsob, ovšem za cenu nutnosti se naučit nové postupy.

Jak vyplývá z výsledků testování, samotná aplikace vykazuje problémy ve funkčnosti. Důvodů je více. Zejména, jak je popsáno, prudké změny v optickém toku, způsobené otřesy kamery při chůzi. Při testování plynulého pohybu byly výsledky lepší, nicméně ne dostačující k navigaci. Další důvod je velké množství stupňů volnosti systému a jejich nastavení. Při testech jsem se snažil vyzkoušet různá nastavení, nicméně nepodařilo se mi nalézt nejvhodnější parametry, které by umožňovaly dostačující funkčnost.

Nabízí se několik možností vylepšení algoritmu. První je implementace optického toku pomocí pyramidové reprezentace vstupního obrazu. Toto vylepšení umožní částečně eliminovat problém při výpočtu optického toku, tedy disbalanci mezi přesností a robustností. Výpočet těchto pyramid, je ale velmi výpočetně náročný, a bylo by potřeba většího výpočetního výkonu. Problém zaznamenaný při testování ve vnitřních prostorách, tedy detekce stropu, by bylo možné eliminovat navrženou metodou a to jiné dělení segmentů a následném váhování segmentů.

Nabízí se otázka, zda-li místo optického toku pro napočítané body pomocí algoritmu pro hledání vzorů, nepočítat optický tok pro všechny pixely v obraze pomocí popsané metody *Horn-Schunck*. Opět zde ale narážíme na problém výpočetní náročnosti tohoto algoritmu. Dále je také jistě na zvážení, zda-li lpět na počítání veškerých operací v reálném čase. Pokud by se od této podmínky upustilo, mohly by se zde navržené metody implementovat a tím vylepšit možnosti navigace. Všechny navržené metody vyžadují rozsáhlejší testování a hledání parametrů, které by optimalizovali navigaci tak aby byla vhodná k použití.

Na závěr bych rád zmínil, přínos této práce pro mě. Je to zejména seznámení se základy programování na platformě *Android*, proniknutí do programování v jazyce *C++* a důkladnější pochopení knihovny pro zpracování obrazu *OpenCV*. V budoucnu bych se dále rád věnoval vyvíjení aplikací se zpracováním obrazu na této platformě.

Příloha A

Tabulky

#	Počet snímků	Počet bodů	seg 0	seg 1	seg 2	seg 3	seg 4	seg 5
1	2	10	25	32	53	0	1	0
2	10	10	29	32	48	0	1	0
3	20	10	38	35	38	0	0	0
4	30	10	14	34	63	0	0	0
5	50	10	48	39	24	0	0	0
6	2	100	40	19	37	6	8	1
7	10	100	32	23	47	2	6	1
8	20	100	26	26	49	2	5	3
9	30	100	25	27	50	2	4	3
10	50	100	30	20	51	0	3	7
11	2	200	32	21	29	24	3	2
12	10	200	26	21	41	14	5	4
13	20	200	18	15	73	3	2	0
14	30	200	29	19	43	14	3	3
15	50	200	20	32	26	26	3	4
16	2	500	20	25	35	23	5	3
17	10	500	17	33	36	20	4	1
18	20	500	19	12	69	8	3	0
19	30	500	24	20	39	19	2	7
20	50	500	23	31	24	25	4	4
21	2	1000	29	24	31	21	3	3
22	10	1000	28	27	29	21	3	3
23	20	1000	21	11	66	9	4	0
24	30	1000	40	14	31	19	0	7

Tabulka A.1: Tabulka četnosti výskytu pro video jízdy vozíku ve vnitřních prostorech

#	**Počet snímků**	**Počet bodů**	**seg 0**	**seg 1**	**seg 2**	**seg 3**	**seg 4**	**seg 5**
1	2	10	48	20	68	0	0	0
2	10	10	51	22	63	0	0	0
3	20	10	59	34	43	0	0	0
4	30	10	54	0	82	0	0	0
5	50	10	39	46	23	19	3	6
6	2	100	58	41	23	13	1	0
7	10	100	41	42	28	18	6	1
8	20	100	72	21	35	4	4	0
9	30	100	43	25	53	11	4	0
10	50	100	41	52	11	5	8	19
11	2	200	41	33	30	7	12	13
12	10	200	50	25	35	13	4	9
13	20	200	55	16	49	6	2	8
14	30	200	50	20	52	3	9	2
15	50	200	36	49	25	15	4	7
16	2	500	47	40	19	19	4	7
17	10	500	41	47	29	16	0	3
18	20	500	63	27	38	4	0	4
19	30	500	48	19	52	7	2	8
20	50	500	33	48	21	24	3	7
21	2	1000	37	34	35	14	5	11
22	10	1000	59	12	36	16	4	9
23	20	1000	39	13	60	18	0	6
24	30	1000	52	6	56	7	8	7

Tabulka A.2: Tabulka četnosti výskytu pro video chůze ve vnitřních prostorech

Literatura

[1] World health organization. Říjen 2013. URL http://www.who.int/mediacentre/factsheets/fs282/en/.

[2] Středisko výcviku vodících psů sons Čr. Únor 2013. URL http://www.vodicipsi.cz/jakziskatpsa.htm.

[3] Massimo Bertozz, Alberto Broggi, and Alessandra Fascioli. Stereo inverse perspective mapping: theory and applications. *Image and Vision Computing*, 16(8):585 – 590, 1998. ISSN 0262-8856. doi: http://dx.doi.org/10.1016/S0262-8856(97)00093-0. URL http://www.sciencedirect.com/science/article/pii/S0262885697000930.

[4] Christof Born. Determining the focus of expansion by means of flowfield projections. *In Proc. Deutsche Arbeitsgemeinschaft fur Mustererkennung DAGM'94*, pages 711–719, 1994.

[5] N.S. Boroujeni, S.A. Etemad, and A. Whitehead. Fast obstacle detection using targeted optical flow. *Image Processing (ICIP), 2012 19th IEEE International Conference on*, pages 65–68, Sept 2012. ISSN 1522-4880. doi: 10.1109/ICIP.2012.6466796.

[6] Dr. Gary Rost Bradski and Adrian Kaehler. Learning opencv, 1st edition. 2008.

[7] Ted Camus. Real-time quantized optical flow. *Real-Time Imaging*, 3(2):71 – 86, 1997. ISSN 1077-2014. doi: http://dx.doi.org/10.1006/rtim.1996.0048. URL http://www.sciencedirect.com/science/article/pii/S1077201496900480.

[8] C. Harris and M. Stephens. A combined corner and edge detector. *Proceedings of the 4th Alvey Vision Conference*, pages 147–151, 1988.

[9] M. Hersh and M.A. Johnson. Assistive technology for visually impaired and blind people. 2010. URL http://books.google.cz/books?id=HIBPnRAhh-gC.

[10] Berthold K.P. Horn and Brian G. Schunck. Determining optical flow. *Artificial Intelligence*, 17(1–3):185 – 203, 1981. ISSN 0004-3702. doi: http://dx.doi.org/

10.1016/0004-3702(81)90024-2. URL http://www.sciencedirect.com/science/article/pii/0004370281900242.

[11] Andreas Hub, Joachim Diepstraten, and Thomas Ertl. Design and development of an indoor navigation and object identification system for the blind. *SIGACCESS Access. Comput.*, (77-78):147–152, September 2003. ISSN 1558-2337. doi: 10.1145/1029014.1028657. URL http://doi.acm.org/10.1145/1029014.1028657.

[12] T. Ifukube, T. Sasaki, and C. Peng. A blind mobility aid modeled after echolocation of bats. *Biomedical Engineering, IEEE Transactions on*, 38(5):461–465, May 1991. ISSN 0018-9294. doi: 10.1109/10.81565.

[13] Kiyohide Ito, Makoto Okamoto, Junichi Akita, Tetsuo Ono, Ikuko Gyobu, Tomohito Takagi, Takahiro Hoshi, and Yu Mishima. Cyarm: An alternative aid device for blind persons. *CHI '05 Extended Abstracts on Human Factors in Computing Systems*, pages 1483–1488, 2005. doi: 10.1145/1056808.1056947. URL http://doi.acm.org/10.1145/1056808.1056947.

[14] L. Itti, C. Koch, and E. Niebur. A model of saliency-based visual attention for rapid scene analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(11):1254–1259, Nov 1998.

[15] D. K. Liyanage and M. U S Perera. Optical flow based obstacle avoidance for the visually impaired. *Business Engineering and Industrial Applications Colloquium (BEIAC), 2012 IEEE*, pages 284–289, April 2012. doi: 10.1109/BEIAC.2012.6226068.

[16] Bruce D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. pages 674–679. William Kaufmann, 1981. URL http://dblp.uni-trier.de/db/conf/ijcai/ijcai81.html#LucasK81.

[17] S. Meers and K. Ward. A vision system for providing 3d perception of the environment via transcutaneous electro-neural stimulation. *Information Visualisation, 2004. IV 2004. Proceedings. Eighth International Conference on*, pages 546–552, July 2004. ISSN 1093-9547. doi: 10.1109/IV.2004.1320198.

[18] P.B.L. Meijer. An experimental system for auditory image representations. *Biomedical Engineering, IEEE Transactions on*, 39(2):112–121, Feb 1992. ISSN 0018-9294. doi: 10.1109/10.121642.

[19] G. Sainarayanan, R. Nagarajan, and Sazali Yaacob. Fuzzy image processing scheme for autonomous navigation of human blind. *Applied Soft Computing*, 7(1):257 – 264, 2007. ISSN 1568-4946. doi: http://dx.doi.org/10.1016/j.asoc.2005.06.005. URL http://www.sciencedirect.com/science/article/pii/S1568494605000670.

[20] J. Shi and C. Tomasi. Good features to track. *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR '94., 1994 IEEE Computer Society Conference on*, pages 593–600, Jun 1994. ISSN 1063-6919. doi: 10.1109/CVPR.1994.323794.

[21] N. Suganuma and N. Fujiwara. An obstacle extraction method using virtual disparity image. *Intelligent Vehicles Symposium, 2007 IEEE*, pages 456–461, June 2007. ISSN 1931-0587. doi: 10.1109/IVS.2007.4290157.

[22] Zehang Sun, G. Bebis, and R. Miller. On-road vehicle detection using optical sensors: a review. *Intelligent Transportation Systems, 2004. Proceedings. The 7th International IEEE Conference on*, pages 585–590, Oct 2004. doi: 10.1109/ITSC.2004.1398966.

[23] Sovira Tan, Jason Dale, Andrew Anderson, and Alan Johnston. Inverse perspective mapping and optic flow: A calibration method and a quantitative analysis. *Image and Vision Computing*, 24(2):153 – 165, 2006. ISSN 0262-8856. doi: http://dx.doi.org/10.1016/j.imavis.2005.09.023. URL http://www.sciencedirect.com/science/article/pii/S026288560500185X.

[24] I. Ulrich and J. Borenstein. The guidecane-applying mobile robot technologies to assist the visually impaired. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 31(2):131–136, Mar 2001. ISSN 1083-4427. doi: 10.1109/3468.911370.

[25] Dirk Walther and Christof Koch. Modeling attention to salient proto-objects. *Neural Networks*, 19(9):1395 – 1407, 2006. ISSN 0893-6080. doi: http://dx.doi.org/10.1016/j.neunet.2006.10.001. URL http://www.sciencedirect.com/science/article/pii/S0893608006002152.

[26] Greg Welch and Gary Bishop. An introduction to the Kalman filter. Technical Report TR 95-041, University of North Carolina at Chapel Hill, Department of Computer Science, 1995.

[27] Jean yves Bouguet. Pyramidal implementation of the lucas kanade feature tracker. *Intel Corporation, Microprocessor Research Labs*, 2000.

[28] Liu Zeyu, Yu Chunxuan, and Zheng Banggui. Any type of obstacle detection in complex environments based on monocular vision. *Control Conference (CCC), 2013 32nd Chinese*, pages 7692–7697, July 2013.