Univerzita Hradec Králové

Fakulta informatiky a managementu

KIKM

Rozpoznávání obrazu a autonomní řízení mobilního robota

Bakalářská práce

Autor: Ivan Melnykovych
Studijní obor: Aplikovaná informatika

Vedoucí práce: Ing. Pavel Kříž, Ph.D.

Hradec Králové srpen 2019

Prohlášení:

Prohlašuji, že jsem bakalářskou práci zpracoval samostatně a s použitím uvedené literatury.

V Hradci Králové dne 14.8.2019 Ivan Melnykovych

Poděkování:

Děkuji vedoucímu bakalářské práce Ing. Pavlu Křížovi, Ph.D., za metodické vedení práce, pravidelné konzultace, odborné rady a trpělivost. Mé poděkování patří též Ing. Brunovi Ježkovi, Ph.D. za konzultace a odborné rady.

Anotace

Tato bakalářská práce je zaměřená na detekci dlaždice v prostoru, který je sledován robotem pomocí sledovacího zařízení s použitím konvolučních neuronových sítí pro poskytnutí funkce rozpoznávání obrazu. Neuronová síť bude fungovat na základě algoritmu nalezení Haarových příznaků. Následně je také součástí úkolu implementovat spolehlivé a rychlé rozpoznávání obrazu s použitím výše uvedených příznaků. K implementaci této funkčnosti bude použita velice známá knihovna OpenCV pro programovací jazyk Python. Dalším úkolem je testovat výsledný model na data zaznamenávaná ručně a také na data sbíraná pomocí mobilního robota. Součástí práce byl sběr dostatečného množství dat pro poskytnutí vstupu do konvoluční neuronové sítě. Výsledkem této práce je model neuronové sítě a zároveň jeho implementace do aplikace rozpoznávání obrazu a poskytnutí základních funkcí a informací pro pohyb mobilního robota. Další implementovanou funkcí je detekce významných bodů v pohybujícím se obraze pro účel vizuální odometrie mobilního robota a počítání jeho ujeté vzdálenosti od počátku. Pro nalezení významných oblastí je použit RANSAC algoritmus.

Klíčová slova

Neuronové sítě, konvoluční neuronové sítě, rozpoznávání obrazu, OpenCV, Haarovy příznaky, významné body, RANSAC

Annotation

Title: Image Recognition and Autonomous Mobile Robot Control

This bachelor thesis is focused on detection of tiles in space, which is being monitored by a robot with help of tracking device with use of convolutional neural networks to provide a function of image detection. Neural network will function based on finding Haar feature algorithm. Another part of the task is to implement fast and reliable image recognition with use of the features mentioned above. Implementation of this functionality will use well known library OpenCV for programming language Python. Next task will be testing final model on handpicked data and also on data acquired from the mobile robot. Part of the thesis was collection of a sufficient amount of data to provide input to convolutional neural network. Result of this thesis is a model of neural network and also its implementation into the application for image recognition and provision of basic functions and information for the movement of mobile robot. Another implemented function is detection of important points and areas in moving image for the purpose of visual odometry of mobile robot and measuring his driven distance from the beginning point. RANSAC algorithm will be used to find important areas.

Keywords

Neural network, convolutional neural network, image recognition, OpenCV, Haar Cascades, features detector, RANSAC

Obsah

Seznam obrázků

Seznam tabulek

1 Úvod

Tato práce se zabývá rozpoznáváním obrazu podle snímků, které bude pořizovat mobilní robot během svého pohybu po dlaždicové podlaze (např. v budově FIM UHK). V dnešní době je rozpoznávání obrazu široce používáno v různých oblastech začínajících detekováním obličejů lidí pomocí fotoaparátu mobilu až po rozpoznávání SPZ a měření rychlosti vozidel, která projela kolem kamery. Ale je pravda, že detekce dlaždic, podle kterých by mobilní robot měl řídit svůj pohyb, není úplně triviální a jednoduchá úloha. Moderní technologie se stále rozvíjí a zejména v kybernetice jsou dnes jednou z nejdůležitějších oblastí neuronové a také konvoluční neuronové sítě, dále používané pro rozpoznávání obrazu anebo řešení problematiky počítačového vidění.

1.1 Důvod výběru tématu práce

Hlavním aspektem pro výběr tohoto tématu byl můj osobní zájem pochopit důležitou a zajímavou látku „deep learning“ a „artificial intelligence“, která je v dnešní době široce používána nejen pro řízení robotů, ale i pro spoustu jiných věcí. Google používá neuronové sítě pro rozpoznávání obličeje, hledání podobných obrázků.

Počítačové vidění je jedna z nejvíce používaných a nejprogresivněji se rozvíjejících oblastí IT.

1.2 Cíl práce

Hlavním cílem této práce je navrhnout a implementovat řešení pro rozpoznávání obrazu z kamery sledující povrch před mobilním robotem. Povrch (podlaha) bude mít předem známý vzor – dlaždičky.

Rozpoznávání obrazu by mělo být spolehlivé, kvalitní a rychlé. Pro rozhodovací algoritmus řízení robota je potřeba detekovat dlaždice, které jsou zaznamenané pomocí kamery a nachází se před mobilním robotem. Je potřeba zajistit pohled pouze na ty dlaždice, které jsou umístěny přímo na cestě, kde se robot pohybuje.

Z obrazu musí být odvozena a pro další použití uložena ujetá vzdálenost robota a případně záznam o potížích, které nastaly. Bude implementován algoritmus řízení robota podle předem navrženého vektoru k zadanému cíli pomocí rozpoznávání obrazu. Výsledný systém by měl být testován pro zjištění a odstranění případných chyb.

2 Rešerše existujících řešení

V současné době jsou vyvíjeny stále novější a dokonalejší metody a algoritmy pro detekování potřebných objektů v obraze. Skoro každá úspěšná metoda používá jako základ Viola-Jones detektor, který byl publikován v roce 2001 [1].

První část této kapitoly se zaměřuje na poskytnutí základních informací a přehledu o tom, jaká řešení problematiky rozpoznávání obrazu (zejména dlaždic) pro robota už existují, případně jaké způsoby řízení a použití mobilního robota (model TurtleBot 2) už jsou implementovány.

Jedním z algoritmů, které se vyskytují v rozpoznávání obrazu je jeho detekce pomocí Haarových příznaků, která je součástí algoritmu definovaného P. Violou a M. Jonesem. Proto je rychlý a spolehlivý v mnoha případech.

2.1 Detekce únavy řidiče pomocí Haarových příznaků

Nejznámější příklad použití Haarových příznaků je rozpoznávání únavy řidiče na silnicích na základě kamerového systému uvnitř auta [2].

Možnost použití Haarových příznaků nabízí už zmíněný Viola-Jones detektor. Nespornou výhodou je rychlost, spolehlivost výsledků a hlavně nezávislost na osvětlení a rozměrech detekovaného objektu, což zajistí téměř stejnou úspěšnost v libovolném časovém okamžiku ve dne i v noci. Z těchto důvodů je algoritmus často používán zejména pro detekci obličejů nebo objektů, u kterých je předpokládán buď jejich pohyb, nebo pohyb zaznamenávajícího zařízení vůči objektům nebo vůči reálnému světu.

Stále aktuálním problémem je únava řidiče. Automobilky se snaží situaci zlepšit výrobou takzvaných chytrých automobilů, které podle chování řidiče odhadnou stupeň jejich únavy a případně bezpečně ukončí jízdu. Jiným řešením situace je snaha s pomocí počítačového vidění vyvinout umělou inteligenci, která dokáže řídit auto zcela sama, bez pomoci člověka. Všechna tato řešení jsou sice funkční, ale použitelná a populární v reálném světě příliš nejsou. Předně ne každý si z finančních důvodů může dovolit koupit auto, které bude obsahovat tuto

funkcionalitu. Druhým důvodem je to, že i přes velmi vysokou pokročilost a spolehlivost umělých neuronových sítí stále existují dopravní situace, ve kterých by člověk reagoval lépe. Právě proto autoři článku vymysleli řešení, které je popsané níže.

2.1.1 Návrh a implementace systému

Nejlepší cestou pro detekci únavy jednotlivých řidičů v reálném čase je udělat kvalitní rozpoznávání nejen obličeje, ale i očí, podle kterých se dají lépe poznat stupně únavy.

Výsledný systém by se skládal ze dvou částí: hardwarové a softwarové. Hardwarová část má za úkol pořídit snímky řidiče a předat je ke zpracovávání softwaru, který by měl pomocí různých algoritmů počítačového vidění a jejich implementace pro sledování očí rychle vyhodnotit, do jaké míry je řidič unavený [2].

Jako základ pro rozpoznávání obrazu byl použit Viola-Jones detektor. Tento algoritmus se skládá ze tří jednoduchých částí: integrálního obrazu, Haarových příznaků a klasifikačního algoritmu. Základní klasifikační algoritmus používaný P. Violou a M. Jonesem je AdaBoost, ale v současné době lze často vidět jeho nahrazení algoritmem GentleBoost, případně jinými variantami klasifikačních algoritmů [3].

Algoritmus detekce obličeje hledá specifické Haarovy příznaky na vstupním zobrazení (snímků). Ve chvíli, kdy je nalezen alespoň jeden z těchto příznaků, rozhodování postupuje na další úroveň. Jelikož hlavní požadavek na systém je rychlost, není možné nechat zpracovávání na aktuální úrovni, proto je potřeba pro urychlení rozpoznávání a klasifikace obrazu zmenšit nebo oříznout obraz tak, aby obsahoval informace o detekovaných příznacích. Nejpoužívanější rozměry pro část obrazu obsahující prvky Haaru jsou 24 × 24 pro kvalitnější výsledky a 8 × 8 pro rychlejší výpočty [4]. Následně ve finální fázi algoritmus porovnává hodnoty, které získal z předem definovaných Haarových příznaků, s právě dosaženými hodnotami.

2.1.2 Sledování pohybu očí

Někdy je možné poznat únavu řidiče pouze podle jeho obličeje, ale jak už bylo řečeno, je lepší používat detekci očí. Pro přesnější sledování toho, jak se mění pozice očí, byl navíc použitý korelační algoritmus zachycení cíle [2]. Detekce očí byla zlepšena pomocí osvětlení obličeje infračerveným světlem (IR). Toto světlo není pro člověka viditelné, ale kamerou je dobře zachytitelné. Cílem je zlepšit kvalitu v případech, kdy řidič používá sluneční brýle nebo v noci, kdy není možné přesně detekovat oči kvůli tmě.

2.1.3 Výsledky

Rozpoznávání obrazu pomocí Haarových příznaků je rychlá, kvalitní a velice spolehlivá cesta detekce potřebných objektů a to je také důvod, proč byla pro tuto bakalářskou práci zvolena právě tato metoda. Autorům článku se podařilo dosáhnout více než 90% přesnosti za normálních podmínek a více než 80% přesnosti v noci a v případech, kdy měl řidič na očích sluneční brýle. Pro základní detekci a rozhodování by však stačil samotný Haarův klasifikátor. Sledováním očí řidiče pomocí kamery můžeme včas rozpoznat příznaky únavy, abychom se vyhnuli nehodě. Průměrná doba odezvy mezi zpracováním obrazu a upozorněním řidiče na únavu pomocí signálů byla zhruba 50 ms [2].

2.2 Klasifikace dlaždic pomocí barevného modelu CIELAB

V článku je řešena problematika rozpoznávání obrazu pro účel klasifikace dlaždic pomocí barevného schématu CIELAB. Jsou zde představeny metody, které mohou být široce využívány pro kontrolu povrchu a klasifikace odstínu různých druhů dlaždic, jak je již poznamenáno v článku [5, s 1]. („*Stín, který se objeví na povrchu dlaždice, je kombinace řady vizuálních charakteristik tohoto povrchu, včetně jeho barvy a distribučního vzoru této barvy nebo dekorace na povrchu dlaždice.*“)

Výroba dlaždic není jednoduchý proces a v momentech, kdy je potřeba vyrábět velké množství podobných, nebo dokonce stejných kusů, dochází k problémům, kdy je potřeba řešit např. odstín barvy a stav výrobku. Při použití přírodních

materiálů a výrobě za vysokých teplot však většinou není možné zajistit správný odstín a podobnost jednotlivých dílů [5]. Pro tento problém článek představuje řešení – systém, který může rozpoznat a třídit dlaždice podle všech parametrů.

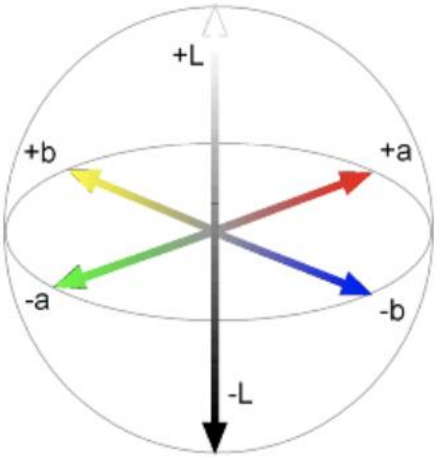
Na začátku jsou stanoveny 4 body, které by měly splnit úspěšně vyvinutý a testovaný systém. Jestliže lze třídění odstínů považovat za úspěšné, jednotlivá měření musí splňovat tato kritéria:

- „*změřit změnu barvy nebo odstínu, který je jasně viditelný lidskému oku*
- *poskytnout spolehlivé a opakovatelné hodnoty měření*
- *změřit změny odstínu, ke kterým dochází nebo by mohlo dojít při výrobě dlaždice*
- *být lineární s lidským vnímáním – rozdíl v jedné jednotce v části měřicí stupnice by se měl řešit stejným způsobem jako stejně velký rozdíl jedné jednotky v jiné části měřicí stupnice*“ [5, s. 2].

Pro takový systém bylo třeba používat spolehlivý a detailní barevný model, který může poskytnout co možná nejsprávnější data. Proto je jako řešení vybrán dobře zdokumentovaný a srozumitelný systém měření barev CIELAB, který je zároveň definovaný mezinárodními standardy.

2.2.1 Barevný model CIELAB

CIELAB je barevný model, který reprezentuje barvy nejblíže lidskému systému vnímání barev. Používá se v barevném tisku. Na rozdíl od jiných barevných modelů zde barva není popsána prvky reprodukovanými zařízeními, ale na základě tří složek barevného vidění osoby. V tomto modelu je libovolná barva určena svítivostí (*L*-Lightness) a dvěma chromatickými komponentami: kanál *a* obsahuje barvy od zelené přes šedou až do červené, kanál *b* obsahuje barvy od modré přes šedou až do žluté. Kanály *a*, *b* se pohybují v rozmezí od −128 do 127 a parametr *L* od 0 do 100 [6][7]. Nulová hodnota barevných komponent při jasu 50 odpovídá šedé v modelu RGB (119, 119, 119). Pokud je hodnota jasu 100, černou barvu reprezentuje 0.



Obrázek 1: Model CIELAB

Zdroj: [8]

V CIELAB je parametr jasu *L* zcela oddělen od obrazu, takže v případech, kdy je potřeba změnit barvu nebo zvýšit sytost obrazu, je vhodné použít tento model a ovlivňovat pouze barevné komponenty *a*, *b*.

2.2.2 Algoritmus segmentace clusterů referenčního snímku s využitím K-Means (Cluster Segmentation of the Reference Image Using K-Means)

Jako základní algoritmus byl pro systém zvolený způsob segmentace clusterů referenčního snímku pomocí K-means.

Clustering je způsob, jak oddělit skupiny objektů. Jedná se o široce používaný přístup pro nalezení a třídění skupin pozorování (ve výsledku nazývaných cluster), které mají podobné nebo stejné vlastnosti. Klastrování K-Means pak znamená, že každý objekt bude mít svoje vlastní místo v prostoru. Najde příslušné clustery tak, aby objekty uvnitř každého clusteru byly co nejblíže k sobě a pokud možno co nejdále od objektů v ostatních clusterech.

Klastrování K-Means vyžaduje, aby byly rozděleny řady klastrů a metrická vzdálenost, aby bylo možné vyčíslit, jak si jsou dva objekty navzájem blízké. Pro navrhovaný algoritmus byl použit K-Means k seskupení objektů do *n* klastrů podle

euklidovské vzdálenosti, kde *n* je počet základních barev v každém typu dlaždice [5].

Při použití výše uvedené metody může uživatel specifikovat počet základních barev, které bude potřebovat pro danou dlaždici bez ohledu na svítivost (luminosity). Tento krok byl navržen tak, aby se zároveň řešila změna osvětlení i ostatní věci, jako např. skvrny, které jsou na povrchu dlaždic. Tento algoritmus se skládá z jednoduchého postupu přehodnocení (re-estimation). Nejprve jsou datové body náhodně přiřazeny do jednotlivých odpovídajících sad K. Poté se pro každou sadu vypočítá její centrum. Tyto dva kroky se střídají, dokud není splněno kritérium zastavení, tedy dokud nedojde k žádné další změně v přiřazení datových bodů [5, s 3]. Protože algoritmus zvolený pro práci používá spíše diskrétní přiřazení než soubor souvislých parametrů, „minimum", jehož bude dosaženo, ve skutečnosti ani nemůžeme správně nazývat lokálním minimem. I přes tato omezení se algoritmus z důvodu snadné implementace používá poměrně často [5].

2.2.3 Výsledky testování

Pro testování algoritmu autoři vybrali 50 jednotlivých kusů dlaždic deseti různých typů a zaznamenáno 50 snímků. Všechny dlaždice byly samozřejmě předem klasifikovány podle klasických metod třídění a odpovídajícím způsobem označeny. Po zpracování obrazu navrženém algoritmickým postupem byly tyto snímky klasifikovány podle kritéria průměrné svítivosti. Odchylky v klasifikaci od rozhodnutí odborníka nastaly, když byla zvážena klasifikační metrika průměrné svítivosti klastru 1 a klastru 2. Pokud je však průměrná svítivost obou skupin považována za klasifikační metriku, existuje úplná shoda mezi algoritmickými výsledky a rozhodnutím odborníka [5].

2.3 Vizuální odometrie a navigace mobilního robota

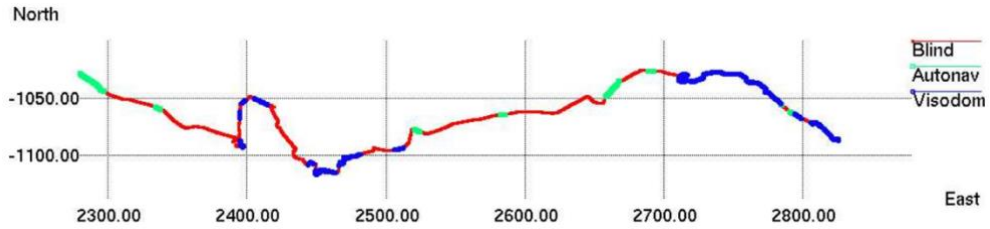
V navrhované aplikaci je základním problémem určení stávající polohy, sledování a následně zobrazení trajektorie pohybu mobilního robota. Nejjednodušší řešení představuje použití jednoho z moderních GPS modulů, který poskytne souřadnice pozice robota a také jeho další pohyb. Takové řešení by ale mělo pár nevýhod –

jedna z nich je, že ani nejnovější a nejdokonalejší GPS nejsou schopné určit správnou pozici na 100 %, a když se bude robot pohybovat uvnitř jedné budovy, není možné zajistit spolehlivost dat.

Naštěstí v dnešní době není GPS jediným řešením pro určení pozice, i když patří mezi nejpoužívanější. V článku [9] se autor snaží vysvětlit a vhodným způsobem popsat moderní algoritmy pro sledování pohybu mobilních robotů, jejich mapování, navigace atd.

2.3.1 Vizuální odometrie

Vzhledem k tomu, že je dostupná výhoda určení startovní pozice umístění mobilního robota pro začátek jeho pohybu, autor, který se setkal se stejným problémem [9] – mapováním pohybu uvnitř budovy a v jejím okolí – doporučuje toto řešit pomocí vizuální odometrie. Vizuální odometrie je proces, který se snaží odhadnout přibližnou vzdálenost odpovídající pohybu robota. Odhad se určí na základě posunu obrazu mezi jednotlivými snímky, které by měly reprezentovat pohyb mobilního robota. Zjištění pozice uvnitř místnosti nebo budovy není jedinou možností použití tohoto algoritmu. Vizuální odometrie byla použita pro řízení robota na Marsu již v roce 2003 [10]. Samozřejmostí je, že pro takový účel byl základní algoritmus vylepšen pro přesnější výsledky, pro poskytnutí obrazu byla použita všesměrová kamera. Výsledky vizuální odometrie jsou graficky vyjádřeny grafem, aby bylo možné vidět, jaký úsek robot překonal. Každý marťanský den je tento graf uložen a dále publikován pro přístup, proto kdyby nastal problém konfigurace či problém v algoritmu, je možné je odvodit, opravit a následně poslat novou konfiguraci pro práce.

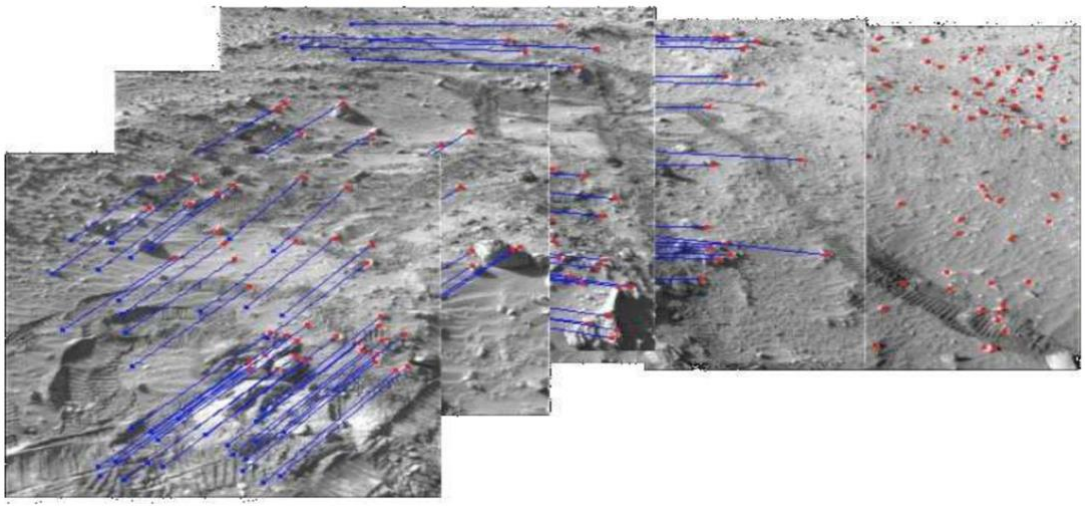


Obrázek 2: Vizuální odometrie, mapování trasy na Marsu

Zdroj: [10]

2.3.2 Nalezení významných bodů

Aby vizuální odometrie mohla fungovat správně, je potřeba sledovat pohyb robota mezi jednotlivými snímky. Určit změnu na dvou po sobě jdoucích snímcích není tak jednoduché, proto je potřeba nalézt v obraze body (objekt), které nebudou mít možnost měnit svoji pozice v průběhu pohybu robota. Následně je možné pomocí významných bodů a všesměrové kamery spočítat, na jakou vzdálenost se robot přiblížil/oddálil vůči těmto objektům v prostoru. Detekce zároveň slouží pro robota i jako navigace, aby mohl určit zóny nebezpečné pro průchod a vyhnul se jim. Na obrázku 3 je znázorněná detekce bodů na povrchu Marsu, které slouží pro navigaci mobilního robota.



Obrázek 3: Detekce významných bodů na povrchu Marsu

Zdroj: [10]

2.3.3 Výsledky

Před 20 lety by si nikdo nedokázal představit potenciál počítačového vidění a algoritmizace robota, ale dneska se mnoho věcí a zejména vizuální odometrie a detekce významných bodů používá k nejrůznějším účelům, a to počínaje jednoduchými robotickými vysavači pro domácnost a až po miliardové projekty, jako je např. průzkum povrchu jiných planet.

Vizuální odometrie je vysoce účinný nástroj pro řízení a udržení bezpečnosti mobilních robotů na neznámých površích. Ačkoliv by bylo lepším řešením aktivní nasměrování lidskými řidiči, pomocí výše uvedených algoritmů, díky menší výpočetní náročnosti je realizovaná většina zařízení určená pro planetární operace.

2.4 Sestavení robotické ruky na platformě TurtleBot

Článek [11] ukazuje, že dobře fungující systém postavený na platformě TurtleBot nemusí být příliš drahý, aby umožnil spravovat více věcí. Robotická ruka je vyrobena z běžně používaných komerčních dílů a je řízená pomocí základní desky Arduino. Pro rozpoznávání obrazu byl použitý senzor Microsoft Kinect. Dva uzly systému ROS (Robot Operating System) spravovaly všechny prvky, ovládaly mobilní základní desku a definovaly cíle, které bylo potřeba uchopit.

Obrázek 4: a) Originální TurtleBot b) Prototyp vytvořený pro daný projekt

Zdroj: [11]

V poslední době se stále více používají různé typy servisních robotů, jejichž cílem je usnadnit člověku život. V článku je ukázáno, jak se dá vytvořit robot, který by byl užitečný např. jako pomocník při uklízení. Základní myšlenka je přidat do robota mechanickou ruku, s jejíž pomocí může uklízet lehké věci. Důležité je umožnit robotovi rozpoznat objekty na jeho trase, a pokud je to možné, odstranit je a pokračovat ve vysávání [11]. Samozřejmé je, že všechno by mělo fungovat za předpokladu využití levných a komerčně dostupných komponent.

2.4.1 Komponenty

Jak už bylo řečeno, výsledek by měl být co nejvíce užitečný a spolehlivý, ale zároveň by měl obsahovat jenom komerční komponenty.

2.4.1.1 Spoje

Pro spojování mechanické ruky byly vybrány servomotory, jelikož umožňují umísťovat jednotlivé spojové části pod úhlem [12], který lze udržovat delší dobu, díky integrovanému zpětnovazebnému ovladači [11].

2.4.1.2 Vazby

„Pro vytvoření vazeb mechanické ruky se prozkoumaly dvě možnosti, které by mohly využít servomotory dostupné pro projekt: zakázkové díly ohýbané a vyříznuté z hliníkových plechů a hotové sestavy standardních dílů“ [11, s. 3].

Jako jedna z možných variant, která ve výsledku byla použita, se v článku zkoumá možnost použití standardních sad, které jsou kompatibilní se servomotory. Tento způsob má výhodu v tom, že není potřeba vyrábět žádné speciální díly.

2.4.2 Kontrola pohybu

Aby se generovaly impulsy vyžadované servomotory pro pohyb, je nutné, aby robot obsahoval nějaký mikrokontroler, který bude tyto impulsy poskytovat. „*Díky předchozím zkušenostem a dostupnosti byl použit mikrokontroler Arduino Uno, který byl následně připojený pomocí USB konektorů*“ [11, s. 4].

Programování probíhalo využitím standardního prostředí Arduino prostřednictvím jazyka C [11].

2.4.3 Rozpoznávání obrazu a uchopení objektu

Detekce objektů je řešena kvůli problému, že na podlaze můžou zůstat malé předměty, které budou robotu bránit v provádění práce. Jako objekty, které je možno odstranit, jsou uvažovány pouze objekty snadné pro uchopení, tedy ty objekty, pro jejichž uchopení a odstranění nejsou potřeba přesně rozpoznané a vypočítané body na jejich povrchu.

Zcela hotové řešení rozpoznávání obrazu obsahuje knihovna „PCL library“ [11].

2.4.3.1 Cloud voxelization

Pro poskytnutí zobrazení sledované plochy byl robotu vybrán standardní senzor vidění Kinect, který po své práce vrátí „3D point clouds“ reprezentující detekované body, z nichž bude následně provedené vyhodnocení, jestli je detekovaný objekt vhodný pro uchopení a odstranění.

Hlavním důvodem použití tohoto algoritmu je urychlení výpočtu, jelikož množství pixelů navrácených z „3D point clouds“ bylo příliš velké s ohledem na přijatelný čas zpracování. Algoritmus diskretizuje prostor do malých svazků nebo boxů a všechny body uvnitř jednoho boxu jsou zastoupeny v cloudu pomocí pouze jednoho bodu, který reprezentuje box [13].

2.4.3.2 Detekce podlahy

Vzhledem k rozumnému předpokladu, že robot pracuje ve vnitřním plochém prostředí (např. pokoj), byla rovina podlahy robustně detekována a odstraněna z cloudu pomocí SACSegmentace [13]. V článku [11] je uvedeno: („*Implementovaný algoritmus se domnívá, že dominantní detekovaná rovina by mohla být stěna; v takovém případě tyto body odstraní a opakuje se, dokud nebude zpracována celá dostupná vodorovná rovina.*“).

2.4.3.3 Detekce objektů

Pro detekce jednotlivých objektů je zde stejně jako v předchozích případech implementován algoritmus založený na segmentaci clusterů.

2.4.3.4 Výběr cíle

Mezi všemi objekty, které byly nalezeny v předchozím kroku, musí být vybrán jeden pro uchopení. Vypočítá se vzdálenost od robota do jednotlivých detekovaných objektů, následně se seřadí od nejbližšího. Nejbližší objekt se označí jako cíl, pouze pokud je v dosahu mechanické ruky a zároveň je jeho velikost v příslušném intervalu.

2.4.3.5 Převod souřadnic

Jako poslední se přepočítají souřadnice objektu na souřadnice mechanické ruky. Proto robot potřebuje udělat pouze dvě věci. První je upravení pohledu podle vyhodnoceného testu na nejbližší objekt. V druhém kroku se, pokud je známá velikost robota a umístění potřebného objektu, použije mechanická ruka pro přesun problémového objektu a poté bude pokračovat v provedení práce.

2.4.4 Shrnutí

Článek [11] popisuje návrh, stavbu, integraci a testování malého a cenově dostupného robota užitečného v domácnosti. Je zřejmé, že TurtleBot je dobrá platforma pro stavbu a vývoj např. komerčních projektů. Za minimum financí i úsilí autoři dokázali vytvořit prototyp, který je schopen detekovat, uchopit a přesouvat malé předměty.

3 Analýza a návrh řešení

V předchozí kapitole byly popsané už existující a implementované aplikace, které jako hlavní funkce poskytují kvalitní a spolehlivé rozpoznávání obrazu. V následující kapitole bude popsána navržená aplikace pro rozpoznávání obrazu pro potřeby řízení mobilního robota. Níže jsou přiblíženy jednotlivé technologie a nástroje použité pro tuto aplikaci.

3.1 Python

Python je objektově orientovaný programovací jazyk, v dnešní době používaný a provozovaný na všech platformách: MacOS, Linux, Windows. Díky jeho snadné instalaci a pochopitelnosti je vhodný i pro úplné začátečníky. Je určen pro tvorbu rozsáhlejších aplikací, protože v porovnání s jinými jazyky je velmi rychlý [14]. Co se týká problematiky řízení mobilních robotů, kybernetiky a robotiky obecně, python je v těchto oblastech nejpoužívanější programovací jazyk.

V následujících odstavcích budou popsány knihovny, které byly využity pro usnadnění práce.

3.1.1 Cv2

Jedná se o implementaci knihovny od společnosti Intel, která se jmenuje „OpenCV“ (Open Computer Vision). Tato knihovna obsahuje algoritmy řešící počítačové vidění a rozpoznávání obrazu. Oficiálně OpenCV poskytuje pro jazyk Python dva typy rozhraní: cv a cv2. Rozdíl mezi nimi je v množství zahrnutých a sdílených funkcí pro použití v algoritmech. Novější verze (cv2) poskytuje mnohem více možností manipulace s obrazem než její předchůdce (cv).

3.1.2 Numpy

Aby byl počítač schopen zajistit rychlý přehled všech informací o aktuálně zpracovávaném obrázku, je nutné ho uchovávat v paměti jako matici NxM, kde N je počet řádků reprezentovaných pomocí pixelů (výška obrázku) a M je počet všech sloupců (šířka obrázku). V některých případech navíc v paměti potřebujeme

uchovávat, jaké barevné schéma je používáno. Jelikož Python neobsahuje základní strukturu, která by byla schopná pracovat s více prostorově rozměrnými daty, je nutné implementovat tuto knihovnu. Numpy obsahuje pro nás potřebnou strukturu.

Kromě toho je další výhodou této knihovny její snadná instalace.

3.1.3 Sys

Základní knihovna pro jazyk Python, která umožní zpracovávat parametr zadaný přes konzoli, se nazývá *sys*. V tomto případě potřebujeme jednu hodnotu – název souboru s navigací pro robota, která určuje, jakou vzdálenost by měl robot ujet. Funkce, která zpracovává vstupní parametr, je znázorněna na obrázku 5.

```
def init_navigation_file():
    file_nav_name = None
    try:
        # load the parameter from console
        # if parameter is not empty set value
        file_nav_name = sys.argv[1] if len(sys.argv[1]) > 0 else None
    except:
        # missing parameter from console/terminal
        file_nav_name = '../nav/nav.txt'
        print("You have to enter file name to terminal/console or will use default file!!!!")
    return file_nav_name
```

Obrázek 5: Inicializace parametru přes terminál

Zdroj: Vlastní zpracování

3.1.4 Math

Knihovna Math umožňuje využívat v naší aplikaci složité matematické algoritmy a poskytuje pro práci všechny potřebné konstanty, např. číslo π. V aplikaci jsou používané její metody pro spočítání absolutní hodnoty.

3.2 OpenCV

OpenCV je multiplatformní knihovna, která poskytuje algoritmy pro manipulaci s obrazem. Je zaměřena především na počítačové vidění, zpracování obrazu a detekce objektu v reálném čase [4].

Ze začátku bylo OpenCV myšleno pouze jako framework pro studium strojového učení, vývoje a optimalizace kódu. Dnes je však tato knihovna jednou z nejpoužívanějších pro strojové učení, počítačové vidění a rozpoznávání obrazu obecně. Aktuálně jsou k dispozici verze jak pro různé operační systémy (MacOS, Linux, Windows), tak i pro různé programovací jazyky. Mezi nejznámější a nejpoužívanější patří: C/C++, Python, Java, JavaScript, PHP a další [4] [15].

V dnešní době existuje opravdu hodně oblastí, kde se používá počítačové vidění. OpenCV je velice používaná v monitorovacích systémech (např. monitorování aut nebo pozemků) a používá se ke zpracování fotografií a videozáznamů na internetu (používají ji velké firmy: Google, Facebook, Apple atd.). Kdo rád tráví čas u počítačových her, mohl slyšet, že v této oblasti se používá také pro herní rozhraní. Aplikace jako Google Maps pro reprezentaci ulic a jiných objektů také používají některé funkce knihovny OpenCV. Málo lidí ví, jak moc se v dnešní době používají algoritmy pro rozpoznávání obrazu a počítačové vidění. Skoro všechny firmy, které mají velký zisk a vytváří velké množství kusů jednotlivých produktů, používají pro kontrolu nebo zlepšení výsledků s velkou pravděpodobností počítačové vidění.

Otevřená licence OpenCV je navržena takovým způsobem, aby pro běžného uživatele-vývojáře bylo možné vytvářet komerční projekt s využitím veškeré funkcionality knihovny [4]. A současně není nutné, aby byl projekt vystavený jako open-source nebo byl veřejně přístupný.

Knihovna OpenCV se skládá z několika modulů, které může každý používat pro své vlastní projekty:

- **Jádro**, které obsahuje základní strukturu, maticovou algebru, algoritmy pro práci s pamětí, algoritmy pro zpracování chyb, funkce pro práci s 2D grafikou a spoustu dalších velice používaných funkcí [15].
- **Modul pro zpracování obrazu**, který obsahuje funkce pro práci s obrázky (konverze, filtrování atd.), funkce analýzy obrazu (vyhledávání obrysů, histogramy atd.), algoritmy analýzy pohybu, sledovací objekty, algoritmy rozpoznávání objektů [4].
- **Strojové učení** – tento modul obsahuje funkce pro klasifikaci a globální nebo lokální analýzu dat.
- **Modul pro vytváření uživatelského rozhraní (GUI)**, který je odpovědný za vytváření a nastavení oken, zobrazení obrázků, zachycování videa ze souborů a kamer, čtení / zápis snímků [15].
- **Zastaralé funkce**, jako jsou prostorové vidění, hledání a popisování vlastností objektu v prostoru, popis textur.

3.3 Neuronové sítě

Abychom dokázali rozumět pojmu konvoluční neuronové sítě, které byly použity pro realizaci částí aplikace řešící rozpoznávání obrazu, nejprve potřebujeme vědět, co to vůbec je obyčejná umělá neuronová síť a proč je v dnešní době tak široce známá a používaná. Umělá neuronová síť je pojem, který definuje přesný matematický model nebo jeho softwarovou nebo hardwarovou implementaci, která je postavená na principu organizace, funkcionality a fungování biologických neuronových sítí, totiž sítí nervových buněk živého organismu [16]. Základní myšlenkou je napodobení lidského chování a činnosti lidského mozku. Jelikož v současné době máme k dispozici poměrně výkonné počítače, můžeme používat neuronové sítě pro zjednodušení vývojů systému nebo pro usnadnění kontroly výroby produktu. Jak je již řečeno ve [17]: „*Neuronové sítě se skládají z jednotlivých umělých neuronů stejně jako biologické neuronové sítě v mozku*.“

Poprvé vědci narazili na tento koncept při studiu procesů probíhajících v našem mozku a při pokusu je namodelovat. Následně až po vývoji algoritmu pro učení neuronové sítě, se modely, které byly výsledkem práce algoritmu, začaly používat v mnoha oblastech. Nejznámější je ale problematika prognózování výsledku určité operace rozpoznávání obrazu a v případech, kdy je potřeba řídit mechanizmy nebo složitější jednotky.

3.3.1 Porovnávání umělé neuronové sítě a neuronové sítě v lidském mozku

Vývoj umělých neuronových sítí je inspirován biologií a jejich funkčnost by měla být alespoň trochu shodná s funkčností lidského mozku. Přestože žijeme v době, kdy se všechny technologie opravdu rychle rozvíjejí, je pravda, že stále máme málo informací o tom, jak funguje lidské tělo a zejména jak složitá konstrukce je mozek. Proto se vývojáři umělých neuronových sítí nemohou inspirovat pouze biologickými informacemi o funkčnosti mozku, ale musí jít dál, než může dosáhnout dnešní progres, aby bylo možné vytvořit opravdu užitečnou a funkční neuronovou síť. V mnoha případech to vede k potřebě zamítnout reálné chování neuronu mozku. Ten se tak stává jenom metaforou, s jejíž pomocí můžeme zjednodušeně popsat, jak vypadají a fungují umělé neuronové sítě. Vznikají sítě, které není možné vytvořit v reálném lidském mozku, nebo tyto modely vyžadují příliš velké znalosti o anatomii a fungování mozku.

I když je zřejmé, že vztah s biologií je opravdu slabý a často bezvýznamný, umělé neuronové sítě se i nadále srovnávají s mozkem. Fungování umělých a mozkových neuronových sítí je opravdu podobné, a proto je těžké se této analogii vyhnout. Taková srovnání jsou bohužel neproduktivní a vytvářejí neodůvodněná očekávání ohledně fungování neuronových sítí, která ve výsledku nejsou možná.

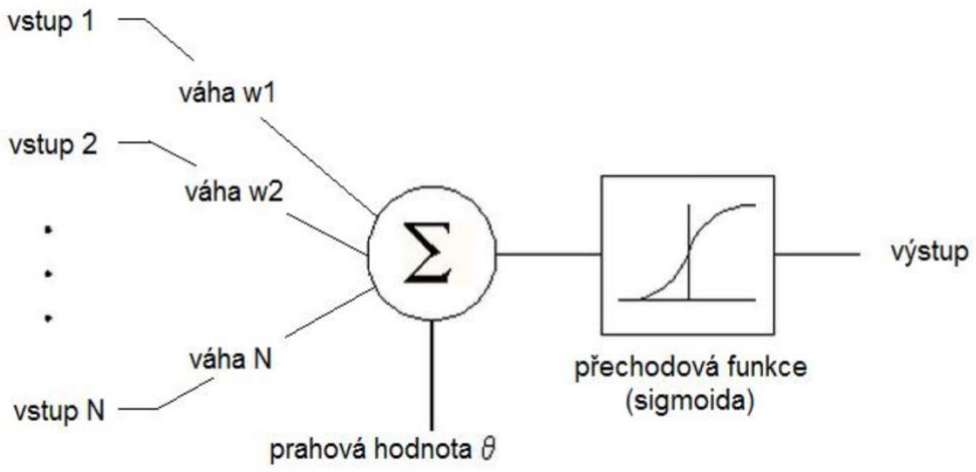
3.3.2 Definice práce a struktury neuronové sítě

Neuronová síť se skládá z mnoha neuronů (někdy také nazývaných perceptrony) a díky spojení mezi nimi je možné dosáhnout struktury modelu znázorňující komunikace mezi jednotlivými neurony jako v lidském mozku. Neuron tedy

můžeme chápat jako snahu pomocí matematických algoritmů popsat chování reálného biologického neuronu v mozku, protože neuron je v neuronové síti základní jednotkou zpracovávání informace a všechny jeho funkce jsou nahrazeny pomocí vhodných matematických algoritmů.

Na obrázku 6 je znázorněna struktura umělého neuronu v neuronových sítích.

Jak už bylo řečeno, každý umělý neuron by pro učení měl mít nějaký konečný počet vstupních dat. Nemusí to být data, která jsou pro neuronovou síť definovaná ručně člověkem, ale jako vstupní hodnota se může použít výstupní hodnota jiného neuronu v této síti. Je při tom možné popsat/definovat významnost jednotlivých vstupů pro určitou neuronovou síť. Hodnotu, která obsahuje tyto informace, nazýváme váha a obvykle se označuje pomocí písmena **w.** V biologickém neuronu prahová hodnota definuje hranice neboli nejvyšší povolené hodnoty dosahu, kde je možné pomocí přenosové (aktivační) funkce převést neuron z pasivního stavu.



Obrázek 6: Schematicky znázorněná struktura umělého neuronu

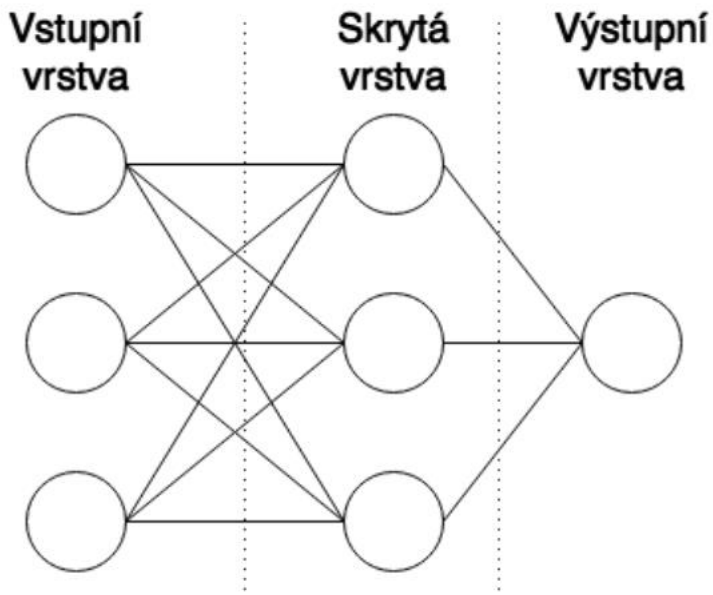
Zdroj: [18]

Jeden z hlavních úkolů, které jsou stanovené pro neuronové sítě a zároveň souvisí s touto bakalářskou prací, je rozpoznávání obrazu. Účel jejich existence v této problematice je klasifikace vstupujících obrazů, to znamená, že pro něj musí přiřadit správnou a už známou třídu obrazu pro tuto síť. Proto ze začátku, kdy probíhá učení, jsou pro síť definované obrazy, u nichž předem víme, kam spadají, resp. přesně známe jejich třídu obrazu. Pak jsou na vstup do sítě poskytovány

neznámé obrazy (obrazy již známé pro člověka, ale nedefinované pro neuronovou síť) a neuronová síť se pomocí určitého algoritmu snaží pro jednotlivé vstupy přiřadit odpovídající třídy.

3.4 Konvoluční neuronové sítě

Jednotlivé neurony neuronových sítí stejně jako neurony v biologických neuronových sítích v našem mozku jsou uspořádány do vrstev. První vrstva se nazývá vstupní, a jak již vyplývá z názvu, má za úkol sběr jednotlivých dat, ze kterých se pak bude provádět učení. Cílem výstupní vrstvy je reprezentovat co nejpřesnější výsledek vzhledem ke stanoveným cílům a počtu vstupních neuronů. Jednoduché neuronové sítě opravdu používají jenom dvě základní vrstvy pro poskytnutí základního modelu výsledku jejich práce. Konvoluční neuronové sítě pro přesnější a spolehlivější výsledek používají vrstvy, kterým se říká skryté. Skrytá vrstva neuronové sítě obvykle není vidět a používá se pro jednotlivé výpočty a pro rozhodování, který neuron obsahuje informace s větší předností. Počet skrytých vrstev není omezen, konvoluční (nebo také vícevrstvová síť) se může skládat jenom z jedné nebo z několika skrytých vrstev. Základní myšlenkou konvoluční neuronové sítě je to, že čím více skrytých vrstev model obsahuje, tím přesnější by měl poskytovat výsledky, ale je jich také potřeba uchovávat rozumné množství.



Obrázek 7: Vzhled jednoduché konvoluční neuronové sítě

Zdroj: [17]

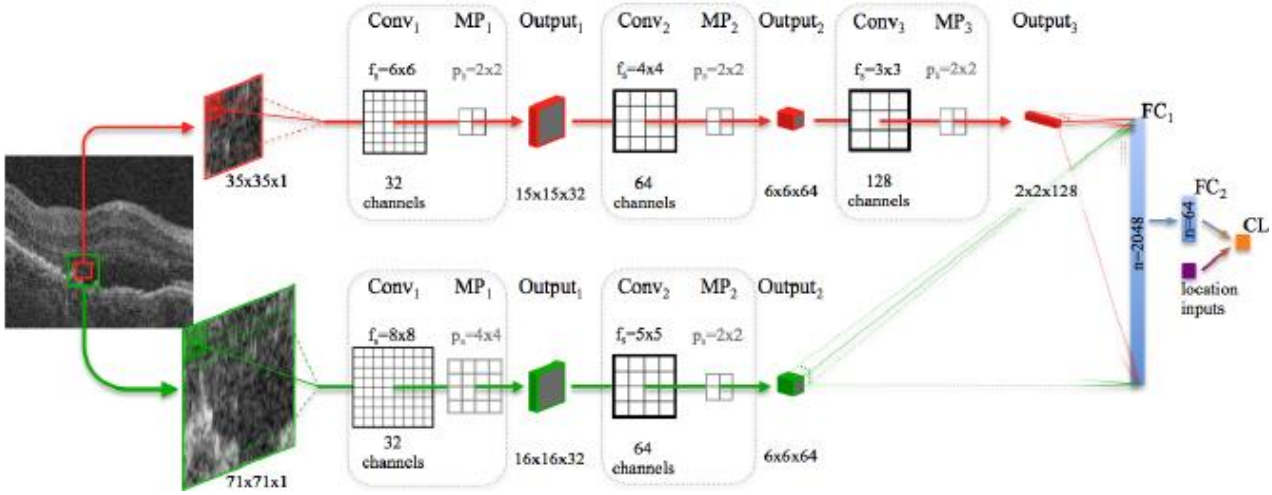
Dlouhou dobu nám stačily jednoduché neuronové sítě, které měly jenom vstupní a výstupní vrstvy, dokonce mohly poskytnout základní rozpoznávání obrazu. Dnes ale nikdo dvouvrstvé neuronové sítě nepoužívá, když se jedná o rozpoznávání obrazu. Hlavním důvodem je to, že tato síť má příliš jednoduchou strukturu, než aby mohla správně řešit problém detekce různých objektů v reálném čase. Člověk více než 70 % informací vnímá právě pomocí svých očí. Dvouvrstvé neuronové sítě se nějakou dobu snažily napodobit lidské chování, ale byla to jenom takzvaná první verze, obsahovala příliš jednoduchý matematický model. Jak obyčejné, tak i konvoluční sítě pro rozpoznávání obrazu potřebují předem zpracovávat obraz a ideálně s ním pracují jenom ve dvou barvách – černé a bílé. Konvoluční síť je kvůli většímu počtu vrstev a složitějšímu matematickému modelu pomalejší a náročnější pro zpracovávání než předchůdce. I přesto se však používá víc. Hlavním důvodem je, že i když obyčejná neuronová síť může říci, kde se nachází objekt v prostoru díky jeho porovnávání s vstupním obrazem, ve chvíli, kdy dojde k nějaké deformaci rozpoznávaného tělesa, šance jeho správného nalezení se blíží k nule. Konvoluční neuronové sítě zpracovávají informace pomocí více vrstev a díky tomu deformace ovlivní minimálně kvalitu rozpoznávání a detekce

potřebného objektu. Samozřejmě záleží na počtu a kvalitě vstupních dat pro učení, na počtu skrytých vrstev a mnoha jiných parametrů, které se pro tento typ sítě dají řešit. Konvoluční neuronové sítě byly navrženy hlavně proto, aby rozpoznávaly objekt nezávisle na jeho velikosti, poloze v prostoru a deformaci.

3.4.1 Vstupní vrstva

Vstupní data pro konvoluční neuronovou síť, která by měla poskytovat funkce rozpoznávání obrazu v reálném čase, je zpravidla obrázek nebo resp. jeho matematická reprezentace prostřednictvím matice pixelů. Rozměry matice jsou definované pomocí rozměrů původního obrázku, musí platit vztah: výška obrázku – šířka obrázku – hloubka, kde hloubka uchovává informace, kolik kanálů obsahuje obrázek [19].

Na obrázku 8 jsou znázorněná vstupní data, která obsahují jenom jeden kanál (jedná se o černobílé snímky).



Obrázek 8: Architektura konvoluční neuronové sítě

Zdroj: [20]

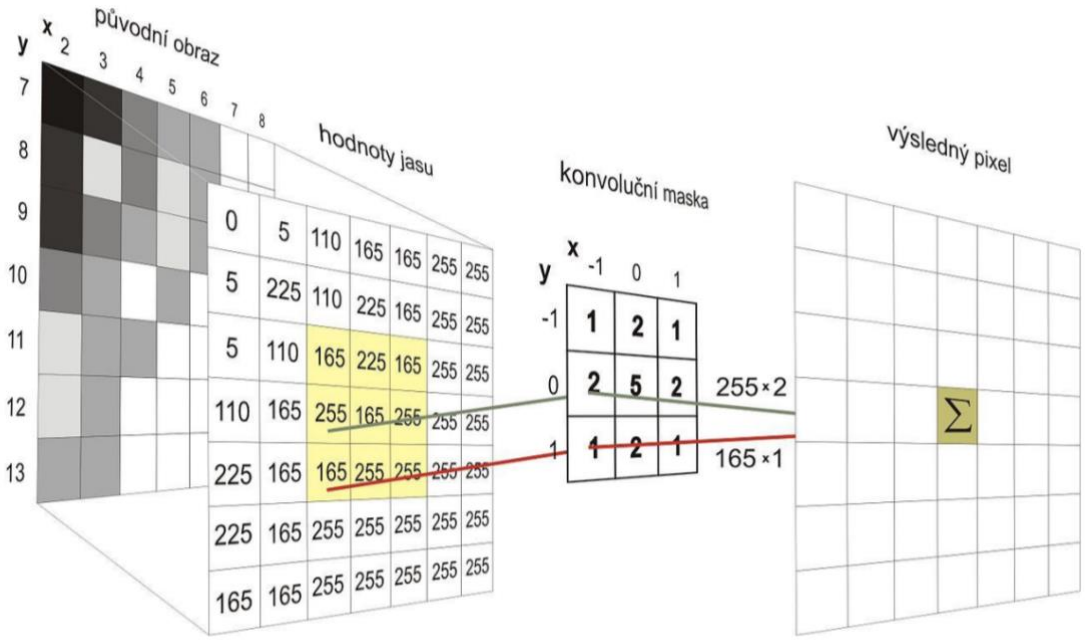
3.4.2 Konvoluční vrstva

Již podle názvu můžeme pochopit, že konvoluční vrstva je základem konvoluční neuronové sítě. Jejím účelem je získat lokální informace o vstupní matici obrazových bodů. Konvoluční vrstva obsahuje sadu konvolučních filtrů, které mají

hodně malý rozměr v porovnání s původním zobrazením [19]. Filtry většinou bývávají velikosti 3 × 3, 5 × 5 nebo minimálně používané 6 × 6 a 8 × 8 (viz obrázek 8), ale musí uchovávat stejný počet kanálů.

Poté je konvoluční filtr aplikován pro každý pixel (každý element matice) zobrazení. Právě tato operace nám zajistí spolehlivost rozpoznávání obrazu v případě deformace, změny polohy či velikosti zobrazení.

Každý element matice obsahuje nějakou hodnotu, která popisuje definovanou barvu pro tento pixel. Většinou se používá známý RGB (Red Green Blue) model. Proces aplikace konvolučních filtrů je také znám jako proces konvoluce, který postupně popisuje obrázek 9. Pomocí definovaného konvolučního filtru se postupně prochází celá matice zobrazení a násobí se jednotlivé body a body konvolučního filtru. Ve výsledku musíme všechny hodnoty sečíst, aby nám zůstalo pouze jedno jediné číslo. Použitím těchto určitých filtrů docílíme zmenšení obrázku v každém kroku počítání hodnoty konvoluční vrstvy [21][22].



Obrázek 9: Aplikace konvolučního filtru, Konvoluce

Zdroj: [23]

Základní posun konvolučního filtru pro násobení je jeden pixel, ale počet kroků je také možné nastavit předem.

3.4.3 Pooling vrstva

Ještě jedna vrstva, která se objevila v konvolučních neuronových sítích, je pooling vrstva, která by měla zajistit nelinearitu jednotlivých procesů. Základním principem fungování této vrstvy je vzorkování obrazu a zmenšení velikosti dat, která neuronová síť potřebuje zpracovávat [21].

Pooling vrstva také poskytuje ochranu sítě proti přetrénovanosti. Nejvíce používaná metoda pro redukci dat je max-polling, tedy hledání maximálních hodnot. Při použití konvolučního filtru 2 × 2 a nastavení kroku násobení tohoto filtru a matice zobrazení na 2 je možné zmenšit objem zpracovávaných dat přibližně o 75 % [23].

3.4.4 Plně propojená vrstva

Obecně tato vrstva definuje, že všechny po sobě jdoucí vrstvy pro lepší výsledek by měly být navzájem propojené. Každý neuron, který se nachází na vyšší vrstvě, musí být správně propojený s každým neuronem, který se nachází na nižší vrstvě, a jelikož se jedná o propojení plné (fully-connected), toto tvrzení by mělo platit i opačně [24].

V konvolučních neuronových sítích se používá více proto, aby se neuronová síť mohla naučit nejen detekovat jednotlivé objekty (nebo jejich příznaky), ale i jejich různé lineární a nelineární kombinace.

3.4.5 Výstupní vrstva

Jak je vidět již z názvu, toto je poslední vrstva v neuronové síti. Také zde platí pravidlo, že výstupní vrstva by měla být propojena s vrstvou předchozí. Obyčejné neuronové sítě mohou mít libovolný konečný počet vstupů, ale pouze jeden výstupní neuron. Pro konvoluční neuronové sítě toto omezení neplatí. Konvoluční neuronová síť může obsahovat tolik výstupních neuronů, kolik je potřeba pro

klasifikační třídy [21]. Pokud je úkolem klasifikovat je do více než dvou tříd, jako příslušná aktivační funkce je vybraná funkce softmax [23]. Funkce softmax je zobecnění logické regrese pro jednotlivé třídy. Výstupem je pravděpodobnost v rozmezí 0–1, která znázorňuje příslušnost detekovaných objektů do jednotlivých klasifikačních tříd.

3.5 Použití konvolučních neuronových sítí

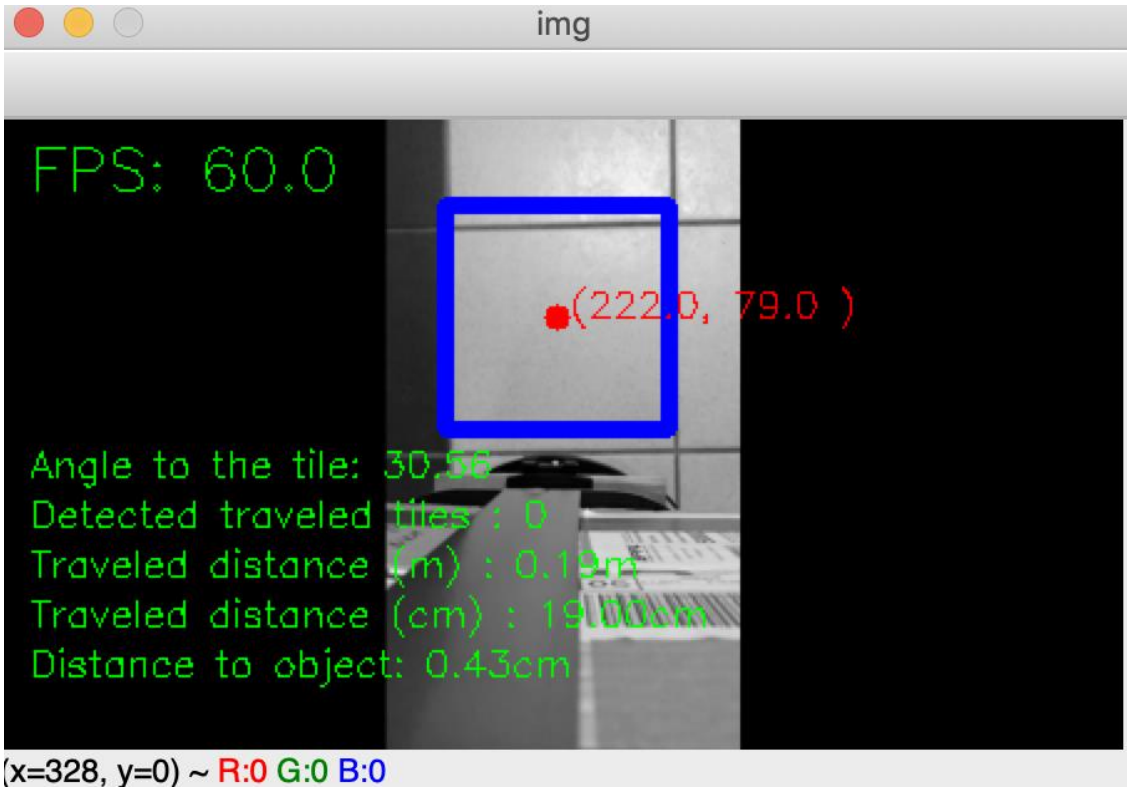
Obecně můžeme problém rozpoznávání obrazu pomocí konvoluční neuronové sítě rozdělit do dvou po sobě jdoucích částí. První je potřeba učení neuronové sítě na už existující a reálná data. Druhou částí je pomocí výsledného modelu samotné rozpoznávání a klasifikace detekovaných objektů na obrazu.

Učení se provádí zobrazením objektů pro neuronovou síť, jejichž klasifikace už známe předem. Pro tuto bakalářskou práci nebylo potřeba detekovat více různých druhů objektu v prostoru. Hlavním cílem byla detekce dlaždice na povrchu před mobilním robotem, proto měla neuronová síť klasifikovat objekty jenom do jedné třídy – dlaždice, přičemž barva jednotlivých dlaždic pro nás nehrála roli. Výsledný model by měl umět správně rozpoznat část obrazu, kde se nachází potřebný objekt, a také v případě klasifikace do více tříd by měl nalezené těleso přiřadit do správné třídy.

Než budeme hovořit o správnosti rozpoznávání obrazu, potřebujeme o objektu zjistit všechny charakteristiky a informace. Ale každý objekt, i když patří do stejné třídy, může mít rozdílné vlastnosti. Konvoluční neuronová síť díky svým skrytým vrstvám pro učení může použít nejen definovaná syrová data, ale i kombinace těchto dat.

Výsledná konvoluční neuronová síť pro tuto práci na vstupu používala jenom 64 obrázků, na kterých byly vidět dlaždice. Každý obrázek obsahoval jeden až šest objektů, jejichž informace byla poskytnuta pro neuronovou síť. Po konfiguraci obsahovala tato síť 24 skrytých vrstev a poskytovala spolehlivost detekce dlaždic přibližně 85 %. Finální aplikace však kvůli poměrně špatným výsledkům při

počítaní ujeté vzdáleností používá feature detekce a vizuální odometrie popsané dále, které poskytují lepší výsledky.



Obrázek 10: Detekce dlaždice pomocí konvoluční neuronové sítě

Zdroj: Vlastní zpracování

3.6 Rozpoznávání obrazu

Nejprve by bylo potřeba popsat celou oblast informatiky, která se věnuje rozpoznávání obrazu, protože v dnešní době je tento název příliš obecný a definuje mnoho případů, kde se používají algoritmy pro simulace reálného vidění člověka.

Rozpoznávání obrazu (jedná se o objekty v reálném prostoru, signály, jevy nebo určité procesy) jako cíl uvažuje přesnou identifikaci objektu nebo určení jeho vlastností podle vstupního obrazu nebo jiných vstupních dat v případě rozpoznávání signálu. Tato práce se však zabývá jenom rozpoznáváním předem definovaných a známých předmětů z dat, která jsou zaznamenávaná v reálném čase pomocí kamery, případně jiného zařízení.

3.6.1 Počítačové vidění

Základní myšlenkou problematiky počítačového vidění je vytváření umělých strojů, které by mohly vnímat svět a okolí jako kterýkoliv člověk. V dnešní době máme k dispozici počítače, které mají velký výpočetní potenciál pro tuto oblast a umožňují řešit úkoly detekce určitých objektů v reálném čase, které dříve byly považovány za nemožné. Přesto ještě není možné vyřešit kompletní rozpoznávání obrazu a vyvinout metodu, která bude obsahovat možnosti vidění a vnímání informace na úrovni vidění člověka.

3.6.2 Viola-Jones detekce

Je pravda, že výpočetně nejnáročnější úkoly jsou v současnosti zpracovávání velkého množství dat v reálném čase. Nejznámější a problematickou úlohou této oblasti je rozpoznávání obrazu a detekce objektu. Obvyklým příkladem je rozpoznávání obličeje nebo aut na silnici. Ale toto není všechno, kde se dá aplikovat daná oblast. Kvůli velké výpočetní náročnosti není možné spolehlivě a rychle rozpoznávat všechno, co se nachází v obraze, ale je možné rozpoznávat jednotlivé potřebné objekty pro částečné řešení stanovených úloh. V dnešní době už existuje mnoho metod a algoritmů pro detekce objektů.

Metoda, která byla navržená Violou a Jonesem v roce 2001, je však považována za základní metodu v problematice rozpoznávání obrazu [1].

Tento způsob detekce objektu získal velkou popularitu pro rozpoznávání jednotlivých objektů, a to díky své vysoké přesnosti a spolehlivosti.

Viola-Jones detekce objektů stejně jako většina algoritmů pro rozpoznávání je založená na principu klasifikace objektů, které se nachází v obraze na předem definované třídy. Algoritmus se nesnaží hledat celý objekt, ale pouze porovnává příznaky nalezené z obrazu s příznaky tříd, o nichž obsahuje všechny informace. Poté přichází fáze rozhodování, jestli nalezené příznaky patří do nějaké třídy.

Většinou se systém pro rozpoznávání obrazu stavěný na Viola-Jones klasifikaci skládá ze 4 částí. První je definice objektu (resp. jejich příznaků) pomocí

matematického modelu, který jsou potřeba rozpoznávat. Druhý krok je nalezení příznaků v obraze, následně jejich rozpoznávání a jako poslední rozhodování, zda patří do definované problematiky.

Je pravda, že učení neuronové sítě dle tohoto principu je pomalé i při použití moderních a výkonných počítačů, ale výsledky, které model poskytuje v reálném čase, jsou velmi rychlé a spolehlivé. Tato metoda nepatří do moderních metod rozpoznávání obrazu a v porovnání s progresem, který má oblast IT celkově, a zvlášť oblast rozpoznávání obrazu, je relativně stará. Díky rychlosti zpracovávání obrazu v reálném čase se stále používá pro detekci jednotlivých objektů a většina moderních přístupů je postavená přímo na principu Viola-Jones detekce.

Je potřeba vědět, že tato detekce je určena nejprve pro práci s šedo-tónovými obrázky. Zpracovávat vstupní obraz v RGB modelu a hledat barevné příznaky není totiž nejlepší cesta pro daný algoritmus, která může velice ovlivnit rychlost jeho práce.

3.6.3 Detekce objektu v obrazu pomocí Haarových příznaků (Haar-Like features)

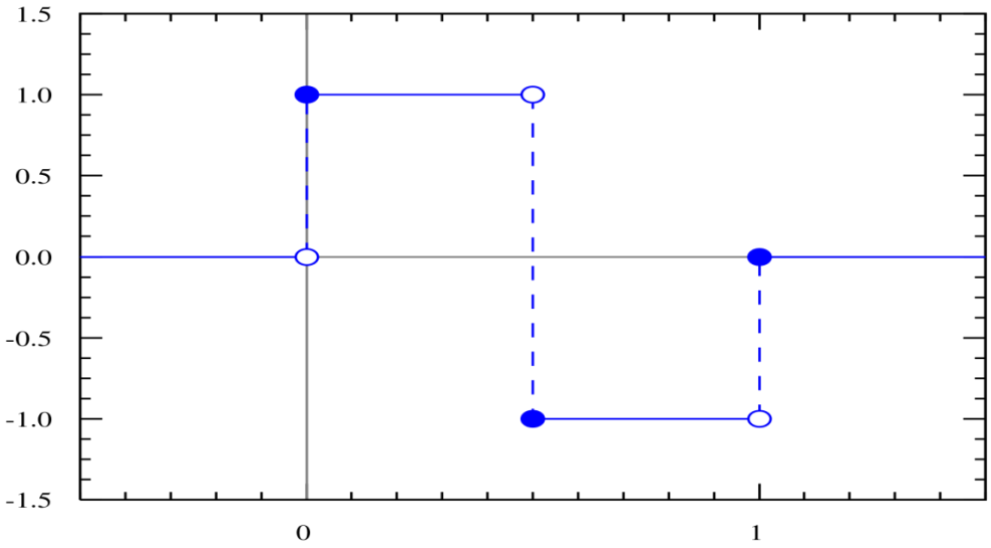
Vyvíjejí se stále nové metody a nové přístupy pro rozpoznávání obrazu. Klasifikace podle Haarova principu byla a stále je jednou z nejpoužívanějších metod pro rozpoznávání obrazu [25].

Většinou algoritmy, které pracují pouze s intenzitou barvy jednotlivých pixelů, např. algoritmy používající pro rozhodování hodnoty RGB, jsou výpočetně složitější, a proto pro jejich správnou a plynulou práci je potřeba mít odpovídající sestavení počítače, případně serveru.

Haarovy příznaky jsou příznaky digitálního snímku, které popisuji každý jednotlivý obrazový bod tohoto snímku. Znamená to, že algoritmus učení pro neuronovou síť nebude zpracovávat pixely celého vstupního zobrazení, ale bude pracovat s příznaky, které mají za úkol tyto pixely co nejvíce popsat. Díky této metodě je zajištěna menší výpočetní náročnost a poskytnuto rychlejší zpracovávání obrazu, a to hlavně v reálném čase. První neuronová síť, která

poskytovala rozpoznávání obličejů v reálném čase pro klasifikace, používala právě Haarovy příznaky.

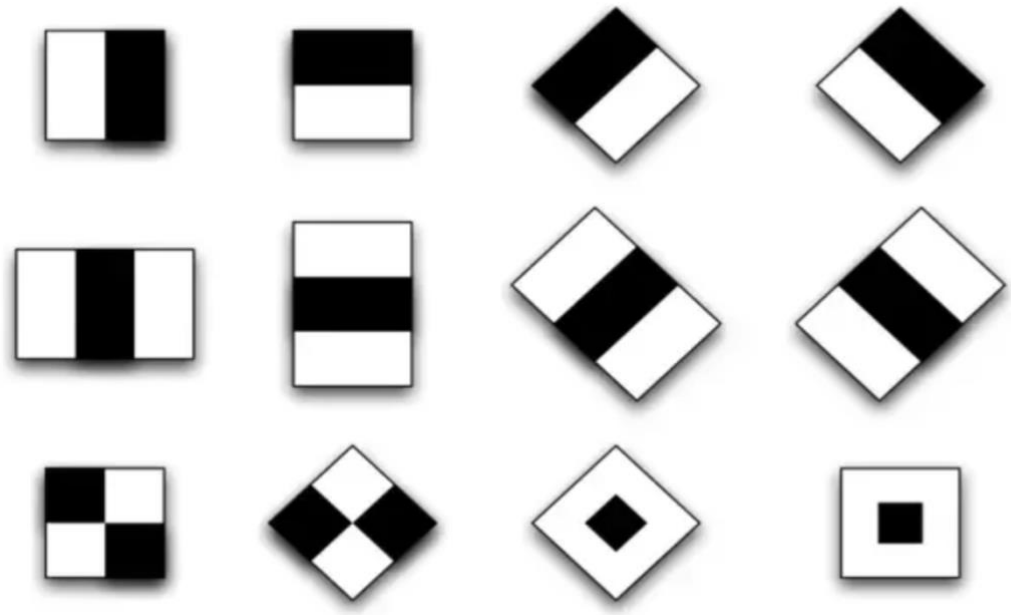
Pro lepší porozumění fungování Haarových příznaků potřebujeme definovat ještě jeden pojem – Haarova vlnka. Haarova vlnka je nejjednodušší a nejstarší typ vlnek, který neumožňuje hladkou rekonstrukci signálu (dle obrázku 11) [26]. Výhodou Haarových vlnek je velmi rychlý a správný výpočet, což nelze říct o spolehlivosti.



Obrázek 11: Haarova vlnka

Zdroj: [27]

Viola a Jones [28] přizpůsobili myšlenku použití Haarovy vlnky a jako výsledek své práce získali to, co pak bylo nazváno Haarovy příznaky [25]. Ve výsledku Haarovy příznaky představují binární aproximace Haarových vlnek. Každý prvek reprezentuje binární masku, nebo lépe řečeno je to příznak skládající se ze dvou barev – černé a bílé – a může se nacházet ve více stavech. Všechny možné stavy jsou prezentovány na obrázku 12.



Obrázek 12: Haarovy příznaky

Zdroj: [26]

Ve většině implementace algoritmů jsou Haarovy příznaky reprezentované pomocí množiny obdélníku. Haarovy příznaky se skládají ze sousedních obdélníkových oblastí. Viola a Jones vyvinuli algoritmus detekce objektu, který pro svoji práci používá Haarovy příznaky a má za úkol získat velkou řadu jednoduchých příznaků. Ty jsou umístěny na obrázku, poté je shrnuta intenzita pixelů v jednotlivých oblastech. Následně se musí spočítat suma hodnot obrazových bodů, které obsahují světlé části obrazu, a jako poslední se vypočítá rozdíl mezi touto hodnotou a hodnotou reprezentující sumu hodnot pixelů obsahujících tmavé části obrazů. Tento rozdíl pak můžeme považovat za hodnotu odpovídajícího příznaku a bude mít svoji vlastní velikost a specifické umístění na obrázku.

3.6.3.1 Zpracovávání vstupního obrazu

Vstupní obraz se zpracovává pomocí metody plovoucího okna (sliding window method). Použít tuto metodu znamená procházet celý vstupní obraz pomocí okna, jehož velikost je definovaná pomocí odpovídající velikosti vstupu detektoru s určitý předem známým krokem. Velikost kroku znamená, o kolik musí být posunuté okno v průběhu jeho průchodu obrazem. Většinou je definovaná pomocí

dvou hodnot *x* a *y*, které uvádí jednotlivé posuny na těchto osách. Pokaždé, když je okno posunuto, se část vstupního obrazu, která se v daný časový úsek nachází v okně, zpracovává pomocí detektoru. V případě pozitivní reakce detektoru se předpokládá, že byl rozpoznán požadovaný objekt. Vzhledem k tomu, že objekty, které je potřeba detekovat, mohou mít různou velikost, je obraz několikrát zvětšen a zmenšen a v každé této fázi se zpracovává pro lepší detekci [29].

Vstupní obraz není zpracováván jako jedna velká matice hodnot najednou, ale v každém posunu okna se zpracovává jenom jeho určitá část. Nevýhodou však je velká náročnost pro paměť, jelikož musíme vytvářet a uchovávat informace o mnoha oknech v každém snímku. Ale ani tento problém není natolik kritický, jak se může zdát. V každém průchodu vstupním obrazem se zamítnou všechna okna, která neobsahovala pozitivní výsledek. Zároveň mezi silné stránky díky měnění velikosti patří přesnější výsledek klasifikace.

3.6.3.2 Klasifikace nalezených Haarových příznaků po zpracovávání vstupního obrazu

Opět se zde vychází ze základní myšlenky definované Violou a Jonesem, kteří se rozhodli pro klasifikaci nalezených příznaků použít AdaBoost.

AdaBoost je zkratka pro Adaptive Boosting. Boosting je proces, v němž se jedná o kombinace špatných klasifikátorů, které už byly nalezeny, do silného klasifikátoru. AdaBoost umožní nejen náhodné, ale i vhodné spojení několika klasifikátorů, které se považují za málo úspěšné, do jednoho, který by ve výsledku měl být úspěšnější než všechny méně spolehlivé dohromady.

3.7 Feature detector

Dalším úkolem byla potřeba sledovat a mapovat, zobrazovat ve vedlejším okně pohyb robota v prostoru budovy, kde se nachází. První problém byl vymyslet řešení monitorování pohybu robota bez použití novějších modulů a technologií sledování polohy objektu, jako je například GPS. Ve výsledku byla zvolena varianta hledat v obrazu, který je sledován pohybujícím se zařízením, (detekovat) významné body. Detekce významných bodů neboli významné oblasti je založena na

obrazu, který před touto operací musí být nutně segmentovaný na superpixely obrazu. Superpixel je určitá část obrazu, kde je barevná hodnota všech jednotlivých pixelů natolik podobná, že by se ji dalo pro zjednodušení vybarvit jednou barvou. Nejčastěji je pro vyplnění vybraná hodnota, která odpovídá průměrné hodnotě všech vybraných barevných kanálů – nová oblast je po vyplnění takzvaný homogenní superpixel. Významné body představují v obraze oblast, která odpovídá superpixelům a při pohybu v různých snímcích bude mít rozdílné souřadnice v prostoru, ale stejnou nebo hodně podobnou hodnotu barevných kanálů. Kvůli tomu, že pohyb nemusí vždy být plynulý a může vzniknout šum v obraze, barevná hodnota v následujícím snímku ne vždy nutně musí odpovídat její barevné hodnotě v předchozím. Proto se nehledá stejná hodnota, ale existuje nějaký chybový rozsah, s jehož pomocí lze zjistit, zda můžeme nalezený bod považovat za významný.

Existuje hodně algoritmů pro usnadnění procesu nalezení významných bodů v pohybujícím se prostoru. Hlavním cílem takových algoritmů je nalezení superpixelů tak, aby obsahovaly barevnou hodnotu, která se bude výrazně lišit od hodnoty jiných superpixelů ve stejný pozorovací okamžik ve stejném snímku. Zároveň se snaží tyto body popsat tak, aby při jejich dalším nalezení ve snímcích bylo možné identifikovat a správně klasifikovat tyto body bez ohledu na jejich transformace mezi jednotlivými snímky.

Podle [30] pro správnou detekci a práci s bodovými příznaky v sobě musí algoritmy hledání významných bodů zahrnovat následující kroky:

- Detekci příznaků.
- Deskripci příznaků.
- Porovnávání deskriptorů.

Úkolem detekce příznaků je nalezení oblastí obrazu a následně její třídění na vhodné superpixely. Ve chvíli, kdy je obraz rozdělen, přichází na řadu další fáze, a to nutnost popsat příznaky tak, aby je bylo možné rozpoznat v následujících snímcích při jejich opětovném nalezení. Porovnávání deskriptoru je určení, jestli

jsou v různých snímcích významné oblasti, které mají stejnou barevnou hodnotu, a to pomocí porovnání výsledků předchozího kroku.

Některé ze současných algoritmů pro nalezení významných oblastí obrazu jsou schopny vyřešit všechny tři problémy, ostatní, používané pro jiné, speciální účely nebo usnadnění výpočtu a menší náročnost, řeší jenom některé kroky, a proto je při jejich použití potřeba chybějící doplnit ručně.

3.7.1 RANSAC

Pro realizaci hledání význačných bodů byl v této bakalářské práci použitý robustní regresní algoritmus RANSAC, který je implementován v knihovně OpenCV a široce se využívá pro odhad prostorové transformace mezi více snímky, na kterých už byly nalezeny významné oblasti.

RANSAC je zkratka pro Random Sample Consensus. Jedná se o algoritmus, který funguje na základě jednoduché a známé metody nejmenších čtverců. Cílem je najít správný vztah dat z jednoho snímku vůči datům, která byla získána z jiných, předchozích, nebo naopak následujících snímků, kde data ze snímku jsou reprezentovaná pomocí množiny získaných bodů [31]. Vztah je považován za správný, pokud se nalezená množina dat vyskytuje ve více snímcích. Matematicky se jedná o řešení předurčené soustavy rovnic.

3.7.2 Princip fungování RANSAC

O tom, jak funguje zmíněný algoritmus, bylo již krátce pojednáno v předchozí kapitole.

Základní myšlenkou tohoto algoritmu je, že všechna vstupní data jsou složena z inliers a outliers. Inliers jsou body, které mohou být vyloženy nějakou množinou parametrů modelu. Outliers jsou data, která se s modelem neshodují. Nejčastější důvod vzniku outliers jsou extrémní hodnoty šumu, nesprávné měření, nestabilní pohyb obrazu, velké natočení kamery (např. tak, že v následujícím kroku zmizí předchozí body, které byly brány jako inliers). RANSAC byl vytvořen s předpokladem, že toto může nastat a algoritmus je schopen tento „problém“

vyřešit, jestliže máme k dispozici nenulovou množinu nalezených a označených bodů jako inliers, pak je uvnitř funkce odhadující parametry modelu, který podle popisu dat dokáže posoudit, zda spadají do modelu a zda mohou být označeny jako inliers nebo outliers.

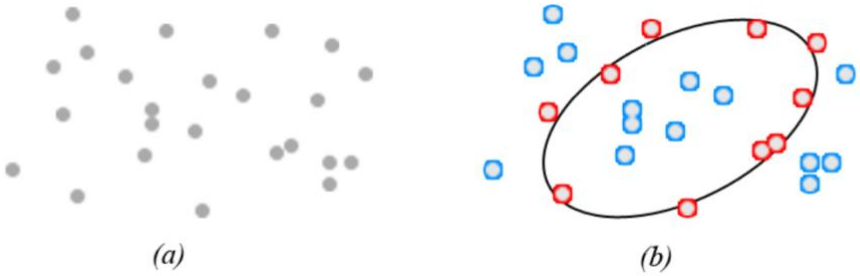
První krok algoritmu je náhodný výběr bodu pro testování. Ze začátku jsou všechny vybrané body označeny jako inliers a v následujících krocích jsou testovány, jestli jsou opravdu shodné s modelem. Ten je porovnáván s hypotetickou množinou bodů, která byla označena jako inliers, což znamená, že všechny volné parametry modelu jsou rekonstruovány z množiny bodu [32]. Je možné, že po určení a testování náhodných bodů se může vyskytnout chyba, proto je počítaná chyba inliers vzhledem k výslednému modelu. Počet bodů výsledný stav, který byl označený jako opravdu inliers, děleno počtem bodů, ze kterých byly vybrány všechny testovací body v prvním kroku, a výsledkem bude odhadová chybnost modelu. Tento výpočet je prováděn několikrát, resp. pro určitý počet iterací, přičemž výstupem každé iterace je model. Ve chvíli, kdy výsledný model neobsahuje žádnou množinu bodů, které by byly označeny jako inliers, nebo pokud je tato množina příliš malá, model je zahozen a pokračuje se na další iteraci, kde jsou vybrané nové body pro testování. Když se po hodnocení ukáže, že model obsahuje dostatečný počet inliers bodů, je přijat zároveň s jeho chybovou hodnotou.

3.7.3 Použití RANSAC

Výhodou algoritmu RANSAC je schopnost odhadnout parametry výstupního modelu s velkou přesností i přesto, že v množině použitých dat jsou outliers ve velkém počtu. Často se používá pro detekce geometrických tvarů např. úsečka, elipsa, obdélník, kruh atd. Jednou z nevýhod je větší časová náročnost, jelikož je postupně testována většina bodů nacházejících se v obrazu.

Jak je vidět na obrázku 13, pomocí RANSAC algoritmu je mezi náhodnou množinou bodů nalezený geometrický tvar – elipsa. Obrázek *(a)* obsahuje počáteční množinu bodů před jejím ohodnocením RANSACem. Výsledkem práce algoritmu je model, ve kterém jsou outliers reprezentované pomocí modrých bodů, inliers pomocí

červených. Přestože počet hypoteticky nesprávných bodů je větší než počet správných, pomocí RANSAC a předem známé rovnice elipsy je možné najít a zobrazit tento tvar.



Obrázek 13: (a) Vstupní množina dat (b) Množina dat, ve kterých je nalezená elipsa pomocí RANSAC algoritmu

Zdroj: [33]

3.7.4 Knihovna Time

I když se na první pohled zdá, že tato knihovna je pouze pro počítání FPS (Frame Per Second) a má za úkol poskytnout základní přehled o plynulosti práce systému, ve skutečnosti je navíc podle plynulosti detekce vypočítáno reálné množství robotem detekovaných a ujetých dlaždic.

```
39  def count_frame_per_second(start_time):
40      return 1.0 / (time.time() - start_time)
41
```

Obrázek 14: Metoda pro počítání aktuálních snímků za uvedenou jednotku času

Zdroj: Vlastní zpracování

Rozpoznávání jednotlivých dlaždic v aplikaci je dostatečně rychlé a spolehlivé. Následně je potřeba počítat, kolik takových detekovaných objektů mobilní robot za celou dobu ujede. To není problém, zároveň je evidován celkový počet detekovaných objektů, abychom měli přehled o funkčnosti neuronové sítě.

Díky informaci, kolik snímků za sekundu zpracovává samostatné vlákno, a informaci, kolik celkových dlaždic bylo nalezeno, můžeme jednoduše spočítat počet ujetých/reálně detekovaných objektů a vydělit celkový počet dlaždic počtem snímků za sekundu.

4 Implementace

V rámci této práce je pomocí knihovny OpenCV a s použitím Haarových příznaků implementován algoritmus pro rozpoznávání obrazu. Všechny metody a algoritmy byly vytvořeny v programovacím jazyce Python, který je v dané problematice velice používaný. V předchozí kapitole bylo popsáno, jak fungují jednotlivé části a algoritmy použité v aplikaci. Tato kapitola má za úkol popsat praktickou implementaci těchto částí a popsat jejich funkčnost.

4.1 Implementace neuronové sítě

Jedním z úkolů, které je potřeba řešit, je rozpoznávání obrazu z kamery, sledující povrch před mobilním robotem. Objekt pro detekce bude mít předem známý vzor – dlaždice. Pomocí funkce, která je poskytnuta přímo z knihovny OpenCV, je do naší aplikace zakomponována konvoluční neuronová síť pro provedení potřebných rozhodnutí a vyhodnocení obrazu.

4.1.1 Cascade Classifier

Každá fáze klasifikátoru označuje specifickou oblast zkoumaného obrazu jako primárního kandidáta na pozici objektu potřebného pro detekce [34]. Rozhodování se provádí pomocí označení daného úseku jako pozitivní, nebo negativní. Pozitivní znamená, že objekt byl nalezen, označení negativní naopak signalizuje, že zadaný objekt nebyl nalezen. Pokud označení přináší negativní výsledek, klasifikace této konkrétní oblasti je dokončena a umístění ukazatele je přesunuto na další místo v příslušném obrazu [34] [15].

Klasifikátor vyhodnotí situaci jako pozitivní, když všechny fáze, včetně té poslední, poskytují výsledek, který říká, že v daný okamžik je na obrazu nalezen objekt, jehož příznaky jsou shodné s příznaky známými z neuronové sítě [34].

Pravdivé pozitivní hodnocení znamená, že objekt je skutečně v obrazu a klasifikátor jej označuje jako shodný s potřebnými daty (označuje ho totiž jako pozitivní výsledek). Falešný pozitivní výskyt znamená, že proces označování

špatně vyhodnotí, že objekt je umístěn v obraze, i když je zřejmé, že není. Falešný negativní výskyt nastane, když klasifikátor není schopen detekovat objekt z obrázku, na kterém se nachází, skutečný negativní znamená, že objekt byl správně klasifikován jako nezajímavý pro dané rozhodování [35].

cv2.CascadeClassifier() je metoda pro detekci objektu v nějakém poskytnutém streamu, lépe řečeno v určitém záznamu. Vyžaduje tedy jeden parametr, a to cestu k umístění předem konfigurované neuronové sítě. Přijatelné jsou jak Haar, tak i LBP klasifikátory. LBP (Local Binary Patterns) je jednoduchá metoda pro extrakci texturních příznaků a byla představená nejprve za účelem rychlejšího měření kontrastu obrazu.

4.1.1.1 detectMultiScale()

Pro úspěch je dále důležité vhodným způsobem připravit video (obraz) pro zpracování a metodou detectMultiScale() inicializovat detekci objektu, a to pomocí předem známého příznaku.

Je vhodné zdůraznit, že tato metoda jako výsledek vrátí seznam bodů reprezentovaný jako čtverec, který pokrývá detekovaný objekt.

4.1.2 Zpracovávání vstupního obrazu

Obvykle je pro dosažení lepších výsledků potřeba obraz nejprve zpracovat – zlepšit kvalitu například odstraněním šumu.

Nativní metoda poskytnutá knihovnou OpenCV pro zpracovávání vstupního obrazu je sice funkční, ale když dochází ke klasifikaci více dat v reálném čase, není tak rychlá, jak by bylo potřeba pro větší spolehlivost naší aplikace. Jelikož však Python umožní udělat vícevláknovou aplikaci, je rozumné přenést úlohu čtení vstupního obrazu na jiné vlákno. Pro hlavní vlákno zůstane úkol zpracovávání této informace a její správné zobrazení. Takový typ zpracování vstupu aplikace nám pomůže dosáhnout knihovna imutils a třída FileVideoStream. Metoda start(), kterou poskytne tato třída, spouští zpracovávání videa v dalším vlákně (viz obrázek 15), tím uvolní prostor pro jiné potřebné operace.

```
cap = FileVideoStream('video/IMG_4416_2.mp4').start()
```

Obrázek 15: Načtení testovacího videa pomocí FileVideoStream

Zdroj: Vlastní zpracování

4.2 Implementace Feature detectoru pomocí OpenCV

Knihovna OpenCV obsahuje v sobě velké množství metod pro usnadnění práce s grafickými objekty a manipulace s obrazem. Feature detektor není výjimka, knihovna má už připravené metody pro detekce významných bodů v obrazu původně napsaných v jazyce C a C++, ale implementovaných i pro jiné jazyky, jako je třeba Python.

První metoda, kterou je potřeba použít, se jmenuje *goodFeaturesToTrack()*. Jedná se o první krok v hledání významných oblastí obrazu, kde je vracený model – množina bodů v předchozím snímku.

V [36] jsou popsané vstupní parametry pro tuto metodu:

- Nejdůležitějším parametrem je vstupní obrázek předchozího snímku speciálně konvertovaný do 8bitového obrazu s jedním barevným kanálem.
- **MaxCorners** uvádí, jakou maximální velikost může mít výsledná množina bodů.
- **QualityLevel** – parametr, který určuje minimální úroveň kvality pro jednotlivé nalezené části obrazu, aby je bylo možné označit jako významné. Z aktuálně evidované množiny bodů se zjistí „nejlepší“ bod, tzv. bod, který má nejvyšší úroveň, tím se vynásobí koeficient. Výsledek je minimální úroveň kvality pro pozitivní označení bodů.
- **MinDistance** – minimální možná euklidovská vzdálenost mezi dvěma nalezenými body.
- **BlockSize** – velikost průměrného bloku pro výpočet derivační kovariační matice na každé okolí pixelů.

Většina grafických algoritmů pracuje s obrázky s pouze jedním barevným kanálem – šedým. Je to proto, aby byl výpočet méně náročný pro procesor. Znamená to, že před detekcí významných bodů ve stávajícím snímku je potřeba ho konvertovat do šedivého odstínu. Metoda pro hledání bodů v aktuálně zpracovávaném snímku je *calcOpticalFlowPyrLK()*. Účelem je porovnávat dva 8bitové snímky a zjistit, jaké body nalezené v předchozím kroku opravdu mohou být významné. Poslední parametr představuje kritérium určující ukončení iterativního vyhledávacího algoritmu, jehož v našem případě není potřeba.

Poté, co jsou k dispozici množiny významných bodů z obou snímků, je potřeba zkontrolovat, jestli jsou opravdu stejné velikosti, a případně je vyfiltrovat a ponechat jenom body, které disponují statusem číslo 1. Jelikož obraz, který se snažíme analyzovat, se skládá z dlaždic, je logické, že mezi 2 snímky souřadnice opravdu významných bodů by se neměly výrazně lišit, proto je následujícím krokem manuální zamítnutí bodů, které mají správný status, ale zároveň mají příliš velkou vzdálenost mezi sebou ve dvou po sobě jdoucích snímcích. Když bod splní podmínku, zůstává a je zobrazen do okna. Pokud ne, jeho index se přidá do pole a následně se zmaže. Poslední metoda *estimateAffinePartial2D()* vrací matice 2 × 3, která obsahuje informace o posunu, natočení, zvětšení a rotace bodů mezi jednotlivými snímky.

```
prev_pts = cv2.goodFeaturesToTrack(prev_gray, maxCorners=200, qualityLevel=0.1, minDistance=30, blockSize=3)

# convert BGR to GRAY only
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# Calculate optical flow (i.e. track feature points)

curr_pts, status, err = cv2.calcOpticalFlowPyrLK(prev_gray, gray, prev_pts, None)

# Sanity check
assert prev_pts.shape == curr_pts.shape

# Filter only valid points
idx = np.where(status==1)[0]
# print(idx)
prev_pts = prev_pts[idx]
curr_pts = curr_pts[idx]

for j in range(0, len(curr_pts)) :

  points_x = prev_pts[j][0][0]
  point_y = prev_pts[j][0][1]

  point_x2 = curr_pts[j][0][0]
  point_y2 = curr_pts[j][0][1]

  if (abs(points_x - point_x2) <= length) and (abs(point_y - point_y2) <= length) :

    cv2.circle(img,tuple(prev_pts[j][0]),5,(0,255,0))
    cv2.circle(img,tuple(curr_pts[j][0]),5,(0,0,255))

  else :
    indexis.append(j)
    cv2.circle(img,tuple(prev_pts[j][0]),5,(255,0,0))
    cv2.circle(img,tuple(curr_pts[j][0]),5,(0,255,255))
    cv2.line(img, tuple(prev_pts[j][0]), tuple(curr_pts[j][0]), (255, 0, 0), 2)


prev_pts = np.delete(prev_pts, indexis, 0)
curr_pts = np.delete(curr_pts, indexis, 0)
indexis.clear()

m, inliers = cv2.estimateAffinePartial2D(prev_pts, curr_pts, ransacReprojThreshold = 10)
```

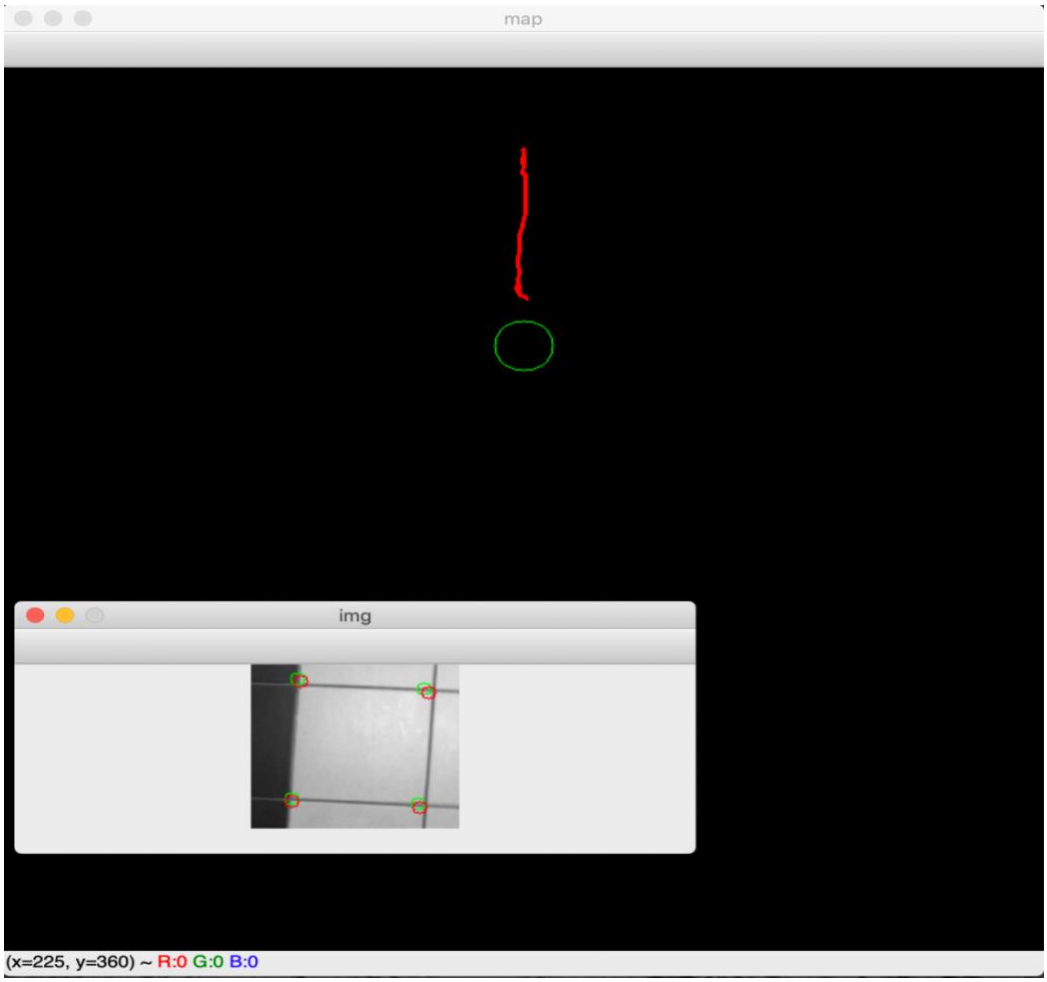
Obrázek 16: Detekce významných bodů v obraze

Zdroj: Vlastní zpracování

4.3 Vizuální odometrie a sledování pohybu robota

Vizuální odometrie je jedním z nejpoužívanějších způsobů měření pohybu objektu vůči reálnému světu. Je založena na zpracovávání obrazu z jedné, nebo více kamer [37]. Při pohybu dochází ke změnám mezi jednotlivými snímky, které jsou zaznamenávané pomocí metody *estimateAffinePartial2D()* popsané v předchozím kroku. Díky těmto informacím je možné určit, o kolik se kamera posunula a jak moc byla natočena. Na obrázku 17 je znázorněný pohyb robota v prostoru, jeho natočení vůči předchozímu snímku.

Ve druhém okně dole je možné vidět záznam z kamery robota, kde jsou zobrazeny významné body ze dvou po sobě jdoucích snímků. Bod označený pomocí zelené barvy je z předchozího snímku, červeně označený bod je ze stávajícího snímku.



Obrázek 17: Vizuální odometrie a průchod checkpointem

Zdroj: Vlastní zpracování

Výsledná transformační matice obsahuje hodnoty změny pozice souřadnic bodů vůči jeho předchozímu snímku v porovnávání se stávajícím. Nejprve je nutné nastavit počáteční bod pro sledování pohybu pomocí odometrie. Ve chvíli, kdy je nastavený počátek a také je k dispozici nenulová matice transformace, je možné spočítat translace bodů pomocí násobení matic. Aktuální stav bodů je reprezentován pomocí vektoru-matice 1 × 2 a změna pozice pomocí matice 2 × 3. Po vynásobení matic je potřeba přičíst dx a dy pro odpovídající souřadnice, aby bylo možné zjistit nejen to, jak se otočil bod, ale i jeho posun.

Poslední věc je přičtení počátku k bodu pro spočítání nových souřadnic pro tento bod a následně jeho vykreslování na plátno.

```
if p1_w == 0 or p1_h == 0:
    width_map, height_map, channels_map = img_map.shape
    p1_w = width_map / 2 # start tracing point
    p1_h = height / 2 # start tracing point

if abs(dy) < length and abs(dx) < length:
  distance += dy

print("distance: ", distance * pixel_to_cm)

if m is not None:

  wX2 = m[0,0] * wX + m[0,1] * wY + m[0,2]
  wY2 = m[1,0] * wX + m[1,1] * wY + m[1,2]

  p2_w = wX2 + width_map / 2
  p2_h = wY2 + height / 2
  print("wX2:", wX2-wX, "wY2: ", wY2-wY)

cv2.line(img_map, (int(p1_w), int(p1_h)), (int(p2_w), int(p2_h)), (0, 0, 255), 2)
```

Obrázek 18: Počítání změny pozice bodů

Zdroj: Vlastní zpracování

4.4 Pohyb podle předem navrženého vektoru k cíli

Další potřebná funkce pro zajištění správného fungování navržené aplikace je pohyb robota podle předem známého vektoru k cíli. Jako vektor v této práci je chápáno předem připravené „zadání“ v textovém souboru, které je znázorněno na obrázku 19. Na každém řádku bude obsahovat hodnoty dvou souřadnic *x* a *y*, rozdělených pomocí dvojtečky, kde první hodnota reprezentuje *x*, druhá *y*.

```
1   0:50
2   20:20
3   0:50
4   20:0
```

Obrázek 19: Zadání pohybu pro robota

Zdroj: Vlastní zpracování

Následně je soubor načten a přečten po jednotlivých řádcích, každý z nich je pomocí metody ***split(':')*** a příslušného parametru rozdělen na jednotlivé části a uložen do pole pro jeho další použití. Je zřejmé, že v souboru nebude jeden vektor s výslednými souřadnicemi ani s hodnotami, které by řekly, kam se má dostat na konci. Zvolena je ale varianta, že se robot bude pohybovat po jednotlivých checkpoitech (průjezdních bodech). Jelikož pohyb může být delší a trajektorie nemusí být přímka, je lepší, pokud se robot bude pohybovat podle menších kousků a cestou „sbírat checkpointy". V souboru je vzdálenost zadaná v centimetrech a při zobrazení na mapu v aplikaci je převedena na pixely.

Na plátně jsou jednotlivé checkpointy zobrazeny pomocí zelené barvy (viz obrázek 20), když je zobrazen, nebude načten další bod, dokud se neuskuteční průchod stávajícím.

```
if isChecked == False:
  startX = startX + int(navigation[start_navigation][0]) / pixel_to_cm
  startY = startY + int(navigation[start_navigation][1]) / pixel_to_cm

  isChecked = True

  cv2.circle(img_map,tuple([int(startX), int(startY)]),20,(0,255,0))

if (p1_h >= (startY - 20) and p1_h <= (startY + 20)) and (p1_w >= (startX - 20) and p1_w <= (startX + 20)):
  start_navigation = start_navigation + 1
  cv2.circle(img_map,tuple([int(startX), int(startY)]),20,(0,0,0))
  isChecked = False
```

Obrázek 20: Navigace robota

Zdroj: Vlastní zpracování

5 Výsledky testování

Aplikace byla testována s reálnými daty. První testy aplikace se uskutečnily z dat, která byla zaznamenávána pouze pomocí kamery mobilního telefonu drženého v ruce. Pro testování bylo natočeno 10 videí, která reprezentují pohyb podobný pohybu mobilního robota po podlaze skládající se z dlaždic. Všechna videa se snažila napodobovat různou délku cest, protože v budoucnosti robot nebude mít konstantní délku, někdy může být kratší, někdy naopak delší. Videa byla natočena v rozsahu od 1 do 10 minut (to znamená, že by se stále zvětšovala vzdálenost, kterou měl robot ujet) a také se pokaždé kamera oddalovala od podlahy, čímž se změnila distance od zařízení do dlaždic.

Testování ukázalo, že nejlepší je případ, kdy je kamera umístěna přibližně 140 až 170 cm od povrchu, který potřebujeme sledovat. Zároveň je vhodná situace, kdy má robot před sebou uprostřed celou dlaždici, aby mohl správně rozhodnout, jakým směrem by se měl pohybovat. A takto byla zjištěna nejlepší pozice pro umístění záznamového zařízení.

Pomocí detekce významných bodů v obrazech je k dispozici transformační matice bodů, která v sobě obsahuje hodnoty posunu dx a dy, díky jím je možné spočítat přibližnou vzdálenost, kterou ujel robot od začátku svojí cesty. Výsledná hodnota však znázorňuje jenom počet pixelů, když je reálně potřeba mít přehled o vzdálenosti v metrech, případně v centimetrech. Proto je nutné odvodit koeficient pro přepočet ujetých obrazových bodů na reálnou vzdálenost. Pro daný účel byl vybrán jeden ze záznamů natočených mobilním robotem.

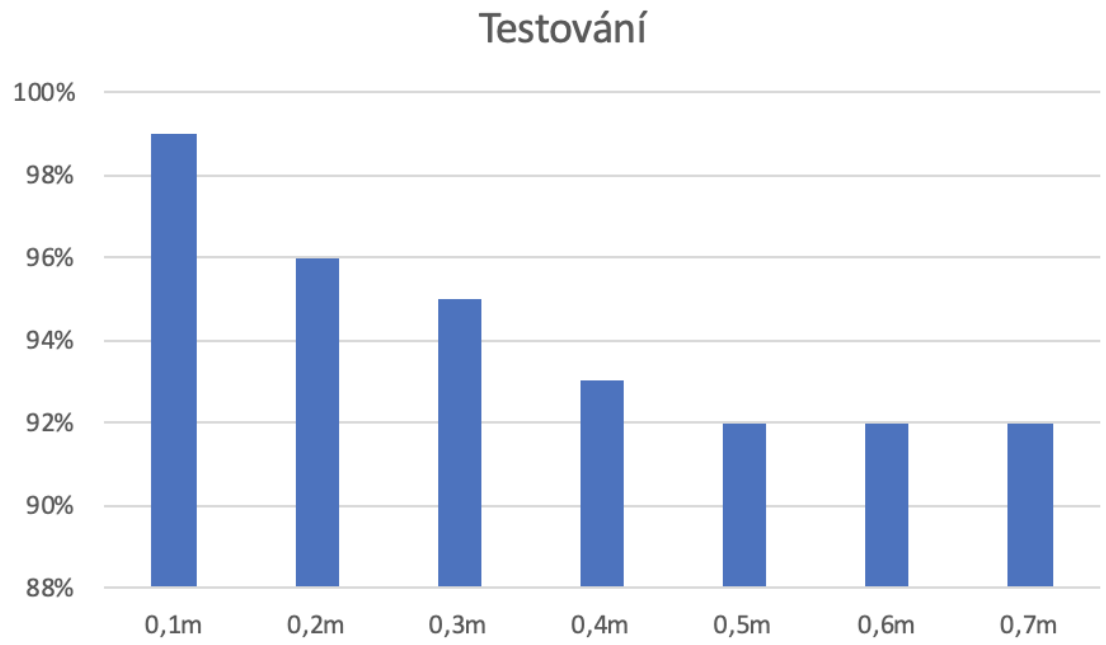
V tabulce 1 můžeme vidět, že robot by ujel 4,5 dlaždice nebo 420 pixelů. K tomu, aby bylo možné odvodit koeficient pro přepočet, je potřeba znát reálný rozměr dlaždice – 30 × 30 cm. Ve výsledku se hodnota pro 4,5 dlaždice rovná 135 cm. Pokud toto číslo vydělíme počtem ujetých pixelů, dostaneme hodnotu 0.32142857. Analýza ukázala, že jeden obrazový bod (pixel) se rovná 0.32 cm.

záznam 7					
ujeté dlaždice	0,5	1,5	2,5	3,5	4,5
ujeté pixely	60	135	210	320	420

Tabulka 1: Porovnávání ujetých dlaždic a ujetých pixelů

Zdroj: Vlastní zpracování

Na obrázku 21 je znázorněna úspěšnost při maximální testované ujeté vzdálenosti 0,7 m. Je zřejmé, že při takovém dojezdu zcela můžeme výsledky považovat za úspěšné. Spolehlivost více než 90 % je dostačující pro úspěšný provoz robota vzhledem k jeho přímému účelu.



Obrázek 21: Testování (míra správnosti výsledku dle ujeté vzdálenosti)

Zdroj: Vlastní zpracování

Testování na větší vzdálenosti prokázalo, že výsledky po ujetých 10 metrech jsou trochu horší, ale přesto jsou velice použitelné pro každodenní běžné použití.

5.1 Reálné testy z dat natočených pomocí robota

Ve chvíli, kdy byla aplikace testovaná na datech zaznamenávaných pomocí člověka imitujícího pohyb robota, a obdrželi jsme výsledky, můžeme jít do další fáze. Potřebujeme zjistit, jaké výsledky bude ukazovat naše aplikace z dat, která byla natočena přímo pomocí robota. Pro testování bylo pořízeno 9 videí

zaznamenávajících reálný pohyb z paluby mobilního robota. Mobilní telefon s kamerou, připevněný na konstrukci robota, sledoval podlahu před robotem.

Jako u předchozího testu, i zde platí obdobné pravidlo – nejlepší je případ, kdy je sledovací zařízení umístěno přibližně 140 až 170 cm od povrchu, který potřebujeme sledovat. Je vhodné kameru umístit tak, aby před sebou měla celou dlaždici.

Rychlost pohybu mobilního robota je nižší, než byla imitovaná odhadová rychlost v předchozí kapitole. Oproti předešlému testu rozpoznávání obrazu a detekce významných oblastí v obraze díky tomu poskytují ještě lepší výsledky. Shodná situace je také u počítání vzdálenosti, kterou robot ujel.

Záznamy	ujeté dlaždice	ujeté pixely
záznam 1	1,5	135
záznam 2	2,0	170
záznam 3	0,5	58
záznam 4	1,0	95
záznam 5	4,0	362
záznam 6	2,5	215
záznam 7	4,5	420
záznam 8	15,0	1587
záznam 9	7,0	614

Tabulka 2: Záznamy pohybu robota

Zdroj: Vlastní zpracování

Tabulka 2 reprezentuje výsledky testování algoritmu počítajícího ujetou vzdálenost. Testy na reálná data zaznamenávaná pomocí pohybu robota zjistily, že při jízdě na větší vzdálenosti je registrováno menší klesnutí spolehlivosti algoritmu. Avšak tuto hodnotu by se také dalo používat pro další potřebné výpočty v běžném provozu robota.

Kdyby bylo potřeba mít přesnější výsledky, je možné počítat a pak následně sčítat jenom vzdálenost mezi jednotlivými checkpointy. Daný způsob výpočtu by měl přes 90 % spolehlivosti a pravděpodobnost chyby by byla opravdu minimální.

5.2 Porovnávání výsledku testování

Provedené testy nejprve ukázaly, že nejlepší formát videa pro zaznamenávání pohybu robota je .mp4. Mezi všemi formáty, které byly zkoušeny, ukázal tento trochu lepší výsledek než ostatní. V průběhu testování se používaly tyto formáty: .mp4, .MOV, .AVI a .WMV. Rozměry záznamu se mění v průběhu práce programu, ale při použití vstupních dat rozměry 1280 × 720 přímo ze zaznamenávacího zařízení aplikace prokázala nejlepší výsledky. Pro usnadnění výpočtu a menší zatížení procesoru je však vstupní záznam zmenšený a oříznutý o obraz robota (který byl v zorném poli kamery částečně vidět), aby detekce významných bodů omylem nemohla detekovat body, které pro nás nejsou opravdu významné, a nemůžeme s nimi počítat, i když se stále budou pohybovat v každém snímku – jedná se o body na robotovi.

Co se týče počítání vzdálenosti ujeté robotem, testování dat poskytnutých přímo z robota demonstrovalo výsledky, které by bylo možné používat v každodenním provozu a případně pro další výpočty.

Vizuální odometrie prokázala horší výsledky. Pro detekce významných bodů je dlaždice až moc jednoduchý objekt, proto RANSAC algoritmus ani přes nízký počet outliers nedokázal vrátit spolehlivé hodnoty natočení bodů vůči předchozímu snímku. To ale neplatí pro posun. Díky správným a spolehlivým hodnotám posunu jednotlivých bodů mezi snímky je možné spolehlivě spočítat vzdálenost, kterou robot překonal.

6 Závěry a doporučení

V této bakalářské práci byly teoreticky popsány a prakticky navrženy a implementovány algoritmy použité pro řízení mobilního robota. První část práce poskytuje přehled o teoretickém úvodu a seznámení s existujícími řešeními v dané problematice. Druhá část je věnovaná teoretické analýze a popisu funkčnosti veškeré funkcionality výsledné aplikace. Cílem praktické části bylo navrhnout aplikaci, která by mohla fungovat jako řešení pro řízení mobilního robota, jejíž implementace a funkčnost jsou popsané v čtvrté části této bakalářské práce. Výsledky praktické části jsou uvedeny v kapitole Testování. Na základě popsaných postupů a metod byla navržena a implementována vlastní aplikace používající a disponující veškerou popsanou funkcionalitou.

Hlavními faktory při návrhu byla rychlost práce aplikace a její spolehlivost při každodenním použití. Další důležitý aspekt představoval přesnost rozpoznávání dlaždic a významných bodů v pohybujícím se obrazu. Posledním, ale neméně potřebným atributem byla přehlednost a srozumitelnost aplikace pro její další úpravy a aktualizace.

Rozpoznávání obrazu bylo řešeno pomocí algoritmu „Haar Cascade Training“, který dokáže rozpoznat dlaždice se spolehlivostí minimálně 80–85 % při vhodném umístění pozorovacího zařízení na robotovi. Počítání vzdálenosti, kterou se robotovi podařilo překonat, poskytuje velice spolehlivé výsledky, které se případně ještě dají vylepšit pomocí sčítání jen vzdálenosti mezi jednotlivými checkpointy, kde se pohybuje robot, za podmínky, že tyto body od sebe nebudou vzdáleny na příliš velké vzdálenosti.

Nicméně algoritmus pro vizuální odometrie a sledování pohybu robota nelze vyhodnotit jako velice úspěšný. Algoritmus RANSAC poskytnutý knihovnou OpenCV vrací na testovacích datech nespolehlivé hodnoty natočení bodů v pohybujícím se prostoru.

Tato bakalářská práce vycházela z poznatků získaných při studiu, nutné bylo hledání zdrojů zaměřených na konkrétní problematiku. V další práci by bylo

možné se zaměřit na optimalizaci vizuální odometrie při sledování pohybu mobilního robota, případně implementaci metod, které nejsou k dispozici v OpenCV např. SIFT a SURF nebo pomocí uvedené knihovny navrhnout vlastní algoritmus pro vizuální odometrie pohybu mobilního robota. Další možností je optimalizovat počítání vzdáleností pomocí jejího rozdělení na jednotlivé úseky.

7 Seznam použité literatury

[1] VIOLA, P. and M. JONES. Rapid Object Detection using a Boosted Cascade of Simple Features. In: *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition* [online]. 2001 [cit. 2019-03-03]. Dostupné z: https://ieeexplore.ieee.org/document/990517

[2] IBRAHIM, L. F. et al. Using Haar classifiers to detect driver fatigue and provide alerts. *Multimedia Tools and Applications.* 2014, 3(71): 1857. DOI: 10.1007/s11042-012-1308-5

[3] FRIEDMAN, J., T. HASTIE and R. TIBSHIRANI. Additive logistic regression: a statistical view of boosting. *The Annals of Statistics.* 2000, 2(28): 337–407. DOI: 10.1214/aos/1016218223.

[4] RAMON, M. C. Using OpenCV. In: RAMON, M. Carlos. *Intel® Galileo and Intel® Galileo Gen 2* [online]. Berkeley, CA: Apress, 2014, s. 319–400 [cit. 2019-01-31]. ISBN 978-1-4302-6839-0.

[5] ANAGNOSTOPOULOS, C. N. et al. Tile Classification Using the CIELAB Color Model. In: SUNDERAM, V. S. et al. (eds.). *Computational Science – ICCS 2005* [online]. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, s. 695–702 [cit. 2019-01-31]. ISBN 978-3-540-26032-5. Dostupné z: http://link.springer.com/10.1007/11428831_86

[6] CIELAB color space. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001-2009 [cit. 2019-01-30]. Dostupné z: https://en.wikipedia.org/wiki/CIELAB_color_space

[7] THOMAS, J. B., P. COLANTONI and A. TRÉMEAU. On the Uniform Sampling of CIELAB Color Space and the Number of Discernible Colors. In: TOMINAGA, S., R. SCHETTINI and A. TRÉMEAU (eds.). *Computational Color Imaging* [online]. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, s. 53–67 [cit. 2019-01-31]. ISBN 978-3-642-36699-4. Dostupné z: http://link.springer.com/10.1007/978-3-642-36700-7_5

[8] ADAMEC, V. *Zpracování a rozpoznávání obrazu*. Olomouc, 2011. Bakalářská práce. Univerzita Palackého v Olomouci, Přírodovědecká fakulta, Katedra informatiky. Vedoucí práce Eduard Bartl.

[9] YOUSIF, K., A. BAB-HADIASHAR and R. HOSEINNEZHAD. An Overview to Visual Odometry and Visual SLAM: Applications to Mobile Robotics. *Intelligent Industrial Systems* [online]. 2015, 1(4): 289 [cit. 2019-05-05]. Dostupné také z: https://doi.org/10.1007/s40903-015-0032-7

[10] CHENG, Y., M. MAIMONE a L. MATTHIES. Visual odometry on the Mars Exploration Rovers. *IEEE Robotics and Automation Magazine*. 2005, 1, 903–910. DOI: 10.1109/ICSMC.2005.1571261.

[11] HERMAN, P., A. R. MOSTEOYZ and A. C. MURI. *A low-cost custom-built robotic arm on the TurtleBot platform.* Zaragoza: University of Zaragoza, Dpt. de Informática e Ingeniería de Sistemas.

[12] FIROOZIAN, R. *Servo Motors and Industrial Control Theory.* New York: Springer, 2009. ISBN 978-1-4419-4665-2.

[13] RUSU, R. B. and S. COUSINS. 3D is here: Point cloud library (PCL). In: *IEEE Int. Conf. Robotics and Automation (ICRA)*, 2011, 1–4.

[14] DOWNEY, A. *Think Python.* 2nd ed. updated for Python 3. Sebastopol, CA: O'Reilly Media, 2016. ISBN 1491939362.

[15] BRADSKI, G. R. *Learning OpenCV*. Sebastopol: O'Reilly, 2008. ISBN 978-0-596-51613-0.

[16] Neural Network Models of Human Learning. In: SEEL, N. M. et al. (eds.). *Encyclopedia of the Sciences of Learning.* Boston, MA: Springer, 2012. ISBN 978-1-4419-1427-9.

[17] HLAVOŇ, D. *Hluboké neuronové sítě pro analýzu 3D obrazových dat.* Brno, 2016. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Michal Španěl.

[18] Mendelova univerzita v Brně. *Neuronové sítě* [online]. Brno: Mendelova univerzita [cit. 2019-05-05]. Dostupné z: https://is.mendelu.cz/eknihovna/opory/zobraz_cast.pl?cast=21471.

[19] REK, P. *Knihovna pro návrh konvolučních neuronových sítí.* Brno, 2018. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Lukáš Sekanina.

[20] SCHLEGL, T. et al. Predicting Semantic Descriptions from Medical Images with Convolutional Neural Networks. In: *Information processing in medical imaging: proceedings of the ... conference.* 2015, 24: 437-48. DOI: 10.1007/978-3-319-19992-4_34.

[21] Humusoft s. r. o. Deep Learning v prostředí Matlab: systémy strojového vidění; identifikace zboží a osob. *Automa* [online]. 2017, (5): 12–14 [cit. 2018-01-02]. Dostupné z: http://www.automa.cz/Aton/FileRepository/ pdf_articles/10482.pdf

[22] DESHPANDE, A. A Beginner's Guide To Understanding Convolutional Neural Networks. *Adeshpande3.github.io* [online]. 20. 6. 2016 [cit. 2018-01-02]. Dostupné z: https://adeshpande3.github.io/adeshpande3.github.io/A-Beginner's-Guide-ToUnderstanding-Convolutional-Neural-Networks/

[23] MITRENGA, M. *Konvoluční neuronová síť pro segmentaci obrazu.* Brno, 2018. Bakalářská práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií. Vedoucí práce Václav Jirsík.

[24] KARN, Ujjwal. An intuitive explanation of convolutional neural networks. *Ujjwalkarn.me* [online]. 11. 8. 2016 [cit. 2019-12-12]. Dostupné z: https://ujjwalkarn.me/2016/08/11/ intuitive-explanation-convnets/

[25] PAPAGEORGIOU, C. P., M. OREN and T. A. POGGIO. General framework for object detection. In: *International Conference on Computer Vision*, 1998, p. 555–562. DOI: 10.1109/ICCV.1998.710772.

[26] HAAR, Alfréd. Zur Theorie der orthogonalen Funktionensysteme. *Mathematische Annalen*. 1910, 69(3): 331–371. ISSN 0025-5831. DOI: 10.1007/BF01456326

[27] LEPIK, Ü. and H. HEIN. *Haar Wavelets: with applications.* New York: Springer, 2014. ISBN 978-3-319-04294-7.

[28] VIOLA, P. and M. J. JONES. Rapid object detection using a boosted cascade of simple features. In: *Conference On Computer Vision And Pattern Recognition*, 2001.

[29] BOGGARAPU, S. and T. SREENIVASU. An Optimized Implementaion of Haar like Feature Based Object Detection. *International Journal of Innovative Research and Development*. 2013, 2(1): 37–47. ISSN 2278–0211.

[30] BARBORKA, P. *Analýza metod pro detekci příznaků v digitalizovaném obraze.* Plzeň, 2016. Bakalářská práce. Západočeská univerzita v Plzni, Fakulta aplikovaných věd Katedra kybernetiky. Vedoucí práce Petr Neduchal.

[31] SONKA, M., V. HLAVAC and R. BOYLE. Image Processing, Analysis, and Machine Vision. *Journal of Intelligent Learning Systems and Applications*. 2017, 9(4).

[32] *International Journal on Smart Sensing and Intelligent Systems*. 2016, 9(3). ISSN 1178-5608. DOI: 10.21307/ijssis-2017-923.

[33] HŘÍBEK, P. *Detekce elipsy v obraze.* Brno, 2008. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií, Ústav počítačové grafiky a multimédií. Vedoucí práce Michal Hradiš.

[34] The MathWorks, Inc. Train a Cascade Object Detector. *Mathworks.se* [online]. 2014 [cit. 2019-05-05]. Dostupné z: http://www.mathworks.se/help/vision/ug/train-a-cascadeobject-detector.html#btugex8

[35] SOO, S. *Object detection using Haar-cascade Classifier*. Tartu: Institute of Computer Science, University of Tartu.

[36] SHI, J. and C. TOMASI. Good Features to Track. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. Seattle, 1994, p. 593–600.

[37] PIVOŇKA, T. *Vizuální odometrie pro dynamickou rekonstrukci obrazu.* Praha, 2018. Diplomová práce. České vysoké učení technické v Praze, Fakulta elektrotechnická. Vedoucí práce Libor Přeučil.

8 Přílohy

Struktura přiloženého CD

- ***dasar_haartrain/*** - řešení s neuronovou sítí – složka s předem připravenými soubory použitými pro trénování neuronové sítě.
  - ***positive/*** - složka obsahující informace, potřebné pro určení pozitivních obrázku následně použitých pří trénování sítě.
    - ***rawdata/*** - složka obsahující pozitivní obrázky, tzv. obrázky na kterých jsou dlaždice.
    - ***objectmarker.exe*** - soubor použitý pro znázornění umístění dlaždice na jednotlivých obrazcích pomocí grafický jejich označení.
    - ***info.txt*** - výstupní soubor obsahující názvy obrázku a souřadnice umístění dlaždice.
  - ***negative/*** - seznam negativních obrázku na kterých nejsou dlaždice.
    - ***create_list.bat*** – soubor pro tvorbu seznamu negativních obrázku.
    - ***bg.txt*** – soubor obsahující informace o umístění negativních obrázku.
  - ***01 samples_creation.bat*** – skript pro spouštění createsamples.exe souboru, obsahuje cestu na složku s vzorové obrázky.
  - ***createsamples.exe*** - vytvoření vector.vec souboru.
  - ***vector/vector.vec*** - vektorový znázornění umístění dlaždic.
  - **02 haarTraining.bat** – skript pro spouštění haartraining.exe souboru, obsahuje informace o počtu pozitivních a negativních obrázcích.
  - ***haartraining.exe*** – soubor pro trénování konvoluční neuronové sítě.
  - ***cascades/*** - složka, obsahující vrstvy konvoluční neuronové sítě, které vytvoří haartraining.exe.
  - ***03 convert.bat*** – skript pro spouštění haarconv.exe.
  - ***haarconv.exe*** – vytvoří myhaar.xml.
  - ***myhaar.xml*** – soubor obsahující strukturu konfigurace sítě pro OpenCV.

- ***tile_detection/*** - řešení s feature-detektorem a vizuální odometrií – složka, kde jsou umístěné zdrojové soubory aplikace.
  - ***app/*** - složka se zdrojovými kódy aplikace.
    - ***tile_detection.py*** – zdrojový kód pro finální verzi aplikace používající feature detekce.
    - ***FramesCounter.py*** – soubor poskytující informace o počtu snímku ve vybraném záznamu.
    - ***file_reader.py*** – soubor jehož kód má za úkol přečíst soubor se „zadáním“ pro navigace mobilního robota.
  - ***cnn/myhaar.xml*** – konvoluční neuronová síť pro rozpoznávání obrazu.
  - ***nav/nav.txt*** – soubor se zadání pro navigace mobilního robota.
- ***app_prev/haar_tile_detection.py*** - zdrojový kód pro rozpoznávání obrazu pomocí konvoluční neuronové sítě bez použití vizuální odometrie.
- ***video/*** - obsahuje všichni záznamy použité pro testování aplikace.

Oskenované zadání práce

Univerzita Hradec Králové
Fakulta informatiky a managementu
Akademický rok: 2018/2019

Studijní program: Aplikovaná informatika
Forma: Prezenční
Obor/komb.: Aplikovaná informatika (ai3-p)

Podklad pro zadání BAKALÁŘSKÉ práce studenta

PŘEDKLÁDÁ:	ADRESA	OSOBNÍ ČÍSLO
Melnykovych Ivan	Kydlinovská 231/67, Hradec Králové - Plácky	I1600578

TÉMA ČESKY:

Rozpoznávání obrazu a autonomní řízení mobilního robota

TÉMA ANGLICKY:

Image Recognition and Autonomous Mobile Robot Control

VEDOUCÍ PRÁCE:

Ing. Pavel Kříž, Ph.D. - KIKM

ZÁSADY PRO VYPRACOVÁNÍ:

Cíl: Navrhnout a implementovat řešení pro rozpoznávání obrazu z kamery sledující povrch před mobilním robotem. Povrch (podlaha) bude mít předem známý vzor - dlaždičky. Z obrazu bude odvozena ujetá vzdálenost robota a relativní úhel natočení. Robot bude schopen jet podle předem navrženého vektoru k cíli.

Osnova:
1. Úvod
2. Rešerše existujících řešení
3. Analýza a návrh řešení
4. Implementace
5. Výsledky testování
6. Závěr

SEZNAM DOPORUČENÉ LITERATURY:

http://answers.opencv.org/question/179977/i-am-trying-to-detect-largest-tile-on-the-floor/
https://varcity.ethz.ch/paper/wacv2017_lettry_repetitions.pdf
https://opencv.org/
http://theconversation.com/how-do-robots-see-the-world-51205

Podpis studenta: ..... **Datum:** 10.10.18

Podpis vedoucího práce: ..... **Datum:** 10.10.18