České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra kybernetiky

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student: Mykhaylo Z e l e n s k y y

Studijní program: Kybernetika a robotika (bakalářský)

Obor: Robotika

Název tématu: Následování člověka mobilním robotem

Pokyny pro vypracování:

Navrhněte a implementujte systém pro navigaci mobilního robotu tak, aby byl schopen následovat člověka, např. jako automatizovaný nosič nákladu. Prostudujte existující metody, které je možné pro řešení úlohy využít a zaměřte se především na vizuální metody založené na zpracování obrazu z kamery umístěné na robotu. Předpokládá se, že na těle následované osoby bude umístěna jedna nebo více vizuálních značek pro detekci kamerou. V případě, že to bude výhodné, použijte další senzory (např. laserový dálkoměr). Výstupem práce bude demonstrační aplikace řídící reálný robot a vyhodnocení kvality sledování a bezpečnosti.

Seznam odborné literatury:

[1] Roland Siegwart, Illah Reza Nourbakhsh and Davide Scaramuzza: Introduction to Autonomous Mobile Robots, MIT Press, 2011.
[2] Tomáš Krajník, Matias Nitsche et al.: A Practical Multirobot Localization System. Journal of Intelligent and Robotic Systems (JINT), 2014.
[3] Milan Šonka a Václav Hlaváč: Počítačové vidění. Praha: Grada, 1992.

Vedoucí bakalářské práce: Ing. Jan Chudoba

Platnost zadání: do konce letního semestru 2017/2018

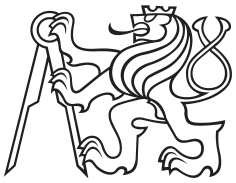
L.S.

prof. Dr. Ing. Jan Kybic
vedoucí katedry

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 10. 1. 2017

Bakalářská práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra kybernetiky

Následování člověka mobilním robotem

Mykhaylo Zelenskyy

Školitel: Ing. Jan Chudoba
Květen 2017

Poděkování

Rád bych poděkoval vedoucímu mé bakalářské práce, Ing. Janu Chudobovi, za jeho cenné rady a spolupráci.

Zvláštní poděkování patří také mé rodině a blizkým za jejich podporu během celého studia.

Prohlášení

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

V Praze, 23. května 2017

Podpis autora práce

Abstrakt

V práci je probírána problematika následování člověka mobilním robotem. Pro detekci vizuální značky, dle které je člověk sledován, je použit korelační algoritmus a detektor ArUco markerů. Pro řízení robotu je použit diskrétní PID regulátor. Pro vyhýbání se překážkám je použit algoritmus Vector Field Histogram. Software byl vyzkoušen v simulátoru V-Rep a také na reálném robotu.

Klíčová slova: robotika, počítačové vidění, řízení robotu, korelace, ArUco, PID, VFH, V-Rep

Školitel: Ing. Jan Chudoba

Abstract

This thesis is focused on the problem of following of human by mobile robot. The correlation algorithm and ArUco marker detector are used for detection of visual mark used for defining the followed person. Discrete PID controller is used for robot control. Also Vector Field Histogram algorithm is used for obstacle avoidance. Software was tested in V-Rep simulator and with real robot.

Keywords: robotics, computer vision, robot control, correlation, ArUco, PID, VFH, V-Rep

Title translation: Following of Human by Mobile Robot

Obsah

Obrázky

Tabulky

Kapitola 1
Úvod

Pomocný robot se může mnohým z nás zdát jako futuristický sen. Takový robot by místo nás nosil věci, pomáhal při nákupech, asistoval v nemocnicích či ošetřoval raněné ve válkách. Měl by tolik výhod, že by v budoucnu mohlo být trendem takového pomocníka vlastnit.

Bylo provedeno mnoho různých výzkumů ohledně návrhu robotů, které by dokázaly sledovat člověka. S tímto problémem jsou spojené dvě důležité otázky. Jaké senzory použit pro lokalizaci člověka? Jak řešit řízení a navigaci robotu tak, aby udržoval určitou vzdálenost a vyhýbal se případným překážkám?

Nejdříve je třeba definovat, co vlastně je robot následující člověka. Jedná se o mobilního robota, který sleduje určitou osobu a zároveň objíždí překážky a chová se k ostatním lidem jako k překážkám, tj. nezmění cíl sledování během jízdy.

Takový robot typicky může být vybaven mnoha různými senzory, např. laserovým, zvukovým nebo IR dálkoměrem, aby mohl měřit vzdálenost od překážek, kamerou pro detekci sledovaného objektu, bezdrátovým přenášečem signálu, GPS apod. Tyto senzory by měly fungovat současně, aby robot dokázal, co se od něj očekává.

1.1 Cíl práce

Cílem této práce je navrhnout systém pro řízení robotu tak, aby dokázal splnit úkoly definované v úvodu.

Při návrhu tohoto systému je třeba uvědomit si několik věcí:

1. Pro detekování člověka by měl být použit takový algoritmus, který dokáže fungovat v prostředí, kde kromě sledovaného člověka jsou i další osoby.

2. Cíl se nesmí pohybovat průměrnou rychlostí větší, než je maximální rychlost robotu. Může se stát, že robot už nikdy nedokáže najít sledovaného člověka, dokud se dotyčný nevrátí dostatečně blízko k robotu.

3. V případě použití dálkoměru pro měření vzdálenosti mezi objekty je důležité, aby systém nedetekoval sledovaný cíl jako překážku.

Obrázek 1.1: Robot následující člověka [1]

1.2 Motivace

Jak již bylo řečeno, robot s podobnou funkcionalitou by byl velice užitečný jako pomocník v různých oblastech lidské činnosti. V medicínských zařízeních a domovech pro seniory by se takový robot mohl starat o pacienty (viz obr. 1.1), také by našel své uplatnění v armádě, kde by pomáhal vojákům s převozem nákladů. Vylepšená varianta robota by mohla posloužit jako tělesná stráž.

Kapitola 2

Návrh systému

Problematiku detekce člověka lze řešit pomocí nevizuálních nebo vizuálních metod. Zatímco co první používají různé senzory, jako např. dálkoměr, druhé spoléhají na RGB kameru. Každá z těchto metod má své výhody a nevýhody, proto je před samotným návrhem programu pro následování člověka mobilním robotem třeba zhodnotit používané přístupy.

Nevizuální metody

Mezi nevizuální metody pro detekci člověka patří například takové, které používají laserový dálkoměr nebo termální kameru. V práci [5] je popsána metoda, která na základě měření z dálkoměru dokáže najít lidský pás. V základu je principem tohoto postupu to, že naměřená data jsou dle definovaných pravidel rozdělena do shluků. Každý shluk je pak otestován, zda vyhovuje určitým parametrům, jako jsou hloubka, šířka a obvod. Shluk, který odpovídá těmto parametrům nejvíce, je vyhodnocen jako člověk, který pak následně může být sledován. Ačkoliv tato metoda při správné volbě parametrů může být robustní, má své nevýhody. Například je složité nastavit parametry tak, aby popisovaly jakoukoliv osobu. Při špatné volbě může docházet ke ztrátě cíle nebo chybným detekcím, při parametrizaci pro každého konkrétního člověka potom naopak klesá znovupoužitelnost metody.

Dále jsou pro detekci lidí používány termální kamery [6]. I když se tato metoda častěji používá pro monitorování různých objektů, lze ji využít i pro následování člověka. Principem této metody je, že ve výstupu použité kamery je detekován "hot spot", který je pak následně klasifikován např. pomocí neuronové sítě nebo jednodušších metod. Tato metoda je velice robustní pro detekci člověka, nicméně použití termální kamery pro sledování může být problematické. Pokud si robot nezapamatuje tvar sledovaného člověka, může dojít ke změně cíle, pokud v záběru bude více lidí.

Bez povědomí o sledovaném člověku nemusí být nevizuální metody ve všech případech vhodné. Od robotu se očekává, že bude sledovat konkrétní cíl v bez ohledu na přítomnost jiných lidí v jeho okolí, proto je vhodnější použít vizuální přístup k detekci a sledování člověka.

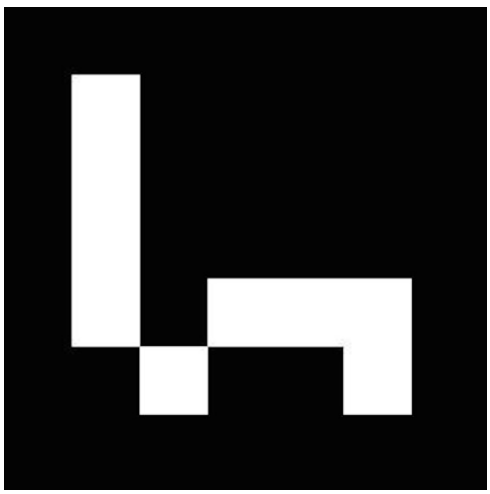
Vizuální metody

Existuje několik výzkumných projektů, které se zabývají problematikou pomocných robotů, které by následovaly člověka za použití vizuálních metod. Většina těchto projektů se zakládá na detekci dynamického objektu, který je následně sledován. V některých pracích jsou použity neuronové sítě pro detekci obličeje spolu s background subtraction pro nalezení pohybu v obraze [7], např.kráčejících nohou. Avšak použití tohoto algoritmu vede k tomu, že sledovaný člověk musí být otočen k robotu obličejem, a prezence několika lidí v místnosti přivede k tomu, že robot nedokáže určit, koho musí následovat. V případě, že neuronová síť bude naučena pouze na konkrétního člověka, což by tento problém mohlo řešit, znovupoužitelnost algoritmu klesá. Jelikož síť by byla naučena pouze pro jednu osobu, bylo by ji třeba přeučovat při každém novém použití.

Dalším možným řešením této problematiky je použití barevného markeru [8], což znamená, že robot hledá určitou shodu barev, kterou pak sleduje. V tomto případě člověk může být otočen k robotu zády, a algoritmus může být použit i v prostorech plných lidí. Nicméně detekce barevného vzoru nemusí být robustní kvůli změně osvětlení nebo špatné volbě barev, které nebudou dostatečně kontrastní v porovnání s oblečením sledovaného člověka.

2.1 Volba vizuální značky

Po vyhodnocení možných přístupu k řešení problému následování člověka mobilním robotem bylo rozhodnuto použít vizuální metodu, která by se zakládala na hledání specifické vizuální značky umístěné na člověku. Cíl pro sledování musí být unikátní, a algoritmus detekce musí být robustní, aby nedocházelo k tomu, že robot nebude vědět, koho má sledovat. Bude známo, jaký typ značky robot musí hledat, proto se snižuje šance, že si splete cíl s jiným člověkem nebo překážkou.



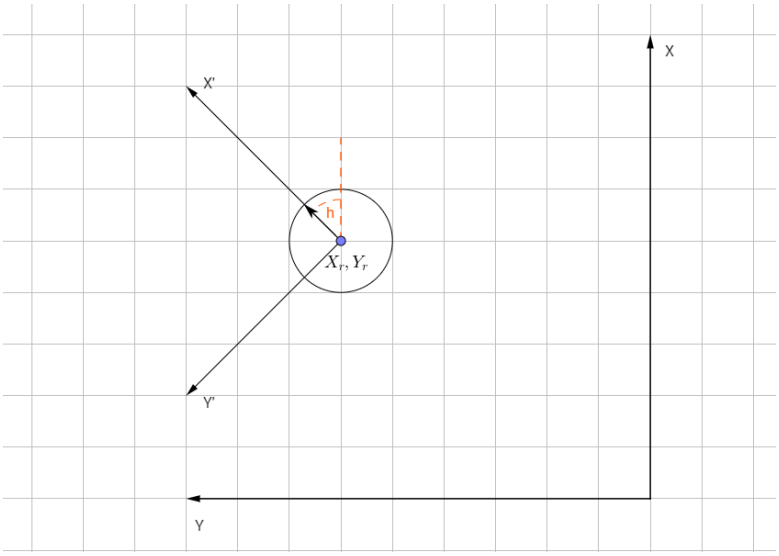
Obrázek 2.1: Příklad vizuální značky

Mobilní robot bude následovat člověka, který má na sobě umístěnou předem známou vizuální značku. Pro navigaci robotu se v praxi běžně používají markery pro rozšířenou realitu [9][10][11]. Jejich detekce je většinou jednoduchá a mohou v sobě uchovávat užitečnou informaci, např. identifikační číslo, podle kterého robot pozná, koho sleduje. Jednu z možných vizuálních značek, které jsou v této práci použity, lze vidět na obrázku 2.1.

2.2 Volba souřadnicového systému

Pro řízení robotu je třeba zvolit souřadnicový systém, ve kterém se bude pohybovat.

Počátek zvolené souřadnicové soustavy je umístěn do počáteční pozice robotu a osa X je ve stejném směru, jako počáteční směr jízdy robotu. Kladný směr jízdy robotu je v kladném směru osy Y, jak je vyobrazeno na 2.2, a je v rozmezí $(-\pi; \pi)$. Poloha cíle (viz sekci 3.3) je poté vztažena k poloze robotu a je počítána v čárkované soustavě, jejíž počátek je umístěn do aktuální pozice robotu, a je vůči původní otočená o h, kde h je směr jízdy robotu.



Obrázek 2.2: Použitý souřadnicový systém

2.3 Diferenciální řízení

Celý systém bude navržen pro řízení robotů s diferenciální kinematikou. Tyto roboty mají dvě nezávisle poháněná kola, jsou schopny otáčet se na místě kolem své osy a pohybují se pouze vpřed či vzad. Polohu takového robotu v relativní souřadnicové soustavě lze spočítat jako:

$$\begin{bmatrix} x \\ y \\ h \end{bmatrix} = \begin{bmatrix} x + \frac{(\Delta R + \Delta L) r_k}{2} \cos(h) \\ y + \frac{(\Delta R + \Delta L) r_k}{2} \sin(h) \\ h + \frac{(\Delta R - \Delta L) r_k}{2 r_r} \end{bmatrix}, \tag{2.1}$$

přičemž ΔL a ΔR značí změnu vnitřní polohy levého a pravého kloubů za čas Δt, r_k je poloměr kola a r_r je poloměr robotu, neboli polovina vzdálenosti mezi koly.

2.4 Implementační záležitosti

Protože se očekává, že běh softwaru bude v reálném čase, musí být rychlý a stabilní. Z těchto důvodů bylo rozhodnuto použit jazyk C++.

V práci je také použita knihovna OpenCV[12], která nabízí obrovské možnosti pro zpracování obrazu. Tato knihovna obsahuje přes 2500 optimalizovaných algoritmů pro počítačové vidění a strojové učení. Některé z nich budou použity pro detekci vizuální značky ve výstupu z kamery. Navíc je v této knihovně implementován algoritmus pro detekci ArUco markeru [9], který je podrobněji popsán v sekci 3.2.2.

Kapitola 3

Detekce

3.1 Zpracování obrazu z kamery

Než bude možné použít rozpoznávací algoritmy, výstup z kamery musí být náležitě zpracován, aby odpovídal formátu, se kterým algoritmy pracují.

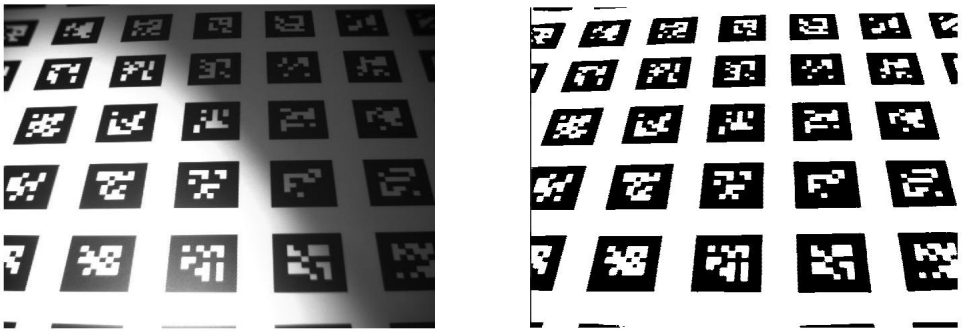
Načtený snímek je převeden do černobílé podoby, je tedy použito prahování. Jelikož se předpokládá, že se robot může pohybovat v prostředí s nerovnoměrným osvětlením, pro binarizaci obrazu je vhodné použit adaptivní prahování [2] (viz obr. 3.1). Tato metoda počítá práh pro malé části obrazu místo globálního nastavení prahu pro celý snímek.

Dále je třeba najít kontury. Cannyho algoritmus pro nalezení kontur je implementován v knihovně OpenCV. Tato funkce navíc zajišťuje zachování hierarchie kontur[3], tedy topologii obrázku, jak je vidět na obr. 3.2. Tato informace je užitečná pro následné rozpoznávání, protože pak mohou být kontury na nejnižší nebo nejvyšší úrovni hierarchie ignorovány, pokud se předpokládá, že vizuální značka bude umístěna v ochranné zóně a bude obsahovat další podvzory.

3.2 Rozpoznávání vzoru

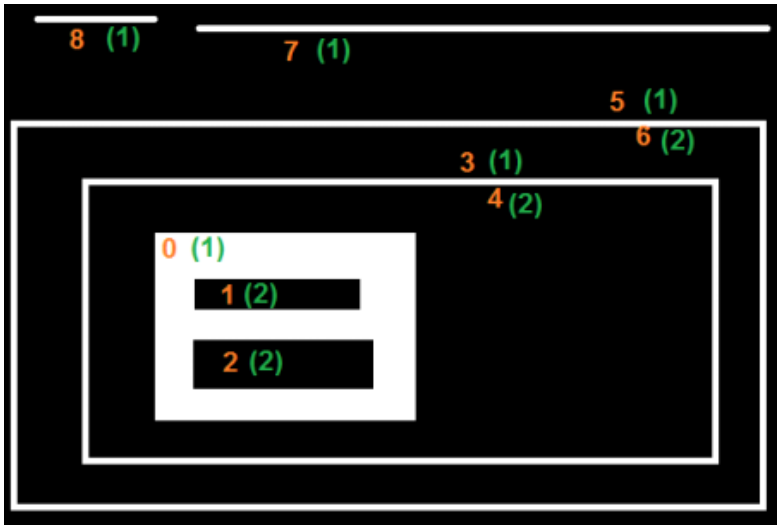
Pro rozpoznávání vizuální značky je vhodné použít algoritmus, který se zakládá na hledání známého vzoru v obraze. Nejběžnější metodou, která se pro této účely používá, je Template Matching [13]. Podstatou této metody je to, že se pro různé velikosti vstupného obrazu počítá jeho korelace se známým vzorem. Kvůli tomu, že vzor může mít v obraze jinou velikost, než je reálná, vytváří se tzv. pyramida obrazů [14]. Tato pyramida se skládá ze vstupních obrazu různé velikosti a orientace. Každý obraz v této pyramidě je rozdělen na oblasti o stejné velikosti, jako je reálná velikost vzoru. Pro každou z těchto oblastí se vypočte korelace se vzorem. Následně je zvolena oblast, kde je korelační koeficient největší.

Kvůli tomu, že korelace musí být spočtena pro velký počet vstupních obrazů o různé velikosti, Template Matching může být pomalý. Jelikož se předpokládá použití čtvercové vizuální značky, je praktičtější nejdříve najít oblast zájmu (Region of Interest, ROI), pro níž se spočte korelace se známým vzorem.



(a) : Před prahováním **(b) :** Po prahování

Obrázek 3.1: Příklad použití adaptivního prahování [2]



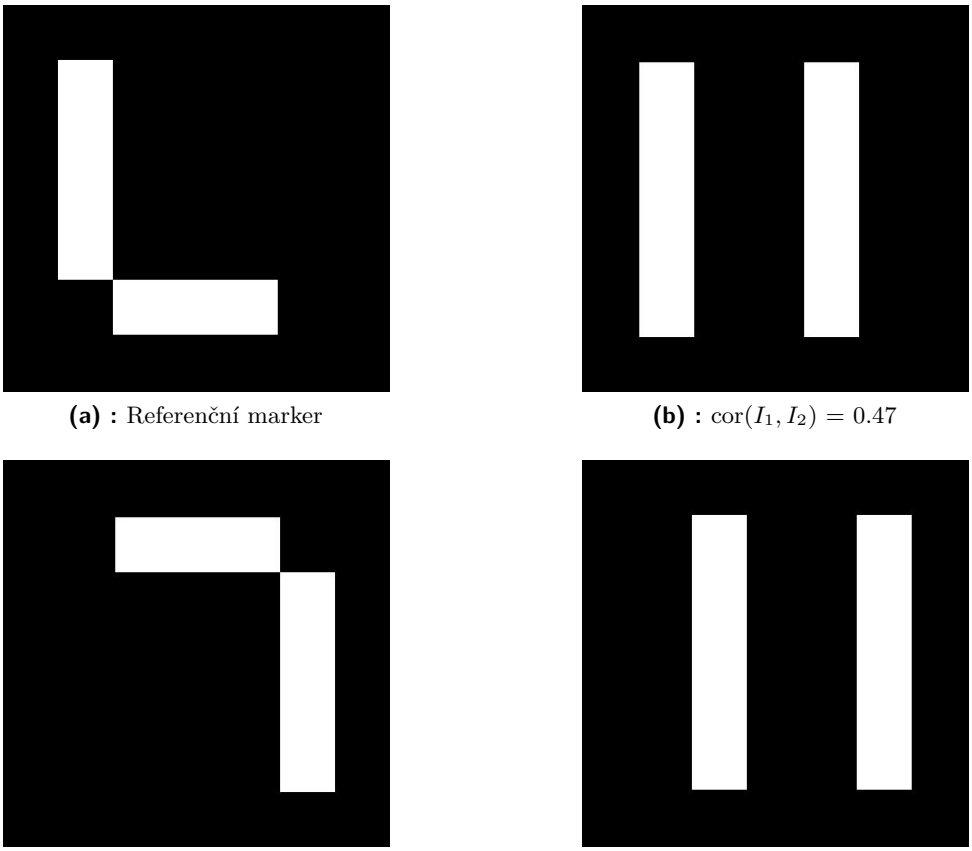
Obrázek 3.2: Hierarchie (topologie) obrazu [3]

Proto pro detekci vizuální značky byl navržen korelační algoritmus, který je popsán v následujícím textu. Pro porovnání funkčnosti tohoto algoritmu byl použit ArUco marker detektor [15], který je k dispozici v knihovně OpenCV.

3.2.1 Korelační algoritmus

Tento algoritmus využívá principu korelaci mezi dvěma veličinami, tedy mezi známým vzorem a nalezenou ROI. Korelace uvádí, jak jsou na sobě tyto veličiny závislé, a může nabývat hodnot od -1 do +1. Hodnota korelačního koeficientu blížící se -1 značí nepřímou závislost veličin, koeficient blízký +1 naopak značí přímou závislost, což je zobrazeno na 3.3.

Hledaná ROI je v zjednodušeném případě konvexní čtyřúhelník, proto stačí s použitím informace o topologii obrazu spočítat polygony nalezených kontur a vybrat pouze ty, co obsahují čtyři strany a nejsou konkávní. Pro aproximaci křivky se používá Ramerův Douglasův Peuckerův algoritmus [16] (viz obr. 3.4), který je již implementován v OpenCV.



(a) : Referenční marker

(b) : $\mathrm{cor}(I_1, I_2) = 0.47$

(c) : $\mathrm{cor}(I_1, I_2) = -0.55$

(d) : $\mathrm{cor}(I_1, I_2) = -0.04$

Obrázek 3.3: Příklad korelace markerů vůči referenčnímu

Následně je spočítána transformační matice mezi nalezeným polygonem a známým vzorem. Pomocí této matice se polygon převede do takové podoby, aby byl porovnatelný se vzorem.

Dále dojde k vyhodnocení korelace nalezené ROI a vzoru. Pokud je vypočtený korelační koeficient větší, než zadaný práh, ROI se vyhodnotí jako správně označena a algoritmus vrátí souřadnice rohů pro následné zpracování.

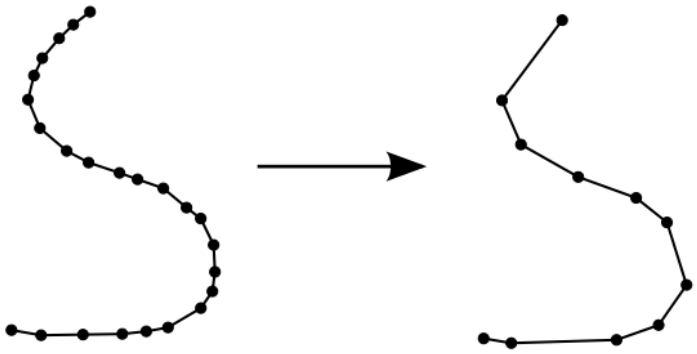
Korelaci mezi dvěma obrázky I_1 a I_2 s N pixely je vypočtena následujícím způsobem:

$$\mu_{1,2} = \frac{\sum_{i,j} I_{i,j}}{N}, \tag{3.1}$$

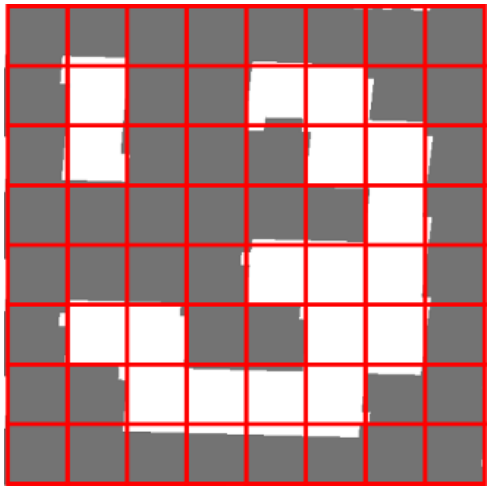
$$\sigma_{1,2} = \sqrt{\frac{\sum_{i,j}(I_{i,j} - \mu_{1,2})^2}{N}}, \tag{3.2}$$

$$covar(I_1, I_2) = \frac{(I_1 - \mu_1) \cdot (I_2 - \mu_2)}{N}, \tag{3.3}$$

$$cor(I_1, I_2) = \frac{covar(I_1, I_2)}{\sigma_1 \sigma_2}. \tag{3.4}$$



Obrázek 3.4: Aproximace křivky pomocí Ramerova Douglasova Peuckerova algoritmu

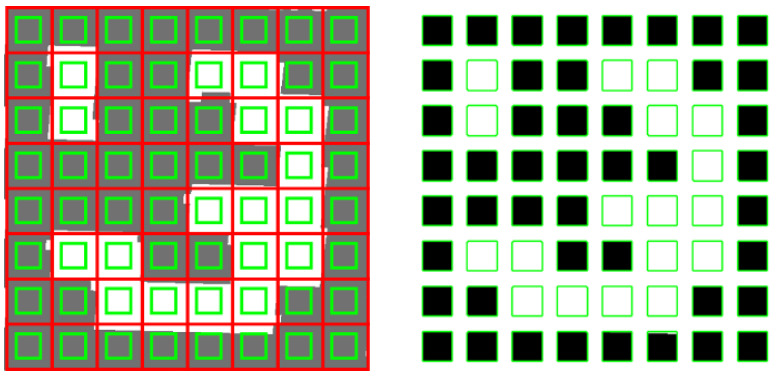


Obrázek 3.5: Buňky ArUco markeru

3.2.2 ArUco marker detektor

Tento algoritmus umožňuje rozpoznávání ArUco markerů nebo podobných vizuálních značek. Podobně jako výše uvedený korelační algoritmus, hledá konvexní čtyřúhelník. Rozdíl pak spočívá v odlišném zpracování nalezené ROI, kde se místo korelace využívá binární mapa kandidáta.

ROI je rozdělena na mřížku s počtem buněk rovným počtu bitů hledaného vzoru (viz obr. 3.5). Následně se spočítá, kolik bílých a černých pixelů obsahuje každá buňka, na základě čehož se vyhodnotí, jestli buňka je černá nebo bílá (viz obr. 3.6). Pokud detektor ve svém slovníku obsahuje vzor se stejnou binární maskou, vrátí souřadnice rohů nalezené ROI.



Obrázek 3.6: Vyhodnocení obrázku pomocí detektoru ArUco

3.3 Měření polohy cíle

Algoritmy popsané výše naleznou polohu vizuální značky v obraze, z čehož lze spočítat její umístění vůči kameře, je-li známa velikost této značky. Pro výpočty lze použít model ideální (dírkové) kamery [17], kde vzdálenost mezi kamerou a značkou je vyjádřena jako

$$d = \frac{fX}{x}, \tag{3.5}$$

kde x značí největší vzdálenost mezi dvěma rohy nalezené značky vyjádřenou v pixelech, f je ohnisková vzdálenost a X je známá délka hrany značky .

Pro nalezení posunutí značky vůči kameře při odklonění od její osy musí být znám zorný úhel kamery. Ten lze spočítat dle vztahu

$$\alpha = 2 \arctan\left(\frac{w}{2f}\right), \tag{3.6}$$

kde w je šířka výstupního obrazu z kamery.

Potom úhel, který pokrývá jeden pixel snímku, je vyjádřen vztahem

$$APP = \frac{\alpha}{w}. \tag{3.7}$$

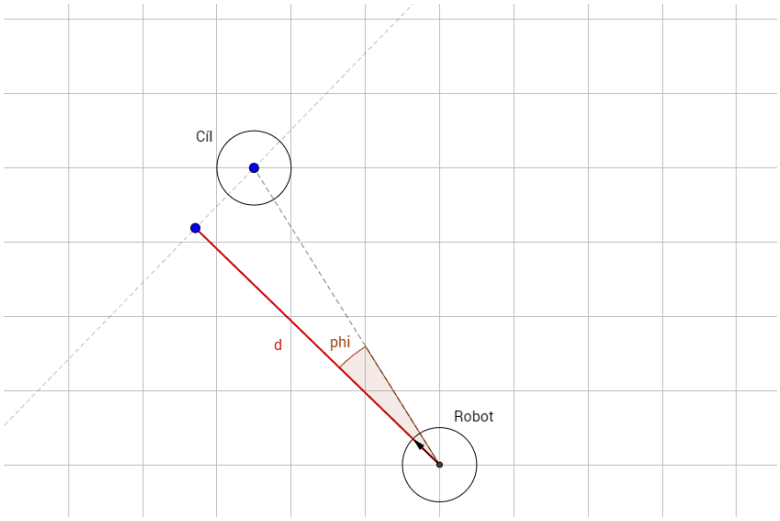
Následně posunutí objektu vůči středu kamery je

$$\phi = (x_c - x_t)APP, \tag{3.8}$$

kde x_c je pixel vyjadřující střed kamery v horizontálním směru, x_t je pixel vyjadřující střed nalezené značky v horizontálním směru.

Ohniskovou vzdálenost lze dostat z kalibrační matice, pokud se provádí úplná kalibrace kamery. S horší, ale vyhovující, přesností lze provést zjednodušenou kalibraci, a to tak, že na určitou vzdálenost d_0 bude před kameru umístěna vizuální značka o velikosti X. Uloží se snímek značky a spočte se x_0. Pak platí, že

$$f = \frac{x_0 d_0}{X}. \tag{3.9}$$



Obrázek 3.7: Měřené veličiny d a ϕ

3.4 Kalmanův filtr

Při sledování člověka může nastat situace, že vizuální značka nebude detekovatelná. Potom není možnost spočítat polohu značky vůči robotu, a tak robot buď zastaví, nebo bude pokračovat pohyb dle posledního měření. Toto může vést k tomu, že se člověk bude muset vrátit k robotu, aby ho znovu začal sledovat. Aby tato situace nenastala, je třeba predikovat polohu cíle z předchozích měření. Pro tyto účely je v práci použit Kalmanův filtr [18].

Model lze popsat pomocí následujícího stavového vektoru:

$$\boldsymbol{x} = \begin{bmatrix} x & y & v_x & v_y \end{bmatrix}^T, \tag{3.10}$$

kde v_x, resp. v_y, značí rychlost ve směru osy X, resp. Y.

Stavový model je pak vyjádřen jako

$$\boldsymbol{x}_{t+1} = \boldsymbol{F}\boldsymbol{x}_t, \tag{3.11}$$

kde $\boldsymbol{F}$ je matice přechodu stavů $\boldsymbol{x}$ z času t do času $t+1$ a platí, že

$$\boldsymbol{F} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & \Delta t & 0 \\ 0 & 0 & 0 & \Delta t \end{bmatrix}, \tag{3.12}$$

kde Δt značí rozdíl mezi časem t a $t+1$.

Pro sledování cíle lze Kalmanův filtr rozdělit do tří kroků:

1. Inicializace ($t = 0$). Během tohoto kroku je nastavena počáteční pozice cíle $\boldsymbol{x}_0$ a výchozí hodnota pro kovarianční matici $\boldsymbol{P}_0$. Jelikož počáteční pozice nemusí být známa, je vhodné zvolit $\boldsymbol{x}_0$ tak, aby v případě, že kamera nedetekuje značku po první iteraci, zůstával robot na místě.

2. Predikce ($t > 0$)**.** V tomto kroku se provádí predikce polohy cíle v čase $t + 1$, tj. $\boldsymbol{x}_{t+1}$, dle 3.11. Také se počítá nová kovarianční matice dle následujícího vztahu:

$$\boldsymbol{P}_{t+1} = \boldsymbol{F}\boldsymbol{P}_t\boldsymbol{F}^T + \boldsymbol{Q}_{t+1}, \tag{3.13}$$

kde $\boldsymbol{Q}$ značí matici kovariancí šumů (výpočet viz [19]).

3. Filtrace ($t > 0$)**.** Během tohoto kroku je poloha cíle upřesněna na základě provedeného měření. Nejdříve se spočte rozdíl mezi reálnou polohou a predikovanou v kroku 2.:

$$\boldsymbol{y}_t = \begin{bmatrix} x_{m_t} \\ y_{m_t} \end{bmatrix} - \boldsymbol{H}\boldsymbol{x}_t, \tag{3.14}$$

kde

$$\boldsymbol{H} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}. \tag{3.15}$$

Dále se vypočítá Kalmanovo zesílení, pro nějž platí:

$$\boldsymbol{K}_t = \boldsymbol{P}_t\boldsymbol{H}^T(\boldsymbol{H}\boldsymbol{P}_t\boldsymbol{H}^T + \boldsymbol{R})^{-1}, \tag{3.16}$$

kde $\boldsymbol{R}$ ja matice kovariancí šumů (výpočet viz [19]).

Následně se provede zpřesnění polohy cíle a aktualizuje se kovarianční matice:

$$\boldsymbol{x}_{t+1} = \boldsymbol{x}_t + \boldsymbol{K}_t\boldsymbol{y}_t, \tag{3.17}$$

$$\boldsymbol{P}_{t+1} = (\boldsymbol{I} - \boldsymbol{K}_t\boldsymbol{H})\boldsymbol{P}_t. \tag{3.18}$$

Celý proces lze shrnout pomocí diagramu 3.8. Z nalezených hodnot x a y lze jednoduše odvodit hodnoty d a ϕ, což je také vidět na obr. 3.9. Jelikož se předpokládá, že poloha robotu je známa, lze spočítat x_t a y_t, což je poloha cíle vůči robotu, a to takto:

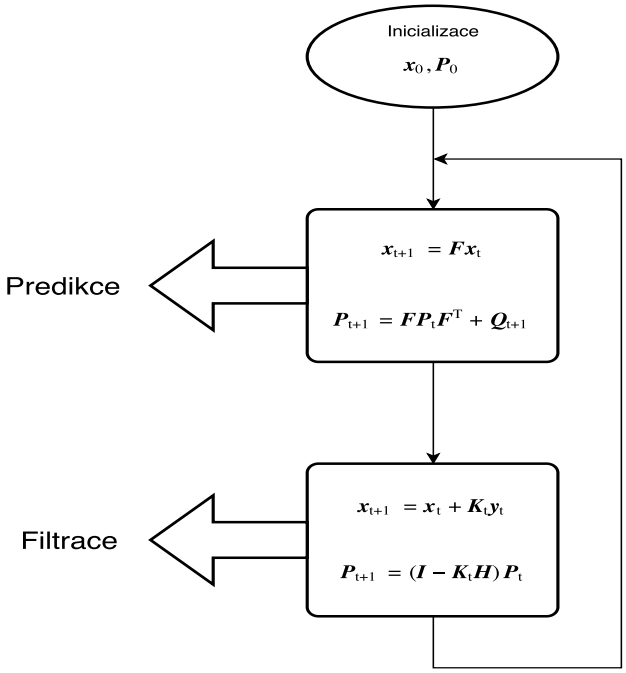
$$x_t = x - x_r, \tag{3.19}$$

$$y_t = y - y_r. \tag{3.20}$$

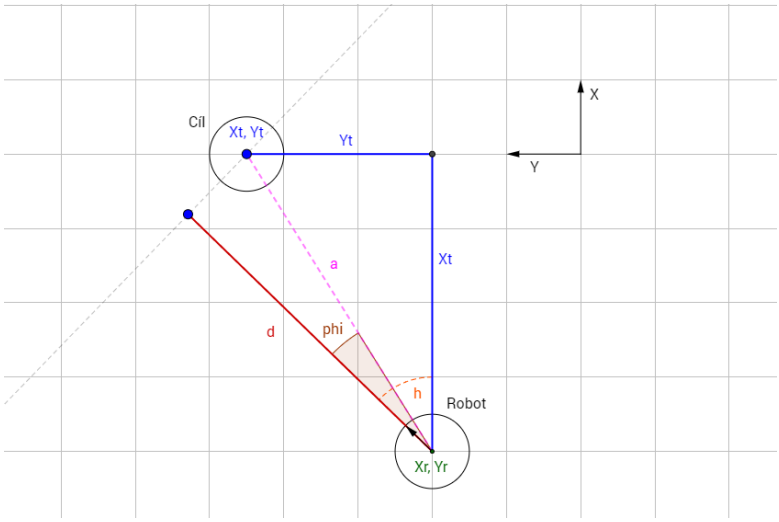
Dále jsou pak ϕ a d nalezeny dle vztahů:

$$\phi = h - \operatorname{atan2}(y_t, x_t), \tag{3.21}$$

$$d = \sqrt{x_t + y_t}\cos(\phi). \tag{3.22}$$



Obrázek 3.8: Shrnutí algoritmu Kalmanova filtru



Obrázek 3.9: Odvození hodnot d a ϕ ze známé polohy cíle

Kapitola 4
Řízení robotu

4.1 Vyhýbání se překážkám

Pro řízení autonomního robotu je důležité, aby se během sledování cíle dokázal vyhýbat překážkám. Algoritmů pro řízení robotu v prostředí s překážkami je několik [4]. Stručný přehled těchto algoritmů je k nalezení v tabulce 4.1.

Omezením na výběr algoritmu pro vyhýbání se překážkám je to, že se robot bude pohybovat v prostředí, které není dopředu známo. Proto je výhodnější použít algoritmus, který pracuje s lokální mapou prostředí vytvořenou kolem robotu místo globální. Navíc při použití vztažné soustavy a diferenciálního výpočtu polohy a rychlosti robotu nelze přesně stanovit, kde se nachází ten nebo překážky kolem něj. Proto je třeba použít algoritmus, který by pracoval pouze se základní kinematikou robotu a prostředí.

S uvážením těchto omezení jsou nejvhodnějšími algoritmy pro řešení zadaného problému algoritmus Bug [4, s. 272–276] nebo Vector Field Histogram [4, s. 276–278][20].

Metoda	Kinematika	Pohled	Mapa	Senzor
Bug	Žádná	Lokální	Žádná Lokální graf tečen	Dotykový Dálkový
VFH	Základní	Lokální	Mřížový histogram	Dálkový
Bubble band	Přesná	Lokální Globální	Globální polygon	Různé
Curvature velocity	Přesná	Lokální	Mřížový histogram	Dálkový
Dynamic window	Přesná	Globální	Obstacle line field	Dálkový
Nearness diagram	Holonomní	Globální	Žádná	Dálkový
Gradient method	Přesná	Globální	Žádná	Dálkový

Tabulka 4.1: Souhrn základních algoritmů pro vyhýbání se překážkám [4, s. 287–290]

4.1.1 Algoritmus Bug

Tento algoritmus je nejjednodušším algoritmem pro vyhýbání se překážkám. Podstatou algoritmu je to, že robot narazí na překážku a následuje její konturu. Existuje několik obdob toho algoritmu, které se liší v tom, jak je obvod překážky sledován robotem. Například algoritmus Bug1 nejprve s použitím dotykového senzoru objede celou překážku, následně zvolí bod, který je nejblíže cíli, vrátí se tam a pokračuje směrem k cíli. Tento postup je dost neefektivní, ale zajistí, že robot vždy dosáhne cíle.

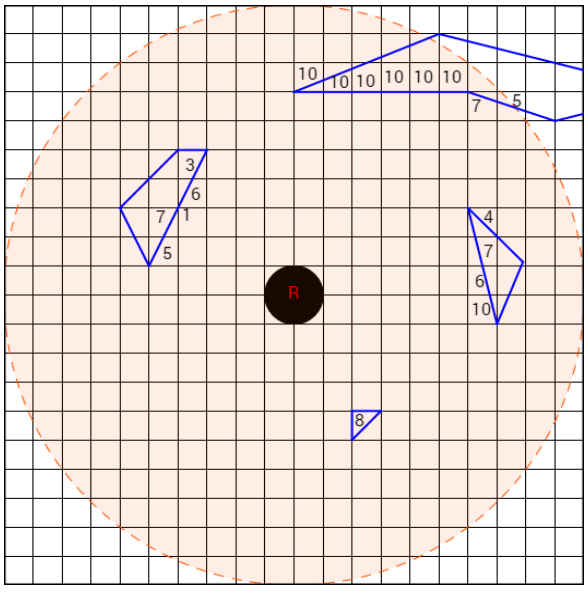
Možným vylepšením tohoto algoritmu je Tangent Bug, který používá dálkoměr pro měření vzdálenosti k překážkám a vytváří model lokálního prostředí reprezentovaný graf.

Nevýhodou algoritmu je to, že když nalezne jakoukoliv překážku, snaží se sledovat její konturu, nikoliv se jí přímo vyhnout. Proto se při následování cíle může stát, že kvůli sledování obrysu překážky dojde ke ztratě cíle ze zorného pole a již nebude možné odhadnout jeho pozici.

4.1.2 Vector Field Histogram (VFH)

S uvážením výše zmíněních omezení a nevýhod algoritmu Bug je v práci použit Vector Field Histogram, který je navržen tak, aby se robot snažil překážky objíždět dříve, než bude v nebezpečné zóně blízko k překážkám.

Princip tohoto algoritmu spočívá v tom, že se na základě dat naměřených dálkoměrem vytvoří lokální mřížková mapa prostředí o poloměru r_{map} a se zvoleným rozlišením. Každá buňka této mapy tak obsahuje hodnotu obsazenosti odpovídající oblasti v reálném světě (viz obr. 4.1). Z této mapy se následně spočítá polární histogram, podle kterého se určí směr jízdy robotu.



Obrázek 4.1: Lokální mapa prostředí o poloměru $r_{map} = 10$

Celý proces lze rozdělit na tři kroky: vytvoření primárního polárního histo-

gramu, prahování primárního histogramu (vytvoření binární reprezentace), a nakonec výběr kandidátů pro směr pohybu.

Primární polární histogram

Polární histogram je rozdělen na sektory tak, aby každý z nich odpovídal úhlu α, který je volen takovým způsobem, aby $\frac{360}{\alpha}$ bylo celé číslo. Tedy například pro $\alpha = 15°$ histogram obsahuje 24 sektorů (viz obr. 4.2).

Pro každou buňku aktivního regionu, tedy lokální mapy vytvořené kolem robotu, se spočte směr, ve kterém se vůči středu nachází, a její význam. Směr je spočítán dle vztahu:

$$\beta_{i,j} = \text{atan2}(y_o - y_i, x_o - x_i), \tag{4.1}$$

kde x_o, y_o značí souřadnice středu mapy, x_i, y_i jsou souřadnice buňky.

Dále pak významnost buňky je:

$$m_{i,j} = c_{i,j}^2(a - bd_{i,j}^2), \tag{4.2}$$

kde $c_{i,j}$ je obsazenost buňky, a $d_{i,j}$ je vzdálenost buňky od pozice robotu. Parametry a a b jsou voleny dle vztahu:

$$a - b\left(\frac{r_{map} - 1}{2}\right)^2 = 1. \tag{4.3}$$

Aby robot nejel blízko k okrajům překážek, zavádí se kompenzace jeho velikosti pomocí poloměru robotu r_{rob} a minimální povolené vzdálenosti mezi robotem a překážkou d_{safety}:

$$\gamma_{i,j} = \arcsin\left(\frac{r_{rob} + d_{safety}}{d_{i,j}}\right).$$

Histogram je potom spočten vztahem:

$$H_k^p = \sum_{i,j \in C_\alpha} m_{i,j} h_{i,j}, \tag{4.4}$$

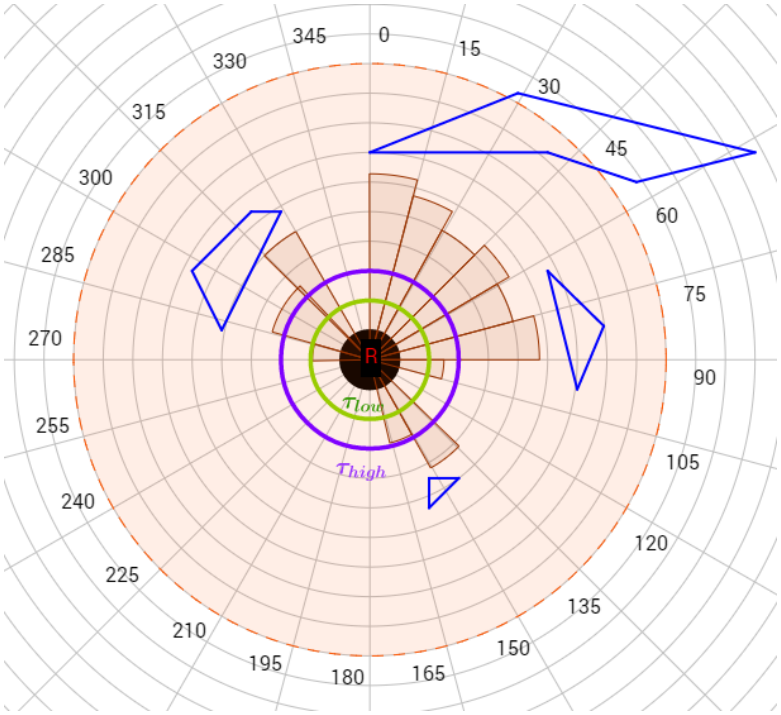
kde

$$h_{i,j} = \begin{cases} 1 & \text{jestli } k\alpha \in [\beta_{i,j} - \gamma_{i,j}; \beta_{i,j} + \gamma_{i,j}], \\ 0 & \text{jinak.} \end{cases} \tag{4.5}$$

Binární polární histogram

Aby bylo možné určit kandidáty pro další směr pohybu, primární histogram musí být převeden do binární podoby, a to takto:

$$H_{k,i}^b = \begin{cases} 1 & \text{jestli } H_{k,i}^p > \tau_{high}, \\ 0 & \text{jestli } H_{k,i}^p < \tau_{low}, \\ H_{k,i-1}^b & \text{jinak.} \end{cases} \tag{4.6}$$



Obrázek 4.2: Polární histogram s vyobrazenými prahy τ_{low} (zeleně) a τ_{high} (fialově)

Výběr kandidátů

Kandidáti pro nový směr pohybu se volí dle toho, do jaké kategorie je zařazeno volné místo v binárním histogramu. Ty průjezdy, které mají velikost (tedy vzdálenost mezi pravým okrajem k_r a levým k_l) menší, než s_{max}, se nazývají úzké, jiné jsou naopak nazývány široké.

Pro úzké průjezdy lze zvolit pouze jednoho kandidáta, a to:

$$c_n = \frac{k_r + k_l}{2} \quad \text{centrální sektor.} \tag{4.7}$$

Široké průjezdy mají tři možné kandidáty:

$$\begin{aligned} c_r &= k_r + \tfrac{s_{max}}{2} && \text{pravý sektor,} \\ c_l &= k_l - \tfrac{s_{max}}{2} && \text{levý sektor,} \\ c_t &= k_t && \text{pokud } k_t \in [c_r; c_l]. \end{aligned} \tag{4.8}$$

Nový směr pohybu se zvolí dle minimální ceny spočítané pro kandidáta c_i pomocí vztahu [21]:

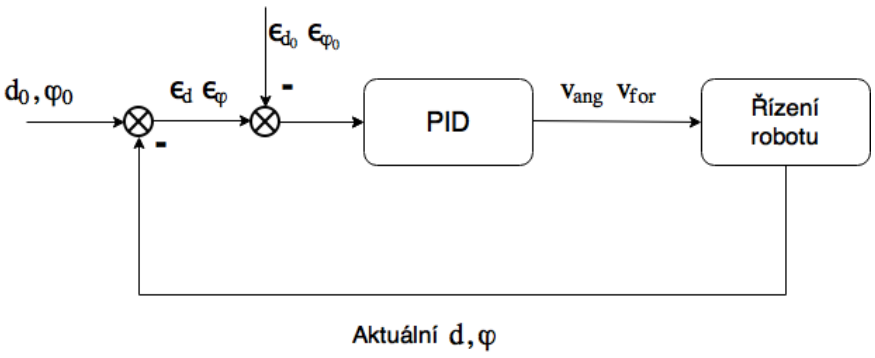
$$g(c_i) = \mu_1 \Delta\left(c_i, k_t\right) + \mu_2 \Delta\left(c_i, \frac{h}{\alpha}\right) + \mu_3 \Delta\left(c_i, k_{d,n-1}\right) \tag{4.9}$$

přičemž $k_{d,n-1}$ je minule zvolený kandidát, μ_1, μ_2, μ_3 jsou kladné konstanty zvolené dle vztahu $\mu_1 > \mu_2 + \mu_3$, a také platí, že

$$\Delta(c_1, c_2) = \min\left\{|c_1 - c_2|, |c_1 - c_2 - \frac{360°}{\alpha}|, |c_1 - c_2 + \frac{360°}{\alpha}|\right\}. \tag{4.10}$$

4.2 PID regulátor

Za jízdy je třeba, aby byla mezi robotem a cílem udržována určitá vzdálenost d a hodnota úhlu ϕ byla co nejmenší, nejlépe nulová. Pro tyto účely je navržen diskrétní PID regulátor [22], který pro aktuální odchylku od referenčních bodů spočte dopřednou a úhlovou rychlost, které se následně převedou na rychlosti motorů. Zjednodušený nákres algoritmu je zobrazen na 4.3. Pokud je ϵ_d, resp. ϵ_ϕ, v pásmu $[-\epsilon_{d_0}; \epsilon_{d_0}]$, resp. $[-\epsilon_{\phi_0}; \epsilon_{\phi_0}]$, pak bude výstup PID regulátoru nulový. Jinak je od odchylky odečtena hodnota ϵ_{d_0}, resp. ϵ_{ϕ_0}, což zaručí to, že robot nebude kmitat na místě, pokud bude blízko referenčního bodu, a řízení bude plynulé.



Obrázek 4.3: PID regulace

Výsledná dopřední, reps. úhlová, rychlost v čase t je vypočtena pomocí vztahu:

$$V_{for,ang} = K_p E + K_i I + K_d D, \tag{4.11}$$

kde K_p, K_i a K_d jsou koeficienty proporcionální, integrální a diferenciální složky PID regulátoru, a pro E, I, D platí, že

$$E = \begin{cases} \epsilon - \epsilon_0 & \text{jestli } |\epsilon| > \epsilon_0, \\ 0 & \text{jinak.} \end{cases} \tag{4.12}$$

$$I = \sum_t E_t \Delta t, \tag{4.13}$$

$$D = \frac{E_t - E_{t-1}}{\Delta t}. \tag{4.14}$$

Anti windup

Jelikož integrální složka pořád sčítá chybu, může dojít k tomu, že výstup PID regulátoru bude přesahovat maximální povolenou rychlost robotu, a tudíž ztrácí schopnost ji regulovat. Tomuto jevu se říká integrální windup. Aby

nedocházelo k nekontrolovanému výpočtu integrální složky, je v práci použita technika anti windup. Ta zajistí to, že integrál v čase $t+1$ bude roven

$$I_{t+1} = \begin{cases} I_t + K_a E \Delta t & \text{jestli } |V| > V_{max}, \\ I_t + E\Delta t & \text{jinak}, \end{cases} \tag{4.15}$$

kde K_a je anti windup koeficient. V případě, že K_a bude roven 0, integrální složka se nezmění. Naopak pokud K_a je 1, anti windup bude ignorován.

Volba koeficientů

Při volbě koeficientů PID regulátoru je třeba mít na mysli vlastnosti každé složky (viz tabulka 4.2). Pro ladění těchto koeficientů lze použít např. Zieglerovu – Nicholsovu metodu [23]. Princip metody spočívá v tom, že se nalezne kritické zesílení K_{p_c}, při kterém systém, v případě robotu – dopředí nebo úhlová rychlost, začne kmitat a zjistí se doba jedné oscilace P_c pro toto zesílení. Na základě těchto údajů se spočtou koeficienty složek použitého PID regulátoru (viz tabulka 4.3).

Koeficient	Doba náběhu	Overshoot	Doba ustálení	Chyba v ekvilibriu
K_p	Snižuje	Zvyšuje	Malá změna	Snižuje
K_i	Snižuje	Zvyšuje	Zvyšuje	Eliminuje
K_d	Malá změna	Snižuje	Snižuje	Žádná

Tabulka 4.2: Vliv zvětšení koeficientů PID regulátoru na kvalitu regulace

K ladění lze přistupovat i empiricky. Když je známa maximální rychlost, se kterou se robot může pohybovat, lze přibližně odhadnout parametr K_p. Mělo by být zajištěno, že se robot bude pohybovat rychlostí, která je přímo úměrná chybě. Pokud je tedy očekávána maximální rychlost V_{max} a maximální odchylka od referenčního bodu je E_{max}, pak by mělo platit, že

$$K_p = \frac{V_{max}}{E_{max}}. \tag{4.16}$$

Dále parametry by se měly odladit tak, aby nedocházelo ke zbytečnému kmitání robotu, a odezva na prudké změny polohy cíle byla co nejmenší.

Regulátor	K_p	K_i	K_d
P	$0.5K_{p_c}$	0	0
PI	$0.45K_{p_c}$	$1.2\frac{K_{p_c}\Delta t}{P_c}$	0
PD	$0.8K_{p_c}$	0	$K_{p_c}\frac{P_c}{8\Delta t}$
PID	$0.6K_{p_c}$	$2\frac{K_{p_c}\Delta t}{P_c}$	$K_{p_c}\frac{P_c}{8\Delta t}$

Tabulka 4.3: Nastavení koeficientů PID regulátoru pomocí Zieglerovy Nicholsovy metody

Kapitola 5

Simulace

Pro simulaci byl zvolen simulátor V-Rep [24]. V tomto prostředí byly vytvořeny dva roboty: jeden s vizuální značkou, druhý vybavený kamerou a laserovým dálkoměrem sledoval prvního (viz obr. 5.1). Robot s vizuální značkou je řízen uživatelem pomocí klávesnice. Referenční vzdálenost mezi dvěma roboty byla zvolena 1 m, požadovaná odchylka vizuální značky od středu kamery 0 rad. Maximální dopředná rychlost obou robotů byla nastavena na $2.5\,\mathrm{m\,s^{-1}}$.

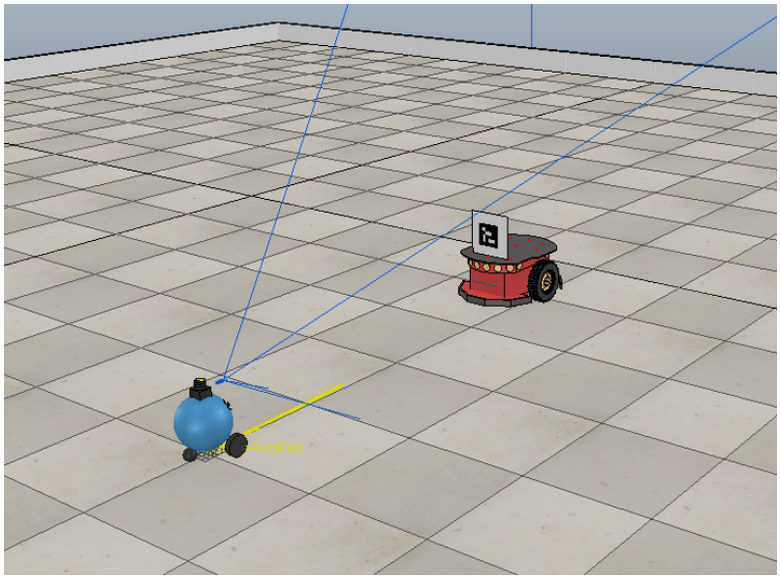
5.1 Porovnání algoritmů detekce

Oba algoritmy byly porovnávány při použití značky o velikosti $0.1\,\mathrm{m} \times 0.1\,\mathrm{m}$ a rozlišením kamery 512 px × 512 px Bylo porovnáno několik parametrů, a to maximální vzdálenost, na které je vizuální značka ještě detekovatelná, stabilní vzdálenost, při které je značka detekována bez přerušení. Dále byl porovnán maximální a stabilní úhel otočení vizuální značky vůči robotu a také průměrný výpočetní čas na jednu iteraci algoritmu. Výsledky jsou znázorněny v tabulce 5.1. Je patrné, že detekční schopnosti korelačního algoritmu jsou poněkud horší, než u ArUco detektoru. Avšak je třeba zdůraznit, že ArUco detektor sice dokázal rozpoznat vizuální značku na vzdálenosti přes 5 metrů od kamery, úspěšnost detekcí na tak velké vzdálenosti byla podprůměrná, přibližně 40 %. Co se týká úhlu otočení značky vůči robotu, tam se korelační algoritmus ukázal být lepší a stabilně detekuje značku při 60° otočení. ArUco má na hranicích detekovatelnosti, tj. při 70° otočení, pouze 15% úspěšnost.

Název algoritmu	Korelační algoritmus	ArUco detektor
Max. vzdálenost [m]	3.5	5.4
Stabilní vzdálenost [m]	3.5	3.8
Max. úhel [°]	60	70
Stabilní úhel [°]	60	55
Výpočetní doba [ms]	15	15

Tabulka 5.1: Porovnání korelačního algoritmu a ArUco detektoru

Hlavní rozdíl mezi těmito algoritmy spočívá v možnostech jejich využití. Zatímco ArUco detektor je zaměřený pouze na ArUco markery, korelační



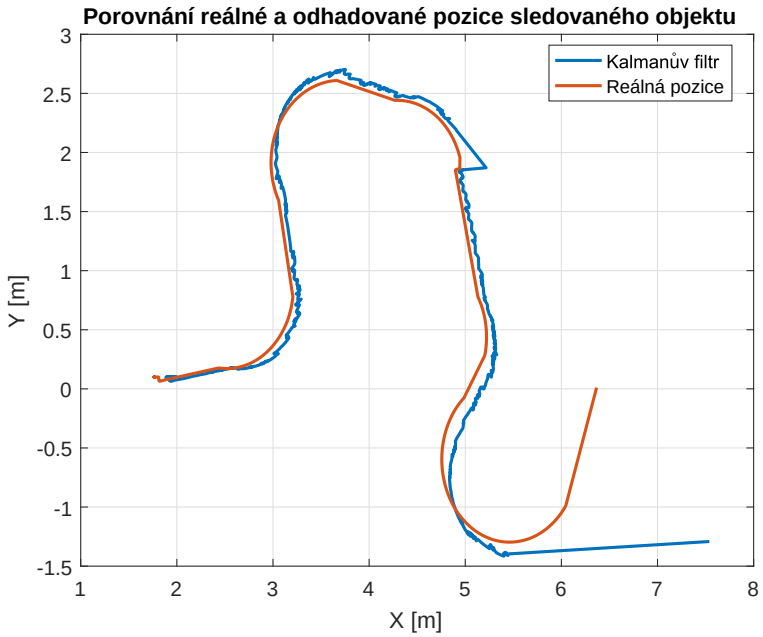
Obrázek 5.1: Roboty vytvořené v simulátoru V-Rep

algoritmus je obecnější a dá se použít na jakýkoliv typ vizuální značky. ArUco detektor umožňuje přidání dalších značek do své knihovny, ale jedná se pouze o binární obrázky podobné tomu na obr. 2.1. Korelační algoritmus by měl teoreticky detekovat i značku, kterou nelze reprezentovat jako bit-mapu, viz např. obr. 5.2.

Obrázek 5.2: Příklad značky pro korelační algoritmus

5.2 Porovnání naměřených a reálných hodnot

Jelikož se v implementovaném programu provádí různé výpočty, je třeba porovnat, jakou mají tyto hodnoty odchylku vůči reálným. Kalmanův filtr, popsaný v 3.4, predikuje polohu cíle a v případě neúspěšné detekce vizuální



Obrázek 5.3: Poloha sledovaného objektu během simulace

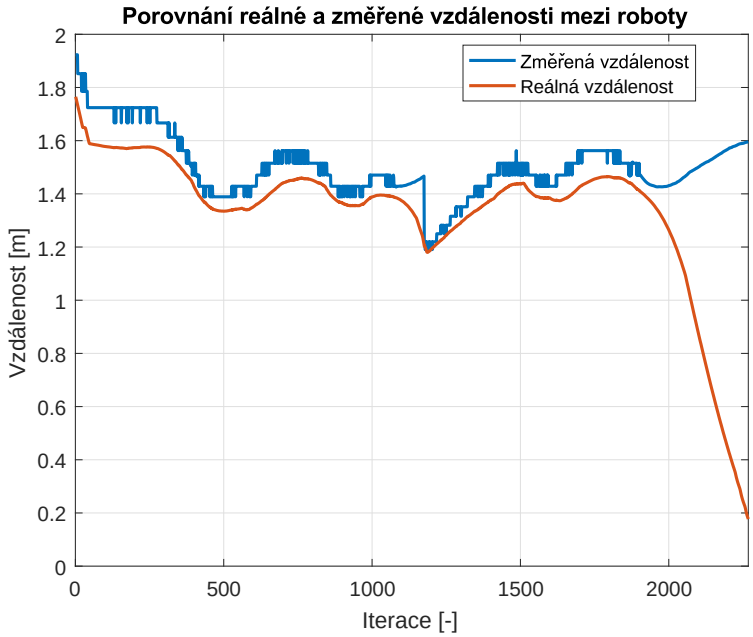
značky je umístění sledovaného objektu vůči robotu odvozeno právě z predikované polohy. Na obrázku 5.3 lze vidět, že výpočet pomocí Kalmanova filtru skoro po celou dobu simulace odpovídá reálné poloze sledovaného objektu. Velká odchylka pak je až na konci simulace, kdy vizuální značka brzo zmizela ze zorného pole robotu a nebylo ji možné v krátké době najít. Toto také odpovídá výsledkům, které jsou na 5.4 a 5.5, kde je vidět, že po cca 2000. iteraci je rozdíl mezi změřenými a reálnými hodnotami výrazný.

Na obrázku 5.4 je také vidět, že změřená vzdálenost není stejná, jako reálná. Je to způsobeno nepřesnou kalibrací referenční vzdálenosti mezi následovaným a sledujícím robotem. Kdyby kalibrace byla přesnější, bylo by možné dosáhnout podobného výsledku, jako pro odchylku od středu kamery (viz obr. 5.5), kde měření odpovídá reálné hodnotě.

5.3 Měření polohy robotu

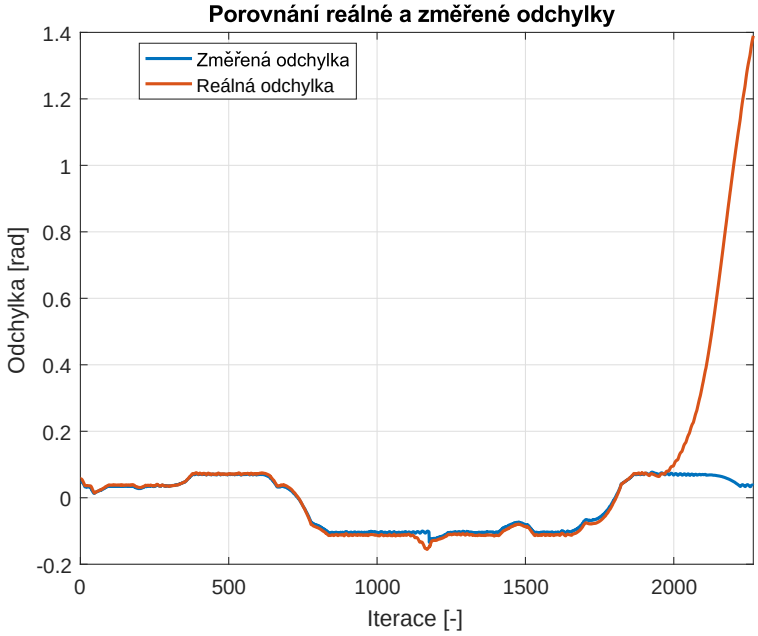
Jak bylo řečeno v 3.4, aby bylo možné spočítat polohu cíle vůči robotu, je třeba znát polohu robotu v globálních souřadnicích. Ačkoliv simulátor V-Rep nabízí možnosti pro přesné měření polohy objektů ve světových souřadnicích, v reálném světě tato možnost nemusí být použitelná, protože např. při navigaci robotu uvnitř budov nelze použít GPS. Proto byl v sekci 2.2 zaveden relativní souřadnicový systém, ve kterém se poloha robotu počítá. Pro toto měření lze v simulátoru odečítat vnitřní polohu rotačních kloubu robotu, na které jsou přidělány kola, a pomocí metody popsané v sekci 2.3 spočítat pozici řízeného robotu.

Porovnání reálné, tj. změřené v simulátoru, a vypočtené polohy robotu ve

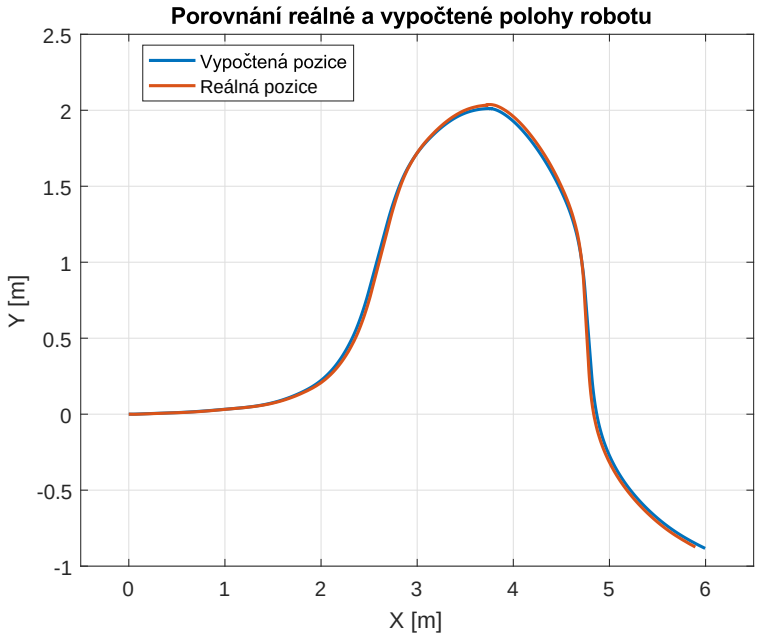


Obrázek 5.4: Vzdálenost mezi robotem a sledovaným objektem (d)

zvoleném souřadnicovém systému lze vidět na obrázku 5.6. Je patrné, že pro použitý typ robotu a relativní souřadnicovou soustavu je výpočet přesný a při použití přesných enkodérů lze tento výpočet aplikovat i na reálném robotu.



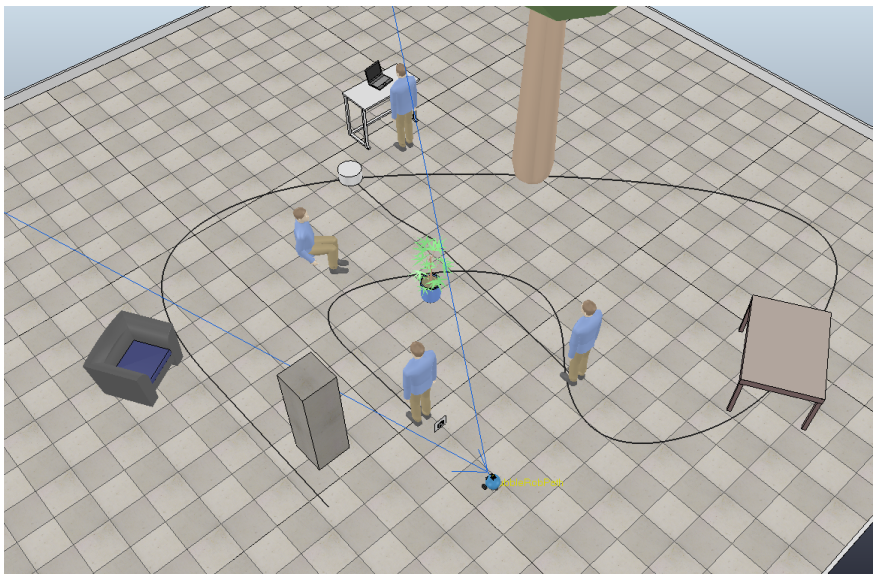
Obrázek 5.5: Odchylka sledovaného objektu od středu kamery ϕ



Obrázek 5.6: Poloha robotu během simulace

5.4 Trajektorie robotu při jízdě mezi překážkami

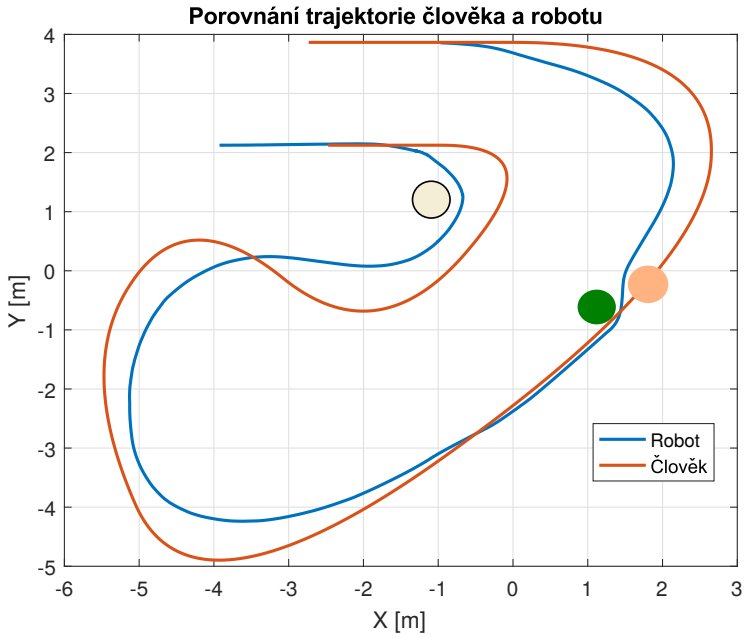
Po ověření funkčnosti detektorů a řízení robotu byl v simulátoru vytvořen model prostředí, ve kterém by robot měl sledovat člověka se značkou. Do scény byly umístěny i jiné osoby, a také různé překážky: květina, stůl, strom, křeslo, sloup, jak je vidět na 5.7.



Obrázek 5.7: Simulace s člověkem a překážkami

Člověk měl předem definovanou cestu, kterou přesně dodržoval nehledě na překážky. Robot se ostatním objektům měl vyhýbat.

Trajektorie člověka a robotu jsou na obrázku 5.8. Šedě je na obrázku znázorněna květina, zeleně strom a růžově kolemjdoucí, který během simulace přišel na uvedenou pozici. Jak je vidět, robot se snažil dodržovat trajektorii, podle které šel sledovaný člověk a přitom dokázal objíždět překážky, které mu v tom bránily. Nejproblematičtější část simulace byla na začátku, kdy člověk strmě zahnul doprava. V ten okamžik byla vizuální značka špatně detekovatelná, protože byla otočena vůči robotu téměř o 90°. Navíc byla v určitém časovém intervalu značka skryta za listím, takže ji použitý algoritmus nedokázal najít ve snímku. Zde je ovšem vidět, jak přínosné je použití Kalmanova filtru, který dokázal předpovědět na základě mála měření, kde se člověk nachází, proto ho robot rychle našel a sledoval i nadále.



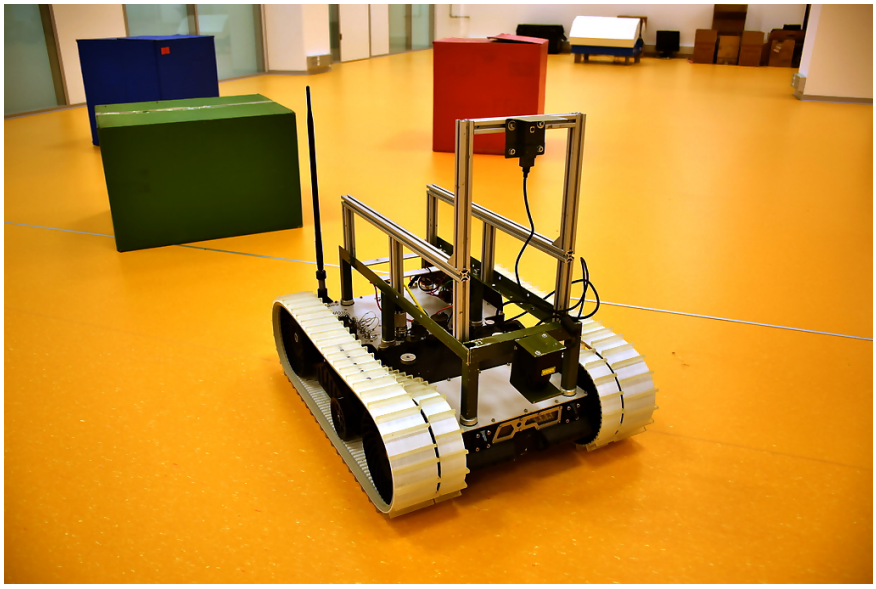
Obrázek 5.8: Trajektorie robotu a člověka

Dalším problematickým úsekem byla cesta mezi stromem a kolemjdoucím. Jelikož sledovaný člověk mohl procházet skrz objekty, došlo k tomu, že, když mu v cestě stal kolemjdoucí, prošel skrz něj a pokračoval dále. Značka tedy opět nebyla detekovatelná. Navíc byl mezi stromem a kolemjdoucím velice úzký průjezd a robot musel snížit rychlost, aby tam dokázal projet. Nakonec však robot nalezl sledovaného člověka a jel za nim až do konce simulace.

5.5 Reálný robot

Po ověření funkčnosti programu v simulátoru bylo možné testovat na reálném robotu. Pro tyto účely byl použit pásový robot, na kterém byla namontována kamera a laserový dálkoměr (obr. 5.9). Komunikace mezi robotem a počítačem

byla řešena pomocí WiFi, z robotu se odečítala jeho odometrie, a pomocí zprav byly nastavovány očekávané dopřední a úhlová rychlosti.



Obrázek 5.9: Použitý pásový robot

Pro algoritmus VFH bylo třeba správně nastavit parametry robotu a vytvářeného histogramu. Jako poloměr robotu r_{rob} byla zvolena polovina jeho vzdálenosti mezi koly, tedy 0.225 m. Jelikož robot nemá kruhový tvar, bylo třeba nastavit poměrně velkou ochrannou vzdálenost d_{safety}, aby nedocházelo ke kolizím zadní části robotu s překážkami. Minimální hodnota d_{safety} by měla odpovídat rozdílu mezi šířkou a délkou robotu, tedy konkrétně byla pro použitého robota hodnota d_{safety} nastavena na 0.2 m.

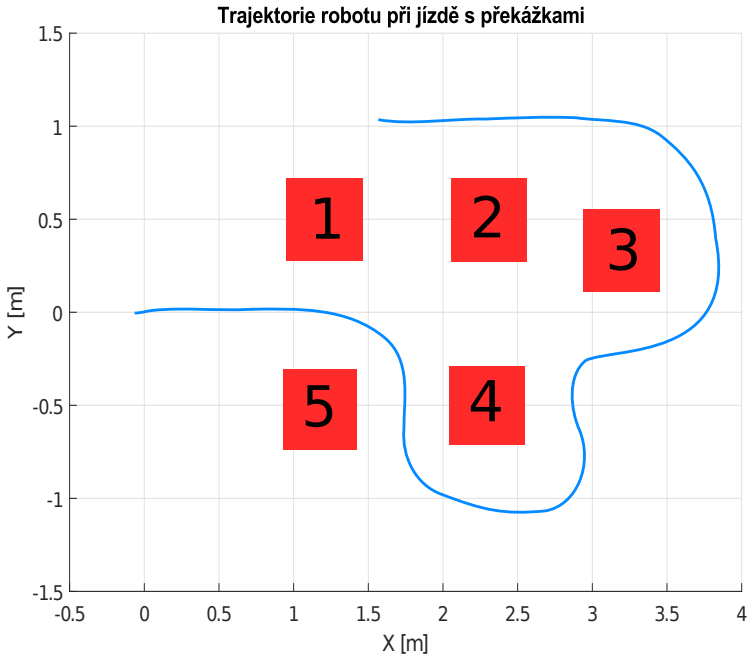
Šířka jednoho intervalu polárního histogramu α byla rovna 3.5×10^{-2} rad, neboli 2°, což znamená, že polární histogram byl rozdělen do 180 úseků. Kolem robotu byla vytvořena mapa o poloměru 2 m, která byla reprezentována jako mřížková mapa o velikosti 81 × 81 s rozlišením 5×10^{-2} m.

Bylo provedeno několik experimentů s různě umístěnými překážkami. Pro ověření funkčnosti detektoru a řízení byly vytvořeny situace, kdy značka zmizela za překážkou nebo byly překážky umístěny tak, aby mezi nimi prošel člověk, ale robot by to kvůli své velikosti nedokázal. Ve většině experimentů robot dokázal najít cestu k cíli nehledě na problematické úseky.

Na obrázcích 5.10 a 5.11 jsou zobrazeny trajektorie robotu při objíždění překážek. Na prvním obrázku je vidět, jak robot objíždí překážku číslo 3. Během tohoto experimentu člověk prošel mezi 2. a 3. překážkou, kde byl úzký průjezd. Ačkoliv algoritmus VFH tento průjezd vyhodnotí jako jeden z možných kandidátu, také počítá s velikostí robotu, proto se ho pokusí nasměrovat tam, kam by bez problémů projel. I když by průjezd mezi 2. a 4. a následně mezi 1. a 2. překážkou byl lepší z hlediska sledování cíle, mezi překážkami by robot neměl dostatečně místo pro otáčení. Proto VFH rozhodne nasměrovat robota tak, aby objel 3. překážku. I když zde došlo

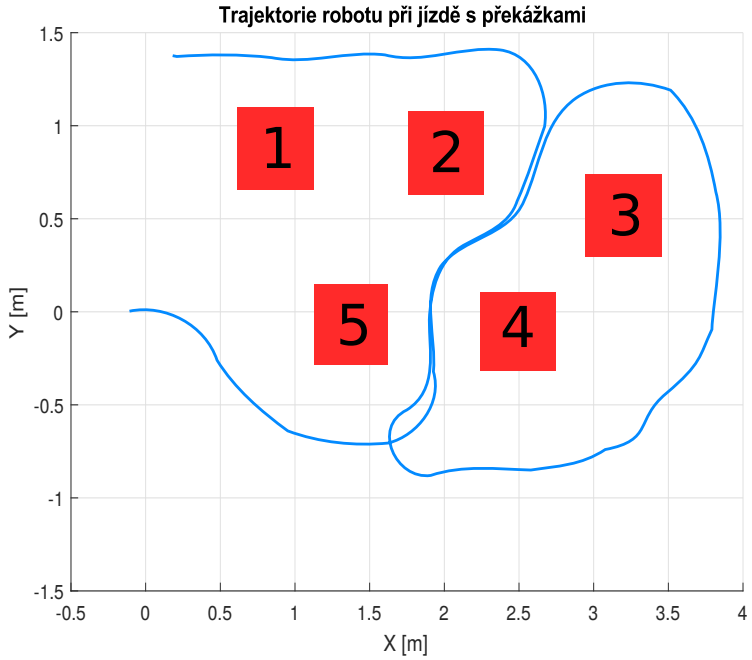
ke krátkodobé ztrátě cíle ze zorného pole robotu, Kalmanův filtr dokázal správně předpovědět pozici člověka, a tak ho robot znovu našel a sledoval, dokud nebyl vypnut.

Průběh dalšího experimentu je znázorněn na druhém obrázku. Zde je vidět, že robot zvládne jezdit i v úzkých prostorech, dokud nemá potřebu prudce změnit směr jízdy. Nicméně během tohoto pokusu došlo k selhání řízení kvůli omezením kinematiky robotu. Po tom, co robot úspěšně projel mezi 2. a 3. překážkou a zatočil doleva, člověk prošel mezi 1. a 2. překážkou. Ačkoliv průjezd mezi nimi byl dostatečně velký, aby tam robot dokázal projet, prudká změna směru pohybu člověka a nedostatek prostoru způsobily to, že robot pokračoval v jízdě podél překážek a nezahnul směrem k člověku. V tomto případě byl robot uvězněn mezi 2. překážkou a zdí, která byla od něj vpravo. Špatně zde také zareagoval Kalmanův filtr, který poměrně dobře odhadl polohu cíle podél osy Y, ale nedokázal správně spočítat souřadnici X.



Obrázek 5.10: Objíždění překážek robotem

Během většiny experimentů chování robotu odpovídalo očekávanému. Při správně nastavených parametrech algoritmu VFH se robot dokázal vyhýbat překážkám a pokud mohl, udržoval k nim bezpečnou vzdálenost. Problematické situace nastávaly, když robot dlouhodobě nedokázal najít značku. Tehdy Kalmanův filtr nepredikoval polohu cíle správně, proto bylo třeba se před robota vrátit.



Obrázek 5.11: Objíždění překážek robotem

Kapitola 6

Závěr

Cílem této práce bylo navrhnout software pro autonomní řízení robotu následujícího člověka.

Pro detekci člověka v obrazu bylo v kapitole 2 rozhodnuto použit vizuální značku, která byla robotem známa, funkčnost takového řešení byla ověřena simulací. Robot dokáže spočítat vzdálenost a směr, ve kterém se detekovaná značka vůči němu nachází, což je popsáno v sekci 3.3. Díky dálkoměru robot také ví, jestli na cestě k cíli jsou překážky. Pokud ano, pomocí Vector Field Histogram algoritmu (viz sekci 4.1) je vypočten nový směr jízdy tak, aby robot nenarazil na žádnou překážku a zároveň byl co nejblíže člověku. Na základě spočtených parametrů se pomocí PID regulátoru, jenž je popsán v sekci 4.2, vyhodnotí rychlosti, které je třeba aplikovat na motory, aby robot dosáhl cílové pozice vůči sledovanému člověku. Navíc pro případ, kdy robot nedetekuje značku, a tedy ani nedokáže spočítat vzdálenost k cíli, je použit Kalmanův filtr (viz sekci 3.4), který předpovídá polohu cíle na základě předchozích měření.

V simulátoru bylo provedeno několik testů s roboty, nastavení simulací a výsledky testů jsou popsány v kapitole 5. Průměrná vzdálenost d mezi roboty byla 0.95 m a odchylka sledovaného robotu od středu kamery ϕ byla stanovena 2.3×10^{-3} rad. Dále byl spočítán rozdíl mezi reálnou polohou a vypočtenou pomocí Kalmanova filtru sledovaného robotu, ten byl stanoven 0.32 m. Tak velká odchylka byla způsobena tím, že byly simulovány situace, kdy značka zmizí ze zorného pole robotu. Nicméně robot dokázal pokaždé najít značku znovu a pokračovat v jízdě.

Software byl také otestován na reálném robotu. Experimenty ukázaly, že při správně volbě parametrů pro všechny použité algoritmy robot dokáže sledovat člověka a vyhýbat se překážkám tak, aby neztrácel sledovaný cíl. Nastávaly také specifické situace, kdy sledování bylo nemožné. Pokud se robot dostal do lokálního minima, algoritmus VFH ho občas nedokázal správně nasměrovat, což bylo také způsobeno kinematikou použitého robotu.

Tato práce byla velkým přínosem hlavně proto, že v ní byla možnost vyzkoušet různé techniky používané pro řízení mobilních robotů. Navíc jsme se díky této práci seznámili se simulačními prostředími pro robotiku a knihovnou OpenCV, která je běžně používána pro implementaci softwaru pro počítačové vidění a strojové učení.

Příloha A

Literatura

[1] Simple and effective budgee robot is a bridge to full assistant robot. http://www.33rdsquare.com/2014/01/simple-and-effective-budgee-robot-is.html.

[2] Derek Bradley and Gerhard Roth. Adaptive thresholding using the integral image. *Journal of Graphics Tools*, 12(2):13–21, 2007.

[3] Opencv: Contours hierarchy. http://docs.opencv.org/trunk/d9/d8b/tutorial_py_contours_hierarchy.html.

[4] Roland Siegwart, Illah Reza Nourbakhsh, and Davide Scaramuzza. *Introduction to autonomous mobile robots*. MIT Press. 272–290, 2004.

[5] Takafumi Matsumaru Jianzhao Cai. Human detecting and following mobile robot using a laser range sensor. *Journal of Robotics and Mechatronics*, 26, Jul 2014.

[6] Michael Teutsch, Thomas Mueller, Marco Huber, and Juergen Beyerer. Low resolution person detection with a moving thermal infrared camera by hot spot classification. *2014 IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2014.

[7] J Blanco, W Burgard, R Sanz, and JL Fernandez. Fast face detection for mobile robots by integrating laser range data with vision. In *Proc. of the International Conference on Advanced Robotics (ICAR)*, volume 2, pages 953–958. Citeseer, 2003.

[8] Muhammad Sarmad Hassan, Mafaz Wali Khan, and Ali Fahim Khan. Design and development of human following robot. *Student Research Paper Conference*, 2, Jul 2015.

[9] Andrej Babinec, Ladislav Jurišica, Peter Hubinský, and František Duchoň. Visual localization of mobile robot using artificial markers. *Procedia Engineering*, 96:1–9, 2014.

[10] Xu Liu, David Doermann, Huiping Li, K. C. Lee, Hasan Ozdemir, and Lipin Liu. A novel 2d marker design and application for object tracking and event detection. *Lecture Notes in Computer Science*, 5358.

[11] S. Garrido-Jurado, R. Muñoz-Salinas, F.j. Madrid-Cuevas, and M.j. Marín-Jiménez. Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recognition*, 47(6):2280–2292, 2014.

[12] Opencv: About. `http://opencv.org/about.html`.

[13] F. Jurie and M. Dhome. A simple and efficient template matching algorithm. *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001.*

[14] Template matching. `http://docs.adaptive-vision.com/4.7/studio/machine_vision_guide/TemplateMatching.html`.

[15] Opencv: Detection of aruco markers. `http://docs.opencv.org/3.1.0/d5/dae/tutorial_aruco_detection.html`.

[16] Urs Ramer. An iterative procedure for the polygonal approximation of plane curves. *Computer Graphics and Image Processing*, 1(3):244 – 256, 1972.

[17] Paul Prober and Bill Wellman. Optimum pinhole camera design. *Hue Candela*, 2002.

[18] Erik Cuevas, Daniel Zaldivar, and Raul Rojas. Kalman filter for vision tracking. Aug 2005.

[19] Rogger Labbe. *Kalman and Bayesian Filters in Python.*

[20] J. Borenstein and Y. Koren. The vector field histogram-fast obstacle avoidance for mobile robots. *IEEE Transactions on Robotics and Automation*, 7(3):278–288, 1991.

[21] R.J. van Breda and W.J. Smit. Applicability of vector field histogram star (vfh*) on multicopters. *International Micro Air Vehicles, Conferences and Competitions*, 2016.

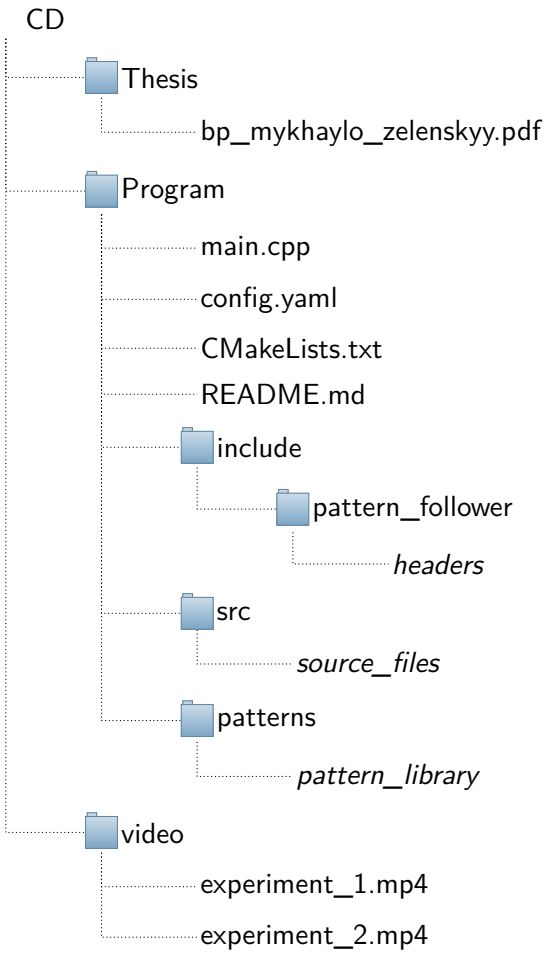
[22] Rich LeGrand. Closed-loop motion control for mobile robotics. *Circuit Cellar*, (169):34–46, Aug 2004.

[23] N. B. Nichols J. G. Ziegler. Optimum settings for automatic controller. *Trans. ASME*, 64, 1942.

[24] Coppelia robotics v-rep. `http://www.coppeliarobotics.com/index.html`.

Příloha B

Obsah CD



V souboru *README.md* je k dispozici krátký popis, jak program nainstalovat a používat. Soubor *config.yaml* obsahuje konfigurační parametry používané pro řízení reálného robotu s jejich podrobným popisem. Složka *patterns* obsahuje příklady používaných vizuálních značek, tuto knihovnu lze použít pro testování korelačního algoritmu.