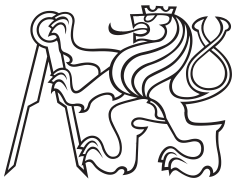
Bakalářská práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra kybernetiky

Následování člověka mobilním robotem

Mykhaylo Zelenskyy

Školitel: Ing. Jan Chudoba
Březen 2017

Poděkování

Chtěl bych poděkovat svému vedoucímu, Ing. Janu Chudobovi, za jeho rady a nápady.

Prohlášení

I declare that this work is all my own work and I have cited all sources I have used in the bibliography.

Prague, March 25, 2017

Prohlašuji, že jsem předloženou práci vypracoval samostatně, a že jsem uvedl veškerou použitou literaturu.

V Praze, 25. března 2017

Abstrakt

Tento mánuál představuje LaTeXovou třídu ctuthesis, její použití, požadavky na systém atd.

Klíčová slova: manuál, závěrečnná práce, LaTeX

Školitel: Ing. Jan Chudoba

Abstract

This manual shows how to use the ctuthesis LaTeX class, what are the requirements, etc.

Keywords: manual, degree project, LaTeX

Obsah

Obrázky

Tabulky

Kapitola 1
Úvod

Pomocný robot může se zdát mnoha z nás jako futuristický sen. Takový robot, který by místo nás nosil věci, pomáhal při nákupech, asistoval v nemocnicích nebo ošetřoval raněné ve válkách. Podobný robot by měl tolik výhod, že by v budoucnu to bylo trendem vlastnit takového pomocníka.

Bylo provedeno hodně různých výzkumů ohledně návrhu robotu, který by dokázal sledoval člověka. S tímto problémem jsou spojené dvě důležité otázky. První je jaké senzory použit pro lokalizaci člověka, druhá řeší řízení a navigaci robotu, 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 hodně různými senzory, např. laserovým, zvukovým nebo IČ 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 všechno, co se od něj očekává.

1.1 Cíl práce

Cílem této práce je navrhnout systém pro řízení robotu, aby dokázal sledovat člověka a vyhýbat se překážkám. Jelikož cíl pro robota by měl být unikátní, použije se vizuální značka, která by tento problém měla vyřešit. Robot dopředu bude vědět, jaký typ značky musí hledat, tím pádem se snižuje šance, že si splete cíl s jiným člověkem nebo překážkou.

Při návrhu tohoto systému je třeba uvědomit si několik věcí:

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

2. Aby značka byla dobře detekovatelná, musí být umístěna v ochranném prostředí, tj. být kontrastní v porovnání s pozadím.

Obrázek 1.1: Robot následující člověka

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

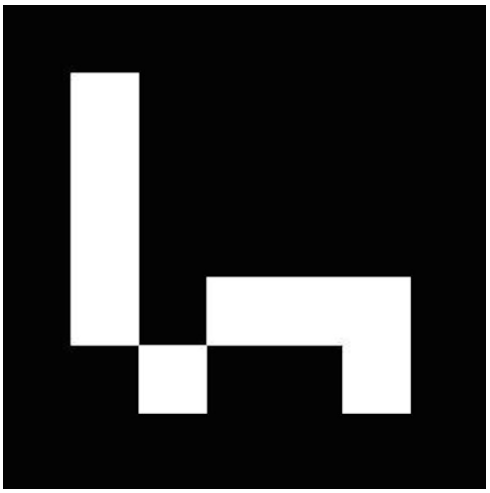
1.2 Motivace

Jak již bylo řečeno, robot s podobnou funkcionalitou by byl hodně užitečný nejen jako pomocník v každodenní činnosti lidí. V medicínských zařízených a domovech pro seniory takový robota by se mohl starat o pacienty (viz 1.1), také by našel své uplatnění v armádě, kde by pomáhal vojákům s různými náklady. Určitě vylepšená varianta robota následujícího člověka by mohla posloužit jako tělesná stráž.

Kapitola 2
Návrh systému

2.1 Volba vizuální značky

Jak již bylo zmíněno, mobilní robot bude následovat člověka, který má na sobě umístěnou předem známou vizuální značku. Pro tyto účely se v praxi běžně používají markery pro rozšířenou realitu. Jejich detekce je většinou jednoduchá a můžou v sobě uchovávat užitečnou informaci, např. identifikační číslo, podle kterého robot pozná, koho sleduje. Jednu z možných vizuálních značek, které jsou použity v práci, lze vidět na obrázku 2.1.



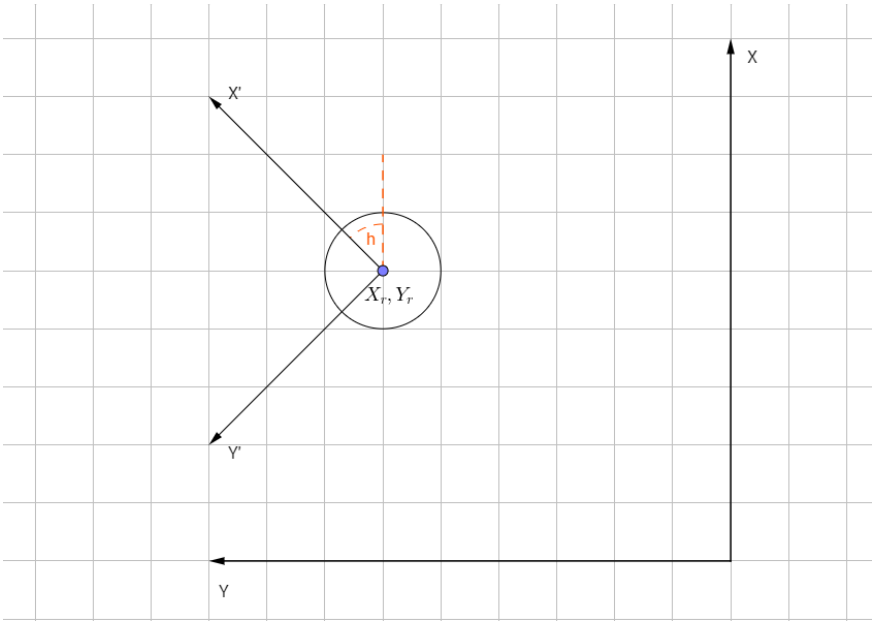
Obrázek 2.1: Příklad vizuální značky

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 osy 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 potom vztažená 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 Implementační záležitosti

Protože se očekává, že software, který bude výsledkem této práce, bude běžet real-time, musí být rychlý a stabilní. Z těchto důvodů bylo rozhodnuto, že celá implementace bude udělána v jazyce C++.

Kromě toho pro C++ je vytvořená OpenCV knihovna, která nabízí obrovské možnosti pro práci s obrazem. 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 v této knihovně je implementován algoritmus pro detekci ArUco markeru, 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 3.1. Tato metoda počítá práh pro malé části obrazu místo globálního nastavení prahu pro celý snímek.

Dále je třeba najít kontury. Algoritmus pro nalezení kontur je implementován v OpenCV knihovně. Tato funkce navíc zajišťuje zachování hierarchii kontur, tedy topologii obrázku, jako je vidět na 3.2. Tato informace je užitečné pro následující rozpoznávání, protože pak kontury na nejnižší nebo nejvyšší úrovni hierarchie můžou být ignorovány, pokud se předpokládá, že vizuální značka bude umístěna v ochranné zóně a bude obsahovat další pod-vzory.

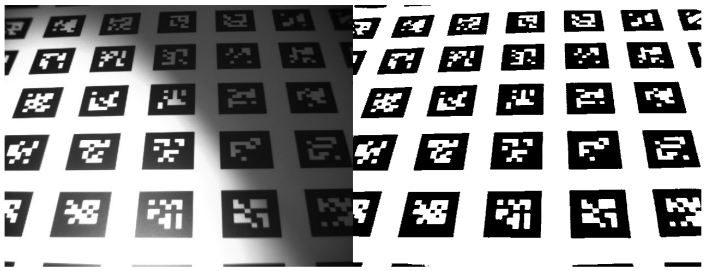
3.2 Rozpoznávání vzoru

Pro rozpoznávání vizuální značky byly zvoleny dva algoritmy: jeden založený na korelaci známých vzorů s nalezeným, druhý přímo z OpenCV knihovny – ArUco marker detektor.

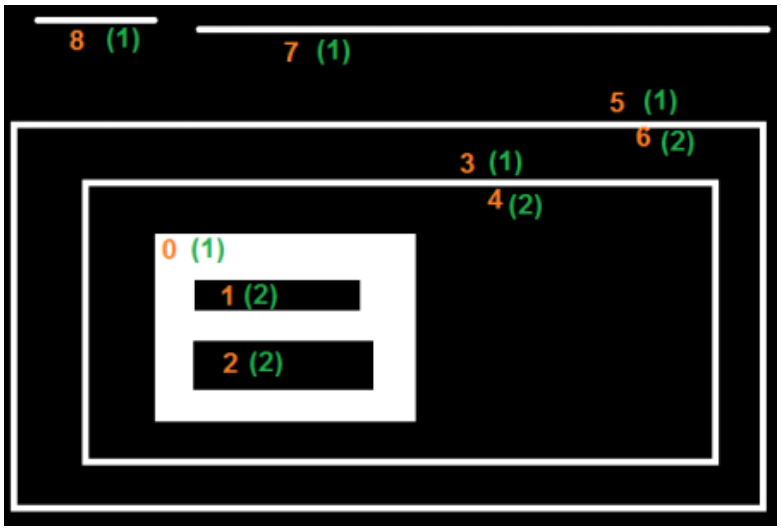
3.2.1 Korelační algoritmus

Tento algoritmus využívá principu korelaci mezi dvěma veličinami, tedy mezi známým vzorem a nalezenou oblasti zájmu (Region of Interest, 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 v zjednodušeném případě je 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



Obrázek 3.1: Příklad použití adaptivního prahování



Obrázek 3.2: Hierarchie (topologie) obrazu

křivky se používá Ramerův Douglasův Peuckerův algoritmus (viz 3.4), který je naimplementován v OpenCV knihovně.

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.

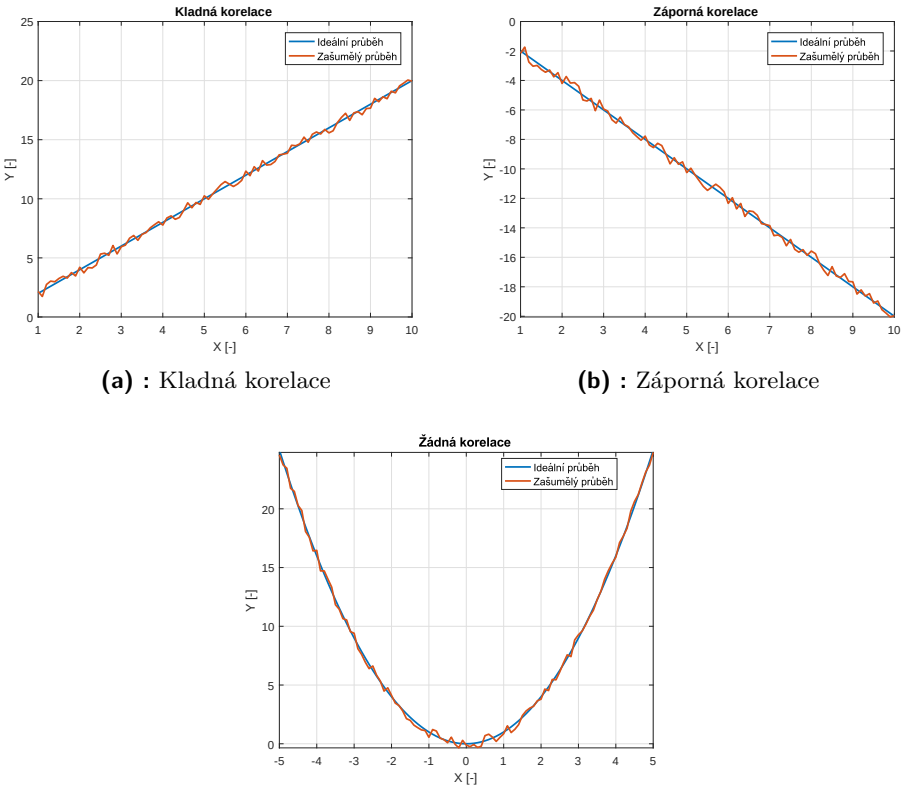
Dále se jenom vyhodnotí, jak nalezená ROI a známý vzor jsou na sobě závislý. Pokud je vypočtený korelační koeficient větší, než zadaný práh, ROI se vyhodnotí jako správně označena a algoritmus vrátí souřadnice jejích rohů pro následující zpracování.

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

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

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

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



(a) : Kladná korelace

(b) : Záporná korelace

(c) : Žádná korelace

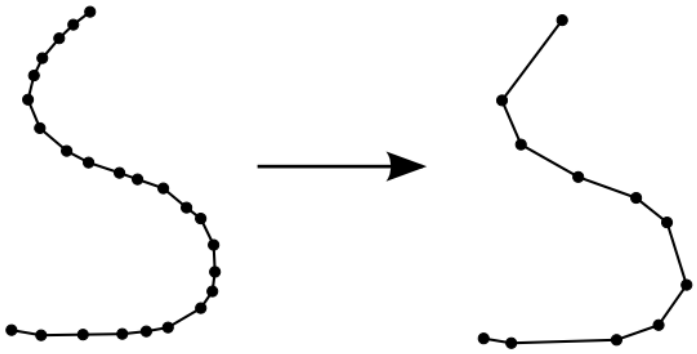
Obrázek 3.3: Příklad korelace mezi dvěma veličinami

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

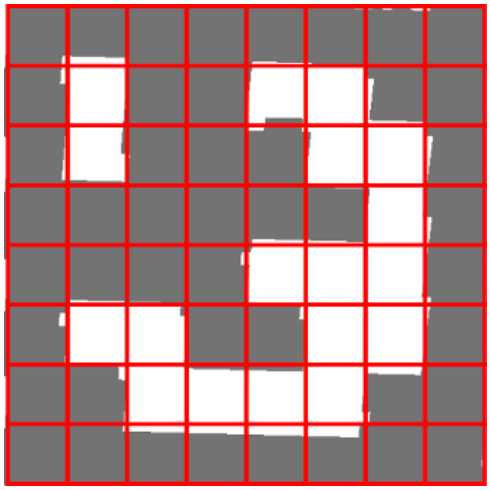
3.2.2 ArUco marker detektor

Tento algoritmus je naimplementován v OpenCV knihovně a 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 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 3.6). Pokud detektor ve svém slovníku obsahuje vzor se stejnou binární maskou, vrátí souřadnice rohů nalezené ROI.



Obrázek 3.4: Aproximace křivky pomocí Ramerova Douglasova Peuckerova algoritmu



Obrázek 3.5: Buňky ArUco markeru

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, kde vzdálenost mezi kamerou a značkou je vyjádřena jako

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

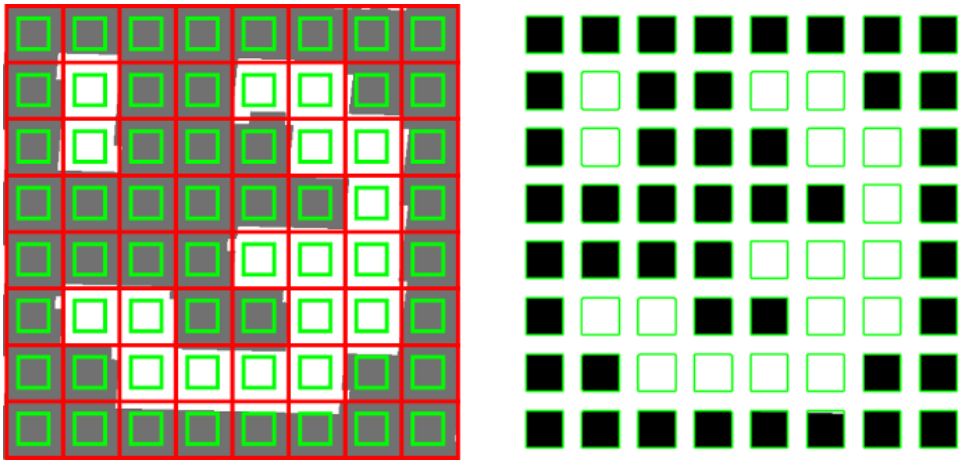
kde

x: Nejkratší vzdálenost mezi dvěma rohy nalezené značky.

f: Ohnisková vzdálenost.

X: 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



Obrázek 3.6: Vyhodnocení obrázku pomocí ArUco detektoru

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

kde
w: 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: Pixel vyjadřující střed kamery v horizontálním směru.
x_t: Pixel vyjadřující střed nalezené značky v horizontálním směru.

3.4 Kalmanův filtr

V případě, že poloha značky vůči kameře není detekovatelná, robot by neměl zastavovat. Z tohoto důvodu je třeba predikovat polohu cíle z předchozích měření. Pro tyto účely v práci je použit Kalmanův filtr.

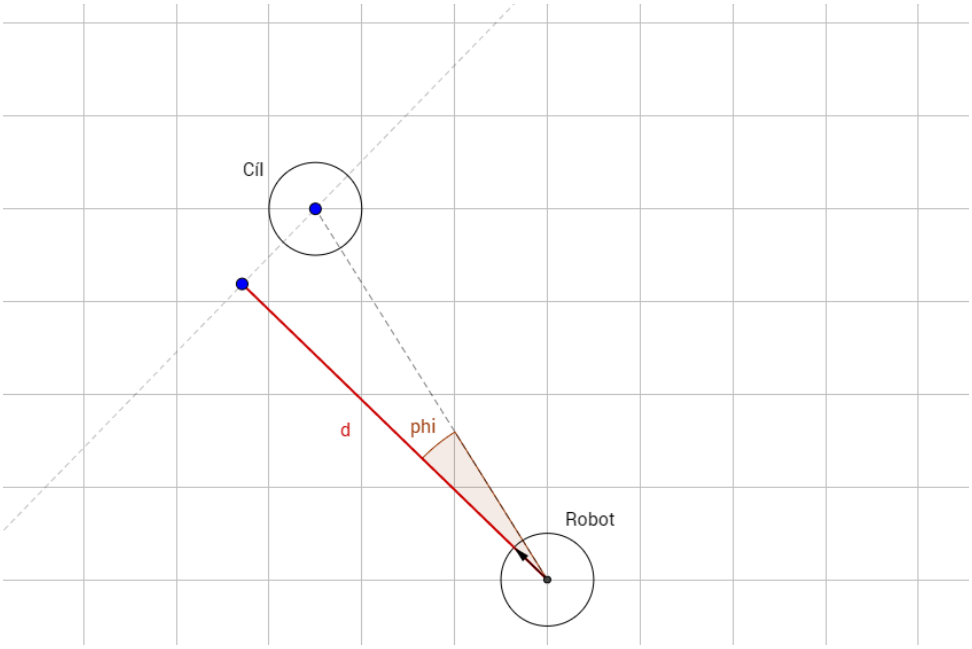
Model lze popsat pomocí následujícího stavového vektoru

$$\boldsymbol{x} = \begin{bmatrix} x & y & v_x & v_y \end{bmatrix}^T, \tag{3.9}$$

kde v_{lin}, resp. v_{ang}, značí rychlost ve směru osy X, resp. Y.
Stavový model je pak vyjádřen jako

$$\boldsymbol{x}_{t+1} = \boldsymbol{F}\boldsymbol{x}_t, \tag{3.10}$$

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



Obrázek 3.7: Měřené veličiny d a ϕ

$$\boldsymbol{F} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & dt & 0 \\ 0 & 0 & 0 & dt \end{bmatrix}. \tag{3.11}$$

Pro sledování cíle lze Kalmanův filtr rozdělit do tří kroků:

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

2. Predikce ($t > 0$). V tomto kroku se provádí predikce polohy cíle v čase $t + 1$, tj. $\boldsymbol{x}_{t+1}$, dle 3.10. Také se počítá nová kovarianční matice dle následujícího vztahu

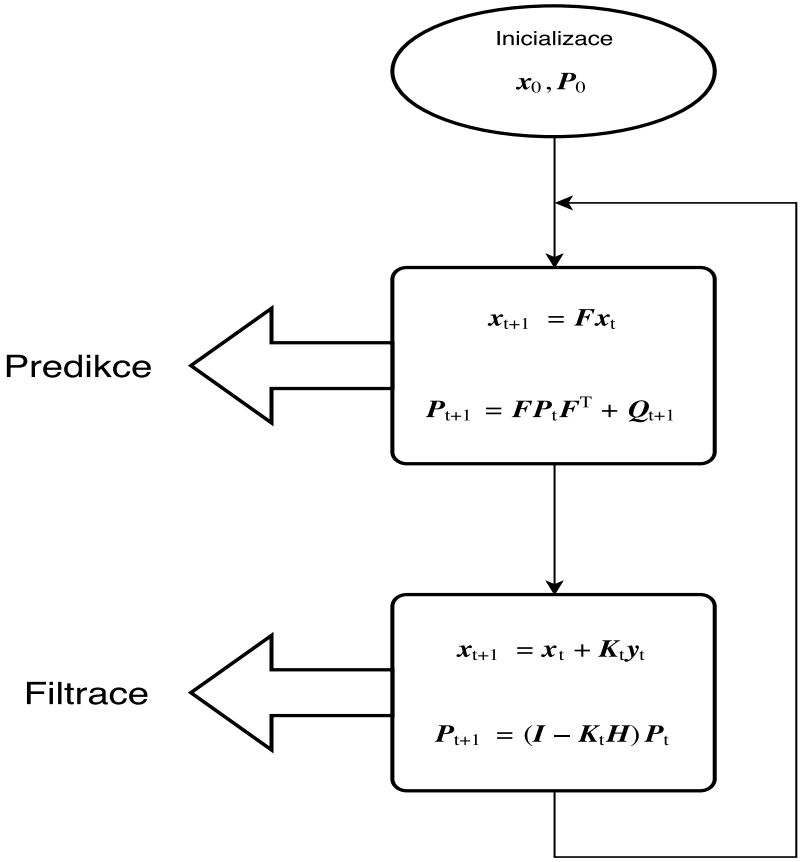
$$\boldsymbol{P}_{t+1} = \boldsymbol{F}\boldsymbol{P}_t\boldsymbol{F}^T + \boldsymbol{Q}_{t+1}, \tag{3.12}$$

kde $\boldsymbol{Q}$ značí matici kovariancí šumů.

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

$$\boldsymbol{y}_t = \begin{bmatrix} x_{m_t} \\ y_{m_t} \end{bmatrix} - \boldsymbol{H}\boldsymbol{x}_t, \tag{3.13}$$

kde



Obrázek 3.8: Shrnutí algoritmu Kalmanova filtru

$$\boldsymbol{H} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}. \tag{3.14}$$

Dále se vypočítá Kalmanovo zesílení, pro nějž platí, že

$$\boldsymbol{K}_t = \boldsymbol{P}_t \boldsymbol{H}^T (\boldsymbol{H} \boldsymbol{P}_t \boldsymbol{H}^T + \boldsymbol{R})^{-1}, \tag{3.15}$$

kde $\boldsymbol{R}$ ja matice kovariancí šumů.

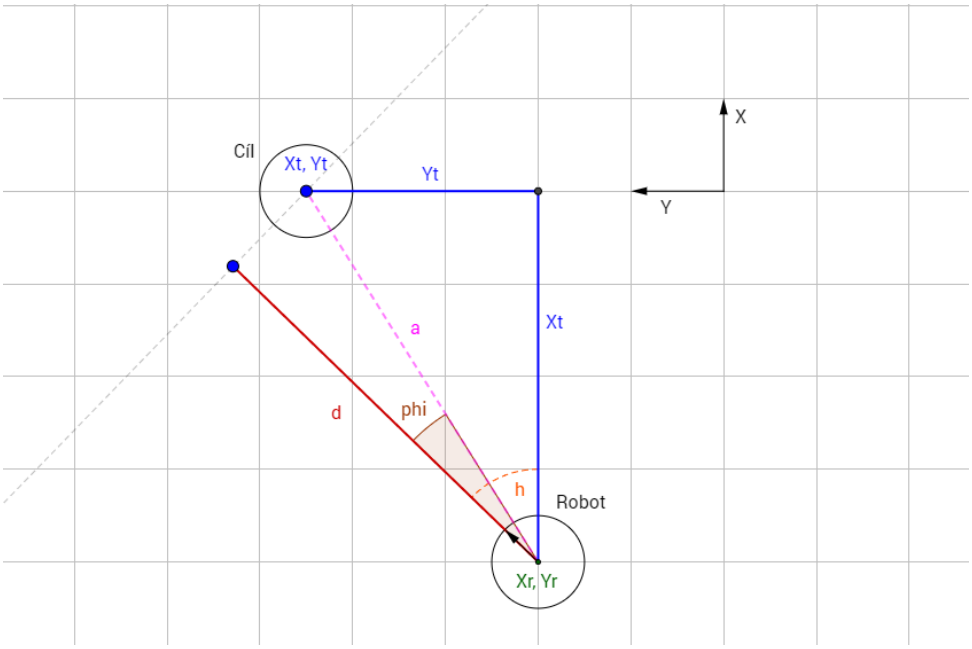
Následně se provede zpřesnění polohy cíle a aktualizuje se kovarianční matice:

$$\boldsymbol{x}_{t+1} = \boldsymbol{x}_t + \boldsymbol{K}_t \boldsymbol{y}_t, \tag{3.16}$$

$$\boldsymbol{P}_{t+1} = (\boldsymbol{I} - \boldsymbol{K}_t \boldsymbol{H}) \boldsymbol{P}_t. \tag{3.17}$$

Z nalezených hodnot x a y lze jednoduše odvodit hodnoty d a ϕ, což je také vidět na 3.9. Jelikož se předpokládá, že poloha robotu je známa, lze spočítat x_t a y_t, což je poloha cíle vůči robotu, jako

$$x_t = x - x_r, \tag{3.18}$$



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

$$y_t = y - y_r. \tag{3.19}$$

Dále pak ϕ a d jsou nalezeny dle vztahů

$$\phi = h - \text{atan2}(y_t, x_t), \tag{3.20}$$

$$d = \sqrt{x_t + y_t}\cos(\phi). \tag{3.21}$$

Kapitola 4
Řízení robotu

4.1 Obstacle avoidance

Pro řízení autonomního robotu je důležité, aby se dokázal vyhýbat překážkám, když sleduje svůj cíl.

4.1.1 Vector Field Histogram

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

Celý proces tímto způsobem můžeme rozdělit na tři kroky: vytvoření primárního polárního histogramu, prahování primárního histogramu a vytvoření z něj binárního a výběr kandidátů pro směr pohybu.

Primární polární histogram

Polární histogram je rozdělen na sektory tak, aby každý z nich odpovídal úhlu α, který se volí takovým způsoben že $\frac{360}{\alpha}$ je celý číslo. Tedy například pro $\alpha = 15°$ histogram obsahuje 24 sektorů (viz 4.2).

Pro každou buňku aktivního regionu, tedy lokální mapy vytvořené kolem robotu, se spočte směr a její význam (magnitude).

Směr spočteme dle následujícího vztahu

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

kde

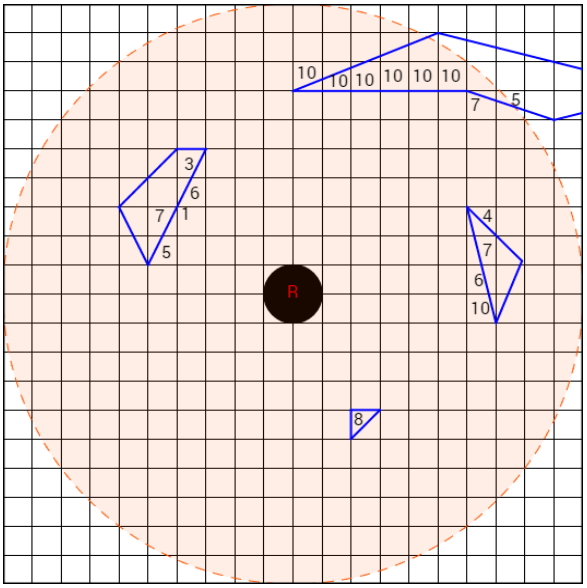
x_o, y_o: Souřadnice robotu v mapě (centrum mapy).

x_i, y_i: Souřadnice buňky.

Dále pak magnitude

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

kde



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

$c_{i,j}$: Obsazenost buňky.
$d_{i,j}$: Vzdálenost buňky od pozice robotu.
Parametry a a b se volí dle vztahu:

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

Aby robot nejel blízko 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 se potom vytvoří dle vztahu

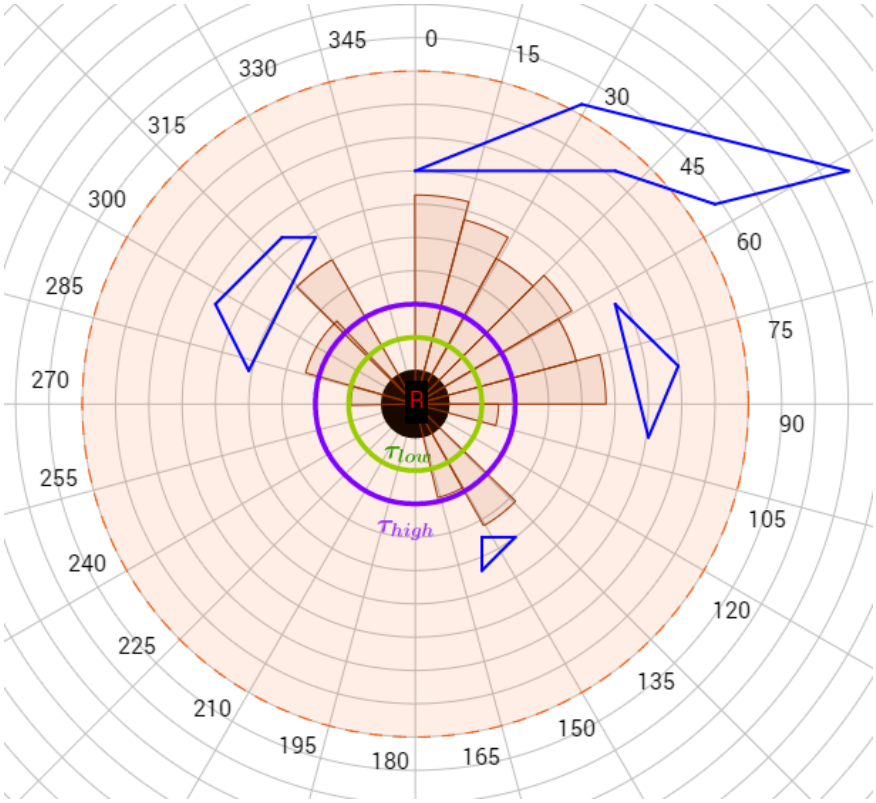
$$H_k^p = \sum_{i,j \in C_\alpha} m_{i,j} h_{i,j}, \tag{4.4}$$

kde

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

Binární polární histogram

Aby bylo možné určit kandidáty pro další směr pohybu, primární histogram musí být převeden do binární podoby.



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

$$H^b_{k,i} = \begin{cases} 1 & \text{jestli } H^p_{k,i} > \tau_{high}, \\ 0 & \text{jestli } H^p_{k,i} < \tau_{low}, \\ H^b_{k,i-1} & \text{jinak.} \end{cases}$$

Výběr kandidátů

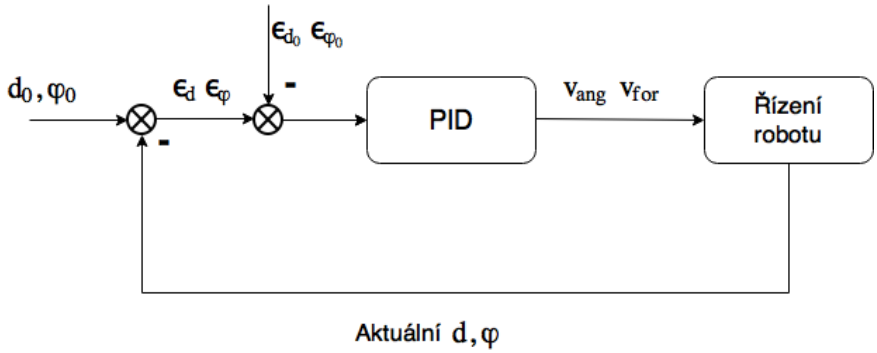
Kandidáty pro směr pohybu se volí podle toho, do jaké kategorie je zařazeno volné místo v binárním histogramu. Ty průjezdy, jež mají velikost (tedy vzdálenost mezi pravým okrajem k_r a levým k_l) menší, než s_{max}, se nazývají úzké, jiné jsou naopak nazývány široké.

Pro úzké průjezdy lze zvolit pouze jednoho kandidáta, a to

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

Široké průjezdy mají tři možné kandidáty:

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



Obrázek 4.3: PID regulace

Nový směr pohybu se zvolí dle minimální ceny spočítané pro kandidáta c_i pomocí vztahu

$$g(c_i) = \mu_1 \Delta\left(c_i, k_t\right) + \mu_2 \Delta\left(c_i, \frac{h}{\alpha}\right) + \mu_3 \Delta\left(c_i, k_{d,n-1}\right) \tag{4.5}$$

přičemž
$\Delta(c_1, c_2) = \min\left\{|c_1 - c_2|, |c_1 - c_2 - \frac{360°}{\alpha}|, |c_1 - c_2 + \frac{360°}{\alpha}|\right\}$.
$k_{d,n-1}$: Minule zvolený kandidát.
μ_1, μ_2, μ_3: Konstanty, zvolené dle vztahu $\mu_1 > \mu_2 + \mu_3$.

4.2 PID regulátor

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

Kapitola 5

Simulace

Pro simulaci byl zvolen V-Rep simulátor. V tomto prostředí byly vytvořeny dva roboty: jeden s vizuální značkou, druhý byl vybaven kamerou a laserovým dálkoměrem a sledoval prvního. Maximální dopředná rychlost obou robotů byla zvolena jako $2\,\mathrm{m\,s^{-1}}$

5.1 Porovnání algoritmů detekce

Oba algoritmu byly porovnávány při použití značky o velikosti 10x10 cm a rozlišením kamery 512x512 pixelů. Bylo porovnáno několik parametrů obou algoritmu, a to maximální vzdálenost, na které je vizuální značka ještě detekovatelná, stabilní vzdálenost, při které značka je detekována bez přerušení. Dále se porovnával 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 sice ArUco detektor 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

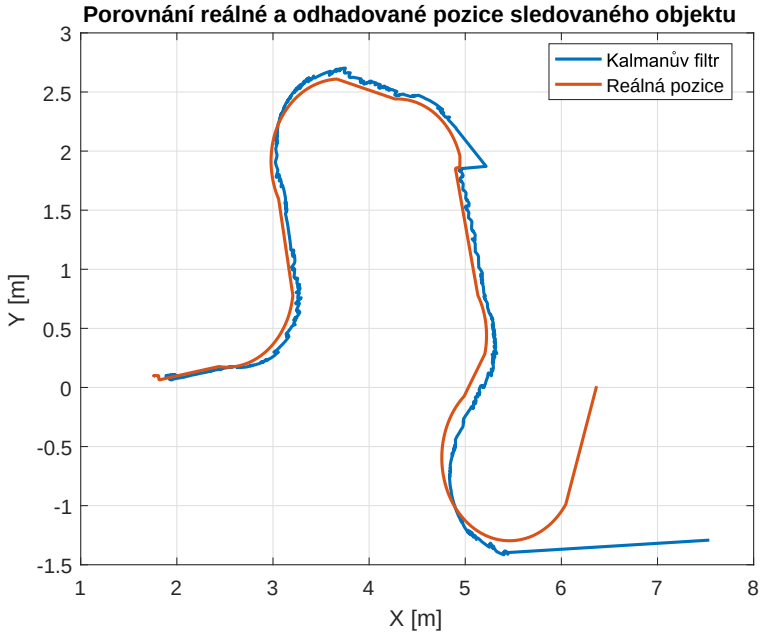
Hlavní rozdíl mezi těmito algoritmy spočívá v možnostech jejich využití. Zatímco ArUco detektor je zaměřený pouze na ArUco markery, korelační algoritmus je obecnější a dá se použít fakticky na jakýkoliv typ vizuální značky. ArUco detektor umožňuje přidání dalších značek do své knihovny, ale

Obrázek 5.1: Příklad značky pro korelační algoritmus

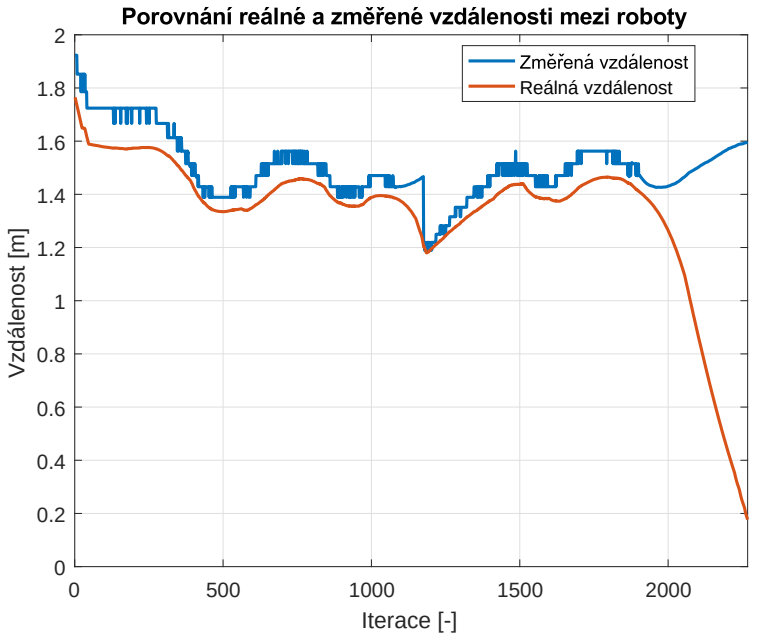
jedná se pouze o binární obrázky podobné znázorněnému na 3.2.2. Korelační algoritmus si poradí i se značkou, kterou nelze reprezentovat jako bit-mapu, viz např. 5.1.

5.2 Porovnání naměřených a reálných hodnot

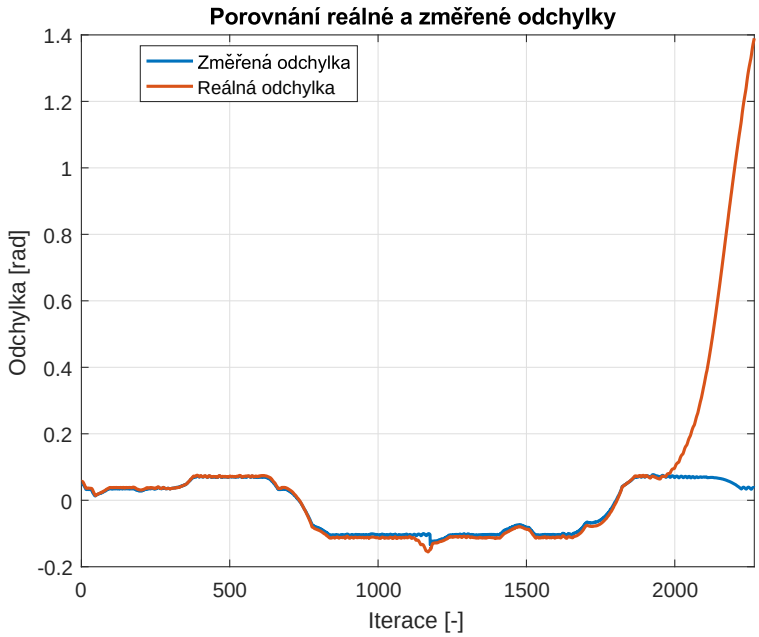
Jelikož se v implementovaném programu provádí různá měření, je třeba porovnat, jak naměřené hodnoty jsou správné vůči reálným. Kalmanův filtr, popsaný v 3.4, predikuje polohu cíle a v případě neúspěšné detekce vizuální značky umístění sledovaného objektu vůči robotu je odvozeno právě z predikované polohy. Na obrázku 5.2 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 hodně brzo zmizela ze zorného pole robotu a nebylo možné ji v krátké době najít. Toto také odpovídá výsledkům, které jsou na 5.3 a 5.4.



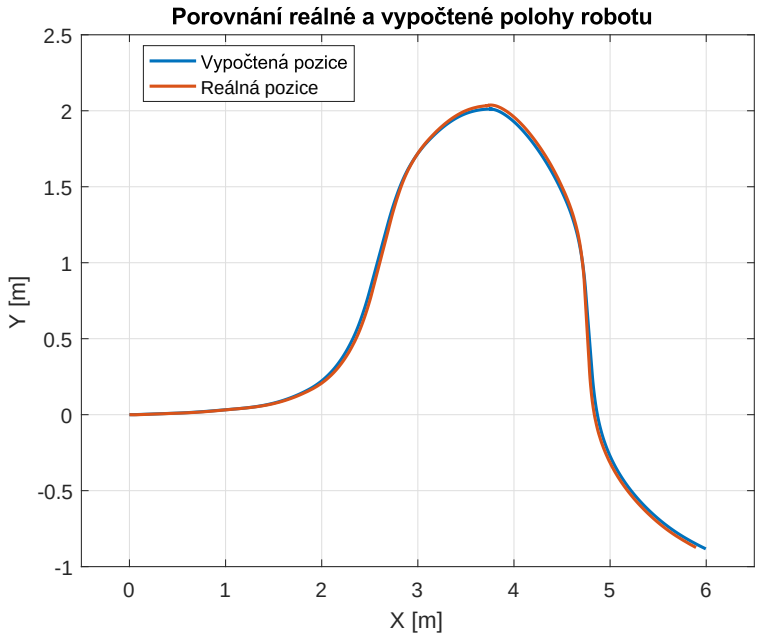
Obrázek 5.2: Poloha sledovaného objektu během simulace



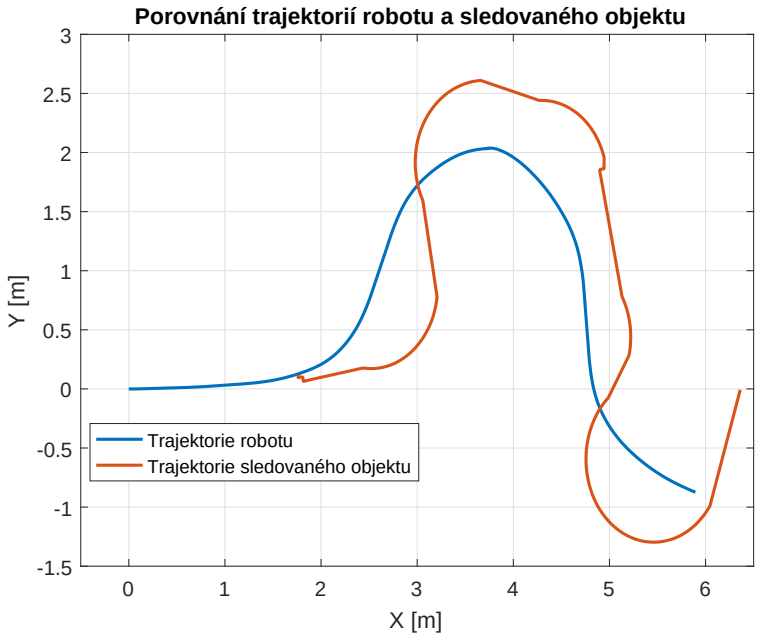
Obrázek 5.3: Vzdálenost mezi robotem a sledovaným objektem (d)



Obrázek 5.4: Odchylka sledovaného objektu od středu kamery ϕ



Obrázek 5.5: Poloha robotu během simulace



Obrázek 5.6: Trajektorie robotu a sledovaného objektu

Kapitola 6

Závěr