Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Bakalářská práce

Robot Kinbot

Plzeň 2014 Michal Bratner

Prohlášení

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 5. května 2014

Michal Bratner

Poděkování

Chtěl bych poděkovat Ing. Petru Vaněčekovi, Ph.D. za ochotu, užitečné rady, trpělivost a čas věnovaný při konzultacích, které mi pomohly k lepšímu vypracování této práce.

Anotace

Tato bakalářská práce se zabývá využitím senzoru Microsoft Kinect pro ovládání robota Kinbo. Cílem práce je vytvořit s pomocí platformy Kinbo^2 moduly pro ovládání robota a interakci s člověkem. V práci je uveden popis určení pozice robota v prostoru za pomoci QR kódů. Dále je popsána interakce s člověkem a reakce na různá gesta. Součástí práce je také navigační část pro pohyb robota v prostoru. Na závěr je uveden popis struktury výsledné práce.

Abstract

This bachelor thesis deals with using Microsoft Kinect sensor to control Kinbo robot. Objective of this thesis is to create a patform Kinbo^2 using modules for control robot and human interaction. The work describes the determination of the robot position in space with help of QR codes. Further describes human interaction and reaction to different gestures. The work also includes the navigation part for the robot movement in space and concludes with a description of the structure of the final work.

Obsah

1 Úvod

Modulární platforma Kinbo^2 poskytuje jednoduché a snadno rozšiřitelné rozhraní pro programování stejnojmenného robota vybaveného senzorem Microsoft Kinect. Modulárnost platformy zajišťuje její snadnou rozšiřitelnost a použitelnost s různými druhy robotů. Architektura platformy je rozdělena na tři samostatné systémy pro server, klienta a robota. Toto rozdělení je provedeno z důvodu běhu jednotlivých systémů na různých počítačích. Systémy pak mezi sebou komunikují pomocí počítačové sítě. Každý systém je kvůli zabezpečení před chybami způsobenými uživatelským projektem rozdělen do samostatných procesů platformy a uživatelského projektu.

Cílem této práce je navrhnout a implementovat moduly pro jednotlivé systémy, které umožní robotovi Kinbo zjistit aktuální pozici v prostoru a umožní mu také interakci s člověkem. Aby bylo možné moduly používat pomocí platformy Kinbo^2, musí být vytvořeny jako dll knihovny v programovacím jazyce C#.

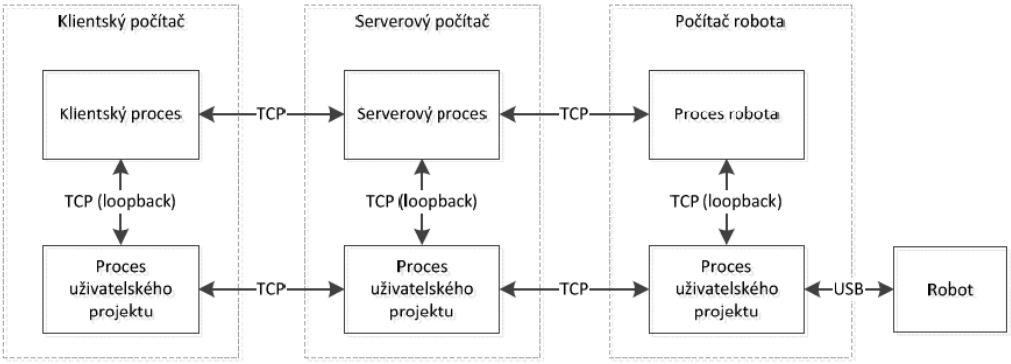
Po přečtení práce získáte přehled o postupech, které jsou využitelné pro dosažení cíle této práce. V kapitole 3 je popsán způsob nalezení pozice robota, kapitola 4 pak popisuje rozhodování robota o následujícím kroku. Kapitola 5 pojednává o způsobu interakce robota s uživatelem pomocí senzoru Microsoft Kinect a kapitoly a popisují výslednou strukturu a činnost projektu vytvořeného v této práci.

2 Použité prostředky

Při vytváření této práce jsem využil několik již existujících produktů, jejichž funkcionalita je vhodná pro cíl této práce, a sice ovládání robota Kinbot.

2.1 Platforma Kinbo[2]

Hlavním účelem této platformy je poskytnutí jednoduchého a rozšiřitelného rozhraní, jehož pomocí lze ovládat mobilního robota s tankovým podvozkem vybaveného zařízením Microsoft Kinect. Jak je uvedeno v [1], architektura platformy je rozdělena na tři systémy: server, klient a robot (viz obr. 2.1). Rozdělení je provedeno z důvodu, že každý systém je určen pro běh na jiném počítači. Systémy spolu tedy musejí komunikovat prostřednictvím počítačové sítě. Každý systém je kvůli zabezpečení před rizikem chyb způsobených uživatelským projektem ještě rozdělen do samostatných procesů platformy a uživatelského projektu. Tyto procesy vzájemně komunikují prostřednictvím síťového loopback rozhraní. Procesy uživatelského projektu mezi sebou komunikují prostřednictvím počítačové sítě nezávisle na procesech platformy a vytvářejí tak komunikační kanál pro zprávy zaslané uživatelským systémem.



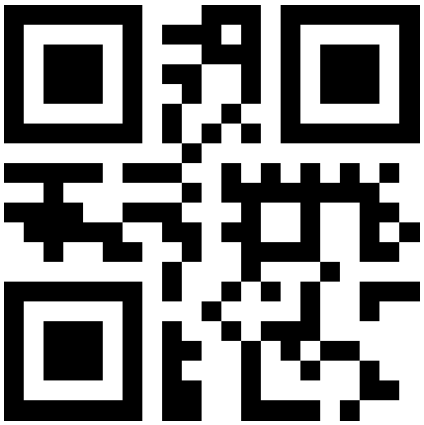
Obrázek 2.1: Blokový diagram vývojové platformy Kinbo[2]. Převzato z [1].

2.2 Navigační kódy

Pro funkčnost této práce jsou využívány kódy rozmístěné v okolí. Slouží jako orientační body pro robota, který po jejich nalezení dokáže určit vlastní polohu v prostoru. Pro tento účel bylo vyzkoušeno více druhů kódů (např. DataMatrix a QR kód). Výsledkem bylo využití QR kódů, které se projevily jako nejvýhodnější z hlediska uložitelné informace a rozpoznatelnosti využitou knihovnou.

Použity jsou QR kódy s členitostí 21x21 bodů. Aby bylo dosaženo co největšího dosahu čitelnosti, ale zároveň zachována rozumná velikost, jsou používány kódy o velikosti 20x20 centimetrů. Uloženy jsou v nich souřadnice, na kterých se daný QR kód nachází a úhel natočení QR kódu vůči ose `x` použitého souřadného systému prostoru, ve kterém se robot nachází. Hodnota úhlu se zvyšuje proti směru hodinových ručiček. Data jsou uložena ve formátu `souradniceX,souradniceY,uhel`. Příklad může být QR kód s uloženou informací `100,100,180`, který je zobrazen na obrázku 2.2. Hodnota každé souřadnice udává počet centimetrů od začátku souřadného systému, úhel je uveden ve stupních.

Maximální možné hodnoty souřadnic při udržení uvedené minimální členitosti kódu jsou devítimístné, což vzhledem k vyjádření v centimetrech odpovídá maximální velikosti mapovatelného prostoru v řádech tisíců kilometrů. Hodnoty jsou zakódovány jako text pomocí online geenerátoru QR kódů dostupného na [8].



Obrázek 2.2: QR kód s informací 100,100,180

2.3 Knihovna ZXing.net

Vzhledem k používanému programovému vybavení platformy Kinbo[2] bylo kritériem pro hledání použitelné knihovny také to, aby ji bylo možno použít v prostředí C# .NET. Tomuto kritériu vyhovuje nalezená knihovna ZXing.Net, která je dostupná na [2]. Jedná se o knihovnu pro dekódování různých druhů kódů, včetně využitých QR kódů, pro které poskytuje kvalitní podporu.

Práce s knihovnou ZXing.Net je poměrně jednoduchá a není potřeba složitých konstrukcí pro využití její funkčnosti. Použitá verze 0.12.0.0 navíc obsahuje také přímou podporu pro datový typ `ColorImageFrame`, který je v zařízení MS Kinect používán pro reprezentaci snímaného obrazu. Toho můžeme s výhodou využít a není třeba provádět komplikované převody do jiného datového typu. Lze tedy obraz reprezentovaný tímto datovým typem přímo poslat do dekodéru implementovaného v této knihovně.

Dekodér následně v obrazu vyhledává QR kódy. Pokud hledání skončí úspěšně a dokáže detekovat kód, vrátí pole výsledků reprezentující všechny nalezené kódy, které se podařilo bez chyby dekódovat. Pokud v obraze žádný QR kód nenajde, vrátí hodnotu `null`. Z výsledku, který je vrácen při úspěšném hledání, můžeme snadno získat dešifrovaný text. Můžeme také získat souřadnice, na kterých se v obraze daný kód nachází. Tyto souřadnice následně využijeme pro nalezení vzdálenosti QR kódu od zařízení.

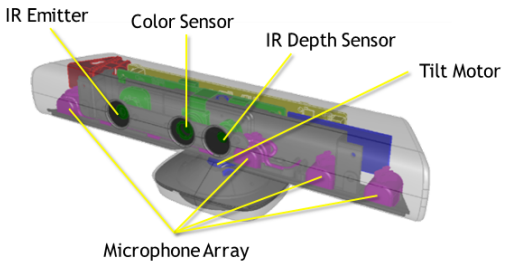
2.4 Microsoft Kinect

Zařízení Microsoft Kinect (viz obr. 2.3) bylo původně vytvořeno jako periferie k herní konzoli Xbox 360. S jeho pomocí může hráč k ovládání používat pohyby svého těla, což umožnilo nový rozměr vnímání a prožívání herních zážitků.

Nás ovšem více zajímá technická specifikace. Kinect snímá své okolí pomocí RGB kamery s rozlišením až 1280x960 pixelů. Takové rozlišení ovšem generuje pouze 12 snímků za sekundu. Využitelnější variantou je tedy rozlišení 640x480 pixelů s frekvencí 30 snímků za sekundu. Je také výhodnější pro synchronizaci dat z hloubkové kamery, která také pracuje s rozlišením 640x480 pixelů a frekvencí 30 snímků za sekundu. Pro získání hloubkových

Obrázek 2.3: Senzor Microsoft Kinect. Převzato z [4].

dat má Kinect emitor infračervených paprsků, který do prostoru vysílá body v pevně stanovené mřížce. Podle [9] senzor tuto mřížku následně snímá infračervenou kamerou. Výsledná hloubka je vypočtena na základě triangulace mezi emitovanou a přečtenou konfigurací mřížky bodů. Tato technologie se jmenuje Light Coding a byla vyvinuta společností PrimeSense. Zorné pole kamer je 57° v horizontálním směru a 43° ve vertikálním směru. Kinect lze také naklonit ve vertikální ose na obě strany až o 27° Na obrázku 2.4 můžeme vidět rozložení komponent v senzoru MS Kinect.



Obrázek 2.4: Rozložení komponent senzoru Microsoft Kinect. Převzato z [4].

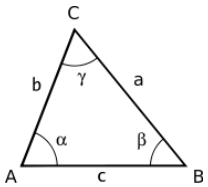
3 Určení pozice robota

Aby mohl robot určit svou polohu v prostoru s daným souřadným systémem, potřebuje nejprve nalézt určitý záchytný bod. K tomu slouží již zmíněné QR kódy se souřadnicemi. Pokud robot pomocí zařízení MS Kinect nalezne ve svém okolí QR kód, má referenční bod, s jehož pomocí dokáže určit vlastní pozici. Bohužel kvůli technickým limitům senzoru je možné určit pozici robota pouze po nalezení kódu ve vzdálenosti od 0,8 do 2,4 metru. Na kratší vzdálenost než 0,8 metru není senzor schopen rozeznat vzdálenost kódu a na větší vzdálenost než 2,4 metru již knihovna ZXing.net nedokáže QR kód o rozměrech 20x20 cm rozeznat kvůli nízkému rozlišení snímaného obrazu.

3.1 Výpočet souřadnic pomocného bodu

Když se podíváme na strukturu QR kódu (např. na obr. 2.2), vidíme tři referenční čtverce v rozích. Knihovna ZXing při nalezení QR kódu vrací pole souřadnic v obraze, které odpovídají právě těmto bodům. Pokud tedy levému hornímu bodu (označme jej `A`) přiřadíme hodnoty souřadnic, které jsou uloženy v QR kódu, pak dopočtením souřadnic pravého horního bodu (označme jej `C`) získáme pomocný bod, s jehož pomocí můžeme vypočítat souřadnice robota. Jelikož velikost všech QR kódů, které jsou k této práci využity, je shodná, zůstává shodná také vzdálenost těchto dvou bodů, kterou tedy můžeme prohlásit za konstantu.

Pro možnost vypočtení souřadnic pomocného bodu potřebujeme ještě třetí bod, který si označíme `B`, a který by s body `A` a `C` vytvořil trojúhelník. Bod `B` můžeme zvolit libovolně, takže budeme znát jeho souřadnice a také vzdálenost od bodu `A`. Spojením těchto bodů nám tedy vznikne trojúhelník. Jeho obecnou podobu popisuje obrázek 3.1.



Obrázek 3.1: Obecný trojúhelník

Známe body `A` a `B` a známe také strany `b` a `c`. Stranu `b` jsme totiž označili za konstantu díky neměnící se velikosti QR kódů. V QR kódech je také uložen úhel, který udává natočení kódu vůči ose `x`, přičemž jeho hodnota se zvyšuje natáčením kódu v protisměru hodinových ručiček. Pokud tedy strategicky umístíme bod `B`, budeme moci úhel α prohlásit za úhel uložený v kódu. Umístíme tedy bod `B` tak, aby strana `c` byla rovnoběžná s osou `x`. Pro jednoduchost bude bod `B` vždy umístěn tak, že jeho souřadnice `x` je nulová a souřadnice `y` je shodná jako u bodu `A`. Díky tomu známe délku strany `c`, která je rovna hodnotě souřadnice `x` bodu `A`. Nyní tedy můžeme vypočítat délku strany `a`. To provedeme za pomoci Cosinové věty, která má tvar

$$a^2 = b^2 + c^2 - 2bc \cdot \cos\alpha. \tag{3.1}$$

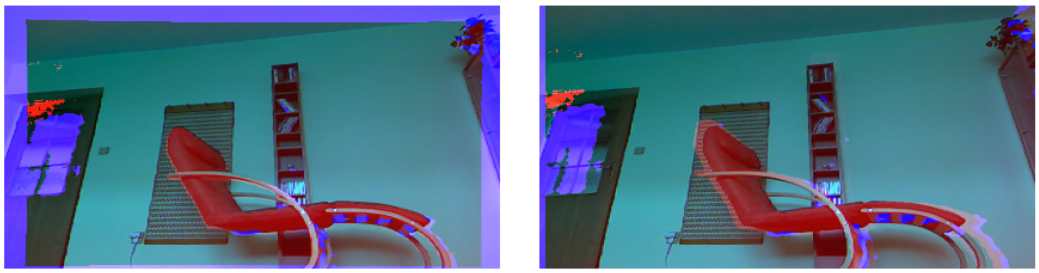
Nyní známe vzdálenost bodu C od bodu A i od bodu B a můžeme dopočítat souřadnice bodu C pomocí průsečíků dvou kružnic. Tento postup je popsán v sekci 3.3.

3.2 Výpočet vzdálenosti bodu od senzoru

Abychom mohli určit pozici robota v prostoru, nestačí nám pouze znát souřadnice nalezených bodů QR kódu, ale potřebujeme také znát jejich vzdálenost od senzoru. K tomu můžeme využít měření hloubky, které provádí senzor MS Kinect.

Prvním problémem, na který při využití hloubkového obrazu, snímaného senzorem MS Kinect, narazíme, je snímání hloubky jinou kamerou, než která snímá barevný obraz. Vzniká tedy situace, kdy jsou jednotlivé snímky vzájemně posunuty. Pokud tedy uvážíme v barevném obraze například bod se souřadnicemi $x = 100$ a $y = 100$, pak v hloubkovém obraze bod se stejnými souřadnicemi nereprezentuje tento bod v barevném obraze. K omezení vlivu tohoto problému je v knihovně Kinect SDK dostupné na [3] implementována mapovací funkce `MapColorFrameToDepthFrame`, jejímž účelem je co nejpřesněji přiřadit každý bod hloubkového obrazu na pozici se souřadnicemi odpovídajícího bodu v barevném obraze tak, aby bylo možné nalézt obě odpovídající hodnoty na stejných souřadnicích. Na obrázku 3.2 je znázorněn snímek z barevné kamery, v jehož červené složce je vložena naměřená hloubka. V pravém obrázku je hloubka bez použití mapovací funkce a v levém

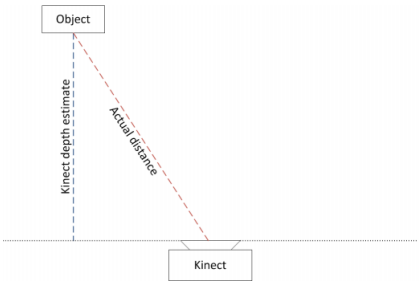
obrázku je použita mapovací funkce. Modrá místa označují neplatné hodnoty. Jde o místa, pro která senzor nedokázal změřit hloubku.



Obrázek 3.2: Snímky ze senzoru MS Kinect znázorňující vztah mezi barevným obrazem a hloubkou. Vlevo po použití mapovací funkce, vpravo bez mapování.

Pokud se na obrázcích zaměříme na křeslo, lze celkem snadno zpozorovat, že po použití mapovací funkce (levý obrázek) hodnoty naměřené hloubky mnohem lépe kopírují tvar skutečného předmětu, zatímco bez využití mapovací funkce (pravý obrázek) jsou rozdíly poměrně značné a jasně viditelné. Na levém obrázku si můžeme také všimnout, že po aplikování mapovací funkce je obraz ze tří stran ohraničen pásem neplatných hodnot. Tento pás je způsoben již zmíněnou problematikou odlišného zorného pole kamer pro barevný a hloubkový obrázek. V těchto místech tedy není možné určit hloubku, tudíž ani dopočítat vzdálenost předmětů v tomto pásmu.

Dalším problémem je to, že nelze přímo využí hloubku naměřenou senzorem, protože Kinect neměří vzdálenost objektu od senzoru, ale jeho hloubku v obraze vůči rovině senzoru (viz obr. 3.3).



Obrázek 3.3: Porovnání měřené hloubky senzorem Microsoft Kinect a skutečné vzdálenosti objektu od senzoru. Převzato z [6].

Z obrázku je ale také patrné, že skutečnou vzdálenost není problém dopočítat. Knihovna ZXing totiž pro bod, jehož vzdálenost chceme určit, vrací souřadnice v obraze, takže na těchto souřadnicích můžeme přečíst hloubku, která nám určí délku jedné strany pravoúhlého trojúhelníku (svislá strana na obr. 3.3). Jelikož známe velikost obrazu a také úhel zorného pole senzoru, lze stanovit konstantu udávající hodnotu úhlu na jeden pixel obrazu. Pokud nyní v horizontální ose určíme vzdálenost bodu od středu obrazu a vynásobíme tuto vzdálenost konstantou pro hodnotu úhlu na jeden pixel, získáme úhel, který svírají naměřená hloubka a skutečná vzdálenost. Nyní tedy v pravoúhlém trojúhelníku známe jeden z úhlů (označme jej α), přilehlou odvěsnu (naměřená hloubka, označíme ji `depth`) a hledáme velikost přepony (počítaná vzdálenost, označíme ji `distance`). Využijeme tedy goniometrickou funkci cosinus:

$$\cos\alpha = \frac{depth}{distance}. \tag{3.2}$$

Jednoduchou záměnou cosinu a neznámé `distance` získáme rovnici

$$distance = \frac{depth}{\cos\alpha}. \tag{3.3}$$

Výsledek této rovnice je námi hledaná vzdálenost bodu od senzoru.

3.3 Nalezení průsečíků kružnic

Když již máme dopočteny také souřadnice pomocného bodu, můžeme uvažovat, že máme trojúhelník, jehož dva vrcholy jsou reprezentovány danými body v QR kódu, pro které máme určené souřadnice a ve třetím vrcholu se nachází robot, jehož pozici hledáme. Známe tedy dva vrcholy trojúhelníku a díky dříve popsanému postupu také jejich vzdálenosti od robota, tedy dvě strany trojúhelníku. Mohli bychom také dopočítat třetí stranu, tedy tu, která spojuje dva známé vrcholy, ale tuto stranu při výpočtech nebudeme potřebovat. Pro nalezení neznámého vrcholu totiž využijeme postupu, který je používán v geometrii při tvorbě trojúhelníku podle věty sss. Tento postup je následující:

1. Sestrojíme stanu AB, kde A a B jsou známé vrcholy

2. Z vrcholu A sestrojíme kružnici o poloměru rovném délce strany AC, kde C je neznámý vrchol
3. Z vrcholu B sestrojíme kružnici o poloměru rovném délce strany BC, kde C je neznámý vrchol
4. Průsečík těchto kružnic nám dává neznámý vrchol C
5. Spojením vrcholů A a B s vrcholem C získáme trojúhelník ABC

Pro náš výpočet hledaného vrcholu C nám stačí využít znalosti bodů 2, 3 a 4 ve zmíněném postupu. Bod 1 je pro nás zbytečný, protože nás zajímají pouze souřadnice bodu C, výsledný trojúhelník také rýsovat nebudeme, tudíž nemusíme využít ani bod 5. Budeme tedy počítat průsečíky dvou kružnic, jejichž středy budou body `A` a `B`.

Pro popis kružnic použijeme obecnou rovnici kružnice, která je ve tvaru

$$(x - s_x)^2 + (y - s_y)^2 = r^2. \tag{3.4}$$

Tuto obecnou rovnici upravíme do tvaru

$$x^2 + y^2 - 2 \cdot s_x \cdot x - 2 \cdot s_y \cdot y + p = 0, \tag{3.5}$$

kde `x` a `y` jsou souřadnice hledaného vrcholu, s_x a s_y jsou souřadnice středu kružnice a `p` je odvozeno ze vztahu

$$p = s_x^2 + s_y^2 - r^2, \tag{3.6}$$

kde `r` je poloměr kružnice. Poloměr kružnice je roven vzdálenosti bodu od senzoru MS Kinect. Tuto vzdálenost získáme při nalezení QR kódu v obraze před samotným výpočtem.

V našem případě máme dvě kružnice, vznikne nám tedy soustava dvou rovnic o dvou neznámých. Naším úkolem je nyní tuto soustavu vyřešit, protože jejím řešením jsou souřadnice průsečíků našich dvou kružnic, přičemž jeden z nich odpovídá námi hledané pozici. Prvním krokem je odstranění druhých mocnin u neznámých. Toho docílíme jednoduchým odečtením druhé rovnice od první.

Obecně tedy dostaneme jednu rovnici ve tvaru

$$(-2 \cdot s_{x1} + 2 \cdot s_{x2}) \cdot x + (-2 \cdot s_{y1} + 2 \cdot s_{y2}) \cdot y + p_1 - p_2 = 0. \tag{3.7}$$

V případě, kdy $s_{x1} = s_{x2}$, nebo $s_{y1} = s_{y2}$, nám po této úpravě zbyde jedna rovnice o jedné neznámé, kterou již není žádný problém vyřešit. Tím získáme hodnotu jedné souřadnice, kterou následně dosadíme zpět do obecné rovnice kružnice (3.5). Na tom, kterou z našich dvou rovnic si vybereme, nezáleží. Výpočet bude fungovat stejným způsobem, ať dosadíme do kterékoliv rovnice. Zvolíme tedy například rovnici první kružnice, do které budeme dosazovat. Tím nám vznikne kvadratická rovnice, jejímž vyřešením získáme také druhou hledanou souřadnici.

Mějme tedy kvadratickou rovnici ve známém tvaru

$$k_1 \cdot x^2 + k_2 \cdot x + k_3 = 0. \tag{3.8}$$

Uvažujme, že z předchozích výpočtů již známe souřadnici `y`. Pak tedy koeficienty kvadratické rovnice odpovídají následujícím vtahům:

$$\begin{aligned} k_1 &= 1, \\ k_2 &= -2 \cdot s_{x1}, \\ k_3 &= y^2 - 2 \cdot s_{y1} \cdot y + p_1, \end{aligned}$$

kde `y` je již vypočtená souřadnice průsečíku a s_{x1}, s_{y1} a $\mathtt{p}_1$ jsou koeficienty z obecné rovnice (3.5) pro první kružnici. Pokud bychom neznali souřadnici `y`, ale souřadnici `x`, byl by výpočet koeficientů obdobný, pouze s tím rozdílem, že bychom z obecné rovnice (3.5) nepoužívali koeficienty pro neznámou `y`, ale použili bychom odpovídající koeficienty pro neznámou `x`.

Pokud ovšem zmíněná rovnost keficientů v rovnici (3.7) neplatí, získáme jednu rovnici o dvou neznámých a postup je mírně složitější. Musíme totiž nejprve vyjádřit jednu neznámou. Uveďme si příklad, kdy vyjádříme neznámou y:

$$y = \frac{(-2 \cdot s_{x1} + 2 \cdot s_{x2}) \cdot x - p_1 + p_2}{(-2 \cdot s_{y1} + 2 \cdot s_{y2})}. \tag{3.9}$$

Takto vyjádřenou neznámou opět dosadíme zpět do obecné rovnice kružnice (3.5). Ani v tomto případě nezáleží, do které z našich dvou rovnic budeme dosazovat, budeme tedy také dosazovat do rovnice první kružnice. Po dosazení získáme opět kvadratickou rovnici. Následující postup je tedy shodný pro oba případy, pouze koeficienty kvadratické rovnice (3.8) budou vypočteny odlišným způsobem. Pokud v rovnici (3.9) provedeme dělení, můžeme ji přepsat do jednoduššího tvaru

$$y = t_1 \cdot x + t_2, \tag{3.10}$$

kde platí následující vztahy:

$$t_1 = \frac{(-2 \cdot s_{x1} + 2 \cdot s_{x2})}{(-2 \cdot s_{y1} + 2 \cdot s_{y2})}$$

$$t_2 = \frac{(-p_1 + p_2)}{(-2 \cdot s_{y1} + 2 \cdot s_{y2})},$$

Zjednodušený tvar 3.10 můžeme využít pro přehlednější vyjádření výpočtu koeficientů kvadratické rovnice, které se v tomto případě počítají následujícím způsobem:

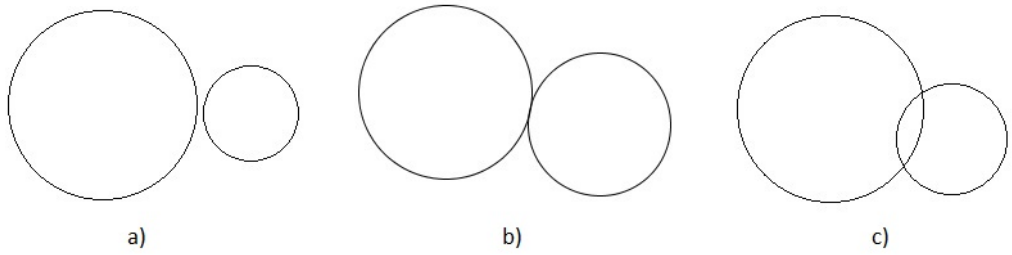
$$\begin{aligned} k_1 &= 1 + t_1^2, \\ k_2 &= -2 \cdot s_{x1} - 2 \cdot t_1 \cdot s_{y1}, \\ k_3 &= t_2^2 - 2 \cdot t_2 \cdot s_{y1} + p_1. \end{aligned}$$

Nyní již tedy máme vytvořenu kvadratickou rovnici a můžeme s její pomocí vypočítat determinant. Nejen, že jej potřebujeme pro určení průsečíků, ale také nám určí, kolik průsečíků námi počítané kružnice mají. Determinant kvadratické rovnice vypočteme ze vztahu

$$D = k_2^2 - 4 \cdot k_1 \cdot k_3, \tag{3.11}$$

kde $\mathtt{k}_1$, $\mathtt{k}_2$, $\mathtt{k}_3$ jsou koeficienty kvadratické rovnice (3.8). Podle výsledku determinantu můžeme rozlišovat 3 různé situace, které jsou znázorněny na obrázku 3.4. První možností je výsledek $D < 0$, který znamená, že kružnice

nemají žádný společný bod. Za předpokladu, že v okolí budou správně rozmístěny QR kódy se správnými hodnotami souřadnic, neměla by tato situace nikdy nastat. Další možností je $D = 0$, kdy kružnice mají jediný společný bod, který by měl odpovídat námi hledané pozici. Poslední a pravděpodobně nejčastější variantou je výsledek $D > 0$, kdy kružnice mají dva společné body. V tomto případě tedy ještě musíme určit, který z těchto průsečíků odpovídá bodu, který hledáme.



Obrázek 3.4: Vzájemné polohy dvou kružnic

Když máme vypočtený determinant, zbývá nám již pouze poslední krok k určení souřadnic průsečíků kružnic. Pokud jsme již po rozdílu obecných rovnic jednotlivých kružnic získali jednu souřadnici průsečíku, je tato souřadnice shodná pro všechny existující průsečíky. Uvažujme, že již známe souřadnici `y` a podle výsledku determinantu jsme zjistili, že existují dva průsečíky. Jejich souřadnice `x` vypočteme tímto vzorcem:

$$x_{1,2} = \frac{-k_2 \pm \sqrt{D}}{2 \cdot k_1}, \tag{3.12}$$

přičemž souřadnice jednotlivých průsečíků jsou $\mathtt{P}_1\mathtt{[x}_1\mathtt{,\ y]}$ a $\mathtt{P}_2\mathtt{[x}_2\mathtt{,\ y]}$.

Pokud jsme ovšem dosud v průběhu výpočtu nezjistili žádnou ze souřadnic průsečíků, vypočteme pomocí vzorce (3.12) jednu ze souřadnic, řekněme souřadnici `x`. Výpočet souřadnice `y` následně provedeme tak, že využijeme rovnici (3.9), do které postupně dosadíme souřadnici `x` všech existujících průsečíků. Tím dostaneme průsečíky se souřadnicemi $\mathtt{P}_1\mathtt{[x}_1\mathtt{,\ y}_1\mathtt{]}$ a $\mathtt{P}_2\mathtt{[x}_2\mathtt{,\ y}_2\mathtt{]}$.

3.4 Určení správného průsečíku

Jak jsme již zmínili v předchozím textu, může nastat situace, kdy námi počítané kružnice mají dva průsečíky. Tyto dva průsečíky jsou našimi potenciálními body, které hledáme. Pouze jeden z nich ovšem může být ten správný. V tomto případě můžeme využít toho, že vidíme v obraze oba pomocné body v QR kódu najednou, tudíž víme, který je vlevo a který vpravo. Pro určení správného průsečíku použijeme výpočet vzájemné polohy bodu a přímky.

Nechť body na přímce označíme `A` a `B`. Kontrolovaný průsečík označíme jako `X`. Nyní určíme směrový vektor přímky $\vec{v}$:

$$\vec{v} = B - A \tag{3.13}$$

a vektor mezi průsečíkem a bodem přímky $\vec{u}$:

$$\vec{u} = X - A. \tag{3.14}$$

Podle [5] pokud tyto dva vektory spojíme do matice

$$\mathbf{M} = \begin{pmatrix} u_x & v_x \\ u_y & v_y \end{pmatrix}, \tag{3.15}$$

tak determinant této matice určí polohu bodu vůči přímce. Vypočteme tedy determinant vzorcem

$$det = u_x \cdot v_y - u_y \cdot v_x. \tag{3.16}$$

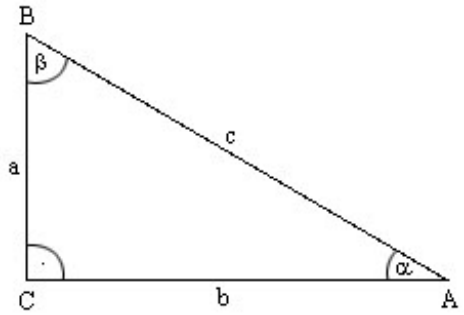
Pokud $det < 0$, pak je bod nalevo od přímky, pokud vyjde $det > 0$, bod je napravo od přímky a pokud $det = 0$, pak bod leží na přímce. V našem případě vždy správný průsečík leží napravo od přímky, protože pokud by ležel nalevo, dívali bychom se na QR kód ze zadní strany a jelikož QR kódy budou umístěny na pevné překážce, byl by to nesmysl.

3.5 Výpočet úhlu natočení robota

Nyní již známe souřadnice robota v prostoru, ale abychom měli kompletní informaci o poloze a mohli s její pomocí následně začít navigovat robota a rozhodnout o jeho dalším pohybu, potřebujeme znát také jeho natočení v prostoru. Tuto informaci bohužel nemůžeme získat přímo, protože robot tento údaj sám neměří. Musíme úhel natočení tedy dopočítat ze vstupních údajů, které máme k dispozici. V našem případě je vstupním zařízením senzor MS Kinect. Z obrazu ale obecně nemůžeme natočení poznat, potřebujeme tedy opět nějaký referenční bod. K tomuto účelu nám opět poslouží využívané QR kódy. Jak již bylo zmíněno dříve, úhel, který je v nich uložen, určuje jejich natočení vůči ose `x`.

Jelikož známe natočení kódu vůči souřadnému systému, můžeme s pomocí úhlu natočení robota vůči QR kódu získat natočení robota vůči souřadnému systému. Musíme ale úhel natočení robota vůči kódu vypočítat. K tomu můžeme využít hloubkovou mapu senzoru MS Kinect. Jak je zmíněno v sekci 3.2, senzor MS Kinect měří hloubku k předmětům vůči rovině senzoru. Námi hledaný úhel tedy můžeme získat pomocí rozdílu hloubky dvou bodů. K tomu opět využijeme referenční body QR kódu umístěné v jeho levém a pravém horním rohu. Známe totiž jejich rozteč.

Když postup znázorníme geometricky, jde o pravoúhlý trojúhelník (viz obr. 3.5). Použité referenční body reprezentují vrcholy `A` a `B`. Jejich rozteč je tedy délka strany `c`. Když uvážíme, že strana `b` je rovnoběžná s rovinou senzoru, vůči které je měřena hloubka, pak rozdíl mezi hloubkami bodů `A` a `B` je roven straně `a`. Jelikož díky hloubkové mapě tyto body známe, známe také délku strany `a`. Úhel α je tedy roven úhlu natočení robota vůči kódu.



Obrázek 3.5: Pravoúhlý trojúhelník

Pro výpočet úhlu využijeme geometrickou funkci sinus:

$$\sin \alpha = \frac{a}{c}. \tag{3.17}$$

Toto ovšem ještě není výsledek, protože naším cílem je úhel natočení robota vůči souřadnému systému. Jelikož robot musí na daný kód vidět, tak pokud by stál rovnoběžně s kódem, byl by vůči němu o 180 ° otočen. Označíme-li tedy úhel natočení kódu jako β, úhel natočení robota vůči kódu je α a výsledný námi hledaný úhel γ, pak bychom výsledný úhel získali ze vztahu

$$\gamma = (\beta + 180° + \alpha) mod 360°. \tag{3.18}$$

Modulo 360 ° je zde použito proto, aby nevznikl úhel větší, než je úhel celého kruhu.

4 Navigace robota

Aby se mohl robot z výchozí pozice přesunout do požadované cílové pozice, musí být schopen pohybovat se prostorem a určovat svojí pozici. Je tudíž implementováno několik pravidel, podle kterých se rozhoduje o dalším pohybu robota. Celou navigaci robota můžeme shrnout do dvou velkých skupin.

4.1 Hledání kódu

První skupina popisuje rozhodování, pokud robot hledá ve svém okolí QR kódy a snaží se určit svoji polohu.

4.1.1 Rozhlédnutí

Rozhlédnutí je krok, který je proveden na začátku každé vyhledávací sekvence a také po každém posunutí během vyhledávání. Jeho cílem je nalézt v okolí robota QR kód, podle kterého by se dala určit poloha robota. Je mu zadána rotace směrem vpravo, tedy ve směru hodinových ručiček. Pokud není nalezen kód, končí otáčení po dovršení otočení o 360 °.

4.1.2 Přesun

Pokud nebyl během rozhlížení nalezen žádný kód, zjevně v okolí robota žádný není a je potřeba hledat jinde. Proto po dokončení otáčení pokračuje robot jízdou vpřed ve směru, ve kterém bylo ukončeno otáčení. Tímto pohybem vpřed urazí robot vzdálenost, která by podle předpokladu odpovídala jednomu metru a nenarazí-li při přesunu na kód, pokračuje dalším rozhlédnutím.

4.1.3 Objevení překážky

Při pohybu vpřed je potřeba kontrolovat, aby robot nenarazil do překážky. K tomu je využita hloubková mapa senzoru MS Kinect, ve které se vyhledávají příliš blízké předměty. Pokud je v obraze takový předmět nalezen, je pohyb robota zastaven a pokračuje otáčením a hledáním volné trasy. Jakmile překážka z obrazu zmizí a podle hloubkové mapy je cesta volná, začne se robot pohybovat tímto směrem do předpokládané vzdálenosti půl metru a následuje opět rozhlížení.

4.1.4 Nalezení kódu

Pokud je v obraze nalezen kód, s jehož pomocí se podařilo najít polohu robota v prostoru, ukončuje se fáze vyhledávání a přechází se do fáze posunu k cíli. Prvním krokem je natočení.

4.2 Posun k cíli

Druhá skupina popisuje rozhodování o pohybu v případě, že robot zjistil svoji pozici a nyní se snaží přesunout do zadané cílové pozice.

4.2.1 Natočení

Natočení je první krok fáze posunu k cíli. Když známe polohu robota a víme, na které místo se chceme posunout, známe směr příštího pohybu robota. Současnou hodnotu natočení robota také známe, takže víme, o jaký úhel bychom měli robota otočit. Nastaví se tedy čas odpovídající otočení robota o požadovaný úhel a zahájí se natáčení. Před zahájením natáčení je také vyhodnoceno, kterým směrem je výhodnější se natočit. Rozhodovací podmínkou je porovnání rozdílů:

$$diff_1 = targetAngle - robotAngle$$
$$diff_2 = targetAngle - (robotAngle + 360°),$$

kde `targetAngle` je úhel, do kterého se chceme natočit a `robotAngle` je aktuální úhel natočení robota. Pokud $diff_1 <= diff_2$, rozhoduje se podle velikosti $diff_1$. Je-li jeho hodnota záporná, je výhodnější otáčet se doprava, jinak se otáčí vlevo. Pokud ale $diff_2 < diff_1$, je výhodnější točit doprava. Jako příklad si uveďme situaci, kdy $targetAngle = 270°$ a $robotAngle = 0°$. Pokud by se robot například vždy točil pouze doleva (uvažoval by se pouze rozdíl $diff_1$), musel by se v tomto případě otočit o 270 °, zatímco točením doprava by mu stačilo pouze 90 ° (získáno z rozdílu $diff_2$).

4.2.2 Pohyb vpřed

Pokud vypršel čas, po který se měl robot otáčet, aby se nasměroval do žádaného směru, rozjede se robot rovně v daném směru. Spustí se odpočet času rovného pohybu, který by měl odpovídat ujetí požadované vzdálenosti v tomto směru.

4.2.3 Objevení překážky

Postup při objevení překážky je totožný, jako ve fázi vyhledávání pozice. Pokud narazíme na překážku ve fázi posunu k cíli, tak po vyhnutí se překážce přechází robot opět do fáze vyhledávání.

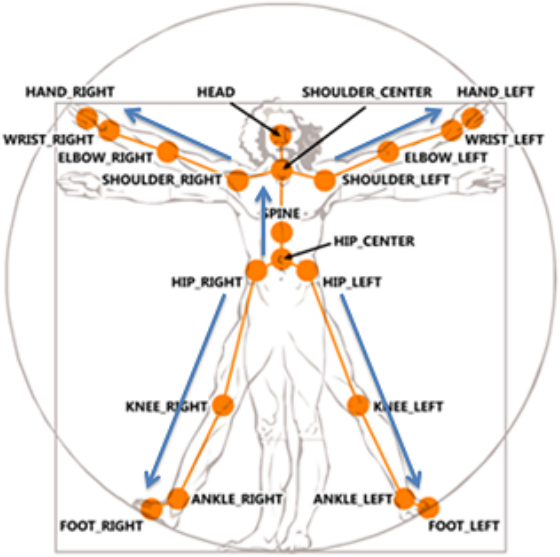
4.2.4 Dokončení pohybu vpřed

Pokud uplyne přednastavená doba, po kterou měl robot jet rovně v zadaném směru, zastaví se a přejde do fáze vyhledávání, když se začne rozhlížet a hledat kód, kterým by ověřil svou novou pozici.

5 Interakce s člověkem

Součástí práce je také schopnost robota rozeznat člověka a případně reagovat na jeho gesta. Rozpoznávání člověka je implementováno již přímo v SDK pro senzor MS Kinect. Stačí pouze při spouštění povolit `SkeletonStream`. Senzor MS Kinect umožňuje sledovat více lidí najednou, pro účely této práce však stačí sledování jediné postavy.

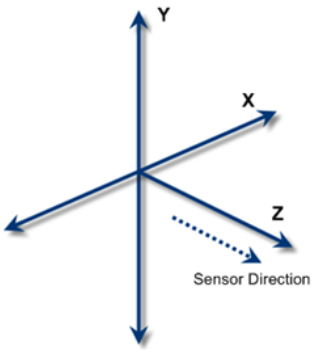
Senzor MS Kinect reprezentuje člověka jako kostru složenou z 20 bodů (viz obr. 5.1). S pomocí těchto bodů a jejich vzájemné polohy lze rozeznávat jednotlivá gesta.



Obrázek 5.1: Reprezentace lidského těla senzorem MS Kinect. Převzato z [7]

Každý bod kostry je reprezentován souřadnicemi `x`, `y` a `z`. Obrázek 5.2 popisuje souřadný systém pro sledování kostry. Ten totiž není shodný se systémem pro hloubkový a barevný obraz. Souřadnice v systému pro sledování kostry jsou reprezentovány v metrech. Senzor je umístěn v počátku tohoto souřadného systému a ve směru, kterým je namířena kamera, roste kladná hodnota osy `z`. Budeme-li tedy pozice v tomto systému reprezentovat trojicí (`x`,`y`,`z`), pak bod `(0,0,z)` je umístěn ve středu obrazu. Při pohledu ve směru senzoru pak od středu obrazu směrem vlevo roste kladná hodnota osy `x` a na-

opak vpravo klesá do záporných hodnot. U osy `y` pak jsou rostoucí kladné hodnoty v horní polovině obrazu a záporné v dolní polovině.



Obrázek 5.2: Souřadný systém pro sledování lidské kostry. Převzato z [7]

5.1 Rozpoznání gesta

Abychom mohli rozpoznávat gesta, potřebujeme znát vzájemnou polohu určitých bodů kostry, které se na daném gestu podílejí. Vzájemnou pozici můžeme zkoumat například porovnáváním souřadnic jednotlivých bodů, ale vzhledem k nutnosti brát v potaz jistou odchylku, není to příliš pohodlné řešení. Mnohem lepším, ačkoliv matematicky o něco náročnějším, řešením je vyhodnocovat úhel, který svírají tři body kostry. Je vhodné využít gesta, která nejsou pro lidský pohyb zcela běžná, aby nedocházelo ke zbytečným neúmyslným rozpoznáním.

Uveďme si tedy příklad, kdy budeme sledovat vzájemnou polohu bodů `HandRight` (označme jej jako bod `A`), `ShoulderRight` (označme jej jako bod `B`) a `Spine` (označme jej jako bod `C`). Tyto body můžeme spojit do vektoru $\vec{u}$, který bude reprezentovat úhel mezi ramenem a rukou:

$$\vec{u} = (A_x - B_x, A_y - B_y) \tag{5.1}$$

a vektoru $\vec{v}$, který bude reprezentovat úhel mezi ramenem a páteří:

$$\vec{v} = (C_x - B_x, C_y - B_y). \tag{5.2}$$

Body kostry jsou sice v trojrozměrném prostoru, ale osu `z` pro tyto účely zanedbáme, protože nám pro rozpoznání gesta stačí vzájemná poloha v rovině. Úhel mezi těmito vektory získáme ze vzorce odvozeného ze skalárního součinu dvou vektorů:

$$\cos\alpha = \frac{\vec{u} * \vec{v}}{|\vec{u}| * |\vec{v}|}. \tag{5.3}$$

Když z tohoto vzorce vyjádříme úhel α, získáme hledanou hodnotu a můžeme ji porovnat, zda odpovídá určitému gestu.

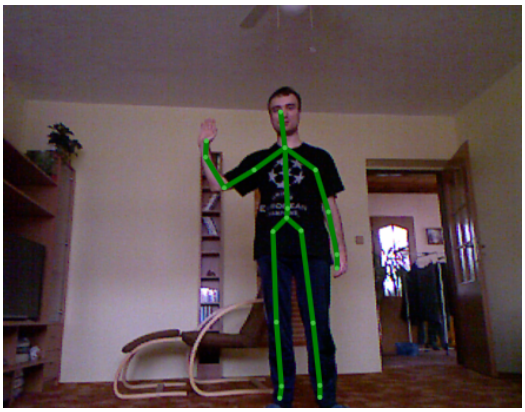
5.2 Implementovaná gesta

V projektu je implementováno několik základních gest, jejich počet lze snadno navýšit o nová gesta. Přiložené obrázky jsou snímky pořízené senzorem MS Kinect. Senzor zachytává obraz zrcadlově obrácený, takže zde přiložené obrázky byly převráceny vůči svislé ose, aby bylo pro člověka snáze rozpoznatelné, která část těla je levá a která pravá. V obrázku je také zeleně vyznačena kostra člověka, jak jí snímá senzor MS Kinect.

5.2.1 Vstupní gesto

Vstupní gesto slouží k zahájení sledování člověka. Dokud robot toto gesto nezaregistruje, tak na žádná jiná gesta nereaguje a jakmile toto gesto rozpozná, začne sledovat člověka a ostatní činnosti přeruší. Vstupní gesto je nejsložitější ze všech implementovaných gest, aby se co nejvíce omezilo riziko nechtěného spuštění sledování.

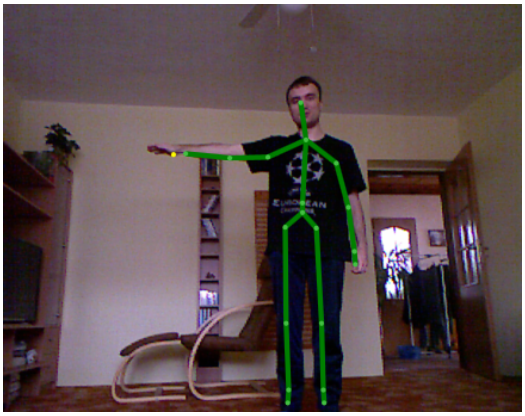
Pro rozpoznání tohoto gesta jsou kontrolovány dva úhly. První svírají body `Spine`, `ShoulderCenter` a `ElbowRight`. Hodnota úhlu se musí pohybovat v rozmezí 50 ° až 90 °. Druhý úhel svírají body `ShoulderCenter`, `ElbowRight` a `HandRight`. Hodnota tohoto úhlu se také musí pohybovat v rozmezí 50 ° až 90 °. Na obrázku 5.3 lze vidět ukázku tohoto gesta.



Obrázek 5.3: Vstupní gesto

5.2.2 Gesta pro otočení vpravo a vlevo

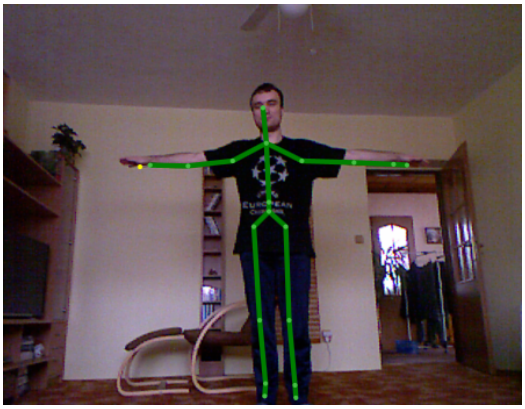
Tato gesta dávají robotovi pokyn, aby se otočil o 360 ° točením ve směru určeném zvednutou rukou. Gesta se liší pouze využitím pravé, případně levé ruky. Popíšeme si příklad s využitím pravé ruky. Toto gesto je opět určeno pozorováním dvou úhlů. První svírají body `Spine`, `ShoulderCenter` a `ElbowRight`. Hodnota úhlu se musí pohybovat v rozmezí 60 ° až 90 °. Druhý úhel svírají body `Spine`, `ShoulderCenter` a `ElbowLeft`. Hodnota tohoto úhlu musí být menší než 60 °. Jde tedy o gesto, kdy pravá ruka je zvednuta přibližně do výše ramen a levá ruka je připažená. Na obrázku 5.4 lze vidět ukázku tohoto gesta.



Obrázek 5.4: Gesto pro otočení robota směrem vpravo

5.2.3 Výstupní gesto

Posledním implementovaným gestem je výstupní gesto, po jehož rozeznání robot ukončuje sledování člověka a vrací se zpět k přesunutí do cílového bodu zadaného uživatelem. Také toto gesto je tedy určeno pozorováním dvou úhlů. První svírají body `Spine`, `ShoulderCenter` a `ElbowLeft`. Hodnota úhlu se musí pohybovat v rozmezí 60 ° až 90 °. Druhý úhel svírají body `Spine`, `ShoulderCenter` a `ElbowRight`. Hodnota tohoto úhlu musí být také v rozmezí 60 ° až 90 °. Jde tedy o gesto, kdy jsou obě ruce zvednuty přibližně do výše ramen. Na obrázku 5.5 lze vidět ukázku tohoto gesta.



Obrázek 5.5: Výstupní gesto

5.3 Sledování člověka

Pokud je robotem sledován člověk a není právě prováděna činnost určená rozpoznaným gestem, snaží se robot udržet člověka uprostřed obrazu a také udržovat vzdálenost od člověka. Pokud se tedy bod kostry `Spine` vychýlí od středu obrazu o hodnotu větší než 0.1 v ose `x`, začne se robot tímto směrem natáčet, aby kompenzoval posun člověka a vrátil jej zpět do středu obrazu. Při zahájení sledování člověka je také poznamenána hodnota osy `z` pro bod kostry `Spine`. Pokud dojde ke zvětšní aktuální hodnoty nad poznamenanou hodnotu, začne se robot posouvat vpřed, dokud aktuální hodnota opět nebude nižší. Hodnoty používané pro tuto korekci jsou souřadnice bodu ze souřadného systému kostry popsaného v sekci 5.

6 Struktura systému

Aby byla dodržena struktura platformy, která je popsána v [1], je i tato práce vytvořena a rozdělena do více systémů. Platforma je rozdělena do systémů serveru, klienta a robota, takže stejné rozdělení musí mít také uživatelský program. Každý systém využívá moduly, které jsou vytvořeny jako samostatné knihovny, jejichž propojení je definováno konfiguračním XML souborem. Systémy mezi sebou navzájem komunikují pomocí počítačové sítě TCP. Z toho plyne nutnost nadefinování zpráv, které budou v komunikaci využívány. Samotnou komunikaci mezi systémy zajišťuje platforma Kinbo[2], uživatel pouze musí nadefinovat vstupy a výstupy systémů a jejich propojení s moduly daných systémů.

6.1 Použité zprávy

Jelikož mezi sebou systémy komunikují pomocí počítačové sítě, je nutné nadefinovat zprávy pro vzájemnou komunikaci. Aby ovšem režie posílání zpráv nezabírala příliš mnoho času, je definováno pouze pár základních zpráv. Další zprávy je možné snadno při dalším vývoji doplnit. Dalším důvodem, proč není definováno větší množství zpráv, je ten, že veškerá výpočetní logika je prováděna pouze na straně robota bez spolupráce zbylých systémů.

6.1.1 Zpráva se souřadnicemi

Jedná se o základní zprávu projektu, ve které jsou odeslány zadané cílové souřadnice pro systém robota. Podporovány jsou pouze souřadnice zadané celým číslem. Typ zprávy je `MyKinboMessage`, který byl převzat z programové přílohy práce [1] a formát zprávy je následující:

<souradniceX>,<souradniceY>; <cil>

`<souradniceX>` označuje zadanou souřadnici `x`, `<souradniceY>` označuje zadanou souřadnici `y` a `<cil>` označuje cílový systém, kterému je zpráva určena. Zpráva je posílána ze systému klienta do systému robota. Pokud

je tato zpráva přijata v systému robota před spuštěním samotné činnosti systému, signalizuje požadavek na spuštění. Pokud již činnost robota byla spuštěna, jsou pouze přenastaveny cílové souřadnice.

6.1.2 Zpráva o chybě

Jde o zprávu, kdy robot klientovi oznamuje neúspěšný pokus o spuštění senzoru MS Kinect. Tato chyba může nastat, pokud není žádný senzor připojen, nebo připojený senzor není připraven k použití. Typ zprávy je `MyKinboMessage` a její formát je:

KinectMissing; <cil>

`<cil>` označuje cílový systém, v tomto případě klienta.

6.1.3 Zpráva pro motory

Další důležitou zprávou je zpráva typu `RobotMotorsMessage`. Tento typ je definován v platformě Kinbo[2] a slouží k nastavení výkonů jednotlivých motorů robota. Formát zprávy je:

L: <levyMotor>; R: <pravyMotor>

`<levyMotor>` označuje nastavovanou hodnotu výkonu levého motoru a `<pravyMotor>` označuje nastavovanou hodnotu výkonu pravého motoru. Hodnoty jsou uvedeny v desetinných číslech a mohou nabývat hodnot z intervalu <-1;1>. Tato zpráva je odesílána systémem robota platformě Kinbo[2], která již zajišťuje samotné nastavení motorů.

6.1.4 Zpráva o dosažení cíle

Jde o zprávu typu `MyKinboMessage`, kterou robot odesílá klientovi poté, co dosáhl požadovaných souřadnic. Její formát je:

Done; <cil>

`<cil>` označuje cílový systém, v tomto případě klienta.

6.1.5 Zpráva o chybných souřadnicích

Tato zpráva typu `MyKinboMessage` je zasílána robotem klientovi, pokud přijaté souřadnice nejsou čísla. Formát zprávy je:

badCoords; <cil>

`<cil>` označuje cílový systém.

6.1.6 Ukončovací zpráva

Tato zpráva typu `MyKinboMessage` je zasílána klientem robotovi při žádosti o ukončení běhu. Stejnou zprávu následně posílá robot klientovi zpět jako potvrzení. Po přijetí této zprávy jsou oba systémy ukončeny. Formát zprávy je:

exit; <cil>

`<cil>` označuje cílový systém.

6.2 Systém serveru

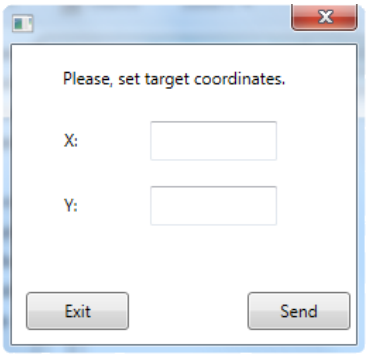
Obecně systém serveru v projektech běžících s platformou Kinbo[2] zajišťuje spojení mezi klientem a robotem. Jeho úkolem je také vykonávat výpočetní

logiku náročnějších operací, protože výkon počítače na straně robota je limitován. V projektu této bakalářské práce se však předpokládá, že počítač robota dokáže výpočetní logiku této práce zvládnout, tudíž server zde plní pouze první zmíněný účel, a sice přeposílá zprávy mezi klientem a robotem. Díky tomu nebylo nutné implementovat vlastní server, protože tuto funkci dokáže poskytnout již samotná platforma Kinbo[2]. Pokud navíc uvážíme, že senzor MS Kinect snímá 30 obrázků za vteřinu, tak pouze komunikace přeposílající veškerá tato data mezi robotem a serverem by zabrala velké množství času.

6.3 Systém klienta

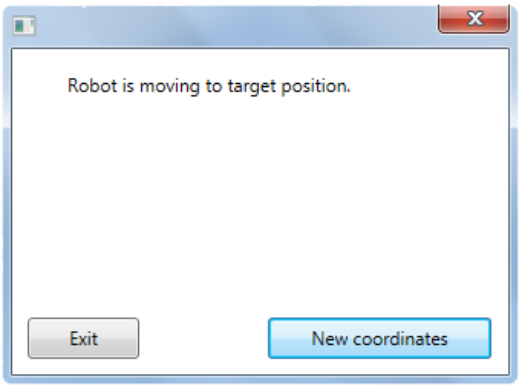
Systém klienta nemá v tomto projektu příliš možností. Jeho úkolem je zadat robotovi cílové souřadnice, na které se má přesunout. Tyto souřadnice může kdykoli za běhu programu změnit. Uživatel pomocí systému robota může také kdykoliv vyvolat ukončení programu. Pro tyto činnosti má uživatel k dispozici jednoduché uživatelské rozhraní.

Po spuštění projektu se objeví úvodní okno uživatelského rozhraní klienta (viz obr. 6.1). Jsou zde dvě vstupní pole pro zadání cílových souřadnic, na které má robot dojet. Zadávat lze pouze celá čísla, jinak modul klienta oznámí, že vstupní hodnoty nejsou v pořádku. Pod vstupními poli jsou dvě tlačítka. Tlačítkem `Send` potvrdíme zadané souřadnice a pokud projdou kontrolou, jsou odeslány do systému robota. Tlačítkem `Exit` můžeme kdykoliv ukončit činnost robota. Po stisknutí tohoto tlačítka je robotovi odeslána ukončovací zpráva a je ukončen i běh klienta.



Obrázek 6.1: Uživatelské rozhraní klienta, zadávací okno

Pokud kontrola zadaných souřadnic proběhne v pořádku, změní se vzhled okna (viz obr. 6.2). Zmizí zadávací pole pro souřadnice a tlačítko `Send` se změní na tlačítko `New coordinates`, které když stiskneme, objeví se opět předchozí okno a můžeme zadat nové souřadnice pro robota. Tlačítko `Exit` má i v tomto okně stejnou funkci.



Obrázek 6.2: Uživatelské rozhraní klienta, informační okno

6.4 Systém robota

Jak je již zmíněno u systému serveru, na výpočetní náročnosti této práce by měl výkon počítače robota postačovat, takže veškeré výpočty jsou prováděny v tomto systému. Po spuštění systém čeká na povinný vstup, kterým jsou cílové souřadnice, na které má robot dojet. Tyto souřadnice jsou od klienta přeposlány pomocí serveru. Po jejich dekódování teprve dojde ke spuštění samotné činnosti robota a senzoru MS Kinect. S pomocí senzoru je snímáno okolí a jsou v něm vyhledávány QR kódy potřebné pro navigaci robota v prostoru. Sledován je také výskyt lidí, kteří chtějí interagovat s robotem a při přesunech robota také výskyt překážek, do kterých by mohl robot narazit. Nedílnou součástí systému je také navigační část popsaná v kapitole 4. Na rozdíl od systému klienta je systém robota pouze s konzolovým výstupem bez grafického rozhraní.

6.5 Propojení modulů

Propojení systémů je vytvořeno v platformě Kinbo[2], ale propojení uvnitř jednotlivých systémů musí vytvořit uživatel sám. Slouží k tomu konfigurační XML soubor, jehož struktura je popsána v [1]. Pokud se podíváme na strukturu konfiguračního XML pro projekt z této práce, najdeme v ní definovány pouze systémy klienta a robota. Důvodem je již zmíněný fakt, že systém serveru nebylo třeba implementovat.

Když si přiblížíme systém klienta (obr. 6.3), vidíme nejprve použité knihovny. Knihovna `ClientModule.dll` obsahuje uživatelský modul pro systém klienta, zbylé knihovny náleží platformě Kinbo[2]. Dále vidíme definovaný výstup systému `outRobot`, který slouží k odeslání zprávy ze systému klienta do systému robota a vstup systému `input`, na který jsou přivedeny příchozí zprávy z jiných systémů. Následuje definování modulů, v tomto případě pouze jednoho, který řídí činnost klienta. Na závěr jsou zde definice propojení. Vstup systému je napojen na vstup modulu a naopak výstup modulu je napojen na výstup systému.

```
<System name="client">
  <Assemblies>
    <Assembly file="ClientModule.dll" />
    <Assembly file="RobotKinbo2Modules.dll" />
    <Assembly file="ModularPlatform.dll" />
    <Assembly file="KinboSDK.dll" />
  </Assemblies>
  <Subsystem name="Main">
    <Outputs>
      <Output message="KinboMessage" name="outRobot" />
    </Outputs>
    <Inputs>
      <Input message="KinboMessage" name="input" />
    </Inputs>
    <Children>
      <Module type="ClientModule" name="ClientModule0">
        <Design x="111" y="123" width="200" height="150" />
      </Module>
    </Children>
    <Connections>
      <Connection from="ClientModule0.OutRobot" to="this.outRobot" />
      <Connection from="this.input" to="ClientModule0.In" />
    </Connections>
  </Subsystem>
</System>
```

Obrázek 6.3: Konfigurace systému klienta

Když si přiblížíme také systém robota (obr. 6.4), vidíme, že struktura je velmi podobná. Nejprve opět definice knihoven, `RobotModule.dll` obsahuje uživatelský modul pro systém robota a zbylé knihovny jsou totožné jako u klienta. Následují opět výstupy systému, `outClient` slouží pro odesílání zpráv pro systém klienta a `motors` je výstup pro nastavování hodnot motorům. Tento výstup je napojen na platformu Kinbo[2]. Na vstup systému `input` jsou poté opět přivedeny příchozí zprávy z jiných systémů. I v tomto systému je definován pouze jeden modul, který po přijetí vstupních dat spouští proces robota. Na závěr jsou zde opět definice propojení. Vstup systému je napojen na vstup modulu a naopak výstup modulu je napojen na výstup systému. Výstup modulu odesílající zprávy motorům je napojen na výstup systému `motors`.

```
<System name="robot">
  <Assemblies>
    <Assembly file="RobotModule.dll" />
    <Assembly file="RobotKinbo2Modules.dll" />
    <Assembly file="ModularPlatform.dll" />
    <Assembly file="KinboSDK.dll" />
  </Assemblies>
  <Subsystem name="Main">
    <Outputs>
      <Output message="KinboMessage" name="outClient" />
      <Output message="RobotMotorsMessage" name="motors" />
    </Outputs>
    <Inputs>
      <Input message="KinboMessage" name="input" />
    </Inputs>
    <Children>
      <Module type="RobotModule" name="RobotModule0">
        <Design x="111" y="123" width="200" height="150" />
      </Module>
    </Children>
    <Connections>
      <Connection from="RobotModule0.OutClient" to="this.outClient" />
      <Connection from="RobotModule0.Motors" to="this.motors" />
      <Connection from="this.input" to="RobotModule0.In" />
    </Connections>
  </Subsystem>
</System>
```

Obrázek 6.4: Konfigurace systému robota

7 Popis běhu programu

Nyní si uvedeme příklad, jak může vypadat průběh programu od spuštění až po dojetí robota na požadovanou pozici.

Po spuštění pomocí Runtime aplikace patřící k platformě Kinbo[2] a zadání konfiguračního XML souboru se spustí uživatelský projekt. Kromě konzolových okének jednotlivých systémů se objeví také grafické okénko klienta, do kterého se zadávají souřadnice, na které má robot dojet. Po zadání a potvrzení souřadnic dojde k odeslání souřadnic do systému robota. V něm jsou dekódovány a uloženy. Po přijetí souřadnic se teprve spustí samotná činnost robota. Dojde tedy k inicializaci a spuštění senzoru MS Kinect a je zahájeno zpracovávání snímků pořízených tímto senzorem.

Prvním zkoumaným faktorem při každém novém snímku je přítomnost člověka v obraze. Pokud je člověk nalezen, zkoumá se, zda robotovi signalizuje implementované gesto. Pokud je gesto rozpoznáno, spustí se určitá činnost, která je s daným gestem spojena. Aby se snížilo riziko falešného rozpoznání člověka, nebo reagování na člověka v obraze, který nechce interagovat s robotem, je nutné nejprve robotovi signalizovat takzvané vstupní gesto, které je blíže popsáno v kapitole 5. Po rozpoznání vstupního gesta se robot soustředí pouze na daného člověka a reaguje na jeho povely až do doby, kdy je rozpoznáno výstupní gesto, nebo člověk nezmizí z obrazu. V tento moment se robot vrací k původní činnosti, tedy ke snaze dojet na zadané souřadnice.

Pokud neprobíhá interakce s člověkem, je zkoumán načtený barevný snímek, ve kterém je vyhledáván QR kód, s jehož pomocí by robot dokázal určit svoji polohu. Pokud není v obraze nalezen žádný kód, z něhož by byl robot schopen rozpoznat svoji pozici, je zahájeno prohledávání okolí. Prvním krokem je spuštění otáčení robota ve směru hodinových ručiček, aby prohlédl své okolí a pokusil se v něm najít potřebný QR kód. Pokud se mu žádný kód nepodaří nalézt ani po otočení o 360 °, je otáčení zastaveno a robot se posune o metr dopředu, kde se opět bude rozhlížet. Tímto způsobem pokračuje do doby, než se mu podaří nalézt kód, nebo dokud není přerušen interagujícím člověkem.

Je-li v obraze nalezen kód, je s jeho pomocí určena robotova poloha. Ta je porovnána s požadovanou cílovou polohou a podle výsledku je zvolen směr, kterým robot bude pokračovat dále. Robot se natáčí do přímého směru k cíli. Po natočení do správného směru je tedy robotovi nastaveno, že má jet

rovně po dobu, která by mu dle odhadu měla stačit na dojetí k cíli. Během takovéhoto přesunu robot nové kódy nevyhledává. Po dojetí na konec takto zadané trasy se robot opět rozhlíží a hledá kód, s jehož pomocí by dokázal určit svoji aktuální polohu. V případě, že robot dojel do požadované cílové pozice, přeruší svoji činnost a ohlásí klientovi dokončení činnosti.

Při každém pohybu vpřed, tedy při vyhledávání kódu i při cíleném přesunu na odhadovanou správnou pozici, robot také v každém snímku kontroluje okolí, zda se mu v obraze neobjevila překážka, do které by mohl narazit. Pokud takovou překážku najde, zastaví se a začne se otáčet do doby, dokud překážka z obrazu nezmizí. Následně se posune o půl metru vpřed ve směru, ve kterém již nevidí překážku, a opět spustí prohlížení okolí a vyhledávání kódu pro určení své nové pozice. Při otáčení se na místě není tato kontrola překážek aktivní, protože díky tankovému podvozku by robot měl být schopen otočit se prakticky na místě.

8 Dosažené výsledky

Bohužel kvůli technickým potížím nebyl k dispozici funkční robot Kinbo, na kterém bych mohl vyzkoušet běh celé práce. Uvedu zde tedy alespoň test přesnosti výpočtu pozice robota v prostoru, pro jehož provedení postačil senzor MS Kinect. Test je proveden na dvou různých QR kódech. V tabulce 8.1 jsou uvedeny naměřené hodnoty s QR kódem `300,300,0`. První sloupec označuje skutečné suřadnice, na kterých byl umístěn senzor, ve druhém sloupci jsou uvedeny vypočtené souřadnice a poslední sloupec obsahuje odchylku těchto dvou pozic. Souřadnice i odchylka jsou uvedeny v centimetrech.

Tabulka 8.1: Naměřené hodnoty s QR kódem 300,300,0

Skutečná pozice	Vypočtená pozice	Odchylka [cm]
300; 200	298,2; 198,4	2,41
290; 200	291,86; 198,33	2,5
310; 200	308,47; 199,13	1,76
310; 180	307,56; 179,02	2,63
300; 180	294,83; 178,9	5,29
290; 180	288,14; 179,35	1,97

V tabulce 8.2 jsou uvedeny naměřené hodnoty s QR kódem `100,100,180`. Struktura tabulky je totožná jako v předchozím případě.

Tabulka 8.2: Naměřené hodnoty s QR kódem 100,100,180

Skutečná pozice	Vypočtená pozice	Odchylka [cm]
100; 200	102,51; 203,67	4,45
90; 200	90,99; 203,26	3,41
110; 200	108,15; 202,27	2,93
110; 220	113,53; 221,64	3,89
100; 220	98,83; 221,2	1,68
90; 220	91,81; 222,06	2,74

Jak je z tabulek vidět, odchylka nepřesahuje 10 centimetrů, což je pro tuto práci dostačující přesnost. Vytvořený algoritmus pro nalezení pozice robota je tedy podle provedených testů dostatečně kvalitní.

9 Závěr

V rámci této práce byly popsány postupy pro určení pozice robota z nalezeného QR kódu a následná pravidla pro pohyb robota prostorem. V práci byl také popsán způsob rozpoznávání člověka a jím signalizovaných gest. Všechny tyto uvedené postupy byly také implementovány do výsledné práce. Odchylka vypočtené pozice robota vůči reálné pozici se během testů pohybovala do 10 centimetrů, což je pro tento účel dostatečná přesnost.

Cíl této práce, vytvořit moduly pro určení pozice robota a interakci s člověkem, byl dosažen vytvořením jednotlivých modulů obsahujících funkcionalitu popsaných problematik. Moduly jsou reprezentovány jako dll knihovny, které lze snadno pomocí konfiguračního XML souboru propojit s platformou Kinbo[2].

Bohužel vzhledem k tomu, že robot nebyl technicky připraven, nebylo možné na něm práci otestovat, takže veškeré testy mohly probíhat pouze na části práce, které pracovaly s obrazem zaznamenaným senzorem MS Kinect. Nebylo tedy možné otestovat především navigační část pro pohyb robota v prostoru.

Přehled zkratek

QR kód - „Quick Response“ kód, tedy kód rychlé reakce

MS - Microsoft

RGB - Red, Green, Blue. Zkratka berevného modelu, kde se všechny barvy skládají ze tří základních barev: červená-zelená-modrá

SDK - software development kit

XML - Extensible Markup Language, značkovací jazyk

TCP - Transmission Control Protocol

Literatura

[1] ALTMAN, P. *Modulární vývojová platforma Kinbo*^{2} Plzeň, 2013. Oborový projekt na Fakultě aplikovaných věd Západočeské univerzity na katedře informatiky a výpočetní techniky. Vedoucí projektu Ing. Petr Vaněček, Ph.D.

[2] *ZXing.Net* [počítačový soubor]. Ver. 0.12.0.0. 2014 [cit. 4.2014]. http://zxingnet.codeplex.com/releases/view/106588

[3] *Microsoft Kinect SDK* [počítačový soubor]. Ver. 1.8 for Windows 2013 [cit. 4.2014]. http://www.microsoft.com/en-us/kinectforwindowsdev/Downloads.aspx

[4] Kinect for Windows Sensor *Microsoft Developer Network* [online]. 2014 [cit. 4.2014]. http://msdn.microsoft.com/en-us/library/hh855355.aspx

[5] *Korespondenční seminář z programování* [online]. 2012 [cit. 4.2014]. http://ksp.mff.cuni.cz/tasks/24/cook5.html

[6] ANDERSEN, M.R. - JENSEN, T. - LISOUSKI, P. - MORTENSEN, A.K. - HANSEN, M.K. - GREGERSEN, T. - AHRENDT, P. *Kinect Depth Sensor Evaluation for Computer Vision Applications* [online]. 2012 [cit. 4.2014]. http://eng.au.dk/fileadmin/DJF/ENG/PDF-filer/Tekniske_rapporter/Technical_Report_ECE-TR-6-samlet.pdf

[7] Skeletal Tracking *Microsoft Developer Network* [online]. 2014 [cit. 4.2014]. http://msdn.microsoft.com/en-us/library/hh973074.aspx

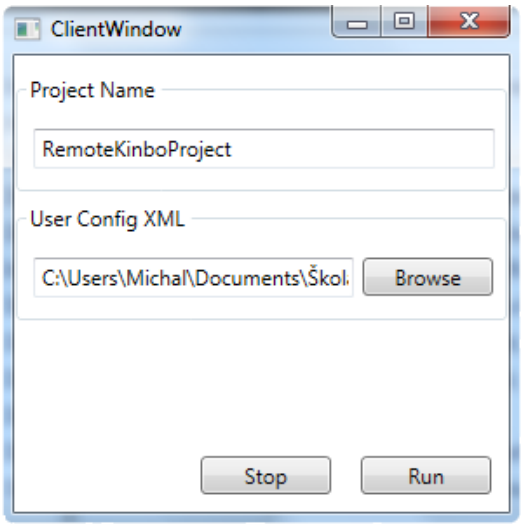
[8] *QR code generator* [online]. 2014 [cit. 5.2014]. http://goqr.me

[9] POSPÍŠIL, A. - PŘINOSIL, J. - ŘÍHA, K. *Detekce a sledování polohy hlavy ve video sekvencích s využitím zařízení Microsoft Kinect* [PDF]. Publikováno 3.1.2012 [cit. 5.2014]. Dostupné z: http://www.elektrorevue.cz/cz/clanky/zpracovani-signalu/5/detekce-a-sledovani-polohy-hlavy-ve-video-sekvencich-s-vyuzitim-zarizeni-microsoft-kinect/

A Příloha

A.1 Uživatelská příručka

Ke spuštění uživatelského projektu vytvořeného v této práci musí být nejprve spuštěny systémy klienta, serveru i robota platformy Kinbo[2]. K tomu je zapotřebí použít runtime aplikaci, která je součástí modulární platformy. Pro snažší spuštění jsou u platformy vytvořeny pro jednotlivé systémy také dávkové soubory. Systémy lze tedy jednoduše spustit soubory `server.bat`, `client.bat` a `robot.bat`. Pokud je platforma spouštěna pouze na lokálním počítači, lze jednoduše spustit všechny tři systémy dávkovým souborem `__lokalKinbo.bat`. Po spuštění systému klienta se objeví spouštěcí okno uživatelského projektu (viz obr. A.1). Tlačítkem `Browse` vybereme konfigurační XML uživatelského projektu, který chceme spustit a potvrdíme volbu tlačítkem `Run`. Pokud jsou k dispozici všechny potřebné knihovny, je spuštěn uživatelský projekt a objeví se uživatelské rozhraní klienta ze spuštěného projektu, jehož ovládání je popsáno v sekci 6.3. Po zadání cílových souřadnic je spuštěna činnost robota, který již pracuje bez interakce s klientem.



Obrázek A.1: Spouštěcí okno platformy Kinbo[2]