VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

# OVLÁDÁNÍ POČÍTAČE POMOCÍ SENZORU KINECT

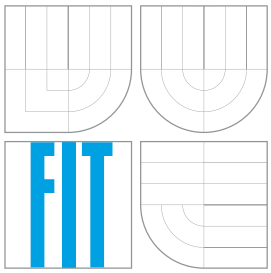
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

STANISLAV KNOT

BRNO 2014

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

# OVLÁDÁNÍ POČÍTAČE POMOCÍ SENZORU KINECT

HUMAN-COMPUTER INCERACTION USING KINECT

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

STANISLAV KNOT

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. ŠTĚPÁN MRÁČEK

BRNO 2014

# Abstrakt

Tato bakalářská práce se věnuje ovládání počítače pomocí zařízení Kinect. V práci je popsán princip, jakým zařízení snímá obraz, hloubku obrazu a zvuk. Po přečtení by měl čtenář mít představu o tom, jaké Kinect používá metody a jak fungují. Dále se dozví o návrhu a implementaci algoritmů použitých ve výsledné aplikaci.



# Abstract

This BSc. thesis is concerned about remote control of computer using Kinect device. Sound, depth of image and image capturing is described in thesis. After reading this thesis, reader should have idea of how kinect works and which methods it uses. Furthermore, thesis describes design and implementation of alghoritms used in the application.



# Klíčová slova

Kinect, gesta, pohyb, ovládání

# Keywords

Kinect, gestures, motion, control


# Citace

Stanislav Knot: Ovládání počítače pomocí senzoru Kinect, bakalářská práce, Brno, FIT VUT v Brně, 2014

# Ovládání počítače pomocí senzoru Kinect

## Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana inženýra Mráčka. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Stanislav Knot
13. května 2014


## Poděkování

Především děkuji vedoucímu práce, panu inženýrovi Mráčkovi, za cenné rady. Mé díky patří i všem kamarádům, kteří mi posloužili jako testovací subjekty, zvláště Davidovi a Katce.

# Obsah

# Kapitola 1

# Úvod

Pro svou bakalářskou práci jsem si vybral téma „Ovládání počítače pomocí senzoru Kinect“. Hlavní motivací bylo, že že jsem se chtěl podrobně seznámit s fungováním tohoto přístroje a jeho dalším využitím.

## 1.1 Cíle práce

Hlavním cílem této práce je navrhnout a implementovat algoritmus pro automatickou detekci gest rukou pomocí dat ze senzoru. Dále vytvořit aplikaci, která na základě detekování gesta bude ovládat program pro zobrazování prezentací. Účelem této zprávy je informovat čtenáře, co zařízení Kinect dělá a jakým způsobem toho dosahuje. Práce obsahuje také popis implementace a grafického prostředí výsledné aplikace.

## 1.2 Historie

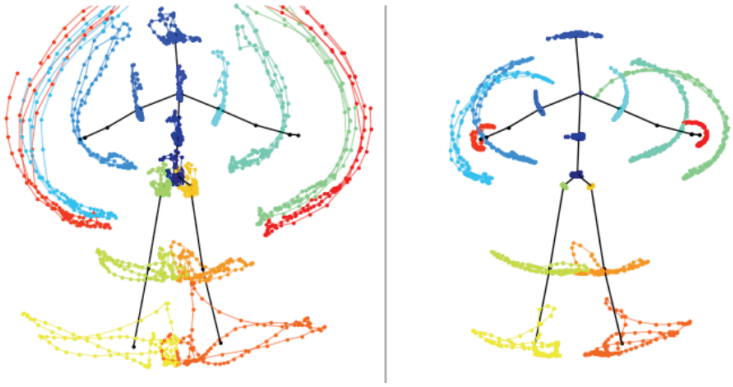
Již od vynálezu počítače bylo potřeba nějakým způsobem interagovat s aplikacemi a operačními systémy. K tomu zprvu sloužila pouze klávesnice, to bylo však pro ovládání některých aplikací nepraktické a neefektivní. Proto roku 1963 vynalezl Douglas Engelbart počítačovou myš [25]. Vynález tohoto polohovacího zařízení zvýšil efektivitu a zpříjemnil práci na počítači. Postupem času a se vznikem nových typů aplikací se vyvíjela další polohovací zařízení jako joystick, tablet, dotykový display, aj. Všechna tato zařízení mají společné to, že uživatel jimi ovládá počítač fyzickým kontaktem. S příchodem senzoru Kinect přišel nový typ ovladače - lidské tělo. U zmiňovaného senzoru není nutné, aby uživatel interagoval fyzickým dotekem, ale počítač je ovládán gesty. Odtud také pochází slovo Kinect, neboli spojení slova „kinetic“ a „connect“.

Poprvé byl Kinect představen 1. června 2009 na E3 2009 v Los Angeles firmou Microsoft. Uveden na trh byl roku 2010, původně byl určen pouze pro konzole Xbox. Na platformu PC vznikaly různé nadšenci vytvořené ovladače. Oficiální ovladače a nekomerční vývojové prostředí pro Windows vydal Microsoft až v létě roku 2011. Komerční verze byla vydána v únoru roku 2012 a odhaduje se, že aplikace pro Kinect vyvíjí přes 300 společností z celého světa [24].

Ve dnech, kdy vznikala tato práce, vydal Microsoft Kinect verzi 2. Rozdíly mezi senzory jsou popsány v sekci 3.2.1.

## 1.3 Existující řešení

Potenciál nového senzoru se projevil tak, že začaly vznikat různé programy. Několik z nich řeší stejnou problematiku jako tato práce. Například zde [7] je jednoduchý program na detekci mávnutí rukou. Velká nevýhoda tohoto řešení je, že se porovnávají kartézské souřadnice a tak uživatel musí gesto provádět přesně na tom samém místě před snímačem a nemůže se k němu ani natočit. Nezávislost na pozici snímače řeší práce [19]. Zde je zmíněna myšlenka převodu kartézských souřadnic na úhly. Spolu s ohodnocovacími funkcemi se podařilo dosáhnout dobrých výsledků. Na obrázku 1.1 vidíme porovnání několikanásobně prováděného stejného gesta. Můžeme vidět, že i při provádění stejného gesta se kartézské souřadnice mění ve výrazně větším měřítku, než je tomu u souřadnic sférických, což je pro detekci gest nevhodné.



Obrázek 1.1: Vlevo kartézské souřadnice, vpravo sférické souřadnice, zdroj: [19].

Pokud se jedná o detekce prstů na dlani, velmi dobrých výsledků bylo dosaženo v práci [5], kde byla pro snímání použita obyčejná kamera a detekce probíhá za pomoci počítačového vidění v kombinaci s umělou inteligencí. Existuje i zařízení podobné Kinectu určené právě k detekci prstů rukou. Toto zařízení se nazývá „Leap Motion“.

# Kapitola 2

# Teoretická část

## 2.1 Informace o stereoskopii

Stereoskopie je technologie, která umožňuje prostorový vjem vyvolaný dvourozměrnou předlohou. U lidí je toho dosaženo tak, že lidská hlava je vybavena dvěma optickými senzory a těmi jsou oči. Každé oko snímá objekt z mírně odlišné pozice. Tyto dva obrazy se spojí v mozku a my tak můžeme vnímat hloubku okolního světa. Pro tvorbu 3D filmů a fotografií se používají dvě dvojrozměrné kamery. Tato metoda je známa pod názvem pasivní triangulace, viz sekci 2.2.3. Obraz je pak divákovi promítán a pomocí různých technologií je zajištěno, že levým okem vidí obraz z levé kamery a pravým okem z pravé kamery. Těmito technologiemi jsou například anaglyfické brýle nebo brýle s různou polarizací skel.

Nevýhodou detekce hloubky z 3D obrazu vytvořeného dvěma dvourozměrnými kamerami, tak jak je to u člověka, je, že při výskytu stejných objektů ve scéně nemůžeme deterministicky určit, na který objekt se má kamera zaměřit, viz obrázek 2.1. Další nevýhodou je možnost splynutí objektu s pozadím, když mají stejnou barvu. Z těchto důvodů musí být spolehlivá detekce hloubky řešena jinak, viz sekci 3.1.

Obrázek 2.1: Perspektivní iluze, zdroj: [10].

Existuje několik metod, jejichž principy budou nastíněny v následujících podkapitolách.

## 2.2 Metody snímání hloubky

### 2.2.1 Time of flight

Tento systém snímání hloubky je založen na známé rychlosti světla. Měří se pak čas trvání, než světelný paprsek dosáhne od senzoru k bodu ve scéně a zpět. Tento typ snímačů přišel až roku 2000, zejména kvůli tomu, že do té doby neexistovaly dostatečně rychlé polovodičové součástky. Pomocí tohoto systému lze měřit vzdálenosti od několika centimetrů až po kilometry. Přesnost je dána na 1 cm. Při rychlosti světla $c = 299\,792\,458\,\mathrm{ms}^{-1}$ lze spočítat, že rychlost obnovování snímání polovodičové součástky při takové přesnosti musí být

$$t = \frac{2}{c \cdot 100} \doteq 66.7\,\mathrm{ps} \tag{2.1}$$

Je nutno brát v úvahu to, že při snímání musí paprsek urazit vzdálenost k předmětu, ale i zpět. Při přesnosti 1 cm je tato vzdálenost 2 cm a to je čitatel ve vztahu 2.1. Vynásobení jmenovatele stem je pak z důvodu převodu metrů na centimetry.

Při měření je vzdálenost neznámou a výpočet se provádí následovně

$$D = \frac{c \cdot t}{2} \tag{2.2}$$

Výhody této metody spočívají v přesnosti a hustotě naměřených bodů. Nevýhodou je zvláště cena a nutnost přesného časování. Problémem může být i měnící se rychlost paprsku v různém prostředí a povrch snímaného objektu.

Tato metoda se nejčastěji používá u laserových snímačů, nebo sonarů. Laserové snímače bývají nejčastěji 2D a třetí dimenze se přidává pomocí mechanické části, která otáčí snímačem [18, 26].

### 2.2.2 Phase shift

Dalším principem je metoda nazvaná Phase shift. Jak už z názvu vyplývá, metoda je založena na měření fázového posunu. Díky tomu je možno měřit jak vzdálenost objektu tak, i rychlost pohybu snímaného objektu. Jednou částí senzoru je vysílač, který vyzařuje buď modulované laserové světlo, rádiové vlny, nebo zvuk. Ve snímači se porovná fázový posun odraženého a vyslaného signálu. Rovnice pro výpočet vzdálenosti je

$$d = \frac{\phi\lambda}{4\pi} = \frac{\phi c}{4\pi f} \tag{2.3}$$

kde $\phi$ je změřený fázový posun, $\lambda$ je vlnová délka signálu, $c$ je rychlost světla a $f$ je modulovaná frekvence signálu [18].

Výhody a nevýhody jsou stejné jako u předchozí metody v sekci 2.2.1.

### 2.2.3 Pasivní triangulace

Pasivní triangulační techniky zahrnují v podstatě různé formy digitální fotogrammetrie[1]. „Pasivní“ znamená, že pro snímání se nepoužívá přídavný zdroj světla. Používají se tyto základní metody:

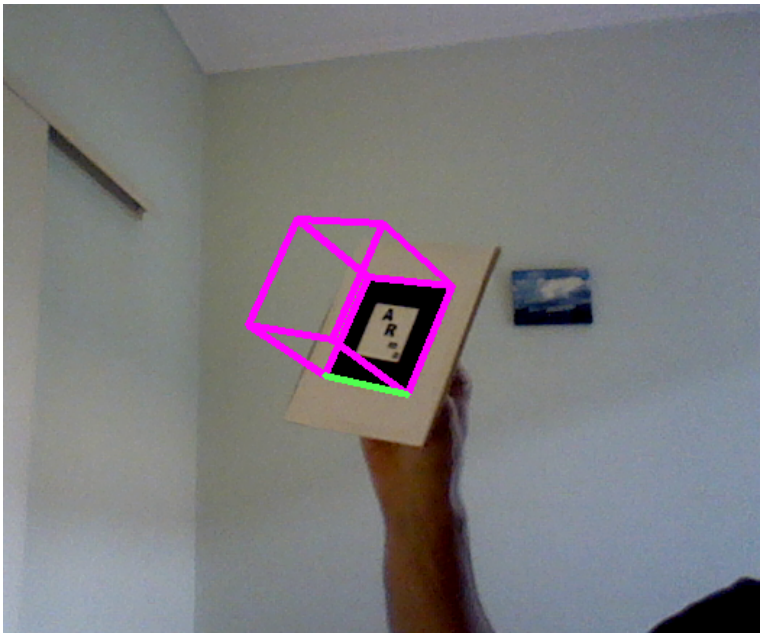
- více kamer se známou orientací,

[1] Fotogrammetrie se zabývá rekonstrukcí tvarů, měřením rozměrů a určováním polohy předmětů, které jsou zobrazeny na fotografických snímcích.

- více kamer se samokalibrací,
- jedna kamera v různých polohách se samokalibrací.

U dynamických systémů se často aplikuje více kamer a využívá se znalosti relativních poloh nebo samokalibrujících se metod. Pro statické scény se používá jedna kamera, která získá snímky ze dvou a více různých pohledů.

U technik se samokalibrací nemusí být dopředu známa poloha kamery, ale přímo ze snímků je určeno relativní umístění kamery vzhledem k měřenému objektu či vzájemná poloha kamer. Pro tyto účely je vhodné vložit do scény kalibrační předmět. Tento předmět je pak třeba nalézt v jednotlivých snímcích a z natočení a změny měřítka předmětu jsou určeny všechny potřebné parametry pro měření. Nejčastějším vzorem kalibračního předmětu je šachovnice nebo podobné vzory. Příklad můžeme vidět na obrázku 2.2.



Obrázek 2.2: Vzor pro určení polohy objektu (kalibrace), zdroj: [9].

Princip této metody je založen na rekonstrukci snímaného bodu z jeho průmětů, jelikož se při projekci ztrácí jeden rozměr a tím je právě hloubka. Rekonstrukce bodu využívá pozice celkem tří bodů. Dvěma z těchto bodů jsou snímače a třetím bodem je rekonstruovaný bod. Pro správné určení je nutné znát vzájemnou pozici těchto dvou snímačů a snímaný bod musí být zachycen oběma. Vzájemná poloha je buď známa, nebo se musí zjistit během snímání v procesu kalibrace. Kalibrace se může provést buď samostatně před měřením nebo až během něho, ale v tom případě musí být ve scéně umístěn kalibrační předmět. Výhodou je jednoduchost metody a při správném nastavení není potřeba zjišťovat úhly natočení (paralaxa se rovná velikosti báze). Z pozice snímačů se zjistí pouze velikost báze $b_x$, která je nutná pro určení měřítka. Čím je tato báze větší, tím je detekce přesnější, ale zmenšuje se prostor, který snímají oba dva snímače [23]. Společný viditelný prostor, velikost báze a pozice tří bodů jsou znázorněny na obrázku 2.3.

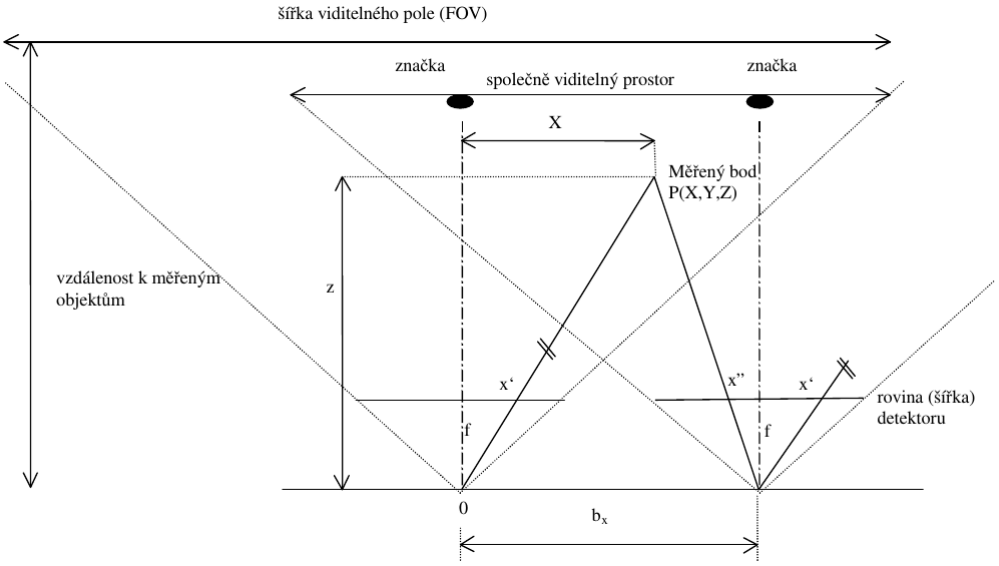
Příslušné výpočetní vztahy jsou podle [23]:

$$p = x' - x'' \tag{2.4}$$

kde $p$ je paralaxa - úhlová změna pozice

$$X = x' \frac{b_x}{p} \quad Y = y' \frac{b_x}{p} \quad Z = f \frac{b_x}{p} \tag{2.5}$$

$x'$ a $y'$ jsou průměty bodu $P(X, Y, Z)$ na první detektor, $x''$ je průmět bodu na druhý detektor, $b_x$ je velikost báze, $f$ je zobrazovací konstanta kamery.




Obrázek 2.3: Základní uspořádání kamer pro vyhodnocení prostorových souřadnic, zdroj: [23].

Další výhodou této metody je nízká cena. Nevýhody však převažují a to zvláště proto, že metoda je závislá na osvětlení scény. Další nevýhody byly zmíněny v sekci 2.1.

### 2.2.4 Aktivní triangulace

Tato metoda je podobná pasivní triangulaci, ale na rozdíl od ní používá přídavný zdroj světla. Tento zdroj může vysílat infračervené nebo viditelné světlo o různých vlnových délkách, což umožňuje zjistit barvu snímaného tělesa. Na obrázku 2.4 je demonstrován princip této metody.

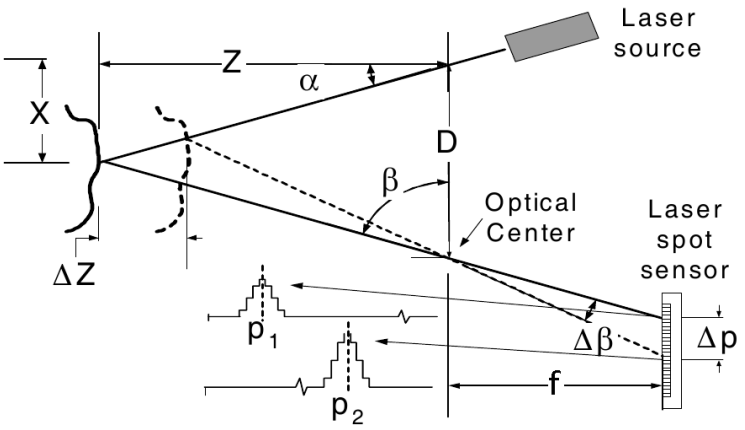
Vztah pro výpočet vzdálenosti objektu od snímače je následujících

$$Z = \frac{D \cdot f}{p + f \cdot tan(\alpha)} \text{ [4]} \tag{2.6}$$

Kde $Z$ je vzdálenost v ose $z$, $D$ je známá vzdálenost snímače od projektoru v ose $x$, $f$ je vzdálenost CMOS[2] senzoru od čočky snímače, $p$ je pozice na senzoru a $\alpha$ je úhel mezi osou $z$ a paprskem projektoru.

Výhody této metody spočívají v nezávislosti na osvětlení scény a možnosti měření vzdálenosti více bodů najednou. Nevýhodou je pak menší přesnost a závislost na povrchu snímaného objektu [18].

[2] *Complementary Metal Oxide Semiconductor.*




Obrázek 2.4: Aktivní triangulace, zdroj: [4].

## 2.3 Gesto

Jelikož cílem je detekce gest, bude nejprve vysvětlen pojem gesto z hlediska této práce. Gest je několik typů a v této práci se detekují pouze pohybová gesta rukou. Kinect může snímat i gesta předváděná pomocí prstů ruky, ale skeletonizace prstů není u tohoto zařízení podporována, tudíž by se musela použít jiná metoda například pomocí knihoven `openCV`. V sekci 1.3 je zmíněna práce, která se věnuje právě této problematice.

Gesto se skládá z posloupnosti pozic v souřadnicovém systému. Tyto pozice lze zpracovávat dvojím způsobem a to pomocí kartézských nebo sférických souřadnic. Při provedení jednoduchého gesta zápěstím znázorňujícího kruh se pozice ramen a lokte nemění. Mění se pouze pozice zápěstí a to v osách $x$ a $y$. Toto platí pouze za předpokladu, že uživatel gesto předvádí pohyb kolmo k ose snímání. Pro ilustraci lze nahlédnout na obrázek 4.4. Pokud bychom pak vykreslili grafy obsahující data těchto dvou os, jednalo by se o dvě vzájemně posunuté sinusoidy. Po převodu na sférické souřadnice pak zůstává hloubkový úhel zápěstí konstantní a výškový se mění lineárně. Stoupání či klesání je pak ovlivněno směrem otáčení.

# Kapitola 3

# Vlastnosti Kinectu

V této kapitole popíši základní vlastnosti Kinectu a teoretický popis jeho funkčnosti.

## 3.1 Detekce hloubky u Kinectu

Aby Kinect předešel problémům zmíněným v sekci 2.1, snímá hloubku scény prostřednictvím infračerveného laserového projektoru a CMOS černobílého snímače, neboli pomocí aktivní triangulace (sekce 2.2.4). Projektor vyzařuje matici bodů s určitým vzorem. Tyto body dopadají na objekty ve snímané scéně a vzhledem k deformaci vzoru, se určí vzdálenost objektu od snímače. Tento princip je znám jako metoda strukturovaného světla.

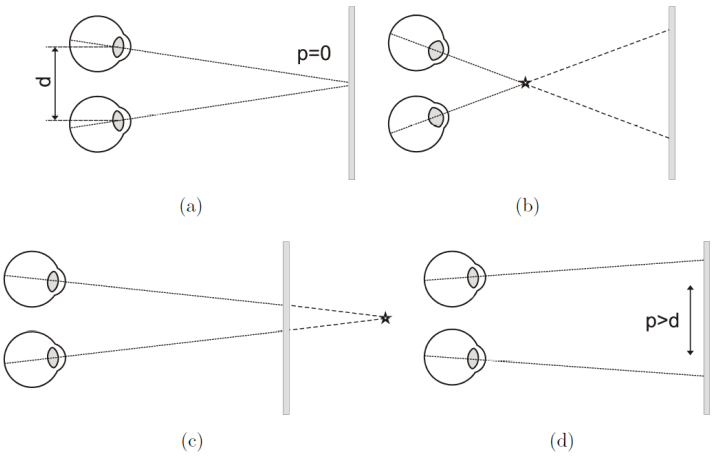
Kinect pak používá dvě techniky počítačového vidění a to hloubku ostrosti a hloubku z prostoru pro zpřesnění detekce hloubky.

Hloubka ostrosti využívá míry rozmazání sejmutého bodu. Čočka na snímači je astigmatická[1], takže vzdálenější body jsou snímány jako více rozmazány a nabývají eliptický tvar. Orientace elipsy se odvíjí od vzdálenosti objektu, na kterém je sejmuta 3.2.

Hloubka z prostoru pak využívá paralaxy. Paralaxa je úhel, který svírají dva paprsky, zaměřené na jeden bod. Různé typy paralax vidíme na obrázku 3.1. Označme vzdálenost středů čoček jako $d$ a velikost paralaxy jako $p$. Rozlišujeme čtyři druhy paralaxy [22]:

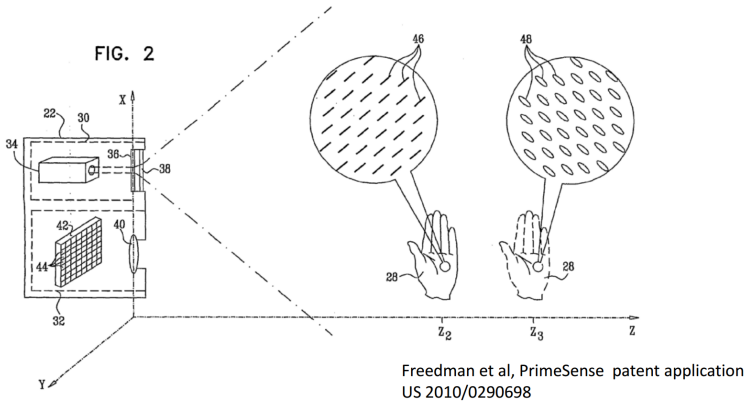
- Nulová paralaxa - dva korespondující body nejsou vzájemně posunuté, osy očí konvergují na zobrazovací rovinu (tzv. rovina konvergence).
- Pozitivní paralaxa - optické osy očí konvergují za zobrazovací rovinu (ke spojení obrazů dochází až za plátnem); na zobrazovací rovině je obraz pro pravé oko napravo a obraz pro levé oko nalevo.
- Negativní paralaxa - opakem pozitivní paralaxy. Optické osy očí konvergují před zobrazovací rovinu (obraz „vystupuje“ před plátno).
- Divergentní paralaxa - speciální případ pozitivní paralaxy, při které vzdálenost mezi korespondujícími prvky přesahuje rozestup očí pozorovatele. Osy očí by v takovém případě musely překročit svou limitní paralelní polohu, přičemž by neexistoval společný bod, na který by obě oči mohly zaostřit. V reálném světě k divergentní paralaxe nedochází a je potřeba se jí vyvarovat.

[1] Má odlišnou ohniskovou vzdálenost v ose $x$ a $y$.




Obrázek 3.1: Různé druhy paralax a) nulová paralaxa, b) negativní paralaxa, c) pozitivní paralaxa, d) divergentní paralaxa, zdroj: [22].

Kinect pak využívá kombinaci těchto dvou metod pro odvození co nejpřesnějších souřadnic. Z uvedených informací můžeme tedy odvodit, že detekce hloubky je závislá jak na rozlišení snímače, tak na rozlišení projektoru.

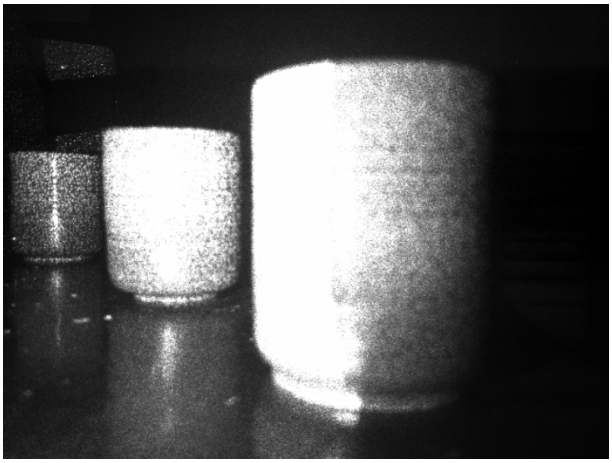



Obrázek 3.2: Patent na snímání míry rozmazání bodu, zdroj: [13].

Jelikož projektor funguje v infračerveném spektru, detekce hloubky funguje uvnitř budovy v jakýchkoliv světelných podmínkách. Při vystavení snímaného objektu přímému slunečnímu záření může dojít k rušení, jelikož vlnová délka slunečních paprsků přesahuje 750 nm. Na podobné vlnové délce pracuje rovněž laserový projektor Kinectu.

Ve vývojovém prostředí, viz 4.3, je mimo jiné ukázkový kód programu, jak snímat a pracovat s daty z hloubkového snímače. Na obrázku 3.3 vidíme tři totožné objekty umístěné v různých vzdálenostech od senzoru. Hrníček vpravo je umístěn nejblíže, hrníček vlevo nejdále.

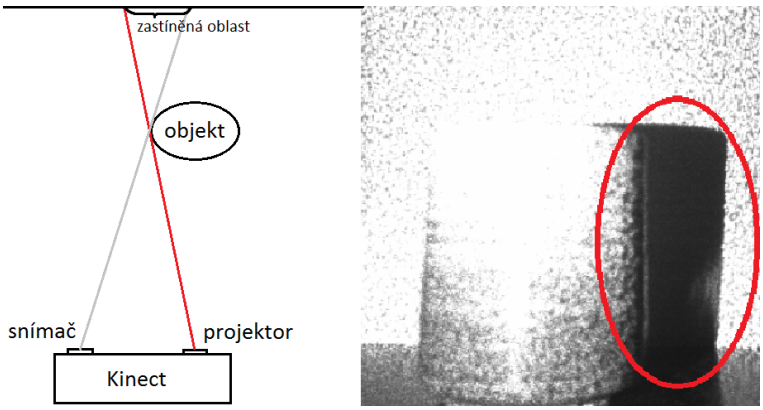
Tímto způsobem snímání je vyloučeno, že se různě velké objekty v různých vzdálenostech zdají stejně velké, jak již bylo zmíněno u obrázku 2.1. Nevýhodou tohoto řešení je, že projektor neosvítí objekty v ose, což je demonstrováno na obrázku 3.4. Toto u snímačů, které používají dvě kamery nehrozí, protože to, co je zastíněno pro jednu kameru, snímá



Obrázek 3.3: Snímání hloubky.

druhá. Řešením by mohlo být přidání dalšího projektoru, ale to vede k problematice, že senzor by musel rozeznat, který paprsek patří ke kterému projektoru, jinak dochází k rušení a přeslechům [14].

Kinect problém zastínění řeší tak, že se snaží pozici zastíněných částí těla odvodit a dopočítat. Při rychlých pohybech uživatele toto funguje celkem spolehlivě, ale pokud uživatel zůstane stát delší dobu v takové poloze, kdy je například loktem zastíněno rameno, začne celá kost kmitat a to výrazně ovlivňuje přesnost dalších výpočtů. Na obrázku 3.5 vidíme, že při detekci stejně poziciované končetiny mohou nastat různé stavy. Toto nastává obzvlášť, pokud je uživatel oblečen ve volném oděvu. Pro utlumení tohoto jevu vývojáři implementovali redukci kmitů.

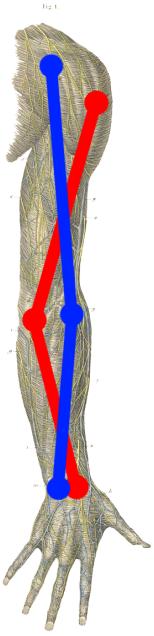



Obrázek 3.4: Ukázka oblasti stínění.

### 3.1.1 Redukce kmitů

Redukce je zajištěna pomocí vyhlazovacích filtrů. Typů filtrů je hned několik. Jejich výčet a popis funkcionality je uveden ve zdroji [3]. Kinect konkrétně používá dvojitý exponenciální filtr a pro jeho nastavení jsem použil parametry uvedené v algoritmu 3.1 [8].

Vidíme, že parametr predikce je nulový a ta je tak zcela vypnuta. Naopak parametr vyhlazování má vysokou hodnotu.



Obrázek 3.5: Různé pozice bodů.

Algoritmus 3.1: Nastavení hodnot pro filtrování vstupních dat

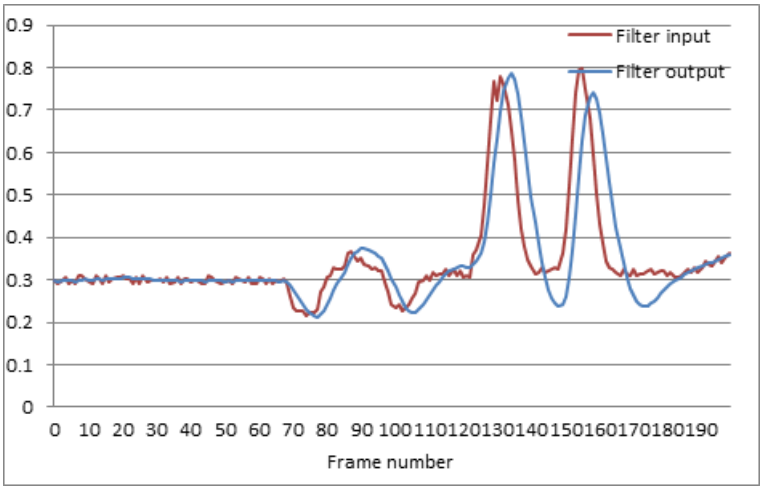
```
sensor.SkeletonStream.Enable(new TransformSmoothParameters()
{
        Smoothing = 0.75f,
        Correction = 0.1f,
        Prediction = 0.0f,
        JitterRadius = 0.05f,
        MaxDeviationRadius = 0.08f
});
```

Na obrázku 3.6 pak můžeme vidět rozdíl mezi filtrovaným a nefiltrovaným výstupem.

## 3.2 Technické vlastnosti

Laser zabudovaný v Kinectu patří do třídy I, takže není zdraví nebezpečný. Ke snímání barevného obrazu slouží CMOS RGB kamera. RGB[2] kamera slouží například pro rozeznávání obličeje nebo může být použita pro videohovory. O snímání zvuku se starají čtyři všesměrové mikrofony. Mikrofony využívají redukci šumu, odstranění ozvěny a adaptaci s prostředím. To umožňuje hlasový chat bez použití sluchátek, ale také účinné rozpoznávání řeči. Podstavec je motorizovaný a jeho prostřednictvím je možno nastavit náklon snímače v ose $x$ a to v rozsahu $\pm 27°$. V Kinectu je zabudován akcelerometr, který zjistí informaci o poloze snímače a správnost polohy je signalizována uživateli LED[3] diodou [14]. Správností polohy je myšlen především náklon v ose $z$, který by měl být nulový.

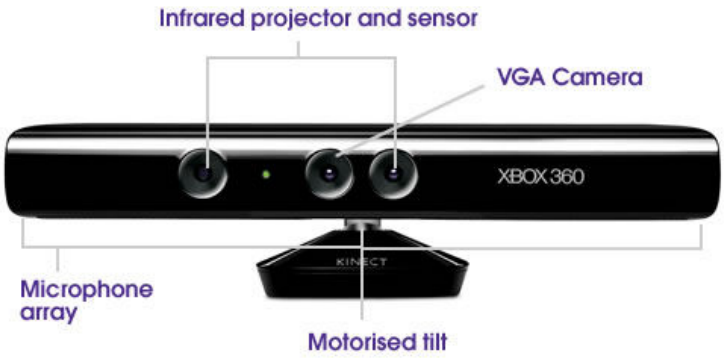
[2] *Red Green Blue*
[3] *Light Emitting Diode*




Obrázek 3.6: Filtrování výstupu - červeně nefiltrovaný, modře filtrovaný. Na svislé ose je vzdálenost v ose $z$, zdroj: [3].

Připojení Kinectu k Xboxu dříve vyžadovalo 10-15% výpočetního výkonu procesoru konzole. Microsoft provedl jisté optimalizace a dnes je využití procesoru menší než 10% [24].

Technické údaje o vybavení senzoru jsou uvedeny v tabulce 3.1, grafický popis je pak na obrázku 3.7 [2].




Obrázek 3.7: Kinect, zdroj: [11].

V úvodu jsem se zmínil, že na trh přichází Kinect verze 2. Rád bych čtenáře informoval, v čem se liší od první verze.

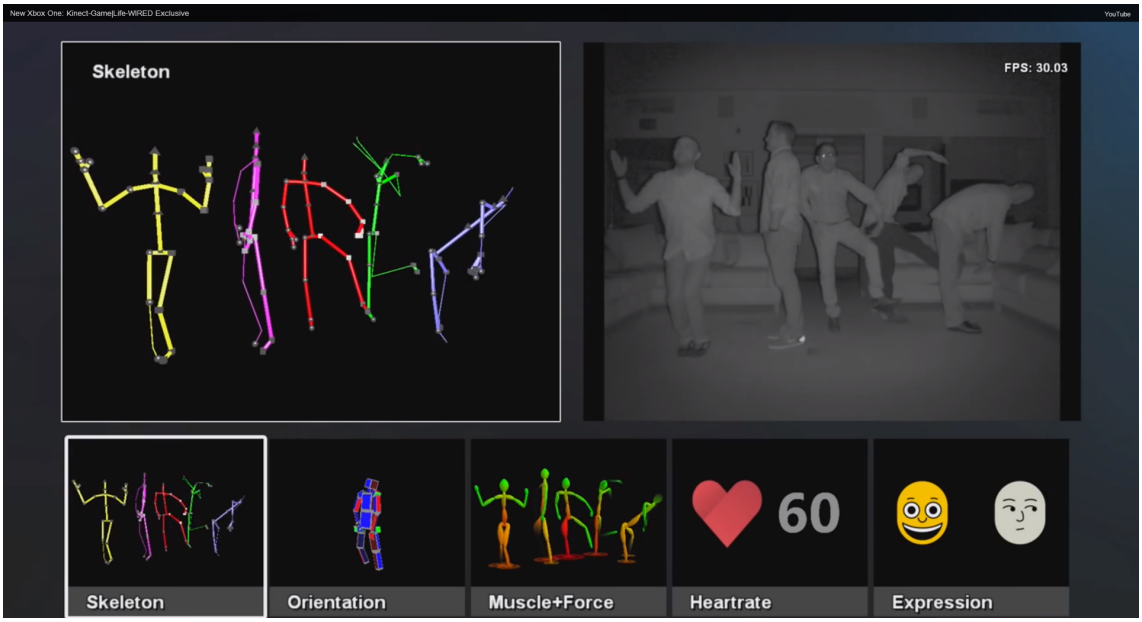
### 3.2.1 Kinect 2

Zásadní rozdíl je zejména v rozlišení snímačů, které u druhé verze podporují FullHD rozlišení. To umožňuje vytváření přesnější hloubkové mapy. U verze 2 se navyšuje maximální počet skeletizovaných hráčů ze dvou na šest. Navíc se skeletonizuje i palec ruky, takže je možno detekovat jednoduchá gesta prstů. Orientaci lze detekovat u všech segmentů těla.

| Vlastnost | Hodnota |
|---|---|
| Zorné pole | 58° H, 40° V |
| Rozlišení hloubky | VGA 640x480 30 fps |
| Rozlišení RGB kamery | 1280x960 12 fps<br>640x480 30 fps |
| Rozlišení bodů X,Y<br>(2 m od snímače) | 3 mm |
| Rozlišení bodů Z<br>(2 m od snímače) | 1 cm |
| Rozsah | 0.8 - 3.5 m |
| Počet mikrofonů | 4 |

Tabulka 3.1: Technická specifikace.

Navíc je možno zjistit zatížení jednotlivých segmentů těl hráčů. Zatížení je ve smyslu, namáhání jednotlivých částí těla. Pokud uživatel stojí v klidu, jsou nejvíce namožené nohy. Toto zatížení je detekováno s ohledem na gravitační přetížení, čili při skoku a následném dopadu se zatížení nohou zvýší odpovídající mírou. Díky RGB kameře s vysokým rozlišením je možno snímat tep hráčova srdce a dokonce jeho výraz ve tváři. Tep srdce je snímán pozorováním změn zabarvení tváře. Novinky jsou zobrazeny na obrázku 3.8.




Obrázek 3.8: Vlastnosti Kinectu2. Zleva - nová skeletonizace, orientace segmentů, zatíženost segmentů, tep srdce a výrazy hráčů, zdroj: [6].

## 3.3 Detekce kostry

Patent na tento algoritmus vlastní společnost PrimeSense, tudíž podrobný popis není zveřejněn. Hardware od této firmy je zabudován přímo do Kinectu. Protože se tato práce zaměřuje na zpracování dat již získaných Kinectem, nebude toto téma rozebráno příliš do hloubky. Uvedu zde informace jen do takové míry, aby měl čtenář představu, jak funguje proces skeletonizace. Tyto informace pocházejí z patentu [13].

Pro tento proces je použito:

- statistika a pravděpodobnost
- vícerozměrové počty
- grafy, topologie
- počítačové učení

Odvození pozice těla se skládá ze dvou fází. V první fázi se vypočítá hloubková mapa pomocí dat ze senzoru. Postup tohoto procesu je popsán v sekci 3.1.

V druhé fázi již probíhá odvozování za použití počítačového učení. Druhá fáze má dvě podfáze. V první podfázi se vezme 100 000 hloubkových obrazů, každý se známou kostrou a z tohoto obrazu se odvozují části těla. Pomocí počítačové grafiky se kostra přizpůsobí dalším tělesným typům a různým parametrům. Takto je získáno přes milion trénovacích vzorů. Z těchto vzorů se pak provádí učení. Pro určení, kterou větví v rozhodovacím lese (větší množství rozhodovacích stromů) se vydat, je použita náhoda a postup je pak ohodnocen funkcí, zjišťující informační zisk $G$ odvozen z entropie $H$.

$$G(\varphi) = H(Q) - \sum_{s \in \{l,r\}} \left( \frac{|Q_s(\varphi)|}{|Q|} \right) H(Q_s(\varphi)) \text{ [13]} \tag{3.1}$$

V druhé podfázi druhé fáze se převádí obraz části těla na kost. K tomu se používá algoritmus `Mean-Shift` [1], který vyhledává shluky bodů. Tyto shluky jsou vyhledávány na snímaném tělese. Pokud se toto těleso podobá lidské postavě, jsou odvozeny pozice kloubů. Proto je nutné, aby uživatel po zapojení Kinectu byl celým svým tělem v zorném poli snímače tak, aby ho „našel“. Po detekování kostry už pak uživatel může využívat „seated“ módu tak, jako to dělám já v této práci. Tento mód umožňuje detekci omezenou na horní končetiny, což může být přínosem, pokud je aplikace ovládána v sedě a detekce dolních končetin by byla zbytečná.

## 3.4 Rozpoznání řeči

Princip této funkčnosti je chráněn patentem, lze tedy pouze odhadnout, jak detekce funguje. Vzhledem k tomu, že se detekují pouze slova, která programu předáme v textové podobě, lze předpokládat, že pro detekci je využit algoritmus DTW[4]. Tato metoda porovnává uložený vzor s přijatým signálem a používá se pro izolovaná slova. Pro spojitou řeč se používají Markovovy modely fonémů [21].

Slova, která má Kinect detekovat, se přidají do seznamu detekovaných slov a každému slovu nebo frázi se přiřadí odpovídající návratová hodnota. Tento seznam lze psát buď

[4] *Dynamic Time Warping.*

programově nebo do samostatného `XML`[5] souboru. To se doporučuje obzvláště, pokud je detekovaných frází velké množství. Pro hlasové ovládání Kinectu je vytvořeno pouze několik slovníků neboli souborů, kde jsou uložena všechna slova a k nim příslušné energetické hodnoty. Tyto slovníky jsou například pro angličtinu, němčinu, francouzštinu atd. Slovník pro češtinu zatím nebyl vytvořen, proto je výsledná aplikace ovládána anglickými frázemi.

Slova jsou detekována s mírným zpožděním, na což je potřeba pamatovat například při ukončení nahrávání. O této problematice se zmíním ještě v závěrečné kapitole.

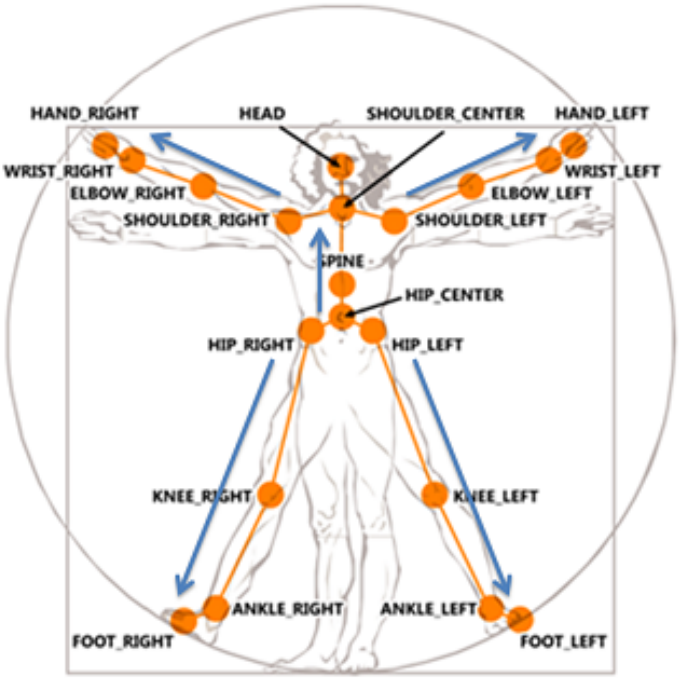
[5] *Extensible Markup Language.*

# Kapitola 4

# Návrh a implementace detekčního algoritmu

## 4.1 Analýza dat

Zprvu bylo potřeba zjistit, jaká data Kinect vlastně poskytuje. Z ukázkových kódů vývojového prostředí jsem použil program pro demonstraci práce se skeletonem [15]. Z dokumentace a z výstupu tohoto programu vyšlo najevo, že Kinect poskytuje hloubkovou mapu, z níž je odvozena pozice každého bodu v kartézském souřadnicovém systému. Vzdálenosti v jednotlivých osách jsou udávány reálným číslem v metrech. Těchto bodů je celkem dvacet viz obrázek 4.1. Na obrázku 4.2 vidíme umístění jednotlivých os snímače. Počátek souřadnicového systému se nachází v bezprostřední blízkosti snímací čočky.

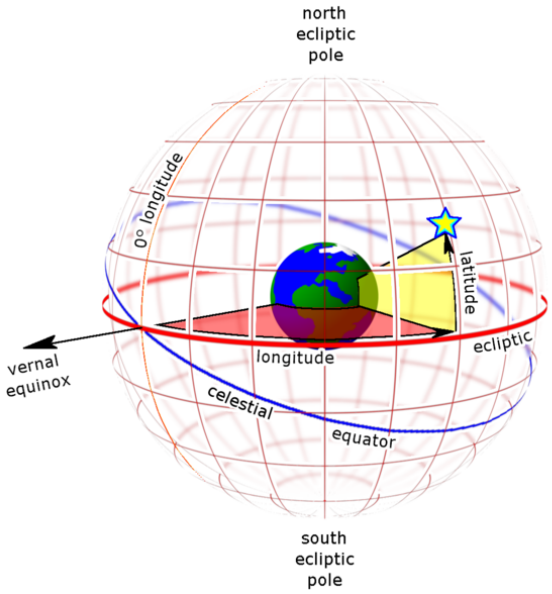



Obrázek 4.1: Popis jednotlivých bodů, zdroj: [15].


Obrázek 4.2: Souřadnicový systém snímače.

## 4.2 Návrh řešení detekce gest

Návrh řešení robustní detekce gest spočívá v použití sférického souřadnicového systému, který se používá například pro určení polohy hvězdy na obloze, viz obrázek 4.3. V této práci si můžeme představit, že pozorovatel hvězdy na Zemi je rodičovský kloub a hvězda je potomkem[1]. K uložení pozice obecného bodu jsou tedy použity dva úhly namísto kartézských souřadnic. Úhly značí, o kolik stupňů je bod odchýlen od referenční osy. Osy, od kterých se měří vychýlení jsou osami snímače (viz obrázek 4.2). Tato idea je popsána v [19] a odtud jsem také čerpal. Se znalostí těchto dvou úhlů můžeme zanedbat vzdálenost hvězdy neboli délku kosti. Tím je zajištěno, že gesta budou detekována na lidech různého vzrůstu.




Obrázek 4.3: Ekliptický souřadnicový systém, zdroj: [20].

Na obrázku 4.3 vidíme, že polopřímky vychází ze středu Země. Kloub je v mém návrhu reprezentován bodem nulové velikosti, takže odpadá nutnost přepočtu pozice pozorovatele na povrchu.

Dalším bodem zadání je, že aplikace má umět nahrávat nová gesta. Možností bylo v grafickém prostředí vytvořit ovládací prvek a nahrávat se zpožděním, aby se uživatel stihl navrátit od počítače do správné polohy před snímačem [14]. Já jsem se rozhodl, že využiji schopnosti Kinectu rozeznávat řeč a ovládání nahrávání gest navrhnu tímto způsobem.

[1]Rodičovský kloub znamená, že je blíže k tělu. Tedy rameno je rodičovský kloub lokte atd.

## 4.3 Použité programovací prostředky

Zmíněný ukázkový program je napsán v jazyce C# a můžeme ho nalézt ve vývojovém prostředí Kinect for Windows SDK[2] verze 1.7. V jazyce C# jsem napsal výslednou aplikaci i já a to zejména díky dobré dokumentaci vývojového prostředí právě pro tento jazyk. Kinect je produktem firmy Microsoft, proto je i výsledná aplikace napsána tak, aby fungovala na systému Windows. Mimo standardních knihoven .NET 4.0 jsou použity knihovny pro práci s Kinectem `Microsoft.Kinect` a pro zpracování řeči `Microsoft.Speech`. Aby bylo možno simulovat stisknutí klávesy, je nutno výslednou aplikaci spouštět s právy administrátora operačního systému. Simulaci stisknutí zajišťuje importovaná knihovna `InputSimulator.SimulateKeyPress` ve spolupráci s knihovnou `user32`.

## 4.4 Řešení přepočtu na sférické souřadnice

K řešení jsem použil zmíněný ukázkový program pro práci s kostrou [15]. Z kartézských souřadnic se provádí výpočty úhlů $\theta$ a $\varphi$, což jsou úhly ve smyslu zeměpisné délky a šířky. Kinect poskytuje mimo detekci celé kostry i detekci pouze horních končetin. Tento mód je nazván „seated“. Jelikož se jedná o detekci gest rukou, pro výpočty jsem tedy použil pouze klouby ramen, loktů a zápěstí. Kinect detekuje i pozici dlaně, ale tato detekce je často chybná, proto jsem ji zanedbal. To nám dává dva páry tří kloubů. Mezi třemi klouby jsou dvě přímky, pro každou potřebujeme určit úhly $\theta$ a $\varphi$. K výpočtu je to tedy celkem osm úhlů.

Použitý sférický souřadnicový systém předpokládá, že pozorovatel se nachází na souřadnicích $[0, 0, 0]$. Před výpočtem úhlů je potřeba všechny body transformovat. Pozice synovského kloubu je pak:

$$p = \{X, Y, Z\}, p = joint0.Position.p - joint1.Position.p \tag{4.1}$$

Kde $joint0$ je rodičovský kloub a $joint1$ potomek.
Vztah pro výpočet úhlů je podle [17]:

$$\theta = \begin{cases} cos^{-1}\left(\frac{X}{\sqrt{(X^2+Y^2)}}\right) & \text{pro } Y >= 0 \\ -cos^{-1}\left(\frac{X}{\sqrt{(X^2+Y^2)}}\right) & \text{pro } Y < 0 \end{cases} \tag{4.2}$$

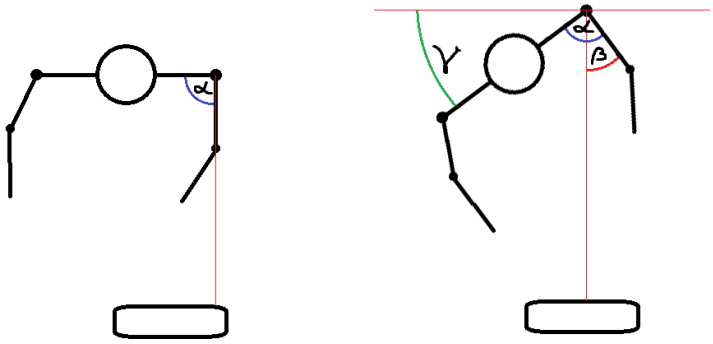
$$\varphi = cos^{-1}\left(\frac{Z}{\sqrt{X^2+Y^2+Z^2}}\right) \tag{4.3}$$

Problematiku různého natočení uživatele před snímačem řeším tak, že vypočítám úhel, který svírají ramena s osou $x$ a tento úhel přičítám k jednotlivým vypočteným hloubkovým úhlům (zeměpisná délka). Výpočet úhlů zápěstí se pak provádí obdobně, ale je potřeba uvažovat i úhel lokte. Pro lepší představu je uveden příklad na obrázku 4.4.

Na obrázku 4.4 vidíme, že v případě znázorněném vlevo, je úhel $\beta$, neboli úhel sevřený mezi osou $z$ a přímkou symbolizující kost mezi ramenem a loktem, nulový. V případě znázorněném vpravo je tento úhel však jiný a je nutno provést korekci

$$\beta = \alpha - \gamma \tag{4.4}$$

[2] Software Development Kit.




Obrázek 4.4: Uživatel předvádí stejné gesto, ale pod jiným úhlem k ose $z$.

kde $\beta$ je substitucí pro $\varphi$.

Tato korekce však funguje jen od 0° do 180°, jelikož Kinect nedokáže rozeznat, zda k němu uživatel stojí zády nebo čelem.

Gesto se skládá z posloupnosti takto vypočtených úhlů a je tedy nutné je ukládat do polí. Pro zjednodušení manipulace s úhly jsem si vytvořil třídu `Frame`, kde jsou uloženy čtyři dvojice úhlů. Kinect poskytuje 30 snímků za vteřinu, viz tabulku 3.1, a každý snímek je nutno přepočítat na úhly a vložit do bufferu. Buffer je implementován jako fronta, do které se vkládá na začátek a poslední prvek se zahazuje. To je kvůli jednoduššímu porovnávání uložených gest a obsahu bufferu. K tomuto tématu se vyjádřím později. Výsledky uvedených vzorců jsou reálná čísla. Pro rychlejší běh aplikace a vzhledem k nastavené toleranci detekování převádím úhly a dále pracuji s celočíselným datovým typem.

Pro gesto je vytvořena třída `Gesture`, která mimo list objektů třídy `Frame` obsahuje jméno, informaci o akci, která se má po detekci provést, a pomocné proměnné. V hlavní třídě programu je pak globální list, obsahující všechna gesta, se kterými se pracuje.

Do seznamu detekovaných řečových frází jsou přidány hodnoty programově a to takto: `newSemanticResultValue("phrase","RETURN_VALUE")`. Část `phrase` představuje slovo nebo slovní spojení, které se má detekovat a `RETURN_VALUE` je pak návratová hodnota pro funkci, která provádí požadované akce.

### 4.4.1 Nahrávání gest

Abychom mohli gesta porovnávat, musíme nejdříve nějaká nahrát. Jak již bylo řečeno, k nahrávání je využita schopnost hlasového ovládání Kinectu. Při detekci slovního spojení „Kinect start“ se aktivuje počítadlo, které počítá, kolik snímků přišlo od tohoto okamžiku a při detekci fráze „Kinect stop“ uloží daný počet snímků a počítadlo je vynulováno. Vzhledem k tomu, že buffer je realizován jako fronta s vkládáním na začátek, je zřejmé, že pořadí uložených snímků gesta je opačné. To však nemá na porovnávání vliv. Pokud zazní jako první „Kinect stop“, neukládá se nic. Pokud počítadlo doběhne na maximální délku bufferu, která je nastavena na 120 snímků (při 30 fps jsou to 4 vteřiny), počítadlo se dále nenavyšuje. Namísto toho se program chová stejně jako by bylo nahrávání ukončeno frází „Kinect stop“. Vícenásobná detekce „Kinect start“ před ukončením nahrávání je ignorována. Nahrávání je pro uživatele signalizováno oznámením `Recording...` a červeným kolečkem v grafickém prostředí aplikace. Po ukončení nahrávání se místo tohoto oznámení vypíše aktuální počet uložených gest. Při ukončení nahrávání je nutno pamatovat na zpoždění, které vzniká při

detekci řeči.

Gestu se přiřadí vygenerované jméno „Gesture $n$“, kde $n$ je $akualni\_pocet\_gest + 1$. Může nastat situace, že se v databázi uložených gest provedlo mazání a tak by bylo jedno jméno přiřazeno vícekrát. To je ošetřeno v cyklu tak, že pokud k tomu dojde, navýší se $n$ o jedničku. Jméno gesta je tedy jeho primárním klíčem.

Akce je defaultně nastavena na `UP`, neboli šipka nahoru. Název i typ prováděné akce může uživatel po nahrání gesta změnit. Avšak je potřeba dávat pozor, zda je vybráno gesto, které chce editovat. Po vypnutí programu jsou jednotlivá gesta serializována a uložena do souboru `gestures.osl`. Obsah téhož souboru je deserializován při spuštění programu a gesta jsou uložena do pracovního listu. O uložení a načtení souboru se stará objekt třídy `BinaryFormatter`. Pokud nebyl soubor nalezen, je program spuštěn s tím, že je nahráno 0 gest. V případě, že soubor neexistuje a chceme ukládat gesta, soubor se vytvoří ve složce, kde je umístěn sestavený `.exe` soubor programu.

Nahrávání se aktivuje pouze v případě, že je detekována nějaká kostra. Kinect implicitně neposkytuje informaci o tom, zda je úspěšně detekován uživatel, ale tuto informaci lze získat tak, že pozice některé z koster je nenulová v jakékoliv ose.

### 4.4.2 Porovnávání úhlů a vykonání akce

Při pohledu na uložená data je zřejmé, že pracujeme s diskrétními signály. Porovnání uloženého gesta a obsahu bufferu lze tedy řešit pomocí operace zvané korelace. Tato operace ze dvou vektorů vypočítá koeficient, podle kterého lze určit, zda je mezi signály závislost nebo nikoliv. Značnou nevýhodou této operace je vysoká náročnost na výpočet. Dalším problémem, na který jsem u této metody narazil je, že pokud se hodnoty mění jen málo, je koeficient ovlivněn sebemenší odchylkou. To se pro porovnávání gest nehodí, jelikož může nastávat situace, že se hloubkový úhel mění málo nebo vůbec a koeficient pak nabývá nevhodných hodnot. Proto jsem metodu považoval za nevhodnou a použil jsem diferenční sumu.

Detekce gest pak probíhá tak, že pro každé uložené gesto probíhá porovnávání s aktuálním obsahem bufferu a probíhá po každém obnovení jeho obsahu, tedy při každém novém snímku. Porovnávání a zjištění rozdílu mezi dvěma úhly zajišťuje funkce `diff(int GestAngle, int KinectAngle)`. Tato funkce vrací rozdíl úhlů s ohledem na modularitu 360. Rozdíl je určen jako

$$diff = 180 - ||GestAngle - KinectAngle| - 180| \text{ [12]} \tag{4.5}$$

kde $GestAngle$ je konkrétní úhel uloženého gesta a $KinectAngle$ je úhel, který se nachází v bufferu.

Tento rozdíl je pak přičítán k celkové chybě. Chyba se počítá pro každou ruku zvlášť a pokud je detekce ruky vypnuta, je chyba nastavena na 0, tudíž se chová, jako by byla detekována za jakékoliv situace. Tato celková chyba každé ruky se vydělí délkou gesta a tak zjistíme relativní odlišnost. Pokud je tato odlišnost u obou rukou menší nebo rovna nastavené toleranci, je gesto prohlášeno za shodné a vykoná se požadovaná akce.

$$n = delka\_gesta$$

$$rozdilnost = \frac{1}{n} \sum_{i=0}^{n-1} diff_i \tag{4.6}$$

Ruce, které chce uživatel detekovat, si vybere v uživatelském prostředí pomocí příslušného `CheckBox`-u. Pokud je vypnuta detekce obou rukou, je každé nastavena maximální chyba a detekce je ukončena. Tím je ošetřeno, že se v takovém případě nedetekuje nic.

Porovnávání gesta neprobíhá jen jednou, ale hned několikrát a to kvůli implementaci časového rozvoje. Ten se provádí tak, že porovnáváme obsah bufferu s uloženým gestem, které v poli indexujeme $k$, kde $k \doteq i \cdot t$, $t$ je časové zrychlení respektive zpomalení a $i$ je index v bufferu. To umožňuje detekovat zrychlená respektive zpomalená gesta, ale zároveň se zvyšuje složitost. Při testování programu jsem používal rychlosti 0.8 až 1.2 s krokem 0.2. Pro ukázku je přiložen pseudokód 4.1.

Algoritmus 4.1: Pseudokód porovnávání a časového rozvoje

```
for (float t = 0.8f; t <= 1.2f; t += 0.2f) // ruzne rychlosti
{
  // inicializace
  for (int i = 0; i < frames.Count; i++)
  {
    int k = (int)(Math.Ceiling(t * i));
    // zpomalene gesto je delsi nez buffer
    if (k >= frames.Count)
    {
      break;
    }
    differenceL=diff(this.frames[k].thetaLS,cbuff[i].thetaLS);
    // dalsi vypocty
    analogyL = differenceL / this.frames.Count;
    analogyR = differenceR / this.frames.Count;
    if (analogyL <= TOLERANCE && analogyR <= TOLERANCE)
    {
      this.executeAction(action);
    }
  }
}
```

Práh pro detekci gesta jsem nastavil po provedení četných experimentů na referenční hodnotu 45. Při vysokých hodnotách prahu hrozí, že se gesto detekuje i když nebylo předvedeno nebo byla předvedena jen jeho část. U nízkého prahu tolerance musí být gesto předvedeno naprosto přesně, což není uživatelsky přívětivé. Hodnotu tohoto prahu si může uživatel nastavit posuvníkem v grafickém prostředí aplikace.

Při detekci gesta se konkrétní instanci nastaví „odstavení“, což je počet snímků, po který nemůže být opětovně detekováno. Je to z toho důvodu, že gesto může být shodné při různých rychlostech nebo se detekuje vícekrát díky nastavené toleranci. Podle experimentů jsem tuto hodnotu nastavil na 45 snímků, což je časově 1,5 vteřiny. Toto odstavení se nastaví i nově vytvořenému gestu, protože ihned po ukončení nahrávání je v bufferu a bylo by tedy detekováno.

Akce, která se má provést při detekci je pak implementována pomocí knihoven, viz 4.3. Funkci `GetProcessesByName` předáme řetězec reprezentující jméno procesu programu, který chceme ovládat. V mém případě je to program Adobe Reader a jméno procesu je „`AcroRd32`“. Tato funkce najde všechna okna se běžícím procesem shodným se zadaným jménem a funkce `SetForegroundWindow` s parametrem „handleru“ okna ho nastaví jako aktivní. Ze seznamu

dostupných akcí [16] jsem pro tento program vybral základní akce pro ovládání prezentace.

Stisk jedné klávesy se simuluje voláním metod třídy `InputSimulator`. Na příkladech vidíme simulování stisku šipky nahoru a kombinaci stisku kláves Ctrl a L:

```
SimulateKeyPress(VirtualKeyCode.UP)
SimulateModifiedKeyStroke(VirtualKeyCode.CONTROL, VirtualKeyCode.VK_L)
```

### 4.4.3 Přehrání gesta

Pokud uživatel zapomene přejmenovat nahrané gesto tak, aby podle intuice věděl, jaký pohyb představuje, je malá pravděpodobnost, že se toto gesto naučí a předvede správně. Proto jsem se rozhodl, že v grafickém prostředí implementuji možnost přehrání gesta. Přehrání se provádí pomocí uložených dat o gestu. Jelikož jsou gesta uložena tak, jak byla v bufferu, je potřeba otočit pořadí přehrávaných snímků. Před samotným přehráváním je ještě nutné provést projekci, neboli jak dlouhou vidíme danou kost z pohledu snímače. Tato projekce se vypočítá z uložených úhlů pro danou kost. Za předpokladu, že rameno je v referenční poloze vypočítáme pozici lokte jako

$$newBone = |(bone \cdot cos(\varphi))| \tag{4.7}$$

$$X_{lokte} = X_{ramene} + \left(-1 \cdot \left(newBone \cdot cos\frac{\pi \cdot \theta}{180}\right)\right) \tag{4.8}$$

$$Y_{lokte} = Y_{ramene} + \left(newBone \cdot sin\frac{\pi \cdot \theta}{180}\right) \tag{4.9}$$

Kde `newBone` je délka kosti, kterou vidíme po projekci. Konstanta `bone` je maximální délka kosti, která se může vykreslit. Pozice zápěstí se pak vypočítá obdobně.

Vybrané gesto si můžeme libovolně přehrávat a to pomocí tlačítka nebo slovní fráze „Kinect replay“. Takto může uživatel stát před snímačem a učit se gesto sledováním z monitoru počítače.

Jelikož i během přehrávání je požadavek na detekci gest, bylo nutno zařídit, aby každý proces běžel v samostatném vláknu. To je implementováno tak, že při spuštění přehrávání se spustí nové parametrizované vlákno s funkcí pro přehrávání, která má jako parametr vybrané gesto a detekce dále běží v hlavním vláknu. Přehrávání se spouští ihned po vybrání gesta, po kliknutí na tlačítko `Replay` nebo hlasovým ovládáním. Pokud během přehrávání gesta vybereme přehrávání jiného gesta nebo chceme gesto přehrát znova, bežící vlákno se ukončí a vytvoří se nové s daným parametrem.

Přehrávání je s uživatelským prostředím propojeno tak, že jsou v XAML[3] souboru pro popis GUI vytvořeny čtyři přímky s inicializačními body a pozice těchto bodů je pak měněna ve funkci `drawBones`. Po každé změně souřadnic se vlákno uspí na $\frac{1s}{30fps} \doteq 33\,ms$ a tak vzniká dojem spojitého pohybu.

### 4.4.4 GUI

Bližší popis nalezneme v příloze. Grafické prostředí výsledné aplikace v prostředí Windows 7 vidíme na obrázku B.1. Při návrhu grafického prostředí jsem se snažil sdružit ovládací prvky podobného významu k sobě tak, aby bylo pro uživatele snadno pochopitelné a ovladatelné.

Rozhodl jsem se pro možnost minimalizace a obnovení okna aplikace hlasovým ovládáním, aby bylo možno zobrazit aplikaci během prezentace, přehrát si zvolené gesto a poté

[3] *Extensible Application Markup Language*

nerušeně pokračovat v prezentaci. To vše bez fyzického kontaktu s počítačem, zobrazujícím prezentaci.

## 4.5 Problémy při řešení

Asi největším problémem, na který jsem během vypracovávání narazil, je zmíněné stínění projektoru a následovně chybné pozice kloubů. Řešením tohoto problému by pak mohlo být snímání scény dvěma projektory. Tyto projektory by musely synchronizovaně vyzařovat i snímat paprsky.

Problémem je i zmíněné zpoždění při detekci řeči (zmíněno v sekci 3.4). Se začátkem nahrávání je problém minimalizován, protože až nahrávání začne, je přehledně signalizováno. Obtíž nastává při ukončení nahrávání, kdy nepoučený uživatel předpokládá ukončení nahrávání ihned po vyslovení příslušného slovního spojení. Z experimentů jsem vypozoroval, že toto zpoždění se pohybuje okolo 0,5 s. Řešením by tedy mohlo být uložení gesta bez 15 posledních snímků (0,5 s při 30 fps). Tuto část programu jsem ponechal tak, že gesto je uloženo celé, tedy až do změny signalizace v grafickém prostředí.

Nepříjemností je i fakt, že o veškeré výpočty pro porovnávání se stará procesor počítače. S narůstajícím počtem uložených gest roste i počet porovnání a to lineárně. Proto je doporučeno pro každou akci mít optimálně jedno odpovídající gesto. Pro menší počet porovnávání by mohl být nastaven počet snímků za vteřinu na 12, což stále působí spojitým dojmem a objem dat by se zmenšil na méně než polovinu.

## 4.6 Experimenty

Podle zadání jsem měl pro funkčnost výsledného programu ověřit provedením experimentů alespoň na 10 lidech. Měření jsem provedl na každém subjektu dvakrát, přičemž jedno měření se skládalo z předvedení uloženého gesta a nahrání a předvedení vlastního gesta. První uložené gesto bylo zamávání rukou a druhé zahrnovalo pohyb obou rukou. Gesta nahrávána uživateli byla různorodých pohybů a tak se dobře otestovala robustnost detekčního algoritmu. Do výsledků jsem zahrnul i přesnost hlasového ovládání. Výsledky jsou uvedeny v tabulce 4.1. Značka „1“ znamená, že detekce proběhla úspěšně, „0“ značí neúspěch.

Z výsledků vidíme, že bylo úspěšně detekováno 77,5 % předvedených gest. Hlasové ovládání bylo velmi spolehlivé, až na třetí subjekt, jímž byla žena. Špatné výsledky při tomto měření lze přisuzovat vyššímu tónu jejího hlasu. První čtyři experimenty byly provedeny v laboratorních podmínkách a zbytek v obytných prostorách kolejí. V druhém zmíněném prostředí se pohybovalo v pokoji více osob, což umožnilo odhalení chyby v programu. Chyba spočívala v tom, že se brala v úvahu data ze všech detekovaných koster a ne jen z jedné konkrétní. To mohlo mít vliv na výsledky některých předešlých neúspěšných měření.

Další experimenty jsem provedl nahráním gesta a postupným předváděním po časovém úseku několika hodin. Výsledky tohoto experimentu jsou zobrazeny v tabulce 4.2.

Předváděné gesto zahrnovalo pohyb obou rukou. Funkce přehrávání se ukázala jako velmi užitečná zvláště při druhé polovině experimentů, které jsem prováděl několik dní po nahrání gesta. Chybné předvádění spočívalo zejména v opačném směru pohybu.

Během provádění experimentů jsem se rozhodl, že do grafického prostředí implementuji prvek, kterým lze nastavit toleranci detekce. Jako mezní hodnoty jsem nastavil minimum 15 a maximum 60. Toto jsou hodnoty zjištěné z experimentů jako přijatelné.

| Výška subjektu | Měření | Uložené gesto | Vlastní gesto | Hlasové ovládání |
|---|---|---|---|---|
| 193 | 1 | 1 | 1 | 1 |
| | 2 | 1 | 1 | 1 |
| 196 | 1 | 1 | 1 | 1 |
| | 2 | 1 | 1 | 1 |
| 180 | 1 | 1 | 1 | 0 |
| | 2 | 1 | 0 | 0 |
| 180 | 1 | 1 | 0 | 1 |
| | 2 | 1 | 1 | 1 |
| 179 | 1 | 1 | 0 | 1 |
| | 2 | 1 | 1 | 1 |
| 175 | 1 | 1 | 0 | 0 |
| | 2 | 0 | 1 | 1 |
| 176 | 1 | 1 | 1 | 1 |
| | 2 | 1 | 1 | 1 |
| 185 | 1 | 0 | 1 | 1 |
| | 2 | 0 | 1 | 1 |
| 190 | 1 | 1 | 1 | 1 |
| | 2 | 1 | 0 | 1 |
| 195 | 1 | 1 | 1 | 1 |
| | 2 | 0 | 1 | 1 |

Tabulka 4.1: Výsledky experimentů na různých lidech.

| Číslo pokusu | Bez přehrání | S přehráním |
|---|---|---|
| 1 | 1 | 1 |
| 2 | 1 | 1 |
| 3 | 1 | 1 |
| 4 | 1 | 1 |
| 5 | 1 | 1 |
| 6 | 0 | 1 |
| 7 | 1 | 1 |
| 8 | 0 | 1 |
| 9 | 0 | 1 |
| 10 | 0 | 1 |

Tabulka 4.2: Výsledky experimentů s časovým odstupem.

# Kapitola 5

# Závěr

Výsledkem této práce je program pro ovládání prezentací pomocí gest. Nepřítomnost ovladače a ovládání pouze vlastním tělem působí efektně avšak efektivita tohoto řešení není příliš vysoká. Při prezentování může mluvčí nevědomky předvést gesto, což má za následek odpovídající akci a prezentace pak působí zmateně. Výhodnější je proto použít například dálkové ovládání. Aplikaci lze však přizpůsobit tak, aby se jí dal ovládat libovolný program. To neznamená že budoucnost ovládání bez ovladače neexistuje. V kombinaci s 3D projekcí je silný potenciál tohoto řešení zvláště v herním průmyslu.

V technické zprávě jsem vysvětlil různé principy snímání hloubky objektu ve scéně a pravděpodobný postup Kinectu při skeletonizaci člověka. Dále jsem popsal metodu k získání souřadnic sférických z kartézských. Popsal jsem, jak jsou tyto souřadnice uloženy, jak se s nimi pracuje a na zdrojovém kódu je demonstrováno, jak probíhá porovnání. Nakonec je popsán návrh grafického prostředí a ovládání aplikace včetně toho hlasového.

V praktické části této práce jsem vytvořil aplikaci, která získává data z Kinectu. Napsal jsem funkce a datové struktury pro výpočty úhlů a následné porovnávání. V grafickém prostředí jsem nad rámec zadání implementoval přehrávání gest.

Jako možné rozšíření bych viděl zajištění schopnosti aplikace vyhledat Kinect po jeho odpojení z USB a následovném připojení. Toto řešení poskytuje knihovna `Microsoft.Kinect.Toolkit`, ale nepodařilo se mi vyhledat žádnou dokumentaci k její funkcionalitě. Vylepšením by mohlo být i zobrazení video vstupu RGB kamery v grafickém prostředí, aby uživatel viděl, zda je správně v záběru i při předvádění rozličných gest. Velkým přínosem by mohla být i sofistikovanější ohodnocovací funkce.

# Literatura


[1] Mean-Shift segmentace. Online, [cit. 1. 4. 2014].
URL http://cmp.felk.cvut.cz/cmp/courses/ZS1/Cviceni/cv4/meanshift.pdf

[2] Andersen, M.; Jensen, T.; Lisouski, P.; aj.: *Kinect Depth Sensor Evaluation for Computer Vision Applications*. Department of Engineering, Aarhus University. Denmark. 37 pp. - Technical report ECE-TR-6, 2012, iSSN 2245-2087.

[3] Azimi, M.: Skeletal Joint Smoothing White Paper. Online, [cit. 11. 4. 2014].
URL http://msdn.microsoft.com/en-us/library/jj131429.aspx

[4] Beraldin, J.-A.; Blais, F.; Rioux, M.; aj.: Optimized Position Sensors for Flying-Spot Active Triangulation Systems. Online, 2003, [cit. 18. 4. 2014].
URL http://www.researchgate.net/publication/221470168_Optimized_Position_Sensors_for_Flying-Spot_Active_Triangulation_Systems

[5] Cai, M.; Goss, A.: Hand Gesture Recognition and Classification. Technická zpráva, Undergraduate Department of Computer Science, Colorado State University, Fort Collins, CO 80523, 2013.

[6] Corellianrogue: Everything Kinect 2 In One Place! (See What I Did There?). Online, [cit. 14. 4. 2014].
URL http://123kinect.com/everything-kinect-2-one-place/43136/

[7] Development, M. U. S.: Writing a gesture service with the Kinect for Windows SDK. Online, [cit. 11. 4. 2014].
URL http://blogs.msdn.com/b/mcsuksoldev/archive/2011/08/08/writing-a-gesture-service-with-the-kinect-for-windows-sdk.aspx

[8] Duncan, G.: That's smooth... Skeleton movement that is. Online, [cit. 11. 4. 2014].
URL http://channel9.msdn.com/coding4fun/kinect/Thats-smooth-Skeleton-movement-that-is

[9] Evangelidis, G.: ARma library: Pattern tracking for Augmented Reality. Online, [cit. 17. 4. 2014].
URL http://xanthippi.ceid.upatras.gr/people/evangelidis/arma/

[10] Gregory, R. L.; Wallace, J. G.: Recovery from Early Blindness. Online, [cit. 14. 4. 2014].
URL richardgregory.org/papers/recovery_blind/3-observations_p2.htm

[11] III, R. I.: Kinect Development. Online, [cit. 14. 4. 2014].
URL http://www.everybodyplays.co.uk/images/screenshots/picsforarticles/kinect.jpg

[12] JasonD: Comparing angles and working out the difference. Online, [cit. 10. 4. 2014].
URL gamedev.stackexchange.com/questions/4467/comparing-angles-and-working-out-the-difference

[13] MacCormick, J.: How does the Kinect work? Online, [cit. 1. 4. 2014].
URL http://pages.cs.wisc.edu/~ahmad/kinect.pdf

[14] Microsoft: Skeletal Tracking. Online, [cit. 2. 4. 2014].
URL http://msdn.microsoft.com/en-us/library/hh973074.aspx

[15] Microsoft: Tracking Users with Kinect Skeletal Tracking. Online, [cit. 2. 4. 2014].
URL msdn.microsoft.com/en-us/library/jj131025.aspx

[16] Microsoft: Windows Input Simulator (C# SendInput Wrapper - Simulate Keyboard and Mouse). Online, [cit. 10. 4. 2014].
URL http://inputsimulator.codeplex.com/

[17] nhahtdh: Cartesian to Polar (3d coordinates). Online, 2012, [cit. 20. 4. 2014].
URL http://stackoverflow.com/questions/10868135/cartesian-to-polar-3d-coordinates

[18] Pilch, B. P.: *3D Mapování vnitřního prostředí senzorem Microsoft Kinect*. Diplomová práce, Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2013.

[19] Raptis, M.; Kirovski, D.; Hoppe, H.: Real-Time Classification of Dance Gestures from Skeleton Animation. Technická zpráva, University of California, Los Angeles, 2011.

[20] shelf3d.com: Ecliptic coordinate system. Online, 2014, [cit. 6. 5. 2014].
URL http://shelf3d.com/i/celestial%20longitude

[21] Szöke, I.: Jak se počítač učí rozpoznávat mluvenou řeč. Online, 2010, [cit. 6. 5. 2014].
URL http://www.osel.cz/index.php?clanek=5152

[22] Vinkler, M.: *Využití pohybového snímače Kinect ve virtuální realitě*. Diplomová práce, Masarykova univerzita, Fakulta informatiky, 2012.

[23] VUT: Pasivní triangulace. Online, [cit. 17. 4. 2014].
URL
midas.uamt.feec.vutbr.cz/APV/exercises-pdf/08_Pasivni_triangulace.pdf

[24] Wikipedia: Kinect. Online, 2014, [cit. 31. 3. 2014].
URL http://en.wikipedia.org/wiki/Kinect

[25] Wikipedia: Počítačová myš. Online, 2014, [cit. 30. 3. 2014].
URL
https://cs.wikipedia.org/wiki/Po%C4%8D%C3%ADta%C4%8Dov%C3%A1_my%C5%A1

[26] Wikipedia: Time-of-flight camera. Online, 2014, [cit. 17. 4. 2014].
URL http://en.wikipedia.org/wiki/Time-of-flight_camera

# Dodatek A

# Obsah CD

- Zdrojové kódy programu a dokumentace.
- Instalační soubor aplikace.
- Sestavená aplikace (pro běh je nutná instalace ovladačů a přítomnost přiložených knihoven).
- Ovladače pro Kinect.
- Technická zpráva ve formátu PDF.

# Dodatek B

# Uživatelská příručka

Pro běh aplikace je nutné, aby na počítači byly nainstalovány ovladače pro senzor Kinect a framework .NET verze 4.0 a vyšší. Aplikace se instaluje pomocí přiloženého instalačního souboru. Spolu s programem se nainstalují další knihovny nutné pro správnou funkčnost. Nainstalovaný program pak spustíme dvojklikem na jeho ikonu, která se po instalaci přidá na pracovní plochu. Jelikož je ovládán jiný proces, je nutné, aby program běžel s právy administrátora. Pokud je tedy uživatel dotázán zda s pustit s těmito právy, zvolte Ano.

Grafické prostředí je navrženo tak, aby bylo maximálně pochopitelné a jednoduché. Hlavní část okna programu tvoří silueta člověka, která slouží pro přehrávání uložených gest. Nad touto siluetou jsou dvě zatržítka určená pro výběr detekovaných rukou a jezdec nastavující citlivost detekce. Vpravo je pak editační část. Ovládání prostředí se provádí myší a klávesnicí. Okno aplikace je zobrazeno na obrázku B.1.

1. Výběr, která ruka se má detekovat či nikoliv.
2. Nastavení citlivosti detekce.
3. Výpis všech uložených gest.
4. Tlačítka pro smazání nebo přehrání vybraného gesta
5. Editační část umožňující měnit název a akci jednotlivých gest.
6. Tlačítko pro potvrzení a uložení změn.
7. Informační panel zobrazující aktuální počet gest, chybová hlášení nebo signalizující nahrávání.
8. Informace, zda byl uživatel úspěšně detekován.
9. Oblast přehrávání gesta.

Dalším prvkem ovládání je řeč. Program detekuje tato slovní spojení

- Kinect start
- Kinect stop
- Kinect replay

- Kinect hide
- Kinect show

Kinect start slouží pro zahájení nahrávání nového gesta. Nahrávání však funguje pouze, pokud je uživatel ve správné poloze před senzorem. Správnost této polohy je signalizována zeleným nápisem `Tracked`. Naopak nesprávnost polohy je signalizována červeným nápisem `Not Tracked`. Nahrávání je signalizováno červenou tečkou ve spodní liště programu.

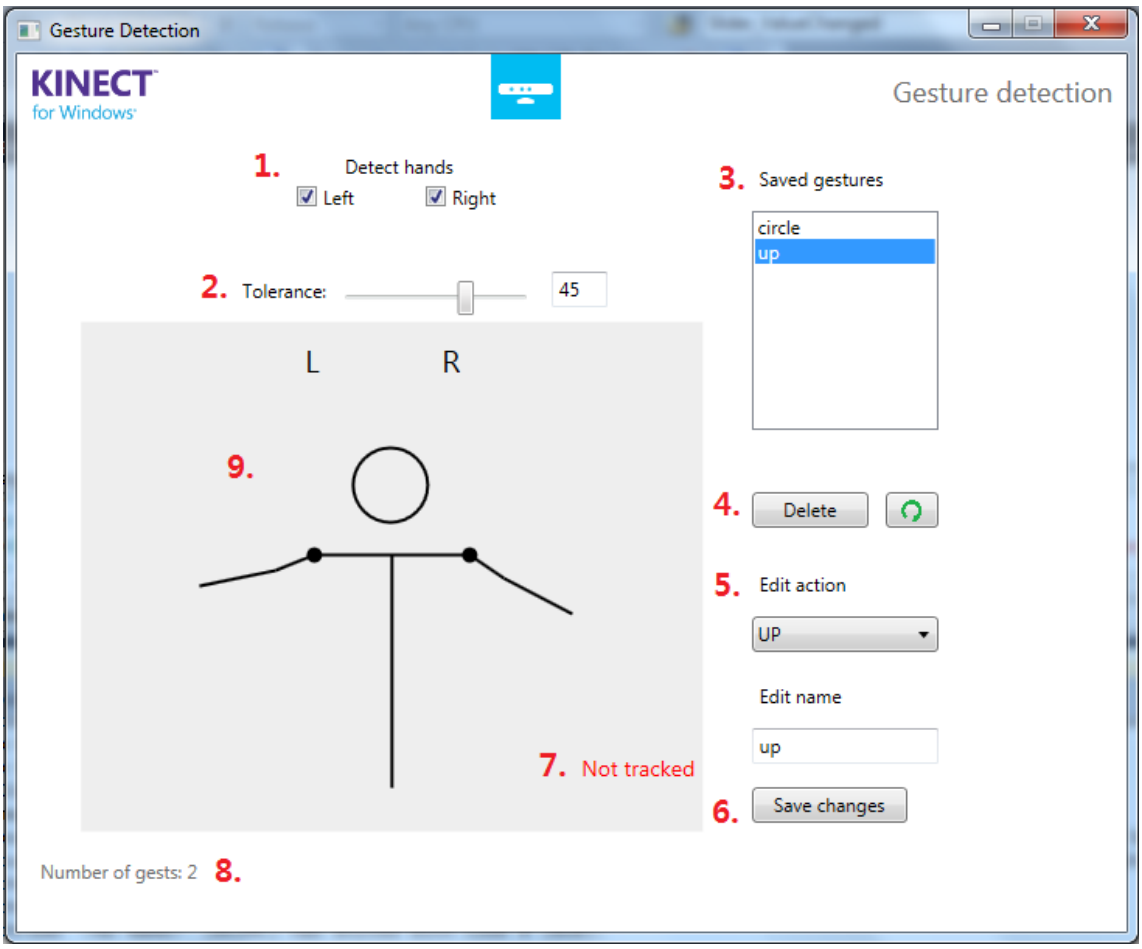
Kinect stop pak slouží pro ukončení a uložení gesta. Nově nahrané gesto se zobrazí v seznamu všech dostupných gest.

Kinect replay umožňuje přehrání označeného gesta bez fyzického kontaktu s počítačem.

Kinect hide minimalizuje aplikaci do „tray“ lišty a umožní tak nerušeně pokračovat v prezentaci.

Kinect show obnoví okno a nastaví ho do popředí.

V editační části nalezneme výpis všech dostupných gest. Pod tímto výpisem je tlačítko pro vymazání zvoleného gesta. Dále je zde tlačítko pro manuální přehrání gesta. Dále se zde nachází prvek pro přiřazení akce příslušnému gestu. V textovém poli se objeví při vybrání gesta jeho aktuální název a tento název můžeme změnit. Po provedení změny odpovídající akce nebo názvu gesta je nutné změny uložit. To se provede tlačítkem `Save changes`. Po vypnutí a opětovném zapnutí programu se ve výpisu gest nacházejí i gesta, která si uživatel nahrál. Pokud si uživatel není jistý, co tlačítko provádí, stačí na něj najet myší a zobrazí se bublina s nápovědou.




Obrázek B.1: Okno aplikace