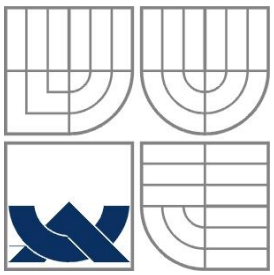
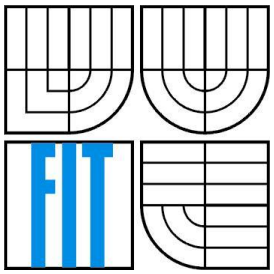


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

DETEKCE GEST PRO SENSOR KINECT

KINECT-BASED GESTURE RECOGNITION

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — FILIP ZAPLETAL
AUTHOR

VEDOUCÍ PRÁCE — ING. PAVEL ŽÁK
SUPERVISOR

BRNO 2012

Abstrakt

Tato práce popisuje způsoby získání dat o poloze jednotlivých částí těla za pomoci různých technologií, se zvláštním zaměřením na Kinect. Na základě těchto poznatků je pak navržena a implementována knihovna sloužící k rozpoznávání dynamických pohybových gest, která je otestována na sadě navržených gest.

Abstract

This paper describes methods of obtaining position data of individual body parts using different technologies, with particular focus on Kinect. Based on this knowledge was designed and implemented library for the detection of dynamic motion gestures, which has been tested on a set of designed gestures.

Klíčová slova

Kinect, gesta, rozpoznávání, Skryté Markovovy modely, detekce gest.

Keywords

Kinect, gestures, recognition, Hidden Markov models, gesture detection.

Citace

Zapletal Filip: Detekce gest pro senzor Kinect, bakalářská práce, Brno, FIT VUT v Brně, 2012

Detekce gest pro sensor Kinect

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Pavla Žáka. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……

Filip Zapletal
13. května 2012

Poděkování

Rád bych poděkoval Ing. Pavlu Žákovi za vedení této bakalářské práce, za uvedení do problematiky detekce gest, za doporučení možných postupů i literatury a v neposlední řadě za jeho cenné rady.

Obsah

1 Úvod

Klasické prostředky ovládání počítače, kterými jsou klávesnice a myš, tu s námi existují již více než 40 let. Jedná se o osvědčený a ve většině případů i velmi efektivní způsob ovládání, ale je nutné si přiznat, že za tuto dlouhou dobu se způsob práce s těmito periferními zařízeními, a tedy i způsob ovládání počítače téměř nezměnil.

Protože člověk byl a vždy bude tvor velice zvídavý, tak jej již od nepaměti lákaly nové, neotřelé technologie, a tím pádem i nové způsoby ovládání počítače, se kterým se v průběhu posledních několika let setkává stále častěji. Stačí se podívat například na některé sci-fi filmy z budoucnosti, kde se počítače často ovládají pouze pomocí hlasu nebo za pomoci pohybu a gest, které někdy bývají doplněny i trojrozměrnou projekcí. Pohyb a s ním související gesta jsou totiž člověku velice přirozené, neboť je používá v každodenní komunikaci se svým okolím. Tak proč by je nemohl používat i pro ovládání počítače? Tuto otázku si nejdříve položili a také realizovali tvůrci her a herních konzolí, protože v tomto způsobu ovládání her viděli velký potenciál. Začaly tak vznikat různé ovladače a senzory pro snímání lidského pohybu a softwarové prostředky pro jeho rozpoznávání.

Ovládání pohybem však asi nikdy (nebo minimálně v několika následujících letech) zcela nenahradí ovládání počítače konvenčními nástroji, jako je klávesnice a myš. Své uplatnění najde především v zábavním průmyslu, kde sloužit jako prostředek k ovládání her či jiných multimediálních systémů a aplikací. V produkčním prostředí jeho používání znepříjemňuje mimo jiné nízká přesnost a špatná ergonomie. Ale i přesto se jistě má smysl ovládáním pohybem a s ním souvisejícím rozpoznáváním gest zabývat.

Tato práce ve své první části (kapitole 2) nejdříve čtenáře seznámí s různými technologickými řešeními a přístupy, které slouží pro získání informací o poloze a pohybu z obrazu, který je často ještě doplněn informacemi z dalších senzorů. U jednotlivých řešení se pokusí zhodnotit jejich vhodnost právě pro rozpoznávání pohybových gest. V další části (kapitole 3) je popsán návrh knihovny pro senzor Kinect sloužící právě k rozpoznávání jednotlivých gest. V této kapitole jsou navrženy možné přístupy, z nichž je vždy některý vhodný zvolen a následně implementován. Způsob implementace je popsán v kapitole 4. Následuje popis a zhodnocení výsledků experimentů s navrženou a vytvořenou knihovnou v kapitole 5. Celá práce je ukončena závěrečným shrnutím a nástinem možností dalšího pokračování ve vývoji zde popsaného řešení.

2 Systémy pro detekci gest

Při detekci a rozpoznávání gest je třeba vyřešit obecně dva problémy. Prvním je samotné zpracování vstupních (často obrazových) dat do podoby, kdy jsou k dispozici pozice jednotlivých zájmových bodů. Druhým je pak sledování pozic těchto bodů v čase, tedy jejich pohybu, a následná klasifikace do tříd odpovídajících jednotlivým gestům.

Následující kapitola obsahuje popis řešení prvního problému, popisuje tedy jednotlivé způsoby sběru pohybových dat. Záběr popisovaných zařízení a principů je zúžen pouze na ta, která jsou běžně a finančně dostupná. Jsou tedy vynechány různá experimentální řešení a produkty, jejichž cena prozatím neumožnuje masové nasazení. U jednotlivých řešení bude popsán princip jejich funkce a budou zhodnoceny jejich výhody a nevýhody vůči ostatním.

2.1 Detekce pohybu v obraze

Základním přístupem k získání informace o scéně, s možností následné detekce gest, vychází z analýzy a zpracování obrazu samotného. Ke své funkci nepotřebuje žádný specializovaný hardware, lze jej použít i na běžné webové kameře s relativně nízkým rozlišením. O to komplikovanější je ovšem třeba použít algoritmy pro samotnou analýzu obrazu.

Pro realizaci detekce gest je nejdříve třeba v obraze určit minimálně jeden bod, na základě jehož pozice pak bude prováděno kýžené rozpoznání gesta. Bod by měl být ideálně obsažen ve všech zaznamenaných snímcích. Nalezení a přesné určení pozice tohoto bodu je jedním ze základních problémů, které je třeba vyřešit.

Jedním z možných řešení je identifikace tohoto bodu v rámci siluety postavy, což je při absenci třetího rozměru získávaných dat velice obtížné. Obrys siluety postavy lze však získat například za použití algoritmů pro detekci hran. I ten ovšem musí mít pro svou funkci splněno několik základních podmínek. Mezi ně patří mimo jiné relativně vysoká kvalita obrazu (především v podobě velmi dobrých světelných podmínek) a také požadavky na samotný obsah snímané scény, neboť je třeba zajistit, aby mezi popředím (tedy snímanou postavou) a pozadím byl zajištěn odpovídající kontrast, který umožní samotnou detekci hran. Určité problémy může dále způsobovat například i oblečení snímané postavy, které je často tvořeno více různobarevnými částmi. Je se také třeba vypořádat s faktem, že snímaná postava nemusí být pouze ve vzpřímeném postoji, ale může i sedět, což přináší další komplikace.

Problémů při sledování celé postavy je u tohoto řešení tedy více než dost. Jedním ze způsobů, jak se jim vyhnout, je místo sledování postavy samotné, sledovat pouze předen daný bod. Tím může být například míček, váleček či jiný tvar předem dané a dostatečně výrazné barvy. Tím se většina problémů redukuje na poměrně jednoduché sledování dobře známého a odlišitelného bodu. Tento způsob je také jedním z nejrozšířenějších v běžné praxi.

Výhodu tohoto řešení je, že k jeho používání není třeba řádné specializované zařízení a tím pádem jeho snadné rozšiřitelnost a nízká cena. Nevýhod je pak celá řada, ať již se jedná o absenci třetího rozměru, problematické rozpoznávání sledované postavy, ale i poměrně vysoká nepřesnost tohoto řešení a velké nároky na osvětlení scény. Tím vším je tedy velmi nevhodný pro detekci složitějších dynamických pohybových gest, kde je pro běžné použití požadována relativně vysoká přesnost, schopnost detekovat pohyb i v třetím rozměru i za mnohdy nekvalitních světelných podmínek.

2.2 Nintendo Wii

Systém Nintendo Wii firmy Nintendo [1] používá pro detekci pohybu speciální sadu ovladačů, které jsou vybaveny akcelerometrem. To je elekromechanické zařízení, které je schopné detekovat zrychlení daného zařízení, ideálně ve všech třech osách, a dle toho určovat předpokládaný směr pohybu.

V systému konzole Wii jsou ideálně použity dva ovladače, Wii Remote a Nunchak, které jsou propojeny kabelem. Doplňuje je 1 megapixelová kamera, která slouží pro určení polohy ovladačů. Celá tato sestava ještě může být doplněna speciální tlakovou podložkou Wii Balance Borard, která sbírá informace o náklonu těla a případném přesunu váhy.

Obrázek 1 – Ovladač Wii Remote a Nunchak [1]

Při použití všech těchto ovládacích částí lze poměrně úspěšně určit všechny potřebné údaje, tedy polohu (pomocí kamery), dynamické pohyby rukou (pomocí ovladačů Remote a Nunchak) a postoj – odhad pohybu celé postavy (pomocí tlakové podložky). Ač se tedy nejedná o skutečné sledování pohybu celé postavy, je díky použití akcelerometrů poměrně dobře možné určit gesta samotná, a to i v trojrozměrném prostoru (i když je snímán pouze dvourozměrnou kamerou).

Tento způsob detekce pohybu je tedy poměrně přesný a levný, a proto dal za vznik jedné z prvních komerčně úspěšných pohybem ovládaných herních konzolí. Prodej byl zahájen na konci roku 2006 a do konce roku 2011 bylo prodáno před 90 milionů kusů. Stala se tak nejvíce úspěšnou konzolí tohoto typu (v porovnání s Microsoft Xbox 360 a Sony PlayStation 3), přestože je na tom se samotným HW výkonem, o mnoho hůře. Je to přisuzováno právě možnosti použití pouze pohybu těla k ovládání her. Proto tato konzole inspirovala i ostatní hráče na trhu, kteří byli nuceni přijít s vlastním řešením ovládání pohybem.

Mezi nesporné výhody patří jednoduchost tohoto řešení, a tím pádem i relativně nízká cena. Mezi nevýhody pak patří fakt, že se nejedná o skutečné sledování pohybu celé postavy, ale pouze daných ovladačů, což redukuje možnosti využití pouze na gesta prováděná za pomoci rukou a částečně pohybem celého těla (při použití podložky).

2.3 PlayStation Move

Jedná se o proprietární rozšíření herní konzole PlayStation 3 od firmy Sony [2]. Jde se o kombinaci kamery s vysokou frekvencí snímkování (60 nebo 120Hz dle rozlišení) a speciálního ovladače. Ten má podobu válečku (pro snadné uchopení do ruky) na kterém je umístěna kulatá část, která emituje světlo v barvách rozsahu celého RGB barevného modelu. Konkrétní barva dynamicky závisí na okolí tohoto světelného bodu, a je vždy zvolena tak, aby byla od pozadí scény co nejvíce rozeznatelná a také aby byla pro každého, z až čtyř možných uživatelů odlišná.

Ovladač dále obsahuje i akcelerometr, pro určení směru pohybu ovladače, a magnetometr, pro určení jeho přesnější polohy v závislosti na magnetickém poli Země. Použitím těchto dvou senzorů lze zpřesnit údaje získané z kamery, ale i umožnit alespoň přibližný odhad pohybu v případech, kdy není ovladač v zorném poli snímající kamery (například je překryt jiným objektem).

Tím, že je přesně definována velikost snímaného objektu (tedy různobarevné koule) a snímající kamera má pevně dané ohnisko, lze i poměrně přesně na základě údaje o velikosti tohoto objektu v obraze určit jeho vzdálenost, což nám přidává potřebnou třetí dimenzi.

Použitím této technologie tedy získáme všechna potřebná data, a to i s relativně nízkými výpočetními nároky. Vyhodnocovací algoritmy totiž přesně vědí, co mají v obraze hledat. Vyhledání bodu tedy spočívá pouze v nalezení shluku bodů odpovídající barvy, vyhodnocení informací ze senzorů umístěných v ovladači a případné úpravě emitované barvy.

Použití ovladače vyzařujícího světlo v sobě dále přináší možnost použití i za nízkých světelných podmínek.

Obrázek 2 – Sada ovladače, kamery a samotného PS3 [2]

Mezi nevýhody lze tedy zařadit pouze fakt, že k snímání pohybu a následné detekci gest je třeba použít dodávaný ovladač do ruky. Tím jsou gesta redukována pouze na ta, kde lze použít pouze ruce a nemáme tak ve skutečnosti informaci o poloze těla, ale pouze o poloze tohoto ovladače. Ovšem i přes tyto nevýhody lze tento způsob ovládání s úspěchem použít. To dokazuje i fakt, že se za 10 měsíců po uvedení prodalo 8,8 milionů [3] těchto zařízení, takže je mezi uživateli poměrně oblíbené.

2.4 Microsoft Kinect

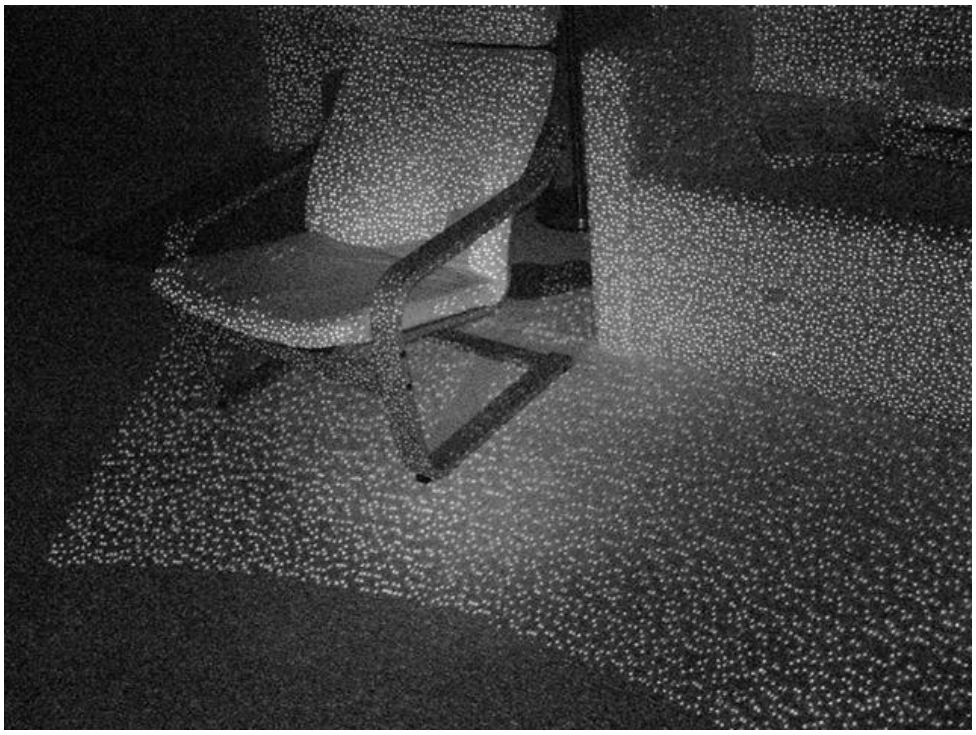
Kinect je pohybový senzor společnosti Microsoft, původně uvedený jako součást herní konzole Xbox 360. Na přelomu let 2011 a 2012 však byly vydány i oficiální ovladače a API ve formě SDK pro osobní počítače a systém Windows 7 pod označením *Kinect for Windows* [4]. Protože se jedná o senzor pro tuto práci nejpodstatnější, tak bude jeho funkce popsána podrobněji a rozdělena na popis hardwarové části (tedy senzoru samotného) a jeho softwarové části (popisující především API a způsob detekce polohy postavy v obraze). Protože se jedná o proprietární zařízení, tak k němu neexistuje oficiální podrobná dokumentace, tak lze popsat pouze obecné principy, často zjištěné za pomoci zpětného inženýrství.

2.4.1 Popis HW části senzoru

Senzor se skládá z několika částí. První je klasické RGB kamera s rozlišením 640x480 pixelů. Pro samotnou funkci rozpoznávání obrazu je v podstatě téměř nevyužívaná a slouží pouze jako zdroj obrazových dat pro případného uživatele nebo jako prostá webkamera.

Druhou je kombinace laserového infračerveného projektoru a infračervené kamery s rozlišením 640 x 480 pixelů. Ty tvoří jádro celého senzoru a složí jako primární zdroj dat pro vyhodnocování obsahu snímané scény.

Úkolem IR projektoru je osvítit snímanou scénu maticí pseudonáhodně, ale rovnoměrně rozmístěných bodů s různě velkou intenzitou. Ukázka této matice je na obrázku 3. Cílem je poskytnout sadu pro senzor ve snímaném obraze jasně definovaných bodů, které ovšem nebudou nijak rušit uživatele.



Obrázek 3 – Ukázka IR projekční matice senzoru Kinect [14]

Infračervená kamera umístěná na senzoru asi 5cm od projektoru poté tuto matici snímá a na základě pevného ohniska, perspektivního pohledu, znalosti přesného směru promítaného bodu a celkové struktury promítané matice, je schopna vypočítat polohu daného bodu v trojrozměrném prostoru. Výpočet samotný probíhá na úrovni senzoru a jeho výstupem je takzvaný *depth frame*, volně přeloženo hloubkový snímek nebo hloubková mapa. Ta je reprezentována jako dvoudimenzionální pole s rozměry odpovídajícími zvolenému rozlišení. Každý prvek je 16bitové číslo, kde horních 11 bitů [5] udává vypočítanou vzdálenost snímaného bodu od senzoru v milimetrech a spodní 3 bity, které jsou doplněny až aplikačním rozhraním, udávají index rozpoznané postavy (pokud je aktivováno jejich rozpoznávání, jinak jsou bity nulové). Ukázka hloubkové mapy převedené do berevného RGB modelu je k vidění na obrázku číslo 4.

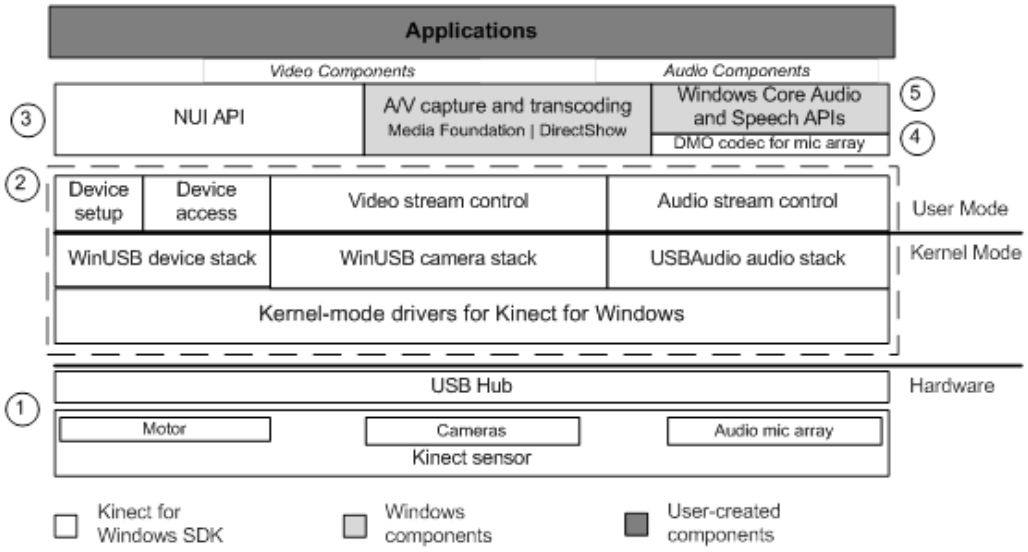
Obrázek 4 - Ukázka hloubkové mapy. Světlejší odstín odpovídá kratší vzdálenosti, zelenohnědé plochy jsou místa, kde nelze vzdálenost přesně určit (například je dané místo zastíněno jiným objektem).

Poslední, ale pro účely této práce nejméně významnou částí je zvuková část tohoto senzoru. Jedná se o sadu mikrofonů, jejichž účelem je určit směr (v horizontální rovině), ze kterého zaznamenávaný zvuk přichází. Ten je pak možné přiřadit odpovídající osobě rozpoznané v obraze a tak rozlišovat mluvčího. Po softwarové stránce je zvukový subsystém senzoru doplněn ještě o poměrně pokročilé API pro rozpoznání hlasu a umožněte tak doplňovat pohybové ovládání ještě tím hlasovým.

2.4.2 Softwarová část – Natural User Interface a jeho API

Kinect byl původně určen pouze pro herní konzoli Xbox 360. Jelikož ale pro připojení používá standartní USB rozhraní, tak zde byly, již po jeho uvedení, pokusy jej použít i ve spojení s osobním počítačem a začali tak vznikat různé, často *OpenSourcové*, knihovny a ovladače. Toho si všiml i sám Microsoft a na podzim roku 2011 vydal beta verzi oficiálních ovladačů a SDK pro PC. Plná verze byla vydána v únoru 2012 společně s upravenou verzí HW senzoru, který je schopen pracovat již ve vzdálenosti 40cm od senzoru (na rozdíl od cca 1 metru v případě konzolové varianty).

Informace o struktuře softwarové části Kinectu pocházejí z dokumentace, která je součástí dodávaného SDK [5]. Struktura je graficky znázorněna na obrázku číslo 5. Nejnižší část tvoří

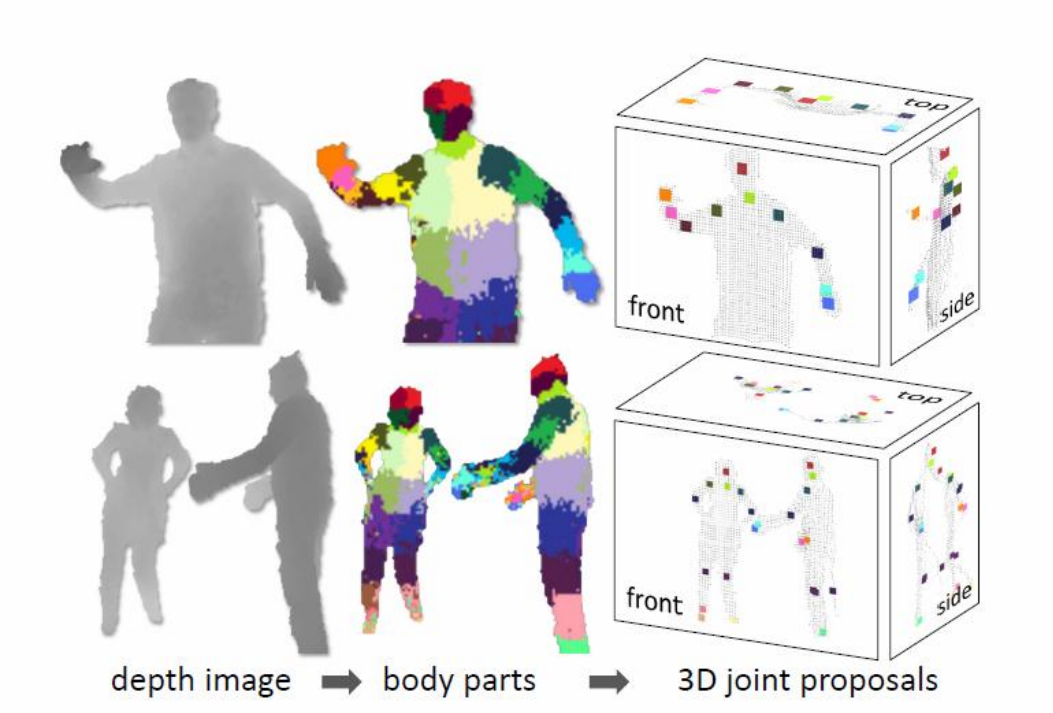


Obrázek 5 - Struktura SW části senzoru Kinect [5]

systémové ovladače pro systém Windows (na obrázku označené číslem 2). Ty se skládají ze systémové části (běžící v režimu jádra systému) a z uživatelské části (běžící v režimu uživatele). Jedná se především o části určené pro zpracování surových dat ze samotného senzoru. Nad nimi se nachází API, které je součástí procesu uživatelské aplikace – připojené jako DLL knihovna (na obrázku označená čísly 3 až 5). Skládá se z části *Natural User Interface* (NUI), která obstarává především *skeletonizaci*, audiovizuální interface, který se stará o správu a dekódování zvukových a obrazových dat a v neposlední řadě i API pro zpracování a rozpoznávání řeči. Pro účely této práce je využíváno primárně NUI, proto se ostatním částem API Kinectu věnovat nebudeme.

Primárním cílem NUI, nalezení a identifikace postavy a jejích částí. Tento proces se nazývá *skeletonizace* a dodávané ovladače ji jsou schopny provádět pouze pro dvě postavy ze šesti, především kvůli velké výpočetní náročnosti tohoto procesu.

Samotný proces skeletonizace není nikde veřejně publikován, protože se jedná o komerční produkt, lze ale vycházet z informací uvedených v [6] a základní postup je tak zobrazen na obrázku číslo 6. Ke své funkci používá převážně data získaná z hloubkové mapy. Její použití přináší mnoho



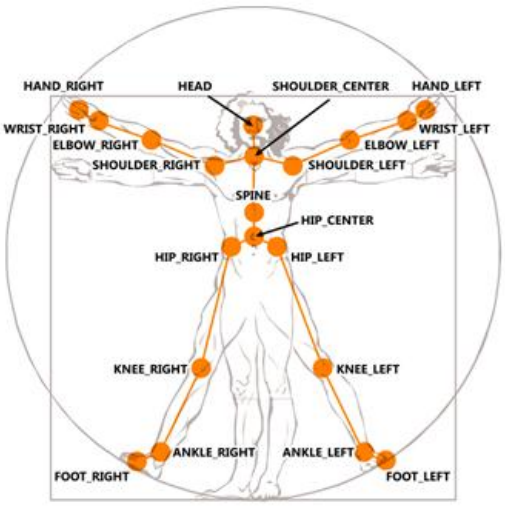
Obrázek 6 - Proces skeletonizace. Hloubková mapa je převedena pravděpodobnost určité části těla pro každý jednotlivý bod a dále převedena na odpovídající pozice bodů - jointů. [6]

výhod, mezi něž patří snadné oddělení postavy od pozadí, nezávislost na barevném rozložení scény a také schopnost pracovat při nízkém osvětlení scény. To vše za poměrně nízkých výpočetních nákladů v porovnání s RGB daty.

Identifikace částí těla je založena na velkém množství nasbíraných dat, zobrazujících velké množství různých póz, různě velkých a různě „tvarovaných“ postav. Na základě těchto dat je sestavena sada náhodných binárních rozhodovacích stromů, takzvaný les. Každý strom je tvořen rozdělovacími uzly (*split nodes*) a listovými uzly (*leaf nodes*). Rozdělovací uzly určují, zda se použije jejich levý či pravý podstrom na základě dat o okolí bodu v hloubkové mapě a jejich prahování. Listové uzly určují svůj podstrom na základě distribuční funkce, udávající relativní pozice sledovaných bodů v 3D prostoru, získané z trénovacích dat.

Dle průměru pravděpodobností všech rozhodovacích stromů v daném lese je poté určena, pro každý zkoumaný bod hloubkové mapy, pravděpodobná část těla, kterou reprezentuje. Z těchto bodů jsou pak vypočítány předpokládané pozice jednotlivých zkoumaných bodů skeletonu (takzvaných *jointů*) v trojrozměrných souřadnicích – viz obrázek číslo 7. Podrobnější způsob tohoto výpočtu včetně způsobu trénování je uveden v [6].

Výsledná data jsou poté k dispozici jako takzvaný *skeleton frame*. Každý snímek (*frame*) sestává z celkem 6 skeletonů, z nichž první dva nalezené jsou sledovány aktivně, tedy co nejpřesněji pro všechny jointy, zbytek pasivně (pouze základní informace o poloze postavy). Každý skeleton poté obsahuje seznam všech 20 jointů včetně jejich polohy v 3D prostoru.



Obrázek 7 - Body těla (jointy) používané jako součást skeletonu v NUI Kinectu [5]

Natural User Interface dále poskytuje metody pro práci se surovou hloubkovou mapou (s takzvaným *depth frame*) a jednotlivými *jointy*. Jedná se především o funkce přepočtu souřadnicových systémů mezi jednotlivými typy zobrazení. Lze tak s jejich pomocí transformovat 3D pozice *jointů* na 2D souřadnice odpovídající skutečné pozici v RGB video obraze.

Celá struktura tohoto API je v jazyce C# a prostředí *Microsoft .NET Frameworku* navržena dle norem objektového, událostmi řízeného výpočetního modelu. Pokud jsou tedy k dispozici jakákoli data (*RGB video, depth frame* či *skeleton frame*) je vyvolána odpovídající událost, pomocí níž jsou tato data předána. Je tedy zcela v režii programátora aplikace, jak s jednotlivými událostmi a daty naloží.

2.5 Zhodnocení

Cílem této kapitoly bylo popsání různých přístupů pro získání dat o pozici uživatele pro účely ovládání softwarové aplikace především z technologické a hardwarové stránky. Pro další část této práce bylo zvoleno zařízení Kinect společnosti Microsoft především proto, že splňuje následující podmínky:

- Je cenově dostupné i pro běžného uživatele.
- Poskytuje vysokou úroveň abstrakce z pohledu programátora, tedy část provádějící analýzu obrazu a následnou skeletonizaci je součástí jeho API a programátor aplikace je od tohoto problému ušetřen.
- Existují pro něj oficiální ovladače, API a SDK i pro osobní počítače, ne jen určitou herní konzoli.
- Je z pohledu uživatele nejzajímavější, protože k němu není třeba žádné další zařízení či ovladače – ovládá se pouze za pomoci vlastního těla a je schopen sledovat celé tělo, ne jen jeho určité části (například ruce).

Výběr tohoto senzoru ovšem neznamená, že dále popsané principy nejsou použitelné i pro jiné typy senzorů a zařízení, protože jsou založeny na obecném zpracování jednotlivých bodů v 3D prostoru, jejich zdroj je nepodstatný.

3 Návrh knihovny pro detekci gest

Cílem první části této kapitoly je prozkoumat možná řešení dílčích částí systému pro detekci gest, zhodnotit jejich klady a zápory, vybrat ta nejvhodnější a z nich celý systém navrhnout. Druhá část se zabývá návrhem API vytvářené knihovny tak, aby byla pro ostatní programátory co nejsnáze použitelná a přitom dostatečně transparentní a přizpůsobitelná (například aby bylo možno naprogramovat i rozpoznávání gest používající jiný, než zde zmiňovaný přístup). Poslední část kapitoly se bude zabývat návrhem samotného gesta, které je navrženo tak, aby byla samotná knihovna na implementaci gesta nezávislá, při dodržení definovaného rozhraní.

3.1 Návrh systému detekce gest a jeho částí

Jednotlivé části systému budou popsány v následujících podkapitolách, kde je také nastíněno, proč je z možných variant řešení zvolena právě zde popsaná. Ovšem pro snazší pochopení zde bude nejdříve popsána koncepce algoritmu jako celku.

Navržený algoritmus je uveden na diagramu číslo 1. Ten mimo jednotlivých částí a s nimi souvisejících akce znázorňuje i data, se kterými dané části pracují.

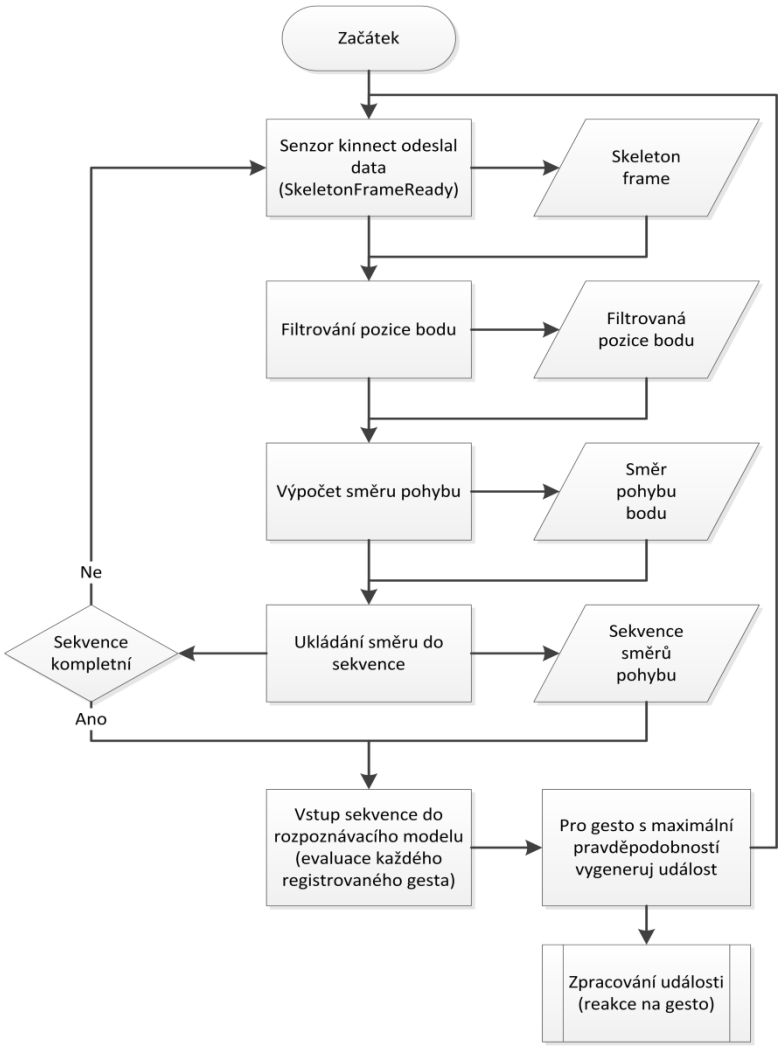


Diagram 1 – Základní algoritmus funkce knihovny se zmíněnými typy dat, se kterými dané části pracují.

Celý algoritmus je spuštěn událostí generovanou *Natural User Interface* API Kinectu v době, kdy jsou připravena výstupní data z procesu skeletonizace. Výstupem je samozřejmě celý skeleton (všechny body uvedené v obrázku 6), ale pro účely dalšího textu se omezíme na to, že sledujeme pouze jeden bod. Podobně je totiž navržena i daná knihovna – sleduje pouze ty body, které jsou pro ni důležité (jsou pro ně naprogramována gesta).

Takto získaná poziční data bodu jsou ovšem poměrně dost nestabilní, proto je na ně třeba aplikovat filtr. Tím pro každý snímek získáme pozici bodu v trojrozměrném prostoru. Z této pozice můžeme při znalosti pozice bodu v předchozím snímku vypočítat směr, kterým se daný bod pohyboval nebo zda se nepohnul vůbec.

Jsou tak získány vektory směrů (v tomto konkrétním případě se ovšem nejedná o vektory, ale pouze směry – viz podkapitola 3.1.2), které se dále ukládají do sekvence. Protože byl pro tuto knihovnu použit přístup, kde je začátek a konec sekvence pevně definovaný (více v podkapitole 3.1.3), je třeba jej nějakým způsobem určit. Jako začátek a konec tak bylo navrženo zastavení pohybu daným bodem po určitou dobu. Každé gesto tedy musí být ohraničeno klidnými pasážemi bez pohybu.

Jakmile je tedy sekvence ukončena, je předána části knihovny, starající se o vyhodnocení pravděpodobnost všech gest reagujících na pohyb daného bodu bod. Z těch je poté vybráno to, u něhož je pravděpodobnost rozpoznání nejvyšší a je pro něj generována událost, ve které na něj může uživatel (programátor aplikace) patřičným způsobem zareagovat. Daná sekvence je poté vyprázdněna a připravena pro rozpoznání dalšího gesta.

Do této chvíle jsme se ale zabývali každým bodem zvlášť, není tedy možné vytvořit gesto, které bude reagovat na pohyb více bodů zároveň, což je mnohdy požadovaná činnost, která však s sebou přináší velké množství problémů. Nejde ani tak o samotné vyhodnocení pravděpodobnosti gest, ale o to, jakým způsobem určit samotný začátek a konec sekvence.

Tady totiž výše popsaný systém s použitím pevně definovaného začátku a konce sekvence selhává, protože není možné u člověka v běžném prostředí nelze zajistit začátek a konec pohybu ve stejném snímku. Proto by pro tento způsob bylo třeba navrhnout systém jiný, nejlépe dynamicky se pro každé gesto zvlášť adaptující a využívající ne pevně definované sekvence, ale nad těmito sekvencemi plovoucí okna. Ta mohou být každé gesto různě velká. Takto vzniklé části se pak průběžně předávají mechanismu vyhodnocujícímu pravděpodobnost jednotlivých gest.

Od tohoto principu je ale v této práci upuštěno, protože je poměrně náročný na realizaci, testování a konfiguraci a sám by jistě vydal i na práci většího rozsahu než je tato. Koncepce ostatních částí této knihovny ovšem již dopředu s podobným principem počítá, a části, zabývající se vyhodnocováním pravděpodobnosti gest, byly již dopředu navrženy tak, aby s ním byly schopny pracovat bez většího zásahu.

3.1.1 Filtr pozice bodu

Jak již bylo nastíněno výše, je třeba vstupní pozici filtrovat. K tomu byl navržen lineární diskrétní filtr, jehož matematický zápis je uveden v rovnici následující rovnici:

$$Y = \frac{5X_i + 5X_{i-1} + 3X_{i-2} + 3X_{i-3} + X_{i-4} + \cdots + X_{i-(N-1)}}{12 + N}, pro\ N \in \mathbb{N}\ a\ N > 4 \qquad (3.1)$$

kde Y je výstupní pozice, X je vstupní pozice, i je index v paměti filtru a N je řád filtru, tedy velikost historie.

Filtr tedy pro každou pozici vypočítá vážený průměr s pozicemi v předchozích snímcích. Nejvyšší důraz je přitom kladen na pozici současného a předchozího bodu, s použitím starších bodů

jednotlivé váhy dále klesají. Spodní hranice je omezena jednotkovou váhou, avšak u tohoto filtru se nepočítá s řádem tak velkým, aby mělo toto omezení nějaký nežádoucí vliv.

Samotná filtrace pozice lineárním filtrem s sebou však přináší jeden velice nepříjemný problém. Pokud se totiž v následujícím kroku bude pro tuto pozici počítat vektor udávající změnu oproti bodu předchozímu, ovlivní každý (i nepatrný) pohyb ve výsledné sekvenci ne jeden, ale hned několik vzorků. Tím pádem se jakékoli rušení bude lavinovitě šířit celou výslednou sekvencí a způsobí tak poměrně velké množství chybných dat.

Z tohoto důvodu je třeba pro výstupní hodnoty filtru použít jistý způsob prahování. Pokud tedy velikost změny vůči předchozímu bodu, pro každý směr zvlášť, nepřekročí určitou definovanou mez, je výstupní hodnoty filtru nezměněna. Jinak řečeno existuje vhodně zvolená tolerance, a pokud se do ní změna pozice daného bodu vejde, je změna pohybu vyhodnocena jako nulová. Tato část funkcionality je navržena tak, aby byla součástí daného filtru, protože se ani tak netýká detekce směru samotného, ale ve skutečnosti se stále jedná o filtrování pohybu.

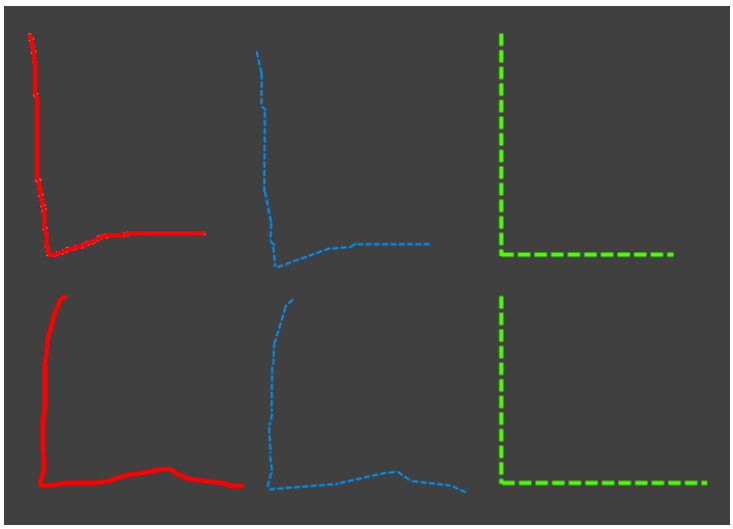
3.1.2 Určení směru pohybu a formát směrových dat

Pokud se obecně zamyslíme nad možnostmi, jakými lze reprezentovat pohybová data, pravděpodobně nás napadnou tyto:

- Reprezentace spojitou křivkou
- Reprezentace sadou směrových vektorů, kde je pohyb rozdělen na časové úseky a každá změna vlastním vektorem
- Reprezentace množinou diskrétních směrů, kde je pohyb také rozdělen na časové úseky, ale namísto vektoru ve spojitém prostoru, je pro určení směru použit prostor.

Jednotlivé způsoby jsou znázorněny na obrázku číslo 8. Reprezentace spojitou křivkou je sice velice intuitivní způsob, ovšem ne nejlepší již z pohledu toho, že data získaná ze senzoru Kinect nejsou spojitá (ostatně jako nic v číslicových počítačích). I když se totiž oprostíme od matematického významu spojité křivka jako takové, a představíme si ji již jako rasterizovanou, tedy trojrozměrné pole binárně udávající zda gesto daným bode prochází či ne, dostáváme se před další problém. A tím není nic jiného než samotné zpracování takových dat.

Bylo by je totiž třeba určitým způsoben normalizovat (jak časově, tak prostorově), což se sebou samo přináší určitou deformaci. Dále je ovšem tato data nějakým způsobem porovnat. Ať již



Obrázek 8 – Porovnání různých způsobů reprezentace pohybu. Červeně je zobrazena reprezentace téhož pohybu spojitou křivkou, modře lineárními směrovými vektory a zeleně diskrétní množinou směrů.

by se k tomu použil jakýkoli algoritmus, je v poměrně velkém (je potřeba dostatečné rozlišení i pro detekci málo výrazných gest ve značné vzdálenosti od senzoru) trojrozměrném prostoru, jeho použití v reálném čase na běžných počítačích značně nevhodné. Navíc gesta samotná si sobě odpovídají pouze logikou a ne provedením, viz například obrázek číslo 8, kde jsou červeně znázorněna pro uživatele totožná gesta, ovšem z pohledu spojitého systému dost odlišná. Proto je tento způsob značně nevhodný.

Druhou možností je data reprezentovat sadou trojrozměrných vektorů. Tím odpadá problém s konverzí dat ze senzoru samotného, jelikož způsob získávání dat po snímcích a tedy rozdělení průběhu scény na krátké časové okamžiky k uložení získaných vektorů do sekvence rovnou vybízí. S takovouto sekvencí se pak velice snadno pracuje a lze na ni snadno aplikovat různé již poměrně pokročilé techniky klasifikace, jako jsou například *spojité skryté Markovovy modely* (HMM), *dynamické borcení času* (DTW) či *neuronové sítě*. Ovšem stále se pohybujeme ve spojitém trojrozměrném prostoru (komplikované a náročné výpočty) a dané vektory je třeba před jejich použitím normalizovat (minimálně jejich velikost v závislosti na čase).

Příklad takové reprezentace je na obrázku 8 znázorněn modrou barvou. Jak jde vidět tak s sebou tento způsob sice nese téměř stejnou informační hodnotu jako červeně znázorněná křivka. Nese si však s sebou i stejný problém, a to že pro uživatelsky stejná gesta jsou v jistých úsecích velmi odlišná data. Ovšem tento způsob je již pro reálnou aplikaci použitelný.

Poslední zmiňovanou reprezentací je uložení dat do sekvence diskrétních směrů. Značně vychází z výše popsané sekvence trojrozměrných vektorů, jen místo reprezentace směru jako vektoru v prostoru spojitém je použit pouze omezený prostor diskrétní. V tomto případě se jako nejvhodnější jevilo použití celkem 8 směrů v každé rovině, celkem tedy 26 směrů (27, pokud započítáme i klidový stav – tedy žádný pohyb). Tím se stavový prostor dostatečně zmenší (v porovnání s jeho lineární verzí) a umožní velmi rychlou a efektivní evaluaci gesta samotného při zachování vysoké rozlišovací přesnosti.

Jak lze vidět na obrázku číslo 8, tak zeleně znázorněná gesta si ač vychází z různých lineárních křivek, odpovídají ze všech nejvíce. Jediné, v čem se liší, je počet, po kterém se jejich směr změní. V kombinaci s použitím v diskrétních skrytých Markovových modelech (HMM) je tento způsob jak nejefektivnější, tak i dostatečně přesný a na různé odchylky v gestu poměrně invariantní.

Další výhodou tohoto způsobuje je, že nepotřebuje ani žádnou výraznou normalizaci. Je to zapříčiněno fyziologií a způsobem lidského pohybu. V průběhu experimentů spojených s touto prací bylo totiž vypozorováno, že stejné gesto provádí člověk konstantní dobu, nezávisle na tom, jak velké ho dělá. Pokud tedy bude provádět gesto tvaru kružnice s poloměrem 10cm a stejné gesto, ale s poloměrem 20cm, tak jej provede za přibližně stejnou dobu, aniž by si to uvědomoval nebo to nějak vědomě ovlivňoval. V kombinaci s tím, že je snímkování a skeletonizace prováděna v přibližně stejně vzdálených časových okamžicích, získáme takto pro obě gesta velice podobnou a stejně dlouhou sekvenci.

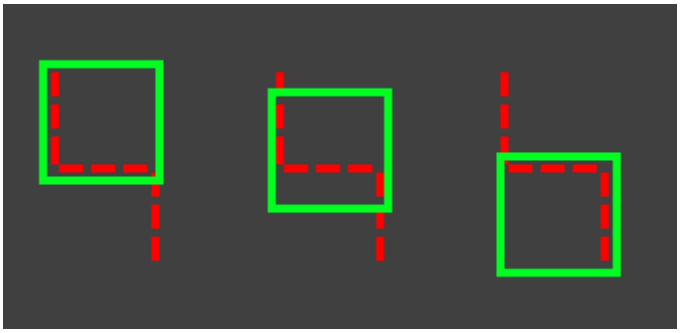
Samozřejmě si ale tento způsob reprezentace nese i své problémy. Mezi nejvýraznější asi na první pohled patří zanedbání poměrně velké části informace. Ovšem i s tímto zanedbáním může celý systém fungovat poměrně dobře, protože toto zanedbání paradoxně koriguje případné chyby snímání.

Samotná diskretizace ze směrového vektoru (získaného rozdílem současné a předchozí pozice filtrovaného bodu) je již poměrně intuitivní. Jediným problémem je snad jen diskretizace v trojrozměrném prostoru, ale ten lze vyřešit poměrně elegantně určením nejvýraznějšího směru jako hlavního a méně výrazné k němu buď připočíst, nebo ne v závislosti na jejich vzájemném poměru.

Pro účely detekce gest byla tedy jako nejvhodnější způsob reprezentace dat v této knihovně zvolena sekvence diskrétních směrových vektorů a ostatními způsoby se práce již dále nezabývá.

3.1.3 Sekvence směrů

Po filtraci a diskretizaci je tedy třeba uložit daný směr do sekvence. U té, jak již bylo zmíněno dříve, je nejzásadnějším problémem určení počátku a konce. Pokud se nebudeme zabývat vzájemnou interakcí těchto sekvencí a budeme je uvažovat jako oddělené, nezávislé celky (tedy budou i všechna rozpoznávaná gesta sledovat pouze jeden bod), tak máme k dispozici několik způsobů řešení.



Obrázek 9 – Znázornění určení okrajů sekvence (červeně) pomocí okna (zeleně). Jde pouze vizualizace, ne reálný přístup.

Prvním a nejkomplexnějším je použití plovoucího okna nad celou zaznamenanou sekvencí a je uvedeno na obrázku číslo 9. Spočívá v tom, po každém přidaném směru (tedy při každém snímku) je nad sekvencí proveden výřez, za pomoci takzvaného okna, které má buď pevnou, nebo pro různá gesta i rozdílnou délku.

Výřezem vybraná část sekvence je následně evaluována (pro každé gesto je vypočteny jeho pravděpodobnosti a na jejich základě pak vybráno gesto, kterému daná část sekvence odpovídá nejvíce). Výběr části sekvence je znázorněn na obrázku 9, kde je červěně znázorněna celá sekvence (její časový průběh je znázorněn tak, že sekvence začíná vlevo nahoře a končí vpravo dole) a zeleně je pak ohraničen daný výřez. Jedná se pouze o vizualizaci, protože sekvence je reprezentována jednorozměrným polem, kde jsou dané směry pro tyto účely irelevantní. Vlevo je výřez ve stavu, kdy výsledné gesto tvoří písmeno L, uprostřed je stav dané sekvence o jeden snímek později (jde tedy defakto již o jiné, neurčité gesto) a vpravo o další dva snímky později, se jedná již o gesto zcela jiné.

Výřez sekvence je tedy evaluován pro každé gesto v každý zaznamenaném snímku, což s sebou přináší poměrně vysoké výpočetní nároky. Další nepříjemnou vlastností je velké množství potencionálně chybně rozenznaných případů gest, zvláště pokud se jejich tvary vzájemně překrývají a následují ve sledu za sebou. Jak lze vidět na obrázku 9, tak stejná sekvence může takto reprezentovat velké množství dílčích gest. Pokud totiž bude rozpoznávané gesto celý červěně zvýrazněný tvar, ale budeme rozenávat gesto i ve tvaru písmene L, vznikne nám tu ona nejednoznačnost, která by se poté musela řešit.

Další nevýhodou je, že samotný začátek gesta není určen přesně, a je tedy vysoce pravděpodobné, že jednotlivá provedení stejných gest budou odlišně dlouhá. Při použití pevně definovaného okna bude pak výřez sekvence začínat v různých fázích prováděného gesta a bude zak velký problém s porovnáním samotným. Řešením by mohlo být například ignorování části sekvence na počátku a konci gesta, to ale v důsledku jen násobí problém předchozí, a tedy překrývání gest, protože odstraňuje i ty potencionálně malé odlišnosti v gestech si vájemně podobných.

Princip je tedy sám o sobě použitelný pouze s velkými obtížemi, ale jistě je využitelný v kominaci s některým z přístupů popsaných dále.

Jiný řešením je použití určité události sloužící pro identifikaci začátku a konce sekvence (a tedy i gesta). Touto událostí může být prakticky cokoliv, co je snadno rozeznatelné a může se jednat například změnu polohy sledovaného bodu vůči poloze bodů ostatních (tedy například natažení ruky

vpřed), ustálení polohy sledovaného bodu na určitou dobu nebo provedení nějakého úvodního gesta (které je rozpoznané třeba pomocí výše popsaného systému s plovoucím oknem).

Teoreticky se však nemusí jednat ani přímo o pohyb, s úspěchem lze použít třeba i zvukový impulz nebo stisk tlačítka. Ukončení sekvence je samozřejmě analogické, tedy například připažení či opět klidová poloha bodu.

Pro účely této knihovny je zvolen právě tento přístup, kde jako počáteční i ukončovací událost je právě klidová poloha bodu po dobu jedná vteřiny. Jedná se o velice jednoduchý, a přitom přirozený způsob separace jednotlivých gest. Pokud totiž ze začátku a konce takto zaznamenané a separované sekvence odstaníme místa, kde je daný bod v klidu, získáme tak pro další porovnávání sekvence vzájemě si podobné i na svých okrajích.

3.1.4 Evaluace gest

Nyní je tedy ke každému sledovanému bodu k dispozici sekvence, která reprezentuje jeho pohyb, která je v případě jejího ukončení (po ukončovací události – klidové poloze) připravena k samotné evaluaci.

Evaluací sekvence se rozumí její předložení vyhodnocovací metodě každého gesta, které na tento bod sleduje a následný výběr gesta nejpravděpodobnějšího. Logickou jednotkou výstupu vyhodnocovací metody gesta je tedy jeho pravděpodobnost, možné způsoby realizace jsou pak popsány v kapitole 3.3. Ty se mohou u různých gest lišit, ale vždy musí být zachována vzájemná srovnatelnost výstupů vyhodnocovací metody tak, aby bylo možné výběrem maxima vybrat gesto, které dopovídá dané sekvenci nejvíce.

Pro každou sekvenci je tedy určeno nejvíce pravděpodobné gesto, ačkoliv se o dané gesto vůbec jednat nemusí. Proto je třeba pro každé gesto určit určitou mezní (prahovou), hodnotu pravděpodobnosti, která tvoří hranici mezi úspěšně a neúspěšně rozeznaným případem.

Po určení pravděpodobného gesta je velmi žádoucí o tom nějakým způsobem informovat uživatele. Jako logické řešení, se v událostmi řízeném programovacím modelu, který dnes implementuje velká část obecně rozšířených programovacích jazyků, jeví vytvoření a emitování odpovídající události. Dále je velice vhodné, ideálně do argumentů této události, pokud to daný jazyk umožňuje, umístit informace jak o rozpoznaném gestu, tak i dodatečné informace o aktuální evaluaci – uživatele bude asi zajímat výsledná hodnota vyhodnocovací metody tohoto gesta a délka zaznamenané sekvence, ze které bylo dané gesto rozpoznáno.

Celý proces evaluace je navržen tak, aby byl co nejvíce transparentní a uživatel knihovny si tak samotnou reakci na gesta mohl přizpůsobit podle svých potřeb. K tomu jistě přispívá i možnost si u některých gest vynutit jejich rozpoznání – dané gesto bude tedy vyhodnoceno a událost pro něj emitována vždy, a je na uživateli, jak na ni bude reagovat, v závislosti například na jeho výsledné pravděpodobnosti.

3.2 Návrh API knihovny

Návrh aplikačního rozhraní vychází z použití objektově orientovaného, událostmi řízeného jazyka, který umožňuje daný návrh knihovny velmi dobře abstrahovat při zajištění pro uživatele co největší možnosti dalších úprav při snadné použitelnosti.

Samotná gesta jsou navržena jako zcela samostatné celky, které sdílejí pouze definované rozhraní a jejich vnitřní implementace je zcela v režii autora tohoto gesta. Návrh aplikačního rozhraní gest je proto uvedeno v samostatné kapitole 3.3.3.

Rozhraní knihovny je navrženo tak, aby bylo tvořeno instancí třídy API samotného. Tato třída by měla umožňovat následující:

- Nastavení a změnu knihovnou aktuálně používaného senzoru Kinect.
- Možnost generování událostí a přípravu dat pro:
  - Výstup RGB obrazu z kamery senzoru.
  - Výstup vizualizace skeletonů a jejich mapování na RGB obraz.
  - Vizualizaci sekvencí.
- Snadnou registraci (přidání gesta do knihovny tak, aby jej bylo možné rozpoznávat) i jiným autorem navržených a implementovaných gest.
- Možnost záznamu sekvencí pro účely učení HMM, neuronových sítí či jiných genetických algoritmů použitých v jednotlivých gestech.

Cílem je tedy navrhnout takovou strukturu knihovny, aby obsahovala všechnu výše uvedenou funkcionalitu, která by při tom měla být dostupná přímo z instance hlavní třídy knihovny. Zbylou funkcionalitu popsanou v předchozích kapitolách by bylo vhodné rozdělit na jim odpovídající části, tedy třída pro filtr pozice bodu, třída pro diskretizaci směru, třída reprezentující sekvenci a v neposlední řadě třída reprezentující hráče (člověka) jako takového, která bude zajišťovat vzájemné propojení včetně rozhraní pro evaluaci gest.

3.3 Návrh gesta

Jak již bylo popsáno, gesto je navrženo tak, aby tvořilo ideálně ucelenou a na jiných částech knihovny zcela nezávislou entitu s jedinou výjimkou, kterou je společného rozhraní. Vnitřní implementace tak může být pro každé gestu libovolná při dodržení podmínky, že výstupy hodnot pravděpodobnosti jednotlivých vyhodnocovacích metod budou vzájemně srovnatelné.

V této kapitole budou popsány jak jednotlivé způsoby implementace vyhodnocovací metody, tak v závěru i navržené rozhraní, které by měla jednotlivá gesta implementovat.

3.3.1 Návrh sady gest

Pro účely ověření návrhu a testování byla navržena sada gest, které jsou detailně popsány v Příloha A. Výběr možných gest a pohybů byl navržen s ohledem na vyloučení si vzájemné podobnosti jednotlivých pohybů, aby se snížilo riziko chybného rozpoznávání. Například mezi gesty vyjadřujícími kruh a elipsu je z pohledu většiny porovnávacích algoritmů, ale i uživatele minimum rozdílů. Tento přístup návrhu sady výlučných gest by měl být použit i v praxi.

Všechny gesta byla navržena na sledování jednoho bodu – pravé ruky. Důvodem je větší názornost následného testovaní, kde by se v případě použití více sledovaných bodů pro různá gesta snížil počet těchto gest v rámci jednoho bodu a tím i zjednodušilo samotné rozpoznání. Tento efekt lze naopak velmi dobře použít v praxi, kde jednotlivá gesta rozložit na více částí těla a tím získat rozlišitelnost i u vzájemně si podobných (ne-li stejných) gest.

3.3.2 Algoritmy pro rozpoznání gest

Tato kapitola obsahuje popis některých systémů a algoritmů použitelných pro porovnání dvou sekvencí složených z prvků diskrétního stavového prostoru, s primárním cílením na použití při porovnávání dynamických pohybových gest. Komplexní přehled technik použitelných pro rozpoznávání gest lze nalézt v [7]. Jedná se velmi často o algoritmy používané i pro rozpoznávání řeči.

3.3.2.1 Konečné automaty

Konečné automaty (FSM) jsou jedním ze základních výpočetních modelů sloužících pro zpracování sekvence konečné množiny vstupních symbolů. V případě rozpoznávání gest se většinou jedná o jejich nedeterministickou verzi, která je formálně popsána v [8] jako:

$$A = (Q, T, \delta, s, F), \text{ kde:}$$

Q je konečná množina (abeceda) stavů
T je konečná množina (abeceda) vstupních symbolů
δ je přechodová funkce $Q \times T \rightarrow 2^Q$, kde 2^Q označuje množinu všech podmnožin množiny Q
s je počáteční stav, kde $s \in Q$
F je množina koncových stavů, kde $F \subseteq Q$

V případě klasifikace gest, kde je gesto tvořeno sekvencí směrů, jsou vstupní abecedou právě tyto směry. Konečný automat pak přechází mezi jednotlivými stavy dle posloupnosti vstupních symbolů (směrů) a skončí nebo neskončí v některém z koncových stavů. Pokud je na konci takovéto sekvence v konečném stavu, je výsledkem vyhodnocení předem definovaná hodnota pravděpodobnosti pojící se s tímto koncovým stavem. Pokud pro danou sekvenci FSM v konečném stavu neskončí, pak je výsledná pravděpodobnost nulová.

Takto definované konečné automaty poskytují velice přesné výsledky, ale vyžadují také velmi exaktní zdroj dat, čímž pohyby rozhodně nejsou, díky velmi velké pravděpodobnosti odchylek v jednotlivých provedeních téhož gesta. Aby se tento jev eliminoval, je třeba vytvořit poměrně velké množství stavů a přechodů, při čemž se pak v případě drobné odchylky bude celý FSM schopný vrátit do požadovaného stavu. Vzhledem k tomu, že je třeba konečný automat definovat ručně pro každé gesto a počet variací těchto odchylek je teoreticky nekonečný, je velmi problematické dobře fungující FSM pro účely detekce gest vytvořit.

Problém lze částečně eliminovat použitím komplexnějšího předřadného filtru, který bude počet těchto odchylek minimalizovat, ale nelze se jim vyhnout úplně. I přes tyto problémy však použití FSM pro vyhodnocovací metodu jednotlivých gest možné je.

3.3.2.2 Dynamické borcení času – DTW

Tato metoda (popsána například v [9]) slouží k rozpoznání jednotlivých izolovaných úseků (vzorků) na základě vzorku referenčního. Je principiálně založena na rozdělení sekvence na krátké úseky, kde jejich délka je normalizována tak, aby byly všechny úseky stejně dlouhé, a byly tedy nezávislé na čase. Algoritmus pak vypočítává vzdálenost mezi jednotlivými prvky (vektory) testovaného a referenčního signálu dle následujícího vzorce (převzato z [9]):

$$D(T,R) = \left[N(\widehat{W})\right]^{-1} \cdot \left\{ \min_{i(n),j(m)} \sum_{n=1}^{I} d[i(k), j(k)] \cdot \widehat{W}(k) \right\} \tag{3.2}$$

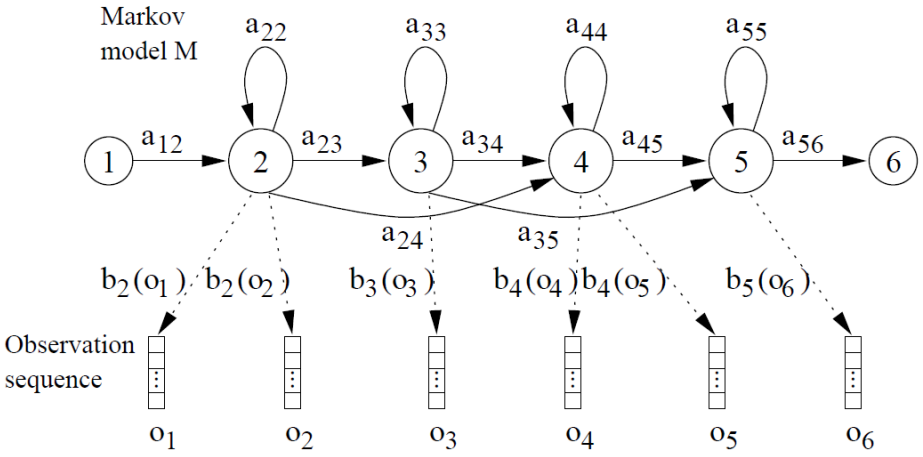
kde D je celková vzdálenost mezi testovanou sekvencí T a referenční sekvencí R, $N(\widehat{W})$ je normalizační vektor pro kompenzaci dálky jednotlivých kroků algoritmu, $\widehat{W}(k)$ je váhová funkce a d je lokální vzdálenost mezi jednotlivými prvky (vektory) testované sekvence.

Pro každé gesto tak existuje jedna referenční sekvence, která jej reprezentuje a pomocí výše popsaného algoritmu se vypočítá vzdálenost neboli jeho podobnost s aktuálně zpracovávanou sekvencí. Po normalizaci je tak možno získat kýženou hodnotu pravděpodobnosti.

3.3.2.3 Skrytý Markovův model – HMM

Jedná se o statistický model podobný konečnému automatu, ale na rozdíl od něj, jsou konkrétní stavy skryty. To znamená, že jejich význam a chování je v době návrhu modelu neznámý.

Na obrázku 10 jsou čísly 1 až 6 znázorněny stavy. Pro každý stav (s výjimkou počátečního a koncového) existuje pravděpodobnostní model, který udává, zda algoritmus v daném stavu setrvá ($a_{i,i}$), přesune se do stavu následujícího ($a_{i,i+1}$) nebo dokonce i nějaký stav přeskočí ($a_{i,i+2}$).



Obrázek 10 – Ukázka Skrytého Markovova modelu (HMM) [10].

Jednotlivé pravděpodobnosti se uvádějí do matice znázorněné na obrázku číslo 11. Jak si lze všimnout, tak všechny přechodové pravděpodobnosti se nacházejí na diagonále nebo pod ní, tudíž se

$$\mathbf{A} = \begin{bmatrix} 0 & a_{12} & 0 & 0 & 0 & 0 \\ 0 & a_{22} & a_{23} & a_{24} & 0 & 0 \\ 0 & 0 & a_{33} & a_{34} & a_{35} & 0 \\ 0 & 0 & 0 & a_{44} & a_{45} & 0 \\ 0 & 0 & 0 & 0 & a_{55} & a_{56} \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Obrázek 11 – Ukázka přechodové matice [10].

v modelu se nelze vracet do stavu předchozího.

Tato přechodová matice je dále doplněna o tzv. vysílací pravděpodobnost (*emmision likehood*) $b_j[o(t)] = p(o_t|j)$, jejíž funkce hustoty rozložení pravděpodobnosti udává, že daný vektor (vzorek) s vlastností *o(t)*, se bude v čase *t* nacházet ve stavu *j*. V závislosti na typu HMM se bude jednat o konkrétní hodnoty pravděpodobností pro modely diskrétní nebo o směsici Gaussových funkcí uvedenou v rovnici 3.3 převzaté z [10].

$$b_j[o(t)] = \sum_{i=1}^{M} \alpha_{ij} N(o(t), \mu_{ji}, \Sigma_{ji}) \tag{3.3}$$

Konfigurace HMM se získává z trénovacích dat, za pomoci *Baum-Welchova algoritmu*. Ten pro každý trénovací vzorek spočítá pomocí *Forward-Backward algoritmu* hodnoty pravděpodobností a dle nich pro každý stav a čas upraví střední hodnoty a rozptyl. Na základě těchto hodnot je pak

proveden výpočet nových parametrů. Celý proces se opakuje do chvíle, kdy je model konvergentní nebo splňuje námi definovanou úroveň přesnosti.

Dekódování pak probíhá na základě *Viterbiho algoritmu*, který slouží k nalezení cesty mezi jednotlivými stavy (tedy posloupnost stavů) na základě určité vstupní sekvence. Dekódování ale není pro potřeby rozpoznávání gest klíčové (zajímá nás pouze pravděpodobnost, s jakou sekvence odpovídá danému modelu), proto se mu tato práce nebude dále věnovat.

Klíčovou součástí je naopak výpočet pravděpodobnosti, s jakou byla aktuálně testovaná sekvence generována právě tímto modelem. Ta se vypočítá například za pomoci *Forward-Backward algoritmu*. Protože se většinou jedná o velice nízké hodnoty, je většina výpočtů prováděna v logaritmickém měřítku (nejde tedy o skutečnou hodnotu pravděpodobnosti, ale její logaritmus). Tím je zajištěna dostatečná přesnost a nízká ztrátovost informace, která jinak hrozí při aritmetických výpočtech ve *floating point* aritmetice u řádově velmi rozdílných čísel.

Při použití HMM pro detekci gest má ve finále každé gesto svůj vlastní model s vlastní konfigurací, která je založena na trénovacích datech. Tato data jsou v případě této knihovny diskrétní sekvence směrů, takže se většinou bude používat primárně diskrétní verze modelu. Posledním problémem je určení přibližného počtu skrytých stavů. Jejich nízký počet snižuje přesnost modelu, vysoký zase zvyšuje výpočetní nároky. Na základě [11] lze ovšem usoudit, že ve většině případů jednoduchých gest je počet skrytých stavů mezi čtyřmi až šesti dostatečný.

3.3.3 Návrh rozhraní vyhodnocovací části gesta

Protože vnitřní implementace gesta může používat různé přístupy, je gesto navrženo tak, aby jeho jediným styčným místem s navrhovanou knihovnou bylo rozhraní splňující následujíc podmínky:

- Gesto by měl být instancovatelný objekt, který je ve svých instancích vzájemně nezávislý.
- Objekt by měl obsahovat informace o daném gestu, jako je jeho jméno, identifikační číslo, jeho předpokládaná délka a práh pravděpodobnosti (viz kapitola 3.1.4).
- U gest používajících učící se (trénovací) algoritmy, by mělo být možné tato gesta trénovat na základě předané sekvence a následně konfiguraci uložit do souboru a následně ji také opět načíst.
- Gesto by mělo samozřejmě obsahovat vyhodnocovací metodu, jejímž výstupem bude pro předanou sekvenci logaritmus pravděpodobnosti (logaritmus kvůli problémům s výpočty ve velmi nízkých řádech).

Těmito podmínkami je v podstatě definováno rozhraní, které by bylo vhodné definovat i formálně, ať již pomocí dědičnosti, abstraktní třídy či přímo použití *interface* (v závislosti na použitých prostředcích implementačního jazyka).

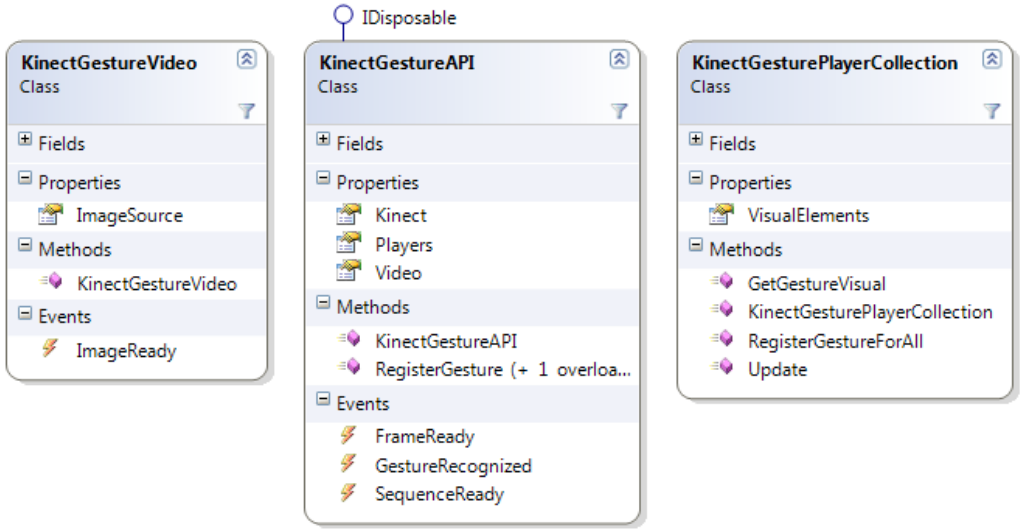
4 Implementace

Implementace knihovny pro detekci dynamických gest a doplňkových nástrojů vychází z principů a postupů navržených v předchozí kapitole. Jako implementační prostředí byl zvolen *Microsoft .NET Framework 4* a *jazyk C#*. Hlavními důvody použití tohoto frameworku a jazyka byly jeho dobrá podpora jak ze strany dodávaného API senzoru Kinect a fakt, že jde o moderní objektově orientovaný jazyk umožňující vysokou úroveň abstrakce při zachování rychlého, agilního vývojového cyklu. V jeho prospěch hraje i fakt, že obsahuje grafické rozhraní, které bylo nutné pro vytvoření demonstračních aplikací.

Mezi ně patří jak nástroj pro záznam gest, tak i jednoduchá aplikace demonstrující použití knihovny pro samotné rozpoznávání. Primární část práce je ale tvořena samotnou knihovnou, které je dále distribuovaná ve formě DLL assembly, a proto bude popsána nejdříve.

4.1 Implementace knihovny pro detekci gest

Samotná knihovna je navržena tak, aby její rozhraní bylo tvořeno jednou třídou, která zaštiťuje veškerou požadovanou funkcionalitu (samozřejmě ve spolupráci s třídami dalšími – ty však uživatel používat nemusí, pokud vyloženě nechce). Rozhraní je zobrazeno formou diagramu tříd na obrázku číslo 12.



Obrázek 12 – Diagram tříd rozhraní knihovny. Vlevo třída pro práci s RGB obrazem, vpravo třída reprezentují kolekci jednotlivých hráčů a uprostřed třída samotné knihovny. Zobrazeny jsou pouze význačné členy.

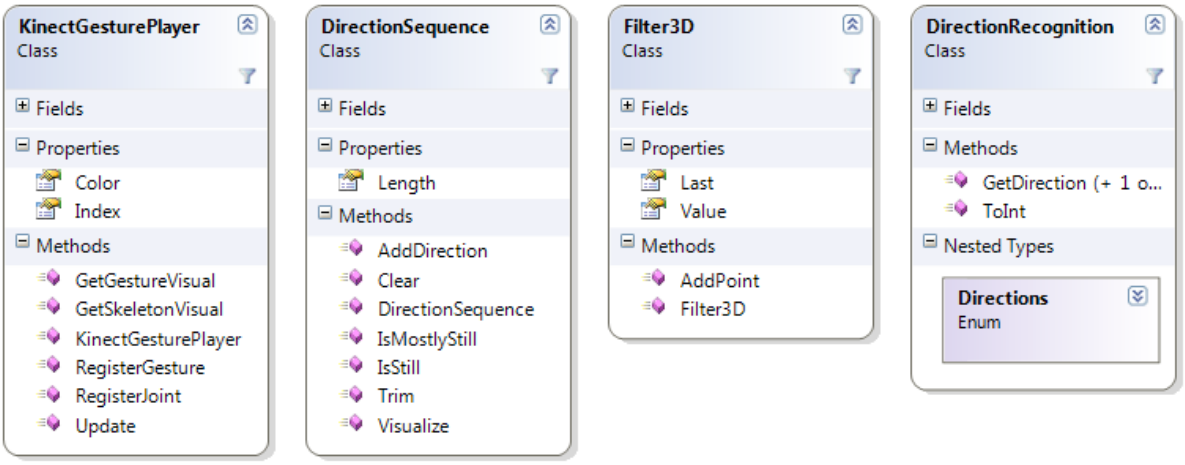
Celé API knihovny se instancuje pomocí třídy `KinectGestureAPI`. Senzor Kinect je zde reprezentován vlastností `Kinect`. Ta je dle doporučení společnosti Microsoft i pro zápis a umožňuje tak použití více na sobě nezávislých senzorů v různých instancích knihovny, ale i dynamickou změnu zařízení za běhu. Třída dále obsahuje člen typu `KinectGestureVideo`, který se stará o zpracování obrazu z RGB kamery.

Protože je senzor Kinect schopen pracovat až se šesti uživateli zároveň (z nichž u dvou je prováděna skeletonizace), je pro každého takového uživatele – hráče vytvořena samostatná instance rozpoznávacího mechanismu. Ty jsou pro zvýšení míry abstrakce sdruženy ve společné třídě `KinectGesturePlayerCollection`.

Poslední významnou metodou celé knihovny je `RegisterGesture`, která se stará o takzvanou registraci gesta. Tou se rozumí zavedení předaného gesta do celého rozpoznávacího mechanismu pro všechny, i aktuálně neaktivní hráče.

4.1.1 Implementace třídy hráče

Jako hráč je v prostředí Kinectu chápána jedna postava. Každá z těchto postav – hráčů by tedy měla mít i vlastní, nezávislou instanci rozpoznávacího algoritmu. Ta je v rámci knihovny implementována v třídě `KinectGesturePlayer`, jejíž diagram tříd je uveden na obrázku číslo 13.



Obrázek 13 – Diagram tříd pro interně používané typy.

Ta obsahuje jak metody sloužící pro vizualizaci a registraci gest (data získaná za pomoci skeletonizace jsou sem předávány pomocí metody `Update`) i jádro rozpoznávacího algoritmu. To je tvořeno několika instancemi třídy `DirectionSequence`, kde pro každý sledovaný bod (sledované body jsou ty, které si jednotlivá gesta zaregistrují) je vytvořena jedna instance této třídy, rovněž uvedené na obrázku 13.

Jedná se o implementaci návrhu uvedeného v kapitole 3.1.3, konkrétně o variantu, kde je začátek a konec sekvence pevně definován jako část s délkou 20 vzorků (při 20FPS tedy 1s), ve které je více než 90% vzorků klidných. Tato konfigurace se praktickými testy osvědčila jako velmi vhodná, protože její délka je dostatečně velká k tomu, aby ji bylo možno rozpoznat, a přitom není natolik velká, aby uživatele obtěžovala.

Třída obsahuje i implementaci mechanismu (metoda `Trim`) pro ořezání klidných částí ze začátku a konce sekvence. Klidná část je chápána jako sekvence, ve které je poměr klidných vzorků vůči vzorkům udávajícím směr (tedy ostatních) větší než 1/3. V sekvenci je tedy více než 30% klidných vzorků (pro sekvenci delší než 2 vzorky). Tato konfigurace byla opět dle výsledků experimentů shledána jako vyhovující.

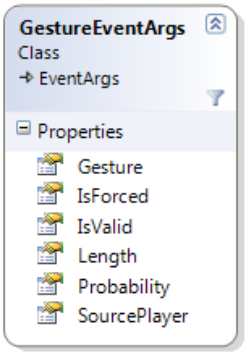
Zároveň s instancí sekvence je vytvořena i instance filtru `Filter3D`. Jednotlivé velikosti vah navržené v kapitole 3.1.1 byly ověřeny experimentálně, stejně tak jako nejlépe vyhovující délka historie (řád filtru). Jako nevhodnější se jeví řád $N = 3$, který poskytuje dostatečnou filtraci při dobrém reakčním čase – vyšší řády způsobovaly nepříjemně dlouhou prodlevu. Jak jste si ale mohli všimnout, tak v návrhu popsaná rovnice číslo 1 s tak nízkým řádem nepočítá, proto je v následné implementaci použita její upravená verze.

Vyfiltrované polohy jednotlivých bodů jsou za pomoci statické třídy `DirectionRecognition` a jeho metody `GetDirection` vzájemně porovnány (filtr slouží zároveň jako úložiště pro pozici bodu v předchozím snímku) a diskretizovány do 26 směrů (27 včetně

klidného stavu, tedy žádného pohybu). Jejich počet vychází z použití právě osmi směrů v každé rovině a byl následně experimentálně ověřen jako vyhovující.

Všechny výše popsané třídy dále obsahují i metody pro vizualizaci jak jednotlivých sekvencí, tak i dat získaných prostřednictvím skeletonizace a vizualizaci pozice filtrovaného bodu, které jsou pro názornost namapovány na skutečnou polohu v RGB obraze z kamery senzoru.

Součástí třídy hráče je i evaluační algoritmus navržený v kapitole 3.1.4. Pokud je tedy daná sekvence ukončena a neskládá se z převážně klidných vzorků, je ořezána a převedena na pole kladných celých čísel (které jsou pro účely vyhodnocovacích metod jednotlivých gest vhodnější). Pole je poté předáno k vyhodnocení každému registrovanému gestu (to, zda sekvenci použije nebo ne ověří v závislosti na typu *jointu* (bodu), jehož pohyb daná sekvence reprezentuje) a následně je pro každé gesto vypočítána jeho pravděpodobnost. Zde nastávají dvě možnosti – pokud je evaluace daného gesta vynucená, je pro něj ihned vytvořena a emitována odpovídající událost, pokud gesto vynucené není, tak je tato událost emitována jen pro gesto, jehož pravděpodobnost byla nejvyšší.

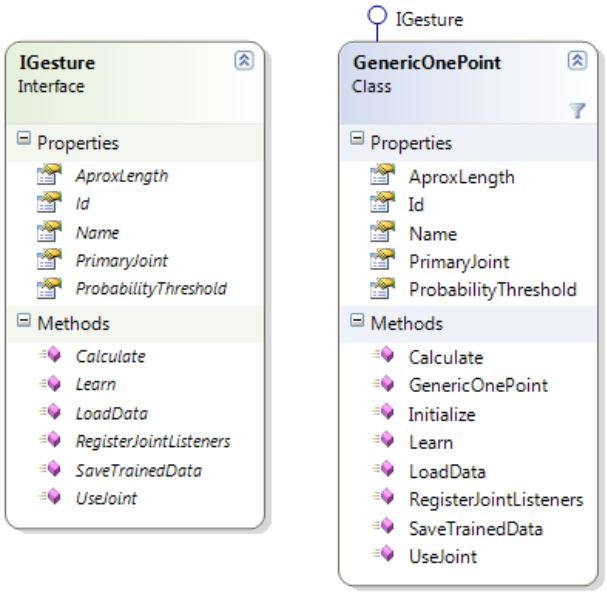


Obrázek 14 – Třída reprezentující argumenty události vyvolané při rozpoznání gesta.

Součástí události je i instance třídy `GestureEventArgs` uvedené na obrázku číslo 14. Ta je naplněna základními daty o aktuálně rozpoznaném gestu, délce sekvence, ze které bylo rozpoznáno, výstupní pravděpodobnost vyhodnocovací metody gesta a také zda byla daná událost vynucena nebo jde o regulérní vyvolání na základě maximální pravděpodobnosti. Obsahuje také atribut určující platnost rozpoznání (zda aktuální pravděpodobnost rozpoznání je vyšší než prahová, která je součástí gesta) a v neposlední řadě i identifikaci hráče, který dané gesto provedl.

4.1.2 Rozhraní gest a implementace generického gesta

Pro všechna gesta bylo vytvořeno rozhraní uvedené na obrázku 15 vlevo. Rozhraní neboli *interface* je speciální konstrukt jazyka C#. Definuje členy, které třídy odvozené (tedy implementující toto rozhraní) musí bezpodmínečně implementovat. Ve své podstatě se jedná o abstraktní třídu, které má abstraktní všechny její členy.



Obrázek 15 – Rozhraní gest (vlevo) a implementace tohoto rozhraní ve formě generického gesta

Rozhraní je navrženo a implementováno tak, aby obsahovalo vlastnosti a metody jak pro rozpoznávání samotné (jde především o metodu `Calculate`, která obstarává výpočet pravděpodobnosti gesta pro danou sekvenci), tak i pro učení u algoritmů, které ho využívají (metoda `Learn`) a v neposlední řadě i metody pro případné uložení a načtení vnitřní konfigurace gesta. Samotný výběr bodů (jointů), se kterými bude gesto pracovat, pak probíhá v metodě `RegisterJointListeners`.

Některé z těchto metod však nemusejí být v případě některých gest vždy potřebné, například metody pro učení a ukládání

trénovaných dat nejsou potřeba, pokud je gesto implementováno za pomoci algoritmů, které mají svoji konfiguraci pevně definovánu, jako je například použití konečných automatů. Rozhraní je ale i tak obsahuje, aby byla i v těchto případech zachována jednotnost a uživateli byly tyto metody transparentní, i když používá gesto vytvořené někým jiným. Pokud tedy dané metody nejsou pro dané gesto využitelné, musejí být i tak implementovány alespoň jako metody s prázdným tělem.

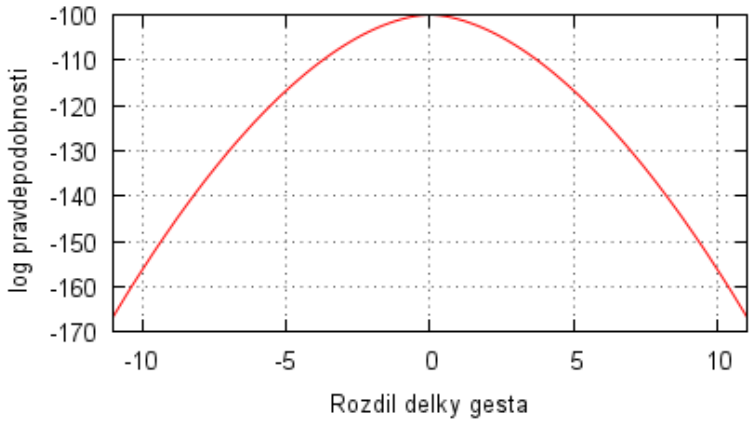
Jako ukázkové bylo implementováno generické gesto pracující s jedním bodem, které je ve formě diagramu tříd uvedené na obrázku 15 vpravo. Gesto jako klasifikační mechanismus používá diskrétní *Skryté Markovovi modely*, konkrétně implementaci popsanou v [12], která je součástí *Accord.NET Frameworku* [13]. Gesto je navrženo jako generické, lze tedy po vytvoření nové instance, nastavení parametrů (sledovaný bod, počet stavů HMM) pomocí metody `Initialize` a trénování voláním metody `Learn`, konfiguraci gesta uložit do souboru.

Implementace gesta využívá jeden HMM, u kterého je ještě výsledný logaritmus pravděpodobnosti upraven délkou gesta dle následující rovnice:

$$P_{out} = P_{HMM} - \left|l_{observed} - l_{aprox}\right|^{1,75} \tag{4.1}$$

kde je hodnota pravděpodobnosti z HMM modelu P_{HMM} snížena o velikost rozdílu aktuálně zpracovávané sekvence $l_{observed}$ a její předpokládané délky l_{aprox}.Ten je ještě upraven exponenciální funkcí s exponentem o hodnotě 1,75 do podoby, která je znázorněna na obrázku 16. Hodnota exponentu, stejně tak jako samotný vzorec byly ověřeny experimentálně jako vyhovující pro většinu navržených gest.

Pro účely testování i budoucí práce bylo k tomuto generickému gestu navrženo grafické rozhraní pro vytváření, úpravy a trénování jednotlivých gest, které je popsáno v kapitole 4.2. Za jeho použití byla vytvořena sada testovacích gest uvedených v příloze A, na nichž byla ověřena funkčnost jak navržených algoritmů, tak i jejich implementace za pomoci testovací aplikace popsané v kapitole 4.3.

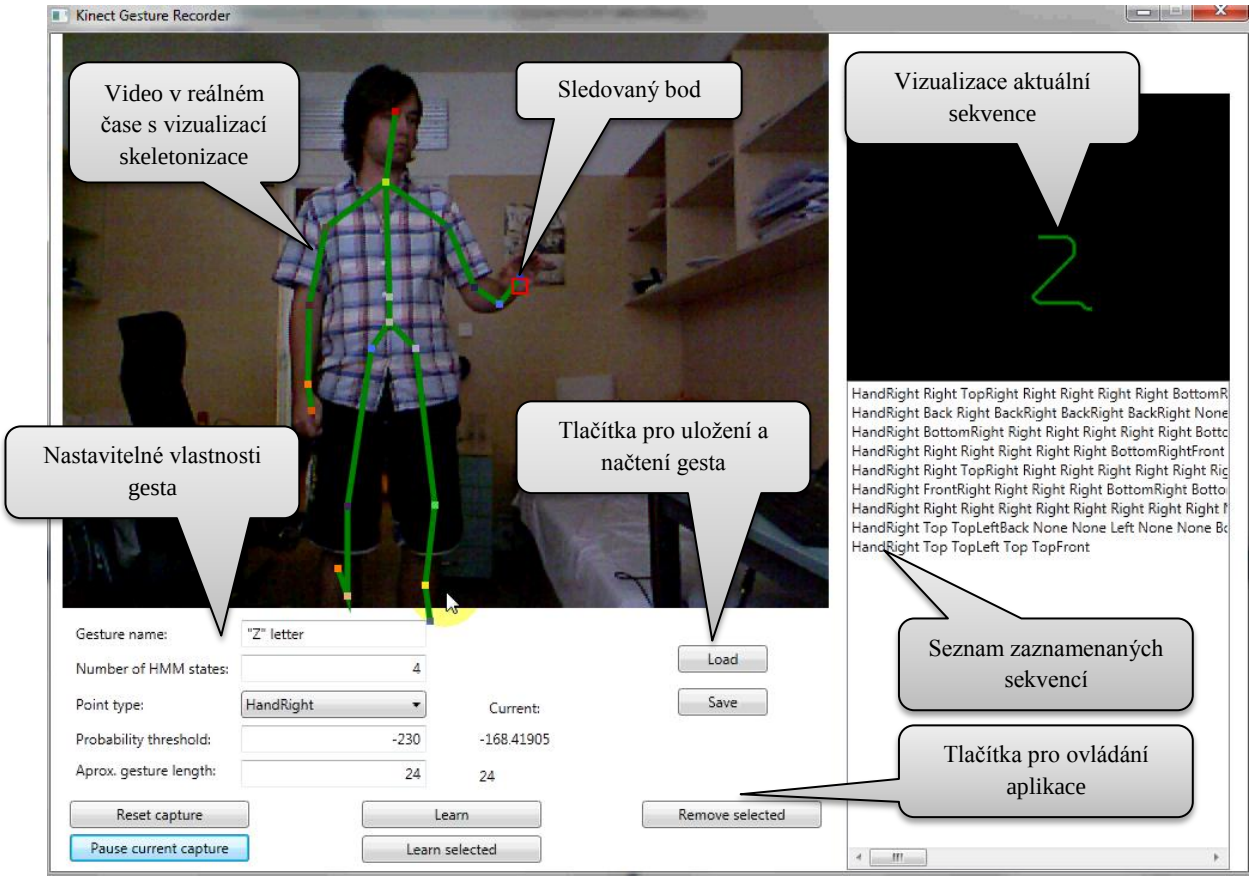


Obrázek 16 – Ukázka snížení hodnoty logaritmu pravděpodobnosti v závislosti na rozdílu délky aktuální sekvence gesta od jeho předpokládané délky

Ke všem částem knihovny včetně implementace tohoto generického gesta byla za pomoci dokumentačních XML tagů vytvořena programová dokumentace, která je součástí datového CD přiloženého k této práci.

4.2 Aplikace pro záznam a trénování gest

Pro účely záznamu a trénování generického gesta popsaného v kapitole 4.1.2 byla vytvořena aplikace, jejíž rozhraní je uvedeno na obrázku číslo 17. Umožňuje vytvoření gesta nového (po vyplnění základních údajů) nebo úpravy či případné dodatečné trénování gesta již existujícího.



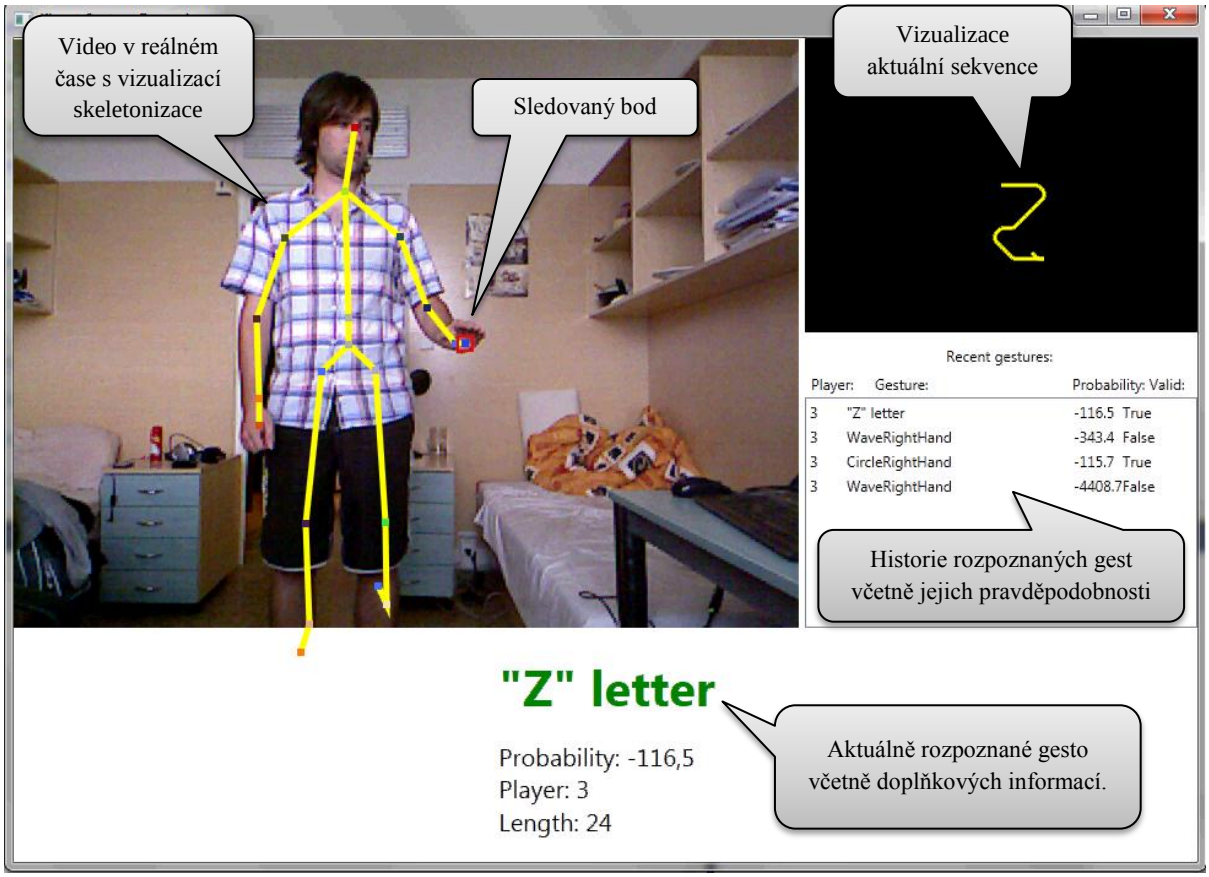
Obrázek 17 – Rozhraní aplikace pro záznam gest

Aplikace během své činnosti zobrazuje v levé horní části okna obraz z kamery umístěné na senzoru, na který je jsou namapována a následně vizualizována výstupní data skeletonizace získaná z NUI knihovny Kinectu. Červeným rámečkem je zobrazena filtrovaná pozice sledovaného bodu. Pod obrazem se nacházejí ovládací prvky pro nastavení vlastností zaznamenávaného gesta a tlačítka pro ovládání záznamu a provedení učení na základě zaznamenaných sekvencí. Ty jsou umístěny v seznamu v pravé části okna, kde jsou uloženy v textové podobě oddělené mezerami tak, že první je uveden název bodu, pro který sekvence vznikla, a následuje chronologický výčet směrů, které tuto sekvenci tvoří.

V pravé horní části se ještě nachází dvourozměrná vizualizace buď aktuálně prováděné, nebo v seznamu vybrané sekvence. Sekvence tak lze před samotným procesem učení ručně vyfiltrovat, a ponechat pouze ty relevantní. Po nakonfigurování a natrénování daného gesta je možné jej uložit do souboru, který lze potom přímo použít v dalších aplikacích využívajících tuto knihovnu.

4.3 Demonstrační aplikace

Aby bylo možno ověřit funkčnost knihovny, byla vytvořena ukázková aplikace, která demonstruje použití navržené a vytvořené knihovny v konkrétní aplikaci. Rozhraní této aplikace je zobrazeno na obrázku číslo 18.



Obrázek 18 – Demonstrační aplikace využívající vytvořenou knihovnu

Nejvíce plochy uživatelského rozhraní zabírá aktuální obraz z kamery senzoru doplněný o vizualizaci skeletonu a sledovaných bodů. Aktuálně zaznamenávaná sekvence (tedy prováděné gesto) je vizualizováno v pravé horní části okna.

Ve spodní části je pak část kde se, po svém vyhodnocení, zobrazí informace o právě provedeném gestu. Jeho název přitom může být buď zelený, kdy je gesto rozpoznáno s dostatečně velkou pravděpodobností (ta je určena dle prahové hodnoty každého gesta – viz kapitola 3.1.4) nebo červeně, kdy sice dané gesto bylo vyhodnoceno jako nejpravděpodobnější, ale ve skutečnosti se o dané gesto jednat nemusí (jeho pravděpodobnost je nižší než prahová). Dále jsou zde zobrazeny i další informace o aktuální evaluaci, a to výsledná pravděpodobnost daného gesta, jeho délka a v neposlední řadě i číslo hráče (tak jak jej indexuje NUI Kinectu), který gesto provedl.

V pravé části se pak ještě nachází historie jednotlivých rozpoznaných gest včetně jejich doplňujících informací.

5 Experimenty

Tématem této kapitoly je popis způsobu experimentování, prezentace výsledků a jejich následná analýza. Samotné experimenty probíhaly na základě gest uvedených v příloze A, kde jsou vypsány včetně jejich použité délky a prahu pravděpodobnosti (určuje rozdíl mezi úspěšným a falešným rozpoznáním).

Všechny případy uvedených gest používají, v kapitole 4.1.2 popsanou implementaci generického gesta založeného na HMM. Byla vytvořena a trénována za pomoci aplikace popsané v kapitole 4.2. Počet trénovacích sekvencí se u všech gest pohyboval mezi 25 až 40. U některých gest by jistě bylo vhodné v případě skutečného nasazení počet trénovacích sekvencí i zvýšit, čímž by se teoreticky docílilo vyšší přesnosti modelu.

Experimenty se zaměřily na použitelnost jednotlivých gest, úspěšnosti jejich rozpoznání a poměr rozpoznání správných vůči chybným. Byly rozděleny na dvě části, kdy nejdříve probíhaly s celou sadou devíti gest a následně se sadou omezenou, ze které byla odstraněna gesta způsobující největší problémy.

Ještě před samotnými experimenty je ale třeba si ujasnit pojmy související s klasifikací výsledků, které se v následujícím textu budou objevovat. Každý pokus o rozpoznání tedy může skončit jedním z následujících stavů:

- **Úspěšně rozpoznané** – případ, kdy je gesto rozpoznáno správně a výsledná hodnota pravděpodobnosti je vyšší než stanovený práh – rozpoznání je tedy validní a lze tento výsledek bez obav použít.
- **Úspěšně zamítnuté** – případ, kdy je gesto označeno chybně jako gesto jiné, ale není překročen práh pravděpodobnosti, gesto je tedy správně zamítnuto a výsledek by se neměl použít.
- **Falešně rozpoznané** – případ, kdy je sice rozpoznání označeno za validní, ale bylo chybně rozpoznáno gesto jiné, než předvedené. Toto je případ nejzávažnější chyby, která může nastat a je třeba jej minimalizovat.
- **Falešně zamítnuté** – gesto bylo sice rozpoznáno správně, ale výsledek byl označen jako nevalidní (nebyl překročen práh pravděpodobnosti). Vysoké procento těchto případů indikuje nevhodně zvolený práh.

5.1 Experimenty s kompletní sadou gest

Nejdříve byly experimenty provedeny na kompletní sadě všech devíti gest uvedených v příloze A. Gesta byla prováděna jednotlivě, vždy v bloku po přibližně padesáti sekvencích. Celý blok byl zaznamenáván na video a následně byly jednotlivé případy roztříděny dle výše uvedených kritérií, tedy na případy úspěšně rozpoznané, úspěšně zamítnuté, falešně rozpoznané a falešně zamítnuté. Při tom byly vyřazeny velmi nejednoznačné sekvence a nepovedená gesta, aby nezkreslovaly výsledek.

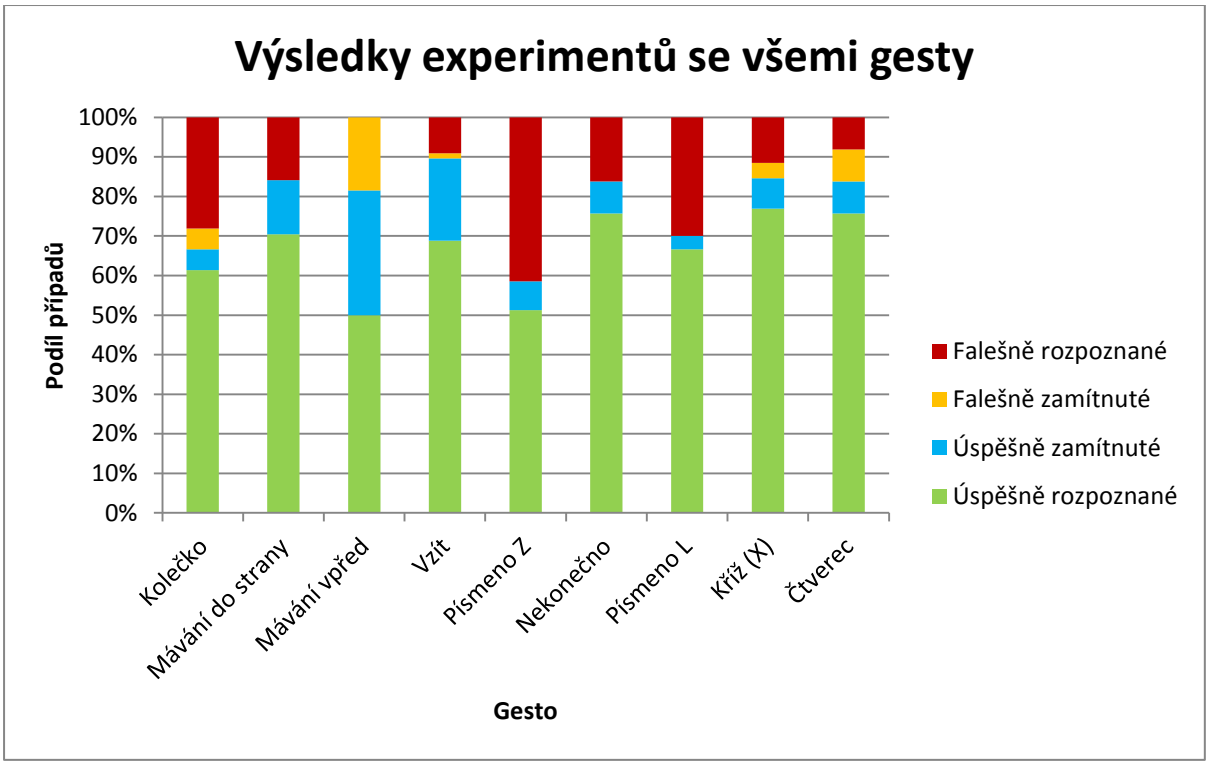
Takto získané konkrétní počty výskytů byly převedeny na procentuální vyjádření a zaneseny do tabulky číslo1. Na základě těchto dat byl následně vytvořen graf číslo 1. Pokud data zanalyzujeme, tak zjistíme, že nejvíce problémové jsou gesta *Kolečko* a *Písmeno „Z“*. U těchto dvou gest byla chybovost (falešné rozpoznání) na úrovni 28 respektive 41 procent. To je velmi vysoké číslo, které v praxi téměř znemožňuje jejich použití v reálné aplikaci. Navíc byla tato gesta sama o sobě častým zdrojem falešně rozpoznaných případů i u jiných gest.

Může to být způsobeno buď jejich velkou podobností s gesty jinými, nebo třeba i nedostatečným počtem nebo chybnými trénovacími daty. Svou roli také může hrát nevhodně zvolený

počet stavů HMM nebo již velmi vysoký počet gest spojených s jedním sledovaným bodem. Proto byla tato gesta na základě těchto výsledků odstraněna z dalšího experimentu.

Gesto	Úspěšně rozpoznané	Úspěšně zamítnuté	Falešně rozpoznané	Falešně zamítnuté
Kolečko	61%	5%	28%	5%
Mávání do strany	70%	14%	16%	0%
Mávání vpřed	50%	31%	0%	19%
Vzít	69%	21%	9%	1%
Písmeno Z	51%	7%	41%	0%
Nekonečno	76%	8%	16%	0%
Písmeno L	67%	3%	30%	0%
Kříž (X)	77%	8%	12%	4%
Čtverec	76%	8%	8%	8%

Tabulka 1 – Kategorizace a procentuální vyjádření četnosti jednotlivých případů rozpoznání kompletní sady gest.



Graf 1 – Grafické znázornění výsledků experimentu se všemi gesty.

5.2 Experiment s vybranými gesty

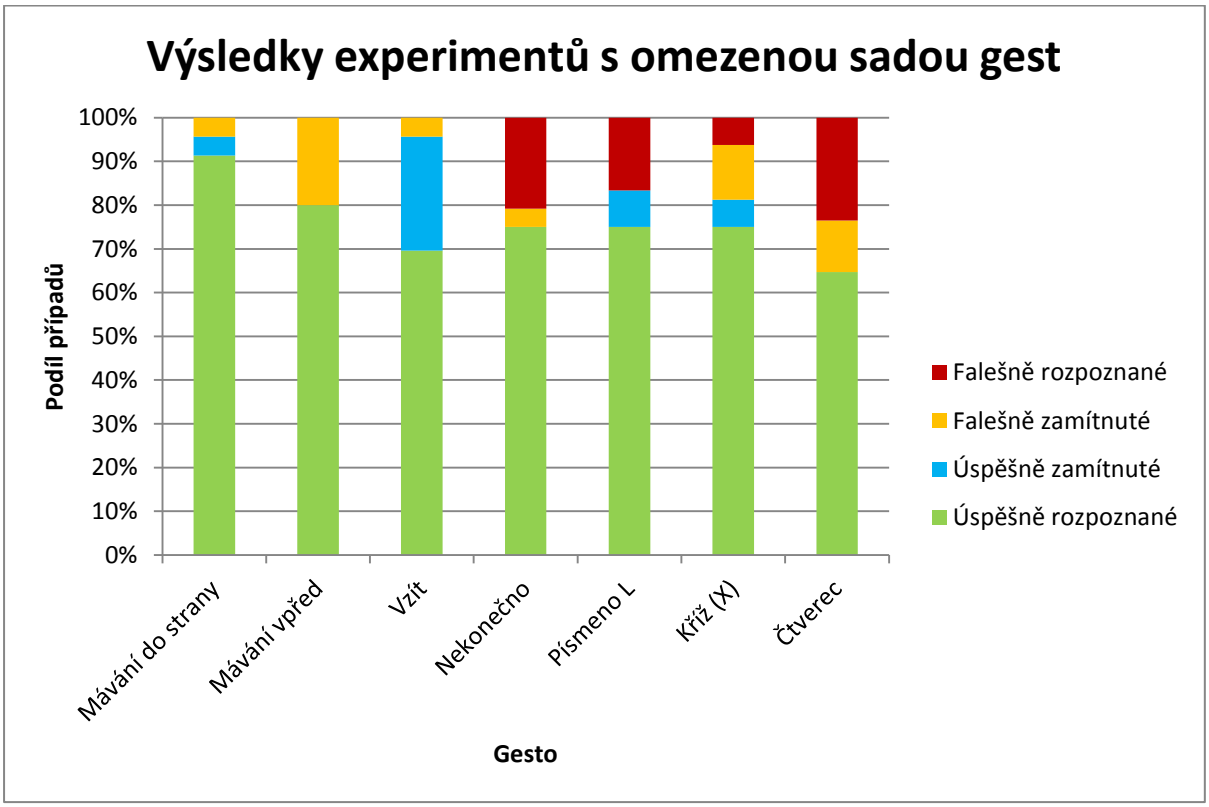
Další experiment měl za úkol zjistit chování celého systému po snížení počtu rozpoznávaných gest vyřazením gest problematických. Byla tedy vyřazena gesta *Kolečko* a *Písmeno „Z“*. Experiment proběhl stejným způsobem jako ten, uvedený v předchozí kapitole s jediným rozdílem, a tím byl nižší počet zaznamenaných sekvencí, konkrétně asi polovina.

Výsledky experimentu jsou zaneseny do tabulky číslo 2 a zobrazeny v grafu číslo 2. Jak lze zpozorovat, tak se podíl falešně rozpoznaných případů velmi snížil, čímž se u většiny gest dosáhlo celkové přesnosti vyšší než 80% (součet úspěšně rozpoznaných a úspěšně zamítnutých případů). Velký počet chybných výstupů zapříčinilo ještě gesto *mávání do strany*, kde jeho úpravou, případně odstraněním, by celková přesnost systému ještě vzrostla.

Je ovšem třeba počítat s tím, že počty testovacích sekvencí byly v tomto případě celkem nízké, takže výsledky nemusejí být zcela exaktní. Dalším problémem může být samotné experimentování s reálnými daty, kde nelze zajistit stoprocentní přesnost člověka provádějícího jednotlivá testovací gesta.

Gesto	Úspěšně rozpoznané	Úspěšně zamítnuté	Falešně rozpoznané	Falešně zamítnuté
Mávání do strany	91%	4%	0%	4%
Mávání vpřed	80%	0%	0%	20%
Vzít	70%	26%	0%	4%
Nekonečno	75%	0%	21%	4%
Písmeno L	75%	8%	17%	0%
Kříž (X)	75%	6%	6%	13%
Čtverec	65%	0%	24%	12%

Tabulka 2 – Kategorizace a procentuální vyjádření četnosti jednotlivých případů rozpoznání omezené sady gest.



Graf 2 – Grafické znázornění výsledků experimentu s omezenou sadou gest.

6 Závěr

Tématem této bakalářské práce byla detekce pohybových gest a následné vytvoření knihovny, využívající zde popsané řešení pro senzor Kinect. Byly však popsány i jiné než ve výsledku použité přístupy, především jiné způsoby sběru pohybových dat a rozdílné metody pro klasifikaci (vyhodnocování a rozpoznávání) gest.

Vznikla tak knihovna, která je lehce použitelná v i dalších aplikacích. Z možných přístupů se vždy využívá ten, který byl vyhodnocen jako nejvhodnější. To ale nevylučuji použití přístupu jiného. Týká se to především návrhu a realizace vyhodnocení pravděpodobnosti jednotlivých gest, kde ač jsou v této práci využívány Skryté Markovovy modely, lze k vyhodnocení použít jakýkoli jiný systém pracující se sekvencí diskrétních dat (v tomto případě směrů).

Pro testování navrženého přístupu a jeho implementace v podobě knihovny byla použita vytvořená testovací aplikace. Ta složila k detekci navržené, a za pomoci trénovací aplikace vytvořené, sady gest. Použití celá sady gest však nedosáhlo, snad z důvodů nízkého počtu trénovacích dat, nebo velké podobnosti rozpoznávaných gest, moc přesvědčivých výsledků. Podíl úspěšně rozpoznaných gest byl v průměru pouze 66%, podíl chybně vyhodnocených gest byl na 18%, zbylých 16% pak tvořili gesta zamítnutá (tedy nerozpoznaná). Po eliminaci dvou gest způsobujících největší problémy, se podíl úspěšných rozpoznání zvedl na 76% a podíl rozpoznání chybných klesl na 10%, při 15% gest nerozpoznaných. I tak to sice nejsou moc vysoké hodnoty, ale vzhledem k velmi vysoké chybovosti samotného snímání dat, jsou již pro reálné využití, při dobře navrhnuté sadě gest, dostatečné. Vyšší úrovně správných vyhodnocení by se jistě dosáhlo i použitím mnohem vyššího počtu trénovacích dat pro jednotlivá gesta.

Další rozšíření této práce by mohlo spočívat v implementaci jiných způsobů klasifikace gest, tedy použití například neuronových sítí či techniky dynamického borcení času (DTW). Vhodným tématem k další práci by bylo také rozšíření knihovny tak, aby byla schopna zpracovávat gesta sestávající z pohybu ne jednoho, ale i více sledovaných bodů současně (možné způsoby řešení byly v této práci okrajově popsány). Toto rozšíření by mohlo, mimo možnosti implementovat složitější gesta, i zvýšit přesnost rozpoznání zde popsaných jednoduchých gest. Bylo by tak možné v případě gesta prováděného rukou sledovat i pohyb okolních částí těla, tedy například pohyb zápěstí či loktu. Zajímavá by také byla možnost využít tuto knihovnu i s jinými senzory, než je právě Kinect.

Samotné využití knihovny je poměrně široké. Lze ji s úspěchem použít například k ovládání her či jiných multimediálních aplikací. Mezi ně mohou patřit například přehrávače videa a hudby, nebo ji lze využít jako součást prohlížeče obrázků. S jistými omezeními ji lze použít i pro ovládání webového prohlížeče, pokud se spokojíme s problematickým zadáváním složitějších dat, které lze řešit například za pomoci rozpoznání řeči. Své uplatnění knihovna jistě může nalézt i v dnes hojně rozšířených multimediálních prezentacích a na výstavách, kde nezvyklý způsob ovládání jistě přitáhne celou řadu účastníků.

Citovaná literatura

1. **Nintendo of America Inc.** Wii - Official Website at Nintendo. [Online] 2012. [Citace: 10. Březen 2012.] http://www.nintendo.com/wii.

2. **Sony Computer Entertainment America LLC.** PlayStation®Move Motion Controller - PlayStation®3 Move Info, Games & Updates. [Online] 2012. [Citace: 14. Březen 2012.] http://us.playstation.com/ps3/playstation-move/.

3. **Eurogamer Network Ltd.** Move sales top 8.8m. *Eurogamer.net.* [Online] 7. Červen 2011. [Citace: 14. Březen 2012.] http://www.eurogamer.net/articles/2011-06-07-move-sales-top-8-8m.

4. **Microsoft Corporation.** Microsoft Kinect for Windows | Develop for the Kinect. [Online] 2012. [Citace: 20. Únor 2012.] http://www.microsoft.com/en-us/kinectforwindows/.

5. **Microsoft Corporation.** Kinect Natural User Interface (NUI) Overview. *Kinect For Windows SDK Documentation.* [Online] Únor 2012. [Citace: 27. Únor 2012.] http://www.microsoft.com/en-us/kinectforwindows/develop/overview.aspx.

6. **Jamie Shotton, Andrew Fitzgibbon, Mat Cook, Toby Sharp, Mark Finocchio, Richard Moore, Alex Kipman, and Andrew Blake.** Real-Time Human Pose Recognition in Parts from a Single Depth Image. *Microsoft Research.* [Online] Červen 2011. [Citace: 3. Duben 2012.] http://research.microsoft.com/apps/pubs/default.aspx?id=145347.

7. **Ying Wu, Thomas S. Huang.** Vision-Based Gesture Recognition: A Review. [Online] 7. Říjen 2010. [Citace: 4. Duben 2012.] http://students.sabanciuniv.edu/~kamer/Magitact/References/Other%20Papers/Vision_Gesture_Recog.pdf.

8. **Meduna, Alexander a Lukáš, Roman.** *Formální jazyky a překladače - studijní opora.* Brno : FIT VUT v Brně, 2006.

9. **Nejedlý, Tomáš.** Recognition of ECG signals by algorithm based on Dynamic time wraping. [Online] 2008. [Citace: 16. Duben 2012.] http://www.feec.vutbr.cz/EEICT/2008/sbornik/01-Bakalarske%20projekty/02-Zpracovani%20signalu%20a%20obrazu/03-xnejed05.pdf.

10. **Černocký, Jan.** *Přednáška SRE 04 – Dekódování HMM - Úvod.* místo neznámé : FIT VUT v Brně, 2010.

11. **Mlích, Jozef.** Wiimote Gesture Recognition. *Proceedings of the 15th Conference and Competition STUDENT EEICT 2009 Volume 4.* 2009.

12. **Souza, César.** Hidden Markov Models in C#. [Online] 23. Březen 2010. [Citace: 25. Únor 2012.] http://crsouza.blogspot.com/2010/03/hidden-markov-models-in-c.html.

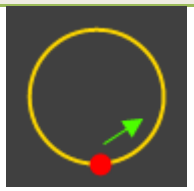
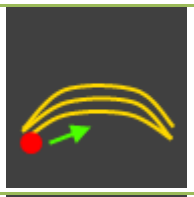
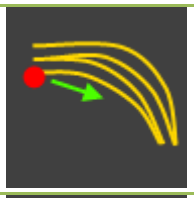
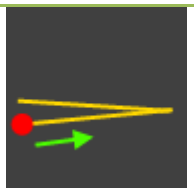
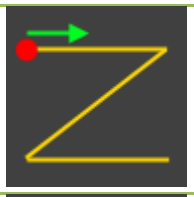
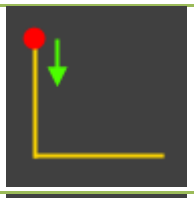
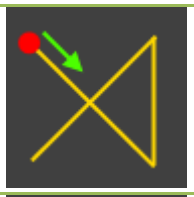
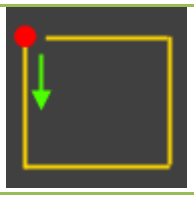
13. **ETH Zürich.** Accord.NET Framework. [Online] [Citace: 25. Únor 2012.] http://accord-net.origo.ethz.ch/.

14. **bbzippo.** Kinect in infrared. *Drawing Blanks.* [Online] 28. Listopad 2010. [Citace: 29. Březen 2012.] http://bbzippo.wordpress.com/2010/11/28/kinect-in-infrared/.

Seznam příloh

- Seznam navržených gest (Příloha A)
- DVD obsahující:
    - Zdrojové texty
    - Programovou dokumentaci
    - Soubory s použitými gesty
    - Aplikaci pro vytváření a trénování gest a demonstrační aplikaci v binární formě
    - Zdrojový text této technické zprávy
    - Videosekvence demonstrující funkci a použití vytvořené knihovny a ukázkových aplikací
    - Instalační soubory .Net Frameworku 4 a SDK Kinectu

Příloha A – Navržená gesta

Název	Ilustrace	Popis	Délka	Práh (log. pravděpodobnosti)
Kolečko		Kruh pravou rukou, který začíná ve své spodní části pohybem vpravo a následně nahoru	33	-250
Mávání do strany		Mávání pravou rukou, které začíná pohybem vpravo od těla	39	-300
Mávání vpřed		Mávání pravou rukou (zápěstím) vpřed – na obrázku pohled ze strany	44	-200
Gesto „vzít“		Natažení pravé ruky vpřed a následně vzad (jako když chceme něco vzít) – na obrázku pohled ze strany	28	-230
Písmeno „Z“		Pohyb pravou rukou doprava (od těla), šikmo dolů vlevo a následně opět vpravo (gesto kopíruje tvar písmene „Z“)	25	-200
Symbol nekonečno		Pohyb pravou rukou ve tvaru symbolu nekonečna s počátečním pohybem vpravo dolů	42	-250
Písmeno „L“		Pohyb pravé ruky směrem dolů a následně vpravo od těla ve tvaru písmene „L“	26	-160
Křížek (písmeno „X“)		Pohyb pravé ruky vpravo dolů, nahoru a nakonec vlevo dolů ve tvaru písmene „X“	43	-200
Čtverec		Pohyb pravé ruky dolů, vpravo, nahoru a nakonec vlevo ve tvaru čtverce	52	-280