VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

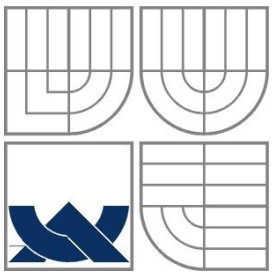
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMEDIÍ
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

KRESLENÍ S PROJEKTOREM POMOCÍ LEAP MOTION

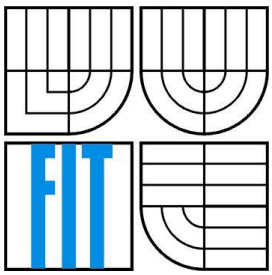
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE Peter Cilip
AUTHOR

BRNO 2015



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMEDIÍ
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

KRESLENÍ S PROJEKTOREM POMOCÍ LEAP MOTION

DATA PROJECTOR DRAWING WITH LEAP MOTION

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — Peter Cilip
AUTHOR

VEDOUCÍ PRÁCE — Ing. Jiří Zahrádka
SUPERVISOR

BRNO 2015

Abstrakt

Cílem této práce bylo navrhnout rozhraní využívajíci projektor a prostorový senzor leap motion pro aplikaci kreslení a kalibračního programu. Navrhované rozhraní implementovat v libovolném jazyce. Práce se zabýva kalibracií kamery a projektoru. Využití kalibrace v systému projektor – senzor, kde se kalibruje projektor podle dat ze senzoru. Výsledkem práce je multiplatformní program napsán v jazyce C++ s využitím knihovny Qt, který využíva senzor leap motion a projektor na kreslení.

Abstract

Purpose of this bachelor´s thesis is to design interface for application, that is using projector and sensor leap motion. Application is divided to 2 applications, calibration and drawing. Designed interface should be implemented in one of optional programming languages. Thesis is focused on calibration and calibration methods used in computer vision and drawing application. The result of work is multiplatform application written in C++ programming language and Qt library, that uses leap motion and projector to draw.

Klíčová slova

Leap motion , snímanie ruky, kreslenie , gestá, C++ , Qt framework, kalibrácia

Keywords

Leap motion, hand tracking, painting, gestures, C++ , Qt framework, calibration

Citace

Cilip Peter: Kreslení s projektorom pomoci Leap Motion, bakalářská práce, Brno, FIT VUT v Brně, 2015

Kreslení s projektorem pomoci Leap Motion

Prohlášení

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

……

Peter Cilip

13.5.2015

Poděkování

Týmto chcem poďakovať vedúcemu práce Ing. Jiřímu Zahrádkovy. Za jeho odbornú pomoc pri riešení problémov a za nasmerovanie na správne riešenia.

Obsah

1 Úvod 2
2 Počítačové videnie 3
2.1 História 4
2.2 Úloha počítačového videnia 4
2.3 Detekcia v počítačovom videní 6
2.4 Identifikácia v počítačovom videní 6
3 Kalibrácia 7
3.1 História 7
3.2 Kalibrácia v počítačovom videní 8
3.3 Metódy kalibrácie 8
3.4 Kalibrácia sústavy projektor – senzor 9
4 Leap Motion 10
4.1 Princíp fungovania Leap Motion 10
4.2 Leap Motion SDK 11
5 Prehľad senzorov 14
5.1 Kinect 14
5.2 PlayStation Eye/Camera 15
5.3 Wired Glove 16
5.4 Rozdiely v technológiách 16
6 Návrh 17
6.1 Svetelný šum 18
6.2 Kalibračný program 19
6.3 Program kreslenia 25
7 Implementácia 30
7.1 Kalibračný program 30
7.2 Program kreslenia 32
7.3 Podporované gestá 33
8 Testovanie 34
8.1 Návrh ďalšieho vývoja 35
9 Záver 36
Zoznam príloh 39
Príloha 1. Manuál – kalibračný program 40
Príloha 2. DVD 41
Príloha 3. Dotazník 42

1 Úvod

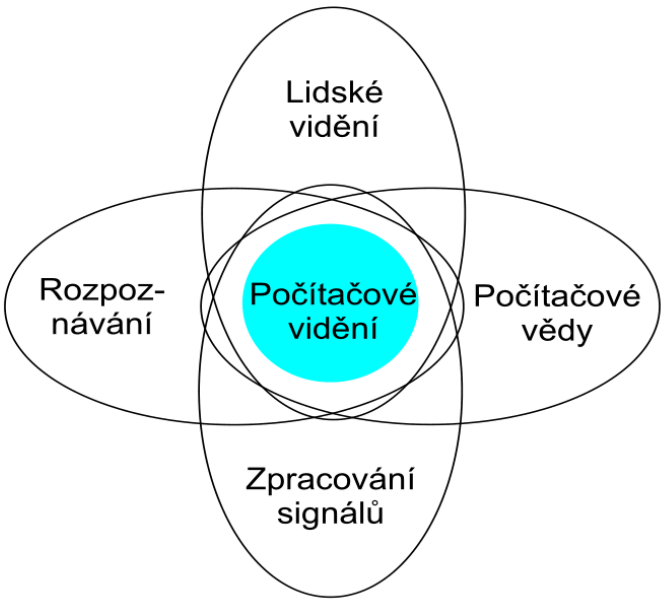
Bakalárska práca sa zameriava na technológie, ktoré slúžia na sledovanie pohybu objektov napríklad pohybu celého ľudského tela alebo pohyb ruky. Využitie počítačov v oblasti videnia bolo vždy zaujímavou témou a v nasledujúcich rokoch o túto oblasť bude stále väčší záujem. Problematike počítačového videnia sa venuje kapitola 2 a jej podkapitoly. Kalibrovanie zariadení je témou, ktorú je nutné pochopiť veľmi dobre, pretože bez dobrej kalibrácie by väčšina zariadení nefungovala správne. Problematike kalibrovania sa venuje kapitola 3, kde je priblížený pojem kalibrácia a jej história. Nutnosť kalibrovať zariadenia je v dnešnej dobe samozrejmosťou a existujú rôzne postupy kalibrácie zariadení. V tejto kapitole je uvedená kalibrácia kamery.

Zariadenie, ktoré je predmetom tejto bakalárskej práce je senzor leap motion. Jeho využitiu a princípu fungovania sa venuje kapitola 4, kde sa vysvetľuje ako funguje a s akým programovým rozhraním a aplikáciami sa dodáva. Okrem senzoru leap motion existuje ešte mnoho ďalších podobných technológií vyvíjaných rôznymi spoločnosťami ako Microsoft(Kinect) alebo Sony Computer Entertainment(PlayStation Eye), o týchto technológiách je možné sa dozvedieť v podkapitolách kapitoly 5. V samotnej kapitole 5 sa práca zameriava na predstavenie a porovnávanie rozdielov podobných technológií senzoru leap motion ich využitiu a výhodám.

Hlavným cieľom práce je spracovanie dát zo senzoru leap motion, kalibráciou systému projektor – leap motion a následne implementáciou rozhrania pre používanie tohto systému v podobe kalibračného programu a jednoduchého programu na kreslenie. Návrh rozhrania a jednotlivých celkov je možné nájsť v práci od kapitoly 6, implementačné detaily je možné nájsť v kapitole 7. V poslednej kapitole, sa práca venuje testovaniu navrhnutého rozhrania a bližšie popisuje odhalené chyby a ich možné riešenie. V prípade pokračovania vo vývoji projektu je vhodné pozrieť sa na podkapitolu 8.1, ktorá obsahuje možné vylepšenia.

2 Počítačové videnie

Počítačové videnie je odvetvie informačných technológií zaoberajúce sa metódami detekcie a rozpoznávania objektov z rôznych vstupných zariadení a následným vytváraním zariadení disponujúcimi týmito metódami. Táto disciplína vyrástla na základe vede o počítačoch, teórií signálov, rozpoznávania a porozumenia ľudského videnia. Využitie princípov počítačového videnia našlo uplatnenie v rôznych sférach ľudského života, ktorými sú napríklad medicínske zobrazovacie techniky, ovládanie procesov autonómnymi zariadeniami, organizácia informácií, modelovanie objektov alebo prostredia, detekcia javov, interakcia človeka s počítačom, virtuálna realita a mnoho ďalších. Všetky tieto odvetvia či už priamo alebo nepriamo súvisia s počítačovým videním, ktoré sa stalo ich neoddeliteľnou súčasťou [1].

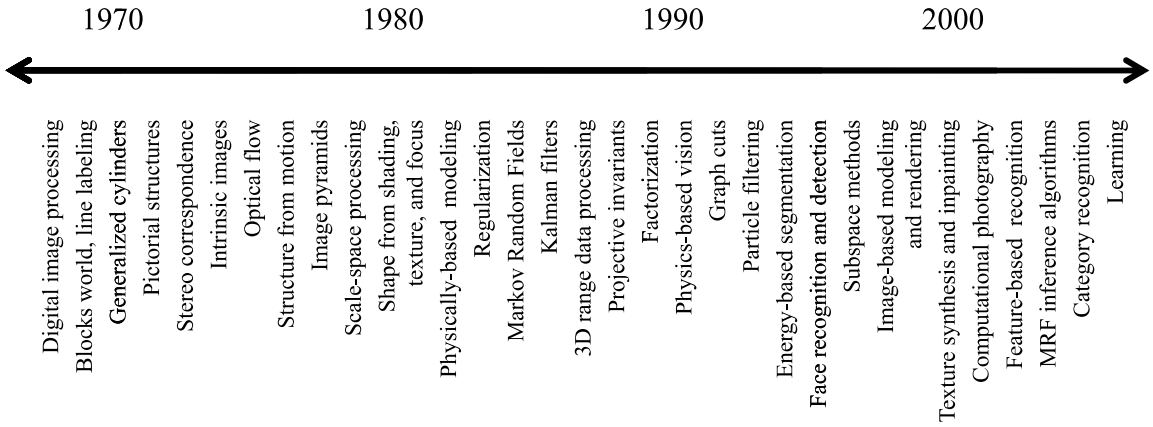


2.1 Piliere počítačového videnia

2.1 História

Vznik počítačového videnia sa datuje od 60. rokov minulého storočia. Všetko začalo digitálnym spracovaním obrazu. V tomto období sa vedci snažili vytvoriť zariadenie, ktoré by napodobilo ľudské videnie snímaním obrazu pomocou ostatných elektronických zariadení. V tomto období sa vytvorenie takéhoto stroja odhadovalo na jedno až dva desaťročia, no ani dnes ešte takýto inteligentný počítač zostrojiť nevieme. V minulosti sa počítače v oblasti počítačového videnia zaoberali dnes už primitívnym spracovaním obrazu a jeho digitálneho záznamu, detekciou hrán, rôznymi 2d transformáciami, odstránením šumu a pod. Dnes sa počítačové videnie zaoberá oveľa komplexnejšími a zložitejšími úlohami akými sú napríklad rekonštrukcia scény typicky z viacerých snímok scény s využitím sofistikovaných metód na produkciu vierohodných povrchov objektov v scéne. Ďalšou dôležitou úlohou je rozpoznávanie objektov a učenie počítačov za účelom vytvorenia sebestačnej umelej inteligencii schopnej sa rozhodovať [2].

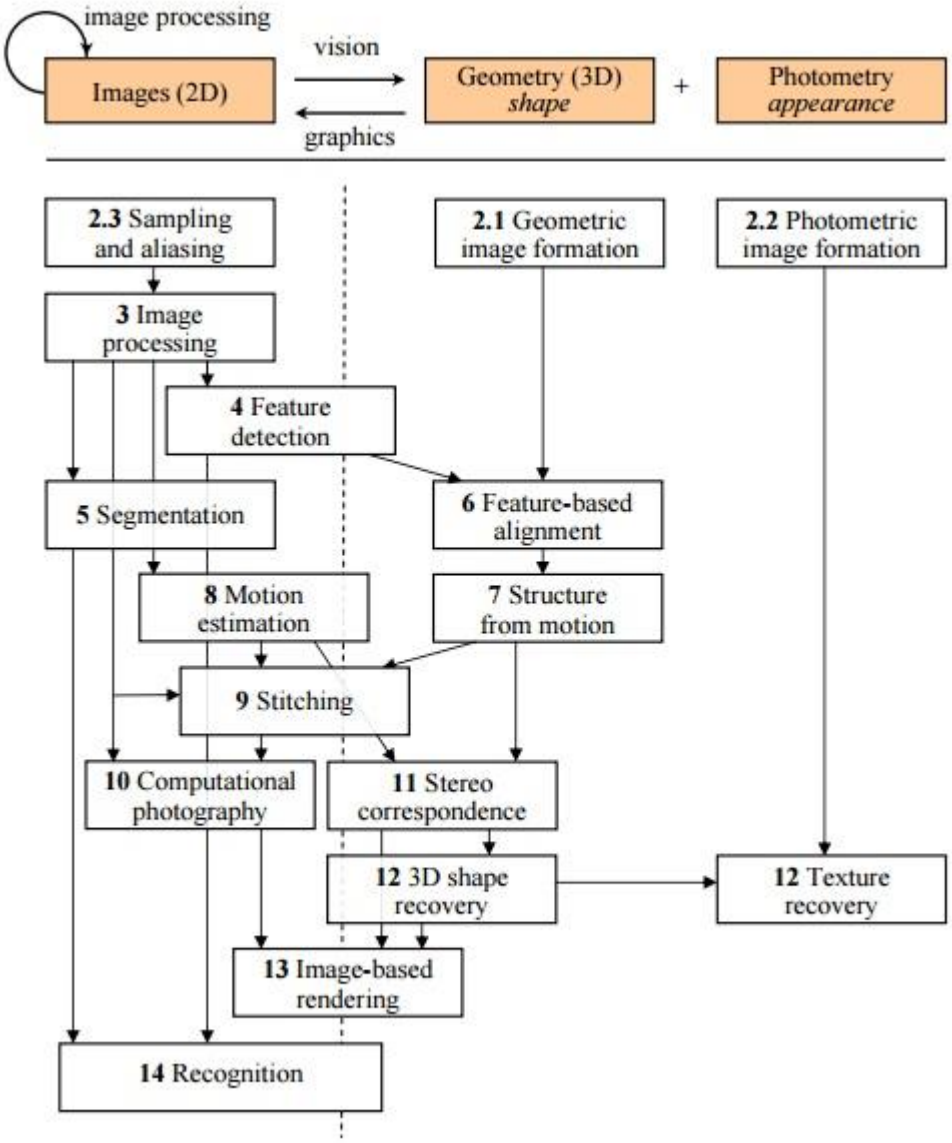
2.2 Úloha počítačového videnia



2.2 História počítačového videnia

V rôznych oboroch je úloha počítačového videnia zakaždým iná. V oblasti medicíny to je získavanie dát z medicínskych zariadení ako sú RTG, ultrazvukové zariadenie, tomograf. Následná analýza dát umožňuje určenie diagnózy pacienta. Úlohou môže byť aj získavanie informácií a následné modelovanie časti ľudského tela, tam kam sa skalpel nedostane. Jedným z najčastejších použití počítačového videnia je armádne použitie. Príklad môže byť detekcia nepriateľských vojakov, vozidiel alebo rakiet. Navádzanie rakiet na cieľ po vlete do bojovej zóny na základe analýzy bojiska, navádzanie autonómnych vozidiel, ponoriek, bezpilotných lietadiel. V aplikáciách umelej inteligencie sa využívajú princípy počítačového videnia na riešenie úloh vyžadujúcich autonómne operácie robotov v reálnom prostredí. Títo roboti využívajú na získavanie informácií z prostredia široké spektrum rôznych senzorov, preto je ďalším blízkym oborom počítačového videnia fyzika. Systémy počítačového videnia sú závislé na senzoroch. Tieto zariadenia detekujú rôzne typy žiarenia, spravidla sa jedná

o elektromagnetické žiarenie vo forme viditeľného alebo infračerveného svetla. Typicky senzory môžu byť zariadenia alebo prístroje ako kamera, tepelné čidlo, mikrofón, hĺbková kamera, radar, a mnoho ďalších rôznorodých a špecializovaných senzorov. Vo všetkých týchto smeroch sú základné úlohy počítačového videnia detekcia, identifikácia a poznanie objektov. V každom z oborov sa jednotlivé úlohy riešia iným spôsobom, a nahliada sa na nich individuálne. Spracovanie obrazu v počítačovom videní možno vidieť na obrázku 2.3, je na ňom znázornený postup spracovania obrazu od obyčajných metód aliasingu až po detekciu objektov, geometrické informácie o objektoch na obrázku a pod. Kvôli všetkým oborom v ktorých sa počítačové videnie používa je jeho funkcia a využitie tak rozmanité.



2.3 Spracovanie Obrazu

2.3 Detekcia v počítačovom videní

Detekcia je proces, kedy sú vstupné dáta(obraz, video nahrávka, zvuková nahrávka) prehľadávané a analyzované kvôli zisteniu konkrétnej podmienky. V medicíne sa môže jednať o analýzu obrazov za účelom detekcie buniek alebo tkanív. V bezpečnostných systémoch môže ísť o detekciu tváre v obraze, v priemysle o detekciu vozidla pri vyberaní mýta alebo pri meraní rýchlosti vozidla radarom.

2.4 Identifikácia v počítačovom videní

Detekovaný objekt je len ohraničenie nejakého zhluku informácií, pre zistenie čo tento zhluk predstavuje je nutné preveriť objekt procesom identifikácie. Identifikácia je proces, kedy sa objekt po detekcii v obraze spracováva ďalej, a vyhodnocuje sa, či daný objekt spĺňa podmienky nejakého známeho objektu uloženého v databáze alebo spĺňa podmienky nejakého vzorca alebo šablóny.

3 Kalibrácia

Kalibrácia je proces porovnávania medzi hodnotami veličín, jeden rozsah hodnôt je nastavený na jednom zariadení a porovnáva sa s hodnotami veličiny na druhom zariadení. Zariadenie so známym, alebo správnym rozsahom hodnôt sa nazýva štandardné. A druhé zariadenie, ktorého jednotky sa testujú, kalibrujú sa nazýva testované zariadenie, alebo má nejaký názov z mien pre kalibrované zariadenia.

3.1 Nekalibrovaná váha. Nezaťažená neukazuje 0.

3.1 História

Prvý výskyt slova vstúpil do anglického jazyka v období americkej občianskej vojny v popise delostrelectva. Avšak niektoré skoršie známe systémy merania, ktoré boli vytvorené starovekými civilizáciami Egyptu a Mezopotámie. Kalibrácia sa odpradávna používa na prepočet jednotiek jedného systému na jednotky iného systému. Napríklad prepočet kilogramu na libry.

3.2 Kalibrácia v počítačovom videní

Proces, ktorý za špecifických podmienok stanovuje vzťah medzi hodnotami veličín jedného a druhého systému. V rôznych prípadoch môže ísť o rozdielne veličiny. Pri kalibrácií systémov pracujúcich so zvukom pôjde o iné veličiny ako napríklad pri systémoch pracujúcich s videom. Každý systém v počítačovom videní môže byť kalibrovaný iným spôsobom a inou metódou, tejto problematike sa venuje nasledujúca podkapitola. Kalibrovaný systém je nutné znovu kalibrovať ak boli zmenené niektoré parametre sústavy, zmenila sa vzájomná poloha zariadení v kalibrovanej sústave alebo sa zmenili optické vlastnosti jedného zo zariadení a podobne.

3.3 Metódy kalibrácie

Základom každej kalibračnej metódy je niekoľko krokov, ktoré by sa pre úspešné kalibrovanie mali uskutočniť. Ako prvé je nutné mať kalibračný vzor, ako kalibračný vzor môže poslúžiť čierno-biela šachovnica, alebo vzor vytvorený z náhodných bodov. V minulosti sa používal na kalibráciu televízií takzvaný testovací vzor, typicky sa vysielal, keď televízia nevysielala žiadny program.

3.2 Testovací vzor televízií pre štandard NTSC(National Television System Committee)

Kalibračný vzor sa premieta pomocou zobrazovacieho zariadenia a následne sa zachytávajú body kalibračného vzoru a ukladajú na spracovanie. Ďalším krokom kalibrácie je spracovanie nameraných hodnôt a to takým spôsobom, aby sme mohli jednoznačne určiť korešpondujúce body z kalibračného vzoru a zachytených bodov. Výstupom kalibrácie by mali byť dáta, ktoré popisujú korešpondenciu súradnicového systému kalibračného vzoru a súradnicového systému zachytených bodov. Cieľom kalibrácie je umožniť transformáciu ľubovoľného bodu v súradnom systéme zachytených bodov na súradnice korešpondujúceho bodu v súradnicovom systéme kalibračného vzoru. V konkrétnom prípade môže ísť o prevod nameraného bodu reálneho sveta v milimetroch do súradnicového systému

obrazovky v pixeloch. V prípade určitých zobrazovacích zariadení môže ísť o posun, rotáciu alebo zväčšovanie obrazu a podobne.

3.4 Kalibrácia sústavy projektor – senzor

Najčastejším použitím v oblasti počítačového videnia je využitie sústavy pozostávajúcej zo zobrazovacieho zariadenia, ktorým je najčastejšie monitor alebo projektor a zariadenia zachytávajúceho reálny svet a tým môže byť kamera, dvojica kamier alebo iný senzor. Systém je nutné kalibrovať, aby program pracujúci s týmto systémom vedel správne pracovať. Cieľom kalibrácie takéhoto systému je zistiť parametre kamery, stred obrazu, koeficienty skreslenia, transformáciu bodov (zo súradnicového systému senzoru do súradnicového systému zobrazovacieho zariadenia) a iné parametre a uložiť ich pre iný program, ktorý ich využije. Výpočet parametrov sa môže uskutočňovať prostredníctvom rôznych metód, podľa toho aký parameter potrebujeme vypočítať. Klasická kalibrácia kamery spočíva v premietaní vzoru(sú známe jeho reálne fyzikálne vlastnosti) zobrazovacím zariadením a jeho analýzu vstupným zariadením za účelom získania parametrov kamery pre proces transformácie bodov [3] [4]. Keďže šošovka kamery nie je dokonalá je nutné poznať koeficienty skreslenia kamery. Na základe týchto koeficientov upraviť snímku, tak aby vyzerala, keby bola šošovka dokonalá. Existuje veľké množstvo metód, väčšinou slúžia pre konkrétne účely a použitia. Väčšina metód je založená na poznaní referenčných súradníc kalibračného vzoru. Takáto kalibrácia je veľmi efektívna, ale pri týchto metódach je nutné vypracovať postup kalibrácie kamery. Základom je odhadnúť premietaciu maticu kamera, čo sa obyčajne realizuje meraním priestorových súradníc a korešpondujúcich bodov zo scény. V konkrétnom prípade kde je použitý systém leap motion, nie je nutné hľadať koeficienty skreslenia jeho kamier, pretože výstupom senzoru nie sú snímky ako z kamery, ale už vypočítané súradnice nasnímaných objektov pomocou komplexných algoritmov. Tento výpočet autonómne rieši ovládač pre senzor leap motion. V spomínanom systéme senzoru a projektoru je nutné riešiť korešpondenciu bodov súradnicového systému projektoru a senzoru. Tento problém sa vyrieši, ak je známa vzájomná poloha týchto dvoch zariadení. Ak sú známe spomínané hodnoty a je možné vypočítať transformáciu bodov z jedného systému do druhého, môžeme prehlásiť systém za nakalibrovaný.

4 Leap Motion

Leap motion je malé zariadenie na snímanie pohybu rúk a všetkých desiatich prstov. Zariadenie na sledovanie pohybu používa dve monochromatické infračervené kamery a tri infračervené diódy. Jeho oblasť snímania je malá. Jeho hlavnou prednosťou je vysoké rozlíšenie snímaného objektu a v kombinácií s vysokou rýchlosťou snímania a infračervenými kamerami je možné zachytiť pohyby rukou za akýchkoľvek svetelných podmienok s vysokou presnosťou. Práve preto sa hodí na aplikácie, kde je nutná vyššia presnosť, ako napríklad kreslenie. Okrem spomínanej funkcie sa využíva namiesto myši, kde nie je nutný žiadny kontakt alebo dotýkanie sa rúk.

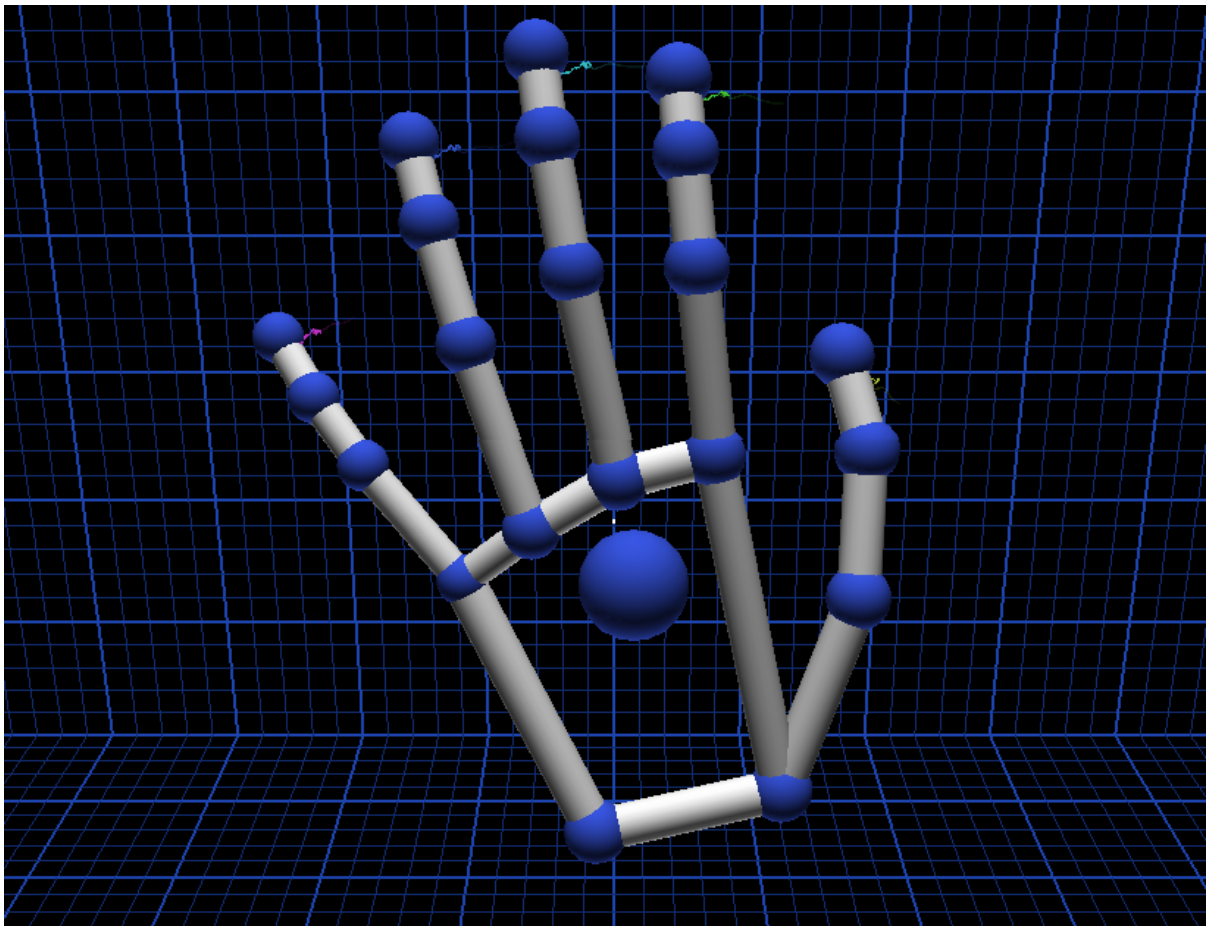
4.1 Leap Motion

4.1 Princíp fungovania Leap Motion

Leap motion je malé USB periférne zariadenie, ktoré bolo navrhnuté na snímanie pohybu rúk. Ako už bolo spomenuté v predchádzajúcej kapitole, tento senzor má tri infračervené diódy a využíva ich na generovanie vzoru bodiek infračerveného svetla, ktorého odrazy snímajú kamery senzoru rýchlosťou skoro 300 snímok za sekundu. Následne sú dáta poslané cez USB kábel do hosťovského počítača. V počítači sa snímky analyzujú, zisťuje sa 3d pozícia nasnímaných objektov z 2d snímok generovaných kamerami pomocou komplexných algoritmov.

4.2 Leap Motion SDK

Okrem hardvéru a softvéru v podobe hier alebo debugovacích aplikácií, je možné si z oficiálnych stránok[1] stiahnuť SDK(software development kit), čiže súbor nástrojov pre vývoj softvéru. Je to programovacie rozhranie dostupné pre operačný systém Windows, Linux a OSX. Na oficiálnych stránkach[2] je možné nájsť dokumentáciu k rôznym programovacím jazykom. Tento vývojový kit obsahuje množstvo tried, ktoré poskytujú údaje o snímaných objektoch a scéne.



4.2 Leap Motion vizualizér - Ruka

Trieda Controller

Táto trieda tvorí hlavné rozhranie k zariadeniu. Vytvorením inštancie tejto triedy, má programátor prístup k snímkam a snímaným údajom, tiež môže meniť konfiguráciu. Trieda obsahuje všetky údaje o načítanej scéne, uchováva si aj údaje o posledných 60 snímkach. Tejto triede je možné pridať triedu `Listener`, ktorej bude zasielať správy aby spracovala udalosti. Ak nejaká udalosť nastane, objekt

[1] https://developer.leapmotion.com/
[2] https://developer.leapmotion.com/documentation

triedy `Controller` invokuje adekvátnu metódu definovanú v triede `Listener`. Objekt je multivláknový a volá metódy triedy `Listener` v samostatnom vlákne.

Trieda `Listener`

Táto trieda definuje súbor takzvaných „callback" funkcií, ktoré budú invokované ak nastane nejaká udalosť a objekt triedy `Controller` bude o týchto udalostiach informovať. Táto trieda poskytuje množstvo funkcií, všetky sú popísané na webovej stránke[3]. Tie hlavné z nich sú tieto metódy:

- onConnect
- onDisconnect
- onFrame

Funkciu `onConnect()` sa zavolá, ak sa fyzicky pripojí Leap Motion, `onDisconnect()` zase ak sa odpojí. Hlavnou funkciou je `onFrame()`, ktorá sa volá ak ma zariadenie k dispozícii novú snímku. Následne programátor v tejto metóde môže pristúpiť k snímke a analyzovať všetky objekty nachádzajúcej sa nasnímanej scéne od nasnímanej ruky, prstov až po detekované nástroje. Ako už bolo spomenuté je možné pristupovať aj k starším snímkam.

Reprezentácia objektov v snímke Leap Motionu

Na reprezentáciu jednotlivých objektov v scéne poskytuje vývojový kit paletu tried. Jednou z hlavných tried je trieda `HandList`, ktorá poskytuje zoznam nasnímaných rúk, ich počet a funkcie na výber ruky na spracovanie. Ruku z tohto zoznamu je možné si vybrať indexom pomocou operátora `[]`, alebo funkciami `frontmost()`, `leftmost()` a `rightmost()`. Tieto funkcie vrátia ruku, ktorá je buď najviac vpredu, vľavo alebo vpravo. Ďalšou triedou reprezentujúcou ruku v snímke je `Hand`, táto trieda zapuzdruje informácie o fyzickej charakteristike detekovanej ruky. Trieda `Hand` obsahuje všetky dostupné údaje o nasnímanej ruke ako id ruky, paža ktorej patrí, informáciu či daná ruka je v scéne ešte platná, či je ľavá alebo pravá, všetky informácie o prstoch, pozíciu v súradnicovom systéme senzoru, vektor smeru ruky a podobne. Trieda `Finger` reprezentuje nasnímaný prst, táto trieda je podtriedou `Pointable` a obsahuje všetky potrebné informácie na analýzu prstov. Prostredníctvom tejto triedy je možné zistiť pozíciu prstu, či je ohnutý alebo rovný, smer prstu atď. Všetky triedy a ich popis je možné nájsť na webových stránkach spoločnosti[4].

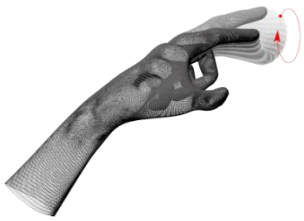
Gestá

V programovom rozhraní sú všetky rozpoznané gestá v scéne reprezentované triedou `GestureList`, ktorá je vlastne zoznamom pre nasnímané gestá. Triedou reprezentujúcu samostatné gesto je trieda

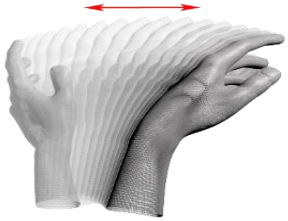
[3] https://developer.leapmotion.com/documentation/cpp/api/Leap.Listener.html
[4] https://developer.leapmotion.com/documentation/cpp/api/Leap_Classes.html

`Gesture`. Od tejto triedy sú odvodené jej podtriedy pre každý typ gesta jedna. Rozhranie dokáže rozpoznať štyri základné gestá a tými sú kruhové gesto, posunutie ruky, ťuknutie prstom dopredu a ťuknutie prstu smerom dole. Kruhové gesto a posunutie ruky sú gestá, ktoré trvajú istý čas a teda majú stavy štart, stop a update. Gestá musia byť povolené v triede `Controller` pomocou metódy `enableGesture()` a ako parameter metóde sa predáva typ gesta. Tieto gestá patria medzi základné, ale nie je problém naprogramovať si vlastné komplexnejšie.

4.3 Gesto ťuknutia prstom



4.4 Kruhové gesto



4.5 Gesto posunutia ruky

5 Prehľad senzorov

Okrem senzoru Leap Motion sa pohyb rúk dá zaznamenať aj iným spôsobom. Najjednoduchším zariadením na snímanie pohybu je obyčajná kamera. Ale pre presnú detekciu je nutné kameru kvalitne skalibrovať a využiť algoritmy na detekciu objektov v obraze a rozpoznávanie gest. Pre takúto detekciu je možné využiť systém viacerých kamier, kde je ale nutné využiť komplexnejšie algoritmy. Nevýhodou kamery je nízky počet snímok za sekundu a to, že nedokáže väčšinou zachytiť objekty za zhoršených svetelných podmienok. Okrem kamery existujú ešte ďalšie technológie zameriavajúce sa sledovaním pohybu objektov a tými napríklad sú:

- Kinect
- Playstation Eye/Camera
- Wired Glove

5.1 Kinect

Kinect je známy kvôli hernej konzole Xbox, v ktorej sa používa. Kinect pozostáva z farebnej kamery, hĺbkového senzoru a skupiny mikrofónov, čo poskytuje snímanie pohybu celého ľudského tela, rozoznávanie tváre alebo rozoznávanie hlasu. Jeho hĺbkový senzor pozostáva z infračerveného

5.1 Kinect pre Xbox 360

laserového projektoru a monochromatického senzoru, čo poskytuje zachytávanie video dát v 3d, za každých svetelných podmienok. Kinect je schopný zachytiť predmety vzdialené aj viac ako 3.5 m, kde ale dochádza k menším skresleniam. Od roku 2014 je Kinect dostupný vo verzií 2, táto verzia je

založená na rovnakom jadre ako Kinect pre Xbox One, ale obsahuje nový senzor. Taktiež jeho kamery nedisponujú vysokým rozlíšením, preto sa pre aplikáciu umožňujúcu kreslenie nehodí.

5.2 Kinect verzie 2

5.2 PlayStation Eye/Camera

PlayStation Eye/Camera je periféria pre hernú konzolu PlayStation 3 a podobá sa webovej kamere. Na spracovanie snímok zariadenie používa počítačové videnie, rozpoznávanie gestikulácie a zabudované pole mikrofónov, to umožňuje hráčom konzoly interagovať s hrami aj pomocou pohybov. PlayStation Eye je nástupca technológie zvanej EyeToy vyvinutej pre PlayStation 2. Oproti Leap Motion a Kinectu, PlayStation Eye nedokáže zachytiť pohyb objektu za všetkých svetelných podmienok.

5.3 PlayStation Eye

5.3 Wired Glove

Wired Glove je vstupné zariadenie pre interakciu človeka s počítačom pomocou rukavice. Toto zariadenie využíva niekoľko senzorov na zachytenie dát ako „ohnutie“ prstov, vzájomná poloha prstov a podobne. Pri využití tejto technológie je nutné použiť aj zariadenie snímajúce polohu alebo rotáciu

5.4 Wired glove

rukavice v globálnom merítku napr. oproti počítaču alebo stolu. Wired Glove sa používa pre aplikácie manipulujúce s virtuálnou realitou. Pri kombinácií s technológiou Oculus rift[5] je možné takýto set používať napríklad aj na hranie hier.

5.4 Rozdiely v technológiách

Nasledujúca tabuľka popisuje rozdiely v spomínaných technológiách.

Parametre	**Leap Motion**	**Kinect**	**Playstation Eye/Camera**	**Wired Glove**
Rozlíšenie kamier	640x480@300	640x480@30	640×480@60 320x240@120	
Plocha interakcie	1 m^2	6 m^2		
Mikrofón	Nie	Áno	Áno	Nie
Presnosť	Vysoká	Stredná	Stredná	Vysoká
Využitie	Rôzne	Rôzne	Hry	Virtuálna realita

[5] https://www.oculus.com/

6 Návrh

Cieľom tejto bakalárskej práce je navrhnúť priestorové rozhranie, ktoré bude využívať projektor a senzor Leap Motion. Rozhranie tvorené dvomi zariadeniami je nutné nakalibrovať, aby aplikácia pracujúca s týmto rozhraním dokázala správne reagovať a zobrazovať objekty na zobrazovacej ploche projektoru. Na kalibráciu takejto sústavy je možno využiť knižnicu OpenCV [5], ktorá poskytuje infraštruktúru pre aplikácie počítačového videnia, poskytuje viac ako 2500 optimalizovaných algoritmov na detekovanie a rozpoznávanie. Pri využití kalibrácie z OpenCV, kalibráciou zistíme všetky parametre kamery senzoru a údaje na prevod bodu jedného súradnicového systému na druhý a to transformačnú maticu, vektor/y posunutia, rotácie atď [6]. Výpočet týchto parametrov trvá istý čas a následne je nutné všetky kalibračné údaje uložiť. V takomto rozhraní nie je nutné riešiť zložitú kalibráciu a zisťovať všetky parametre sústavy. A teda systém zložený z projektoru a senzoru leap motion s cieľom vytvorenia aplikácie kreslenia nemusí poznať všetky parametre sústavy, čiže v zjednodušenej verzii je dôležité zistiť tieto veci:

- vzájomnú polohu senzoru a projektoru
- pomer milimetrov na pixely
- poloha zobrazovacej plochy

6.1 Navrhovaný systém – Projektor hore na statíve, leap motion otočený smerom k stolu na drevenom stojane

Navrhnutá aplikácia je tvorená dvoma celkami, kalibračným programom a aplikáciou kreslenia. Kalibračný program získa určité body v súradnicovom systéme leap motion-u a vypočíta vzájomnú polohu senzoru a projektoru. Následne výsledok uloží do xml súboru. Aplikácia kreslenia spomínaný súbor načíta a používa jeho obsah na správne počítanie určitých hodnôt. Bez kalibračného súboru sa aplikácia ani nenačíta. V aplikácií sú využité klasické nástroje pri kreslení s využitím tretieho rozmeru, ktorý poskytuje práve 3d senzor. Pre manipuláciu s aplikáciou sú definované určité gestá, ktoré budú neskôr spomenuté. Rozhranie bude pozostávať zo senzoru umiestneného na držiaku otočeného kamerami smerom k zobrazovacej ploche, v tomto prípade na stôl. Ďalšou zložkou rozhrania je projektor umiestnený na statíve alebo na stojane otočený rovnakým smerom ako leap, čiže na stôl. Kvôli nevyhovujúcim svetelným podmienkam je na stole umiestnená čierna látka. Tento popisovaný systém je možno vidieť na obrázku 6.1.

Pre samotné využívanie aplikácie je teda nutné vytvoriť si niekoľko pomôcok, ktoré sú nutné pre kalibrovanie a beh samotnej aplikácie. Tiež je nutné aplikáciu v počítači spúšťať na pripojenom projektore, všetky problémy týkajúce sa problémov sú popísané v nasledujúcich kapitolách a podkapitolách. Navrhnuté programy boli implementované v jazyku C++ s využitím Qt frameworku [7] pre tvorbu užívateľského rozhrania. Bližšie detaily implementácie jednotlivých programov sú spomenuté v kapitolách, alebo podkapitolách. Pri znázorňovaní popisovaných problémov, alebo vysvetľovaní určitých situácií je použitý matematický softvér GeoGebra[6].

6.1 Svetelný šum

Senzor leap motion funguje na princípe vyžarovania infračerveného svetla a snímania jeho odrazu, preto je nutné aby pomôcky použité pri kalibrácií alebo pri behu aplikácie neodrážali infračervené svetlo. Kvôli odrážanému svetlu je nutné na zobrazovaciu plochu umiestniť materiál, ktorý neodráža infračervené svetlo tak intenzívne. Jedným z takýchto materiálov môže byť aj čierna matná bavlna. Bez tejto úpravy je senzor otočený smerom k fyzickej ploche nepoužiteľný lebo nerozozná objekt pred kamerami z dôvodu veľkého množstva odrazeného svetla.

[6] http://www.geogebra.org/

6.2 Kalibračný program

Ako bolo spomenuté v kapitole 3, je to proces, ktorý pozostáva z niekoľkých krokov. Na úspešné kalibrovanie sústavy je nutné si vytvoriť niekoľko pomôcok.

- Držiak na senzor Leap Motion
- Držiak plátna
- Statív/držiak na projektor

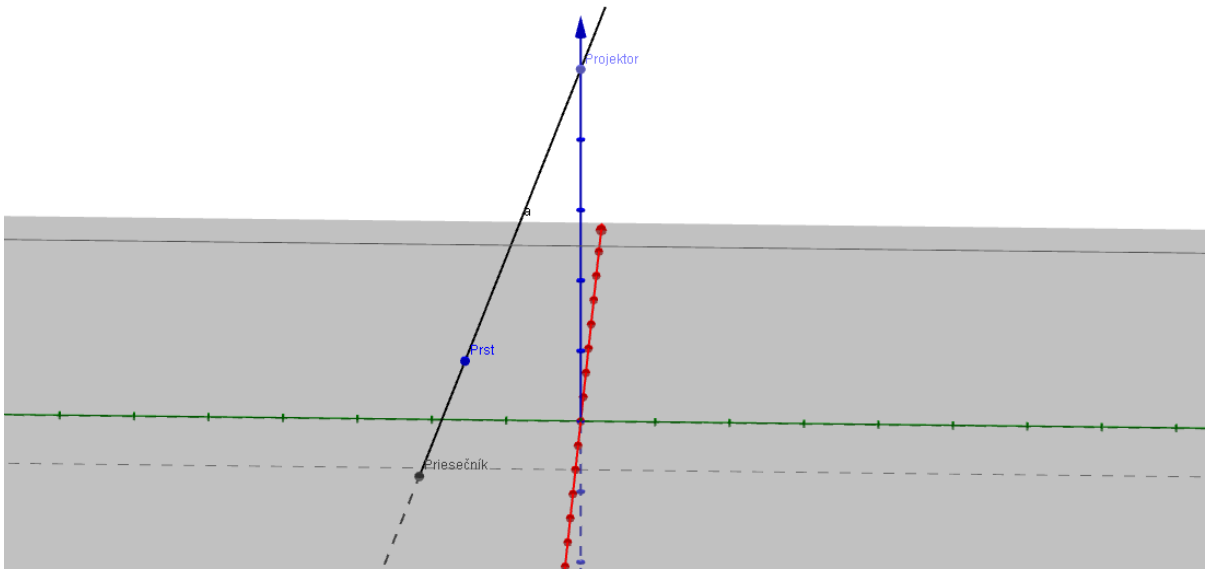
6.2 Držiak na senzor leap motion

6.3 Držiak na plátno

6.4 Projektor na statíve

Spomenuté pomôcky je možné si jednoducho vyrobiť. Ako držiak na projektor som použil klasický statív na kamery alebo projektory. Držiak na senzor Leap Motion som vyrobil z dreva, ale je možné tiež použiť stavebnicu Merkúr alebo podobné nástroje. Držiak plátna je v tomto prípade tiež vyrobený z dreva, ale nie je nutné ho vyrábať z dreva. Úplne postačí podložiť plátno vhodnou krabicou.

Navrhnutý kalibračný program, postupne zvýrazňuje na zobrazovacej ploche štyri body, ktoré sú umiestnené v rohoch plochy, predpokladá sa, že osoba kalibrujúca systém bude na zobrazované body ukazovať prstom a program si bude tieto hodnoty ukladať. Po načítaní prvých štyroch bodov na zobrazovacej ploche osoba kalibrujúca systém vsunie držiak s plátnom pod projektor a znova bude ukazovať na zvýrazňované rohové body na plátne. Po načítaní bodov sa vypočíta poloha projektora a uložia sa tri body zobrazovacej plochy, tri body v súradnicovom systéme projektoru a poloha projektoru. Po uložení sa kalibračný program ukončí. Využitá kalibračná metóda je popísaná



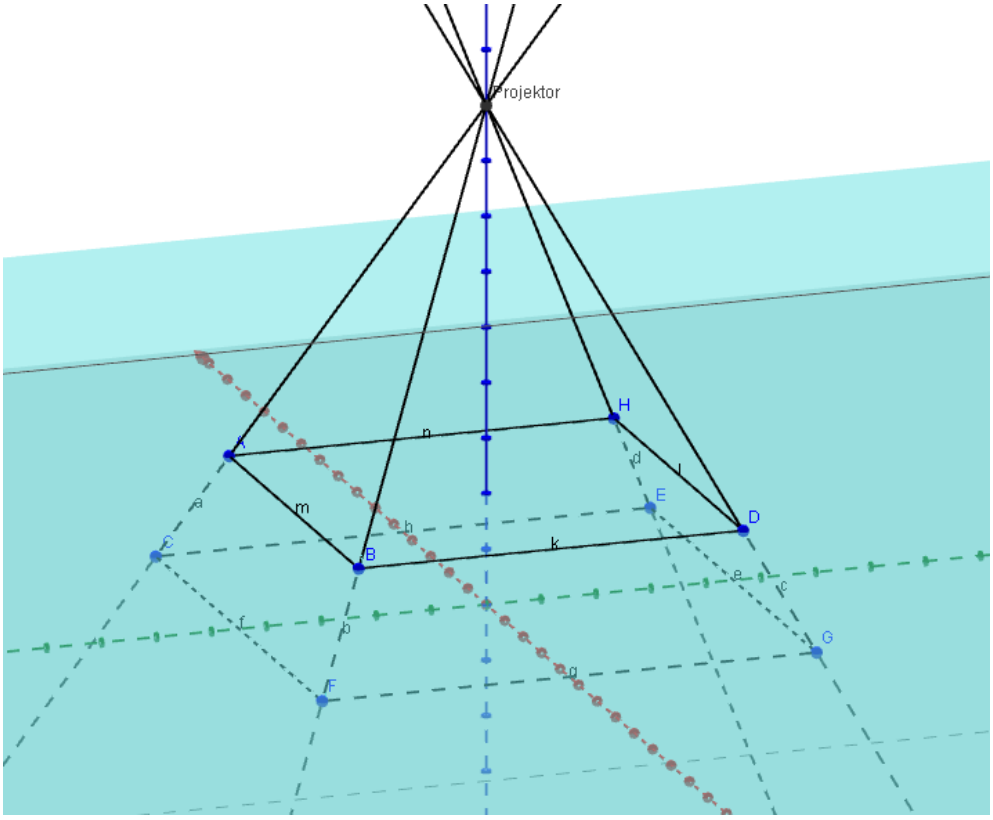
6.5 Príklad: projekcia prstu na projekčnú plochu

v nasledujúcej sekcii. Nasledujúca sekcia vysvetľuje prečo stačí načítať iba 8 bodov a vedieť pozíciu projektoru.

Kalibračná metóda

Kalibračný program využíva jednoduchý princíp na zistenie polohy projektoru a na zistenie údajov na transformáciu bodov zo súradnicového systému senzoru do súradnicového systému projektoru. Projektor pri svojej funkcií vyžaruje lúče svetla z jedného bodu a to šošovky na projekčnú plochu v tvare kužeľa. Ak by som do vyžarovacieho poľa kužeľa dal prst, jeho tieň by bol vlastne projekciou prstu na projekčnej ploche. Na tento výpočet postačujú dva body, a to bod prstu a body z ktorého vyžarujú svetelné lúče projektoru. Takýto príklad je možno vidieť na obrázku 6.5. Tento jednoduchý princíp využíva aj samotná aplikácia kreslenia.

Pre takýto výpočet je nutné poznať polohu projektoru a prstu, polohu projektoru je nutné vypočítať a polohu prstu nám bude dávať leap motion. Pre výpočet polohy projektoru je znova využitý rovnaký princíp, a to tak, že kalibračný program načíta štyri body v projekčnej rovine a štyri body v rovine, ktorá je síce rovnobežná s projekčnou, ale je bližšie k projektoru, čím dostanem dvojice bodov, ktoré keď preložím priamkami a vypočítam ich priesečník dostanem práve polohu projektoru. Ideálnym prípadom pretnutia priamok, je že sa všetky priamky pretnú v jednom bode. V reálnom svete to tak nie je. V drvivej väčšine prípadov sa priamky v priestore nepretnú a je nutné to riešiť inak, čomu sa venuje nasledujúca sekcia. Ideálny prípad pretnutia viacerých priamok je možné vidieť na obrázku



6.6 Príklad: ideálny prípad pretnutia priamok v priestore

6.6. V aplikácií sa využíva osem bodov, na zistenie pozície projektoru by stačili štyri, a to také, ktoré by definovali dve priamky, ktorých priesečník by bol hľadaným bodom.

Výpočet vzájomnej polohy projektoru a Leap Motion

Výpočet polohy projektoru sa nedá určiť jednoduchých matematickým určením priesečníka viacerých priamok. Keďže dve priamky v priestore sa v reálnom svete pretnú asi tak v 1% prípadov, teda vypočítať priesečník priamok, ktoré sa nikdy nepretnú je nereálne. Avšak tento problém sa da obísť, a to tak, že program bude počítať najbližší bod k jednej aj druhej priamke. Takýto bod je vlastne stredom najkratšej úsečky, ktorá je kolmá na jednu aj druhú priamku. Výpočet tohto bodu prebieha nasledujúcim spôsobom, ktorý je ďalej opisovaný.

Na začiatku výpočtu si program určí parametrické vyjadrenie priamky. Priamka v programe je určená dvomi bodmi načítanými pri získavaní bodov rovín, parametrické vyjadrenie priamky v priestore určenej bodmi A a B vyzerá nasledovne:

$$x = A_x + s_x * t \tag{6.1}$$

$$y = A_y + s_y * t \tag{6.2}$$

$$z = A_z + s_z * t \tag{6.3}$$

Kde smerový vektor s je vypočítaný ako rozdiel bodu **A** a **B**.

$$s = \boldsymbol{B} - \boldsymbol{A} = (B_x - A_x\,, B_y - A_y\,, B_z - A_z) \tag{6.4}$$

Po výpočte smerových vektorov dvoch priamok program určí ich vektorový súčin. Vektorový súčin dvoch vektorov je daný výrazom *(6.5)*, ktorého výsledkom je vektor, ktorý je kolmý na obidva pôvodné vektory [8].

$$a \times b = \left[a_y * b_z - a_x * b_y, a_z * b_x - a_x * b_z, a_x * b_y - a_y * b_x\right] \tag{6.5}$$

Po vytvorení vektoru, ktorý je kolmý na pôvodné smerové vektory priamok, je teda tento vektor kolmý aj na priamky. Ak tento vektor nie je jednotkovým vektorom je nutné vytvoriť z tohto vektora jednotkový. Jednotkový vektor je vektor, ktorého veľkosť sa rovná jeden, teda nesie iba informáciu o smere vektora. Vytvorenie jednotkového vektora pozostáva z jednej operácie a to podeliť každú jeho zložku dĺžkou vektora.

$$dĺžka\ vektora = \sqrt{x^2 + y^2 + z^2} \tag{6.6}$$

$$jednotkový\ vektor = [\frac{x}{dĺžka\ vektora}, \frac{y}{dĺžka\ vektora}, \frac{z}{dĺžka\ vektora}] \tag{6.7}$$

Potom program pomocou dvoch bodov, kde jeden patrí jednej priamke a druhý bod druhej priamke vytvorí priamku. Pomocou novo získanej priamky, jej smerového vektoru a spomínaného jednotkového vektoru sa pomocou vektorovej projekcie(6.8) vytvorí vektor, ktorý je kolmý na pôvodné priamky a jeho dĺžka je práve najkratšou vzdialenosťou pôvodných priamok.

$$proj_b a = \frac{\vec{a} * \vec{b}}{|\vec{b}|} \tag{6.8}$$

$$kde\ \vec{b}\ je\ jednotkový\ vektor$$

Pričítaním jednotlivých zložiek vypočítaného vektora k bodom definujúcim prvú priamku, program určí priamku, ktorá je v rovine s druhou priamkou a teda je možné matematicky vypočítať ich priesečník.

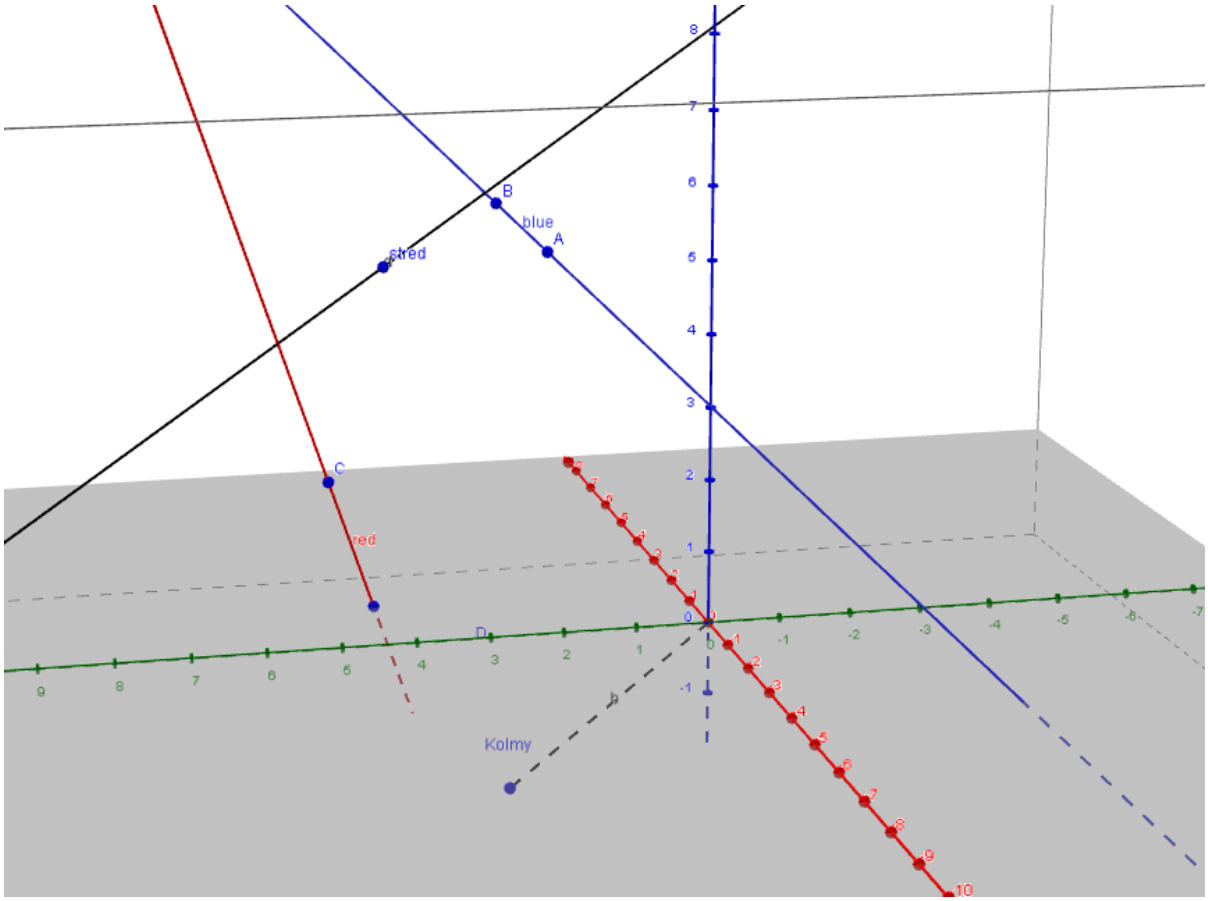
$$Priesečník\ dvoch\ priamok\ p\ a\ q,$$
$$sp\ je\ smerový\ vektor\ priamky\ p,$$
$$sq\ je\ smerový\ vektor\ priamky\ q$$
$$A\ \in p\ , B\ \in q$$
$$t = \frac{(B_y - A_y) * sq_x - (B_x - A_x) * sq_y}{sp_y * sq_x - sp_x * sq_y} \tag{6.9}$$
$$Priesečník:$$
$$x = A_x + t * sp_x \tag{6.10}$$
$$y = A_y + t * sp_y \tag{6.11}$$
$$z = A_z + t * sp_z \tag{6.12}$$

Ak už vieme jeden priesečník a vlastne vektor ktorý je kolmý na jednu aj druhú priamku pomocou nich vytvoríme priamku a vypočítame druhý priesečník rovnakým spôsobom. Hľadaný bod, ktorý je na najkratšej úsečke kolmej na obidve priamky je vlastne stredom úsečky, definovanej vypočítanými dvomi priesečníkmi [9] [10]. Vzorový príklad výpočtu je možné nájsť na pribalenom DVD vo formáte, ktorý používa spomínaný matematický softvér GeoGebra.

Algoritmus popisujúci predchádzajúci výpočet vyzerá takto:

- Vyjadri smerové vektory priamok a vypočítaj ich vektorový súčin
- Transformuj vypočítaný vektor na jednotkový
- Vytvor ľubovoľnú priamku, ktorá pretína obidve pôvodné priamky
- Smerový vektor vytvorenej priamky použi na vektorovú projekciu so spomínaným jednotkovým vektorom
- Pomocou vypočítaného vektora vypočítaj priesečník s prvou a druhou priamkou

- Vypočítaj stred úsečky definovanej dvomi priesečníkmi



6.7 Vzorový príklad: výpočet bodu, ktorý je najbližšie k červenej a modrej priamke

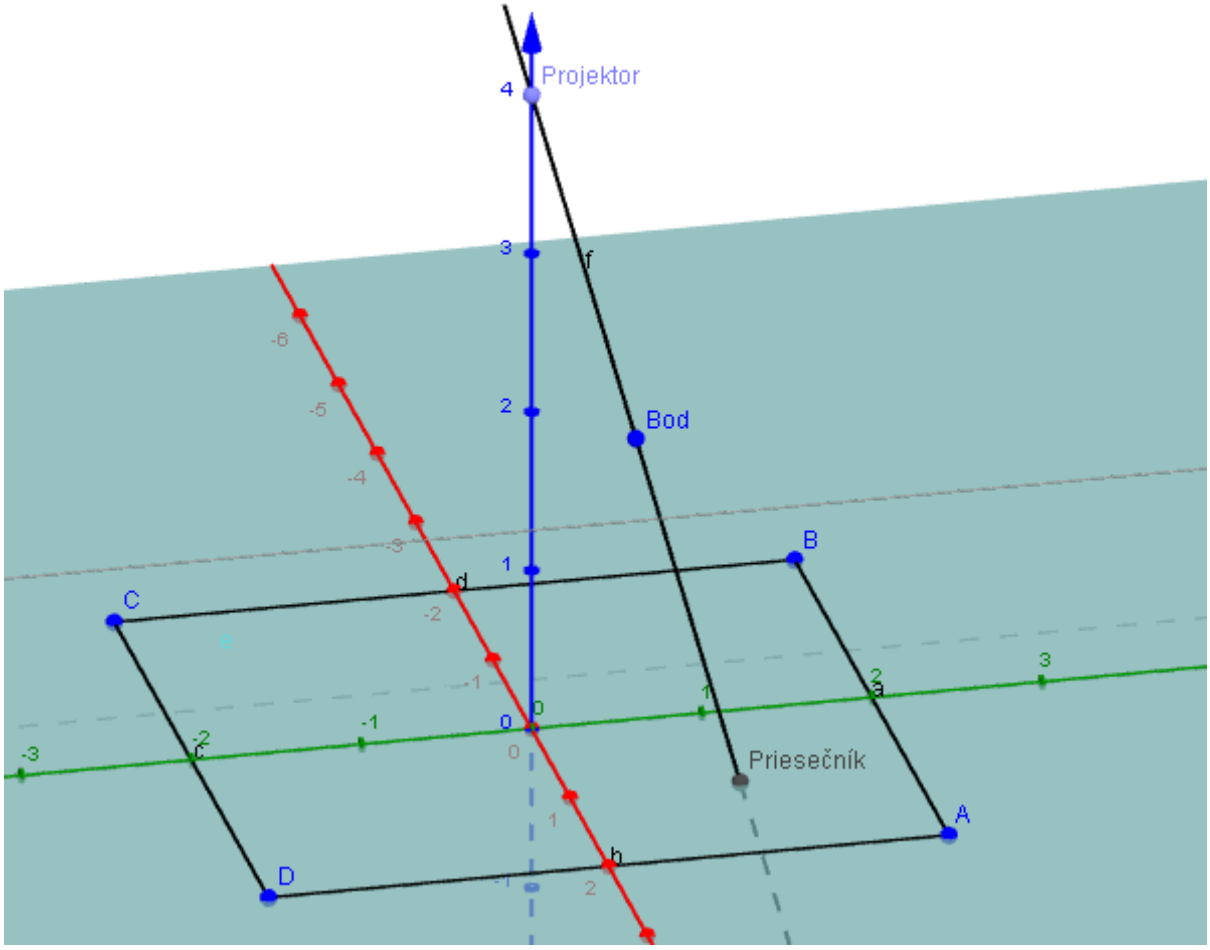
6.3 Program kreslenia

Aplikácia bola navrhovaná aby bola čo najjednoduchšia s využitím jednoduchých gest a použitím jednoduchých nástrojov s využitím tretieho rozmeru. Ak aplikácií chýba kalibračný súbor je schopná spustiť kalibračný program. V aplikácií okrem grafického rozhrania je nutné riešiť načítavanie kalibračných údajov z xml súboru a ich použitie na transformáciu bodu zo súradnicového systému senzoru leap motion do súradnicového systému projektoru.

6.8 Aplikácia kreslenia

Výpočet obrazových súradníc projektoru

V aplikácií je nutné zabezpečiť presnú transformáciu bodov, ktoré poskytuje senzor do súradnicového systému projektoru. Táto činnosť sa deje zakaždým pri načítaní nového snímku zo senzoru. Pri tejto činnosti sa vyberie pozícia aktívneho prstu zo snímky a následne sa prepočíta do súradnicového systému projektoru. Proces prepočítavania rieši trieda `Calculator`, ktorá má na tento problém implementovanú metódu `to2D()`, ktorá ako parameter dostane jeden 3d bod a výstupom je korešpondujúci 2d bod v súradnicovom systéme projektoru. V tomto výpočte ide o priesečník priamky a roviny v priestore. Priamky je v tomto prípade definovaná dvomi bodmi a to bod, kde sa nachádza projektoru a bod, pre ktorý počítame priesečník. Rovina je v tomto prípade definovaná ľubovoľnými tromi bodmi zo zobrazovacej plochy. Teda okrem bodu pre ktorý sa počíta priesečník je nutné poznať niekoľko údajov, ktoré boli uložené do kalibračného súboru. Je nutné si vybrať nulový bod, ktorý bude



6.9 Priesečník s rovinou

slúžiť na výpočet prírastku v osi x a osi y. V tomto konkrétnom prípade som volil ako nulový bod ľavý horný bod kalibračného vzoru. Z matematického hľadiska údaje nutné pre výpočet sú

- Body zobrazovacej roviny
- Bod polohy projektoru

Názorný príklad je možné vidieť na obrázku 6.9, kde je zobrazená priamka vytvorená z nasnímaného bodu a bodu projektoru s rovinou.

Algoritmus na výpočet transformácie bodu z priestoru do súradnicového systému projektoru vyzerá nasledovne:

- Vyjadri parametricky priamku z načítaného bodu a bodu polohy projektoru
- Vyjadri parametricky dve priamky zobrazovacej roviny
- Vyjadri rovnicu definujúcu rovinu v priestore
- Vypočítaj priesečník priamky a roviny
- Vypočítaj posuv v osi X oproti nulovému bodu
- Vypočítaj posuv v osi Y oproti nulovému bodu
- Zohľadni pomer milimetrov na pixely

- K nulovému bodu pripočítaj posuvy
- Výsledný bod vráť ako výstupný

Program na začiatku výpočtu zostaví rovnicu roviny a parametricky vyjadri priamku nasnímaného bodu a bodu polohy projektoru. Parametrické vyjadrenie priamky je popísané v rovnici *(6.1)*. Rovina v priestore je definovaná nasledujúcou rovnicou.

$$a * x + b * y + c * z + d = 0 \tag{6.13}$$

Na vyjadrenie roviny pomocou rovnice *(6.13)* je nutné vyjadriť dve priamky zobrazovacej roviny pomocou uložených bodov v kalibračnom súbore. Vyjadrením smerových vektorov*(6.4)* dvoch priamok je možné následne vytvoriť vektorový súčin [8], ktorý je definovaný rovnicou *(6.5)*. Pomocou vytvoreného vektorového súčinu a jedného bodu roviny je možné zostrojiť rovnicu roviny*(6.13)*. Do tejto rovnice dosadíme známe hodnoty. Neznáme a, b, c nahradíme za známe hodnoty vektorového súčinu, a za neznáme x, y ,z dosadíme hodnoty bodu. Poslednú neznámu d v tejto rovnici dopočítame. Výpočet priesečníka je definovaný postupom:

$$Priesečník\ priamky\ a\ roviny$$
$$su\ je\ vektorový\ súčin,$$
$$s\ je\ smerový\ vektor\ priamky,$$
$$A\ je\ nasnímaný\ bod\ patriaci\ priamke$$

$$k = \frac{-d - (su_x * A_x) - (su_y * A_y) - (su_z * A_z)}{su_x * s_x + su_y * s_y + su_z * s_z} \tag{6.14}$$

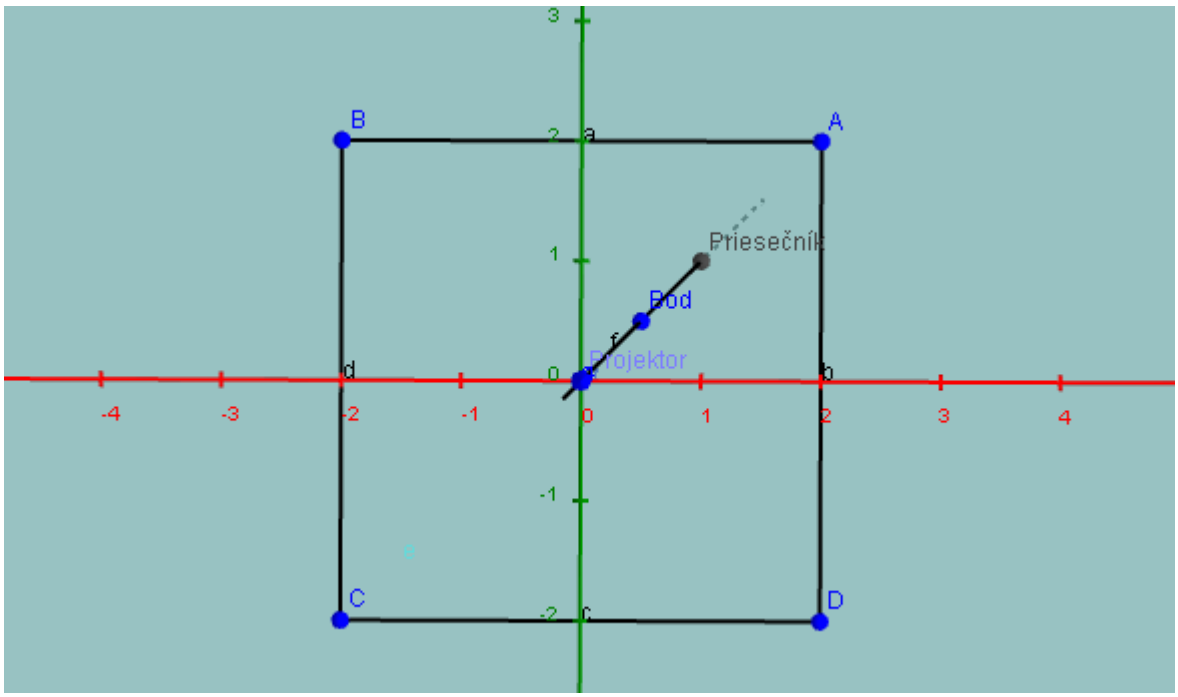
$$Priesečník:$$

$$x = A_x + k * s_x \tag{6.15}$$

$$y = A_y + k * s_y \tag{6.16}$$

$$z = A_z + k * s_z \tag{6.17}$$

Po vyjadrení priesečníka priamky s rovinou sa počíta posuv v osi x a osi y. Posuvy sa počítajú ako vzdialenosť bodu od priamky, v tomto prípade vzdialenosť vypočítaného priesečníka s priamkami definovanými bodmi zobrazovacej roviny. Na obrázku 6.11 je vidieť vzdialenosti dvoch priamok od priesečníka a ako nulový bod zvolený ľavý horný bod. Teda je nutné vypočítať dĺžku úsečky ktorá je kolmá na priamku a prechádza priesečníkom. Priamky sú definované nulovým bodom, teda ľavým horným a ďalším bodom roviny, pravým horným bodom a ľavým dolným.



6.10 Priesečník v rovine

Výpočet bodu od priamky v rovine sa dá matematicky vypočítať. Je nutné poznať iba smerové vektory priamok. Bod ktorého vzdialenosť od priamky poznáme je to vypočítaný priesečník. Výpočet prebieha ako je popísané všeobecnou rovnicou *(6.13).* Za neznáme a, b, c sa dosadia hodnoty smerového vektoru priamky a za neznáme x, y, z sa dosadia hodnoty priesečník a z rovnice sa vypočíta neznáma *d*. Po výpočte poslednej neznámej pokračuje výpočet nasledujúcou rovnicou

$$Vzdialenosť\ bodu\ od\ priamky$$
$$vec\ je\ smerový\ vektor\ priamky,$$
$$A\ je\ bod\ patriaci\ priamke$$

$$k = \frac{-(vec_x * A_x) - (vec_y * A_y) - (vec_z * A_z) + d}{{vec_x}^2 + {vec_y}^2 + {vec_z}^2} \tag{6.18}$$

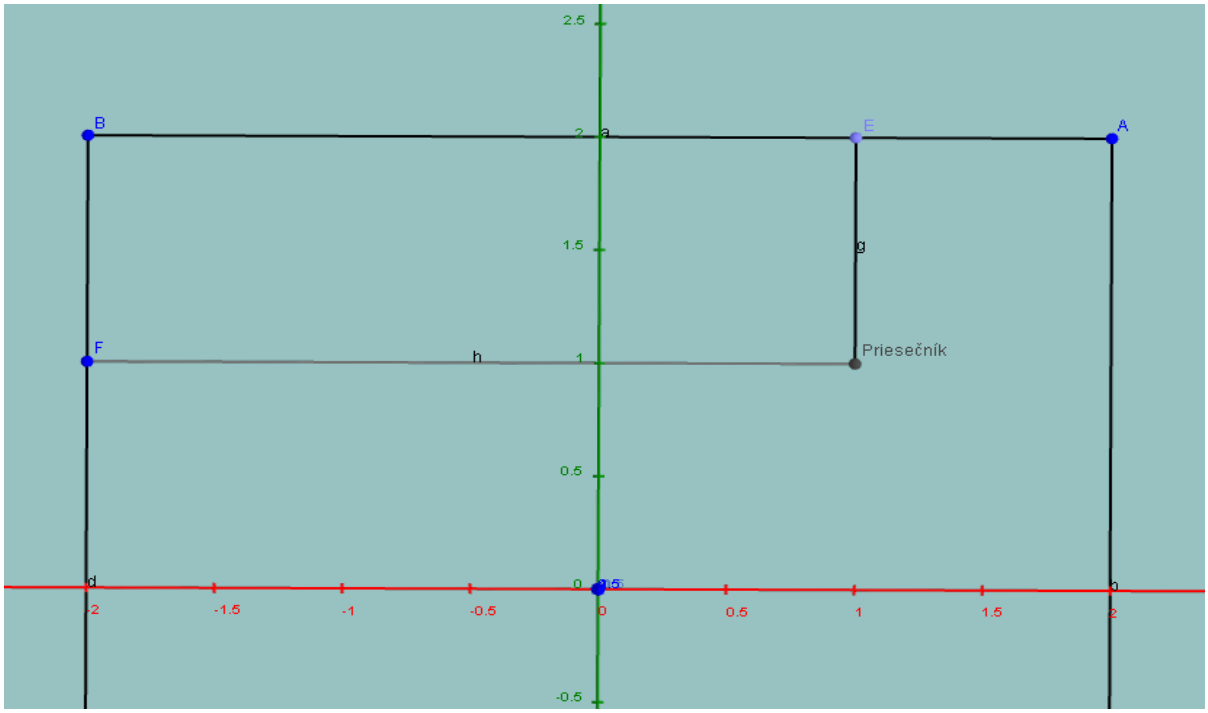
Bod úsečky $patriaci\ priamke, ktorá\ je\ kolmá\ na\ priamku\ a\ prechádza\ bodom$:

$$x = A_x + k * vec_x \tag{6.19}$$

$$y = A_y + k * vec_y \tag{6.20}$$

$$z = A_z + k * vec_z \tag{6.21}$$

Predchádzajúcimi rovnicami vypočítame bod ležiaci na pôvodnej priamke, ktorý leží na kolmici, ktorá prechádza priesečníkom. Na obrázku 6.11 je možné vidieť kolmice a výpočet bodov *E* a *F*, ktoré ležia



6.11 Kolmice cez priesečník

na priamkach definujúcich rovinu a zároveň ležia na kolmiciach prechádzajúcich priesečníkom. Vzdialenosť bodov v priestore sa dá vypočítať rovnicou *(6.22)*, ak už sú známe tieto posuvy v osi x a osi y, tak sa následne k bodu, ktorý korešponduje s nulovým bodom, ale je súradnicovom systéme

$$|AB| = \sqrt{(B_x - A_x)^2 + (B_y - A_y)^2 + (B_z - A_z)^2} \qquad (6.22)$$

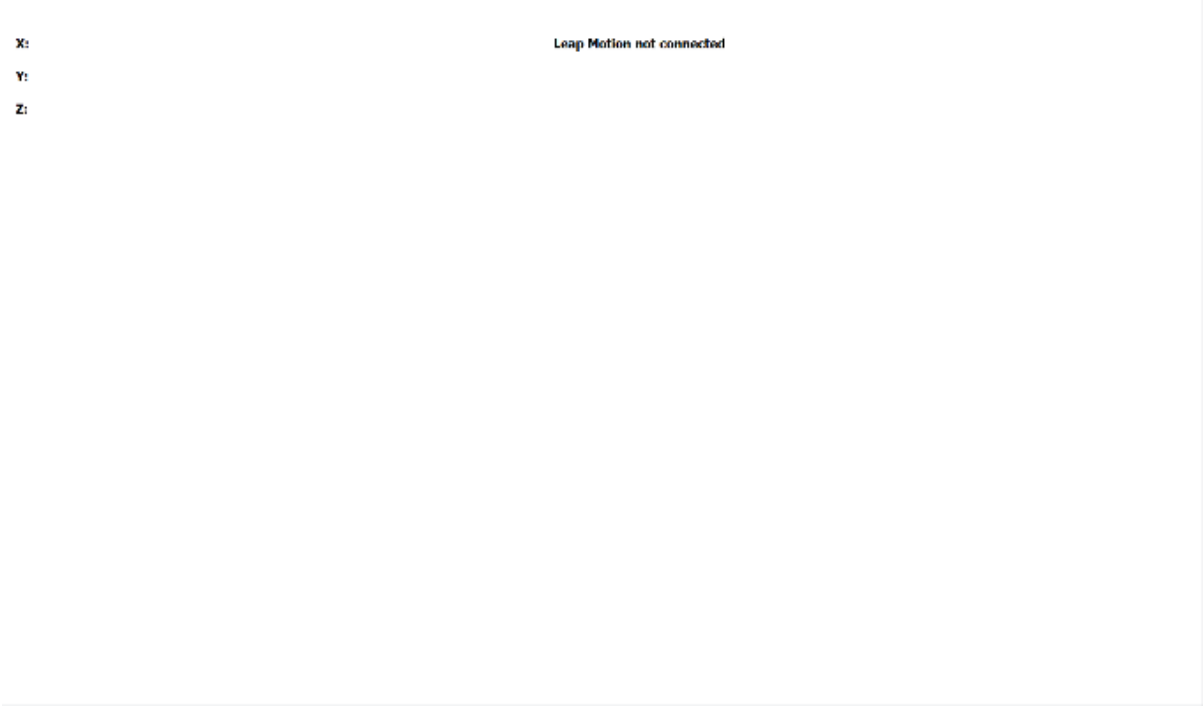
projektoru pripočítajú hodnoty posuvov, ktoré sú vynásobené konštantou reprezentujúcou pomer milimetrov a pixelov. Výsledný bod je transformovaný bod zo súradnicového systému senzoru do súradnicového systému projektoru [9].

7 Implementácia

Aplikácia je rozdelená na dva funkčné nezávislé celky implementované v jazyku C++ s využitím knižnice Qt [7]. V aplikáciách je možné nájsť iba komunikáciu objektov pomocou signálov. Kalibračný program rieši iba načítavanie kalibračných bodov výpočet polohy projektoru a uloženie získaných dát. V aplikácií kreslenia sa program stará o kreslenie správnym nástrojom, umožňuje výber farby alebo definovanie vlastnej.

7.1 Kalibračný program

Hlavnú obrazovku tvorí plátno na ktorom sú zobrazené body kalibračného vzoru v rohoch, v strede sa zobrazuje informácia o priebehu kalibrácie a načítavané informácie o polohe prstu na ruke. Pri vytvorení okna aplikácie sa vytvorí inštancia triedy `Leap_handler`, ktorá je následne signálmi prepojená s hlavným oknom. Hlavné okno je triedou `Leap_handler` informované o zmene pozície



7.1 Okno kalibračného programu – nepripojený leap motion

prstu. Aplikácia zaznamenáva na ruke prst označovaný indexom jeden, čiže ukazovák. V hlavnom okne sa udržiavajú posledné tri pozície prstu, z ktorých sa následne robí priemer a výsledný bod sa ukladá. Celá aplikácia je riadená triedou `Calibration_thread`, ktorá vlastne predstavuje vlákno riadiace aplikáciu. Vlákno sa vytvorí a spustí ak už bol pripojený Leap Motion, alebo až po pripojení zariadenia do USB portu. Vlákno postupne zvýrazňuje body na obrazovke, medzi jednotlivými zvýrazneniami je doba päť sekúnd a až po piatich sekundách sa vypočíta priemer a bod sa uloží. Po načítaní prvých

štyroch bodov aplikácia informuje, aby si osoba kalibrujúca systém vložila na stôl držiak s plátnom na ktorom budú namerané ďalšie štyri body.

7.2 Okno kalibračného programu počas kalibrácie

Teda po načítaní všetkých ôsmich bodov sa vypočíta poloha projektoru v súradnicovom systéme senzoru, následne sa do súboru s názvom `calib_data.xml` uloží bod projektoru, tri body zobrazovacej plochy v súradnicovom systéme senzoru a tri body v súradnicovom systéme projektoru. Po uložení výsledkov aplikácia informuje o ich uložení a po potvrdení informačného okna sa ukončí.
Grafické rozhranie kalibračného programu bolo vytvorené pomocou multiplatformovej knižnice Qt [7]. Na vytvorenie okna bola použitá základná trieda `MainWindow`, ktorá poskytuje hlavné okno aplikácie, ktoré obsahuje ďalšie objekty užívateľského rozhrania. Hlavné okno aplikácie sa zobrazuje na celú obrazovku a nie je možné s ním nijako manipulovať. Na zobrazenie plátna, v ktorom sa zobrazujú kalibračné body bola využitá trieda `QGraphicsScene`, ktorá sprostredkuje 2d plátno pre manažovanie veľkého množstva 2d grafických objektov. Pre zobrazenie informácií na plátne ako poloha prstu alebo informácie o postupe kalibrácie je využitá trieda `QGraphicsTextItem`.

7.2 Program kreslenia

Po načítaní programu sa program snaží načítať súbor s názvom `calib_data.xml`, ktorý by mal obsahovať dáta z už vykonanej kalibrácie. Ak tento súbor neexistuje program na túto skutočnosť upozorní sa zobrazí dialógové okno s požiadavkou na spustenie programu kalibrácie. Ak súbor už bol vytvorený aplikácia pokračuje načítaním údajov z tohto súboru, následne vytvorí objekt triedy `Leap_listener`, ktorý implementuje spracovanie pozície prstu a jednoduchých gest. Následne za touto triedou je objekt triedy `Leap::Controller`, v ktorom je povolené gesto rýchleho pohybu ruky a je mu priradená trieda `Leap_listener`, ktorá definuje metódy, ktoré budú zavolané pri vyskytnutí sa určitej udalosti, taktiež rieši prepočet bodu prstu z súradnicového systému senzoru do súradnicového systému projektoru. Následne je trieda `Leap_listener` prepojená signálmi s hlavným oknom poskytujúcim grafické rozhranie.

Hlavné okno je ako v prípade kalibračného programu roztiahnuté na celú obrazovku a nie je možné s ním nijako manipulovať. Hlavné okno je informované o polohe prstu už v súradnicovom systéme projektoru a o vzdialenosti prstu od plochy na ktorú premieta projektor. Okrem týchto údajov dostáva údaje o rozpoznaných gestách. Hlavné okno rieši zobrazovanie bočného panela nástrojov, zobrazovanie okna na výber farby, zobrazovanie animácií a ukladá si stav aplikácie, aby sa dokázala správne zachovať.

Bočný panel zobrazuje nástroje aplikácie, medzi ktoré patria nastavenie aktívneho prstu, nastavovanie aktívnej farby, nastavovanie farby prednastavenej, alebo definovanej užívateľom, nastavenia nástroja na kreslenie a nastavenie hrúbky kresliaceho nástroja. Panel nástrojov sa zobrazuje na pravej strane aplikácie a jeho zobrazovanie a skrývanie je možné ovládať pomocou gesta. Na paneli je možné nájsť štyri nástroje: pero, štetec, gumu, štvorec. Z týchto nástrojov je možné vybrať si jeden, s ktorým sa bude kresliť. Kreslenie je možné začať už spomínaným potvrdzovacím gestom pre pero, štetec a gumu. Kreslenie pomocou nástroja štvorec prebieha ináč ako spomínané ďalšie tri nástroje, a to tak, že aj sa rozpozná potvrdzovacie gesto a zapamätá sa poloha prstu. Následne sa naťahuje štvorec na obrazovke podľa pohybu prstu. Pre ukončenie naťahovania sa jednoducho uvoľnia prsty do vystretej polohy. Okrem spomínaných nástrojov je možné si zobraziť paletu farieb a vybrať si jednu z preddefinovaných farieb alebo definovať vlastnú farbu. Pre definovanie vlastnej farby je nutné kliknúť v palete farieb na tlačidlo „vlastná farba“ a posúvaním prstu po zobrazovacej ploche je možné meniť jednotlivé zložky farby. Zmenou výšky je možné nastaviť priehľadnosť farby. Plátno na ktoré aplikácia kreslí

7.3 Manipulácia s aplikáciou

zabezpečuje trieda `QGraphicsScene` rovnako ako v prípadne kalibračného programu. Plátno je tmavo šedej farby, kvôli lepšiemu zobrazovaniu na tmavej látke.

7.3 Podporované gestá

Aplikácia podporuje iba dve gestá, jedným je gesto na zobrazenie bočného panelu nástrojov a druhé potvrdzovacie. Pre zobrazenie, či skrytie bočného panelu bolo zvolené gesto z SDK pre leap motion a to gesto posunutia ruky s vystretými prstami. Pri rozpoznaní tohto gesta, ak bol bočný panel skrytý, sa panel zobrazí, a ak bol zobrazený sa pri rozpoznaní rovnakého gesta skryje. Pre potvrdzovacie gesto je zvolené gesto dotyku palca a ľubovoľného prstu, ale je lepšie voliť pri manipulácií s aplikáciou dotyk palca a ukazováka. Pri dotyku palca a ukazováka sa nemení o toľko pozícia ukazováka, na ktorý aplikácia reaguje aj zmenou polohy pri zobrazovaní. Okrem ukazováka aplikácia dovoľuje zmeniť aktívny prst a teda umožňuje používať ukazovák aj prostredník. Je vhodné jedným prstom kresliť a druhým prstom potvrdzovať.

8 Testovanie

Testovanie aplikácie prebiehalo interakciou užívateľa s navrhnutým rozhraním. Užívateľ používal nakalibrované rozhranie a pracoval už iba so samotnou aplikáciou kreslenia.
Jednou z testovaných vlastností rozhrania bola odozva a presnosť na pohyb prstu. Test prebiehal formou zobrazovania známych bodov na plochu a následným porovnávaním známych bodov s prepočítanými bodmi.

Pokus č.	Referenčný / nameraný bod			
1	[200,200] / [204.45,198]	[200,400] / [201.78,405.03]	[400,200] / [402.89,198.43]	[400,400] / [399.25,396.78]
2	[200,200] / [202.3,203.04]	[200,400] / [203.66,404.51]	[400,200] / [400.45,203.10]	[400,400] / [398.03,398.47]

Tabuľka 1 Presnosť merania

Z predchádzajúcej tabuľky vyplynulo, že aplikácia nepracuje úplne presne ale má určitú mieru odchýlky(1mm-5mm), čo môže byť spôsobené ľudským faktorom, alebo programovým rozhraním. Okrem tohto problému aplikácia nereaguje na príliš malé zmeny pohybu prstu (5 – 10 pixelov) a ukazovateľ pohybu neprekreslí, čo je zapríčinené už spomínaným programovým rozhraním a spôsobom zobrazovania samotného ukazovateľa. Tento test potvrdzuje, že aplikácia dokáže správne a rýchlo reagovať na zmenu pozície prstu iba v osi x, teda pri pohybe doľava a doprava. V osi y bola odozva aplikácie výrazne horšia, aplikácia na zmenu pozície prstu nereagovala dostatočne rýchlo, čo spôsobovalo trhanie a defekty. Test odozvy teda nedopadol dobre kvôli chybe pri posúvaní prstu v osi x, túto chybu by mohla odstrániť zmena v prepočítavaní bodov. Ďalšou z testovaných vlastností bolo rozpoznávanie gest. Ak bola dlaň v horizontálnej polohe pod senzorom, tak nastávali určité komplikácie pri rozpoznávaní. A to z dôvodu, že senzor dokáže rozpoznať iba najbližšie objekty a teda zohnutý prst pod dlaňou nemôže detekovať. Pre zlepšenie detekcie je možné pootočiť dlaň tak, aby bolo vidno zohnuté prsty pod dlaňou.
Test, ktorý overoval funkciu kreslenia s rôznymi nástrojmi odhliadnuc od spomínaných chýb, neodhalil žiadne ďalšie chyby a nepresnosti. Pri testovaní sa môže užívateľ stretnúť s anomáliou v podobe náhodne odrazeného svetla, čo môže spôsobovať zle načítané údaje senzorom zo scény.
Ďalšie testovanie prebiehalo formou interakcie užívateľov so samotnou aplikáciou. Užívatelia mali za úlohu nakresliť obrázok pomocou rôznych nástrojov. Táto úloha zahŕňala kreslenie s využitím rôznych nástrojov, zmenu farby a možnosť zmeniť vlastnosti kresliaceho nástroja. Užívateľom bol systém vopred nakalibrovaný a boli im vysvetlené základné gestá pre ovládanie aplikácie. Na konci testu každý jeden zo zúčastnených mal za úlohu vyplniť vopred pripravený dotazník, ktorý obsahoval otázky týkajúce sa práce s rozhraním a aplikáciou. Tohto testu sa zúčastnilo niekoľko respondentov a z výsledkov vyplynulo nasledovné.

Kreslenie obrázku

Test spočíval v opakovanom kreslení vzorového obrázku. Pri jednotlivých pokusoch bol užívateľovi meraný čas a jeho hodnoty zaznamenávané v nasledujúcej tabuľke.

Pokus č.	Čas				
	Respondent 1	Respondent 2	Respondent 3	Respondent 4	Respondent 5
1	10m 13s	15m 42s	13m 22s	11m 40s	9m 35s
2	8m 34s	12m 56s	11m 56s	10m 48s	8m 49s
3	9m 10s	10m 14s	10m 01s	9m 10s	7m 20s
4	7m 50s	10m 47s	10m 42s	8m 32s	6m 59s

Tabuľka 2 Čas kreslenia obrázku respondentami

Z predchádzajúcej tabuľky vyplýva, že respondenti sa s každým pokusom zlepšovali. Každému respondentovi po prvom pokuse boli poskytnuté rady, čo sa prejavilo na ich zlepšení.

Vyhodnotenie výsledkov dotazníka

Z dotazníka poskytnutého po teste respondentov s rozhraním vyplynulo, že pre väčšinu respondentov bolo stretnutie s aplikáciou zaujímavou skúsenosťou. Nemali problém sa zorientovať v aplikácii no nakoľko s týmto typom užívateľského rozhrania nikto nepracoval, jeho prvotné použitie bolo náročné. Väčšina respondentov uviedla, že aplikáciu za účelom kreslenia obrázkov by v budúcnosti nepoužila. Užívatelia by privítali zlepšenie presnosti a pridanie nových nástrojov na kreslenie.

8.1 Návrh ďalšieho vývoja

Ak by mal vývoj aplikácie pokračovať bolo by nutné zaoberať sa už známymi chybami, ktoré odhalilo testovanie aplikácie. Pre zlepšenie odozvy programu a na rýchlejšie prepočítavanie bodu, by bolo treba zmeniť alebo optimalizovať algoritmus výpočtu.
Okrem spomenutých zlepšení by bolo vhodné sa zamerať na nasledujúce veci:

- Pridanie projektoru – zlepšenie zobrazovacích vlastností, rozšírenie zobrazovacieho plátna
- Nové gestá – pridanie nových gest, ktoré zjednodušia prácu
- Nástroje využívajúce tretí rozmer
- Automatická kalibrácia – zautomatizovať kalibráciu, ktorá by prebiehala pred štartom programu
- Využitie pri kreslení fyzických nástrojov – napr. písanie perom
- Využitie kamery pri zobrazovaní na otočených/posunutých/rotovaných plochách

9 Záver

Práca sa zaoberá využitím senzoru leap motion v aplikácií kreslenia. Cieľom bolo navrhnúť rozhranie, ktoré by bolo vytvorené z počítača využívajúceho projektor ako výstupné zariadenie a senzor leap motion ako vstupné zariadenie. Výsledkom práce je dvojica programov, jednoduchý program pre kalibrovanie sústavy projektor-senzor a program umožňujúci kreslenie. Kalibrácia sústavy prebieha rýchlo a bez väčších komplikácií. V aplikácií kreslenia je prepočet z 3d súradníc do 2d súradníc pre projektor o niečo pomalší čo sa prejavuje v odozve aplikácie hýbať s ukazovateľom prstu. Ak aplikácia nestíha počítať súradnice a užívateľ rýchlo hýbe prstom, odozva sa iba zväčšuje. Prejavuje sa to trhaním či sekaním ukazovateľa polohy prstu. Pri hýbaní prstu v osi y je výpočet časovo náročnejší ako pri hýbaní prstu v osi x. Taktiež je nevýhodou, že senzor dokáže rozoznať veci, ktoré sú v horizontálnej pozícií voči jeho kamerám. Vo vertikálnej pozícií dokáže rozoznať iba najbližší objekt. Tento nedostatok by sa v ďalšom vývoji aplikácie mohol odstrániť pridaním ďalšieho senzoru, ktorý by umožňoval aj vertikálne snímanie. Ak by bol pridaný ďalší senzor skomplikovalo by to kalibráciu a transformáciu bodu zo senzorov.

Z testovania vyplynulo, že aplikácia má veľa možností na zlepšenie funkcionality v budúcnosti. Pri testovaní sa užívateľom pracovalo dobre a odniesli si z testovania zaujímavú skúsenosť.

V ďalšom vývoji by bolo vhodné sa zamerať sa na ďalšiu funkcionalitu s využitím tretieho rozmeru, pridať zaujímavejšie gestá, ktoré by urýchlili prácu ale hlavne zrýchliť a optimalizovať výpočet. Taktiež by bolo nutné zmeniť spôsob kalibrácie, pretože pri každej zmene sústavy je nutné znova kalibrovať a pre užívateľa je kalibrácia iba stratou času.

Zoznam obrázkov

2.1 Piliere počítačového videnia 3
2.2 História počítačového videnia 4
2.3 Spracovanie Obrazu 5
3.1 Nekalibrovaná váha. Nezaťažená neukazuje 0. 7
3.2 Testovací vzor televízií pre štandard NTSC(National Television System Committee) 8
4.1 Leap Motion 10
4.2 Leap Motion vizualizér - Ruka 11
4.3 Gesto ťuknutia prstom 13
4.4 Kruhové gesto 13
4.5 Gesto posunutia ruky 13
5.1 Kinect pre Xbox 360 14
5.2 Kinect verzie 2 15
5.3 PlayStation Eye 15
5.4 Wired glove 16
6.1 Navrhovaný systém – Projektor hore na statíve, leap motion otočený smerom k stolu na drevenom stojane 17
6.2 Držiak na senzor leap motion 19
6.3 Držiak na plátno 19
6.4 Projektor na statíve 20
6.5 Príklad: projekcia prstu na projekčnú plochu 20
6.6 Príklad: ideálny prípad pretnutia priamok v priestore 21
6.7 Vzorový príklad: výpočet bodu, ktorý je najbližšie k červenej a modrej priamke 24
6.8 Aplikácia kreslenia 25
6.9 Priesečník s rovinou 26
6.10 Priesečník v rovine 28
6.11 Kolmice cez priesečník 29
7.1 Okno kalibračného programu – nepripojený leap motion 30
7.2 Okno kalibračného programu počas kalibrácie 31
7.3 Manipulácia s aplikáciou 33

Literatúra

[1] R. Szeliski, Computer Vision Algorithm and Applications, Springer, 2011.

[2] S. J. D. Prince, Computer Vision Models, Learning and Inference, Cambridge: Cambridge University Press, 2012.

[3] T. A. Clarke a J. F. Fryer, „The development of camera calibration methods and models," rev. *Photogrammetric Record*, 1998, pp. 51-66.

[4] T. Hanning, High Precision Camera Calibration, Morlenbach: Springer Fachmedien Wiesbaden GmbH, 2011.

[5] „OpenCV," [Online]. Dostupné z : http://opencv.org/. [Cit. 16.4.2015].

[6] G. D. Haeger a J. Corso, „Image description with features that summarize," rev. *Computer Vision and Image Understanding*, 2009, pp. 446-458.

[7] „Qt Cross-platform application & UI development framework," [Online]. Dostupné z : http://www.qt.io/. [Cit. 19.4.2015].

[8] „Vektorový súčin," Wikipédia - Slobodná encyklopédia, [Online]. Dostupné z : http://sk.wikipedia.org/wiki/Vektorov%C3%BD_s%C3%BA%C4%8Din. [Cit. 19.4.2015].

[9] E. W. Weisstein, „Point-Line Distance--3-Dimensional.," MathWorld - A Wolfram Web Resource, [Online]. Dostupné z : http://mathworld.wolfram.com/Point-LineDistance3-Dimensional.html. [Cit. 1.5.2015].

[10] D. H. Eberly, 3D Game Engine Design: A Practical Approach to Real-Time Computer Graphics (Morgan Kaufmann Series in Interactive 3D Technology), CRC Press , 2006.

[11] „Skeletal Tracking," Leap Motion Inc., [Online]. Dostupné z : https://developer.leapmotion.com/. [Cit. 14.4.2015].

[12] „Jednotkový vektor," Wikipédia - Slobodná encyklopédia, [Online]. Dostupné z : http://sk.wikipedia.org/wiki/Jednotkov%C3%BD_vektor. [Cit. 19.4.2015].

[13] R. Hartley a A. Zisserman, Multiple View Geometry in Computer Vision Second Edition, Cambridge University Press, 2004.

[14] S. Dance, Z. Q. Liu a T. M. Caelli, Picture Interpretation: A Symbolic Approach, World Scientific , 1995.

[15] L. P. Wesley, Evidential knowledge-based computer vision, A. I. Centre: SRI International, Menlo Park, 1986.

[16] R. Stolkin, „Scene Reconstruction Pose Estimation and Tracking," InTech, 2007.

Zoznam príloh

Príloha 1. Manuál – kalibračný program

Príloha 2. DVD

Príloha 3. Dotazník

Príloha 1. Manuál – kalibračný program

Na spustenie aplikácie sa pomocou inštalačného balíčka doinštalujú všetky potrebné knižnice a nie je nutné nič riešiť. Ak sa aplikácia zapína po prvý krát a teda nie je vytvorený kalibračný súbor, aplikácia nato upozorní a dá užívateľovi na výber, buď skončiť alebo spustiť kalibráciu.
Pri kalibrácií aplikácia zobrazí štyri body, ktoré postupne zvýrazňuje, užívateľ má iba na body ukazovať prstom hneď na zobrazovacej ploche. Po načítaní štyroch bodov užívateľ vloží na zobrazovaciu plochu držiak s plátnom, aby sa načítali body v druhej rovine. Po vložení plátna sa znova vyznačujú štyri body. Po načítaní všetkých bodov sa vypočíta poloha projektoru a následne sa uložia tri body zobrazovacej roviny, tri body zo súradnicového systému projektoru a bod polohy projektoru.
Príklad kalibračného súboru:

```
<LeapCalibration>
    <Body_zobr_plochy>
        <Bod_zobr_plochy_1>
            <Data>-7 0 </Data>
        </Bod_zobr_plochy_1>
        <Bod_zobr_plochy_2>
            <Data>-7 708 </Data>
        </Bod_zobr_plochy_2>
        <Bod_zobr_plochy_3>
            <Data>1300 0 </Data>
        </Bod_zobr_plochy_3>
    </Body_zobr_plochy>
    <Rovina>
        <Bod_roviny_1>
            <Data>191.563 298.291 -107.337 </Data>
        </Bod_roviny_1>
        <Bod_roviny_2>
            <Data>201.033 297.076 117.25 </Data>
        </Bod_roviny_2>
        <Bod_roviny_3>
            <Data>-207.026 301.588 -89.938 </Data>
        </Bod_roviny_3>
    </Rovina>
    <Bod_projektoru>
        <Data>-58.5163 -221.076 -160.557 </Data>
    </Bod_projektoru>
</LeapCalibration>
```

Popis obsahu súboru calib_data.xml:

Tagy:

- `LeapCalibration` – hlavný tag
- `Body_zobr_plochy` – obaľuje tagy `Bod_zobr_plochy_x` obsahujúce dáta bodov v súradnicovom systéme projektoru
- `Rovina` – obaľuje tagy `Bod_roviny_x` obsahujúce dáta bodov v súradnicovom systéme senzoru

Príloha 2. DVD

Obsahom DVD je:

- src – Projekt pre QtCreator so zdrojovými kódmi aplikácie, testovací súbor výpočtu najbližšieho bodu dvoch priamok
- bin – Preložená aplikácia pre Windows x64 vo formáte inštalačného súboru
- doc – Technická správa vo formáte PDF a docx
- video – Video prezentujúce kalibrovanie sústavy a manipuláciu s aplikáciou kreslenia

Príloha 3. Dotazník

1. Ako ste boli spokojný s grafickým spracovaním aplikácie?
   - Výborné
   - Priemerné
   - Neuspokojivé
2. Ako sa Vám pracovalo s týmto typom aplikácie ?
   - Výborne
   - Dobre
   - Zle
3. Využívali by ste túto aplikáciu pre kreslenie obrázkov aj v budúcnosti?
   - Určite áno
   - Možno
   - Nie
4. Stretli ste sa už s podobnou aplikáciou?
   - Áno
   - Nie
5. Čo by ste na tejto aplikácii zlepšili?
   - Doplňte