Ovládanie počítača pohybom s využitím senzora Leap Motion

Patrícia Szepesiová

3Ib, 2017-2018

Abstrakt Bezdotykové ovládanie počítača v porovnaní s dotykovým rieši otázku hygieny a bezpečnosti, pretože odstraňuje potrebu dotyku s dotykovou plochou zariadenia, ktorá predstavuje zdroj nákaz a ochorení. Obzvlášť rizikové sú verejné obrazovky, ktorých sa denne dotýka nespočetné množstvo ľudí. Práca sa zaoberá návrhom a implementáciou nástroja na ovládanie počítača pohybom ruky s využitím senzora Leap Motion. Hlavným cieľom práce je implementovať nástroj, ktorý s dostatočnou presnosťou emuluje dotykové ovládanie. S využitím analytickej geometrie poskytuje ovládanie absolútneho pohybu kurzora, ktorý je závislý od vzájomnej polohy ruky a obrazovky počítača. Klasická, ale aj menej bežná funkcionalita zariadení na ovládanie počítača je nahradená gestami kurzora, na ktoré je aplikované rozpoznávanie.

Kľúčové slová: interakcia človek-počítač, ovládanie, Leap Motion, kamera pre prirodzenú interakciu, rozpoznávanie gest

1 Úvod

Štandardom v spôsobe ovládania zariadení v poslednom desaťročí je jednoznačne dotyk. Okrem rozmachu používania dotykových mobilných telefónov sa čoraz viac dostávajú do popredia dotykové monitory. Objavujú sa na rôznych verejných priestranstvách, zobrazujú informácie, reklamy alebo poskytujú používateľom rôzne služby. Mnoho z nich poskytuje interakciu práve dotykom.

Napriek tomu, že ovládanie dotykom je veľmi obľúbené, nesie so sebou mnoho negatívnych stránok. Jednou z nich je cena dotykových monitorov. V porovnaní s klasickými monitormi je približne troj- až štvornásobne vyššia. Pri monitoroch väčších rozmerov sú rozdiely až v tisícoch eur. Ďalším dôvodom, prečo nevyužívať verejné dotykové obrazovky, je fakt, že sa denne dostávajú do kontaktu s množstvom ľudí. Výsledky štúdií poukazujú na to, že dotykové plochy obsahujú obrovské množstvo nebezpečných baktérií. Teplota zariadenia vytvára ideálne podmienky pre ich množenie.

V bakalárskej práci sa preto zaoberáme návrhom a implementáciou nástroja na emuláciu dotykového ovládania, ktorý by s dostatočnou presnosťou dokázal nahradiť dotykové ovládanie. Pri implementácii nástroja na ovládanie počítača bez dotyku sa stretávame s niekoľkými problémami. Miesto kliknutia je pri dotykovom ovládaní jednoznačné. Avšak v prípade ovládania bez dotyku je potrebné

určiť, na ktoré miesto obrazovky používateľ v danom momente ukazuje a tiež definovať spôsob, ktorým používateľ dáva pokyn pre kliknutie. S cieľom implementovať nástroj, ktorý bude čo najpresnejšie emulovať dotykové ovládanie, sa zaoberáme rozpoznávaním gest, ktoré ovládajú posúvanie obsahu, presúvanie objektov a ďalšiu bežnú funkcionalitu. Ovládanie bez dotyku implementujeme s využitím zariadenia pre prirodzenú interakciu.

1.1 Zariadenia pre prirodzenú interakciu

Zariadenia pre prirodzenú interakciu sú také zariadenia, ktoré umožňujú používateľovi prirodzene ovládať počítač bez manipulácie s akýmkoľvek zariadením a jeho ovládacími prvkami. I napriek tomu, že pri ovládaní je toto zariadenie prítomné, pre používateľa môže byť neviditeľné, pretože na ovládanie využíva prirodzený pohyb, napríklad rúk, preto je veľmi intuitívne. Zariadenia majú zabudované senzory, ktoré zaznamenávajú polohu jednotlivých častí tela. Spomedzi nich je asi najznámejší Microsoft Kinect [4]. S použitím hĺbkovej a farebnej kamery zariadenie poskytuje informácie o polohe osôb v jeho zornom poli. Ďalším zariadením je Tobii EyeX [3], ktoré poskytuje interakciu s počítačom očami používateľa. Nás bude zaujímať Leap Motion [2], ktorý sníma polohu rúk a prstov. V porovnaní so zariadením Kinect sníma menšie zorné pole, ale poskytuje omnoho vyššiu presnosť, preto sme sa rozhodli práve preň.

1.2 Leap Motion

Leap Motion (viď obrázok 1) je malé USB periférne zariadenie, ktoré má zabudované dve infračervené kamery a tri infračervené LED diódy, pomocou ktorých sníma okolie do vzdialenosti zhruba 1 meter (vzdialenosť je odlišná pri rôznych verziách zariadenia). Priemerná frekvencia snímania je 115 snímok za sekundu. Snímky sa po prenose do počítača zložitými algoritmami prevádzajú na 3D výstup. Výstupom sú pozície kĺbov snímaných rúk, ktoré sa nachádzajú v zornom poli. Priemerná dosahovaná presnosť je 0,7 mm, čo popisuje štúdia[6] z roku 2013.

1.3 Prehľad súčasného stavu

Existuje viacero softvérových nástrojov, ktoré umožňujú ovládanie počítača pohybom ruky s využitím senzora Leap Motion. Medzi najznámejšie patria:

– **Mudra Mouse** poskytuje funkcionalitu podobnú ovládaniu počítača touchpadom, umožňuje pravý a ľavý klik, posúvanie obsahu krúžením prsta. Jej plnú verziu si je potrebné zakúpiť. Funguje v dvoch režimoch, absolútneho a relatívneho pohybu kurzora, pričom relatívny režim povoľuje funkcionalitu

[1] Zdroj: `https://3bonlp1aiidtbao4s10xacvn-wpengine.netdna-ssl.com/wp-content/uploads/2017/03/leap-motion-3d-motion-gesture-controller-10-large.jpg`

Obr. 1. Senzor Leap Motion[1].

podobnú zdvihnutiu myši a presunu na iné miesto bez pohybu kurzora na obrazovke. Avšak ani jeden z týchto režimov neberie do úvahy reálnu pozíciu obrazovky.

– **Touchless** umožňuje používateľovi posúvanie obsahu všetkými smermi jednoduchým posunom prstov vo vzduchu. Ďalšou jeho funkcionalitou je priblíženie a oddialenie obsahu. Neberie do úvahy reálnu pozíciu obrazovky. Dostupný je vo verzii pre Windows a OS X.

Ďalšie nástroje umožňujú podobnú funkcionalitu, alebo jej majú menej, pričom neriešia presúvanie kurzora na miesto, na ktoré smeruje prst používateľa. Prvé verzie Leap Motion SDK podobnú funkcionalitu podporovali, avšak tá bola v určitom momente odstránená a v žiadnej dostupnej verzii sa nenachádza. Vývoj softvéru pre Leap Motion je v súčasnosti viac zameraný na virtuálnu realitu.

2 Návrh riešenia

Prvá časť, ktorú je potrebné navrhnúť a implementovať, je presúvanie kurzora na pozíciu obrazovky, na ktorú ukáže používateľ prstom. Problém je v tom, že zo senzora vieme získať iba pozície jednotlivých kĺbov rúk. Aby sme určili pozíciu kurzora na obrazovke, potrebujeme okrem pozície ruky poznať aj pozíciu obrazovky v zornom poli senzora. Preto je potrebné pred začatím používania kalibráciou získať vyjadrenie plochy obrazovky v zornom poli senzora.

2.1 Kalibrácia

Všetky body obrazovky (pozície kurzora) vieme parametricky popísať pomocou bodu a kombinácie dvoch lineárne nezávislých vektorov. Označme si bod v ľavom hornom rohu obrazovky P_1. Jeho súradnice na obrazovke sú $[0, 0]$. Pravý horný roh P_2 má súradnice $[s, 0]$, kde s predstavuje šírku obrazovky, ktorú dostaneme z rozlíšenia monitora. Súradnice ľavého dolného rohu P_3 sú $[0, v]$, kde v je výška obrazovky, rovnako získaná z rozlíšenia monitora. Plochu obrazovky

vieme definovať ako množinu všetkých bodov X takých, pre ktoré platí nasledujúca rovnica:

$$X = P_1 + k * u + l * v, \tag{1}$$

kde u je vektor $P_2 - P_1$, v je vektor $P_3 - P_1$, a k, l sú parametre z intervalu $< 0, 1 >$.

Týmto vyjadrením vieme popísať plochu obrazovky v trojrozmernej sústave, teda v zornom poli senzora. Avšak na to potrebujeme poznať 3D súradnice bodov P_1, P_2 a P_3.

Kalibráciou určíme zobrazenie troch bodov z 2D sústavy do 3D. Voľba troch bodov, ktoré sa nachádzajú v rohoch obrazovky, nie je náhodná, okrem jednoduchšieho ohraničenia parametrov k a l je dôležité, aby vzdialenosť medzi týmito bodmi bola čo najväčšia. Dôvodom je to, že ich pozíciu v zornom poli senzora nedokážeme určiť so stopercentnou presnosťou, a preto čím väčšia je medzi nimi vzdialenosť, tým sa odchýlky menej prejavia.

Pri získaní pozícií troch bodov v zornom poli senzora sme testovali dve metódy, určenie súradníc bodov dotykom prsta a určenie súradníc bodov odhadovaním. Popísané sú v nasledujúcich sekciách.

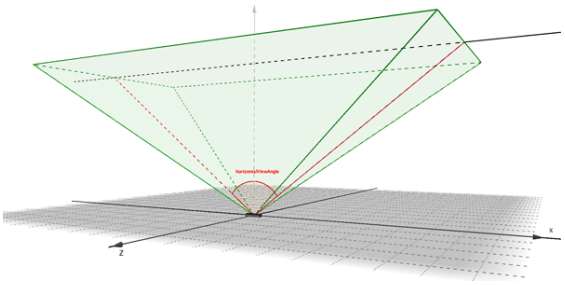
2.2 Určenie súradníc bodov dotykom prsta

Prvý spôsob určenia 3D súradníc bodu je priamočiary. Princíp spočíva v tom, že priložíme prst na pozíciu bodu na obrazovke a dáme počítaču pokyn pre zaznamenanie súradníc konca prsta. I keď je tento spôsob ľahko implementovateľný, má niekoľko nevýhod:

- Bod, ktorý je senzorom označený za koniec posledného článku prsta, sa nemusí zhodovať s reálnym koncom prsta, čím vznikajú odchýlky medzi reálnou a vypočítanou pozíciou plochy obrazovky v zornom poli senzora.
- Podstatou ovládania bez dotyku je nedotýkať sa obrazovky, čomu by sme sa v tomto prípade pri kalibrácii nevyhli.
- Najväčší nedostatok tohto riešenia súvisí s tvarom zorného poľa senzora. Ten sníma zorné pole pod uhlom 150 stupňov v tvare "obrátenej pyramídy" (viď obrázok 2). V mnohých prípadoch použitia senzora na ovládanie bez dotyku sa obrazovka nenachádza v dostatočnej výške na to, aby sa celá nachádzala v zornom poli senzora.

Aj v prípade, že senzor umiestnime na najideálnejšie miesto pred obrazovkou, je pravdepodobné, že časti obrazovky, ako napríklad dolné rohy, nebudú v zábere. Pri samotnom ovládaní to už nie je problém, keďže na obrazovku ukazujeme rukou z určitej vzdialenosti a dôležité je iba to, aby sa prst, ktorý používame na ukazovanie, nachádzal v zornom poli. No začiatočná kalibrácia, ktorá využíva túto metódu získavania súradníc bodov, nie je vo väčšine prípadov možná. Preto sme zvolili druhú metódu získavania 3D súradníc bodov.

[2] Zdroj: `https://di4564baj7skl.cloudfront.net/documentation/images/Leap_horizontalViewAngle.png`



Obr. 2. Zorné pole senzora[2].

2.3 Určenie súradníc bodov odhadovaním

Ako je vyššie spomenuté, pri samotnom ovládaní neprekáža fakt, že sa celá obrazovka nenachádza v zornom poli senzora. Na určenie súradníc bodu preto využijeme ukazovanie prstom na bod, ktorého súradnice chceme zaznamenať.

Princíp tejto metódy je, že vopred určeným prstom ukážeme na bod X, ktorého 3D súradnice $[x_1, x_2, x_3]$ hľadáme, tak, aby sa prst nachádzal v zornom poli senzora a zaznamenáme súradnice dvoch bodov, predstavujúcich kĺby prsta. Tento postup opakujeme n-krát. Označme jeden z dvoch zaznamenaných bodov i-teho opakovania bod $A_i = [a_{i1}, a_{i2}, a_{i3}]$, druhý zaznamenaný bod označme $B_i = [b_{i1}, b_{i2}, b_{i3}]$. Týmito dvomi bodmi je určená priamka p_i, ktorá pretína plochu obrazovky v blízkom okolí bodu X, ktorého súradnice hľadáme.

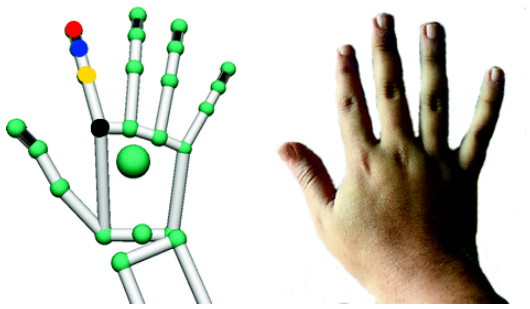
Pri každom z n opakovaní meníme uhol, pod ktorým ukazujeme prstom na bod X, čím dostaneme n rôznych priamok, ktoré by sa v ideálnom prípade mali pretnúť v hľadanom bode X. Počet opakovaní môže byť rôzny (väčší ako 1), treba brať do úvahy, aby bolo určenie súradníc dostatočne presné, no zároveň príliš veľký počet ukazovaní môže byť pre používateľa obťažujúci. Na začiatku sme zvolili počet opakovaní 3. Avšak po početných pokusoch sa ukázalo, že vzhľadom na nepresné určenie súradníc kĺbov ruky senzorom je potrebné počet opakovaní zvýšiť. Preto sme ďalej pracovali s počtom opakovaní 5 pre jeden bod.

S cieľom dosiahnuť čo najvyššiu presnosť sme spomínaný postup testovali pre rôzne dvojice kĺbov prsta, popísané v nasledujúcich bodoch:

– Spočiatku sme považovali za ideálnu dvojicu bodov súradnice začiatku a konca posledného článku prsta (na obrázku 3 zobrazené červenou a modrou farbou). Zdalo sa prirodzené použiť práve tieto dva body, pretože v skutočnosti by nemalo záležať na tom, ako je prst vo svojich kĺboch ohnutý, smer ukazovania by mal byť určený iba smerom jeho posledného článku. Problém je opäť v odchýlkach súradníc prstov získaných zo senzora od ich reálnej polohy. Vzdialenosť medzi bodmi je príliš malá, čo v konečnom dôsledku spôsobuje veľké odchýlky v smere získanej priamky od skutočného smerovania prstu.

– Preto je žiaduce, aby sa body, ktorými určujeme priamku, nachádzali v čo najväčšej vzdialenosti. Vhodné by mohli byť body začiatku a konca prsta (na obrázku 3 zobrazené čiernou a červenou farbou). Pri ideálnej polohe ruky nad senzorom, kedy senzor sníma otvorenú dlaň, sme pozorovali lepšie výsledky ako v prvom prípade. Veľké odchýlky nastávali, keď bola ruka v inej polohe (napr. s ohnutými prstami).
– Súradnice kĺbu medzi prvým a druhým článkom prsta (na obrázku 3 zobrazeného žltou farbou) sú vo väčšine prípadov určené s menšou odchýlkou, preto sme sa napokon rozhodli využiť práve ten.

Preto priamky prekladáme bodmi,ktoré sú na obrázku 3 znázornené červenou (špička prsta) a žltou (kĺb medzi prvým a druhým článkom prsta).



Obr. 3. Model ruky v zornom poli senzora[3].

Ako je spomenuté vyššie, v ideálnom prípade by sa priamky $p_1, ..., p_n$ pretli v hľadanom bode. Prakticky dosiahnuť takúto situáciu nie je možné. Do úvahy treba zobrať nepresné namierenie prsta používateľa, ako aj skutočnosť, že prst ma určitú hrúbku, ktorá spôsobuje odchýlky. Hľadaný prienik z týchto dôvodov vo väčšine prípadov neexistuje. Na odhad súradníc bodu v trojrozmernej sústave využívame regresnú analýzu.

2.3.1 Regresná analýza Regresná analýza je štatistická metóda, ktorá skúma závislosti veličín. Pozostáva z troch krokov, a to definovanie chybovej funkcie, funkcie globálnej chyby a nájdenie minima funkcie globálnej chyby. Pre zaznamenané priamky hľadáme bod, ktorého vzdialenosť od týchto troch priamok je najmenšia.

[3] Zdroj: `https://media.springernature.com/original/springer-static/image/chp\%3A10.1007\%2F978-3-319-26410-3_22/MediaObjects/385589_1_En_22_Fig1_HTML.gif`

Definovanie chybovej funkcie

Prvým krokom regresnej analýzy je definovanie chybovej funkcie. V našom prípade je chybou vzdialenosť hľadaného bodu Xod priamky p_i. Každú priamku p_i máme definovanú bodom $A_i = [a_{i1}, a_{i2}, a_{i3}]$ a vektorom $v_i = B_i - A_i = [v_{i1}, v_{i2}, v_{i3}]$. Kolmú vzdialenosť bodu X od priamky p_i vypočítame ako podiel $\frac{|(X-A_i)\times v_i|}{|v_i|}$ podľa [1]. Pre každé $i < n$, kde n je počet priamok, potom definujeme funkciu f_i nasledujúcim vzťahom:

$$f_i(x_1, x_2, x_3) = \frac{\sqrt{((x_2-a_{i2})v_{i3}-(x_3-a_{i3})v_{i2})^2+((x_3-a_{i3})v_{i1}-(x_1-a_{i1})v_{i3})^2+((x_1-a_{i1})v_{i2}-(x_2-a_{i2})v_{i1})^2}}{\sqrt{v_{i1}^2+v_{i3}^2+v_{i3}^2}}. \tag{2}$$

Definovanie globálnej chyby

Funkciu pre výpočet globálnej chyby definujeme ako súčet štvorcov všetkých chýb:

$$f(x_1, x_2, x_3) = \sum_{i=1}^{n} f_i(x_1, x_2, x_3)^2, \tag{3}$$

potom funkcia pre výpočet globálnej chyby vyzerá nasledovne:

$$f(x_1, x_2, x_3) = \sum_{i=1}^{n} \frac{((x_2-a_{i2})v_{i3}-(x_3-a_{i3})v_{i2})^2+((x_3-a_{i3})v_{i1}-(x_1-a_{i1})v_{i3})^2+((x_1-a_{i1})v_{i2}-(x_2-a_{i2})v_{i1})^2}{v_{i1}^2+v_{i3}^2+v_{i3}^2}. \tag{4}$$

Minimum funkcie globálnej chyby

Na nájdenie minima funkcie globálnej chyby potrebujeme parciálne derivácie funkcie f podľa premenných x_1, x_2 a x_3.

$$\frac{\partial f}{\partial x_1} = \sum_{i=1}^{n} \frac{2v_{i3}^2 + 2v_{i2}^2}{v_{i1}^2 + v_{i2}^2 + v_{i3}^2} x_1 + \sum_{i=1}^{n} \frac{-2v_{i1}v_{i2}}{v_{i1}^2 + v_{i2}^2 + v_{i3}^2} x_2 + \sum_{i=1}^{n} \frac{-2v_{i1}v_{i3}}{v_{i1}^2 + v_{i2}^2 + v_{i3}^2} x_3 + \sum_{i=1}^{n} \frac{2v_{i1}(a_{i2}v_{i2} + a_{i3}v_{i3}) - 2a_{i1}(v_{i2}^2 + v_{i3}^2)}{v_{i1}^2 + v_{i2}^2 + v_{i3}^2} \tag{5}$$

$$\frac{\partial f}{\partial x_2} = \sum_{i=1}^{n} \frac{-2v_{i1}v_{i2}}{v_{i1}^2+v_{i2}^2+v_{i3}^2}x_1 + \sum_{i=1}^{n} \frac{2v_{i1}^2+2v_{i3}^2}{v_{i1}^2+v_{i2}^2+v_{i3}^2}x_2 + \sum_{i=1}^{n} \frac{-2v_{i2}v_{i3}}{v_{i1}^2+v_{i2}^2+v_{i3}^2}x_3 + \sum_{i=1}^{n} \frac{2v_{i2}(a_{i1}v_{i1}+a_{i3}v_{i3}) - 2a_{i2}(v_{i1}^2+v_{i3}^2)}{v_{i1}^2+v_{i2}^2+v_{i3}^2} \tag{6}$$

$$\frac{\partial f}{\partial x_3} = \sum_{i=1}^{n} \frac{-2v_{i1}v_{i3}}{v_{i1}^2+v_{i2}^2+v_{i3}^2}x_1 + \sum_{i=1}^{n} \frac{-2v_{i2}v_{i3}}{v_{i1}^2+v_{i2}^2+v_{i3}^2}x_2 + \sum_{i=1}^{n} \frac{2v_{i1}^2+2v_{i2}^2}{v_{i1}^2+v_{i2}^2+v_{i3}^2}x_3 + \sum_{i=1}^{n} \frac{2v_{i3}(a_{i1}v_{i1}+a_{i2}v_{i2}) - 2a_{i3}(v_{i1}^2+v_{i2}^2)}{v_{i1}^2+v_{i2}^2+v_{i3}^2} \tag{7}$$

Položíme parciálne derivácie rovné 0. Dostávame maticovú rovnicu tvaru $AX = B$:

$$\begin{pmatrix} \sum_{i=1}^{n} \frac{2v_{i2}^2+2v_{i3}^2}{v_{i1}^2+v_{i2}^2+v_{i3}^2} & \sum_{i=1}^{n} \frac{-2v_{i1}v_{i2}}{v_{i1}^2+v_{i2}^2+v_{i3}^2} & \sum_{i=1}^{n} \frac{-2v_{i1}v_{i3}}{v_{i1}^2+v_{i2}^2+v_{i3}^2} \\ \sum_{i=1}^{n} \frac{-2v_{i1}v_{i2}}{v_{i1}^2+v_{i2}^2+v_{i3}^2} & \sum_{i=1}^{n} \frac{2v_{i1}^2+2v_{i3}^2}{v_{i1}^2+v_{i2}^2+v_{i3}^2} & \sum_{i=1}^{n} \frac{-2v_{i2}v_{i3}}{v_{i1}^2+v_{i2}^2+v_{i3}^2} \\ \sum_{i=1}^{n} \frac{-2v_{i1}v_{i3}}{v_{i1}^2+v_{i2}^2+v_{i3}^2} & \sum_{i=1}^{n} \frac{-2v_{i2}v_{i3}}{v_{i1}^2+v_{i2}^2+v_{i3}^2} & \sum_{i=1}^{n} \frac{2v_{i1}^2+2v_{i2}^2}{v_{i1}^2+v_{i2}^2+v_{i3}^2} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^{n} \frac{2a_{i1}(v_{i2}^2+v_{i3}^2)-2v_{i1}(a_{i2}v_{i2}+a_{i3}v_{i3})}{v_{i1}^2+v_{i2}^2+v_{i3}^2} \\ \sum_{i=1}^{n} \frac{2a_{i2}(v_{i1}^2+v_{i3}^2)-2v_{i2}(a_{i1}v_{i1}+a_{i3}v_{i3})}{v_{i1}^2+v_{i2}^2+v_{i3}^2} \\ \sum_{i=1}^{n} \frac{2a_{i3}(v_{i1}^2+v_{i2}^2)-2v_{i3}(a_{i1}v_{i1}+a_{i2}v_{i2})}{v_{i1}^2+v_{i2}^2+v_{i3}^2} \end{pmatrix} \tag{8}$$

Výpočtom pomocou determinantov dostávame súradnice bodu $X = [x_1, x_2, x_3]$.

Touto metódou dostaneme súradnice troch bodov obrazovky, ktoré potrebujeme na popísanie celej plochy obrazovky.

2.4 Určenie pozície kurzora na obrazovke

Po kalibrácii už máme popísanú plochu obrazovky parametrickým vyjadrením (1).

Keďže v jednej chvíli sa v zornom poli senzora môže nachádzať viac prstov, môžeme si zvoliť spôsob, ktorým algoritmus jednoznačne určí, ktorý prst bude ovládať pohyb kurzora na obrazovke. Prirodzene vyberieme prst, ktorý sa nachádza v najmenšej vzdialenosti od obrazovky alebo vyberieme konkrétny prst (napr. ukazovák pravej ruky).

Rovnako ako v sekcii 2.1, v ktorej popisujeme kalibráciu, opäť vezmeme súradnice špičky prsta a kĺbu medzi prvým a druhým článkom prsta (znázornených na obrázku 3 červenou a žltou farbou). Označme ich A a B. Preložme nimi

priamku, ktorú popisuje nasledujúca rovnica:

$$X = A + m * w, \tag{9}$$

kde w je vektor $B - A$ a m je ľubovoľné reálne číslo.

Kurzor premiestnime na miesto, kde priamka pretne plochu obrazovky. Na výpočet prieniku dáme do rovnosti vyjadrenie plochy obrazovky a vyjadrenie priamky vedenej prstom:

$$P_1 + k * u + l * v = A + m * w, \tag{10}$$

z čoho dostávame sústavu rovníc:

$$\begin{aligned} p_1 + k * u_1 + l * v_1 &= a_1 + m * w_1 \\ p_2 + k * u_2 + l * v_2 &= a_2 + m * w_2 \\ p_3 + k * u_3 + l * v_3 &= a_3 + m * w_3 \end{aligned} \tag{11}$$

Vyriešením sústavy dostávame parametre k, l a m. V prípade, že získané parametre k, l sú z intervalu $< 0, 1 >$, bod prieniku v zornom poli senzora získame dosadením získaných parametrov do 3D vyjadrenia plochy obrazovky, pozíciu kurzora na obrazovke dostávame dosadením parametrov k, l do 2D vyjadrenia plochy.

Výpočet prieniku priamky a plochy obrazovky prebieha v cykle počas celej doby používania nástroja na ovládanie počítača bez dotyku. Používateľovi umožňuje ovládať pohyb kurzora na obrazovke pohybom ruky. Na plne funkčné ovládanie počítača je potrebné nejakým spôsobom nahradiť funkcionalitu, ktorú zväčša poskytujú tlačidlá myši, touchpad alebo dotyková obrazovka. Nahradili sme ju rozpoznávaním gest, ktorému sa venujeme v nasledujúcej sekcii.

2.5 Ovládanie počítača

Ovládacie prvky môžeme rozdeliť do dvoch skupín. Jednou z nich sú udalosti, ktoré je možné jednoducho implementovať napríklad použitím podmienky. Druhou skupinou sú gestá, ktorými rozumieme rôzne sekvencie pohybu kurzora. Je potrebné pre nich implementovať rozpoznávanie. Obe skupiny a ich funkcie popíšeme v tejto sekcii.

Najdôležitejšou funkcionalitou, ktorú potrebujeme emulovať, je kliknutie a posúvanie obsahu, známe pod pojmom scrollovanie.

Kliknutie

Pri emulácii kliknutia myši sme značne obmedzení tým, že keďže chceme kliknúť na konkrétne miesto, po ukázaní prstom naň už musí prst ostať v takej polohe, aby sme kurzorom nepohli. Ďalším obmedzením je to, že spôsob emulácie kliknutia musí byť pre používateľa prirodzený, v opačnom prípade by bolo ovládanie počítača nepohodlné.

Jednou z možností, ktorá v značnej miere spĺňa obe kritéria, je dotyk koncov palca a prostredníka rovnakej ruky, ktorej ukazovákom ovládame pohyb kurzora. Takéto gesto, resp. udalosť sa dá jednoducho popísať aj implementovať ako zmenšenie vzdialenosti medzi končekmi dvoch prstov pod určitú prahovú hodnotu. Pri jeho testovaní sme však nedosahovali požadované výsledky. Práve pri ňom najviac zavážili odchýlky pri určovaní súradníc senzorom. Vzdialenosť prstov aj pri ich konštantnom dotyku kolísala natoľko, že sme možnosť využitia opísanej možnosti museli vylúčiť.

Ďalšou možnosťou, ktorú sme testovali, je rýchle priblíženie ukazujúceho prsta v smere k ploche obrazovky. Rovnako neuspokojivé výsledky sme dostali kvôli tomu, že nie je jednoduché urobiť rýchly pohyb a zároveň ukazovať na presne dané miesto - kurzor sa pri pohybe vpred pohybuje aj po obrazovke.

Problém rýchleho pohybu však vieme odstrániť. Ostaneme pri rovnakom pohybe v smere k ploche obrazovky, avšak signálom pre gesto bude dosiahnutie určitej vzdialenosti prsta od obrazovky monitora. Takáto pomyselná dotyková plocha sa nachádza vo vzdialenosti niekoľkých centimetrov (5-10 cm) od reálnej polohy obrazovky a je s ňou rovnobežná.

Posúvanie obsahu

Vzhľadom na to, že knižnica, ktorá prevádza snímky na 3D súradnice kĺbov rúk, dokáže sama rozpoznať 4 gestá (*swipe, circle, screen-tap, key-tap*), rozhodli sme sa 2 z nich ich využiť. Jedným je krúženie prstom v smere alebo proti smeru hodinových ručičiek, ktorým ovládame posúvanie obsahu nahor alebo nadol. Rovnaké gesto využívajú na posúvanie obsahu aj iné existujúce nástroje na ovládanie počítača.

Druhým gestom je swipe, ktoré emuluje stlačenie šípok vpravo, resp. vľavo.

Pre ďalšie gestá sme implementovali rozpoznávanie, ktoré je popísané v nasledujúcej sekcii.

2.6 Gestá kurzora

Kurzorovými gestami rozumieme sekvencie rôznych smerov, ktorými sa kurzor pohybuje v čase. Príkladom môže byť pohyb v tvare štvorca, kružnice alebo akéhokoľvek iného tvaru. Tie majú mať účel podobný klávesovým skratkám - či budú otvárať aplikácie, alebo budú spĺňať iný účel, závisí od preferencie používateľa.

Rozpoznávanie gest patrí medzi klasifikačné úlohy, ktoré pozostávajú z dvoch fáz - fáza učenia a klasifikácie. Rozhodli sme sa pre jednoduchý klasifikátor k-najbližších susedov. V jeho fáze učenia vytvárame tréningovú množinu, vo fáze klasifikácie porovnávame zhodu vykonaného gesta s tréningovou množinou. Obe fázy si popíšeme v nasledujúcich sekciách.

2.6.1 Vytvorenie tréningovej množiny Vo fáze vytvorenia tréningovej množiny vykonávame každé gesto, ktoré chceme rozpoznávať, niekoľkokrát za

sebou. Zvolili sme počet opakovaní 5. Pri každom z opakovaní spočítame vektor príznakov vykonaného pohybu, predstavujúceho gesto a každý z týchto vektorov príznakov uložíme.

Predovšetkým treba určiť, čo v prípade gesta považujeme za príznak, aby sme neskôr dokázali jednotlivé vektory príznakov porovnať.

Inšpirovali sme sa dokumentáciou knižnice Qt[8], ktorá pri rozpoznávaní gest myši ako vektor príznakov využíva postupnosť za sebou nasledujúcich smerov, ktorými sa kurzor pohybuje. Narozdiel od knižnice Qt, kvôli rozmanitosti gest sme sa rozhodli namiesto limitácie 4 smerov (nahor, nadol, vpravo, vľavo) využiť limitáciu 8 smerov, a to každých 45 stupňov, počnúc 0. Gestá sa pri menšom počte smerov jednoduchšie rozpoznávajú, ale ich rozmanitosť je značne obmedzená. Pri vyššom počte smerov by úspešnosť rozpoznávania mohla výrazne klesnúť a čas rozpoznania by sa naopak zvýšil.

Preto vektor príznakov gesta definujeme ako postupnosť $d_1, d_2, ...d_n$, kde pre každé i z 1,2,...,n, d_i je jeden z 8 smerov (nahor, nadol, vpravo, vľavo, vľavo-hore, vpravo-hore, vľavo-dole, vpravo-dole). V nasledujúcej sekcii popíšeme spočítanie vektora príznakov vykonaného pohybu.

Extrakcia vektora príznakov

Najskôr potrebujeme určiť spôsob, ktorým ohraničíme vykonávanie gesta. Program musí jednoznačne vyhodnotiť, kedy používateľ začal gesto vykonávať a kedy jeho vykonávanie ukončil a v tomto rozmedzí zbierať pozície kurzora na obrazovke. Jednoduché riešenie spočíva v tom, že počas krátkeho časového intervalu (1-2s) ukazujeme prstom na nejaké miesto bez pohybu. Keďže nie je možné, aby sme držali ruku v úplne nehybnej polohe, postačí, ak sa v danom čase kurzor nepohne mimo oblasti kruhu s malým polomerom. Pri testovaní sme použili polomer 10 px, pričom takéto určenie začiatku a konca gesta je jednoduché a prirodzené pre používateľa a zároveň jednoducho rozpoznateľné programom.

Napriek tomu, že tento signál začiatku alebo konca gesta môže program počas ovládania rozpoznať aj v prípade, že používateľ nemal v pláne vykonávať gesto, chod programu to neovplyvní. Rovnako ako ohraničujeme zber dát začiatkom a koncom gesta, definujeme aj maximálnu dĺžku gesta. Vykonanie gesta by nemalo trvať dlhšie ako 3-5 sekúnd. Ak program rozpoznal signál začiatku gesta a následne začal uchovávať dáta o pohybe kurzora, ale po určenej maximálnej dĺžke gesta nerozpozná signál jeho ukončenia, dáta sa vymažú a program opäť čaká na signál začiatku gesta.

Po úspešnom rozpoznaní začiatku a konca gesta v dostatočne malom časovom rozmedzí máme zaznamenanú postupnosť pozícií kurzora na obrazovke. Pri zbere 20 pozícií za sekundu získame postupnosť s maximálnou dĺžkou 100. Na postupnosť pozícií kurzora v ďalšom kroku aplikujeme filtrovanie pohybu kurzora a následný prevod zoznamu pozícií na zoznam smerov.

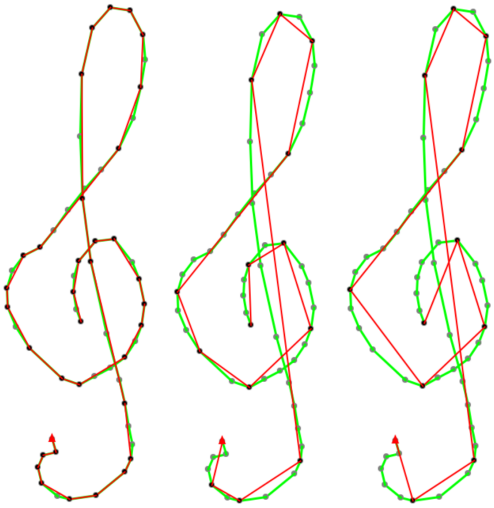
Filtrovanie pohybu kurzora

V procese vykonávania gesta sa nie vždy podarí opísať tvar dostatočne hladko, často sa môže stať, že na krátkom úseku nechtiac zmeníme smer. Tieto odchýlky, ako aj fakt, že pri pohybe jedným smerom sa zozbiera viacero pozícií, vieme odstrániť algoritmom zjednodušovania priamok. Na zjednodušenie pohybu kurzora využívame rekurzívny algoritmus Douglas-Peucker[5], ktorý funguje nasledovne:

Vstup : Zoznam bodov, epsilon
Výstup : Zjednodušený zoznam bodov
Kroky :

- 1. Nájdi v zozname bod, ktorý sa nachádza od priamky, vedenej prvým a posledným bodom zoznamu, najďalej zo všetkých.
- 2a. Ak je vzdialenosť najvzdialenejšieho bodu väčšia ako epsilon, rekurzívne zjednoduš zoznam bodov od začiatku po najvzdialenejší bod a zoznam bodov od najvzdialenejšieho bodu po koniec zoznamu. Vráť zoznam vytvorený spojením dvoch zjednodušených zoznamov
- 2b. Inak do výsledku pridaj iba prvý a posledný bod.

Algoritmus sme testovali s rôznymi hodnotami pre vstupný parameter epsilon. Na obrázku 4 je zobrazená jednoduchá vizualizácia výsledku algoritmu pre hodnoty parametra $\varepsilon = 5, 20, 40$. Zelenou farbou je vyznačená postupnosť bodov pred aplikovaním algoritmu, červenou po jeho aplikácii.



Obr. 4. Porovnanie algoritmu Douglas-Peucker s hodnotami pre parameter ε 5(vľavo), 20(v strede) a 40(vpravo).

Pozorovali sme, že pre $\varepsilon = 5$ bol výsledok takmer rovnaký ako pred aplikáciou algoritmu, nepatrná zmena smeru v spodnej časti obrázka vľavo nebola algoritmom vyhladená.

Naopak pre $\varepsilon = 40$ už boli niektoré smery príliš zmenené, kreslenie gesta začína v strede obrázka ťahaním smerom nahor, avšak po zjednodušení by sa prvý smer mohol vyhodnotiť ako vpravo-hore, čo je vidieť na obrázku vpravo.

Najuspokojivejšie výsledky sme dosahovali pri hodnote $\varepsilon = 20$, na obrázku v strede je tiež vidieť, že z pôvodných vyše 50 bodov sme po zjednodušení dostali počet bodov 15. Preto sme ďalej využívali algoritmus Douglas-Peucker s hodnotou $\varepsilon = 20$.

Po aplikovaní algoritmu zjednodušovania priamok nasleduje prevod postupnosti bodov na postupnosť smerov.

Postupnosť smerov

Výsledkom algoritmu Douglas-Peucker je zjednodušený zoznam súradníc bodov $[x_1, y_1], ..., [x_m, y_m]$, kde m je počet bodov po zjednodušení.

Postupnosť smerov vytvoríme tak, že pre každú dvojicu za sebou idúcich pozícií určíme smernicu k priamky, ktorá nimi prechádza, teda pre každú dvojicu $[x_{i-1}, y_{i-1}], [x_i, y_i]$, pre $i = 2, ..., m$ vypočítame smernicu vzorcom:

$$k = \frac{\Delta y}{\Delta x} = \frac{y_i - y_{i-1}}{x_i - x_{i-1}}. \tag{12}$$

Na základe porovnania x-ových a y-ových súradníc dvoch bodov a hodnoty smernice k vieme pre každú dvojicu určiť jeden z 8 vybraných smerov.

Týmto spôsobom dostávame postupnosť smerov. Môže však nastať situácia, keď dostaneme v zozname viackrát rovnaký smer za sebou. Tento problém odstránime už počas vytvárania zoznamu smerov, a to tak, že do zoznamu pridávame vypočítaný smer len v prípade, ak je rôzny od naposledy pridaného.

Teraz už máme extrahovanú postupnosť smerov, ktorá predstavuje vektor príznakov gesta. Pre každé gesto zaznamenáme vykonaný pohyb viackrát a uložíme jeho vektorové príznaky - deskriptory. Po vytvorení tréningovej množiny už môžeme vykonané gesto klasifikovať.

2.6.2 Klasifikácia gesta Princíp klasifikácie je jednoduchý - opäť spočítame vektor príznakov vykonaného gesta a ten porovnáme so všetkými vektormi príznakov uložených gest. Po porovnaní s každou postupnosťou vyberáme k (v našom prípade 5) takých, s ktorými má najväčšiu zhodu. Ak je nadpolovičná množina vektorov príznakov priradená rovnakému uloženému gestu, môžeme povedať, že bolo dané gesto rozpoznané. V opačnom prípade sa gesto klasifikuje ako neznáme. Na určenie podobnosti dvoch vektorov príznakov využívame výpočet Levenshteinovej vzdialenosti.

Levenshteinova vzdialenosť

Výpočet Levenshteinovej vzdialenosti[7], inak známy ako algoritmus *edit distance*, má vopred určenú tzv. cenu troch operácií - vymazania, vloženia a zámeny prvku, štandardne sú tieto hodnoty 1. Levenshteinova vzdialenosť určuje najmenšiu možnú celkovú cenu všetkých operácií, ktoré potrebujeme vykonať, aby sme z jednej sekvencie prvkov, v našom prípade smerov, urobili druhú. Čím je vzdialenosť väčšia, tým sú si sekvencie navzájom menej podobné. Pri porovnávaní dvoch vektorov určíme maximálnu vzdialenosť, ktorú môžu dva príznaky dosiahnuť, aby sme o nich mohli povedať, že sú podobné. Maximálna vzdialenosť sa pri rôznych gestách líši v závislosti od dĺžky vektora príznakov.

3 Záver

Doposiaľ sme sa zaoberali výpočtom pozície kurzora na obrazovke z polohy ruky v zornom poli senzora, čo nám umožňuje jeho ovládanie. Pri precíznej kalibrácii vieme dosiahnuť uspokojivú presnosť za predpokladu, že ukazujúca ruka je v takej polohe, v ktorej vie senzor s dostatočnou presnosťou určiť pozície jej kĺbov. Implementovali sme bežnú funkcionalitu ovládania počítača, akou je klikanie a posúvanie obsahu. Venovali sme sa rozpoznávaniu gest, ktoré funkcionalitu ovládania rozširuje. Úspešne sa nám podarilo rozpoznávať základné útvary, ako sú štvorec, či kruh. Ďalej sa chceme zaoberať nastavením najvhodnejších parametrov a prahových hodnôt pri celom procese rozpoznávania ako aj ich závislosťou na rozlíšení a veľkosti monitora.

Poďakovanie

Týmto sa chcem poďakovať vedúcemu mojej bakalárskej práce RNDr. Františkovi Galčíkovi, PhD., ako aj konzultantovi RNDr. Matejovi Nikorovičovi, za pomoc s výberom témy bakalárskej práce, ochotu a cenné rady a pripomienky pri jej vypracovaní.

Literatúra

1. HARTLEY, Richard; ZISSERMAN, Andrew. Multiple view geometry in computer vision. Cambridge university press, 2003.
2. Leap Motion, Inc. Leap motion controller. `https://www.leapmotion.com`, 2015.
3. Tobii Technology AB. Tobii EyeX. `https://tobiigaming.com`, 2016.
4. Microsoft Corporation. Microsoft Kinect One. `https://developer.microsoft.com/en-us/windows/kinect`, 2013.
5. DOUGLAS, David H.; PEUCKER, Thomas K. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. Cartographica: The International Journal for Geographic Information and Geovisualization, 1973, 10.2: 112-122.
6. WEICHERT, Frank, et al. Analysis of the accuracy and robustness of the leap motion controller. Sensors, 2013, 13.5: 6380-6393.

7. LEVENSHTEIN, Vladimir I. Binary codes capable of correcting deletions, insertions, and reversals. In: Soviet physics doklady. 1966. p. 707-710.
8. THELIN, Johan. Recognizing mouse gestures [online]. Dostupné na internete: `http://doc.trolltech.com/qq/qq18-mousegestures.html` 2009 [citované 18.4.2018].