UNIVERZA V LJUBLJANI

FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Elvis Žlender

SLEDENJE LASERSKEMU KAZALNIKU ZA UPRAVLJANJE Z RAČUNALNIKOM

DIPLOMSKO DELO

NA UNIVERZITETNEM ŠTUDIJU

Mentor: doc. dr. Matija Marolt

Ljubljana, 2010

Univerza
v Ljubljani

Fakulteta za računalništvo
in informatiko

Tržaška 25
1000 Ljubljana, Slovenija
telefon: 01 476 84 11
faks: 01 426 46 47
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 01702/2010
Datum: 01.09.2010

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Kandidat: **ELVIS ŽLENDER**

Naslov: **SLEDENJE LASERSKEMU KAZALNIKU ZA UPRAVLJANJE Z RAČUNALNIKOM**

LASER POINTER TRACKING FOR HUMAN-COMPUTER INTERACTION

Vrsta naloge: Diplomsko delo univerzitetnega študija

Tematika naloge:

V okviru diplomske naloge izdelajte program, ki bo z uporabo kamere omogočal sledenje laserskemu kazalniku in s tem simuliral delovanje računalniške miške. Namenjen naj bo predvsem predstavitvam s pomočjo projektorja. Omogoča naj avtomatsko kalibracijo med kamero in projicirano sliko, izvajanje akcij, ki jih omogoča običajna računalniška miška in izvajanje dodatnih akcij s pomočjo prepoznavanja kretenj.

Mentor:

doc. dr. Matija Marolt

Fakulteta za računalništvo in informatiko
2
UNIVERZA V LJUBLJANI

Dekan:

prof. dr. Nikolaj Zimic

I Z J A V A O A V T O R S T V U

diplomskega dela

Spodaj podpisani **Elvis Žlender**,

z vpisno številko **63020187**,

sem avtor diplomskega dela z naslovom:

Sledenje laserskemu kazalniku za upravljanje z računalnikom.

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom

  doc. dr. Matije Marolta
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela
- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki »Dela FRI«.

V Ljubljani, 5.11.2010 Podpis avtorja:

Zahvala

Najprej bi se rad zahvalil svojim staršem in sestri Barbari za moralno in finančno podporo skozi vsa moja študijska leta.

Zahvaljujem se tudi mentorju, doc. dr. Matiji Maroltu, za strokovno vodstvo in pomoč ter vsem ostalim, ki so kakorkoli pomagali pri diplomski nalogi.

Kazalo

Povzetek ... 1
Abstract ... 2
1 Uvod ... 3
2 Kalibracija in homografija ... 6
2.1 Kalibracija ... 6
2.1.1 Zunanji parametri ... 6
2.1.2 Notranji parametri ... 6
2.1.3 Končni model kamere ... 7
2.2 Homografija med kamero in projektorjem ... 8
2.3 Houghova transformacija ... 9
3 Detekcija laserskega žarka ... 11
3.1 Metoda segmentacije ... 11
3.2 Metoda ločevanja ozadja ... 13
3.3 Metoda povprečenja ... 14
4 Upravljanje miške z laserskim žarkom ... 16
4.1 Premik miškinega kazalca ... 16
4.2 Levi klik ... 16
4.3 Dvojni levi klik ... 17
4.4 Vleci in spusti ... 17
4.5 Desni klik ... 18
4.6 Pomik s sledilnim koleščkom ... 18
5 Prepoznavanje kretenj ... 19
5.1 Prepoznavanje kretenj miške ... 19
5.2 Kretnje, ki vključujejo periferni del projekcijskega zaslona ... 21
5.3 Problemi pri interakciji s pomočjo kretenj ... 23
6 Implementacija vmesnika in njegova uporaba ... 25
6.1 Predpostavke ... 25
6.2 Avtomatsko določanje transformacijske matrike ... 26
6.2.1 Glajenje šuma ... 26
6.2.2 Upragovanje ... 26
6.2.3 Detekcija robov s Houghovo transformacijo ... 28

6.2.4 Detekcija vogalov ... 28
6.2.5 Homografija ... 31
6.3 Detekcija laserske pike ... 31
6.4 Glavna zanka ... 32
6.5 Uporaba vmesnika brez projektorja ... 34
7 Rezultati ... 35
7.1 Oprema ... 35
7.1.1 Strojna oprema ... 35
7.1.2 Programska oprema ... 36
7.2 Parametri vmesnika ... 37
7.2.1 Prag za upragovanje ... 37
7.2.2 Parametri Houghove transformacije ... 38
7.2.3 Parametri detekcije laserske pike ... 38
7.2.4 Parametri glavne zanke ... 38
7.3 Kriteriji ... 39
7.3.1 Zanesljivost ... 39
7.3.2 Latenca ... 40
7.3.3 Natančnost ... 40
8 Zaključek ... 41
Kazalo slik ... 43
Kazalo tabel ... 43
Viri ... 44

Povzetek

Običajne predstavitve se izvajajo s pomočjo računalnika in nanj priklopljenega projektorja. Predavatelj za nadzor nad predstavitvijo uporablja vhodne naprave računalnika (miška ali tipkovnica). S tem se lahko prekinja interakcija z občinstvom. V diplomski nalogi podam eno izmed možnih rešitev tega problema v obliki vmesnika za uporabo laserskega kazalnika, s katerim upravljamo računalniško miško.

Kot dodatno opremo sem poleg računalnika in projektorja uporabil laserski kazalnik, s katerim upravljamo miško, in na računalnik priklopljeno nekalibrirano kamero, ki zajema celotno projekcijsko sliko. Kalibracija se izvede s pomočjo homografije med projekcijsko sliko in sliko kamere. S projektorjem aktivno obvladujemo okolje, tako da na projekcijsko površino projiciramo značilne točke. Kamera s pripadajočimi implementiranimi metodami procesiranja slik omogoča sledenje miškinega kazalca laserskemu žarku. S pomočjo vmesnika lahko z laserskim kazalnikom izvajamo vse akcije v operacijskem sistemu Microsoft Windows, ki jih omogoča običajna računalniška miška s tremi gumbi: levi klik, dvojni levi klik, vleci in spusti, desni klik ter drsenje s sledilnim koleščkom. Zadnji dve akciji sta implementirani z uporabo prepoznavanja kretenj. Ta del vmesnika omogoča intuitivno nadzorovanje predstavitve znotraj aplikacije Microsoft PowerPoint in nadzorovanje aplikacije Windows Media Player, saj lahko s preprosto kretnjo (npr. poteg preko desnega roba projekcijskega zaslona) izvedemo marsikatero akcijo (npr. prehod na naslednjo prosojnico). Možna je nadgradnja vmesnika z dodatnimi kretnjami za upravljanje še katerih drugih aplikacij.

Ključne besede: sistem projektor-kamera, laserski kazalnik, kalibracija, homografija, prepoznavanje kretenj

Abstract

To control standard presentation systems consisting of a laptop connected to a projector the presenter uses computer input devices such as keyboard or a mouse. The speaker is forced to interact with the computer rather than the audience. This thesis introduces one of the possible solutions in the form of an interface that uses laser pointer to control the computer mouse.

As additional equipment I used a laser pointer to control the mouse and an uncalibrated camera to capture the projection image. The projector-camera system calibrates itself by exploiting the homography between the projected image and the camera image. The projector actively manipulates the enviroment by projecting feature points onto the projection surface. With the implementation of some image processing methods camera enables the mouse pointer to follow the laser beam. The implemented interface allows us to perform all the elementary three-button mouse actions inside the Microsoft Windows operating system with the laser pointer (i.e. left click, double left click, drag and drop, right click and scroll). Right click and scroll are implemented using gesture recognition methods. This part of interface gives us an intuitive control over Microsoft PowerPoint presentation and Windows Media Player application. With simple gestures (e.g. laser beam motion across the right side of the projection screen) we can execute many different commands (e.g. move to the next slide). The interface can be upgraded with additional gestures to control additional applications.

Key words: projector-camera system, laser pointer, calibration, homography, gesture recognition

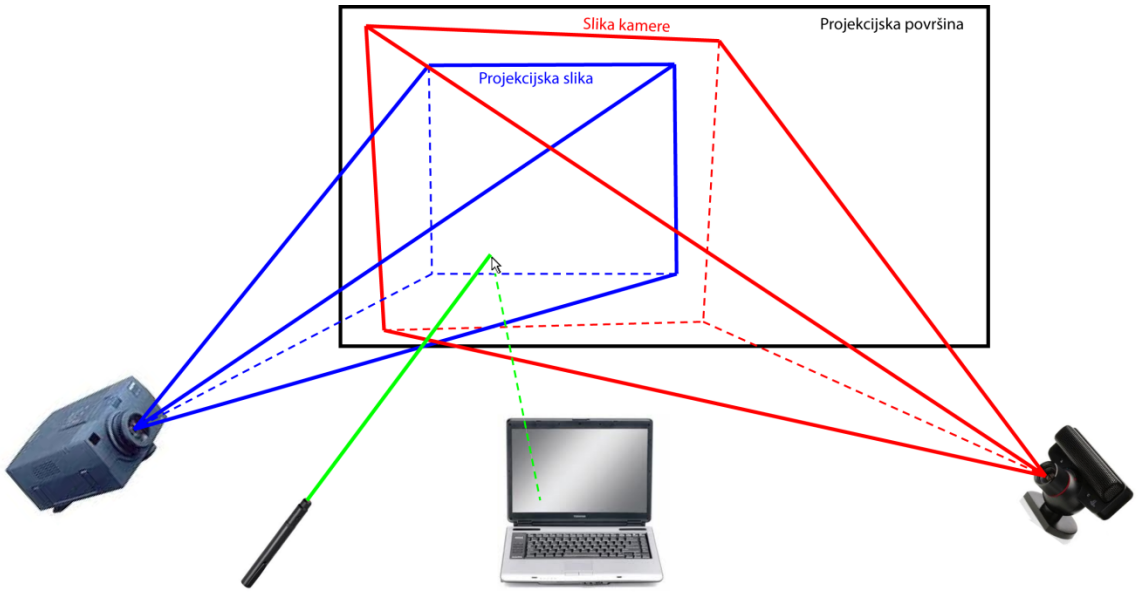
1 Uvod

Projektorji so danes zelo popularni in dostopni. Njihova uporaba se je tako doma kot v službi in drugod zelo povečala. Interakcija s tako napravo oz. s projekcijskim zaslonom se razlikuje od tiste z namiznim računalnikom. Miška in tipkovnica namreč potrebujeta (vodo)ravno površino za upravljanje. Ob zaenkrat še dokaj neučinkovitih sistemih prepoznavanja kretenj in govora ter precej dragih velikih zaslonih na dotik je interakcija omejena na laserske kazalnike in druge brezžične ročne naprave.

Uporaba laserskega kazalnika je lahko pri predstavitvah zelo uporaben pripomoček. Predstavljalcu omogoča, da hitro pokaže na določeno stvar na oddaljeni in veliki projekcijski površini. Sam kazalnik ne omogoča nadzora predstavitve, torej pomikanja med prosojnicami. Za cilj diplomske naloge sem si zadal izdelati vmesnik, ki omogoča uporabo laserskega kazalnika kot vhodne naprave za interakcijo med človekom in računalnikom. Poleg samega nadzora predstavitve bo vmesnik zmožen upravljati tudi nekatere druge aplikacije. Njegova uporaba bo zelo podobna uporabi računalniške miške.

Laserski kazalnik je kot vhodna naprava omejen, saj ima le gumb za vklop in izklop žarka. Za vse akcije, ki jih bo omogočal, je potrebna programska implementacija. Vmesnik mora znati prepoznati projekcijski zaslon, na njem detektirati lasersko piko in kretnje te pike pretvoriti v pripadajoče ukaze oz. akcije. Če želimo kazalnik uporabljati namesto miške, moramo v vmesnik implementirati levi klik, dvojni levi klik, desni klik, akcijo vleci in spusti ter (neobvezno) pomik s sledilnim koleščkom.

Osnovna oprema, potrebna za uporabo tega vmesnika, je sestavljena iz projektorja, ki projicira zaslon na projekcijsko površino, kamere, s katero zajemamo projekcijsko sliko, laserskega kazalnika kot vhodne naprave ter računalnika, ki procesira podatke in izvaja potrebne ukaze. V tej sestavi je možna uporaba kazalnika namesto miške (slika *1*).



Slika 1: Sistem za uporabo laserskega kazalnika namesto miške

Poleg tega lahko vmesnik uporabljamo za nadzor aplikacij, pri katerih ne potrebujemo vizualne povratne informacije (npr. avdio predvajalnik). V takem primeru je projektor odveč. Vsa ostala oprema pa je nujna za nadzor takšne aplikacije.

Na to temo obstaja že kar veliko literature ([2, 4, 6]), vendar nikjer ni natančnega opisa same implementacije. Vsi omenjeni članki vsebujejo opis kalibracije projektorja in kamere ter opis detekcije laserske pike. Poleg tega vsebuje aplikacija, opisana v [6], še avtomatski popravek projekcijske slike, kadar ni poravnana s projekcijsko površino, ter virtualne gumbe za prehod med prosojnicami in aktivacijo menijev. Dan R. Olsen Jr. in Travis Nielsen [4] sta se podrobneje posvetila interaktivnim tehnikam za uporabo laserskega žarka, medtem ko članek [2] podrobneje predstavi same rezultate ter kriterije, iz katerih izhajajo rezultati.

Moja implementacija vmesnika je zasnovana na zgoraj omenjeni literaturi, za procesiranje slik pa uporablja knjižnico OpenCV. Poleg kalibracije projektorja in kamere ter detekcije laserske pike sem se odločil implementirati še prepoznavanje kretenj miškinega kazalca, ki olajša uporabo določenih aplikacij oz. izvajanje določenih akcij. Uporabnost vmesnika je v največji meri odvisna od zanesljivosti uporabe v realnem času. Ozko grlo procesiranja se tu nanaša na

pogosto procesiranje slik oz. detekcijo laserskega žarka. Za primerjavo naj omenin, da sodobni računalniki sledijo premikom miške s frekvenco 40 do 125 Hz [2].

V diplomski nalogi so opisani naslednji koraki izvajanja vmesnika za laserski kazalnik ter sama uporaba vmesnika:

- detekcija projekcijskega zaslona in izračun transformacijske matrike med projekcijsko sliko in sliko kamere,
- detekcija laserskega žarka,
- uporaba primerne metode za interakcijo.

2 Kalibracija in homografija

Kalibrirati kamero in projektor ter najti homografijo med njima sta ključna postopka pri pretvorbi koordinat najdene laserske pike na zajeti sliki kamere v koordinate miške na projekcijskem zaslonu. Za iskanje homografije nam je lahko v veliko pomoč Houghova transformacija za detektiranje črt.

2.1 Kalibracija

Predpostavimo, da uporabljamo točkasto kamero (ang. pinhole camera). Postopek kalibracije kamere pomeni določitev njenih parametrov, ki se delijo na zunanje in notranje.

2.1.1 Zunanji parametri

Zunanji parametri kamere izražajo zvezo med koordinatnim sistemom kamere in nekim referenčnim koordinatnim sistemom v opazovanem prostoru. En sistem je glede na drugega v splošnem poljubno obrnjen (poljubna smer in kot rotacije) ter poljubno zamaknjen v prostoru. Zunanja parametra sta tako samo dva, in sicer 3D translacijski vektor T in rotacijska matrika R [7]. Relacija med koordinatami točke v koordinatnem sistemu kamere (P_c) in koordinatami točke v opazovanem prostoru (P_w) se izraža kot

$$P_c = R(P_w - T), \tag{2.1}$$

kjer je R matrika dimenzije $R=3\times3$ in je ortogonalna, torej zanjo velja:

$$RR^T = R^T R = I, \; |I| = 1. \tag{2.2}$$

2.1.2 Notranji parametri

Za popoln opis zveze med zunanjim svetom (tj. referenčnim sistemom) in poljem slikovnih elementov v pomnilniku nam manjka še opis zveze med slikovno ravnino v koordinatnem sistemu kamere in poljem slikovnih elementov v pomnilniku. To zvezo opisujejo notranji parametri kamere, ki jih sestavljajo goriščna razdalja f, koordinati središča slike (o_x, o_y) in efektivni velikosti slikovnega elementa (s_x, s_y), merejeni v milimetrih [7]. Koordinati slikovnega elementa v slikovnem pomnilniku x_{im} in y_{im} (merjeni v slikovnih elementih) sta povezani s

koordinatami pripadajoče točke na slikovni ravnini v koordinatnem sistemu kamere *x* in *y* (merjeni v milimetrih) z naslednjima enačbama:

$$x = -(x_{im} - o_x) \cdot s_x \tag{2.3}$$

in

$$y = -(y_{im} - o_y) \cdot s_y. \tag{2.4}$$

Dogovor je, da imajo osi koordinatnega sistema kamere nasprotno smer od smeri naraščanja koordinat slikovnih elementov. Poleg tega smo v enačbi zanemarili radialno geometrijsko popačenje, ki se povečuje v radialni smeri iz centra slike.

2.1.3 Končni model kamere

Z uporabo osnovnih enačb perspektivne projekcije

$$x = f \cdot \frac{X}{Z} \tag{2.5}$$

in

$$y = f \cdot \frac{Y}{Z} \tag{2.6}$$

izpeljemo končni model kamere:

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = M_{int} \cdot M_{ext} \cdot \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}, \tag{2.7}$$

ki predstavlja direktno povezavo med referenčnim koordinatnim sistemom v prostoru in koordinatami slikovnih elementov v slikovnem pomnilniku [7]. M_{int} je matrika notranjih parametrov in je oblike

$$M_{int} = \begin{bmatrix} -\frac{f}{s_x} & 0 & o_x \\ 0 & -\frac{f}{s_y} & o_y \\ 0 & 0 & 1 \end{bmatrix}, \tag{2.8}$$

M_{ext} pa matrika zunanjih parametrov in je oblike

$$M_{ext} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & T_x \\ r_{21} & r_{22} & r_{23} & T_y \\ r_{31} & r_{32} & r_{33} & T_z \end{bmatrix}. \tag{2.9}$$

Iz teh enačb lahko izračunamo koordinati slikovnega elementa v slikovnem pomnilniku:

$$x_{im} = \frac{x_1}{x_3}, \tag{2.10}$$

$$y_{im} = \frac{x_2}{x_3}. \tag{2.11}$$

Pri kalibraciji projektorja njegove zunanje in notranje parametre določimo na enak način kot pri kameri. Koordinatni sistem kamere samo zamenjamo s koordinatnim sistemom projektorja.

2.2 Homografija med kamero in projektorjem

Vzamemo točko na projekcijski sliki s koordinatama (*x, y*). To točko projektor projicira v neznano točko na projekcijski površini (pozicija, orientacija in goriščna razdalja projektorja glede na projekcijsko površino niso znani). Kamera zajame sliko in s tem projicira omenjeno točko v svoj koordinatni sistem v točko s koordinatama (*X, Y*). Tudi pri tej drugi transformaciji ne poznamo parametrov, ker so odvisni od neznane pozicije, orientacije in goriščne razdalje kamere glede na projekcijsko površino. Glede na vse omenjene neznanke poskušamo najti povezavo med točkama (*x, y*) in (*X, Y*), točneje, iz dane točke (*X, Y*), ki predstavlja detektiran laserski žarek na sliki, zajeti s kamero, poskušamo najti točko (*x, y*) na projekcijski sliki.

Zaradi toliko neznank se sprva zdi nemogoče najti zgoraj omenjeno preslikavo. Ker pa vemo, da vse opazovane točke ležijo na isti ravnini (privzeli smo, da je projekcijska površina ravna), lahko koordinatni sistem kamere in projektorja povežemo s homografijo oz. s transformacijsko matriko. Preslikava točke (*x, y*) iz koordinatnega sistema projektorja v točko (*X, Y*) koordinatnega sistema kamere je tako definirana z naslednjo projekcijsko transformacijo [6]:

$$(x, y) = \left(\frac{p_1X+p_2Y+p_3}{p_7X+p_8Y+p_9}, \quad \frac{p_4X+p_5Y+p_6}{p_7X+p_8Y+p_9}\right). \tag{2.12}$$

Če 2D točko (*x, y*) zapišemo s homogenimi koordinatami, jo predstavimo kot vektor $\vec{x} = \begin{bmatrix} xw \\ yw \\ w \end{bmatrix}$, za katerega velja: $\forall w: w \neq 0$[1]. Tako predstavljene točke koordinatnega sistema projektorja in kamere lahko med sabo povežemo z naslednjo enačbo:

$$H\vec{x} = \vec{y}, \qquad (2.13)$$

pri čemer je *H* normalizirana transformacijska matrika oblike $H = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix}$, zanjo pa velja $h_{33}=1$. Vektor *y* je oblike $\vec{y} = \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}$.

Enačbi *2.12* in *2.13* sta ekvivalentni, razlikujeta se le v predstavitvi koordinat. Pri iskanju transformacijske matrike *H* rešujemo enačbo *2.13* oz. $\vec{y} \times H\vec{x} = 0$. Tri vrstice, ki predstavljajo to enačbo, med sabo niso linearno neodvisne, zato lahko tretjo vrstico brez posledic zavržemo. Če poznamo koordinate vsaj štirih korespondenčnih točk (točk na sliki kamere, ki predstavljajo iste točke na projekcijski sliki), dobimo problem, ki ga je možno rešiti z metodo singularnega razcepa (ang. singular value decomposition).

V sistemu projektor-kamera teh korespondenčnih točk ni težko najti, saj nam projektor omogoča aktivno manipuliranje scene. Z njim projiciramo kalibracijske slike, na katerih so nam znane koordinate določenih točk v koordinatnem sistemu projektorja. Te iste točke potem s pomočjo Houghove transformacije najdemo na sliki, ki jo zajame kamera (določimo njihove koordinate v koordinatnem sistemu kamere).

2.3 Houghova transformacija

Houghova transformacija se uporablja za detektiranje vseh vrst krivulj, ki jih lahko zapišemo v parametrični obliki. V našem primeru pride v poštev pri detekciji robov projekcijske slike, kar je del računanja transformacijske matrike. Vhod v algoritem Houghove transformacije predstavlja slika robnih točk (dobimo jo kot izhod iz detektorja robov). Glavna ideja Houghove transformacije je preslikati težek problem detektiranja vzorca na sliki v enostaven problem

[1] Homogeni vektorji in homografije so invariantni glede na skaliranje s skalirnim faktorjem, različnim od nič.

iskanja maksimuma v parametričnem prostoru dane krivulje. Poglejmo si primer detekcije premice v polarni obliki:

$$R = x \cdot cos\,(T) + y \cdot sin\,(T). \qquad (2.14)$$

Parameter R predstavlja razdaljo med premico in izhodiščem, parameter T pa predstavlja orientacijo premice. Osnovna koraka Houghove transformacije za detekcijo premice sta:

- pretvori problem detekcije premice v problem iskanja presečišča premic,
- pretvori problem iskanja presečišča premic v problem iskanja maksimuma (vrha) v parametričnem prostoru [7].

Houghov algoritem je glasovalni algoritem, kjer vsaka točka na robni sliki glasuje za svojo premico. Premice, ki imajo več glasov, kot je določen prag, se smatrajo kot detektirane. Sorazmerno z zgornjim opisom lahko implementiramo tudi detekcijo krogov. Vse, kar potrebujemo, je parametrična predstavitev kroga.

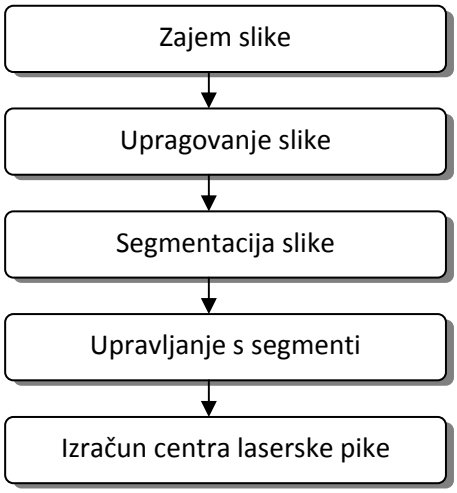
Knjižnica OpenCV omogoča táko iskanje premic v binarni sliki z uporabo funkcije `cvHoughLines2`. Funkcija sprejme več nastavljivih parametrov. Določimo lahko metodo transformacije (standardna ali verjetnostna), ločljivost parametrov R in T, prag, ki predstavlja najmanjše število glasov za detektirano črto, najmanjšo dolžino premice ter največji razmak med dvema segmentoma premice, da ju funkcija združi [9].

3 Detekcija laserskega žarka

Pri iskanju laserskega žarka na zajeti sliki gre za iskanje določenih značilk. V računalniškem vidu in procesiranju slik je značilka poimenovana kot del informacije, ki je ključnega pomena za reševanje določenega računskega problema znotraj aplikacije [13]. Detekcija je sestavljena iz več faz. Najprej seveda potrebujemo sliko, na kateri iščemo značilko. Preden pričnemo sam postopek detekcije, je potrebno sliko zgladiti, in sicer z Gaussovim filtrom. Potrebujemo tudi mehanizem, ki zna značilko izločiti, tako da iz opazovane slike izlušči pomembne informacije. V našem primeru se mora pri vsakem slikovnem elementu odločiti (lokalno), ali je iskana značilka prisotna. Nazadnje moramo sliko pravilno klasificirati glede na prej pridobljene informacije. Tu gre za določanje, ali je laserska pika prisotna na sliki ali ne. Od uspešnosti odkrivanja značilk je odvisno celotno izvajanje vmesnika. V nadaljevanju predstavim tri različne metode za detekcijo laserskega žarka. Vsaka izmed njih ima svoje dobre in slabe lastnosti, ki pa so opisane v podpoglavju *6.3*.

3.1 Metoda segmentacije

To metodo detekcije laserske pike je možno razdeliti na 5 korakov [2], predstavljenih na sliki *2*.



Slika 2: Koraki metode segmentacije

Zajem slike

S kamero zajamemo sliko. Barva vsakega slikovnega elementa je predstavljena z osmimi biti, torej je vrednost vsakega slikovnega elementa med 0 in 255.

Upragovanje slike

Vsaka slika je binarizirana z določenim pragom, kar pospeši nadaljnje procesiranje. Prag je določen predhodno. Preveri se intenzivnost vsakega slikovnega elementa na sliki brez laserske pike. Vrednost, malo nad največjo intenzivnostjo na sliki, predstavlja prag.

Segmentacija slike

Slika se razdeli na posamezne segmente, kjer vsak segment predstavlja svoje povezano območje z vrednostmi nad pragom za binarizacijo. Vsako območje je oštevilčeno in shranjeno skupaj s podatki o območju, kjer se nahaja (v slikovnih elementih).

Upravljanje s segmenti

- Če prejšnji korak ne vrne nobenega segmenta, predpostavimo, da laserske pike ni na sliki.
- Če je detektiran samo en manjši segment, nadaljujemo z naslednjim korakom.
- V primeru več segmentov se odločimo glede na njihovo število in njihovo razširjenost na sliki. Če je število segmentov majhno in je razdalja med njimi manjša od v naprej dane (v slikovnih elementih), te segmente združimo v enega. Tak segment predstavlja hitro se premikajočo lasersko piko. Kamera namreč slika v diskretnem času, čas med zajetimi posameznimi slikami pa je lahko nekaj milisekund. Ko pa je na sliki prisotnih veliko segmentov in so razkropljeni po večjem delu slike, je ta metoda neuspešna. Do te situacije lahko pride, kadar je prag za binarizacijo premajhen ali pa se na projekcijskem zaslonu pojavi več kot ena laserska pika.

Izračun centra laserske pike

Kadar prejšnji korak vrne samo en segment, lahko začnemo z računanjem njegovega središča. Uporabimo naslednjo enačbo:

$$(x, y) = \left(\frac{\sum_{i=1}^{n} x_i}{n}, \quad \frac{\sum_{i=1}^{n} y_i}{n}\right), \tag{3.1}$$

kjer *n* predstavlja število slikovnih elementov v segmentu, (x_i, y_i) pa sta koordinati posameznega slikovnega elementa znotraj segmenta.

Metoda je uspešna, če je projekcijski zaslon v celoti znotraj vidnega polja kamere, če ni relativnih premikov med projekcijsko površino in kamero, odkar se aplikacija začne izvajati, in če je laserska pika svetlejša od česarkoli v vidnem polju kamere.

3.2 Metoda ločevanja ozadja

Metoda ločevanja ozadja (ang. background subtraction) je proces, ki odkriva gibanje ali pomembne razlike med zajeto in referenčno sliko. Hkrati odstrani vse nepomembne komponente (ozadje). Najpogosteje se uporablja v nadzornih sistemih. Idejo te metode predstavlja formula *3.2*, kjer je I_t slika, posneta v času *t*, *B* slika ozadja, *T* pa prag razlike.

$$\|I_t - B\| > T. \tag{3.2}$$

Pri tehniki, opisani s formulo *3.2*, se za model ozadja vzame kar prva slika iz videa, ki ji pravimo tudi referenčna slika. Pomembno je, da je referenčna slika brez predmetov, ki jih želimo kasneje opredeliti kot predmete v ospredju. Slikovni elementi trenutne slike, ki se od istoležečih slikovnih elementov referenčne slike razlikujejo za več kot *T*, predstavljajo predmet v ospredju (v našem primeru je to laserska pika). Ta tehnika daje zadovoljive rezultate v okolju, kjer se osvetlitev ne spreminja veliko [3].

Za boljše prilagajanje spremembam v osvetlitvi prostora in rahlim spremembam ozadja vzamemo namesto ene povprečje več slik. Najprej zajamemo *n* (npr. *n=10*) referenčnih slik in v vsakem slikovnem elementu izračunamo povprečje μ in standardno deviacijo σ. Enačbi sta naslednji:

$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{3.3}$$

in

$$\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2}, \tag{3.4}$$

kjer je x_i vrednost posameznega slikovnega elementa v i-ti sliki. Določimo prag ozadja T kot $\mu \pm 2\sigma$. Vsak slikovni element novo zajete slike nato primerjamo s to vrednostjo. Če je slikovni element del ozadja, bo njegova vrednost znotraj praga v 95 % primerov. Vsak slikovni element z vrednostjo nad pragom se smatra, da ni del ozadja. Metoda je še vedno hitra, vendar zahteva več pomnilniškega prostora.

3.3 Metoda povprečenja

Metoda deluje na principu shranjene zgodovine določenega števila slik (ter povprečja med njimi, izračunanega v vsakem slikovnem elementu) in zgodovine vrednosti v točkah, kjer so bili detektirani laserski žarki (ter njihovega povprečja).

Najprej se izvede kalibracija. Ob njenem izvajanju je pomembno, da laserski žarek ne sveti na območje vidnega polja kamere. S kalibracijo se shrani prva zgodovina slik. Za pravilno detekcijo mora biti laserska pika svetlejša od najsvetlejšega dela slike kamere.

Detekcija laserskega žarka se izvede ob vsakem zajemu slike s kamero. Postopek detekcije je naslednji:

- Vsako sliko, zajeto s kamero, najprej zgladimo z Gaussovim filtrom.
- Slika, ki jo zajame kamera, je tipa RGB, kar pomeni, da so podatki o barvi vsakega slikovnega elementa shranjeni v treh kanalih. V enem je podatek o intenzivnosti rdeče barve (rdeč kanal), v drugem zelene (zelen kanal) in v tretjem modre (moder kanal). Ker naj bi laser predstavljal najintenzivnejši oz. najsvetlejši del slike in ker so najsvetlejši deli najbolj povdarjeni v rdečem kanalu, iz zajete slike izločimo samo rdeč kanal. S tem zmanjšamo količino podatkov za procesiranje.
- Iz slik v zgodovini izračunamo povprečno vrednost v vsakem slikovnem elementu. Tako dobimo povprečeno sliko.
- Sprehodimo se skozi vsak slikovni element zajete slike oz. njenega rdečega kanala in izračunamo razliko tega slikovnega elementa z istoležečim slikovnim elementom rdečega kanala povprečene slike. Slikovni element z največjo razliko je najverjetneje lokacija, kjer se nahaja laserski žarek. To razliko, kakor tudi lokacijo tega slikovnega elementa, shranimo.

- Vsako zajeto sliko shranimo v zgodovino slik, in sicer na mesto najstarejše.
- Izračunamo povprečno vrednost vseh shranjenih točk, v katerih je bil detektiran laserski žarek. Če je razlika med najintenzivnejšim slikovnim elementom trenutne slike in tukaj izračunanim povprečjem večja od danega praga za klasifikacijo, smo našli točko, v kateri se nahaja laserski žarek. To točko še shranimo na mesto najstarejše prej detektirane točke.

4 Upravljanje miške z laserskim žarkom

Računalniška miška je vhodna naprava za interaktivno delo z računalnikom. Običajno je opremljena s tremi gumbi (levi, desni in srednji). Z računalnikom komunicira preko pripadajočega gonilnika. Osnovne akcije, ki jih omogoča, so klik, dvojni klik, vleci in spusti ter vrtenje sledilnega koleščka (v primeru, da ga ima). Pomen posamezne akcije je odvisen od aplikacije, v kateri miško uporabljamo.

Da sistem ve, kaj mora izvesti, mora poznati koordinate miške na zaslonu in vedeti, katero akcijo je miška izvedla. Ko poznamo transformacijsko matriko, ki povezuje koordinatni sistem slike, zajete s kamero, in koordinatni sistem projekcijske slike, je preračunavanje koordinat detektiranega laserskega žarka v koordinate miške na zaslonu enostavno. Gre za preprosto množenje, opisano v enačbi *2.13*. Problem nastane, ko želimo z laserskim kazalnikom posnemati akcije, ki jih omogoča miška. Laserski kazalnik ima namreč samo en gumb, ki omogoča vklop in izklop laserskega žarka.

4.1 Premik miškinega kazalca

Da izvedemo premik miške na določeno lokacijo, moramo na to lokacijo posvetiti z laserskim žarkom. Algoritem sproti preračunava koordinate zaznane laserske pike na sliki kamere v koordinate miške na projekcijski sliki. Kot je že bilo rečeno, za to preračunavanje uporablja transformacijsko matriko.

4.2 Levi klik

Z laserskim žarkom premaknemo miškin kazalnik na željeno lokacijo. Da bi izvedli levi klik, moramo laserski žarek na tej isti lokaciji izklopiti, vklopiti in zopet izklopiti. Drugi vklop in izklop žarka sta ekvivalenta pritisku in spustu levega gumba miške. Vemo, da je praktično nemogoče žarek ponovno vklopiti na točno isti lokaciji, torej da detekcija žarka vrne isti slikovni element. Zato je programsko potrebno določiti okolico, v kateri se ponovni vklop žarka smatra kot vklop na isti lokaciji. Tu gre za okolico detektiranega žarka, merjeno v slikovnih elementih.

Poleg tega za levi klik obstajajo tudi časovne omejitve. Program mora vseskozi meriti čas, kako dolgo v vidnem polju kamere ni detektiranega žarka. Čas se meri za vsak izklop posebej. Levi klik se izvede, če je čas med prvim izklopom žarka in naslednjim vklopom manjši od nastavljenega praga. Čas se meri tudi za vsak vklop žarka posebej. S tem ločimo klik ter akcijo vleci in spusti. Lahko se namreč zgodi, da preden uporabnik drugič izklopi žarek (spusti levi gumb miške), kamera zajame dve sliki ali več. Med tem uporabnik skoraj gotovo vsaj minimalno premakne lasersko piko, kar aktivira akcijo vleci in spusti, kot je opisano v podpoglavju *4.4*. Temu primerno je definiran časovni prag, ki pove, kako dolgo mora biti vklopljen laserski žarek, preden se dejansko sproži premik miškinega kazalca. Običajno je ta prag nastavljen na zelo majhno vrednost, npr. 0,15 sekunde.

Obstajajo tudi druge implementacije levega klika, ki pa ponavadi ne dopuščajo implementacije še ostalih akcij miške ali pa so nenaravne za uporabo. Pri eni izmed takih mora uporabnik z žarkom svetiti na eno mesto oz. njegovo okolico določen čas, da se izvede klik. Ena izmed možnosti je tudi ta, da se klik izvede tam, kjer se laserski žarek izklopi.

4.3 Dvojni levi klik

Gre za dva zaporedna leva klika, ki se morata izvesti na isti lokaciji oz. v njeni okolici. Poleg že omenjenih omejitev za levi klik moramo pri dvojnem kliku upoštevati še čas med prvim in drugim klikom. Ta čas je določen s strani operacijskega sistema oz. gonilnika za računalniško miško. V okolju Windows je te parametre možno spreminjati v nastavitvah za miško na nadzorni plošči.

4.4 Vleci in spusti

Za to akcijo se lahko uporabijo enake rešitve kot pri levem kliku. Laserski žarek moramo na lokaciji, kjer želimo izvesti vlečenje, izklopiti in vklopiti nazaj v določenem časovnem intervalu. S tem se izvede pritisk levega gumba miške. Če sedaj premaknemo laserski žarek na drugo lokacijo (preteči mora določen čas, kot je opisano v podpoglavju *4.2*), se temu primerno spremenijo tudi koordinate miške, njen levi gumb pa je še vedno pritisnjen (akcija vleci). Na lokaciji, kjer izklopimo laserski žarek, se izvede akcija spusti (spusti se levi gumb miške).

4.5 Desni klik

Ponavadi laserski kazalnik oddaja samo en žarek ene barve, zato za aktivacijo desnega klika potrebujemo nek dodatni mehanizem. V večini primerov se ta akcija aktivira programsko, obstajajo pa tudi strojne rešitve [11]. Slednje se pojavljajo v obliki dodatnih gumbov ali stikal, vezanih preko podatkovnega kabla na računalnik. Akcija desnega klika se sproži ob pritisku na gumb ali stikalo. Programske rešitve ponavadi uporabljajo prepoznavanje kretenj z laserskim kazalnikom. Ob določeni kretnji (npr. s kazalnikom narišemo krog [15]) se aktivira desni klik. Ta akcija se nato izvede ob naslednjem kliku. Več o prepoznavanju kretenj je predstavljeno v naslednjem poglavju.

4.6 Pomik s sledilnim koleščkom

Večina računalniških mišk ima poleg gumbov še sledilni kolešček, ki olajša premik v smeri gor ali dol po vsebini okna. Omenjeno drsenje lahko izvedemo tudi brez tega dodatnega koleščka. Aplikacije, katerih vsebina se razteza zunaj vidnega polja zaslona, vsebujejo ponavadi drsni trak z drsnikom, s katerim se pomikamo po tej vsebini. Drsenje je torej možno izvesti z akcijo vleci in spusti. Vrtenje sledilnega koleščka pa je možno implementirati tudi s prepoznavanjem kretenj laserskega žarka. Takšna implementacija je opisana v podpoglavju *6.4*.

5 Prepoznavanje kretenj

Vloga računalnikov se v družbi povečuje, zato ima interakcija med človekom in računalnikom pomembno vlogo v vsakdanjem življenju. Prepoznavanje kretenj omogoča razvoj naravnih in učinkovitih vmesnikov za omenjeno interakcijo. Cilj tega področja je interpretacija človeških kretenj preko matematičnih algoritmov. Kretnja izvira iz kateregakoli stanja telesa ali njegovega premikanja. Največkrat se kot kretnje smatrajo premiki ali stanja rok in glave. Področje je zelo široko. Med drugim se zadnje čase ukvarjajo s prepoznavanjem čustev z obraza ali iz kretenj rok, z interpretacijo znakovnega jezika, z interpretacijo in prepoznavanjem drže, hoje in človeškega obnašanja [14].

Preko prepoznavanja kretenj lahko računalniki razumejo človeško govorico telesa. S tem se, v primerjavi s tekstovnimi in grafičnimi uporabniškimi vmesniki, gradi močnejša povezava med človekom in strojem, saj pride do naravnejše interakcije med njima. Primer tega je, ko s prstom upravljamo miškin kazalec. V prihodnosti bi to lahko pomenilo, da vhodne naprave, kot so miška, tipkovnica in zaslon na dotik, postanejo odvečne.

Kot kretnje smatramo tudi netekstovne, ročno narisane simbole, ki jih računalnik zaznava preko grafične tablice, večdotičnega zaslona ali miške. Tu gre za interakcijo z računalnikom preko risanja simbolov s kazalno napravo.

Za prepoznavanje kretenj se uporabljajo tehnike računalniškega vida in procesiranja slik. Ključni element za prepoznavanje je kamera, ki prenese podatke do računalnika. Ta uporablja kretnje kot ukaze za upravljanje naprav ali aplikacij.

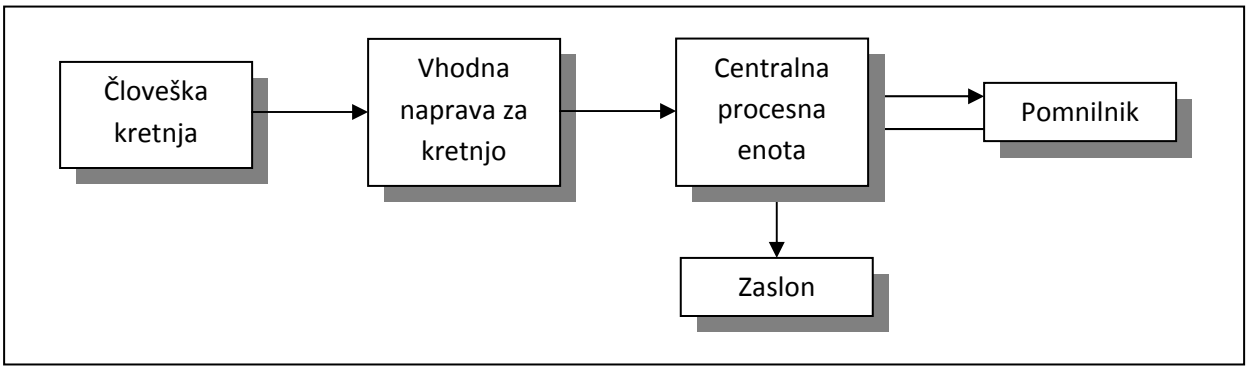
5.1 Prepoznavanje kretenj miške

Če želimo kretnje uporabljati za nadzor naprav, moramo odgovoriti na naslednja vprašanja:

- Katere naprave hočemo nadzorovati?
- Katere ukaze za nadzor predstavlja posamezna kretnja?
- Kako se kretnja pretvori v ukaz?

- Kako daleč sta fizično ločena iniciacija ukaza in njegov odziv (lokalni ali oddaljeni nadzor)?

V našem primeru gre za nadzor miške, kjer je pomen posamezne kretnje odvisen od tega, katero aplikacijo nadzorujemo. Pretvorba kretnje v ukaz je povsem programska. Vsak premik kazalca z laserskim kazalnikom se klasificira kot vodoravni (v levo ali desno) ali navpični (gor ali dol) gib. Te štiri kretnje se lahko sestavljajo v kompleksnejše (npr. kretnja levo-dol-desno-gor). Kretnja tipa levo-dol-desno lahko v našem sistemu npr. aktivira desni gumb miške. V primeru neprepoznanih kretenj program vrne prazno kretnjo. Fizično sta laserski žarek in pripadajoči premik miškinega kazalca na isti površini (projekcijska površina), torej gre za lokalni nadzor. Glede na te odgovore se naša implementacija interpretacije kretenj uporablja za nadzor nad računalniškim pomnilnikom in njegovim zaslonom [10]. Arhitektura takega sistema je predstavljena na sliki *3*. V tem sistemu se kretnja preko vhodne naprave (kamere) prenese v računalniški pomnilnik, rezultat pripadajoče akcije pa je viden na računalniškem zaslonu oz. na projicirani sliki. Preden se akcija izvrši, se mora kretnja izvesti do konca.



Slika 3: Bločni diagram arhitekture za nadzor računalniškega pomnilnika in zaslona s kretnjami

Osnova za prepoznavanje kretenj miške v našem sistemu je detektirana laserska pika. Detektor vrne koordinate miške, ki jih uporabimo za prepoznavanje kretenj. Sámo prepoznavanje se lahko implementira na različne načine. Ena izmed možnosti je, da uporabimo nevronsko mrežo. Taka implementacija je obsežna, njene lastnosti pa presegajo obseg te diplomske naloge.

Druga možnost je preprostejša, ampak tudi bolj omejena. Gre za primerjavo sosednjih koordinat premika kazalca. Če se trenutni koordinati od prejšnjih bolj razlikujeta v smeri x kot v smeri y, gre najbrž za vodoravno kretnjo. Da ugotovimo, ali gre za kretnjo v levo ali desno smer, preverimo, ali se je koordinata x povečala (kretnja v desno) ali zmanjšala (kretnja v levo). Na enak način realiziramo prepoznavanje navpičnih kretenj. Zanje je značilno, da se kazalec bolj premakne v smeri y kot v smeri x. Če se koordinata y povečuje, gre za gibanje kazalca navzdol, če pa se manjša, gre za gibanje navzgor.

Kretnje so ponavadi hitri in preprosti gibi, zato je nesmiselno shranjevati celotno pot laserske pike. Temu se izognemo tako, da prepoznavanje kretenj časovno omejimo. Kadar je laserska pika na sliki kamere prisotna dlje kot npr. tri sekunde, najbrž ne gre za kretnjo, ki bi predstavljala nek ukaz. S tem tudi zmanjšamo obremenjenost procesorja in pomnilnika.

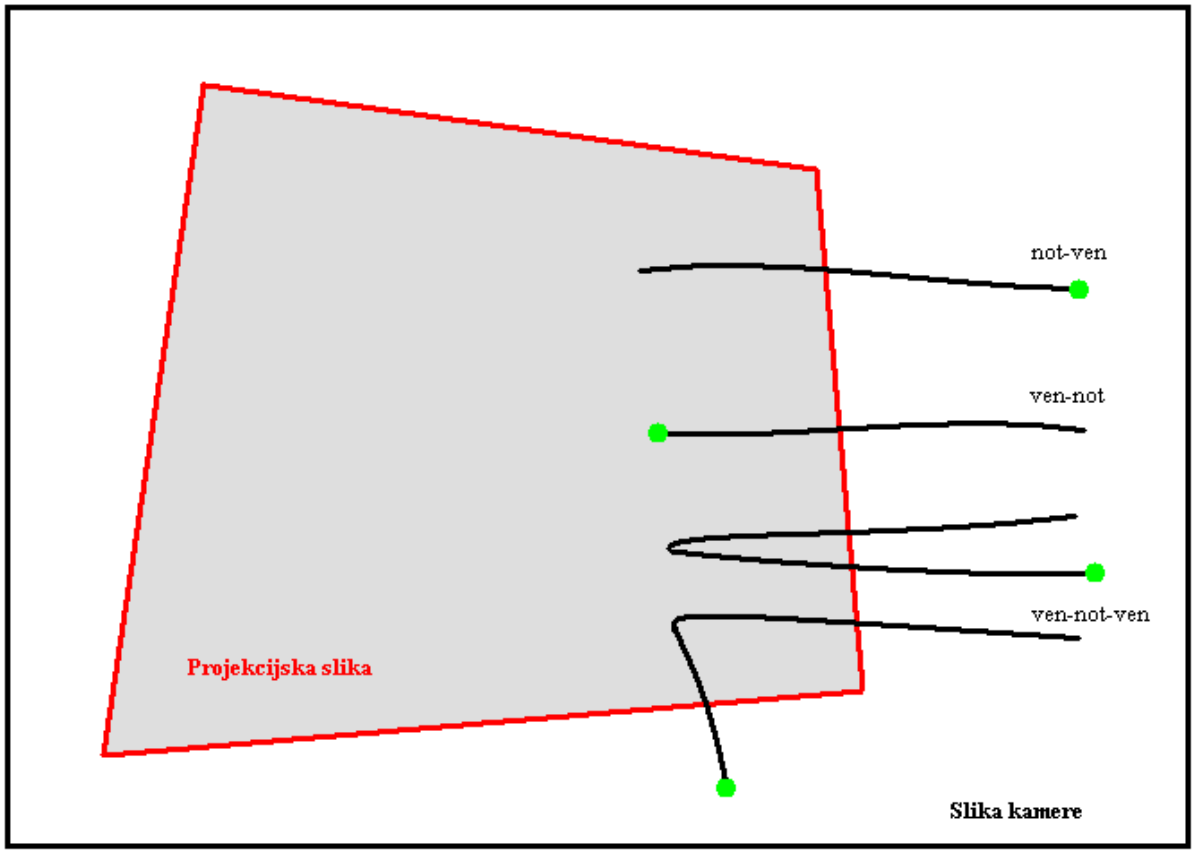
5.2 Kretnje, ki vključujejo periferni del projekcijskega zaslona

Poleg že omenjenih kretenj, ki jih vmesnik prepozna, lahko vključimo še dodatne. Ker vsaki kretnji dodelimo posamezeno akcijo, lahko s tem povečamo število akcij, torej tudi število aplikacij, ki jih lahko nadzorujemo z laserskim kazalnikom. Kot osnovo za dodatne kretnje lahko uporabimo prehod čez robove projekcijskega zaslona [5]. Da lahko uporabimo to metodo, moramo na začetku postaviti kamero tako, da njeno vidno polje poleg projekcijskega zaslona zajema še del njegove okolice na projekcijski površini. Najbolje je, da kamera zajema del projekcijske površine zunaj vseh štirih robov projekcijske slike (slika *11*), ker s tem maksimiramo število dodatnih kretenj. Na sliki *4* so prikazani trije različni prehodi preko roba projekcijske slike. Krog predstavlja konec kretnje, črte pa njeno pot.

- Prehod not-ven: kadar z laserskim žarkom posvetimo znotraj projekcijske slike in z njim prestopimo enega izmed robov na zunanjo stran slike, govorimo o prehodu not-ven.
- Prehod ven-not: do tega prehoda pride, kadar z laserskim žarkom najprej posvetimo zunaj projekcijske slike, nato pa s prižganim kazalnikom naredimo kretnjo preko enega izmed robov v notranjost projekcijske slike.
- Prehod ven-not-ven: zgornji dve kretnji lahko kombiniramo. Kadar najprej izvedemo prehod ven-not in nato, ne da vmes ugasnemo laserski kazalnik, nadaljujemo s prehodom not-ven, dobimo prehod ven-not-ven. Pri tem kombiniranju nam, v nasprotju s prehodom

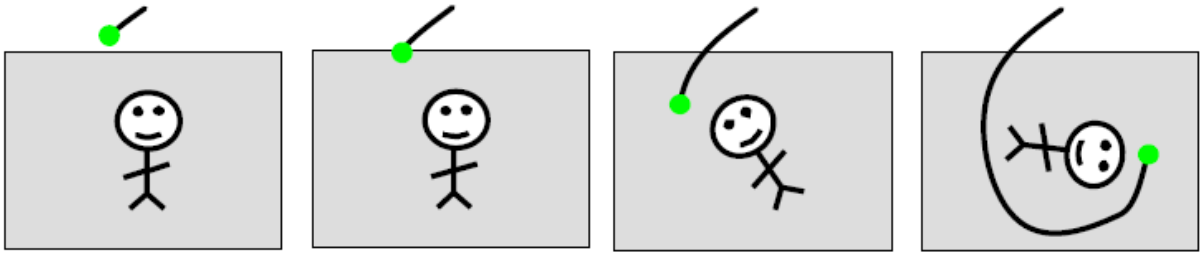
not-ven-not, ni potrebno paziti, da laserski žarek ne zapusti vidnega polja kamere. Ta prehod lahko še dodatno razčlenimo glede na to, ali gre za prehod preko enega ali več robov (npr. okrog vogala slike).

Pri predstavitvenih aplikacijah, kot je npr. Microsoft Office PowerPoint, lahko te prehode zlahka uporabimo. Prehod not-ven preko desnega roba projekcijske slike lahko uporabimo za prehod na naslednjo prosojnico, medtem ko prehod not-ven preko levega roba prikaže prejšnjo prosojnico.



Slika 4: Kretnje, ki vključujejo periferni del projekcijskega zaslona

Vsi trije prehodi lahko predstavljajo ukaze brez parametrov, lahko pa jih uporabimo tudi kot ukaze s parametri. Ena izmed možnih implementacij je predstavljena na sliki *5*. Tu prehod ven-not preko zgornjega roba povežemo z akcijo rotacije slike. Takoj ko laserska pika prestopi rob, določimo osnovni vektor med centrom slike in to točko. Kotna razlika med osnovnim vektorjem in vektorjem, ki ga določata center slike ter trenutna lokacija laserske pike, določa kot rotacije slike. Ko izklopimo laserski žarek, se akcija konča.



Slika 5: Primer kretnje s parametrom

Kot parametre za izvedeno akcijo lahko uporabljamo naslednje značilnosti prehodov:

- preko katerega roba se je izvedel prehod,
- pozicija na robu, kjer je prišlo do prehoda,
- kot, pod katerim se je izvedel prehod,
- hitrost izvedenega prehoda,
- pot, ki jo je prepotovala laserska pika.

5.3 Problemi pri interakciji s pomočjo kretenj

Z natančnostjo in uporabnostjo programske opreme za prepoznavanje kretenj je povezanih veliko izzivov. Prepoznavanje kretenj iz slik ali videa je omejeno z opremo, ki jo uporabljamo, in s šumom, prisotnim na slikah. Svetlobni pogoji so med samim izvajanjem lahko nekonsistentni. Prepoznavanje lahko otežujejo tudi stvari v ozadju slike.

Uporaba številnih implementacij prepoznavanja kretenj je za splošno uporabo vprašljiva. Algoritem, ki je kalibriran za določeno kamero, lahko pripelje do neželjenih rezultatov, če uporabimo drugačno kamero. Količina šuma v ozadju prav tako otežuje sámo prepoznavanje, še posebej, če pride do delnega ali popolnega zakrivanja (ang. partial or full occlusion). Variacije v natančnosti prepoznavanja se pojavaljajo glede na oddaljenost kamere ter njeno kvaliteto in ločljivost slik, ki jih zajema.

V zgodnjih 80-ih letih prejšnjega stoletja so zasnovali vertikalne zaslone na dotik. Zaradi stranskega učinka, imenovanega roka gorile (ang. gorilla arm), tehnologija ni prišla v vsakdanjo uporabo [14]. Oblikovalci sistema namreč niso opazili, da ljudje nismo prilagojeni na takšno držo rok. Kmalu po začetku uporabe dobimo v roki občutek bolečine in utesnjenosti. Uporabnik

med uporabo izgleda kot gorila, na koncu pa se tudi sam počuti tako. Oblikovalci, ki morajo pri uporabi določene tehnologije upoštevati človeške faktorje, morajo zato biti zelo previdni. Tu gre za vprašanje, kako se bo tehnologija obnesla v realnem okolju. Ta učinek ne predstavlja problema pri kratkoročni uporabi laserskega kazalnika za prepoznavanje kretenj, kjer prihaja do kratkih interakcij, ki ne trajajo dovolj dolgo, da bi ga povzročile. Kadar pa laserski kazalnik uporabljamo dlje časa, do tega stranskega učinka lahko pride.

6 Implementacija vmesnika in njegova uporaba

V tem poglavju so opisane metode, ki sem jih izbral za svojo implementacijo vmesnika, razlogi za izbiro le-teh ter celoten postopek uporabe vmesnika. Z zagonom datoteke `laser.exe` se prične izvajanje vmesnika.

6.1 Predpostavke

Za uspešno izvajanje vmesnika je najprej potrebna pravilna postavitev opreme. Projektor mora biti dovolj oddaljen od projekcijske površine, slika dovolj velika za tekočo in natančno interakcijo z laserskim kazalnikom, hkrati pa tudi dovolj jasna. Pozorni moramo biti, da vidno polje kamere zajema celotno projekcijsko sliko z njeno bližnjo okolico. Vmesnik na začetku prikazuje sliko kamere in s tem omogoča njeno pravilno namestitev.

Pri svoji implementaciji sem predpostavil, da je projekcijska površina ravna ter da nepoznavanje notranjih parametrov kamere in projektorja, skupaj z radialnim popačenjem slik, nima ključnega pomena pri končni uporabi vmesnika. Dejansko bi določanje teh parametrov vplivalo na uporabnost vmesnika. Za njihovo pridobitev moramo namreč pri roki imeti natisnjen vzorec, prilepljen na ravno podlago. Ta vzorec nato slikamo s kamero iz različnih pozicij z različno orientacijo. To dosežemo tako, da premikamo ali podlago ali kamero. Na teh slikah detektiramo značilne točke (ang. feature points), iz katerih lahko določimo vseh pet notranjih parametrov kakor tudi koeficiente radialnega popačenja [8].

Kot neznanke še vedno ostanejo pozicija kamere in projektorja ter njuna orientacija oz. njuna medsebojna povezava v obliki transformacijske matrike. Za njeno določitev potrebujemo vsaj štiri korespondenčne točke. Za te točke sem izbral vogale projekcijske slike. Na začetku izvajanja aplikacije lahko izberemo avtomatsko ali ročno določanje teh točk. V primeru, da se odločimo za ročno določanje korespondenčnih točk, nam vmesnik prikaže trenutno sliko kamere, mi pa v pravilnem vrstnem redu kliknemo na vsak vogal projekcijskega zaslona. Najprej kliknemo na levi zgornji vogal, nato na desni zgornji vogal, sledi desni spodnji vogal in nazadnje levi spodnji vogal. Vmesnik nadaljuje izvajanje pri točki *6.3*. Avtomatsko določanje korespondenčnih točk je opisano v nadaljevanju.

6.2 Avtomatsko določanje transformacijske matrike

Postopek vključuje štiri kalibracijske slike. Vsaka izmed njih predstavlja svoj rob projekcijskega zaslona (bela črta na črni podlagi). Projiciranje teh slik poteka v celozaslonskem načinu.

6.2.1 Glajenje šuma

Kamera zajame vsako izmed štirih slik in jo najprej spremeni v sivinsko sliko. Vemo, da se na slikah pojavlja šum. V računalniškem vidu se šum nanaša na katerokoli entiteto na sliki, v podatkih ali vmesnih rezultatih, ki nas v končni fazi ne zanima ali pa celo moti [7]. Poznamo več vrst šumov, najpogostejši med njimi pa je Gaussov. To pomeni, da je pravim vrednostim slikovnih elementov dodana vrednost šuma, ki ima naključno spremenljivko, porazdeljeno po Gaussovi porazdelitvi. Tega odpravimo, če izračunamo konvolucijo slike z Gaussovo funkcijo. Vsak slikovni element s slike konvoliramo z 2D matriko, ki predstavlja vzorce realnega Gaussovega jedra z določeno deviacijo σ. OpenCV omogoča različne tipe glajenja šuma (funkcija `cvSmooth`), pri Gaussovem pa uporabo jeder različnih velikosti. Izbral sem jedro velikosti 5 × 5, ki še vedno omogoča hitro računanje konvolucije.

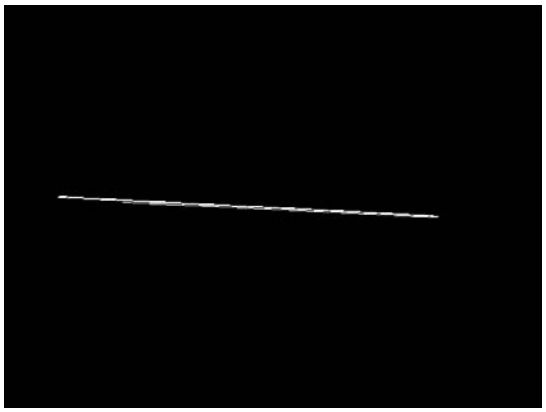
6.2.2 Upragovanje

Ta postopek spremeni sivinsko sliko v binarno po naslednjem pravilu:

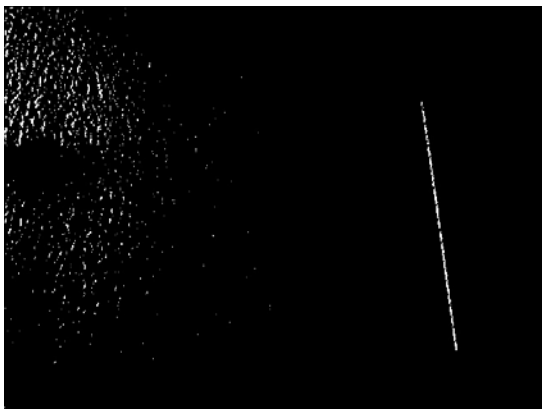
$$g(i,j) = \begin{cases} 1, & \text{če velja } f(i,j) \geq T \\ 0, & \text{sicer} \end{cases}, \qquad (6.1)$$

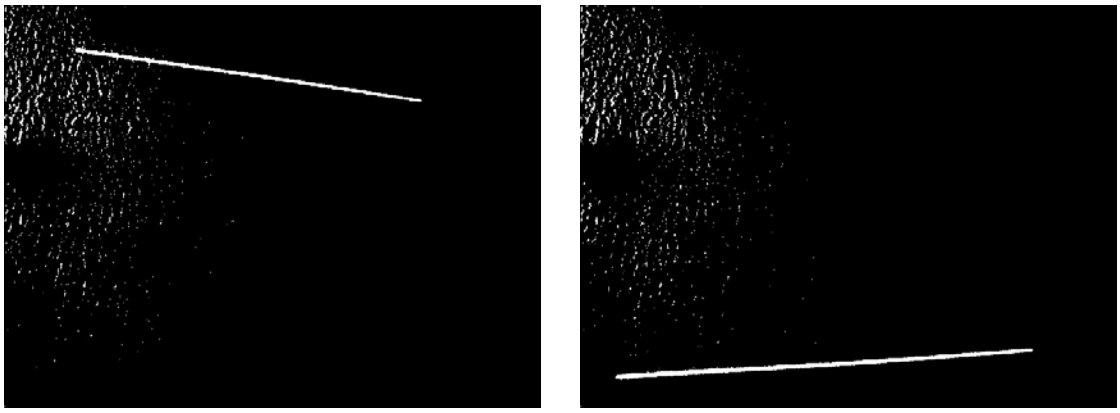
kjer je *f(i, j)* vrednost posameznega slikovnega elementa sivinske slike, *g(i, j)* vrednost posameznega slikovnega elementa binarne slike in *T* vrednost praga. Enačba *6.1* pomeni, da bodo slikovni elementi nad določenim pragom dobili vrednost 1 (postanejo beli), tisti pod pragom pa 0 (postanejo črni). Upragovanje (ang. thresholding) je izredno pomembno, saj je od njega odvisna detekcija robov in detekcija vogalov, posledično pa tudi preračunavanje detektirane laserske pike v koordinate kazalca miške. Če je prag postavljen prenizko, ne izstopajo samo robovi, ampak tudi ostali svetli elementi na sliki. V primeru previsokega praga pa tudi robovi niso več vidni na binarni sliki. Vse skupaj je poleg kvalitete kamere in posledično njene zajete slike ter kvalitete projicirane slike v veliki meri odvisno od osvetljenosti prostora. Prag je potrebno določiti, še preden se lotimo detekcije robov. Moj vmesnik ga določa

samodejno. Najprej nastavim prag na minimalno smiselno vrednost. Na projekcijsko površino projiciram črno sliko z belo črto na sredini v celozaslonskem načinu. Pri vsakem obhodu zanke povečam prag za pet. Ta prag uporabim za spreminjanje zajete slike v binarno, na njej pa s Houghovo transformacijo najdem vse črte. Prag je dovolj visok, če se vse detektirane črte nahajo nekje na polovici slike in so vodoravne (slika *6*). V primeru, da so črte detektirane tudi na drugih predelih slike, gre program ponovno skozi zanko. Prag se povečuje, dokler le-ta ne doseže maksimuma, ki je pri sivinskih slikah enak 255. To pomeni, da je prostor najbrž preveč osvetljen, kar onemogoča pravilno detekcijo robov projekcijske slike, zato program preneha z delovanjem. Reševanje tega problema je opisano v razdelku *7.2.1*. Tako določen prag vstopa v upragovanje pri detekciji robov. Rezultat samega upragovanja slik robov je viden na sliki *7*.



Slika 6: Detektirane črte kalibracijske slike samodejnega določanja praga za upragovanje





Slika 7: Binarne slike robov

6.2.3 Detekcija robov s Houghovo transformacijo

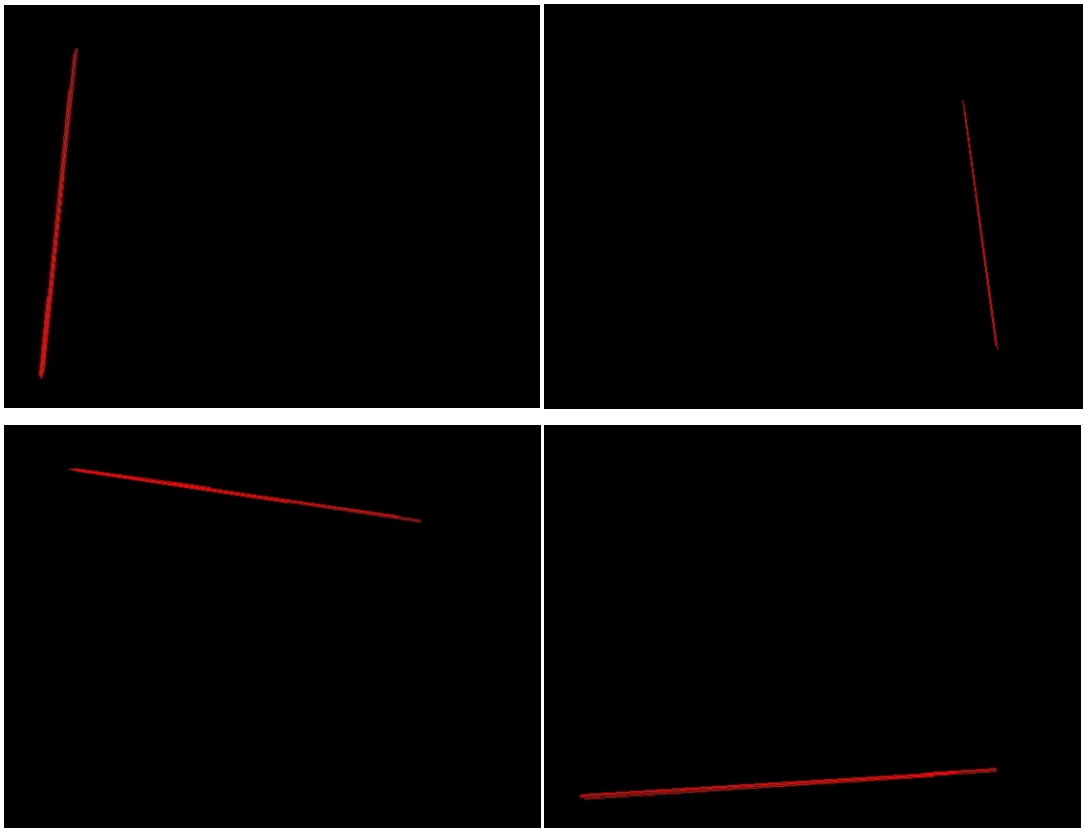
Z binarno sliko vstopamo v metodo iskanja robov. Za ta proces uporabljam Houghovo transformacijo. Ker so robovi na projekcijski sliki široki tri slikovne elemente, je pričakovati, da bo ta metoda vrnila več kot eno daljico znotraj roba (slika *8*). Poleg tega se problemi pojavljajo tudi pri določanju vseh parametrov funkcije `cvHoughLines2`, kar je podrobneje opisano v razdelku *7.2.2*. Iz vseh detektiranih daljic enega roba na koncu sestavimo táko, da se dejanskemu robu najbolj prilega.

6.2.4 Detekcija vogalov

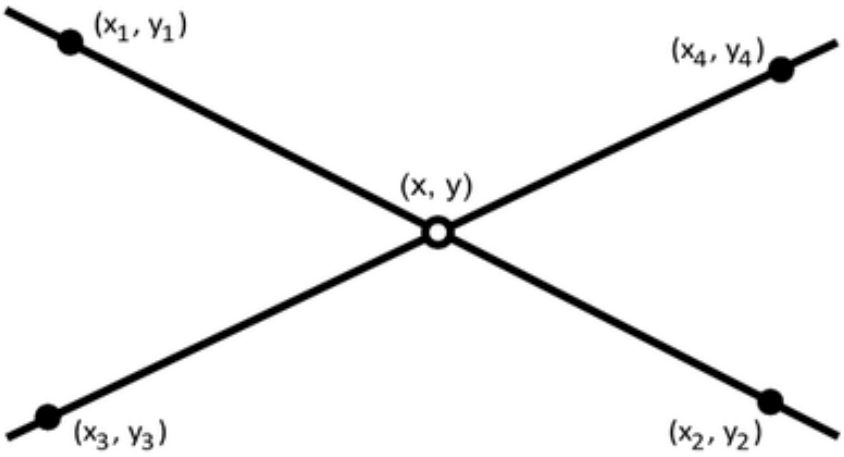
Ko so znani vsi robovi, se določijo premice, ki vsebujejo te robove. Presečišča premic določajo vogale projekcijske slike. Presečišče prvega in tretjega roba določa levi zgornji vogal, presečišče drugega in tretjega roba desni zgornji vogal, presečišče drugega in četrtega roba desni spodnji vogal, presečišče prvega in četrtega roba pa levi spodnji vogal (slika *10*). Izhod funkcije `cvHoughLines2` so robovi, predstavljeni kot koordinate začetne in končne točke. Točko presečišča dveh premic izračunamo po naslednji formuli:

$$P(x,y) = \left(\frac{(x_1y_2 - y_1x_2)(x_3 - x_4) - (x_1 - x_2)(x_3y_4 - y_3x_4)}{(x_1 - x_2)(y_3 - y_4) - (y_1 - y_2)(x_3 - x_4)}, \frac{(x_1y_2 - y_1x_2)(y_3 - y_4) - (y_1 - y_2)(x_3y_4 - y_3x_4)}{(x_1 - x_2)(y_3 - y_4) - (y_1 - y_2)(x_3 - x_4)} \right), \quad (6.2)$$

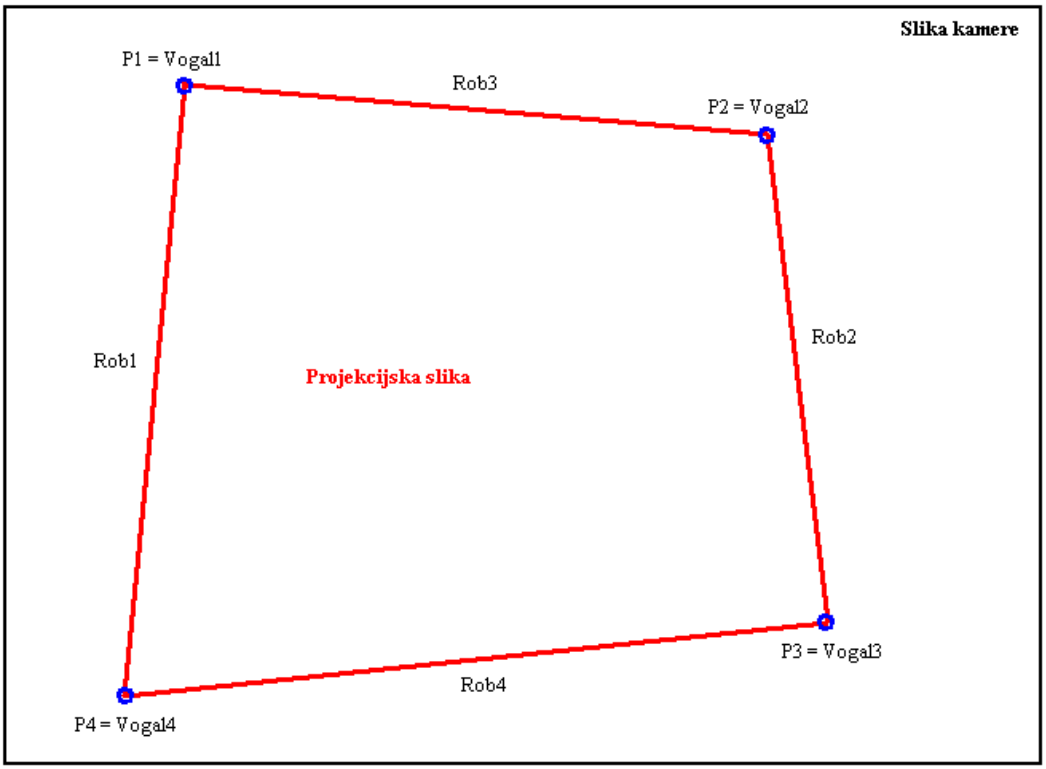
kjer sta (x_1, y_1) in (x_2, y_2) točki na prvi premici, (x_3, y_3) in (x_4, y_4) pa točki na drugi premici (slika *9*). Če sta imenovalca koordinat x in y enaka nič, pomeni, da se premici ne sekata, torej sta vzporedni.



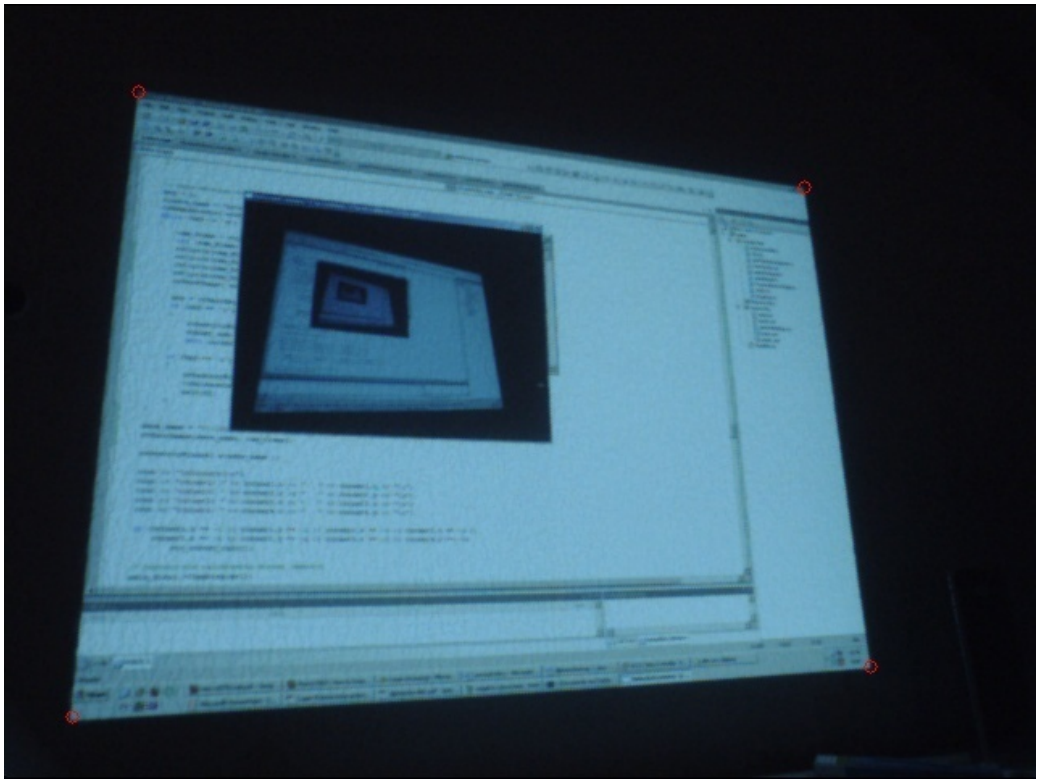
Slika 8: Slike robov, detekriranih s Houghovo transformacijo



Slika 9: Presečišče dveh premic



Slika 10: Označeni robovi in vogali projekcijske slike



Slika 11: Detektirani vogali

6.2.5 Homografija

Detektirani vogali projekcijske slike nam dajo štiri zahtevane korespondenčne točke (slika *11*). Korespondenčni pari točk so predstavljeni v tabeli *1*. S pomočjo funkcije OpenCV `cvFindHomography` se izračuna transformacijska matrika med projekcijsko sliko in sliko kamere [1, 12].

Po končanem izračunu nam vmesnik pokaže avtomatsko ali ročno določene vogale projekcijske slike. Če z rezultatom nismo zadovoljni, lahko detekcijo vogalov in izračun transformacijske matrike ponovno sprožimo. V nasprotnem primeru nadaljujemo s kalibracijo laserske pike.

Točka na projekcijski sliki	Točka na sliki kamere
P_1(0, 0)	Vogal1
P_2(širina_zaslona, 0)	Vogal2
P_3(širina_zaslona, višina_zaslona)	Vogal3
P_4(0, višina_zaslona)	Vogal4

Tabela 1: Korespondenčne točke

6.3 Detekcija laserske pike

Metoda segmentacije, opisana v podpoglavju *3.1*, je v veliki meri odvisna od praga za upragovanje slik. Ker se svetlobni pogoji med izvajanjem vmesnika lahko spreminjajo, bi bilo potrebno ta prag določevati sproti, kar pa zahteva dodaten procesorski čas. V nasprotnem primeru pride do preveliko napačnih detekcij. Problemi določevanja praga so opisani v razdelku *7.2.1*.

Za detekcijo laserske pike lahko uporabljamo metodo ločevanja ozadja le v primeru, če slika kamere, z izjemo laserske pike, ostaja ista ali pa se spreminja le minimalno skozi celotno izvajanje vmesnika. Uporabna je torej samo takrat, kadar naš vmesnik uporabljamo za nadzor določene aplikacije brez projektorja. Kamero usmerimo na določeno površino, vmesnik pa nam služi za prepoznavanje kretenj, s katerimi ukazujemo aplikaciji v teku. V običajnem delovanju vmesnika, to vemo, pa se ozadje konstantno spreminja. Že če pogledamo uporabo vmesnika pri predstavitvi, ugotovimo, da vsaka prosojnica predstavlja svoje ozadje, torej zanjo potrebujemo

svojo referenčno sliko (podpoglavje *3.2*). To moramo zajeti brez prisotnosti laserske pike, kar pa je s strani uporabnika praktično nemogoče doseči.

Ker pri metodi povprečenja ne pride do zgoraj navedenih problemov, sem se odločil implementirati le-to. Postopek izvajanja te metode je opisan v podpoglavju *3.3*. Preden vmesnik preide v glavno zanko izvajanja, mora kalibrirati lasersko piko. Med izvajanjem kalibracije vmesnik poišče vrednost najsvetlejšega slikovnega elementa in shrani prvo zgodovino slik. Pomembno je, da je laserska pika svetlejša od te vrednosti za vsaj nek prag, določen v programu. Da to dosežemo, moramo včasih spremeniti parametre kamere ali projektorja (razdelek *7.1.1*). Po tej kalibraciji preide izvajanje vmesnika v glavno zanko.

6.4 Glavna zanka

Znotraj glavne zanke se za vsako zajeto sliko izvedejo tri stvari:

- na zajeti sliki se detektira laserska pika,
- iz poti detektirane pike vmesnik poskuša sestaviti kretnjo,
- glede na kretnjo se izvede pripadajoča akcija.

Če je na zajeti sliki prisotna laserska pika, detektor kot rezultat vrne koordinati točke v koordinatnem sistemu kamere, ki predstavljata to lasersko piko. S pomočjo transformacijske matrike se ta točka preračuna v koordinate miške na projekcijskem zaslonu. Glede na pogoje, opisane v 4. poglavju, se izvede pripadajoča akcija miške (premik, levi klik, dvojni levi klik, vleci ali spusti). Desni klik in drsenje se izvedeta samo v primeru, če je bila predhodno zaznana kretnja za njuno aktivacijo. Kretnje se zaznavajo sproti in so omejene s časovnim pragom. Kadar je laserska pika prisotna dlje kot ta prag, pomeni, da ne gre za kretnjo. Vmesnik prepozna tudi kretnje, ki vključujejo periferni del projekcijskega zaslona (podpoglavje *5.2*). Vsaka laserska pika se sproti primerja s prejšnjo zaznano. Glede na razmerje med njima (podpoglavji *5.1* in *5.2*) se tvori trenutna kretnja. Ko laserski žarek ugasne, je sestavljena celotna kretnja in izvede se pripadajoča akcija. Tu gre, poleg aktivacije desnega gumba miške ter drsenja, ponavadi za pritisk kombinacije tipk na tipkovnici (tabela *2*). Katera tipka oz. kombinacija tipk se pritisne, je odvisno od tega, katero aplikacijo trenutno nadziramo z vmesnikom. V osnovi gre za nadzor miške. Vse ostale nadzore moramo naknadno vklopiti. Ob vsakem obhodu zanke se preverja, ali

je bila pritisnjena katera tipka na tipkovnici, kar omogoča dodaten nadzor nad vmesnikom (vklop in izklop nadzora nad aplikacijami). S pritiskom na tipko ‘1’ se prične nadzor aplikacije Windows Media Player, s pritiskom na tipko ‘2’ pa aplikacije Microsoft Office PowerPoint. Z malo dodatnega programiranja je možno dodati nadzor še za druge aplikacije.

Kretnja	Akcija		
	Windows	Windows Media Player	Microsoft Office PowerPoint
oUiUiRoR[2]	Aktivacija/deaktivacija desnega gumba miške		
oDiD	Drsenje navzdol		Naslednja prosojnica
oUiU	Drsenje navzgor		Prejšnja prosojnica
oDiDiRoR	ALT + F4		
oDiDiLoL	ESC		
oDiDiUoU		Igraj/premor	
oLiLiUoU		Stop	
iUoU		Povišaj glasnost	
iDoD		Znižaj glasnost	
iRoR		Naslednja skladba	Naslednja prosojnica
iLoL		Prejšnja skladba	Prejšnja prosojnica
iLiR			Izberi pisalo
iRiL			Izberi kazalec
oDiDoD			Zaženi predstavitev

Tabela 2: Kretnje in njihove pripadajoče akcije

V glavni zanki se izvajata tudi dva časovnika. Prvi se sproži, ko laserski žarek posveti na sliko kamere, in se ustavi, ko laserska pika ni več prisotna na sliki. Čas, kako dolgo je laserska pika prisotna na sliki, je pomembna informacija. Njena vrednost se primerja s pragom za ločevanje

[2] Kretnje so predstavljene z nizi znakov. Veliki znaki predstavljajo smer kretnje ('L' – levo, 'R' – desno, 'U' – gor, 'D' – dol), majhni znaki pa lokacijo kretnje ('i' – znotraj projekcijskega zaslona, 'o' – zunaj projekcijskega zaslona). Kretnja 'oDiD' pomeni prehod ven-not preko zgornjega roba projekcijskega zaslona.

med klikom ter akcijo vleci in spusti. Drugi časovnik meri čas, kako dolgo je laserski žarek izklopljen. To informacijo potrebujemo, kadar preverjamo, ali gre za klik. Oba časovnika se sprožita in ustavita za vsak vklop in izklop laserskega žarka posebej. Več o teh časovnikih in pripadajočih pragih je opisano v podpoglavju *4.2*.

Posebno pozornost si zasluži drsenje, kajti gre za primer akcije, ki se izvede ob določeni kretnji, kot parameter pa vzame dolžino poti kretnje. Drsenje navzdol se izvede, kadar naredimo z lasersko piko prehod ven-not (podpoglavje *5.2*) preko zgornjega roba. Ko pika prestopi rob, se prične meriti dolžina njene poti v smeri *y* znotraj projekcijskega zaslona. Daljša kot je pot, večkrat se bo zavrtel sledilni koleščk. Na enak način je implementirano drsenje navzgor. Sprožimo ga s prehodom ven-not preko spodnjega roba projekcijskega zaslona.

6.5 Uporaba vmesnika brez projektorja

Vmesnik lahko uporabimo tudi pri nadziranju aplikacij, kjer ne potrebujemo vizualne povratne informacije (npr. avdio predvajalnik), ali pa kot dodatek za prepoznavanje kretenj laserskega žarka. V teh primerih ne potrebujemo projektorja.

Po zagonu vmesnika izberemo način delovanja brez projektorja s pritiskom na tipko ‘c’. Ta način delovanja preskoči določanje transformacijske matrike, kot je opisano v podpoglavju *6.2*. Kamero moramo usmeriti tako, da njeno vidno polje zajema ravno površino (npr. steno). Če želimo uporabljati vmesnik za prepoznavanje kretenj, moramo vseeno določiti robove, preko katerih se bodo kretnje izvajale. Robovi se določijo samodejno. Levi in desni rob sta za petino širine slike kamere pomaknjena navznoter, zgornji in spodnji rob pa za petino višine slike kamere. Tudi v tem načinu delovanja moramo kalibrirati lasersko piko (podpoglavje *6.3*). V glavni zanki se koordinate detektirane laserske pike, preračunane v koordinate miškinega kazalca, ne uporabljajo za nadzor nad miško (premiki in kliki), ampak samo za prepoznavanje kretenj. Sámo prepoznavanje kretenj in izvajanje pripadajočih akcij je enako, kot je opisano v podpoglavju *6.4*. Kretnje, ki vključujejo periferni del projekcijskega zaslona, težko izvedemo, ker ne vemo, kje točno se nahajajo robovi projekcijskega zaslona.

7 Rezultati

V tem poglavju bom kritično ocenil razvit vmesnik. Ocena temelji na določenih kriterijih ter na testiranju vseh delov vmesnika, ki lahko vplivajo na rezultate. Najprej naj omenim, da je delovanje vmesnika v veliki meri odvisno od uporabljene opreme, zato je v prvem podpoglavju (*7.1*) opisana oprema, ki sem jo uporabljal pri izdelavi in testiranju vmesnika. Delovanje vmesnika pa je odvisno tudi od same implementacije, torej od programskega jezika, uporabljenih knjižnic in samih metod, ki se izvajajo med delovanjem vmesnika.

7.1 Oprema

Za uporabo vmesnika potrebujemo na osebni računalnik povezana projektor in (spletno) kamero ter laserski kazalnik. Vmesnik je implementiran za uporabo na operacijskem sistemu Windows. Za razvoj in testiranje vmesnika sem uporabljal spodaj opisano opremo.

7.1.1 Strojna oprema

Računalnik, na katerm je potekala implementacija, poganja procesor Intel Core2 Duo E8400 s frekvenco 3,00GHz. Vsebuje 3,5 GB pomnilniškega prostora.

Projiciranje se je izvajalo s projektorjem NEC VT570, ki omogoča prikaz barvnih slik in videoposnetkov v ločljivost od XGA (1024 × 768) do ločljivosti UXGA (1600 × 1200) z uporabo tehnologije Advanced AccuBlend.

Za zajem posnetkov sem uporabljal kamero Playstation Eye. Gre za spletno kamero proizvajalca Sony Computer Entertainment, namenjeno igralni konzoli PlayStation 3. Omogoča zajem slik ločljivosti 640 × 480 slikovnih elementov pri 60 Hz ali 320 × 240 slikovnih elementov pri 120 Hz. Čeprav uradni gonilniki za uporabo v operacijskem sistemu Windows ne obstajajo, so le-ti dostopni preko posameznih razvijalcev, kot je Alexander Popovich.

Tako projektor kot spletna kamera imata nekaj parametrov, ki vplivajo na delovanje vmesnika. Kameri lahko spreminjamo tri parametre: občutljivost senzorja (ang. gain), osvetlitev (ang. exposure) in nastavljanje beline (ang. white balance). Kamera v osnovi te parametre med

snemanjem samodejno prilagaja, kar povzroča težave tako pri samodejnem določanju praga za upragovanje kot pri detekciji laserske pike. Najboljše rezultate izvajanja vmesnika dobimo, če vse parametre določimo ročno. Višja kot je občutljivost senzorja, svetlejša je slika, na njej pa je tudi več šuma. Dobro jo je nastaviti na majhno vrednost, da laserska pika na sliki dovolj izstopa. Prav tako je dobro vrednosti ostalih dveh parametrov nastaviti na manj kot polovico njune največje vrednosti. Na projektorju je priporočljivo rahlo potemniti projekcijsko sliko, saj lahko, zaradi njenih svetlih delov, pride do napačnih detekcij laserske pike.

Laserski kazalnik, uporabljen pri razvoju, oddaja rdečo svetlobo valovne dolžine 640-660 nm, ki je vidna do 50 m. Testiral sem tudi kazalnik, ki oddaja zeleno svetlobo valovne dolžine 530-545 nm. Vmesnik ni prilagojen za detekcijo več laserskih pik. Kadar z obema laserskima žarkoma istočasno posvetimo v vidno polje kamere, vmesnik enkrat detektira zeleno lasersko piko kot najsvetlejšo, drugič pa rdečo. Žarka, ki ga oba kazalnika oddajata, sta torej za metodo povprečenja enako intenzivna. Pri uporabi zato med njima ne prihaja do razlik.

7.1.2 Programska oprema

Vmesnik sem implementiral v programskem jeziku C++, za samo programiranje pa sem uporabljal orodje Microsoft Visual Studio 2008. Za potrebe procesiranja slik sem uporabil knjižnico OpenCV (Open Source Computer Vision Library). Je odprtokodna knjižnica (licenca BSD – Berkley Software Distribution), ki se uporablja na področju računalniškega zaznavanja, in sicer na več platformah (FreeBSD, Linux, Mac OS, Windows). Namenjena je predvsem procesiranju slik v realnem času. Poleg osnovnih, nizkonivojskih metod, kot so upragovanje, filtriranje in statistika slik, je osredotočena predvsem na implementacijo visokonivojskih algoritmov za kalibracijo, detekcijo značilk, sledenje, analizo oblik, analizo gibanja, 3D rekonstrukcijo, segmentacijo in prepoznavanje objektov. Razvil jo je Intel, zato je tudi optimizirana za procesorje Intelove arhitekture, še posebej za tehnologijo MMX. Gre za uporabo in nadgradnjo Intelove knjižnice za procesiranje slik (Intel Image Processing Library – IPL) [9]. Vsebuje tudi svoj prilagodljiv mehanizem za rokovanje z napakami, ki omogoča, da napaka prekine program in izpiše razlog, ali pa se napaka shrani, program pa se nadaljuje. Knjižnica je napisana predvsem za uporabo v programskih jezikih C in C++.

Uporaba te knjižnice je odločilno vplivala na čas, ki sem ga porabil za implementacijo vmesnika. Njena uporaba je zelo intuitivna. Ukazi, kot so `cvHoughLines2`, `cvSmooth` in `cvFindHomography`, so mi bili v veliko pomoč tako pri programiranju kot pri razumevanju samih problemov (v [9] je opisana tudi teorija, ki stoji za temi ukazi). Ker so ukazi v knjižnici optimizirani, porabijo manj procesorskega časa in pomnilniškega prostora, kot če bi jih implementiral sam.

7.2 Parametri vmesnika

Znotraj programa je veliko nastavljivih parametrov. Spodaj so opisani tisti, katerih vrednost odločilno vpliva na izvajanje vmesnika.

7.2.1 Prag za upragovanje

Kot je že bilo omenjeno, se njegova vrednost giblje med 0 in 255. Od njega je neposredno odvisna detekcija robov projekcijskega zaslona, posredno preko transformacijske matrike pa preračunane koordinate kazalca miške glede na lasersko piko, torej natančnost izvajanja vmesnika. Ta prag je pomemben samo v primeru, kadar želimo, da vogale projekcijskega zaslona vmesnik detektira sam. S tem se tudi prag določi samodejno (razdelek *6.2.2*).

Kadar se zgodi, da so vogali detektirani narobe, je prag prenizek ali previsok. V tem primeru moramo preveriti postavitev kamere in njene nastavitve. Zgodi se namreč lahko, da kamera fokusira njej bližje robove projekcijskega zaslona, oddaljenih pa ne zazna (previsok prag). Kamero postavimo tako, da je njena oddaljenost od vseh štirih robov projekcijskega zaslona približno enaka, in kalibracijo ponovno poženemo. Do napačnih detekcij prihaja tudi, kadar kamera samodejno določa svoje parametre. Rešitve tega problema so opisane v razdelku *7.1.1*. Kadar med kalibracijo program preneha z delovanjem, pomeni, da je prag dosegel maksimalno vrednost (255). Do te situacije pride zaradi samodejnega nastavljanja parametrov kamere. Možno pa je tudi, da je prostor preveč osvetljen in projicirani beli robovi projekcijskega zaslona ne izstopajo dovolj. Temu se izognemo tako, da prostor zatemnimo. Vse omenjene probleme pa lahko rešimo preprosto tako, da vogale projekcijskega zaslona določimo ročno.

7.2.2 Parametri Houghove transformacije

Že v podpoglavju *2.3* smo ugotovili, da ima funkcija za iskanje premic (`cvHoughLines2`) več parametrov. Da lahko določimo vrednost teh parametrov, se moramo najprej odločiti za eno izmed variant transformacije. Sam sem izbral verjetnostno, ker je učinkovitejša, kadar slika vsebuje nekaj dolgih linearnih segmentov [12]. V nadaljevanju določimo najmanjšo dolžino črte, ki jo funkcija vrne. Vemo, da iščemo robove projekcijskega zaslona in da bo projekcijski zaslon zavzemal velik del slike kamere. Funkcijo zato nastavimo tako, da nam vrne vse črte, ki so daljše od šestine višine slike kamere. Nazadnje določimo razmak med dvema segmentoma črte, da ju funkcija združi. Ker so projicirani robovi projekcijskega zaslona močno poudarjeni, obstaja majhna verjetnost, da jih bo funkcija za detekcijo premic razdelila na več segmentov. Če se to že zgodi, so ti segmenti blizu skupaj, zato ta parameter nastavimo na neko majhno vrednost (npr. 15 slikovnih elementov). Kot rezultat dobimo črte, predstavljene s koordinatami njihovih začetnih in končnih točk.

7.2.3 Parametri detekcije laserske pike

Izbrana metoda za detekcijo laserske pike (metoda povprečenja) vsebuje dva parametra, ki ju lahko spreminjamo, in sicer velikost zgodovine ter prag za klasifikacijo. Velikost zgodovine določa, koliko preteklih slik bo vedno shranjenih. Iz teh slik se pri vsakem obhodu skozi glavno zanko računa povprečna vrednost vsakega slikovnega elementa. Takoj vidimo, da več kot je slik v zgodovini, več pomnilniškega prostora porabimo, hkrati pa potrebujemo več procesorskega časa za izračun povprečja. Tega parametra zato ne nastavimo na veliko vrednost. Pri pragu za klasifikacijo določamo, za koliko naj bi bila laserska pika svetlejša od ostalih delov slike kamere, da jo detektor klasificira kot prisotno. Zadovoljive rezultate sem dosegel že pri nizkem pragu (10).

7.2.4 Parametri glavne zanke

Parametri, ki se uporabljajo pri simulaciji levega klika miške, so opisani v podpoglavju *4.2*. Tu gre za prostorsko in časovno omejevanje. Klik se mora izvesti v določeni okolici trenutno postavljene laserske pike (100 slikovnih elementov) in znotraj določenega časovnega okvirja, predstavljenega z dvema parametroma. Prvi določa, kako hitro moramo laserski žarek vklopiti nazaj na isti lokaciji, da ta vklop smatramo kot pritisk levega gumba miške, drugi pa, kako hitro

moramo žarek zopet izklopiti, da se ne aktivira akcija vleci in spusti. Da bi bila uporaba laserskega kazalnika čimbolj podobna uporabi miške, sem s testiranjem prišel do naslednjih vrednosti parametrov: ponovni vklop žarka se mora izvesti prej kot v pol sekunde, ponovni izklop, da izvedemo klik, pa prej kot v 0,16 sekunde. Če prekoračimo zadnji prag, se prične izvajati akcija vleci. Ta prag ne sme biti veliko večji, saj nam določa tudi, kako dolgo moramo z laserskim žarkom svetiti na projekcijsko površino, da se začne izvajati premik kazalca miške.

Pri prepoznavanju kretenj moramo najprej določiti prag, ki predstavlja minimalno razdaljo, ki jo mora prepotovati laserska pika med dvema sosednjima slikama, da premik smatramo kot del kretnje. S tem določimo hitrost, s katero se morajo izvajati kretnje. To hitrost moramo prilagoditi hitrosti oz. frekvenci zajemanja slike s kamero. Če jo nastavimo na preveliko vrednost, vmesnik velikokrat ne detektira kretenj zunaj roba projekcijskega zaslona, saj je periferni del ponavadi dosti manjši od projekcijskega zaslona. Sam sem to vrednost nastavil na 30 slikovnih elementov. Poleg tega vemo, da izvedene kretnje nikoli niso povsem ravne. Definirati moramo še en prag, ki določa, za koliko slikovnih elementov lahko kretnja odstopa od ravne, da jo še vedno zaznamo kot ravno. V osnovi je ta vrednost nastavljena na 120 slikovnih elementov. Kot je opisano v podpoglavju *5.1*, je prepoznavanje kretenj omejeno tudi časovno. Kadar je laserska pika na sliki prisotna dlje kot 3 sekunde, predvidevamo, da ne gre za kretnjo.

7.3 Kriteriji

Da lahko kritično ocenimo uporabnost vmesnika, potrebujemo določene kriterije. Najboljši pokazatelji uspešnosti delovanja so naslednji trije: zanesljivost, latenca in natančnost [2]. Katero opremo sem uporabljal pri ocenjevanju in kako so nastavljeni njeni parametri, je opisano v podpoglavju *7.1*, kakšno vrednost imajo programski parametri znotraj aplikacije, pa v podpoglavju *7.2*.

7.3.1 Zanesljivost

Nanaša se na razmerje med slikami, na katerih se laserski žarek pojavi, in slikami, na katerih je žarek detektiran. Za ocenjevanje zanesljivosti sem implementiral dodatek k vmesniku, ki na sliki kamere v realnem času vidno obkroži lasersko piko, če je le-ta detektirana. Pri opazovanju te slike sem ugotovil, da je laserska pika vedno pravilno detektirana, če so svetlobni pogoji v

prostoru primerni. To pomeni, da v vidnem polju kamere ni direktne sončne svetlobe ali bleščanja od projekcijske površine. Na kratko, laserska pika mora biti na sliki kamere svetlejša od najsvetlejšega dela slike. Le tako lahko zagotovimo stoodstotno zanesljivost. V primeru, da določeni deli projekcijske slike preveč izstopajo (so presvetli), lahko to rešimo s spreminjanjem parametrov na projektorju. Ponavadi je že dovolj, da projekcijsko sliko rahlo zatemnimo, kar omogoča vsak projektor.

7.3.2 Latenca

Latenca se nanaša na razliko v času, ko laserski žarek posveti na projekcijsko površino in pripadajočo akcijo na zaslonu (npr. premik kazalca miške). Ko opazujemo, kdaj se bo kazalec miške premaknil do laserske pike, moramo upoštevati časovni prag, ki se uporablja za razlikovanje med akcijama klik ter vleci in spusti (podpoglavje *4.2*). Prvi premik kazalca torej traja minimalno ta določen prag (0,16 sekunde v mojem primeru) in maksimalno ta prag plus čas enega obhoda glavne zanke. V povprečju se en obhod zanke izvede v 0,042 sekunde (42 ms), torej je maksimalna latenca 202 ms. Vsak naslednji premik kazalca se izvede v času enega obhoda zanke.

7.3.3 Natančnost

Natančnost opisuje razdaljo, merjeno v slikovnih elementih, med dejanskim laserskim žarkom in pripadajočimi koordinatami miškinega kazalca, preračunanimi z danim algoritmom. Meril sem jo tako, da sem laserski žarek usmeril na določen del projekcijskega zaslona, zajel sliko in na njej izmeril, koliko sta si narazen laserska pika ter konica kazalca miške. Glavni vzrok za nenatančnost preslikave leži v transformacijski matriki, saj je izračunana brez upoštevanja radialnega popačenja. V vogalih, kjer se korespondenčne točke točno ujemajo z vogali projekcijskega zaslona, deluje vmesnik zelo natančno. Bolj ko se oddaljujemo od vogala, manjša je natančnost, vendar odstopanje miškinega kazalca nikjer ne presega štirih slikovnih elementov. Omenjeno natančnost lahko zagotovimo samo, kadar ne pride do relativnih premikov med projekcijsko površino in kamero.

8 Zaključek

Uporaba vmesnika omogoča predavatelju, da z laserskim kazalnikom nadzoruje predstavitev od daleč. S tem ne zakriva pogleda na projekcijsko površino. Poleg tega mu za pomikanje med prosojnicami ni treba stopiti do računalnika in s tem prekiniti svoje koncentracije ter koncentracije poslušalcev. Vmesnik, poleg kontroliranja predstavitve, uporabniku omogoča tudi nadzor nad drugimi aplikacijami. Primer tega je kontrola avdio predvajalnika. S preprostimi kretnjami lahko uporabnik z laserskim kazalnikom zamenja predvajano skladbo, jo ustavi ali ponovno zažene, zviša ali zniža glasnost predvajanja … Možno je dodati tudi druge akcije za druge aplikacije.

Kalibracija kamere in projektorja se lahko izvede samodejno, kar uporabniku prihrani čas, uporabo vmesnika pa poenostavi. Seveda mora to potekati v pravilno osvetljenem prostoru. Največja pomankljivost implementiranega vmesnika je ravno občutljivost na svetlobne pogoje. V zelo svetlem prostoru in v prostoru, kjer se svetlobni pogoji zelo spreminjajo, postane uporaba nezanesljiva. Temu problemu se je težko izogniti. Ena izmed delnih rešitev bi bila, da bi v metodi za detekcijo laserske pike upoštevali točno določen spekter svetlobe, ki jo oddaja laserski kazalnik. V primerno osvetljenem prostoru in s primerno nastavljeno opremo pa vmesnik, s svojo dobro zanesljivostjo, natančnostjo, ki ne omejuje uporabe ter solidno latenco, poenostavi interakcijo človeka z računalnikom.

Izvajanje kretenj je dokaj intuitivno. Poteg z laserskim žarkom v desno recimo prestavi na naslednjo prosojnico, poteg v levo pa na prejšnjo. Določene kretnje, ki izvedejo akcijo, specifično za neko aplikacijo (npr. izbira pisala pri predstavitvi), pa se moramo naučiti. Temu bi se lahko izognili s vpeljavo nevronskih mrež za prepoznavanje kretenj. S tem bi povečali nabor vseh možnih kretenj, izboljšali zanesljivost prepoznavanja ter dodane nove kretnje lažje prilagodili posameznemu uporabniku.

Izvajanje glavne zanke je časovno omejeno s frekvenco zajemanja slik. Ta frekvenca je odvisna od kamere, ki jo uporabljamo, in je v določenih primerih lahko tako nizka, da onemogoča tekočo

uporabo vmesnika. Da bi se temu izognili, bi morali izvajanje glavne zanke razdeliti na dve niti, kjer bi prva samo zajemala slike, druga pa bi te zajete podatke procesirala.

Uporaba vmesnika je odvisna od velikega števila parametrov (parametri kamere, parametri projektorja, programski parametri). Najti enolično nastavitev, ki bi upoštevala vso možno opremo in vse možne situacije, je velik izziv.

Vmesnik je bil implementiran predvsem z namenom, da se preveri, ali je možno izdelati uporabnega. Zato uporabniku ni najbolj prijazen. Večino parametrov je možno spreminjati samo znotraj same kode programa. Potrebno bi bilo izdelati še grafični uporabniški vmesnik, ki bi omogočal spreminjanje parametrov kakor tudi dodajanje novih kretenj in povezovanje le-teh z določenimi akcijami.

Kazalo slik

Slika 1: Sistem za uporabo laserskega kazalnika namesto miške 4
Slika 2: Koraki metode segmentacije 11
Slika 3: Bločni diagram arhitekture za nadzor računalniškega pomnilnika in zaslona s kretnjami 20
Slika 4: Kretnje, ki vključujejo periferni del projekcijskega zaslona 22
Slika 5: Primer kretnje s parametrom 23
Slika 6: Detektirane črte kalibracijske slike samodejnega določanja praga za upragovanje 27
Slika 7: Binarne slike robov 28
Slika 8: Slike robov detekriranih s Houghovo transformacijo 29
Slika 9: Presečišče dveh premic 29
Slika 10: Označeni robovi in vogali projekcijske slike 30
Slika 11: Detektirani vogali 30

Kazalo tabel

Tabela 1: Korespondenčne točke 31
Tabela 2: Kretnje in njihove pripadajoče akcije 33

Viri

[1] G. Bradski, A. Kaehler, *Learning OpenCV*, O'Reilly Media, september 2008

[2] J. F. Lapointe, G. Godin, "On-Screen Laser Spot Detection for Large Display Interaction," v zborniku *IEEE International Workshop on Haptic Audio Enviroments and their Applications*, Ottawa, Ontario, Canada, oktober 2005, str. 72-76

[3] M. Lipanje, *Klavir za pešče – ustvarjanje glasbe z računalniškim vidom*, Univerza v Ljubljani, 2010

[4] D. R. Olsen Jr., T. Nielsen, *Laser pointer interaction*, Brigham Young University, Utah

[5] B. Shizuki, T. Hisamatsu, S. Takahashi, J. Tanaka, *Laser Pointer Interaction Techniques using Peripheral Areas of Screens*, University of Tsukuba, Japan

[6] R. Sukthankar, R. G. Stockton, M. D. Mullin, "Smarter Presentations: Exploiting Homography in Camera-Projector Systems," v zborniku *International Conference on Computer Vision*, 2001

[7] E. Trucco, A. Verri, *Introductory Techiques for 3-D Computer Vision*, Prentice Hall, 1998

[8] Z. Zhang, "A Flexible New Technique for Camera Calibration," v zborniku *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2000

[9] *Open Source Computer Vision Library – Reference Manual*, Intel Corporation, december 2000

[10] (1999) A Brief Overview of Gesture Recognition. Dostopno na: http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/COHEN/gesture_overview.html

[11] (2007) Laser Guided Mouse. Dostopno na: http://www.youtube.com/watch?v=AKVApHEss48

[12] (2009) OpenCV 2.0 C Reference. Dostopno na: http://opencv.willowgarage.com/documentation

[13] (2010) Feature (computer vision). Dostopno na: http://en.wikipedia.org/wiki/Feature_(computer_vision)

[14] (2010) Gesture recognition. Dostopno na: http://en.wikipedia.org/wiki/Gesture_recognition

[15] (2010) LaserGrabber – control your mouse using laser pointer. Dostopno na: http://www.youtube.com/watch?v=Zhk0KaNFFR8