UNIVERZA NA PRIMORSKEM
Fakulteta za matematiko, naravoslovje in informacijske tehnologije
Koper

Računalništvo in informatika – 1. stopnja

Duško Topić

Interaktivna vizualizacija tridimenzionalnih slik

Zaključna naloga

Mentor: doc. dr. Peter Rogelj

Koper, 2011

UNIVERSITY OF PRIMORSKA
Faculty of Mathematics, Natural Sciences and Information Technologies Koper

Computer Science – 1^{st} degree

Duško Topić

Interactive visualization of three-dimensional images

Final Project Paper

Mentor: doc. dr. Peter Rogelj

Koper, 2011

Zahvala

Posebej bi se zahvalil mentorju doc. dr. Petru Roglju za dragocene nasvete pri oblikovanju zaključne projektne naloge, pomoči pri zbiranju gradiva in implementaciji aplikacije.

Zahvalil bi se tudi staršem, ki so mi omogočili študij, bratu, punci ter vsem prijateljem, ki so me vzpodbujali, podpirali in mi vlili še dodatno motivacijo za študij, ko sem jo najbolj potreboval.

Duško Topić

V Kopru, 1. septembra 2011

Povzetek

V zaključni projektni nalogi je predstavljen pristop k reševanju problema prikazovanja tridimenzionalnih medicinskih slik. Pregledovanje $3D$ slik je namreč omejeno z $2D$ vpogledom, ki ga omogoča računalniški zaslon, zato zahteva dobro predstavo opazovalca. Predlagan pristop to predstavo izboljša z vpeljavo intuitivne povezave med $3D$ sliko in njeno $2D$ predstavitvijo. Prostor $3D$ slike navidezno postavimo v vidno polje kamere, kjer z interakcijsko ploščo določamo $2D$ ravnino prikaza. Na ta način uporabnik lahko spreminja svoj vpogled v sliko, saj ravnino, ki seka $3D$ sliko lahko premika, obrača oziroma nagiba v želeno smer. Posledično se na zaslonu sproti prikazuje prerez tridimenzionalne slike, kot ga določa postavitev interakcijske plošče pred kamero. Na ta način uporabnik pridobi predvsem na otipljivosti vsebine v sliki, saj z večjim občutkom analizira želene dele telesa, torej lahko na hiter in predvsem bolj učinkovit način odkrije morebitne bolezni oziroma druge anomalije.

Ključne besede

Računalniški vid, kalibracija kamere, geometrijske transformacije, linearna interpolacija, medicinske slike

Abstract

This article presents an approach to the problem of displaying three-dimensional medical images. Examining $3D$ images is limited with a $2D$ view provided by the computer screen. Therefore, a great visual perspective is required by the observer. The suggested approach improves comprehension of $3D$ images by introducing an intuitive relation between the $3D$ image and its $2D$ demonstration. The $3D$ area of the image is virtually placed inside the field of view of a camera, where the $2D$ plane for visualization can be defined by positioning an interactive board. This way the user can improve his insight into the image, because the plane that crosses the $3D$ image can be moved, rotated or tilted in the desired direction. Consequently, the image on the screen is continuously showing a cross-section of the three-dimensional image as determined by the interactive board positioned in front of the camera. In this manner the user gains on tangibility of the image content, enabling him to more efficiently and better analyse the desired parts of the body, and to discover potential diseases or other anomalies.

Keywords

Computer vision, camera calibration, geometric transformations, linear interpolation, medical images.

Kazalo

Uvod **1**
Opredelitev področja in opis problema 1
Namen in cilj projektne naloge 1
Omejitve pri reševanju problema 1
Napovednik 2

1 Teoretične osnove **3**
1.1 Geometrijska odvisnost med prostorom prizora in slike kamere 3
1.1.1 Kalibracija kamere 4
1.1.2 Geometrijske transformacije 7
1.2 Interpolacija 12
1.2.1 Linearna interpolacija 13
1.2.2 Trilinearna interpolacija 13

2 Postopki delovanja aplikacije **15**
2.1 Interakcijska plošča 15
2.1.1 Zaznavanje oglišč 16
2.2 Računanje transformacij 18
2.2.1 Preslikava oglišč v koordinatni sistem kamere 20
2.3 Potek interpolacije 20
2.4 Predstavitev aplikacije 21
2.4.1 Implementacija 21
2.5 Primer uporabe 23
2.5.1 Postopek kalibracije 24
2.5.2 Glavna aplikacija 25

Zaključek **27**

Literatura **29**

Uvod

Opredelitev področja in opis problema

Medicinske slike se uporabljajo predvsem v zdravstvenih ustanovah z namenom prikazovanja človeškega telesa oziroma posameznih delov le-tega. Uporabljajo se različne slikovne tehnike, ki so lahko v dvodimenzionalni (rentgen, ultrazvok itd.) ali tridimenzionalni obliki (MRI, CT, itd). Slednje so sicer sestavljene iz ti. rezin (ang. *slices*), ki so prav tako dvodimenzionalne, vendar več rezin skupaj tvori tridimenzionalno sliko.

Medicinske slike so običajno zapisane v ti. DICOM formatu (*Digital Imaging and Communications in Medicine*). To je standard za shranjevanje, pošiljanje in urejanje medicinskih slik [1]. Eden izmed večjih problemov medicinskih slik je težavnost izrisovanja tridimenzionalnih slik na zaslon, ki premore največ dve dimenziji. Večina aplikacij je omejenih le na izrisovanje posameznih rezin. Nekatere aplikacije sicer omogočajo prikaz slike v poljubni presečni ravnini, vendar uporabnik za predstavo o $3D$ vsebini slik potrebuje izkušnje in dober občutek o položaju in velikosti prikazanih struktur.

V naslednjih poglavjih bo predstavljen način, kako prikazati tridimenzionalno sliko na bolj interaktiven način, ki uporabniku omogoča lažje določanje prikazanega prereza slike. Gre za aplikacijo, ki s pomočjo poljubne kamere zazna interakcijsko ploščo, katero uporabnik premika v želen položaj in s tem določi presečno ravnino prikaza. Na primer, če uporabnik premakne ploščo navzgor, se bodo izrisovale zgornje rezine na sliki, če pa uporabnik spusti ploščo nižje, se temu ustrezno izriše določena rezina na spodnjem delu slike.

Namen in cilj projektne naloge

Namen projektne naloge je omogočiti analizo medicinskih slik na bolj učinkovit način, ki uporabniku nudi neposreden občutek o velikosti in legi struktur, prikazanih na sliki. Cilj naloge je ustvariti aplikacijo, ki bo s pomočjo kamere in interakcijske plošče na zaslon izrisovala želen prerez medicinske slike.

Omejitve pri reševanju problema

Največja omejitev pri reševanju navedenega problema je v tem, da je za izris želenega prereza medicinske slike potrebno imeti posebno orodje, s katerim nakažemo položaj prereza slike, ki ga želimo videti (v našem primeru bo to interakcijska plošča). To omejitev smo sicer minimizirali na ta način, da je zahtevano orodje praktično za uporabo in predvsem preproste oblike.

Naslednja pomembna omejitev je v tem, da uporabnik želi spremljati sliko v ti. realnem času (ang. *realtime*) - torej brez daljših zakasnitev, sicer lahko hitro pride do trenutka, ko uporabnik več ne prepozna lokacije, kjer naj bi se nahajal v sliki. To omejitev bo prav tako potrebno minimizirati z raznimi pohitritvami ter optimizacijo izvajanja aplikacije.

Napovednik

Projektna naloga se deli na dve poglavji, in sicer:

- Poglavje 1 bo opisovalo teoretične osnove, potrebne za razumevanje poteka postopkov izvajanja aplikacije. Najprej bo predstavljena geometrijska odvisnost med prostorom prizora in slike kamere. Tu bosta sprva opisana postopka kalibracije kamere in geometrijske transformacije, zatem bodo prikazane še osnove poteka linearne interpolacije.

- Poglave 2 bo predstavilo način delovanja aplikacije. Poglavje se bo pričelo z opisom, kako naj bi izgledala interakcijska plošča. Temu primerno bo sledil opis postopka zaznavanja oglišč na interakcijski plošči. Nato bodo metode, opisane v poglavju 1, ustrezno umeščene v aplikacijo. Na konkretnem primeru bosta predstavljena poteka postopka računanja geometrijskih transformacij in interpolacije. Opisno bo v odseku 2.4.1 predstavljena izvorna koda aplikacije, pa tudi opis poteka nekaterih funkcij, ki so ključnega pomena pri izvajanju aplikacije.

V zaključku bo poudarek na možnih izboljšavah aplikacije, ki bi lahko pripomogle pri hitrosti izvajanja oziroma področja, kjer bi lahko aplikacijo uporabili v podobne namene.

Poglavje 1

Teoretične osnove

Da bi razumeli delovanje aplikacije, je najprej potrebno razumeti nekaj osnovnih pojmov in postopkov, s katerimi se bomo srečali med njenim izvajanjem. Postopki se v splošnem delijo na dva dela, in sicer:

- določitev geometrijske odvisnosti med prostorom prizora in slike. V tem delu se bomo posvetili problematiki prikazovanja slike iz stvarnega $3D$ prostora na zaslon ter prepoznavanju ustreznih premikov interakcijske plošče,
- interpolacijo vrednosti želenega preseka ravnine v prostoru. V tem delu se bomo posvetili temu, kako izrisati presečno ravnino na učinkovit način, ne da bi čutili posledice pri hitrosti izvajanja aplikacije.

Oba tipa postopkov bosta podrobneje opisana v naslednjih razdelkih.

1.1 Geometrijska odvisnost med prostorom prizora in slike kamere

Ena izmed večjih težav pri prikazovanju želenega preseka ravnine je ta, da je težko določiti globino slike iz kamere, ki prostor prizora zajema v $2D$ obliki. Da bi bili zmožni omogočiti natančnejšo oceno globine slike, je najprej potrebno opraviti določene meritve s postopkom, imenovanim kalibracija kamere. To je postopek za določitev neznanih vrednosti zunanjih (ang. *extrinsic*) in notranjih (ang. *intrinsic*) parametrov kamere, s katerimi lahko $2D$ prostor slike povežemo s $3D$ prostorom prizora [2]. Tako notranji kot zunanji parametri bodo na primerih opisani v nadaljevanju.

Šele po uspešni in natančno opravljeni kalibraciji lahko določimo odvisnost med prostorom prizora in slike kamere. To pomeni, da lahko tudi v $2D$ obliki slike razberemo, kje se določen objekt nahaja v prostoru. V našem primeru nam to omogoča ugotovitve, kot so npr. za kolikšen faktor se je interakcijska plošča premaknila v določeno smer oziroma za kolikšen kot je nagnjena. Določanje odvisnosti izvajamo z geometrijskimi transformacijami.

1.1.1 Kalibracija kamere

Ustrezna kalibracija kamere je osnova za pravilno prikazovanje slike prostora v $2D$ načinu. Brez opravljene kalibracije lahko na kameri pride do določenih motenj pri prikazovanju slike prostora. Te motnje se velikokrat pojavijo v obliki raztegnjenih ali skrčenih predelov slike (največkrat na robovih) in so lahko posledica različnih tipov leč oziroma drugih komponent na kameri. Zato je potrebno pri postopku upoštevati naslednje predpostavke:

- položaj kamere se ne sme spreminjati, saj s tem lahko pride do napačnih dojemanj pri koordinatnem sistemu kamere,
- nastavitve kamere morajo ostati enake med celotnim postopkom kalibracije iz enakega razloga, navedenega v prejšnji točki,
- kalibracijski vzorec naj bo čim bolj prepoznaven. Priporočljiva je kombinacija črno-belih kvadratkov v obliki šahovnice, saj le-ta vsebuje največji kontrast med oglišči, katere je s tem najlažje razbrati,
- oglišča na kalibracijskem vzorcu morajo ležati v isti ravnini - potrebno je torej imeti ravno ploščo za kalibracijski vzorec, saj lahko v nasprotnem primeru kamera napačno dojame koordinate oglišča na kalibracijskem vzorcu.
- kalibracijski vzorec naj zasede večino vidnega polja kamere, na ta način se izboljša natančnost postopka,
- zaželjena je prisotnost naravne dnevne svetlobe, saj se v nasprotnem primeru lahko zgodi, da aplikacija ne razbere dejanskega kalibracijskega vzorca.

Med postopkom kalibracije so prisotni trije koordinatni sistemi, in sicer zunanji koordinatni sistem, ki predstavlja celoten zunanji prostor, koordinatni sistem kamere, ki predstavlja vidno polje kamere ter na podlagi zunanjega koordinatnega sistema zaznava morebitne spremembe položaja kalibracijskega vzorca. Tretji prisoten koordinatni sistem se imenuje koordinatni sistem interakcijske plošče, na podlagi katerega se pridobijo zunanji parametri kamere.

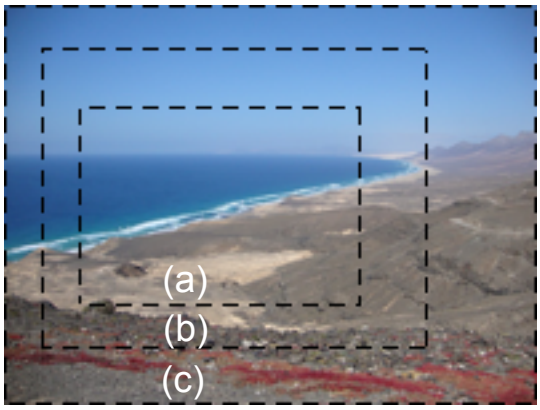
Postopkov kalibracije je več (npr. DLT - direktna linearna transformacija, Tsai metoda itd.), osredotočili se bomo na postopek kalibracije z ravninami, ki je uporabljen v naši aplikaciji. Postopek za pravilno izvajanje kot kalibracijski vzorec zahteva ploščo v obliki šahovnice, katero je potrebno premikati v različne smeri pod različnimi naklonskimi koti. Medtem postopek išče oglišča na šahovnici. Ta korak se ponavlja vse dokler postopek ne zazna vseh potrebnih oglišč. Podrobnejši potek postopka zaznavanja oglišč, vključno z opisom funkcij, uporabljenih pri implementaciji, je naveden v odseku 2.1.1.

Ko so koordinate oglišč znane, je naslednji in hkrati zadnji korak pridobitev vrednosti notranjih parametrov kamere na podlagi zajetih slik z različnimi položaji kalibracijskega vzorca. Za vsako sliko oceni parametre kamere, nato izbere srednjo vrednost kot končne vrednosti.

Notranji parametri

Notranja kalibracija povezuje slikovne koordinate pikslov s koordinatnim sistemom kamere in določa notranje parametre kamere, ki so [3]:

- Goriščna razdalja (ang. *focal length*) prikazuje oddaljenost gorišča od leče na kameri. Goriščna razdalja se v primeru povečanja slike (ti. "zoom") spremeni za določen faktor (lahko je npr. $2\times, 3\times$ faktor itd. (Slika 1.1)) [4].



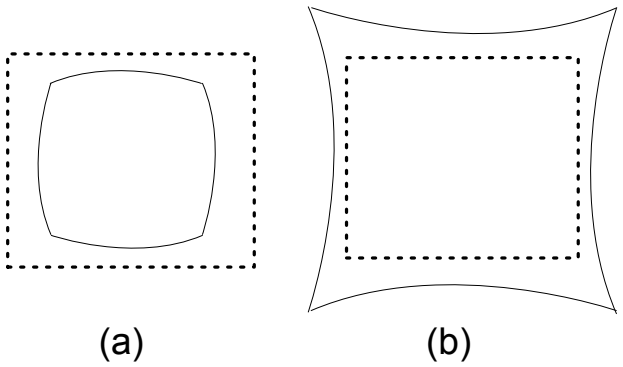
Slika 1.1: *Primer obsega vidnega polja kamere z goriščno razdaljo* $50mm$ *(a),* $35mm$ *(b) ter* $28mm$ *(c).*

- Center slike (ang. *principal point*) ni potrebno, da se nahaja na središču slike. Kot center slike lahko določimo npr. poljuben rob slike oziroma tisti del, okoli katerega želimo obračati sliko. Funkcije za določitev centra slike praviloma kot parameter prejmejo vektor dimenzije 2×1, v katerem prvi element predstavlja oddaljenost od izhodišča slike v smeri x osi, drugi element pa oddaljenost od izhodišča slike v smeri y osi.

- Koeficient nagiba (ang. *skew coefficient*) je vrednost (skalar), ki definira naklonski kot med x in y osjo koordinatnega sistema slike kamere.

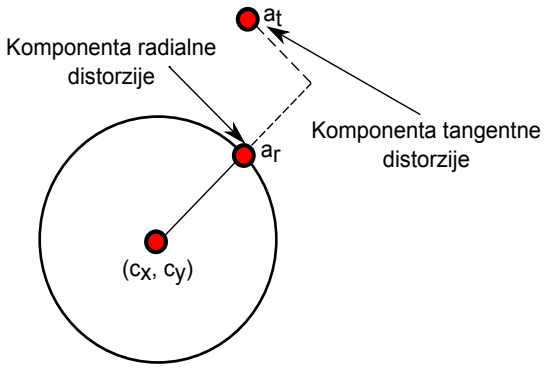
- Radialna distorzija (ang. *radial distortion*) oziroma radialno popačenje slike je še posebej opazno pri kamerah s širokokotnimi objektivi. Kvaliteta slike se sicer ohrani, spremeni se le oddaljenost točk od centra slike.

Oddaljenost se lahko poveča, v tem primeru slika dobi obliko "blazine" (ang. *pinchusion*) oziroma zmanjša, takrat dobi obliko "soda" (ang. *barrel*). Navedene spremembe ponazoruje slika 1.2. V našem primeru radialno distorzijo opišemo s polinomsko funkcijo 4. reda [5].



Slika 1.2: *Neprekinjena črta prikazuje učinek radialne distorzije v obliki "soda" (a) oziroma "blazine" (b). Prekinjena črta prikazuje obliko slike brez učinka radialne distorzije.*

- Tangentna distorzija (ang. *tangential distortion*) prav tako kot radialna distorzija pomeni popačenje slike. Razlika je le v tem, da se pri radialni distorziji spremeni oddaljenost točk od centra slike (c_x, c_y), pri tangentni pa se položaj točke spremeni pravokotno glede na popačeno točko radialne distorzije (Slika 1.3).



Slika 1.3: *Razlika med radialno in tangentno distorzijo.*

Notranje parametre nato zapišemo v matriko $\mathbf{M_{int}}$ dimenzije 3×3:

$$\mathbf{M_{int}} = \begin{bmatrix} f_{11} & \alpha f_{11} & p_{11} \\ 0 & f_{21} & p_{21} \\ 0 & 0 & 1 \end{bmatrix},$$

kjer je vektor $\mathbf{f}$ dimenzije 2×1 goriščna razdalja (prva komponenta je razdalja v smeri x osi, druga komponenta pa v smeri y osi), vektor $\mathbf{p}$ dimenzije 2×1 predstavlja center slike (prva komponenta je x, druga pa y koordinata) ter skalar α, ki predstavlja koeficient nagiba.

Koeficiente radialne in tangentne distorzije zapišemo ločeno v vektor $\mathbf{d}$ dimenzije 5×1:

$$\mathbf{d} = \begin{bmatrix} d_{11} \\ d_{21} \\ d_{31} \\ d_{41} \\ d_{51} \end{bmatrix}.$$

Zunanji parametri

Za razliko od notranje, zunanja kalibracija določa transformacijsko matriko, ki vsebuje parametre geometrijskih transformacij, na podlagi katerih aplikacija razbere položaj in usmerjenost interakcijske plošče. To stori tako, da razbere odstopanja od izhodiščne ravnine interakcijske plošče.

Zunanji parametri kamere opisujejo relacijo med presečno ravnino, določeno z interakcijsko ploščo in koordinatnim sistemom kamere, ki jih opišemo z geometrijsko transformacijo. Tako opisani zunanji parametri med drugim obsegajo tudi premik oziroma translacijo ter zasuk oziroma rotacijo in so podrobneje opisani v naslednjem razdelku, saj se s pomočjo teh parametrov izvajajo geometrijske transformacije.

Pridobljene zunanje parametre premika in rotacije lahko zapišemo v skupno matriko $\mathbf{M_{ext}}$ dimenzije 3×4:

$$\mathbf{M_{ext}} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_{11} \\ r_{21} & r_{22} & r_{23} & t_{21} \\ r_{31} & r_{32} & r_{33} & t_{31} \end{bmatrix},$$

kjer vrednosti prvih treh stolpcev predstavljajo koeficiente rotacijske matrike 3×3, vrednosti četrtega stolpca pa koeficiente vektorja translacije 3×1.

1.1.2 Geometrijske transformacije

Velikokrat pride do primera, ko je potrebno določen objekt na sliki zasukati za določen kot oziroma ga povečati, da bo na sliki bolj izpostavljen, ali pa ga preprosto premakniti na drugi položaj. Vse zgoraj navedene primere je možno opraviti s pomočjo geometrijskih transformacij.
Temeljne geometrijske transformacije so [6]:

- premik (ang. *translation*),
- skaliranje (ang. *scaling*) in
- zasuk (ang. *rotation*).

Poleg treh osnovnih je znanih še več drugih, kot je npr. afina oziroma perspektivna transformacija, vendar so za našo aplikacijo irelevantne, zato ne bodo podrobneje opisane.

V naši aplikaciji se geometrijske transformacije uporabljajo za opis relacij med koordinatnimi sistemi kamere, $3D$ slike in presečne ravnine oz. $2D$ slike. Za lažjo predstavitev je najprej potrebna obrazložitev in definicija nekaterih oznak.

Vektor $\mathbf{t_1} = [x_1, y_1, z_1, 1]^T$ naj predstavlja koordinate objekta v tridimenzionalnem prostoru pred transformacijo, vektor $\mathbf{t_2} = [x_2, y_2, z_2, 1]^T$ pa naj predstavlja nove koordinate objekta po opravljeni transformaciji.
Če je prisotnih več transformacij hkrati (npr. premik in zasuk), sta možna dva načina izračuna. Prvi način je, da se hkrati izračuna vsako vrsto transformacije. Bolj učinkovit način pa je, da se vse prisotne transformacije združi z množenjem posameznih elementarnih transformacij med seboj, pri takem pristopu pa se ne sme pozabiti na pomembnost vrstnega reda množenja matrik, zato je potrebno postaviti transformacijske matrike v pravilen vrstni red.

V nadaljevanju bodo predstavljene temeljne geometrijske transformacije.

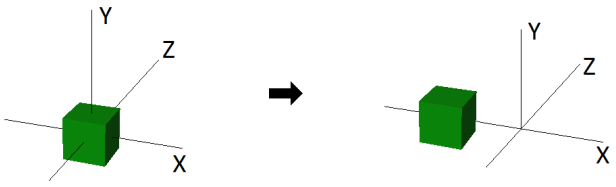
Premik

Recimo, da vektor $\mathbf{t_1}$ premaknemo v smeri x za vrednost d_x, v smeri y za vrednost d_y in v smeri z za vrednost d_z. Če je katera izmed vrednosti d_x, d_y ali d_z neničelna, pomeni, da je prišlo do premika (Slika 1.4). Nove koordinate vektorja $\mathbf{t_2}$ se izračuna na naslednji način:

$$\begin{aligned} x_2 &= x_1 + d_x \\ y_2 &= y_1 + d_y \\ z_2 &= z_1 + d_z \end{aligned}$$

Nove koordinate je možno dobiti tudi s pomočjo translacijske matrike $\mathbf{T}$, in sicer:

$$\mathbf{t_2} = \mathbf{T} \cdot \mathbf{t_1}, \qquad \mathbf{T} = \begin{bmatrix} 1 & 0 & 0 & d_x \\ 0 & 1 & 0 & d_y \\ 0 & 0 & 1 & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



Slika 1.4: *Primer 3D premika v smeri osi x [7].*

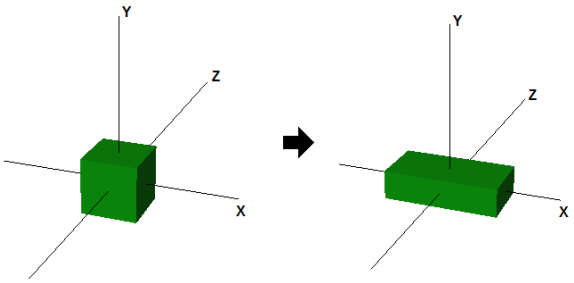
Skaliranje

Recimo, da vektor $\mathbf{t_1}$ raztegnemo oziroma skrčimo v smeri x za vrednost s_x, v smeri y za vrednost s_y ter v smeri z za vrednost s_z (Slika 1.5). Nove koordinate vektorja $\mathbf{t_2}$ dobimo s pomočjo naslednjih enačb:

$$x_2 = x_1 \cdot s_x$$
$$y_2 = y_1 \cdot s_y$$
$$z_2 = z_1 \cdot s_z$$

Nove koordinate je možno dobiti tudi s pomočjo matrike skaliranja $\mathbf{S}$, in sicer:

$$\mathbf{t_2} = \mathbf{S} \cdot \mathbf{t_1}, \qquad \mathbf{S} = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



Slika 1.5: *Primer 3D skaliranja [7].*

Čeprav skaliranje trenutno ni upoštevano nikjer v aplikaciji, ga je potrebno omeniti, saj je implementacija le-tega relativno preprosta, hkrati pa utegne biti uporabno v primeru, ko bi bilo potrebno določen predel slike povečati oziroma zmanjšati.

Zasuk

Pri zasuku oziroma rotaciji računanje novih koordinat vektorja $\mathbf{t_2}$ ni tako trivialen postopek kot recimo pri prejšnjih dveh vrstah transformacije. Rotacija pomeni zasuk okrog izhodišča koordinatnega sistema – če zasuka ne opravimo okrog izhodišča, moramo sliko pred tem (in za tem) ustrezno premakniti. Torej je potrebno ugotoviti, okoli katere osi je prišlo do zasuka, šele nato je potrebno izbrati ustrezno rotacijsko matriko, katero se zmnoži z vektorjem $\mathbf{t_1}$. Za lažje razumevanje postopka je potrebno definirati nekaj oznak, in sicer α, β in γ naj predstavljajo kote pri zasuku okrog x, y in z osi med vektorjema $\mathbf{t_1}$ in $\mathbf{t_2}$, matrika

$$\mathbf{R_x}(\alpha) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha & 0 \\ 0 & \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

naj predstavlja rotacijsko matriko pri zasuku okoli osi x, matrika

$$\mathbf{R_y}(\beta) = \begin{bmatrix} \cos\beta & 0 & \sin\beta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\beta & 0 & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

naj predstavlja rotacijsko matriko pri zasuku okoli osi y, matrika

$$\mathbf{R_z}(\gamma) = \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 & 0 \\ \sin\gamma & \cos\gamma & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

pa naj predstavlja rotacijsko matriko pri zasuku okoli osi z.

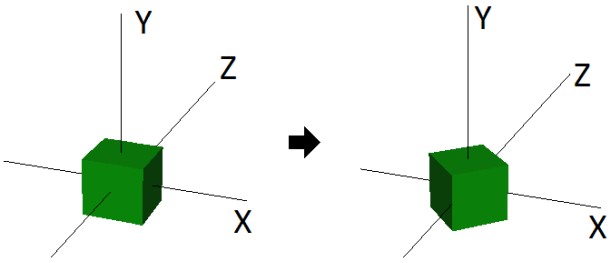
Recimo, da je prišlo do zasuka okoli osi y za kot β (Slika 1.6). V tem primeru uporabimo naslednji izračun:

$$\begin{aligned} x_2 &= x_1 \cdot \cos\beta + y_1 \cdot 0 + z_1 \cdot \sin\beta \\ y_2 &= x_1 \cdot 0 + y_1 \cdot 1 + z_1 \cdot 0 \\ z_2 &= -x_1 \cdot \sin\beta + y_1 \cdot 0 + z_1 \cdot \cos\beta \end{aligned} \tag{1.1}$$

Iz enačbe (1.1) je razvidno, da se vrednost koordinate y_2 ne spremeni, saj predstavlja središče zasuka in predstavlja ti. mirujočo točko (ang. *fixed point*).

Nove koordinate vektorja $\mathbf{t_2}$ je prav tako možno poiskati s pomočjo ustrezne rotacijske matrike, in sicer pri zasuku okoli osi y se uporabi naslednja enačba:

$$\mathbf{t_2} = \mathbf{R_y}(\beta) \cdot \mathbf{t_1},$$



Slika 1.6: *Primer 3D rotacije okoli osi y [7].*

kjer $\mathbf{R_y}(\beta)$ predstavlja rotacijsko matriko okoli osi y.

Zgoraj navedeni izračuni bi zadostovali le v primerih, če bi prišlo do rotacije zgolj okoli ene osi (podobni enačbi veljata tudi pri zasuku okoli osi x in z). Realno pa je pričakovati, da bo v našem primeru prišlo do rotacije pri vseh treh oseh oziroma vsaj dveh oseh. Da bi se izognili računanju treh posameznih zasukov, uporabimo ti. splošno matriko zasuka $\mathbf{R}(\alpha, \beta, \gamma)$, pri kateri se vnaprej dogovorimo glede določenega vrstnega reda zasukov (v našem primeru naj bo najprej zasuk okoli osi x, nato okoli osi y in na koncu še okoli osi z), saj matrično množenje ni komutativna operacija.

$$\mathbf{R}(\alpha, \beta, \gamma) = \mathbf{R_z}(\gamma) \cdot \mathbf{R_y}(\beta) \cdot \mathbf{R_x}(\alpha)$$

$$\mathbf{R}(\alpha, \beta, \gamma) = \begin{bmatrix} A & B & C & 0 \\ D & E & F & 0 \\ G & H & I & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \text{ kjer velja:}$$

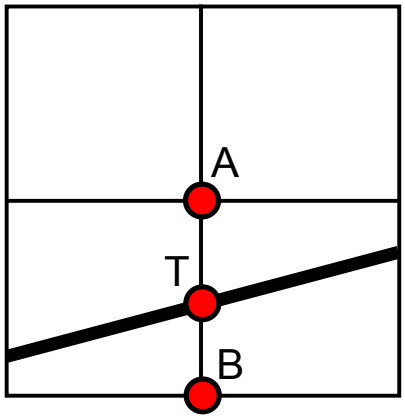
$$\begin{aligned} A &= \cos\gamma \cdot \cos\beta \\ B &= -\sin\gamma \cdot \cos\alpha + \cos\gamma \cdot \sin\beta \cdot \sin\alpha \\ C &= \sin\gamma \cdot \sin\alpha + \cos\gamma \cdot \sin\beta \cdot \cos\alpha \\ D &= \sin\gamma \cdot \cos\beta \\ E &= \cos\gamma \cdot \cos\alpha + \sin\gamma \cdot \sin\beta \cdot \sin\alpha \\ F &= -\cos\gamma \cdot \sin\alpha + \sin\gamma \cdot \sin\beta \cdot \cos\alpha \\ G &= -\sin\beta \\ H &= \cos\beta \cdot \sin\alpha \\ I &= \cos\beta \cdot \cos\alpha \end{aligned}$$

Sedaj lahko z enim izračunom dobimo koordinate vektorja $\mathbf{t_1}$, upoštevajoč vse prisotne rotacije:

$$\mathbf{t_1} = \mathbf{R}(\alpha, \beta, \gamma) \cdot \mathbf{t_2}$$

1.2 Interpolacija

Ko se pridobijo ustrezne relacije med prostorom prizora, sliko kamere ter interakcijsko ploščo, je naslednji korak izris prereza tridimenzionalne medicinske slike. Potrebno je izrisati tak prerez slike, kot se v prostoru v določenem trenutku nahaja interakcijska plošča. Ker pa je plošča v večini primerov nagnjena za določen kot (ni v popolnoma vodoravni legi), hitro pride do problema, ko v določeni točki interakcijske plošče nimamo znane vrednosti piksla iz medicinske slike (Slika 1.7). V tem primeru je potrebno poiskati neko vmesno vrednost sosednih pikslov. To opravimo s postopkom, ki se imenuje interpolacija.



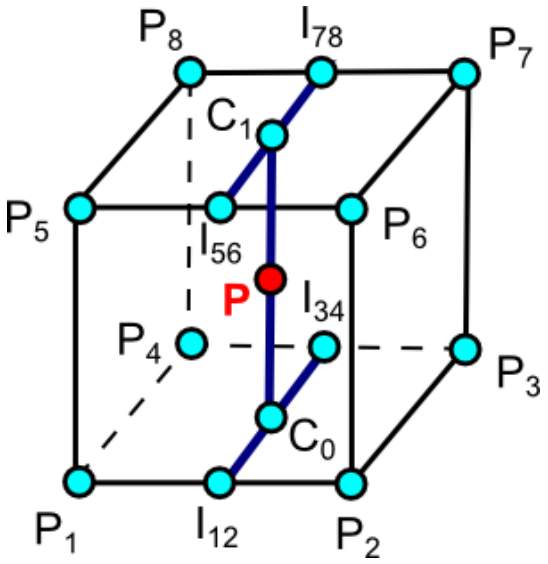
Slika 1.7: *Primer, ko v iskani točki T nimamo znane vrednosti piksla, znane so le vrednosti sosednih točk A in B. Potrebno je izbrati ustrezno vmesno vrednost.*

Interpolacija je numerična metoda, ki izračuna približno neznano vrednost iskane točke na podlagi vsaj dveh sosednih točk z znano vrednostjo [8]. Veliko uporabnost ima pri obdelavi slik, npr. izboljšanju ali zmanjševanju ločljivosti slike oziroma določitvi novih vrednosti na iskani točki.

Možnih postopkov interpolacije je več. Med drugim je postopek možno izvesti z metodo iskanja najbližjega soseda, kjer se izbere vrednost tiste sosedne točke, ki ima najkrajšo razdaljo od iskane točke z neznano vrednostjo. Naša aplikacija uporablja drugačen pristop, in sicer večkratno izvajanje linearne interpolacije, natančneje sedemkratno, v vseh treh dimenzijah prostora. Iz tega razloga se bomo osredotočili na podrobnejšo razlago postopka izvajanja linearne interpolacije.

Definirajmo sledeče oznake, ki jih bomo potrebovali v nadaljevanju. Položaj navedenih točk v prostoru prikazuje slika 1.8. Naj bo $P\left(x,y\right)$ točka z iskano vrednostjo p, točke

$$P_1\left(x_1,y_1\right), P_2\left(x_2,y_2\right), \ldots, P_8\left(x_8,y_8\right)$$



Slika 1.8: *Prikaz 3D slikovnega elementa z ustreznimi oznakami.*

pa naj bodo sosedne točki P na tridimenzionalni sliki s pripadajočimi vrednostmi $p_1, p_2, \ldots, p_8$. Slednje točke predstavljajo položaje $3D$ slikovnih elementov (vokslov) in so podane v slikovnih koordinatah (razdalja med sosednimi točkami je 1). Točke I_{12}, I_{34}, I_{56} in I_{78} naj predstavljajo koordinate pripadajoče interpolirane vrednosti med točkama P_1 in P_2, P_3 in P_4, P_5 in P_6 oziroma P_7 in P_8. Interpolirane vrednosti teh točk ustrezno poimenujmo z oznakami i_{12}, i_{34}, i_{56} ter i_{78}. Nazadnje še definirajmo točki C_0 ter C_1, ki vsebujeta interpolirano vrednost c_0 med točkama I_{12} in I_{34} oziroma vrednost c_1 med točkama I_{56} in I_{78}.

1.2.1 Linearna interpolacija

To je ena izmed najbolj osnovnih in s tem tudi najhitrejših metod interpoliranja, saj izračuna vmesno vrednost le na podlagi dveh znanih vrednosti točk.

Naj bo vrednost a razdalja med točkama P_1 in I_{12}. Interpolirano vrednost i_{12} točke I_{12} izračunamo z naslednjo formulo:

$$i_{12} = (1 - a) \cdot p_1 + a \cdot p_2 \tag{1.2}$$

1.2.2 Trilinearna interpolacija

Za razliko od linearne interpolacije, ki poišče vmesno vrednost na ravnini, trilinearna interpolacija poišče vmesno vrednost v prostoru. Sedaj je potrebno imeti znane vrednosti osmih točk, ki obkrožajo iskano točko z vseh

strani. Zelo priporočljiva je izbira sosednih točk, saj bližje kot so si točke med seboj, tem bolj natančen bo izračun. Postopek trilinearne interpolacije je v bistvu zgolj razširitev linearne interpolacije na tri dimenzije, v osnovi se še vedno uporablja enačba (1.2).

Izvedimo linearni interpolaciji med točkama P_1, P_2 in P_3, P_4, da dobimo iskani vrednosti i_{12} in i_{34}. V tem primeru naj bo vrednost a razdalja med točkama P_1 in I_{12} v smeri osi x, vrednost b pa razdalja med točkama P_3 in I_{34}, prav tako v smeri osi x.

$$i_{12} = (1 - a) \cdot p_1 + a \cdot p_2$$
$$i_{34} = (1 - b) \cdot p_3 + b \cdot p_4$$

Sedaj lahko interpoliramo še neznano vrednost c_0. Tokrat naj bo vrednost a razdalja med točkama I_{34} in C_0 v smeri osi y:

$$c_0 = (1 - a) \cdot i_{34} + a \cdot i_{12}$$

S podobnim izračunom interpolirajmo še vrednost točke c_1. V tem primeru naj bo vrednost a razdalja med točkama P_5 in I_{56} v smeri osi x, vrednost b pa razdalja med točkama P_7 in I_{78}, prav tako v smeri osi x. Enačba za izračun neznanih vrednosti točk I_{56} in I_{78}:

$$i_{56} = (1 - a) \cdot p_5 + a \cdot p_6$$
$$i_{78} = (1 - b) \cdot p_7 + b \cdot p_8$$

Sedaj lahko dobimo še interpolirano vrednost c_1. Vrednost a naj bo razdalja med točkama I_{78} in C_1 v smeri osi y:

$$c_1 = (1 - a) \cdot i_{78} + a \cdot i_{56}$$

Za konec je potrebno izvesti še zadnjo linearno interpolacijo. Vrednost a naj bo razdalja med točkama C_0 in P v smeri osi z. Interpolirati je potrebno vrednost med točkama C_0 in C_1:

$$p = (1 - a) \cdot c_0 + a \cdot c_1$$

Vrednost p je torej končni rezultat trilinearne interpolacije.

Poglavje 2

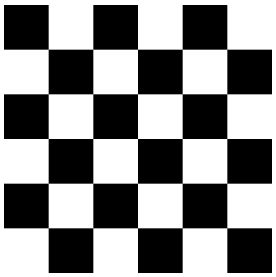
Postopki delovanja aplikacije

V tem poglavju bo poudarek na praktičnem delovanju aplikacije. Opisani bodo tako priporočljivi napotki za pravilno uporabo aplikacije (npr. oblika in dimenzije interakcijske plošče), kot tudi postopki delovanja pomembnejših funkcij, implementiranih v aplikaciji. Na koncu poglavja bo tudi slikovno ponazorjen postopek opravljanja kalibracije kamere in primer delovanja aplikacije.

2.1 Interakcijska plošča

Najpomembnejše orodje aplikacije je interakcijska plošča, katere oblika mora omogočati zaznavanje njenega položaja in orientacije v prostoru. To dosežemo tako, da je na plošči narisan vzorec, ki omogoča enostavno in nedvoumno lociranje plošče na sliki s kamere. V našem primeru je interakcijska plošča oblike šahovnice s 36-imi kvadratnimi polji velikosti 30×30 mm. Kombinacija črno-bele barve je najbolj primerna zato, ker igra pomembno vlogo pri zaznavi oglišč (stik robov med črnim ter belim kvadratnim poljem), saj jih je ravno zaradi velikega kontrasta najlažje zaznati in natančno locirati tudi v težjih pogojih (npr. slaba vidljivost, slaba osvetljenost itd.). Bolj ko interakcijska plošča ustreza zgoraj navedenim pogojem, natančneje bo aplikacija izrisala prerez tridimenzionalne slike.

Ključnega pomena je pravilna zaznava oglišč, saj na podlagi odstopanja od njihovih začetnih lokacij aplikacija pridobi transformacijsko matriko, ki določa geometrijsko relacijo med koordinatnim sistemom kamere in koordinatnim sistemom plošče ter s tem pove, kje se nahaja interakcijska plošča. Zato bi bilo primerno postopku zaznavanja oglišč posvetiti več pozornosti.



Slika 2.1: *Primer interakcijske plošče. Slika je pomanjšana, dejanska velikost vsakega polja je* 3×3 *cm.*

2.1.1 Zaznavanje oglišč

Oseba s prostim očesom brez večjih težav poišče oglišča na sliki, saj se mu zdijo samoumevne, medtem ko je tako "samoumevnost" nekoliko težje implementirati, saj je s pomočjo programske kode težko definirati vzorec oglišča. Za lažjo implementacijo je zato potrebno predpostaviti, kakšne vrste oglišč bodo prisotne na sliki. V našem konkretnem primeru je potrebno predpostaviti, da bomo za interakcijsko ploščo uporabljali vzorec šahovnice. S tem se postopek iskanja oglišč znatno olajša, saj v tem primeru aplikacija natančno pozna obliko vzorca, katerega mora zaznati.

Aplikacija zaznava oglišča na interakcijski plošči s pomočjo funkcije *cvFindChessboardCorners*, vgrajene v knjižnici OpenCV. Funkcija deluje tako, da zaznava oglišča na vzorcu šahovnice, za iskanje ostalih vzorcev (npr. krog) so namenjene druge funkcije, ki se prav tako nahajajo v OpenCV knjižnici (npr. *cvFindCirclesGrid*). Za delovanje potrebuje funkcija naslednja vhodna podatka [9]:

- 8-bitno sliko (lahko je tako barvna kot v črno-beli kombinaciji), kjer se nahaja vzorec šahovnice,
- število oglišč, ki se nahajajo na vzorcu.

Na podlagi vhodnih podatkov funkcija najprej preveri, ali se kjerkoli v vhodni sliki nahaja vzorec šahovnice. V primeru, da prepozna tak vzorec, prične z iskanjem oglišč tako, da išče stike robov med črnim in belim poljem. Če se število najdenih robov ujema z vhodnim podatkom, pomeni, da je funkcija uspešno zaznala ter locirala vsa oglišča. Nato kot izhodno vrednost vrne vektor, ki vsebuje koordinate vseh oglišč. Oglišča so v vektorju zapisana v takem vrstnem redu, da so v prvi komponenti shranjene vrednosti koordinat najvišjega levega oglišča. Nato zaporedoma shranjuje vrednosti oglišč iz iste vrste proti desni smeri. Po enakem vzorcu se shranjujejo vrednosti tudi v naslednjih vrstah (po vrstah se "sprehaja" od najvišje proti najnižji). Na ta

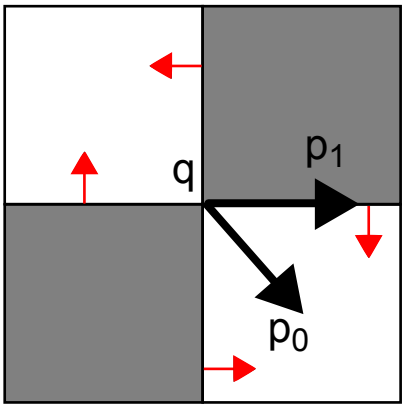
način bo kot zadnja komponenta shranjeno najnižje oglišče na skrajni desni strani.

Dobljene koordinate oglišč so približna ocena, za še natančnejšo določitev koordinat se uporabita dve dodatni funkciji, ki sta prav tako vgrajeni v knjižnici OpenCV. Najprej se izvede funkcija *cvCvtColor*, ki pretvori sliko iz običajnega, ti. RGB formata v črno-bel format. S tem se omogoči uporaba naslednje, pomembnejše funkcije, in sicer *cvFindCornerSubPix*, ki na podlagi že znanih približnih koordinat oglišč določi še natančnejše koordinate. Funkcija za delovanje potrebuje naslednje vhodne podatke [9]:

- vhodno sliko s prisotnimi oglišči (v našem primeru je to slika v črno-belem formatu),
- število prisotnih oglišč,
- urejeni par (x, y), ki definira razdaljo v pikslih, okoli katere naj funkcija išče natančnejše koordinate po naslednji formuli:

  $$(2x + 1) \times (2y + 1).$$

  Na primer, če je vhodni parameter enak $(5, 5)$, bo funkcija iskala natančnejšo vrednost v obsegu 11×11 pikslov,
- opcijski parameter omogoča določitev omejitve števila korakov iteracije pri postopku iskanja natančnejših koordinat.



Slika 2.2: *Primer iskanja natančnosti znotraj piksla. Na sliki so prikazani štirje piksli ter smer gradienta slike (rdeče puščice).*

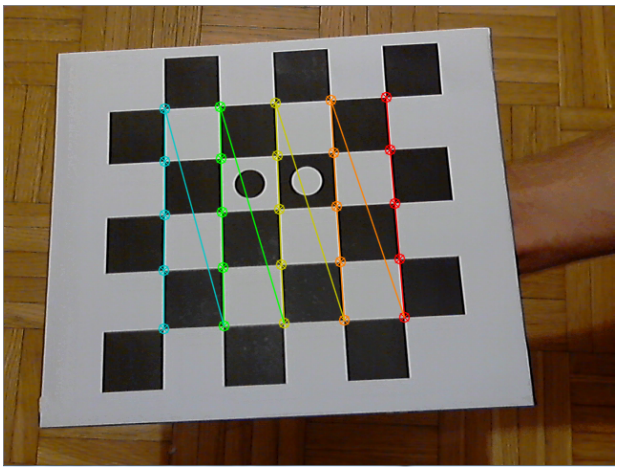
Funkcija poteka na iterativen način tako, da poizkuša poiskati natančno vrednost koordinate znotraj piksla (ang. *sub-pixel accuracy*). Pri iskanju velja

predpostavka, da je vsak vektor od centra q do poljubne točke p pravokoten na gradient slike (Slika 2.2). Poglejmo si naslednji izraz:

$$\epsilon_i = DI_{p_i}^T \cdot (q - p_i),$$

kjer je DI_{p_i} gradient slike na eni izmed točk p_i v okolici q. Okolica je definirana z vhodnim podatkom (x, y). Potrebno je poiskati tako vrednost q, da bo vrednost ϵ_i minimalna. Enačbo se rešuje iterativno, vse dokler se ne doseže minimalne vrednosti oziroma meje števila iteracij v vhodnem parametru. Ko se iteracija ustavi, je vrednost q nova, natančnejša koordinata oglišča.

Med uporabo aplikacije je zaželjeno, da se interakcijska plošča ne razprostira po celotnem vidnem polju kamere, med robom vidnega polja in interakcijsko ploščo naj bo prostora vsaj za dolžino enega kvadratka na plošči (Slika 2.3). S tem omogočimo natančnejšo detekcijo oglišč, saj v primeru slabe vidljivosti funkcija ne more natančno razbrati lokacije, kjer se nahajajo oglišča, v skrajnem primeru lahko celo zazna oglišče na napačni lokaciji.



Slika 2.3: *Največja priporočljiva višina interakcijske plošče. Če jo dvignemo še nekoliko višje, lahko pride do napačnega zaznavanja položaja oglišč.*

Sedaj, ko so položaji vseh oglišč natančno locirani, jih je potrebno še preslikati v koordinatni sistem kamere. Postopek bo opisan v odseku 2.2.1, saj je najprej potrebno definirati določene transformacijske matrike.

2.2 Računanje transformacij

V času delovanja aplikacije zaporedoma izračunavamo transformacijo, ki določa trenutno lego interakcijske plošče glede na izhodišče koordinatnega sistema prostora navidezne $3D$ slike. To transformacijo predstavimo s transformacijsko matriko $\mathbf{T_0}$ na ta način, da najprej postavimo interakcijsko ploščo

v tako lego, da bo določala izhodišče koordinatnega sistema navidezne $3D$ slike, v našem primeru bi to bila prva rezina medicinske slike.

S transformacijsko matriko $\mathbf{T_1}$ prikazujemo trenutni položaj interakcijske plošče. Za razliko od matrike $\mathbf{T_0}$, kjer vrednosti določimo le takrat, ko želimo spremeniti izhodišče koordinatnega sistema navidezne $3D$ slike, se vrednosti transformacijske matrike $\mathbf{T_1}$ spreminjajo sproti ob vsakem premiku interakcijske plošče.

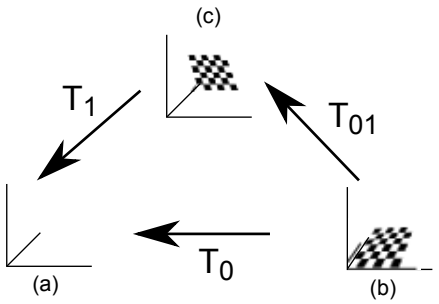
Ko so znane vrednosti obeh matrik $\mathbf{T_0}$ in $\mathbf{T_1}$, je možno izračunati geometrijsko relacijo med koordinatnim sistemom prostora navidezne $3D$ slike in koordinatnim sistemom preseka slike, ki ga prikazujemo na zaslonu. Opisano relacijo označimo s transformacijsko matriko $\mathbf{T_{01}}$ in jo izračunamo na naslednji način:

$$\mathbf{T_0} = \mathbf{T_1} \cdot \mathbf{T_{01}} \tag{2.1}$$

$$\mathbf{T_{01}} = \mathbf{T_1}^{-1} \cdot \mathbf{T_0} \tag{2.2}$$

Omeniti je potrebno, da so transformacije definirane tako, da potekajo od koordinatnega sistema, ki ga določa interakcijska plošča proti koordinatnemu sistemu kamere. Enakost v enačbi (2.1) velja, saj najprej apliciramo transformacijo iz koordinatnega sistema izhodišča $3D$ slike proti koordinatnemu sistemu s trenutnim položajem interakcijske plošče, nato izvedemo še transformacijo proti koordinatnemu sistemu kamere. Tako dobimo transformacijo $\mathbf{T_1} \cdot \mathbf{T_{01}}$, kar je ekvivalentno transformaciji $\mathbf{T_0}$ (Slika 2.4).

Na podoben način lahko izpeljemo transformacijsko matriko $\mathbf{T_{01}}$, ob tem pa moramo imeti znane vrednosti matrik $\mathbf{T_0}$ in $\mathbf{T_1}$. Najprej izvedemo transformacijo $\mathbf{T_0}$, zatem še transformacijo $\mathbf{T_1}$ v inverzno smer. S tem dobimo enakost (2.2), ki določa ravnino prikaza $3D$ slike ob upoštevanju koordinate $z = 0$ zaradi $2D$ načina prikaza slike.



Slika 2.4: *Grafična ponazoritev transformacijskih matrik* $\mathbf{T_0}, \mathbf{T_1}$ *in* $\mathbf{T_{01}}$. *Koordinatni sistem (a) predstavlja koordinatni sistem kamere, (b) predstavlja koordinatni sistem navidezne* $3D$ *slike z izhodiščnim položajem interakcijske plošče, (c) pa predstavlja koordinatni sistem navidezne* $3D$ *slike s trenutnim položajem interakcijske plošče.*

2.2.1 Preslikava oglišč v koordinatni sistem kamere

Sedaj, ko so definirane vrednosti potrebnih transformacijskih matrik, je možno opisati postopek preslikave oglišč v koordinatni sistem kamere.

Aplikacija najprej razbere koordinate oglišč iz koordinatnega sistema kamere. Nato na podlagi zunanjih kalibracijskih parametrov pridobi koordinate v koordinatnem sistemu prostora navidezne $3D$ slike, posledično se izoblikuje transformacijska matrika $\mathbf{T_1}$. Na identičen način se pridobijo vrednosti transformacijske matrike $\mathbf{T_0}$.

2.3 Potek interpolacije

V aplikaciji poteka interpolacija v inverzno smer. To pomeni, da moramo vsaki točki na novi sliki določiti novo približno vrednost glede na originalno sliko.

Za uspešno izvedbo izrisa interpolirane vrednosti posameznega piksla je potrebno poznati vrednosti sosednih pikslov v vseh smereh, le-ti so zapisani v vhodni datoteki. Za posamezen izračun vrednosti piksla se izvede naslednje zaporedje korakov (poimenovanja se navezujejo na sliko 1.8):

1. Dobljeni $3D$ slikovni koordinati poiščemo 8 sosednih točk tako, da vrednosti x, y ter z iz slikovne koordinate ustrezno zaokrožimo navzdol oziroma navzgor. S tem pridobimo točke $P_1, P_2, \ldots, P_8$.

2. Iz vhodne datoteke poiščemo vrednosti posamezne sosedne točke.

3. Izvedemo linearni interpolaciji nad točkami P_1 in P_2 ter P_3 in P_4, s katero pridobimo vrednosti točk I_{12} in I_{34}.

4. Izvedemo linearni interpolaciji nad točkami P_5 in P_6 ter P_7 in P_8, s katero pridobimo vrednosti točk I_{56} in I_{78}.

5. Izvedemo linearno interpolacijo nad točkama I_{12} in I_{34}, s katero pridobimo vrednost točke C_0.

6. Izvedemo linearno interpolacijo nad točkama I_{56} in I_{78}, s katero pridobimo vrednost točke C_1.

7. Izvedemo linearno interpolacijo nad točkama C_0 in C_1, s katero pridobimo iskano interpolirano vrednost (točka P).

Naveden postopek se ponovi za izris posameznega piksla prereza $3D$ medicinske slike.

2.4 Predstavitev aplikacije

Vse omenjene metode iz poglavja 1 so implementirane v aplikaciji, ki je namenjena predvsem zdravnikom in ostalemu osebju iz zdravstvenega sektorja, ki bi želeli analizirati medicinsko sliko na bolj otipljiv način.

Kot je že bilo predstavljeno, aplikacija izrisuje $3D$ slike v $2D$ obliki s takim prerezom, kot uporabnik nastavi interakcijsko ploščo pred kamero. Za pravilno delovanje aplikacije je potrebno imeti kamero poljubne ločljivosti, interakcijsko ploščo v obliki šahovnice, potrebno je tudi zagotoviti zadostno prisotnost svetlobe za natančnejše zaznavanje interakcijske plošče.

2.4.1 Implementacija

Aplikacija je v celoti implementirana v programskem jeziku C++ z uporabo OpenCV knjižnice. To je odprtokodna knjižnica s področja računalniškega vida (ang. *Computer Vision*), izdana pod BSD licenco. Knjižnica ima velik poudarek na aplikacijah, ki se izvajajo v realnem času, saj vsebuje funkcije, ki sproti obdelujejo sliko, npr. sproti zaznava obraze, razbere črke z besedila na sliki itd. Knjižnica deluje tudi v večjedrnem načinu. Trenutno vsebuje več kot 500 funkcij, ki so razporejene po celotnem področju računalniškega vida [10].

Ob zagonu aplikacija najprej prebere vsebino medicinske slike v surovem formatu (ang. *raw image format*) ter jo shrani v medpomnilnik (ang. *buffer*). Nato se aplikacija do preklica izvaja znotraj zanke, v kateri zajame sliko iz kamere, poišče vzorec interakcijske plošče ter poišče vseh 25 oglišč. Na osnovi položaja oglišč na sliki in poznavanja geometrije interakcijske plošče zazna njeno lego. Sliko izrisuje za področje, večje od interakcijske plošče in sicer velikosti 400×400 mm z ločljivostjo 400×400 pikslov velikosti 1×1 mm.

S transformacijsko matriko se določi lega vsakega piksla prikazane slike v prostoru navidezne $3D$ slike. V tem koraku se postopek množenja in interpolacije ponovi kar 160.000-krat (400×400 pikslov) za vsak izris slike, kar pomeni, da je potrebno postopku posvetiti veliko pozornost v smislu optimalnosti oziroma hitrosti delovanja. To je bil tudi eden izmed razlogov za implementacijo v programskem jeziku C++, ki z neposrednim dostopom do pomnilniških lokacij omogoča optimalnejšo realizacijo obdelave podatkov. Tako se pridobljena slika izriše na zaslon.

Celoten postopek se ponavlja v neskončni zanki, s čimer se doseže zadostna hitrost obdelave za uporabo v realnem času.

Rodrigues-ova formula

Aplikacija pridobi rotacijsko matriko s pomočjo funkcije *cvRodrigues2*, ki je vgrajena v knjižnici OpenCV. Kot samo ime pove, funkcija pridobi rotacijsko matriko s pomočjo Rodrigues-ove formule, deluje pa tako, da sprejme dva parametra, in sicer:

- prvi parameter predstavlja rotacijski vektor, ki pretvori točke iz modela koordinatnega sistema v koordinatni sistem kamere,
- drugi parameter predstavlja rotacijsko matriko, v katero shrani vrednosti, pretvorjene iz rotacijskega vektorja.

Formulo bo najbolj razumljivo ponazoriti z naslednjim primerom. Recimo, da želimo zasukati vektor $\mathbf{v}$ okrog vektorja $\mathbf{n}$ za kot θ (Slika 2.5). Vektor $\mathbf{v}$ lahko zapišemo kot vsoto dveh vektorjev, enega kolinearnega, drugega prakovotnega na vektor $\mathbf{n}$:

$$\mathbf{v} = (\mathbf{v} \cdot \mathbf{n})\mathbf{n} + (\mathbf{v} - (\mathbf{v} \cdot \mathbf{n})\mathbf{n}) \tag{2.3}$$

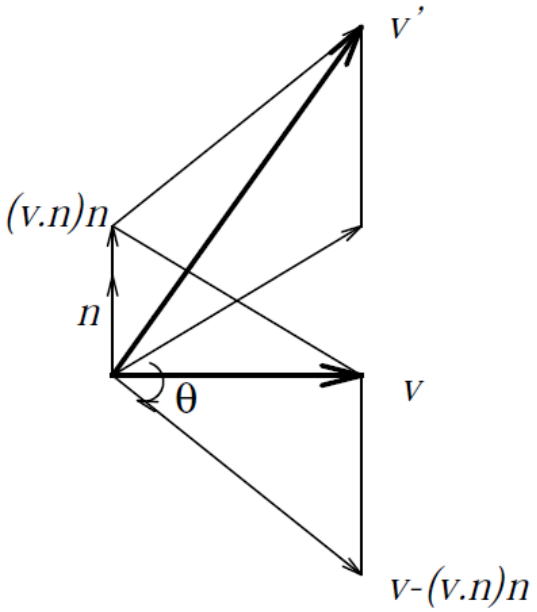
Ko zavrtimo vektor $\mathbf{v}$ za kot θ, se prva komponenta iz enačbe (2.3) ne spremeni, saj le-ta leži na isti premici kot vektor $\mathbf{n}$, medtem ko drugo komponento sedaj zapišemo kot:

$$(\mathbf{v} - (\mathbf{v} \cdot \mathbf{n})\mathbf{n}) \cos\theta + (\mathbf{n} \times (\mathbf{v} - (\mathbf{v} \cdot \mathbf{n})\mathbf{n})) \sin\theta$$

Iz tega sledi:

$$\mathbf{v}' = \mathbf{v} \cdot \cos\theta + (\mathbf{n} \times \mathbf{v}) \sin\theta + \mathbf{n}(\mathbf{v} \cdot \mathbf{n})(1 - \cos\theta) \tag{2.4}$$

Enačbo (2.4) imenujemo Rodriguesova rotacijska formula.



Slika 2.5: *Primer rotacije vektorja* $\mathbf{v}$ *okoli vektorja* $\mathbf{n}$ *za kot* θ.

Če Rodriguesovo formulo zapišemo kot

$$\mathbf{v}' = \mathbf{v} + (\mathbf{n} \times \mathbf{v}) \cdot \sin\theta + \mathbf{n} \times (\mathbf{n} \times \mathbf{v})(1 - \cos\theta),$$

kjer velja $\mathbf{n} \times (\mathbf{n} \times \mathbf{v}) = (\mathbf{v} \cdot \mathbf{n})\mathbf{n} - \mathbf{v}$, lahko sedaj rotacijsko matriko $\mathbf{R}_\theta^n$ zapišemo kot:

$$\mathbf{R}_\theta^n = \mathbf{I} + \mathbf{X}(n)\sin\theta + \mathbf{X}(n)^2(1 - \cos\theta), \tag{2.5}$$

kjer velja:

$$\mathbf{X}(n) = \begin{bmatrix} 0 & -n_z & n_y \\ n_z & 0 & -n_x \\ -n_y & n_x & 0 \end{bmatrix}, \quad \mathbf{n} = [n_x, n_y, n_z].$$

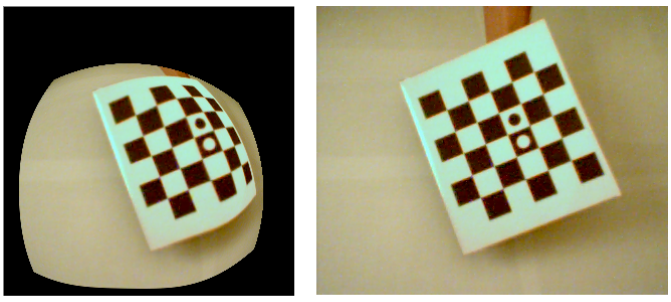
Funkcija *cvRodrigues2* za izračun rotacijske matrike uporablja ravno enačbo (2.5) [9].

2.5 Primer uporabe

Priporočljivi napotki za uporabo aplikacije so, da se kamera postavi pravokotno nad mizo oziroma kakšno drugo večjo trdno podlago pri višini največ enega metra. Prisotnost svetlobe naj bo čim večja, da ne bi prihajalo do napačnega zaznavanja oglišč na interakcijski plošči.

V nadaljevanju bo na primeru prikazan potek izvajanja aplikacije. Uporabljena spletna kamera je *Logitech HD Webcam C270*, nastavljene lastnosti kamere med uporabo aplikacije so naslednje:

- zajem slike standardne ločljivosti 1024×768 pikslov,
- vključena "RightLight" tehnologija za samodejno prilagajanje svetlobe v sliki.



Slika 2.6: *Primer slabo (a) in dobro (b) opravljenega postopka kalibracije. Pri slabo opravljeni kalibraciji se vidijo očitne sledi radialne distorzije (ukrivljenost slike v obliki "soda"), medtem ko pri dobro opravljeni kalibraciji ni videti sledi ukrivljenosti.*

2.5.1 Postopek kalibracije

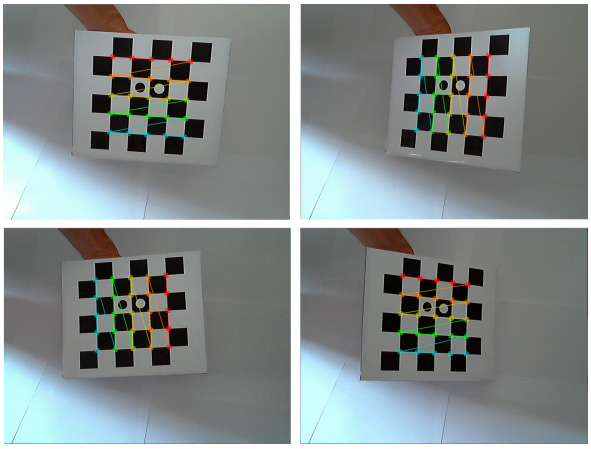
Pred prvo uporabo glavne aplikacije je najprej potrebno opraviti kalibracijo kamere, da program pridobi potrebne notranje parametre, katerih rezultate shrani v matriko za kasnejšo uporabo.

Postopek kalibracije se prične z zajemanjem slike iz kamere. Zaradi izredno hitrega izvajanja postopka je aplikacija sposobna zajeti več kot 20 slik v sekundi, kar ne bi imelo nobenega smisla, saj v tako kratkem času uporabnik ne zmore zadostno premakniti interakcijske plošče. Zato je nastavljena zanka, da aplikacija zajame sliko enkrat do dvakrat v sekundi - odvisno od dejanske hitrosti izvajanja aplikacije, tako uporabniku ni potrebno hiteti s spreminjanjem položaja interakcijske plošče.

Ko aplikacija zajame sliko, najprej poišče približne koordinate oglišč. Nato s pomočjo vgrajene funkcije *cvFindCornerSubPix* poišče še bolj natančne koordinate oglišč. Postopek se ponavlja do natančnega lociranja vseh koordinat oglišč, medtem je potrebno premikati interakcijsko ploščo v vse možne smeri.

Ko aplikacija locira vsa oglišča, se na zaslonu pojavita dve okni - eno je z nepopačeno sliko, drugo pa s popačeno sliko, torej s prisotnostjo distorzije. Če se sliki med seboj ne razlikujeta in ni opaziti učinka distorzije, pomeni da je postopek kalibracije uspešno opravljen in lahko zaključimo z izvajanjem aplikacije. V primeru, da so odstopanja velika (na sliki 2.6 je videti očiten učinek distorzije), je potrebno aplikacijo ponovno zagnati ter opraviti nov postopek kalibracije.

Ob uspešno opravljenem postopku aplikacija določi interne parametre kamere. Med postopkom kalibriranja je potrebno poudariti še, da je priporočljivo interakcijsko ploščo premikati tako, da se čim bolje pokrije celoten vidni prostor (Slika 2.7).



Slika 2.7: *Štiri slike vzorca kalibracije.*

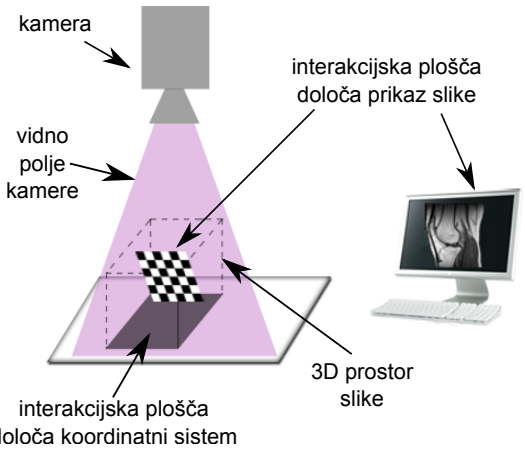
2.5.2 Glavna aplikacija

Po uspešno opravljeni kalibraciji lahko pričnemo z uporabo glavne aplikacije (Slika 2.9). Ob zagonu se prikaže okno, ki sproti izrisuje trenutno sliko prikaza kamere. Pri tej aplikaciji ni nastavljene zanke, ki bi zajela približno eno sliko na sekundo, saj v tej fazi želimo doseči čim bolj gladko izvajanje aplikacije.

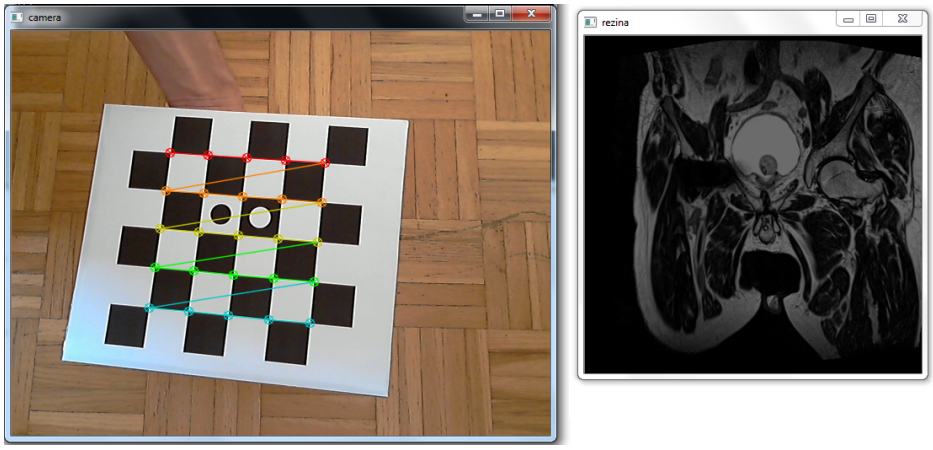
Ob pritisku na tipko "0" se glede na trenutno lego interakcijske plošče določi koordinatno izhodišče $3D$ slike, ki določa prvo (spodnjo) rezino $3D$ slike. Če uporabnik ni zadovoljen s trenutnim položajem koordinatnega izhodišča, ga lahko kadarkoli spremeni tako, da ponovno postavi interakcijsko ploščo v želen položaj in ponovno pritisne tipko "0".

Ko je uporabnik zadovoljen z nastavljenim izhodiščem, lahko prične s premikanjem oziroma nagibanjem plošče v želeno smer, ustrezen prerez pa se bo sproti v realnem času prikazoval na zaslonu računalnika. Med izvajanjem aplikacije lahko uporabnik pritisne tipko "p" v primeru, da želi ustaviti izrisovanje prereza slike, da bi lahko na primer podrobneje analiziral doseženi prerez slike.

Ko uporabnik zaključi s pregledovanjem slike in želi zapustiti program, pritisne tipko "ESC", nakar se bodo vsa alocirana mesta v spominu izbrisala.



Slika 2.8: *Grafična ponazoritev uporabe aplikacije.*

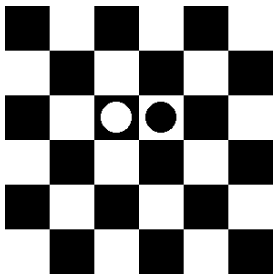


Slika 2.9: *Primer izrisa medicinske slike na podlagi trenutnega položaja interakcijske plošče.*

Zaključek

Trenutna izvedba aplikacije sledi osnovni ideji interaktivnega vpogleda v $3D$ sliko. Gre za prvo različico aplikacije, ki še ima določene pomanjkljivosti, katere bo potrebno izboljšati v nadaljnjem delu.

Prvi problem je zaznavanje rotacije plošče. Glede na trenutni vzorec interakcijske plošče je možno zaznavanje kota le v obsegu 90-ih stopinj, saj ob tolikšnem zasuku kamera zazna vzorec, ki je enak tistemu, ko ni prisotne rotacije. Problem je možno odpraviti z rahlo modifikacijo interakcijske plošče, na primer tako, da si na plošči izberemo dva sosedna polja različnih barv (Slika 2.10). V črno polje vstavimo razpoznaven vzorec bele barve, v belo polje pa enak vzorec črne barve, na primer obarvan krog. S tako obliko plošče lahko zaznamo, kako si vzorca sledita in na podlagi tega razberemo, ali je rotacija presegla $90, 180$ oziroma 270 stopinj in s tem omogočimo rotacijo okoli celotne osi.



Slika 2.10: *Modificirana interakcijska plošča za zaznavanje rotacij nad 90-imi stopinjami.*

Naslednji problem aplikacije se pojavi takrat, ko želimo pogledati $3D$ sliko, kakršno pridobimo iz medicinskih slikovnih naprav, saj jo moramo najprej pretvoriti iz formata DICOM v surov format, kar pa se ne izkaže za preveč praktično rešitev. V prihodnje bo aplikacijo potrebno razširiti tako, da bo neposredno odpirala slike v DICOM formatu.

Možna bi bila tudi izvedba s pomočjo grafičnega procesorja, ki bi aplikacijo občutno pohitrila. Ideja je toliko bolj izvedljiva, saj knjižnica OpenCV vsebuje GPU modul, ki podpira izvajanje operacij na grafičnem procesorju.

Edina pomanjkljivost je, da modul podpira delovanje zgolj na grafičnih procesorjih, proizvedenih s strani Nvidie, saj je modul implementiran na podlagi NVIDIA Cuda Runtime programskega vmesnika.

Opisana aplikacija ima največjo uporabnost pri pregledovanju medicinskih slik, za katere je značilno, da je njihova gostota zelo visoka, torej vsak del ali prerez slike lahko razkrije veliko podrobnosti. Vendar bi se jo z nekaj prilagoditvami lahko uporabilo tudi v arhitekturi pri prikazovanju raznih modelov in načrtov, zapisanih v $3D$ obliki.

Z razvojem prenosnih naprav z vse večjimi procesorskimi zmogljivostmi in zasloni, bomo lahko v prihodnosti z njimi nadomestili interakcijsko ploščo in prikaz prenesli neposredno v navidezni prostor $3D$ slike. S tem bo dojemanje slike še bolj enostavno in hkrati priročno za uporabo, saj bo ogled $3D$ slike mogoč tudi prek tabličnih računalnikov oziroma ostalih naprednih naprav (npr. mobilni telefon) z vgrajenim operacijskim sistemom. V tem primeru bi namesto interakcijske plošče uporabili fiksni kalibracijski vzorec, na katerem bi s pomočjo vgrajene kamere zaznali spremembo višine oziroma nagiba naprave.

Literatura

[1] DICOM standard, najdeno 1.9.2011 na spletnem naslovu: http://medical.nema.org

[2] M. Shah: Fundamentals of Computer Vision, University of Central Florida, Orlando, USA, 1997, str. 11-18

[3] Z. Zhang: Flexible Camera Calibration By Viewing a Plane From Unknown Orientations, Microsoft Research, Redmond, USA, 1999, str. 1-7

[4] M. B. Karbo: Digital Foto, Denmark, 2004, str. 78-216

[5] J. Perš, S. Kovačič: Nonparametric, Model-Based Radial Lens Distortion Correction Using Tilted Camera Assumption, Proceedings of the Computer Vision Winter Workshop 2002, Bad Aussee, Austria, 2002, str.286-295

[6] R. Szeliski: Computer Vision: Algorithms and Applications, 2010, str. 31-45

[7] Primer $3D$ transformacij, najdeno 1.9.2011 na spletnem naslovu: http://msdn.microsoft.com/en-us/library/ee416440%28v=vs.85%29.aspx

[8] C. Browm, D. Ballard: Computer Vision, University of Rochester, New York, USA, 1982, str. 489-490

[9] OpenCV Reference Manual, v2.2, December 2002, str. 347 - 349, 438, 456

[10] A. Kaehler, G. Bradski: Learning OpenCV, O'Reily Media, USA, 2008, str. 1-8