Virtualna tipkovnica

Proleta, Miro

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

ELEKTROTEHNIČKI FAKULTET

Sveučilišni studij

VIRTUALNA TIPKOVNICA

Diplomski rad

Miro Proleta

Osijek, 2016

Sadržaj

1. UVOD

Napretkom tehnologije računala već neko vrijeme se iz upotrebe izbacuju uređaji povezani kabelom, primjerice miš, tipkovnica, web kamera. Bežično povezivanje takvih uređaja uvelike olakšava i ubrzava rad. Naravno uz velike prednosti bežičnog spajanja postoje i nedostaci, a među najvažnijim je potrošnja baterija, što uzrokuje njihovu čestu izmjenu, te time povećava trošak i isplativost uređaja.

U ovom radu razmatra se stvaranje virtualne tipkovnice koja radi bez sklopovlja. Virtualna tipkovnica isprintana je na papir, a ideja se temelji na prepoznavanju pozicije slova na papiru i vrha prsta, putem web kamere. Prilikom preklapanja pozicije vrha prsta i određenog slova, simulira se rad tipkovnice, odnosno program ispisuje slovo iznad kojeg se nalazi prst.

Kako bi realizirali ideju koriste se operacije digitalne obrade slike kao što su: filtriranje, morfološke operacije, binarizacija slike i ostalo. Digitalna obrada slike izvršava se u programskom okruženju MATLAB, a operacije digitalne obrade izvedene su s alatom *Image aquistion tool*.

Prvo poglavlje rada ukratko opisuje programski jezik MATLAB i daje teorijske osnove za digitalnu obradu slike. U drugom poglavlju detaljno je objašnjen način izrade virtualne tipkovnice, dok treće poglavlje govori o nedostacima ovakvog pristupa izvedbi virtualne tipkovnice pomoću web kamere i mogućim rješenjima za uklanjanje tih nedostataka.

2. SLIKA U MATLABU

Prilikom izrade diplomskog rada korišten je programski jezik MATLAB (engl. *Matrix Laboratory*). MATLAB je programski jezik visoke razine i interaktivna je okolina za numeričko i matrično računanje. Prva verzija MATLAB-a napisana je krajem 1970. godine s ciljem primjene u matričnoj teoriji, linearnoj algebri i numeričkoj analizi. Danas, osim osnovnog sustava razvijeni su brojni programski paketi i alati koji pokrivaju gotovo sva područja inženjerske djelatnosti kao što su: obrada signala i slike, 2D i 3D grafički prikaz, automatsko upravljanje, identifikacija sustava, statistička obrada, analiza u vremenskoj i frekvencijskoj domeni, simbolička matematika i brojne druge. MATLAB korisniku omogućava kreiranje vlastitih alata i biblioteka, te daje mogućnost modificiranja postojećih biblioteka. Razvijeni programski paketi i alati daju prednost u korištenju nad ostalim programskim jezicima. Neki od programskih paketa i alata za digitalnu obradu slike u MATLAB-u biti će prikazani i korišteni u ovom diplomskom radu.

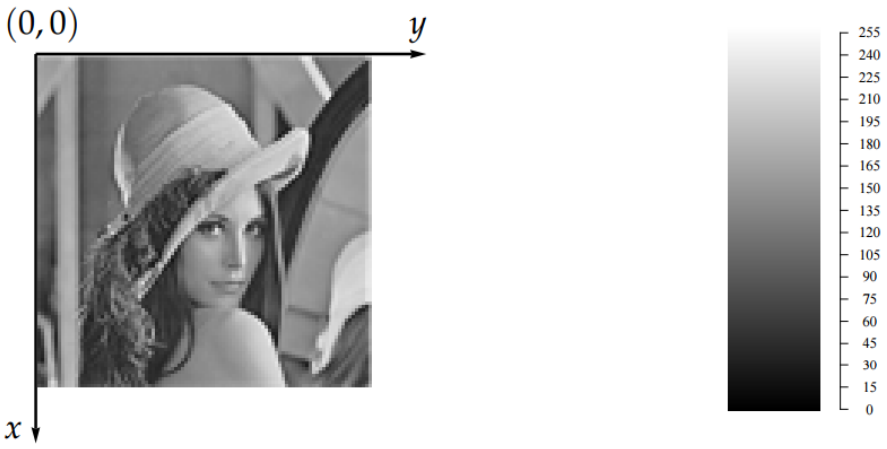
2.1 TIPOVI DIGITALNIH SLIKA

MATLAB ima mogućnost spremanja i prepoznavanja slike kao dvodimenzionalna matrica podataka gdje svaki element matrice predstavlja piksel (engl. *picture element*) prikazane slike. Svaki piksel označava jednu točku na zaslonu računala.

Slika u digitalnom zapisu prezentirana je funkcijom [2-1]

$$f: \Omega \subset Z^2 \rightarrow Z^s, \qquad 2\text{-}1$$

gdje domena Ω funkcije f predstavlja pozicijske indekse piksela, koji se u tom području dodjeljuju načinom prikazanim na slici (Sl. 2.1). Vrijednost $f(i,j) \in Z^s$ predstavlja intenzitet boje piksela prikazane u različitim skalama ([0,255] ili [0,65535]).



Slika 2.1 Pozicijski indeksi piksela i intenzitet slike u sivim tonovima [1]

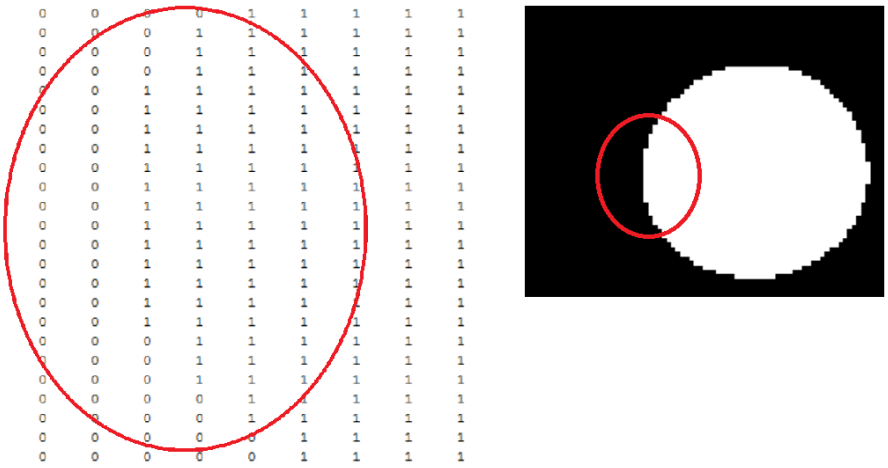
MATLAB ima mogućnost obrađivati podatke matrice na razne načine, što daje veliku mogućnost i mnoštvo ideja za obradu slike te izradu aplikacija temeljenih na obradi slika. Tipovi slika koje su korištene za izradu virtualne tipkovnice su slika u tonovima sive boje (engl. *grayscale*), binarna slika te slika u boji (engl. RGB).

2.1.1 Slika u boji

Slika u boji je slika u kojoj je svaki piksel prikazan s tri vrijednosti, gdje svaka vrijednost predstavlja intenzitet crvene, zelene i plave boje, a kombinacija tih vrijednosti odgovara boji piksela. Slika u boji može zauzimati 24, 48, 96 ali i više bita memorije. Najčešće korištena 24 bitna slika prikazana je sa 8 bita po pikselu, što znači da je intenzitet crvene, zelene i plave boje prikazan skalom od 0-255, dok bi na 48 bitnoj slici svaka boja bila prikazana skalom od 0-65535. Na primjer na 8 bitnoj slici bijela boja nastaje kombinacijom svih boja, a označava se [255,255,255]. Prvi broj predstavlja crvenu, drugi broj zelenu i treći broj plavu boju. 24 bitna slika može prikazati približno 16.7 milijuna boja.

2.1.2 Binarna slika

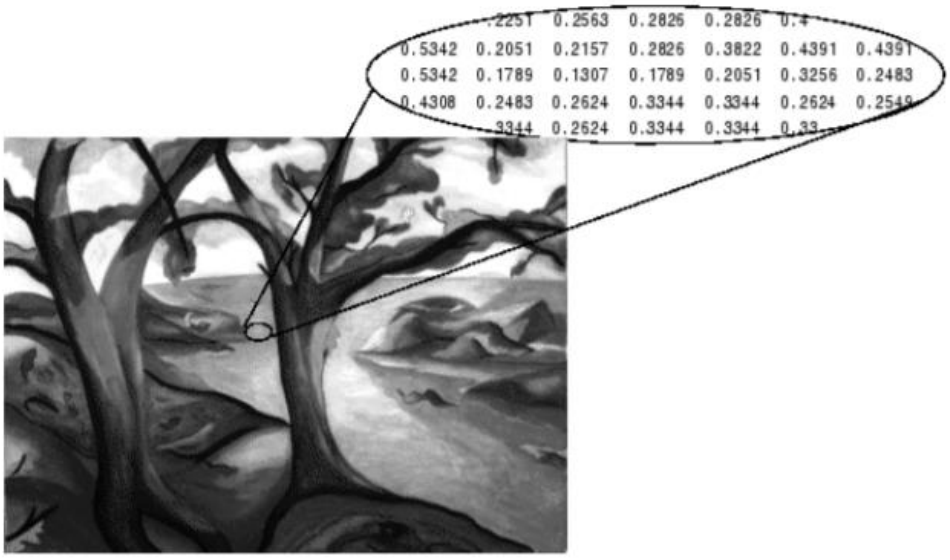
Svaki piksel binarne slike može poprimiti jednu od dvije moguće vrijednosti. Matrica binarne slike u MATLAB-u prikazana je logičkim veličinama “0” i “1” gdje “0” predstavlja crnu boju, odnosno pozadinu slike, a “1” bijelu boju, odnosno objekt koji želimo prikazati, što je vidljivo na primjeru slike novčića (Sl. 2.2) pretvorene u binarni tip slike.



Slika 2.2 Binarna slika

2.1.3 Slika nijansirana sivom bojom (engl. grayscale image)

Slika u sivim tonovima ili *grayscale image* prikazuje sliku nijansiranu sivom bojom. Na slici (Sl. 2.3) prikazan je intenzitet boja između 0 i 1. Nula najtamniji dio, odnosno crna boja dok „1“ predstavlja najsvjetliji dio, odnosno bijelu boju.



Slika 2.3 Slika u sivim tonovima [1]

2.2 DIGITALNA OBRADA SLIKE

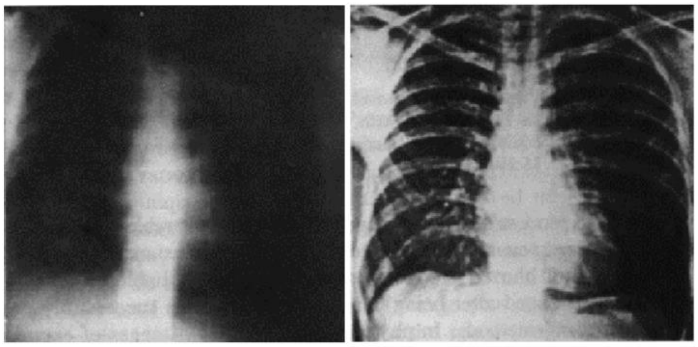
Digitalna obrada slike (engl. *digital image processing*) je metoda kojom se izvode razne operacije na odabranoj slici kako bi dobili poboljšanu sliku ili izvukli željene informacije. Na slici (Sl. 2.4) je prikazan princip digitalne obrade slike.

Slika 2.4 Digitalna obrada slike

Vrijednost svakog piksela na digitalnoj slici je dohvatljiva, što sliku čini upravljivom i pogodnom za daljnju obradu. Digitalnom obradom slike možemo izdvajati promatrane objekte, izdvojiti određenu boju, izoštravati rubove promatranog objekta, ukloniti smetnje i izvoditi mnoge druge operacije korištene u raznim područjima kao što su: medicina, industrijska primjena, astronomija, daljinska snimanja, itd.

U ovom diplomskom radu korištene su operacije izdvajanja određene boje sa slike, operacije filtriranja slike, morfološke operacije, te izdvajanje promatranog objekta što će biti prikazano i opisano u sljedećim poglavljima.

Na slici (Sl. 2.5) je prikazano poboljšanje kontrasta i isticanje vrhova na slici grudnog koša. Digitalna obrada slike primjena u medicini.



Slika 2.5 Digitalna obrada slike, primjena u medicini [2]

2.3 BINARIZACIJA MONOKROMATSKE SLIKE

U sljedećim potpoglavljima opisan je postupak binarizacije slike

2.3.1 Pretvorba slike u boji u sliku s nijansama sive *rgb2gray*

Kako bi slika u boji bila pretvorena u binarnu, potrebno je ukloniti sve krominantne komponente i pretvoriti u sliku s nijansama sive. MATLAB ima mogućnost pretvoriti sliku u boji u sliku s nijansama sive, a pretvorba se izvodi pomoću ugrađene naredbe *rgb2gray*. Naredbom se uklone sve krominantne komponente, pikseli primaju vrijednosti u razmacima između 0-255, gdje je 0 najsvjetliji dio slike, a 255 najtamniji dio slike, odnosno crna boja. Uklanjanje krominantnih komponenata slike prikazano je formulom [2-2]

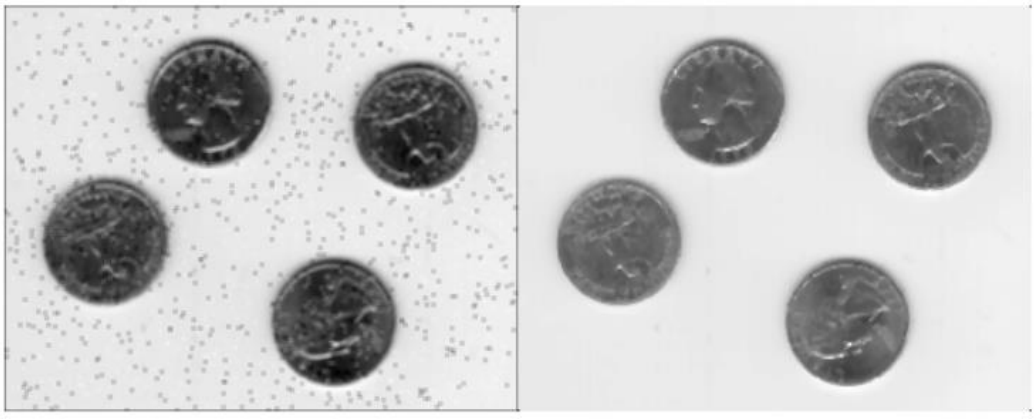
$$0{,}2989 * R + 0{,}5870 * G + 0{,}1140B, \qquad 2\text{-}2$$

a izgled slike u boji nakon pretvorbe na slici (Sl. 2.6).

Slika 2.6 Pretvorba slike u boji u *grayscale image* [2]

2.3.2 Filtriranje slike

Filitriranje se koristi za uklanjanje šuma u digitalnom zapisu slike. Filteri svaki piksel digitalnog zapisa slike procesuiraju strukturnim elementom i na taj način stvaraju rekonstruiranu sliku. Primjer filtriranja prikazan je na slici (Sl. 2.7).

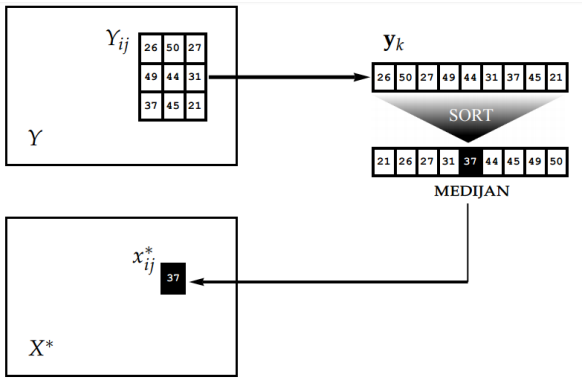


Slika 2.7 Filtrirana slika [1]

Najčešće se koristi 3x3 strukturni element prikazan na slici (Sl. 2.8). U ovome radu za filtriranje se koristi medijan filter. Prema [2, str.63] medijan filter stvara rekonstrukciju X^* degradirane slike Y, na način da se svaki piksel iz X^* dobije kao

$$x^*_{ij=med(y_k)}, \forall(i,j) \in \Omega, k = i * N + j \qquad 2\text{-}3$$

gdje vektor y_k predstavlja strukturni element, prikazan na slici (Sl. 2.8) odgovarajućeg piksela y_{ij} degradirane slike Y.



Slika 2.8 Strukturni element [3]

2.3.3 Rastezanje vrijednosti monokromatske slike *imadjust*

Prije postupka binarizacije koristimo MATLAB funkciju *imadjust* kako bi rastegnuli vrijednost sive boje od 0 – 255. Što znači ako su vrijednosti piksela slike definirani područjem od 100-200 piksela, gdje 100 predstavlja najtamniji dio, a 200 najsvjetliji dio. Pozivanjem funkcije *imadjust* rastežemo područje vrijednosti od 0-255 piksela. rastezanjem vrijednosti osiguravamo sliku od gubitka piksela prilikom određivanja praga radi binarizacije.

2.3.4 Binarizacija slike

Kako bi dovršili postupak binarizacije monokromatske slike, potrebno je odrediti prag na određenoj vrijednosti piksela. Iznad određenog praga sve vrijednosti piksela pretvaraju se u vrijednost crne boje, a ispod vrijednosti praga sve vrijednosti piksela pretvaraju se u vrijednost bijele boje. Najčešće se koristi Otsu metoda određivanja praga. Metoda pretpostavlja da se slika sastoji od dvije kategorije piksela, istaknuti dijelovi i pozadina, a zatim se izračunava prag razdvajanja navedenih kategorija.

Prag t je prikazan maksimiziranjem funkcije σ_B^2 iz prikazane formule [2-4] iz skupova piksela $x_0[0,1,\dots,t-1]$, $x_1[t,t+1,\dots,L-1]$

L - ukupni broj razina

r_q-pojedina razina intenziteta

p_q- vrijednost intenziteta piksela

t – vrijednost praga

gdje su:

$$\sigma_B^2 = \omega_0(\mu_{0-}\mu_T)^2\omega_0(\mu_{1-}\mu_T)^2 \qquad 2\text{-}4$$

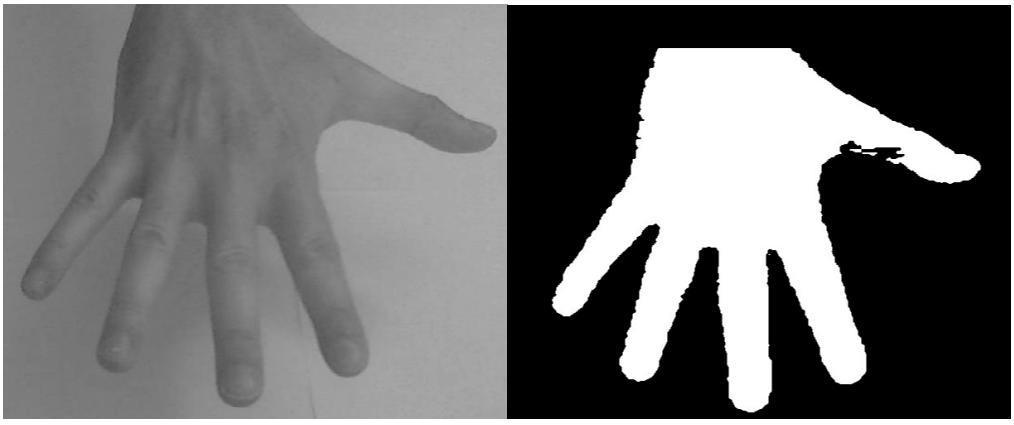
$$\omega_0 = \sum_{q=0}^{k-1} p_q(r_q) \qquad \omega_1 = \sum_{q=k}^{L-1} p_q(r_q)$$

$$\mu_0 = \frac{\sum_{q=0}^{k-1} qp_q(r_q)}{\omega_0} \qquad \mu_1 = \frac{\sum_{q=k}^{L-1} qp_q(r_q)}{\omega_1}$$

$$\mu_T = \sum_{q=0}^{L-1} p_q(r_q)$$

Slika 2.9 Parametri maksimiziranja funkcije σ_B^2 [5]

Primjenom Otsu metode na monokromatsku sliku, dobivena binarna slika odjeljuje bitne i nebitne dijelove slike. Prikaz željenih objekata na binarnoj slici uvelike će ovisiti o osvjetljenju prostora u kojem je slika nastala. Osvjetljenje predstavlja problem, stoga je ponekad potrebno ručno odabrati vrijednost piksela praga. Na slici (Sl. 2.9) prikazana je monokromatska slika s lijeve strane, a s desne slika u binarnom obliku. Prikazani binarizirani oblik nastao je koristeći medijan filter strukturnog elementa [3,3], rastezanjem monokromatske vrijednosti i određivanjem praga vrijednosti piksela za binarizaciju slike korištenjem Otsu metode.



Slika 2.10 Binarizacija slike ruke

2.4 MORFOLOŠKE OPERACIJE

Za poboljšanje odziva i lakšu segmentaciju željenog objekta slike koristimo morfološke operacije. Morfološka obrada slike sastoji se od operacija dilatacije i erozije. Kombiniranjem redoslijeda izvođenja dilatacije i erozije odnosno erozije i dilatacije vrši se tzv. „zatvaranje“ odnosno „otvaranje“. „Zatvaranje“ je postupak povezivanja razdvojenih regija koje pripadaju istom objektu, te uklanjanje nepotrebnih piksela kako bi slika bila pripremljena za detekciju bridova i daljnju obradu. Za pravilan postupak zatvaranja prvo se vrši dilatacija zatim erozija promatrane slike. Najveći nedostatak zatvaranja je zatvaranje bliskih objekata u jednu regiju i ne uklanjanje šuma koji nastaje prilikom pomicanja kamere. „Otvaranje“ je metoda kojom se rješava nastali šum na slici. Kako bi postupak otvaranja bio pravilno izveden, prvo je potrebno izvesti operaciju erozije, a zatim operaciju dilatacije promatrane slike. Erozija će napraviti razmak između slabo povezanih objekata, dok će dilatacija obnoviti rubne piksele. Ukoliko je potrebno detektirati i pratiti veće objekte na promatranoj slici koristi se operacija zatvaranja, zato što nastali šum neće utjecati na zatvaranje regije većeg objekta. U suprotnome, kod manjih objekata koristi se operacija otvaranja zato što šum može utjecati na željeni rezultat. Na slici (Sl. 2.10) je prikazana redukcija šuma uz pomoć operacije otvaranja.

Slika 2.10 Morfološke operacije [1]

2.4.1 Dilatacija

Dilatacija je operacija dodavanja piksela na rubove objekata promatrane slike. Dilatacija se izvodi kada detektirani rubovi nisu dovoljno dobro izraženi za daljnju obradu slike. Dodavanjem piksela vidljiv je rezultat povezivanja regija razmaka manjeg od veličine strukturnog elementa, pojačanja rubova, povećanja objekta i popunjavanja rupa objekata na slici.

Matematička formula glasi:

$$A \oplus B = \{z \in E \vee (B)_z \cap A \neq \emptyset\} \qquad 2\text{-}5$$

A – binarna slika u E

E – Euklidski prostor

B – strukturni element je binarna slika koja prikazuje određeni oblik (krug, pravokutnik,...)

B_z – translatirani strukturni element s ishodištem u z

Prema formuli [2-5], dilatacija binarne slike A strukturnim elementom B je skup svih točaka z, za koje vrijedi da je presjek translatiranog strukturnog elementa B_z i binarne slike A različit od praznog skupa. Naredba korištena u MATLAB-u za dilataciju *imdilate*.

Pseudo kod dilatacija

Dilatacija3x (izvor, dilatirana, maska){ // jednoprolazni algoritam

Ako je maska == 1

okvir ={{255, 255, 255}, //255 bijeli piksel, 0 crni

{255, 255, 255},

{255, 255, 255}};

Inicijaliziraj pointere za piksele izvorne i dilatirane slici

Za svaki redak izvorne slike {

Za svaki stupac izvorne slike{

Odgovarajući piksel (redak,stupac)

Ako je odgovarajući piksel == „pozadinski“ piksel

(vrijednost 0){

Postavi masku preko njega, ako se poklapa vrijednost

piksela maske s vrijednošću piksela izvorne slike:

Onda dati „pozadinski“ postavi u prednji plan (255)

Zapiši odgovarajući piksel u dilatiranu sliku

}

}

2.4.2 Erozija

Erozija je operacija uklanjanja piksela sa rubova objekata promatrane slike. Osnovna ideja slična je kao i kod dilatacije, uklanjanjem piksela povećava se razmak između dva različita odziva i na taj način olakšava njihovu detekciju. Rubovi promatranog objekta bit će precizno određeni, ali se mogu pojaviti rupe odnosno crne točke na objektu. Nastale rupe ispravljaju se dilatacijom.

Matematička formula glasi:

$$A \ominus B = \{z \in E \vee (B)_z \subseteq A\} \qquad 2\text{-}6$$

A – binarna slika u E

E – Euklidski prostor

B – strukturni element je binarna slika koja prikazuje određeni oblik (krug, pravokutnik,...)

B_z – translatirani strukturni element s ishodištem u z

Prema formuli [2-6], erozija binarne slike A strukturnim elementom B je skup svih točaka z, za koje vrijedi da je translatirani strukturni element B_z podskup binarne slike A.

Naredba korištena u MATLAB-u za eroziju *imerode*.

Pseudo kod erozija

Za svaki redak izvorne slike {

Za svaki stupac izvorne slike{

Odgovarajući piksel (redak,stupac)

Ako je odgovarajući piksel == piksel „prednjeg plana“

(vrijednost 255){

Postavi masku preko njega, ako se poklapa vrijednost

piksela maske s vrijednošću piksela izvorne slike:

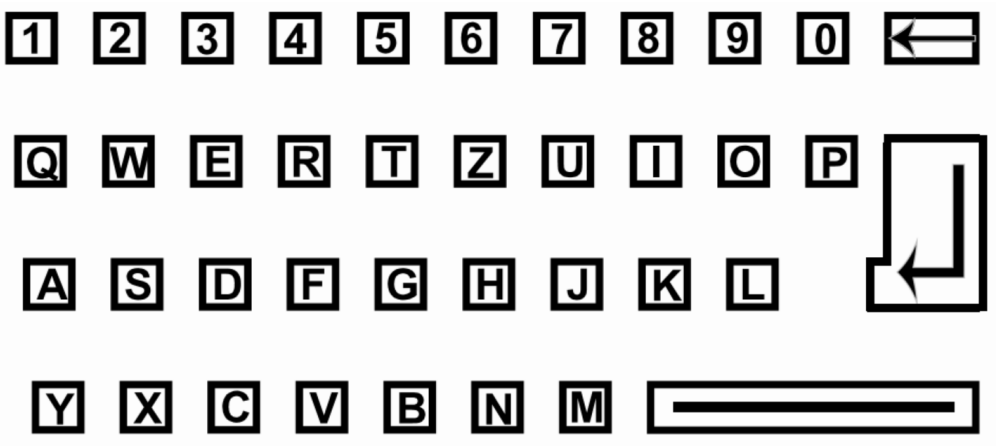
Onda odgovarajući piksel postavi u pozadinu (0)

Zapiši odgovarajući piksel u dilatiranu sliku

}

3. VIRTUALNA TIPKOVNICA TEMELJENA NA WEB KAMERI

Ideja ovog diplomskog rada je izrada virtualne tipkovnice, koja radi na principu prepoznavanja lokacije tipki i vrha prsta prikazanih u području koje snimak web kamere obuhvaća. Web kamera se nalazi zakačena na vrhu monitora. Namještena je tako da snima područje ispod monitora gdje se nalazi tipkovnica koja je prikazana na slici (Sl. 3.1).



Slika 3.1 Tipkovnica

Web kamera je upravljana *image aquisting tool*-om iz MATLAB-a. U MATLAB-u se izvršava kod, koji ima mogućnost izdvojiti vrh prsta, te prepoznati lokaciju svakog slova tipkovnice nacrtane na papiru. Kako bi što točnije izdvojili promatrani objekt tj. prst ili locirali slova nacrtane tipkovnice, prilikom izrade korištene su operacije digitalne obrade slike, opisane u prethodnim poglavljima. Izdvajanje područja interesa (engl. ROI), u ovom slučaju izdvajanje vrha prsta, lociranje i način rada virtualne tipkovnice, prikazan je u nastavku ovog diplomskog rada.

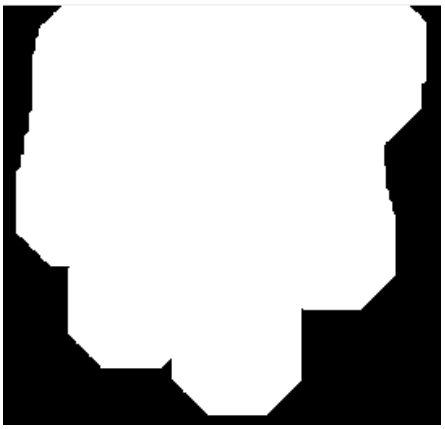
3.1. Izdvajanje vrha prsta

Nakon digitalne obrade i dobivene binarne slike ruke slika (Sl. 3.2), potrebno je izdvojiti područje interesa, kako bi dobili željeni rezultat. U ovom slučaju područje interesa je vrh jednog prsta ruke, pomoću kojega ćemo odabirati tipke sa virtualne tipkovnice. Vrh prsta prikazan je bijelom bojom, dok su ostali objekti na slici pretvoreni u boju pozadine tj. crnu boju. Ideja izdvajanja vrha prsta temeljena je na morfološkim operacijama erozije i dilatacije. Operacijom erozije, te pomoću odabranog strukturnog elementa disk radijusa 20px uništene su sve regije binarne slike koje su manje od korištenog strukturnog elementa. Vrijednost od 20px postavljena je ručno, a predstavlja prosječan radijus vrha prsta, te je temeljena na uzorku vrha prsta 5 različitih osoba, postavljenih na istoj udaljenosti između *web* kamere i tipkovnice. Nakon ove operacije vidljiv je dio šake s izrezanim prstima, prikazano na slici (Sl. 3.2). Udaljenost između *web* kamere i tipkovnice na kojoj je testiran rad je 60cm. Bitno je napomenuti kako je vrijednost strukturnih elemenata potrebno mijenjati ukoliko se kamera postavi približno 10 cm niže ili više od testne *web* kamere.

Slika 3.2 Izdvajanje područja interesa proces erozije

Sljedeći postupak je operacija dodavanja piksela na rubove objekta promatrane slike ili operacija dilatacije. Na objekt dobiven procesom erozije (Sl. 3.2), izvršava se proces dilatacije. Dilatacijom popunjavamo rubove dobivenog objekta strukturnim elementom „disk“ radijusa 40px. Vrijednost

40px dodana je ručno na temelju testova provedenih tijekom kodiranja. Slika (Sl. 3.3) prikazuje objekt dobiven procesom dilatacije.



Slika 3.3 Proces dilatacije

Sljedeći postupak je oduzimanje objekta sa prethodne slike od binarizirane slike cijele ruke, izvršava se naredbom *imsubtract*. Ovaj postupak prikazuje rezultat izdvajanja željenog područja interesa, a to je vrh prsta prikazan na slici (Sl. 3.4).

Slika 3.4 Izdvajanje područja interesa

Naredbom *rgb2gray* ulazna slika u boji „data“ pretvorena je u monokromatski sliku.

Naredbom *medfilt2* provodi se filtracija medijan filterom pomoću filtrirajućeg prozora veličine [3x3].

Naredba *imadjust* služi za rastezanje vrijednosti sive boje monokromatske slike.

Naredba *graytresh* određuje prag za binarizaciju slike putem Otsu metode.

Naredba *im2bw* pretvara dobivenu sliku u binarnu na temelju *graytresh* praga.

Naredba *imfill* popunjava sve rupe na binarnoj slici koje su manje od strukturnog elementa [3x3].

Korišteni kod za binarizaciju slike:

```
diff_im = imsubtract(rgbData(:,:,1), rgb2gray(rgbData));
diff_im = medfilt2(diff_im, [3 3]);
diff_im = imadjust(diff_im);
level = graythresh(diff_im);
bw = im2bw(diff_im,level);
bwfill = medfilt2(imfill(bw,'holes'), [5 5]);
```

Slika 3.5 Korišteni kod za binarizaciju slike

Korišteni kod za izdvajanje vrha prsta:

```
se1 = strel('disk',20);
imErode = imerode(bwfill,se1);
se2 = strel('disk',40);
imDilate = imdilate(imErode,se2);
result = imsubtract(bwfill,imDilate);
se3 = strel('disk',5);
finger = imerode(result,se3);
finger = im2bw(finger);
```

Slika 3.6 Korišteni kod za izdvajanje vrha prsta

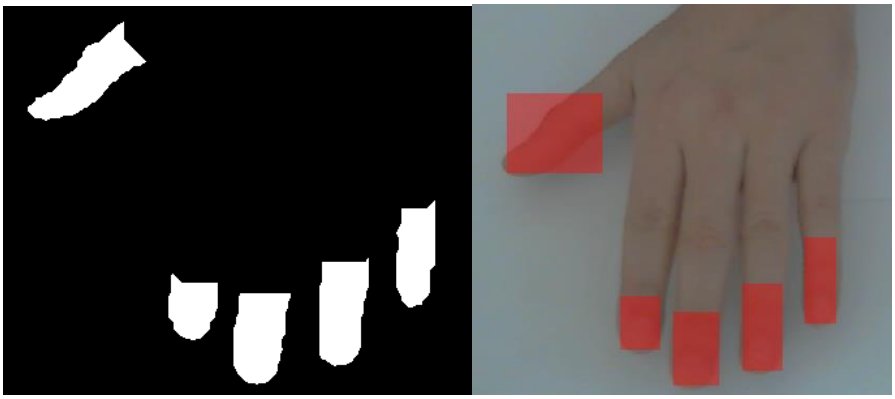
3.2. Metoda analize izdvojenog područja interesa (engl. *Blob analysis*)

Nakon izdvajanja željenog područja interesa i prikaza na slici, potrebno je na neki način uokviriti željeno područje kako bi pristupili potrebnim informacijama za mjerenje veličine, te lociranje vrha

prsta. MATLAB-ov alat *Blob analysis* ima mogućnost izračunati potrebne podatke (širina, visina, centar, radijus…) svih zatvorenih objekata prikazanih na binarnoj slici. Ishodište promatranog objekta postavlja se u gornji lijevi kut, zatim se na temelju postavljenog ishodišta iscrtava granični okvir, koji može biti pravokutnog oblika (engl. *bounding box*), oblika kružnice (engl. *bounding circle*) ili poligon. U ovom radu koristimo granični okvir pravokutnog oblika. Promatrano iz ishodišta graničnog okvira, širina graničnog okvira jednaka je najudaljenijoj koordinati objekta promatranoj po X osi, dok je visina jednaka najudaljenijoj koordinati objekta promatranoj po Y osi prikazano je na slici (Sl. 3.7). Na slici (Sl. 3.8) prikazan je postavljeni granični okvir na svim promatranim objektima binarne slike.

$$\begin{bmatrix} x_1 & y_1 & w_1 & h_1 \\ x_2 & y_2 & w_2 & h_2 \end{bmatrix}$$

Slika 3.7 Granični okvir



Slika 3.8 Granični okvir na izdvojenim vrhovima prstiju

```
hblob1 = vision.BlobAnalysis('AreaOutputPort', false, ...
                             'MajorAxisLengthOutputPort',true,...
                             'MinorAxisLengthOutputPort',true,...
                             'CentroidOutputPort', false, ...
                             'BoundingBoxOutputPort', true', ...
                             'LabelMatrixOutputPort',true,...
                             'MinimumBlobArea', 100, ...
                             'MaximumBlobArea', 1800, ...
                             'MaximumCount', 1);
```

Slika 3.9 Korišteni kod za poziv Blob analysis metode

```
hshapeinsRedBox = vision.ShapeInserter('BorderColor', 'Custom', ...
                                       'CustomBorderColor', [1 0 0], ...
                                       'Fill', true, ...
                                       'FillColor', 'Custom', ...
                                       'CustomFillColor', [1 0 0], ...
                                       'Opacity', 0.4);
```

Slika 3.10 Korišteni kod za uređivanje graničnog okvira

Prethodno je definiran kod za poziv *Blob analysis* metode slika (Sl. 3.9). Odredit će se maksimalna i minimalna površina promatranog objekta, maksimalan broj objekata na binarnoj slici te oblik graničnog okvira.

Površina promatranog objekta je broj koji govori od kolikog broja piksela je sastavljen objekt. Postavljanjem minimalne i maksimalne površine objekta također možemo raditi neku vrstu filtriranja manjih ili većih objekata na binarnoj slici.

Koordinate centroida graničnika pravokutnog oblika računaju se prema formuli [3-1]

$$x_{bb} = \frac{x_{min}+x_{max}}{2} \qquad 3\text{-}1$$

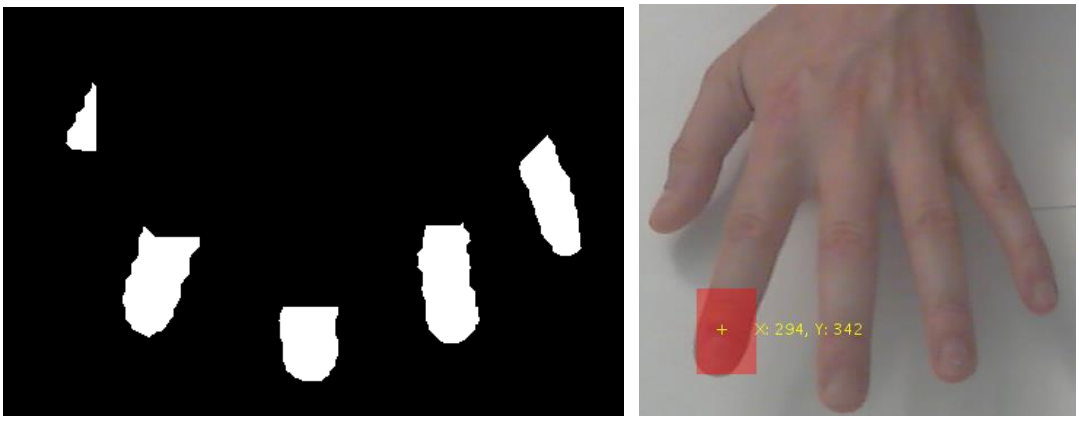
$$y_{bb} = \frac{y_{min}+y_{max}}{2}$$

Gdje x_{min} i x_{max} predstavljaju najbližu i najdalju koordinatu promatranog objekta po x osi, y_{min} i y_{max} predstavljaju najbližu i najdalju koordinatu promatranog objekta po y osi.

Maksimalan broj promatranih objekata postavljen je na 1, zato što se samo jednim prstom odabiru slova sa tipkovnice. Podaci promatranog objekta spremljeni su u tablicu i potrebni su za funkcionalnost tipkovnice. Naredba *step* je implementirana funkcija u MATLAB-u koja poziva video preglednik i ima mogućnosti primiti i prikazati vrijednosti koje su prethodno implementirane. Naredbe *hshapeinsRedBox, bbox* definiraju pravokutni granični okvir oko vrha prsta u crvenoj boji i centar vrha prsta kako je vidljivo na slici (Sl. 3.12).

```
vidIn = step(hshapeinsRedBox, rgbData, bbox1);
step(hVideoOut, vidIn);
vidIn = step(htextinsCent, vidIn, [centX centY], [centX-6 centY-9]);
```

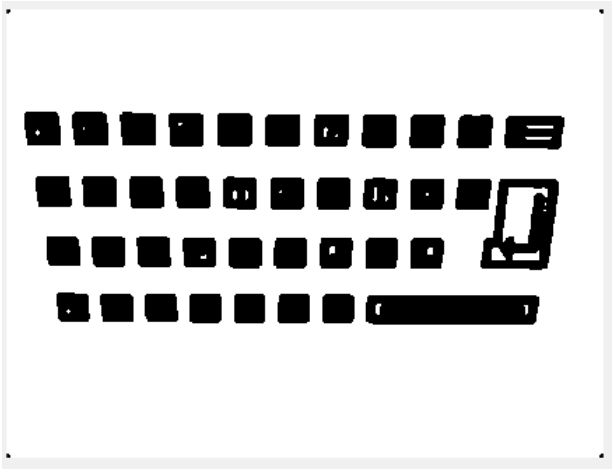
Slika 3.11 Kod za definiranje graničnog okvira i centa graničnog okvira



Slika 3.12 Prikaz centra vrha prsta

3.3. Prepoznavanje tipkovnice

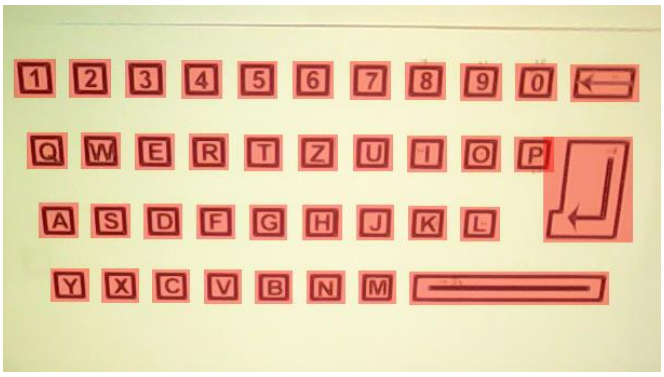
U prethodnom poglavlju opisan je način izdvajanja vrha prsta. Sljedeći korak pri stvaranju funkcionalne virtualne tipkovnice je prepoznavanje pojedinog znaka nacrtane tipkovnice kao objekt. Na slici (Sl. 3.1.) prikazana je korištena tipkovnica. Morfološkom operacijom otvaranja nad slikom tipkovnice svaki znak pretvoren je u poseban objekt (Sl. 3.13.).



Slika 3.13 Morfološka operacija otvaranja nad slikom tipkovnice

Na slici (Sl. 3.13) se vidi kako su svi znakovi, odnosno svi dobiveni objekti odvojeni. *Blob analysis* alatom postavljeni su pravokutni granični okviri (engl. *bounding box*) na svaki odvojeni objekt (Sl.

3.14). Postavljanjem pravokutnih graničnih okvira dobiveni su podaci o visini, širini te lokaciji svakog znaka tipkovnice. Spremljeni podaci prikazani su u tablici (Sl. 3.15), a potrebni su za daljnju obradu i funkcionalnost tipkovnice.



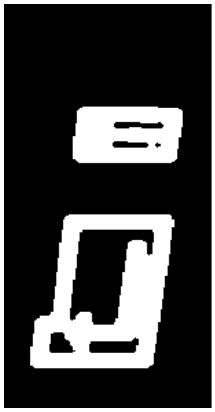
Slika 3.14 Postavljanje pravokutnog graničnog okvira

	1	2	3	4
1	25	114	39	36
2	37	182	38	34
3	48	246	37	33
4	59	307	36	31
5	76	114	39	36
6	87	182	37	34
7	96	247	37	32
8	106	307	35	32
9	128	115	38	35
10	137	183	37	34

Slika 3.15 Zapis pravokutnog graničnog okvira u tablicu

Na slici (Sl. 3.15) kolumne 1 i 2 predstavljaju koordinate ishodišta x, y lijevog gornjeg kuta pojedinog objekta, dok kolumne 3 i 4 predstavljaju širinu i dužinu istog objekta. Vidljivo je da koordinate prvog reda tablice x, y (25,114) u usporedbi sa slikom (Sl. 3.14) odgovaraju pravokutnom graničniku iznad broja 1. Koordinate x, y (37,182), odnosno drugi red tablice odgovara pravokutnom graničniku iznad slova Q, dok koordinate x, y, (48, 246), odnosno treći red tablice odgovara pravokutnom graničniku iznad slova A. Istim redoslijedom, od gore prema dolje, s lijeva na desno, zapisani su svi ostali pravokutni graničnici. Ukoliko postoji više objekata na binarnoj slici koje je potrebno obilježiti pravokutnim graničnikom, prvi objekt koji će biti obilježen je objekt najbliži Y osi. Takav način

pretraživanja objekata nazvan je *Column-wise scan.* Na slici (Sl. 3.16) prvi znak koji će biti označen je znak Enter.

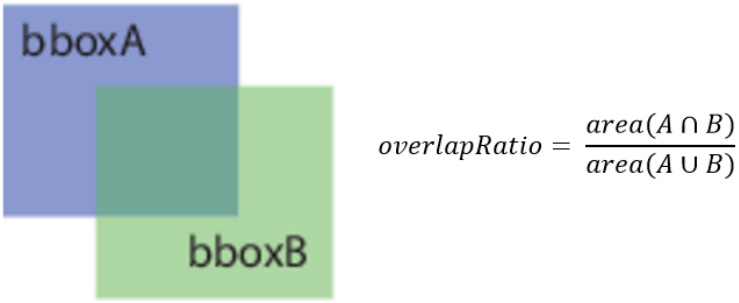


Slika 3.16 Pojašnjenje column-wise scan

3.4. Vrijednost preklapanja pravokutnih graničnika (engl. *Overlap ratio*)

Izvršavanjem koda za prepoznavanje, tipkovnica je jednom slikana, a podaci o lokaciji, visini i širini svake tipke spremljeni su u tablicu i ostaju fiksni. U sljedećem koraku poziva se *while* petlja unutar koje se izvršava kod koji slika, izdvaja vrh prsta, ukoliko je ruka u području *web* kamere i sprema podatke o lokaciji, visini i širini vrha prsta. Petlja se ponavlja sve dok se ne pozove funkcija za prestanak rada tipkovnice, a pomicanjem prsta spremaju se i novi podaci.

Dobiveni podaci vrha prsta te podaci prepoznate tipkovnice korišteni su za funkciju omjera preklapanja (engl. *Overlap ratio*) (Sl. 3.17).



Slika 3.17 Omjer preklapanja

Pozvana funkcija provjerava postoji li preklapanje između pravokutnog graničnika vrha prsta, te pravokutnog graničnika jedne od označenih tipki. Ukoliko postoji preklapanje, računa se *overlapRatio,* vrijednost koja je jednaka omjeru presjeka i unije dvaju pravokutnih graničnika. Rezultat preklapanja je vrijednost između 0 i 1. Ukoliko nema preklapanja vrijednost je 0, a ukoliko vrh prsta cijelom površinom prekrije površinu tipke vrijednost je 1.

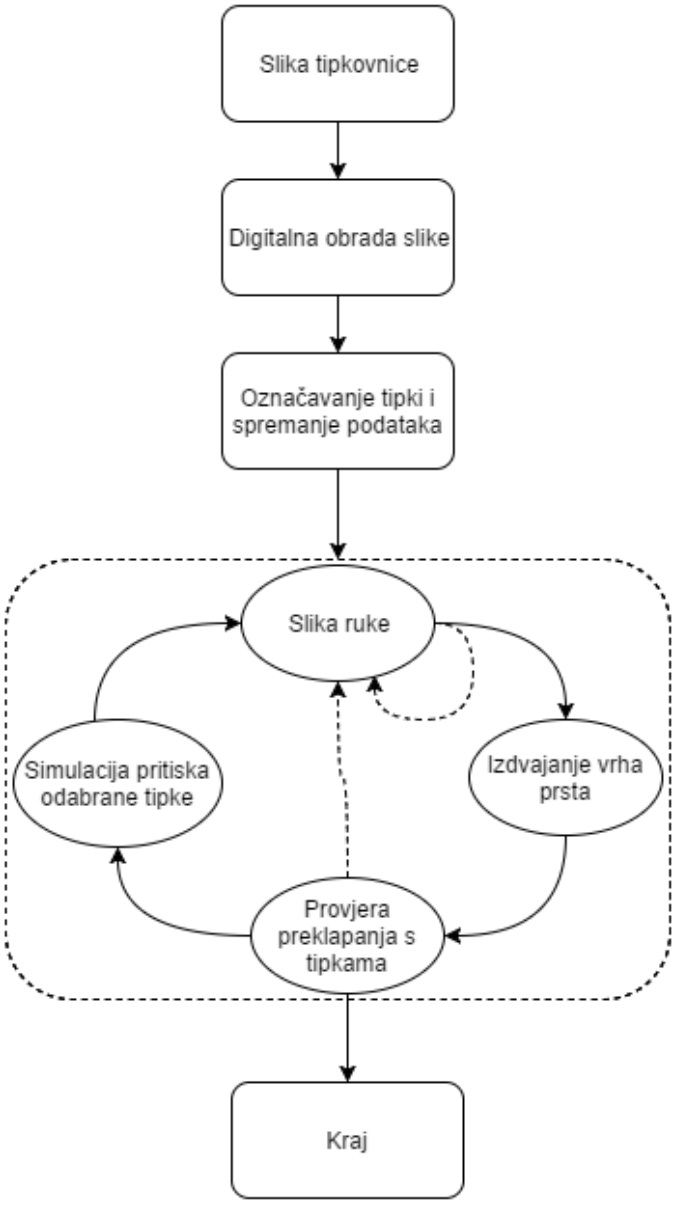
3.5. Funkcionalnost tipkovnice

Za potpunu funkcionalnost tipkovnice postavljeni su uvjeti koji ispituju postoji li preklapajući omjer između bilo koje tipke i vrha prsta. Ukoliko je preklapajući omjer veći od zadane vrijednosti, izvršava se kod unutar zadane *if* petlje. Zadana vrijednost preklapajućeg omjera postavljena je na 0.1, zbog mogućnosti preklapanja vrha prsta sa dvije tipke.

Nakon što je zadovoljen uvjet preklapajućeg omjera, prije simulacije pritiska tipke, poziva se brojač (engl. *timer*). Brojač je korišten iz razloga što je teško prepoznati okidanje, odnosno pritisak određene tipke putem web kamere. Ukoliko brojač ne bi bio korišten, prolazak prsta iznad svake tipke simulirao bi pritisak i ispisivala bi se ne željena slova. Brojač odbrojava 1 sekundu i poziva funkciju koja će još jednom provjeriti postoji li preklapajući omjer između iste tipke i vrha prsta. Ukoliko je preklapajući omjer i dalje prisutan, poziva se *java.awt.robot* funkcija koja ima mogućnost ispisati znak sa odabrane tipke u bilo koji program Windows sučelja. Nakon ispisa određenog znaka postavljena je zastavica *status* za svaku tipku posebno kako ne bi došlo do ponavljanja simulacije pritiska.

4. REZULTATI

Predstavljena je virtualna tipkovnica koja radi putem web kamere. Web kamera pozicionirana je na monitoru te snima područje ispod monitora, gdje se nalazi tipkovnica isprintana na papir. Pozicioniranjem prsta iznad pojedinog slova isprintane tipkovnice, ispisuje se slovo u odabranom programu na računalu. Princip rada odnosno izdvajanje vrha prsta, lociranje te ispis slova prikazano je na sljedeći način (Sl. 4.1).



Slika 4.1 Dijagram rada virtualne tipkovnice

Traženi rezultat je ispis odabranog slova u bilo kojem programu Windows sustava. Pozicioniranjem prsta iznad odabranog slova prikazuje se traženi rezultat. Prikazana ideja i princip rada daje zadovoljavajuće rezultate uz određene nedostatke.

Najveći nedostatak prikazane metode izrade virtualne tipkovnice je osjetljivost na osvjetljenje prostorije. Brzina rada tipkovnice nije zadovoljavajuća zbog korištenih brojača za odgodu pozivanja funkcije ispisa slova. Ideja brojača je sprječavanje odabira neželjenog slova. Brojač odgađa pozivanje funkcije za ispis slova za jednu sekundu, što znači da će se 10 slova ispisivati najmanje 10 sekundi.

Kao bolje rješenje potrebno je koristiti *stereo* kameru. *Stereo* kamera ima mogućnost mjeriti udaljenost željenog objekta na slici od pozicije kamere. U slučaju virtualne tipkovnice, potrebno je postaviti udaljenost od *stereo* kamere do isprintane tipkovnice. Zatim postaviti uvjet gdje se isključuju svi objekti koji se nalaze približno 1 cm iznad isprintane tipkovnice. Na taj način nije potrebno raditi segmentaciju vrha prsta. Kamera bi prepoznala spuštanje prsta iznad određenog slova, zatim bi istim načinom kao u prikazanoj virtualnoj tipkovnici ispisalo određeno slovo.

5. ZAKLJUČAK

Rad prikazuje metode digitalne obrade slike, pomoću kojih je uspješno realizirana ideja virtualne tipkovnice na principu web kamere. Metodom pretvaranja slike u boji, u sliku nijansiranu sivom bojom uklonjene su sve krominantne komponente. Minimalna vrijednost piksela dobivene slike predstavlja bijelu, a maksimalna vrijednost crnu boju. Metoda filtriranja procesuira sliku određenim strukturnim elementom i uklanja šum. Rastezanjem vrijednosti monokromatske slike smanjuje se vjerojatnost gubitka piksela prilikom binarizacije Otsu metodom. Primjenom Otsu metode određivanja praga prikazani su željeni dijelovi slike i upotpunjen je postupak binarizacije. Postupkom binarizacije, slika je pripremljena za daljnju obradu. Morfološkim operacijama primijenjenim na binariziranu sliku ruke ili sliku tipkovnice uspješno je izdvojen vrh prsta odnosno tipke.

Realizirani algoritam pronalazi tipkovnicu, obilježava tipke te izdvaja vrh prsta. Kada je vrh prsta pozicioniran iznad odabrane tipke, znak sa tipke se ispisuje u bilo kojem Windows programu.

Prilikom izrade uočeni su određeni nedostaci za koje je moguće pronaći bolja rješenja te dobiti puno bržu reakciju ispisa odabranog slova. Virtualna tipkovnica koja radi na principu web kamere može se primjenjivati u svakodnevnici, ali gledajući prikazane nedostatke u radu, te uzimajući u obzir nedostatke poboljšanih verzija u kojima se koristi *stereo* kamera, teško je pronaći prednost korištenja u odnosu na tipkovnice sa sklopovljem.

6. LITERATURA

[1] The MathWorks, Image Proccessing Toolbox, User's Guide (1993–2016)

[2] The MathWorks, Image Proccessing Toolbox, Reference (1993–2016)

[3] V. Novoselac, S. Rimac-Drlje, Svojstva i primjena medijana i aritmetičke sredine, Osječki matematički list 14, 51-67, 2014

[4] V. Crnojević, Digitalna obrada slike, 2005.

[5] OTSU, N. A threshold selection method from gray-level histograms, IEEE Trans.Sys., Man., Cyber. 9, 1, (1979), 62-66

[6] The MathWorks, Image Acquisition Toolbox, User's Guide (1993–2016)

[7] International Journal of Advanced Research in Computer and Communication Engineering, AN INNOVATIVE APPROACH TOWARDS VIRTUAL KEYBOARD, ožujak 2015

[8] O. Marques, Practical Image and Video Processing Using MATLAB, John Wiley and Sons, 2011.

[9] V. Novoselac, B. Zovko-Cihlar, Image Noise Removal by Vector Median Filter, Proceedings ELMAR-2012, Croatian Society Electronics in MarineELMAR, 2012, 57–62

[10] Java SE Documentation, Copyright 1993, 2016, Oracle and/or its affiliates

7. SAŽETAK

U ovom radu prikazana je ideja izrade virtualne tipkovnice na principu web kamere. Kako bi došli do zadanog cilja, korištene su metode digitalne obrade slike, poput binarizacije, morfoloških operacija i filtriranja. Postavljen je algoritam koji omogućava ispis slova u bilo kojem programu na Windows operacijskom sustavu. Web kamera je pozicionirana na monitoru i snima područje ispod monitora gdje se nalazi tipkovnica isprintana na papir. Ukoliko prstom pokažemo na određeno slovo tipkovnice, postavljeni algoritam locira vrh prsta i odabrano slovo prikazuje u željenom programu. Prilikom izrade korišten je *Image aquistion tool* u programskom okruženju MATLAB. Najveći nedostatak prikazane metode izrade virtualne tipkovnice je brzina rada i osjetljivost na osvjetljenje prostorije.

8. ABSTRACT

This paper presents the idea of making the virtual keyboard on the principle of web camera. In order to reach the default target, methods are used that include digital image processing like binarization, morphological operations and filtration. Written algorithm enables printing letters in any Windows operating system program. Web camera is positioned on the monitor and it scans the area under the monitor, where paper printed keyboard is located. If the certain letter on the keyboard is pointed by finger, the algorithm locates the top of the finger and shows the selected letter in desired program. In making of this paper, Image aqusition tool in the programming environment of MATLAB was used. The biggest drawback of the virtual keyboard production method shown in this paper is the operating speed and its sensitivity in illuminated spaces.

9. ŽIVOTOPIS

Miro Proleta rođen je 07. ožujka 1989. godine u Novom Sad-u. Živi u Osijeku. Prva četiri razreda osnovne škole pohađao je u Briješću, a druga četiri u Osnovnoj školi Vladimira Nazora u Čepinu. Nakon završetka osnovne škole upisuje Elektrotehničku i prometnu školu u Osijeku, smjer mehatroničar. Po završetku srednjoškolskog obrazovanja, 2008. godine upisuje Elektrotehnički fakultet u Osijeku. 2014. godine završava tečaj za programera mobilnih aplikacija, te „Span“ akademiju u trajanju od 3 mjeseca.

PRILOG

```
import java.awt.Robot;
import java.awt.event.*;
robot = Robot;

%webcam aquistion
vidDevice = imaq.VideoDevice('winvideo', 2, 'I420_640x480', ...
'ROI', [1 1 640 480], ...
'ReturnedColorSpace', 'rgb');

%keyboard bounding boxes
hVideoIn = vision.VideoPlayer;
hVideoIn.Name = 'Keyboard bounding boxes';
hVideoIn.Position = [30 100 640 480];

%finger tracking
hVideoOut = vision.VideoPlayer;
hVideoOut.Name = 'Fingers Tracking Video';
hVideoOut.Position = [700 100 640 480];

%određivanje veličine blob-a
hblob = vision.BlobAnalysis('AreaOutputPort', false, ...
                            'MajorAxisLengthOutputPort',true,...
                            'MinorAxisLengthOutputPort',true,...
                            'CentroidOutputPort', false, ...
                            'BoundingBoxOutputPort', true', ...
                            'LabelMatrixOutputPort',true,...
                            'MinimumBlobArea', 450, ...
                            'MaximumBlobArea', 6000, ...
                            'MaximumCount', 40);

 hblob1 = vision.BlobAnalysis('AreaOutputPort', false, ...
                            'MajorAxisLengthOutputPort',true,...
                            'MinorAxisLengthOutputPort',true,...
                            'CentroidOutputPort', false, ...
                            'BoundingBoxOutputPort', true', ...
                            'LabelMatrixOutputPort',true,...
                            'MinimumBlobArea', 100, ...
                            'MaximumBlobArea', 1800, ...
                            'MaximumCount', 1);

hshapeinsRedBox = vision.ShapeInserter('BorderColor', 'Custom', ...
                                       'CustomBorderColor', [1 0 0], ...
                                       'Fill', true, ...
                                       'FillColor', 'Custom', ...
                                       'CustomFillColor', [1 0 0], ...
                                       'Opacity', 0.4);
%%%%%%%%%%%%%%%%%%%%%%%%% Program %%%%%%%%%%%%%%%%%%%%%%%%%
while(1)
    rgbData = step(vidDevice);
    rgbData = imrotate( rgbData , 180 );
    im = rgb2gray(rgbData);
    diff_im = medfilt2(im, [3 3]);
```

```
    diff_im = imadjust(diff_im);
    level = graythresh(diff_im);
    BW = im2bw(diff_im,level);
    se1 = strel('disk',5);
    imErode = imerode(BW,se1);
    subplot(2,2,1);imshow(imErode);
    se2 = strel('disk',1 );
    imDilate = imdilate(imErode,se2);
    subplot(2,2,2);imshow(imDilate);
    bbox = step(hblob, ~imDilate);
    rgbData(50:60,1:640,:)=1;
    keyboard = step(hshapeinsRedBox, rgbData, bbox);
    step(hVideoIn, keyboard);

    statQ = true;
    statU = true;
    statF = true;
    statX = true;
    statW = true;
    statI = true;
    statG = true;
    statC = true;
    statE = true;
    statO = true;
    statH = true;
    statV = true;
    statR = true;
    statP = true;
    statJ = true;
    statB = true;
    statT = true;
    statA = true;
    statK = true;
    statN = true;
    statZ = true;
    statS = true;
    statL = true;
    statM = true;
    statD = true;
    stat0 = true;
    stat1 = true;
    stat2 = true;
    stat3 = true;
    stat4 = true;
    stat5 = true;
    stat6 = true;
    stat7 = true;
    stat8 = true;
    stat9 = true;
    statEnter = true;
    statBack = true;
    statSpace = true;
    defaultOverlapRatio = 0.1;
    t=0.8;
end
while(true)
    rgbData = step(vidDevice);
```

```
    rgbData = imrotate( rgbData , 180 );
    diff_im = imsubtract(rgbData(:,:,1), rgb2gray(rgbData));
    diff_im = medfilt2(diff_im, [3 3]);
    diff_im = imadjust(diff_im);
    level = graythresh(diff_im);
    bw = im2bw(diff_im,level);
    bwfill = medfilt2(imfill(bw,'holes'), [5 5]);

    se1 = strel('disk',20);
    imErode = imerode(bwfill,se1);
    subplot(2,2,1);imshow(imErode);
    se2 = strel('disk',40);
    imDilate = imdilate(imErode,se2);
    subplot(2,2,2);imshow(imDilate)
    result = imsubtract(bwfill,imDilate);
    subplot(2,2,3);imshow(result)
    se3 = strel('disk',5);
    finger = imerode(result,se3);
    finger = im2bw(finger);
    subplot(2,2,4);imshow(finger)
    bbox1 = step(hblob1, finger);
    vidIn = step(hshapeinsRedBox, rgbData, bbox1);
    step(hVideoOut, vidIn);
    VK_CheckOverlapStartTimer;
end

release(hVideoOut);
release(hVideoIn);
release(vidDevice);
```

```
VK_CheckOverlapStartTimer {
  overlapRatio1 = bboxOverlapRatio(bbox(1:1,:),bbox1);
  overlapRatioQ = bboxOverlapRatio(bbox(2:2,:),bbox1);
  overlapRatioA = bboxOverlapRatio(bbox(3:3,:),bbox1);
  overlapRatioY = bboxOverlapRatio(bbox(4:4,:),bbox1);
  overlapRatio2 = bboxOverlapRatio(bbox(5:5,:),bbox1);
  overlapRatioW = bboxOverlapRatio(bbox(6:6,:),bbox1);
  overlapRatioS = bboxOverlapRatio(bbox(7:7,:),bbox1);
  overlapRatioX = bboxOverlapRatio(bbox(8:8,:),bbox1);
  overlapRatio3 = bboxOverlapRatio(bbox(9:9,:),bbox1);
  overlapRatioE = bboxOverlapRatio(bbox(10:10,:),bbox1);
  overlapRatioD = bboxOverlapRatio(bbox(11:11,:),bbox1);
  overlapRatioC = bboxOverlapRatio(bbox(12:12,:),bbox1);
  overlapRatio4 = bboxOverlapRatio(bbox(13:13,:),bbox1);
  overlapRatioR = bboxOverlapRatio(bbox(14:14,:),bbox1);
  overlapRatioF = bboxOverlapRatio(bbox(15:15,:),bbox1);
  overlapRatioV = bboxOverlapRatio(bbox(16:16,:),bbox1);
  overlapRatio5 = bboxOverlapRatio(bbox(17:17,:),bbox1);
  overlapRatioT = bboxOverlapRatio(bbox(18:18,:),bbox1);
  overlapRatioG = bboxOverlapRatio(bbox(19:19,:),bbox1);
  overlapRatioB = bboxOverlapRatio(bbox(20:20,:),bbox1);
  overlapRatio6 = bboxOverlapRatio(bbox(21:21,:),bbox1);
  overlapRatioZ = bboxOverlapRatio(bbox(22:22,:),bbox1);
  overlapRatioH = bboxOverlapRatio(bbox(23:23,:),bbox1);
  overlapRatioN = bboxOverlapRatio(bbox(24:24,:),bbox1);
  overlapRatio7 = bboxOverlapRatio(bbox(25:25,:),bbox1);
  overlapRatioU = bboxOverlapRatio(bbox(26:26,:),bbox1);
  overlapRatioJ = bboxOverlapRatio(bbox(27:27,:),bbox1);
  overlapRatioM = bboxOverlapRatio(bbox(28:28,:),bbox1);
  overlapRatio8 = bboxOverlapRatio(bbox(29:29,:),bbox1);
  overlapRatioI = bboxOverlapRatio(bbox(30:30,:),bbox1);
  overlapRatioK = bboxOverlapRatio(bbox(31:31,:),bbox1);
  overlapRatioSpace = bboxOverlapRatio(bbox(32:32,:),bbox1);
  overlapRatio9 = bboxOverlapRatio(bbox(33:33,:),bbox1);
  overlapRatioO = bboxOverlapRatio(bbox(34:34,:),bbox1);
  overlapRatioL = bboxOverlapRatio(bbox(35:35,:),bbox1);
  overlapRatio0 = bboxOverlapRatio(bbox(37:37,:),bbox1);
  overlapRatioP = bboxOverlapRatio(bbox(36:36,:),bbox1);
  overlapRatioBack = bboxOverlapRatio(bbox(39:39,:),bbox1);
  overlapRatioEnter = bboxOverlapRatio(bbox(38:38,:),bbox1);

  if overlapRatioQ == 0
     statQ = true;
   end;
  if overlapRatioU == 0
     statU = true;
  end;
  if overlapRatioF == 0
     statF = true;
  end;
  if overlapRatioX == 0
     statX = true;
  end;
  if overlapRatioW == 0
     statW = true;
  end;
  if overlapRatioI == 0
```

```
   statI= true;
end;
if overlapRatioG == 0
   statG = true;
end;
if overlapRatioC == 0
   statC = true;
end;
if overlapRatioE == 0
   statE = true;
end;
if overlapRatioO == 0
   statO = true;
end;
if overlapRatioH == 0
   statH = true;
end;
if overlapRatioV == 0
   statV = true;
end;
if overlapRatioR == 0
   statR = true;
end;
if overlapRatioP == 0
   statP = true;
end;
if overlapRatioJ == 0
   statJ = true;
end;
if overlapRatioB == 0
   statB = true;
end;
if overlapRatioT == 0
   statT = true;
end;
if overlapRatioA == 0
   statA = true;
end;
if overlapRatioK == 0
   statK = true;
end;
if overlapRatioN == 0
   statN = true;
end;
if overlapRatioZ == 0
   statZ = true;
end;
if overlapRatioS == 0
   statS = true;
end;
if overlapRatioL == 0
   statL = true;
end;
if overlapRatioM == 0
   statM = true;
end;
if overlapRatioD == 0
```

```
        statD = true;
    end;
    if overlapRatioY == 0
        statY = true;
    end;
        if overlapRatio0 == 0
        stat0 = true;
    end;
        if overlapRatio1 == 0
        stat1 = true;
    end;
        if overlapRatio2 == 0
        stat2 = true;
    end;
        if overlapRatio3 == 0
        stat3 = true;
    end;
        if overlapRatio4 == 0
        stat4 = true;
    end;
        if overlapRatio5 == 0
        stat5 = true;
    end;
        if overlapRatio6 == 0
        stat6 = true;
    end;
        if overlapRatio7 == 0
        stat7 = true;
    end;
        if overlapRatio8 == 0
        stat8 = true;
    end;
        if overlapRatio9 == 0
        stat9 = true;
    end;
        if overlapRatioBack == 0
        statBack = true;
    end;
        if overlapRatioEnter == 0
        statEnter = true;
    end;
        if overlapRatioSpace == 0
        statSpace = true;
    end;


    if overlapRatioQ > defaultOverlapRatio & statQ == true;

tQ=timer('TimerFcn','if(overlapRatioQ>defaultOverlapRatio);robot.keyPress(java.a
wt.event.KeyEvent.VK_Q);robot.keyRelease(java.awt.event.KeyEvent.VK_Q);end;',...
            'StartDelay',t);
        start(tQ)
        statQ = false;
    end
    if overlapRatioU > defaultOverlapRatio & statU == true;
```

```
tU=timer('TimerFcn','if(overlapRatioU>defaultOverlapRatio);robot.keyPress(java.a
wt.event.KeyEvent.VK_U);robot.keyRelease(java.awt.event.KeyEvent.VK_U);end;',...
            'StartDelay',t);
        start(tU)
        statU = false;
    end
    if overlapRatioF > defaultOverlapRatio & statF == true;

tF=timer('TimerFcn','if(overlapRatioF>defaultOverlapRatio);robot.keyPress(java.a
wt.event.KeyEvent.VK_F);robot.keyRelease(java.awt.event.KeyEvent.VK_F);end;',...
            'StartDelay',t);
        start(tF)
        statF = false;
    end
    if overlapRatioX > defaultOverlapRatio & statX == true;

tX=timer('TimerFcn','if(overlapRatioX>defaultOverlapRatio);robot.keyPress(java.a
wt.event.KeyEvent.VK_X);robot.keyRelease(java.awt.event.KeyEvent.VK_X);end;',...
            'StartDelay',t);
        start(tX)
        statX = false;
    end
    if overlapRatioW > defaultOverlapRatio & statW == true;

tW=timer('TimerFcn','if(overlapRatioW>defaultOverlapRatio);robot.keyPress(java.a
wt.event.KeyEvent.VK_W);robot.keyRelease(java.awt.event.KeyEvent.VK_W);end;',...
            'StartDelay',t);
        start(tW)
        statW = false;
    end
    if overlapRatioI > defaultOverlapRatio & statI == true;

tI=timer('TimerFcn','if(overlapRatioI>defaultOverlapRatio);robot.keyPress(java.a
wt.event.KeyEvent.VK_I);robot.keyRelease(java.awt.event.KeyEvent.VK_I);end;',...
            'StartDelay',t);
        start(tI)
        statI = false;
    end
    if overlapRatioG > defaultOverlapRatio & statG == true;

tG=timer('TimerFcn','if(overlapRatioG>defaultOverlapRatio);robot.keyPress(java.a
wt.event.KeyEvent.VK_G);robot.keyRelease(java.awt.event.KeyEvent.VK_G);end;',...
            'StartDelay',t);
        start(tG)
        statG = false;
    end
    if overlapRatioC > defaultOverlapRatio & statC == true;

tC=timer('TimerFcn','if(overlapRatioC>defaultOverlapRatio);robot.keyPress(java.a
wt.event.KeyEvent.VK_C);robot.keyRelease(java.awt.event.KeyEvent.VK_C);end;',...
            'StartDelay',t);
        start(tC)
        statC = false;
    end
    if overlapRatioE > defaultOverlapRatio & statE == true;
```

```
        tE=timer('TimerFcn','if
(overlapRatioE>defaultOverlapRatio);robot.keyPress(java.awt.event.KeyEvent.VK_E)
;robot.keyRelease(java.awt.event.KeyEvent.VK_E);end;',...
            'StartDelay',t);
        start(tE)
        statE = false;
    end
    if overlapRatioO > defaultOverlapRatio & statO == true;

tO=timer('TimerFcn','if(overlapRatioO>defaultOverlapRatio);robot.keyPress(java.a
wt.event.KeyEvent.VK_O);robot.keyRelease(java.awt.event.KeyEvent.VK_O);end;',...
            'StartDelay',t);
        start(tO)
        statO = false;
    end
    if overlapRatioH > defaultOverlapRatio & statH == true;

tH=timer('TimerFcn','if(overlapRatioH>defaultOverlapRatio);robot.keyPress(java.a
wt.event.KeyEvent.VK_H);robot.keyRelease(java.awt.event.KeyEvent.VK_H);end;',...
            'StartDelay',t);
        start(tH)
        statH = false;
    end
    if overlapRatioV > defaultOverlapRatio & statV == true;

tV=timer('TimerFcn','if(overlapRatioV>defaultOverlapRatio);robot.keyPress(java.a
wt.event.KeyEvent.VK_V);robot.keyRelease(java.awt.event.KeyEvent.VK_V);end;',...
            'StartDelay',t);
        start(tV)
        statV = false;
    end
    if overlapRatioR > defaultOverlapRatio & statR == true;

tR=timer('TimerFcn','if(overlapRatioR>defaultOverlapRatio);robot.keyPress(java.a
wt.event.KeyEvent.VK_R);robot.keyRelease(java.awt.event.KeyEvent.VK_R);end;',...
            'StartDelay',t);
        start(tR)
        statR = false;
    end
    if overlapRatioP > defaultOverlapRatio & statP == true;

tP=timer('TimerFcn','if(overlapRatioP>defaultOverlapRatio);robot.keyPress(java.a
wt.event.KeyEvent.VK_P);robot.keyRelease(java.awt.event.KeyEvent.VK_P);end;',...
            'StartDelay',t);
        start(tP)
        statP = false;
    end
    if overlapRatioJ > defaultOverlapRatio & statJ == true;

tJ=timer('TimerFcn','if(overlapRatioJ>defaultOverlapRatio);robot.keyPress(java.a
wt.event.KeyEvent.VK_J);robot.keyRelease(java.awt.event.KeyEvent.VK_J);end;',...
            'StartDelay',t);
        start(tJ)
        statJ = false;
    end
    if overlapRatioB > defaultOverlapRatio & statB == true;
```

```
tB=timer('TimerFcn','if(overlapRatioB>defaultOverlapRatio);robot.keyPress(java.a
wt.event.KeyEvent.VK_B);robot.keyRelease(java.awt.event.KeyEvent.VK_B);end;',...
            'StartDelay',t);
        start(tB)
        statB = false;
    end
    if overlapRatioT > defaultOverlapRatio & statT == true;

tT=timer('TimerFcn','if(overlapRatioT>defaultOverlapRatio);robot.keyPress(java.a
wt.event.KeyEvent.VK_T);robot.keyRelease(java.awt.event.KeyEvent.VK_T);end;',...
            'StartDelay',t);
        start(tT)
        statT = false;
    end
    if overlapRatioA > defaultOverlapRatio & statA == true;

tA=timer('TimerFcn','if(overlapRatioA>defaultOverlapRatio);robot.keyPress(java.a
wt.event.KeyEvent.VK_A);robot.keyRelease(java.awt.event.KeyEvent.VK_A);end;',...
            'StartDelay',t);
        start(tA)
        statA = false;
    end
    if overlapRatioK > defaultOverlapRatio & statK == true;

tK=timer('TimerFcn','if(overlapRatioK>defaultOverlapRatio);robot.keyPress(java.a
wt.event.KeyEvent.VK_K);robot.keyRelease(java.awt.event.KeyEvent.VK_K);end;',...
            'StartDelay',t);
        start(tK)
        statK = false;
    end
    if overlapRatioN > defaultOverlapRatio & statN == true;

tN=timer('TimerFcn','if(overlapRatioN>defaultOverlapRatio);robot.keyPress(java.a
wt.event.KeyEvent.VK_N);robot.keyRelease(java.awt.event.KeyEvent.VK_N);end;',...
            'StartDelay',t);
        start(tN)
        statN = false;
    end
    if overlapRatioZ > defaultOverlapRatio & statZ == true;

tZ=timer('TimerFcn','if(overlapRatioZ>defaultOverlapRatio);robot.keyPress(java.a
wt.event.KeyEvent.VK_Z);robot.keyRelease(java.awt.event.KeyEvent.VK_Z);end;',...
            'StartDelay',t);
        start(tZ)
        statZ = false;
    end
    if overlapRatioS > defaultOverlapRatio & statS == true;

tS=timer('TimerFcn','if(overlapRatioS>defaultOverlapRatio);robot.keyPress(java.a
wt.event.KeyEvent.VK_S);robot.keyRelease(java.awt.event.KeyEvent.VK_S);end;',...
            'StartDelay',t);
        start(tS)
        statS = false;
    end
    if overlapRatioL > defaultOverlapRatio & statL == true;
```

```
tL=timer('TimerFcn','if(overlapRatioL>defaultOverlapRatio);robot.keyPress(java.a
wt.event.KeyEvent.VK_L);robot.keyRelease(java.awt.event.KeyEvent.VK_L);end;',...
            'StartDelay',t);
        start(tL)
        statL = false;
    end
    if overlapRatioM > defaultOverlapRatio & statM == true;

tM=timer('TimerFcn','if(overlapRatioM>defaultOverlapRatio);robot.keyPress(java.a
wt.event.KeyEvent.VK_M);robot.keyRelease(java.awt.event.KeyEvent.VK_M);end;',...
            'StartDelay',t);
        start(tM)
        statM = false;
    end
    if overlapRatioD > defaultOverlapRatio & statD == true;

tD=timer('TimerFcn','if(overlapRatioD>defaultOverlapRatio);robot.keyPress(java.a
wt.event.KeyEvent.VK_D);robot.keyRelease(java.awt.event.KeyEvent.VK_D);end;',...
            'StartDelay',t);
        start(tD)
        statD = false;
    end
    if overlapRatioY > defaultOverlapRatio & statY == true;

tY=timer('TimerFcn','if(overlapRatioY>defaultOverlapRatio);robot.keyPress(java.a
wt.event.KeyEvent.VK_Y);robot.keyRelease(java.awt.event.KeyEvent.VK_Y);end;',...
            'StartDelay',t);
        start(tY)
        statY = false;
    end
    if overlapRatio0 > defaultOverlapRatio & stat0 == true;

t0=timer('TimerFcn','if(overlapRatio0>defaultOverlapRatio);robot.keyPress(java.a
wt.event.KeyEvent.VK_0);robot.keyRelease(java.awt.event.KeyEvent.VK_0);end;',...
            'StartDelay',t);
        start(t0)
        stat0 = false;
    end
    if overlapRatio1 > defaultOverlapRatio & stat1 == true;

t1=timer('TimerFcn','if(overlapRatio1>defaultOverlapRatio);robot.keyPress(java.a
wt.event.KeyEvent.VK_1);robot.keyRelease(java.awt.event.KeyEvent.VK_1);end;',...
            'StartDelay',t);
        start(t1)
        stat1 = false;
    end
    if overlapRatio2 > defaultOverlapRatio & stat2 == true;

t2=timer('TimerFcn','if(overlapRatio2>defaultOverlapRatio);robot.keyPress(java.a
wt.event.KeyEvent.VK_2);robot.keyRelease(java.awt.event.KeyEvent.VK_2);end;',...
            'StartDelay',t);
        start(t2)
        stat2 = false;
    end
    if overlapRatio3 > defaultOverlapRatio & stat3 == true;
```

```
t3=timer('TimerFcn','if(overlapRatio3>defaultOverlapRatio);robot.keyPress(java.a
wt.event.KeyEvent.VK_3);robot.keyRelease(java.awt.event.KeyEvent.VK_3);end;',...
            'StartDelay',t);
        start(t3)
        stat3 = false;
    end
    if overlapRatio4 > defaultOverlapRatio & stat4 == true;

t4=timer('TimerFcn','if(overlapRatio4>defaultOverlapRatio);robot.keyPress(java.a
wt.event.KeyEvent.VK_4);robot.keyRelease(java.awt.event.KeyEvent.VK_4);end;',...
            'StartDelay',t);
        start(t4)
        stat4 = false;
    end
    if overlapRatio5 > defaultOverlapRatio & stat5 == true;

t5=timer('TimerFcn','if(overlapRatio5>defaultOverlapRatio);robot.keyPress(java.a
wt.event.KeyEvent.VK_5);robot.keyRelease(java.awt.event.KeyEvent.VK_5);end;',...
            'StartDelay',t);
        start(t5)
        stat5 = false;
    end
    if overlapRatio6 > defaultOverlapRatio & stat6 == true;

t6=timer('TimerFcn','if(overlapRatio6>defaultOverlapRatio);robot.keyPress(java.a
wt.event.KeyEvent.VK_6);robot.keyRelease(java.awt.event.KeyEvent.VK_6);end;',...
            'StartDelay',t);
        start(t6)
        stat6 = false;
    end
    if overlapRatio7 > defaultOverlapRatio & stat7 == true;

t7=timer('TimerFcn','if(overlapRatio7>defaultOverlapRatio);robot.keyPress(java.a
wt.event.KeyEvent.VK_7);robot.keyRelease(java.awt.event.KeyEvent.VK_7);end;',...
            'StartDelay',t);
        start(t7)
        stat7 = false;
    end
    if overlapRatio8 > defaultOverlapRatio & stat8 == true;

t8=timer('TimerFcn','if(overlapRatio8>defaultOverlapRatio);robot.keyPress(java.a
wt.event.KeyEvent.VK_8);robot.keyRelease(java.awt.event.KeyEvent.VK_8);end;',...
            'StartDelay',t);
        start(t8)
        stat8 = false;
    end
    if overlapRatio9 > defaultOverlapRatio & stat9 == true;

t9=timer('TimerFcn','if(overlapRatio9>defaultOverlapRatio);robot.keyPress(java.a
wt.event.KeyEvent.VK_9);robot.keyRelease(java.awt.event.KeyEvent.VK_9);end;',...
            'StartDelay',t);
        start(t9)
        stat9 = false;
    end
    if overlapRatioBack > defaultOverlapRatio & statBack == true;
```

```
tBack=timer('TimerFcn','if(overlapRatioBack>defaultOverlapRatio);robot.keyPress(
java.awt.event.KeyEvent.VK_BACK_SPACE);robot.keyRelease(java.awt.event.KeyEvent.
VK_BACK_SPACE);end;',...
            'StartDelay',t);
        start(tBack)
        statBack = false;
    end
    if overlapRatioEnter > defaultOverlapRatio & statEnter == true;

tEnter=timer('TimerFcn','if(overlapRatioEnter>defaultOverlapRatio);robot.keyPres
s(java.awt.event.KeyEvent.VK_ENTER);robot.keyRelease(java.awt.event.KeyEvent.VK_
ENTER);end;',...
            'StartDelay',t);
        start(tEnter)
        statEnter = false;
    end
        if overlapRatioSpace > defaultOverlapRatio & statSpace == true;

tSpace=timer('TimerFcn','if(overlapRatioSpace>defaultOverlapRatio);robot.keyPres
s(java.awt.event.KeyEvent.VK_SPACE);robot.keyRelease(java.awt.event.KeyEvent.VK_
SPACE);end;',...
            'StartDelay',t);
        start(tSpace)
        statSpace = false;
    end

    }
```