UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO
LABORATORIJ ZA RAČUNALNIŠKI VID

DIPLOMSKO DELO

DETEKCIJA POLOŽAJA KROGLICE NA IGRALNIŠKI RULETI S POMOČJO RAČUNALNIŠKEGA VIDA

Tadej Žgur

Mentor: prof. dr. Franc Solina, univ. dipl. inž.

Ljubljana, april 2005

Kazalo

Slike

Tabele

DETEKCIJA POLOŽAJA KROGLICE NA IGRALNIŠKI RULETI S POMOČJO RAČUNALNIŠKEGA VIDA

Tadej Žgur

Mentor: prof. dr. Franc Solina

POVZETEK

Delo obsega opis implementacije sistema računalniškega vida za problem, ki bi ga lahko šteli med industrijske probleme računalniškega vida, to je problem detekcije številke, na katero je padla kroglica pri igralniški ruleti. Opisani so algoritmi, ki so bili razviti za implementacijo tega specialnega sistema. Podani so tudi rezultati testiranja opisanega sistema nad bazo 42-ih posnetkov, ki pokažejo šibke točke sistema. Predlagane so tudi smernice za nadaljnji razvoj opisanega sistema.

KLJUČNE BESEDE

računalniški vid, detekcija robov, igralniška ruleta, detekcija številke iz igralniške rulete, igralnica, hough, canny

DETECTION OF BALL POSITION ON CASINO ROULETTE WITH ASSISTANCE OF COMPUTER VISION

Tadej Žgur

Supervisor: prof. dr. Franc Solina

ABSTRACT

This work contains the implementation description of a system which could belong to industrial problems of computer vision, that is the problem of number detection on casino roulette, where the ball is. All algorithms which have been developed for the implementation of this very special system, are described. The effectiveness of this system was tested on 42 snapshots which shows weak points of the system.

KEY WORDS

computer vision, edge detection, casino roulette, number detection on casino roulette, casino, hough, canny

Poglavje 1

Uvod v računalniški vid

Vid je čut, preko katerega ljudje pridobivamo informacije o zunanjem svetu, da si ga lahko predstavljamo. Predstavljanje tega zunanjega sveta se nam zdi zelo preprosto in zdi se nam, da je vid nekaj preprostega in lahko predstavljivega. Samoumevno se nam tudi zdi, da lahko interpretiramo sliko, na njej prepoznamo predmete, obraze, napise itd. V težave pa pridemo, ko poskušamo najti postopke oz. pravila, po katerih razpoznamo in si interpretiramo stvari na sliki. Šele takrat se zavedamo kompleksnosti analize, ki jo opravljajo naši možgani, da lahko interpretirajo informacijo, pridobljeno iz slike, ki jo pridobivajo preko oči. Tako postane zelo težavna naloga, naučiti računalnik oz. z drugo besedo napisati algoritem po katerem bi ta lahko predmete iz slik razpoznaval ali interpretiral na enak način kot naši možgani in bi bil pri tem enako uspešen.
Področje računalništva, ki se ukvarja s tem problemom, se imenuje računalniški vid. Računalniški vid je področje umetne inteligence, ki poskuša prenesti vizualno informacijo iz dvodimenzionalnih ali pa iz globinskih slik (range images) v računalnik. Pri tem poskuša posnemati človeški vid. Do nedavnega je bila največja težava računalniškega vida premajhna procesorska moč računalnikov. Algoritmi računalniškega vida so večinoma časovno in prostorsko zelo kompleksni in s tem procesorsko zelo zahtevni, zato je procesorska moč, potrebna za uporabo računalniškega vida v praktičnih problemih, zelo pomemben faktor. Ta težava je onemogočala uporabo računalniškega vida v problemih, ki zahtevajo delovanje v realnem času. Hitro povečevanje ter pocenitev procesorske moči in spomina računalnikov ter pocenitev kamer v zadnjih letih, je omogočilo uporabo algoritmov računalniškega vida v praktičnih problemih, kar je pripomoglo k zelo hitri širitvi tehnologije računalniškega vida. Računalniški vid ima zelo širok spekter uporabnosti. Srečamo ga na zelo veliko področjih, kot npr.: nadzor, industrijska kontrola kvalitete, promet, robotika, multimedija, vojska, vesolje in aeronautika, medicina itd.
Računalniški vid je relativno mlada veja znanosti, stara približno 30 let, in obstaja veliko problemov, za katere se ve, da obstaja rešitev z računalniškim vidom, vendar

še ni bila odkrita. Zaradi tega in zaradi napredka tehnologije, ki omogoča uporabo metod računalniškega vida, je to področje v zelo hitrem vzponu. Na svetu je veliko raziskovalnih skupin, ki poskušajo razviti nove metode oz. uporabiti obsoječe algoritme za reševanje ogromnega števila problemov, ki se jih da rešiti z računalniškim vidom. Algoritmi računalniškega vida omogočajo detekcijo značilk na slikah, pridobivanje 3D informacije iz 2D slik, prepoznavanje vzorcev, detekcijo gibanja, prepoznavanje objektov itd. Aplikacije računalniškega vida so lahko namenjene pomoči ljudem, npr. v medicini, prepoznavanju prometnih znakov in opozarjanju na nevarnosti itd. ali pa nadomeščanju človeške prisotnosti in odpravljanju faktorja človeške zmotljivosti, utrujenosti itd., to so npr. razni nadzorni sistemi, kontrola kvalitete itd.

V pričujočem delu bomo opisali in implementirali sistem računalniškega vida za problem, ki bi ga lahko šteli med industrijske probleme računalniškega vida, to je problem detekcije številke, na katero je padla kroglica pri igralniški ruleti. Najprej bomo opisali industrijske probleme računalniškega vida in njihove značilnosti ter opisali potek igre igralniške rulete. Nato bomo opisali problem detekcije številke, na katero je padla kroglica pri igralniški ruleti in težave pri reševanju tega problema. Sledil bo opis implementacije tega sistema in opis algoritmov, ki so bili razviti in uporabljeni. V poglavju o rezultatih bodo opisani rezultati testiranja nad testno množico posnetkov in prikazane šibke točke sistema, ki so razvidne iz negativnih rezultatov testiranja. Na koncu bodo podane smernice za nadaljnje delo in razvoj opisanega sistema.

Poglavje 2

Računalniški vid v industriji

Kot je bilo povedano že v prejšnjem poglavju, lahko srečamo aplikacije računalniškega vida na veliko področjih. Zelo široko področje, kjer jih lahko srečamo je industrija. V industriji obstaja zelo veliko problemov, ki jih je mogoče elegantno in poceni rešiti s pomočjo računalniškega vida, zato obstaja veliko komercialno dostopnih sistemov, ki so namenjeni reševanju industrijskih problemov. Nekatere vrste industrijskih problemov, ki so rešene s pomočjo računalniškega vida, so [7]:

- Iskanje napak na površini raznih predmetov iz različnih materialov.
- Izdelava tri dimenzionalnih CAD (Computer Aided Design) modelov s pomočjo dvo dimenzionalnih slik.
- Pregledovanje profila avtomobilskih pnevmatik.
- Prepoznavanje raznih oblik predmetov in črk ter njihovo štetje in klasificiranje.
- Prepoznavanje objektov iz mikroskopskih slik.
- Branje raznih kod.

Računalniški vid na mnogih področjih v industriji poskuša nadomestiti ljudi oz. človeški vid. Mnoga dela, ki jih opravljajo ljudje v industriji, so namreč zelo naporna in monotona. Tako delo je npr. nadzor kvalitete in iskanje morebitnih napak proizvodov za tekočim trakom. V mnogih takih primerih lahko aplikacija računalniškega vida nadomesti človeški vid in se izkaže kot vsaj enako uspešna. V nasprotju z delavcem, ki opravlja enako delo, je tudi cenejša. Upoštevati je potrebno, da mora biti delavec skoncentriran, ko opravlja tako delo, kar postane naporno. Dejstvo je, da smo ljudje, v nasprotju z računalniki, zmotljivi in da na nas vpliva mnogo zunanjih dejavnikov, ki na računalnik nimajo vpliva. Zaradi vsega tega srečamo največ aplikacij računalniškega vida v industriji pri taki vrsti del, ki so monotona in naporna,

zato se je računalniški vid uveljavil v industriji najbolj pri iskanju raznih proizvodnih napak za tekočim trakom, ki se odkrivajo s pregledovanjem površine predmetov.
Drugi razlog, zakaj se je računalniški vid uveljavil in izkazal kot zelo uspešen v industriji je, da lahko pri večini industrijskih problemov vplivamo ne samo na notranje parametre sistema, ampak tudi na zunanje parametre sistema, torej imamo vpliv na okolje, v katerem se uporablja računalniški vid. Nekateri izmed zunanjih parametrov sistema računalniškega vida so:

- osvetlitev,
- položaj kamere v prostoru,
- usmeritev kamere.

Zunanjih parametrov se, v nasprotju z industrijskimi problemi, v veliki večini problemov računalniškega vida ne da spreminjati ali pa se jih da le delno spreminjati. Eden izmed takih problemov je npr. detekcija prometnih znakov. Ti zunanji parametri imajo zelo veliko vlogo pri uspešnosti algoritmov računalniškega vida. Ponavadi so spremenljivi zunanji parametri vzrok za slabše delovanje sistemov računalniškega vida. Parameter, ki najbolj vpliva na uspešnost sistema, je osvetlitev. Osvetlitev spreminja barve predmetov, ustvarja sence, povzroča zrcalne svetlobne odboje, deformira predmete na slikah itd. Osvetlitev je največji problem v računalniškem vidu. Na osvetlitev večinoma nimamo nikakršnega vpliva, še posebej, če gre za sistem, ki deluje na prostem, kjer je osvetljenost odvisna od sončnih žarkov. V nasprotju z večino drugih problemov računalniškega vida, imamo pri industrijskih problemih opravka s sistemom, ki deluje v zaprtih prostorih, tako da lahko vplivamo na osvetlitev predmetov. Ta možnost ustvarjanja pogojev, v katerih sistem računalniškega vida deluje, je eden najpomembnejših faktorjev, zakaj se industrijski problemi učinkovito rešujejo z računalniškim vidom. Naprimer pri sistemih nadzora kvalitete proizvodov, kjer se iščejo napake na površini, se sistem ponavadi povsem izolira od okolice, tako da zunanji pogoji nimajo vpliva nanj in da se lahko ustvari primerna osvetlitev itd.
Še ena posebnost industrijskih problemov računalniškega vida v primerjavi z ostalimi problemi je, da imamo pri tej vrsti problemov veliko informacij, ki so vnaprej definirane in jih lahko uporabimo pri razvoju sistema. Nekatere take informacije so naprimer:

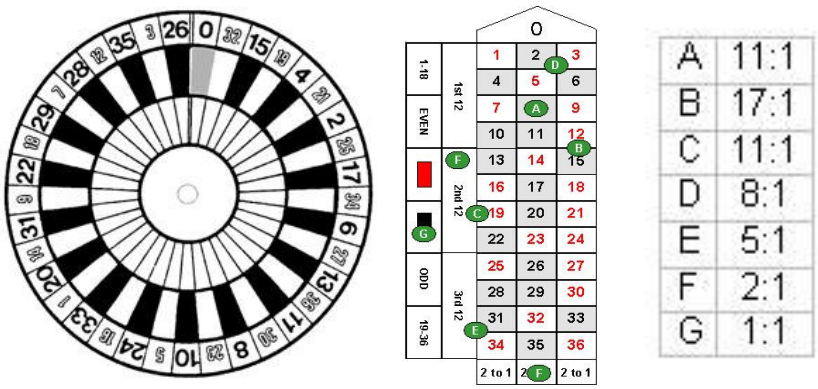
- vemo, kaj se nahaja na sliki,
- poznamo obliko predmetov, ki jih iščemo na sliki,
- vemo vsaj približno, kje na sliki so predmeti, ki jih iščemo,

- poznamo velikost predmetov na sliki.

Kot na dlani je, da te vnaprej definirane informacije o sliki, ki jo zajemamo, bistveno pripomorejo k uspešnosti in hitrosti sistemov računalniškega vida. Tako vemo, na katere dele slike se moramo osredotočiti in s tem takoj zanemarimo ostale dele slike, ki bi samo motili sistem oz. ga upočasnili. Tako postanejo sistemi računalniškega vida v industriji zelo specialni in prilagojeni reševanju točno določenega problema. Zaradi vsega tega je te sisteme večinoma težko prilagoditi za reševanje drugih vrst problemov.
Principi, ki se uporabljajo pri industrijskih problemih računalniškega vida, so večinoma zelo podobni. Vsi težijo k temu, da so kar se da enostavni ter posledično hitri. Tipične metode oz. algoritmi, ki se uporabljajo v takih problemih, so:

- Detekcija robov (Edge detection).
- Upragovljenje (Tresholding).
- Analiziranje barvnih histogramov slik.
- Razpoznavanje vzorcev (Pattern recognition).
- Detekcija značilk (Feature detection).

V nadaljevanju pričujočega dela bomo implementirali sistem računalniškega vida za problem, ki bi ga lahko šteli tudi med industrijske probleme, in sicer bomo predlagali implementacijo za problem detekcije številke, na katero je padla kroglica pri igralniški ruleti.
Preden začnemo s podrobnejšim opisom tega sistema in njegovih problemov, je dobro, da razložimo, kaj igralniška ruleta sploh je in kako poteka igra rulete. Igralniška ruleta je sestavljena iz mize s številčnico, na katero se polaga žetone za stave, in iz kolesa rulete (glej sliko 2.1). Na kolesu rulete so številke, ki so narisane na mizi rulete. Na številke se stavi tako, da se nanje postavlja enega ali več žetonov. Žetoni so lahko različnih vrednosti. Oseba, ki vodi igro, se imenuje croupier (izgovorjava krupje). Ko croupier oznani, da so stave odprte, se lahko polaga žetone na številke na mizi in se s tem nanje stavi (kot je prikazano na sredinski sliki 2.1). Stavi se lahko na posamezno številko, na vogale številk, na posamezen stolpec, na barvo itd. Stave so odprte lahko tudi, ko se kroglica že vrti po kolesu rulete, dokler croupier ne oznani, da so stave zaprte. Takrat se ne sme polagati žetonov na mizo in čaka se, da kroglica pade na določeno številko. Ko kroglica pade na številko, croupier na glas pove številko, na kateri je kroglica, ter našteje vse tipe stav, ki so zmagale (glej desno sliko 2.1). Stave, ki so zgubile, se poberejo iz mize, dobitne stave (oz. žetoni) pa ostanejo na mizi na istih številkah, dokler dobitki niso izplačani (v žetonih). Kot



Slika 2.1: Od leve proti desni: Kolo rulete s številčnico, prikaz možnih stav in pozicije žetonov pri igri rulete ter dobitki pri stavah, ki so obratno sorazmerni verjetnosti dobitka.

je prikazano v tabeli in na sliki 2.1, velja pravilo, da je dobitek večji, če je manjša verjetnost dobitka in manjši, če je verjetnost dobitka večja. Ko so stave izplačane, se lahko začne nova igra.
V nadaljevanju pričujočega dela bomo predlagali sistem računalniškega vida, ki bo delno nadomestil croupierja. Sistem, ki ga bomo opisali, bo znal prebrati, na katero številko je padla kroglica. Enak sistem, ki uporablja senzorje in elektroniko je že narejen, poskušali ga bomo implementirati še s pomočjo računalniškega vida. Namesto mize, na kateri so številke, na katere se stavi, imamo implementiran program, preko katerega se sprejemajo stave. Kolo rulete je enako kot pri navadnih ruletah, razlika je samo v tem, da je pokrito s steklom, ki preprečuje zunanje posege v igro. Za razliko od nekaterih industrijskih problemov računalniškega vida, pri tem problemu nimamo vpliva na svetlobo. Za vse igralniške avtomate in za igralnice nasploh je značilno, da se vsi prisotni predmeti bleščijo kolikor se da. Tudi avtomatska igralniška ruleta ni nobena izjema. Svetloba se od kolesa rulete in številčnice večinoma odbija zrcalno, kar je zelo moteče za sistem računalniškega vida, ki ga bomo implementirali, saj je svetloba največji problem računalniškega vida. Kljub temu, da na svetlobo nimamo vpliva, lahko vplivamo in pridobimo informacije o drugih zunanjih parametrih sistema računalniškega vida, kar je značilno za vse industrijske probleme računalniškega vida.
Informacije, ki jih imamo o zunanjih parametrih tega sistema in na katere lahko delno vplivamo so:

- Velikost kroglice.
- Položaj številčnice kolesa rulete na sliki.

- Položaj kamere.
- Usmeritev kamere.
- Položaj kroglice, ko pade na številko.

Kot pri drugih industrijskih problemih računalniškega vida, tudi pri tem sistemu, te informacije bistveno pripomorejo h hitrosti in uspešnosti implementiranega sistema. S sistemom za detektiranje številke, na katero je padla kroglica, ki ga bomo implementirali, bi lahko nadomestili croupierja.
Druga možnost uporabe sistema za detektiranje, na katero številko je padla kroglica je, da bi služil za pomoč croupierju pri igri. Ker vemo, da so sistemi implementirani z metodami računalniškega vida le redko stoodstotno zanesljivi in da je v igralnicah stoodstotna zanesljivost nujna zahteva, bi lahko uporabili sistem v smislu, ki bo opisan v nadaljnjem besedilu. Računalniški program bi pobiral stave, vodil igro in zavrtel kroglico ter ruleto. Prisotem bi bil tudi croupier, ki bi nadzoroval igro več avtomatskih rulet hkrati. Nadzoroval bi jo lahko kar preko kamere, ki snema sliko za vhod v implementiran sistem računalniškega vida za detekcijo številke, na katero je padla kroglica. Ta sistem bi, potem ko je kroglica padla, prebral številko, na katero je kroglica padla. V primeru, da bi se sistem zmotil, bi croupier popravil napako, tako da bi on povedal kje je kroglica. Tako bi lahko en croupier nadzoroval več rulet hkrati in stroški igralnice bi se zmanjšali. Kot že rečeno bi lahko to croupier delal kar preko kamere, ki je v sistemu uporabljena, isto kamero pa bi lahko uporabili tudi za igranje rulete na daljavo, recimo iz hotelske sobe ali pa preko interneta, lahko tudi preko mobilnega telefona. Tako bi lahko kamera bila uporabljena za več namenov hkrati.
Problemi računalniškega vida pri igralniških igrah so torej neka posebna vrsta industrijskih problemov računalniškega vida, z edino razliko, ki se izkaže kot precej oteževalna, da nimamo vpliva na svetlobo oz. na osvetlitev. Težave so tudi zaradi nepoštenih igralcev, ki bi hoteli zavesti naš sistem v njihovo korist.

Poglavje 3

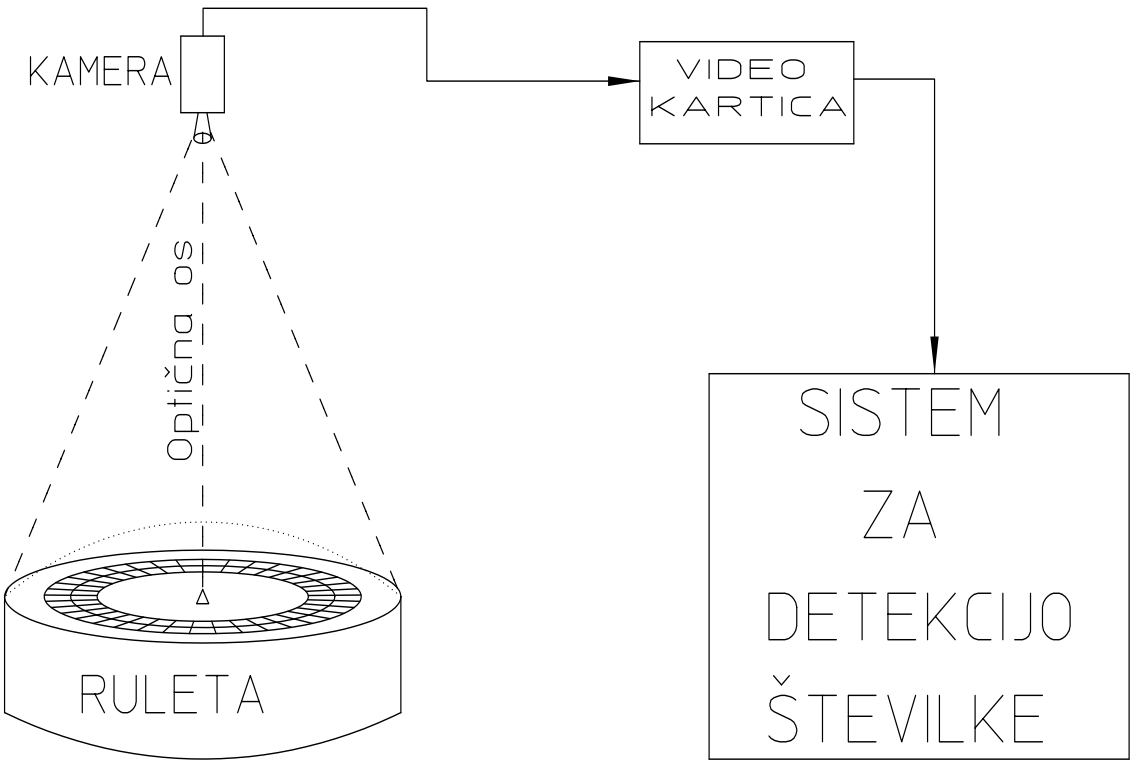
Opis problema detekcije številke

Imamo avtomatsko igralniško ruleto. To je taka vrsta rulete, kjer obstaja sistem, ki avtomatsko zazna, na katero številko je padla kroglica. Sistem je implementiran s senzorji in elektroniko, ki ta sistem upravlja. Sistem deluje tako, da zavrti kolo rulete, nato izpihne (s pomočjo zraka) kroglico, nato počaka, da senzor na določeni številki zazna, da je kroglica padla nanjo ter kroglico potem pobere iz številčnice in prikaže številko na zaslonu, ne da bi čakal, da se kolo rulete ustavi (umiri). Opisan sistem senzorjev in elektronike se je izkazal kot zelo uspešen. Sistem za detektiranje številke, na katero je padla kroglica, je že implementiran z elektroniko in senzorji. Radi bi implementirali enak sistem, ki bi detektiral številko, na katero je padla kroglica, s pomočjo kamere in ustrezne programske opreme. Zanima nas predvsem, kako bi se tak sistem obnesel v praksi.

Nad avtomatsko igralniško ruleto je postavljena črnobela kamera, kot to prikazuje slika 3.1. Kamera je postavljena centralno nad kolo rulete (roulette wheel), optična os kamere gre skozi središče kolesa rulete. Ugotoviti želimo, kdaj in na katero številko je padla kroglica. Številk na številčnici rulete, iz slike 3.2, je 37 in sicer 36 številk ter ničla, vendar obstajajo rulete tudi z drugačnim številom številk na številčnici, zato je želja narediti sistem, ki bo neodvisen od tega, koliko številk je na številčnici. Avtomatska ruleta je pokrita s pokrovom, ki je steklen ali pa je iz pleksi stekla. Ta pokrov varuje ruleto pred zunanjimi posegi, večinoma s strani nepoštenih igralcev.

Kot je značilno za industrijske probleme računalniškega vida, imamo tudi tukaj veliko uporabnih informacij, ki nam olajšujejo implementacijo sistema, vnaprej znanih. Informacije, ki jih lahko upoštevamo pri razvoju sistema, so naslednje:

- Kamera je postavljena pravokotno nad ruleto.
- Vemo, da je na zajeti sliki kolo igralniške rulete in da se razteza čez celo sliko.
- Znana je velikost kroglice (radij).



Slika 3.1: Skica opisanega sistema

- Poznamo center številčnice oz. kolesa rulete, na sliki.
- Vemo, kje na sliki se nahaja številčnica.
- Vemo, kje se nahaja kroglica, ko je padla na številko.
- Vemo, da so predalčki (sockets) na številčnici zaporedno črno-svetli, le pri ničli je barva svetlejša od črne in temnejša od svetle.

Te informacije pripomorejo k uspešnosti implementacije, saj se z njimi izognemo nekaterim problemom. S temi informacijami pridobimo tudi na hitrosti implementiranega sistema, saj se lahko osredotočimo le na zanimive regije na sliki. Vsi ti podatki, ki jih dobimo iz vnaprej znanih informacij o sistemu, se morajo nanašati na trenutno resolucijo kamere, torej morajo biti podani v slikovnih elementih slike. Določi se jih potem, ko se nastavi kamera in je zajeta slika kolesa rulete raztegnjena čez celo sliko, kot je to prikazano na sliki 3.2. Pri implementaciji točno vemo, kaj se nahaja na sliki in kaj iščemo. Kroglico npr. iščemo samo tam, kjer se nahaja, ko je padla na številko.

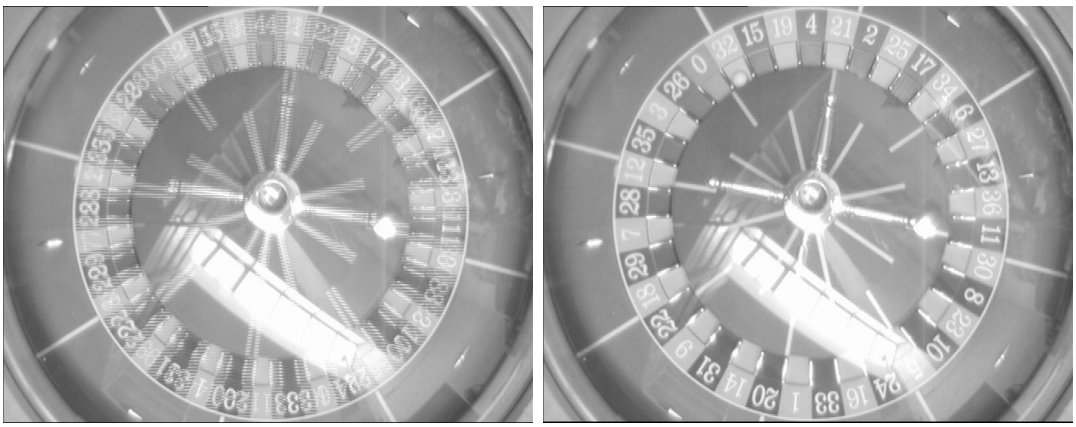
Zgoraj naštete vnaprej znane informacije veliko pripomorejo k uspešnosti sistema, vendar pa obstaja zelo veliko lastnosti, ki otežujejo implementacijo sistema in povzročajo težave pri delovanju sistema. Največji problemi pri implementaciji opisanega sistema so:

Slika 3.2: Slika kolesa rulete (roulette wheel, v nadaljevanju ruleta), kot jo snema kamera postavljena pravokotno nad njo.

- Kamera, ki uporablja PAL video standard. To povzroči, da so sode in lihe vrstice zamaknjene, ko se ruleta vrti.
- Močni svetlobni odboji.
- Steklen ali plastičen pokrov nad kolesom rulete lomi svetlobo in povzroča močne svetlobne odboje (tudi zrcalne), ki močno pokvarijo zajeto sliko.
- Zajeta slika se zamegli, ko se ruleta zelo hitro vrti.

Ker sem imel na voljo PAL video kamero, sem imel probleme pri zajemanju slike, ko se ruleta vrti. PAL video standard deluje tako, da najprej zajame sliko vseh lihih vrstic, nato zajame sliko še vseh sodih vrstic. Frekvenca zajemanja slik, pri PAL video standardu je 25 slik na sekundo, v resnici pa se zajema 50 slik na sekundo, ker se zajamejo sode in lihe vrstice za vsako zajeto sliko posebej. PAL video standard uporablja tak način delovanja zato, da slika, npr. na televizorju, ne utripa med zaporednimi slikami, ki jih zajema kamera. S tem PAL video standard dosega mehkejše prehode med zaporednimi slikami. To, kar je pri gledanju filma zaželjeno, povzroča pri sistemu za branje številke, na katero je padla kroglica na ruleti, velike probleme, kajti slika ni stabilna. Ko se ruleta vrti in slika ni stabilna, so sode

in lihe vrstice slike, zajete po PAL video standardu, zamaknjene, in slika je zelo popačena, kar je razvidno iz leve slike 3.3. Iz take slike je težko karkoli razbrati, še posebej, če se ruleta vrti zelo hitro in je zamik vrstic zelo velik. Če bi bili na sliki horizontalni premiki scene, bi lahko omenjeni problem rešil tako, da bi vrstice zamikal v nasprotni smeri horizontalnega premikanja scene, ob predpostavki, da bi vedel, kakšna je hitrost premikanja scene in kakšni so intervali zajemanja slike. To v tem primeru ni možno, saj je vrtenje rulete krožno. Ta problem sem rešil tako, da sem lihe vrstice zajete slike podvajal na mesta sodih vrstic. Rezultat je prikazan na levi sliki 3.3. Ta rešitev se je v praksi izkazala kot sorazmerno dobra rešitev, kar je tudi razvidno iz slike 3.3.



Slika 3.3: Levo je slika rulete, zajeta po PAL video standardu (s PAL video kamero), medtem ko se ruleta vrti in desno je ista slika, po podvajanju lihih vrstic leve slike.

Velike težave povzroča tudi steklen ali plastičen varovalni pokrov nad ruleto. Ker imamo opravka s problemom računalniškega vida, kjer nastopa veliko svetlobe, ta pokrov povzroča ogromno svetlobnih odbojev, ker se svetloba na njem lomi. Ti odboji so vidni tudi na slikah 3.2 in 3.3. Svetlobni odboji od pokrova rulete so v veliki večini primerov zrcalni, tako da je zajeta slika na mestih teh odbojev večinoma nerazpoznavna. Največji problem nastane, ko imamo tak zrcalni odboj na številčnici rulete. Če se v takem primeru nahaja kroglica na mestu zrcalnega odboja, na sliki ni vidna in je ne moremo detektirati. Dokaj dobra rešitev tega problema je, da se pusti kolo rulete, da se počasi vrti in se ne ustavi, ko je kroglica padla na številko. Ker so zrcalni odboji od pokrova statični, se v primeru, ko se ruleta vrti, lahko detektira kroglico na drugem mestu rulete. Tako se izognemo problemu zrcalnih odbojev svetlobe od pokrova rulete.
Svetloba se ne odbija samo od pokrova rulete. Tudi številčnica igralniške rulete je narejena iz svetlečih materialov, ki povzročajo ogromno svetlobnih odbojev. Ker so

ti odboji večinoma le na delu številčnice, se jim izognemo na enak način kot odbojem od pokrova rulete.

Poglavje 4

Implementacija sistema

Sistem za detektiranje številke pri igralniški ruleti, na katero ja padla kroglica, ki ga bomo opisali v pričujočem delu, se deli na dva podsistema:

1. Kalibracijski podsistem.
2. Podsistem za detektiranje številke, na katero je padla kroglica.

Pri implementaciji sistema so bile upoštevane določene predpostavke opisanega sistema. Predpostavke sistema so naslednje:

- Kamera mora biti postavljena centralno nad ruleto.
- Optična os kamere mora biti pravokotna glede na ruleto.
- Kolo rulete (roulette wheel) se mora raztezati čez celotno zajeto sliko.
- Slika, ki jo zajema kamera, mora biti dovolj velika.
- Sistem mora delovati v realnem času.

Namen kalibracijskega podsistema je v tem, da se posreduje drugemu podsistemu, podsistemu za detektiranje številke, na katero je padla kroglica, vse znane informacije o ruleti. Ta del je tipičen za uporabo računalniškega vida v industrijskih problemih, ker je pri tej vrsti problemov veliko informacije, ki pomaga pri reševanju tega problema, že vnaprej znane. Tako lahko ta informacija pripomore k boljši rešitvi problema. Same rešitve industrijskih problemov v računalniškem vidu so tako zelo specialne. Kalibracijski podsistem se izvede enkrat, to je takrat, kadar se igralniška ruleta postavi na novo ali pa kadar želimo spremeniti parametre, ki se nastavljajo s tem podsistemom oz. kadar se parametri rulete spremenijo. Kalibracijskemu podsistemu se poda informacijo o velikosti žogice, o poziciji številčnice

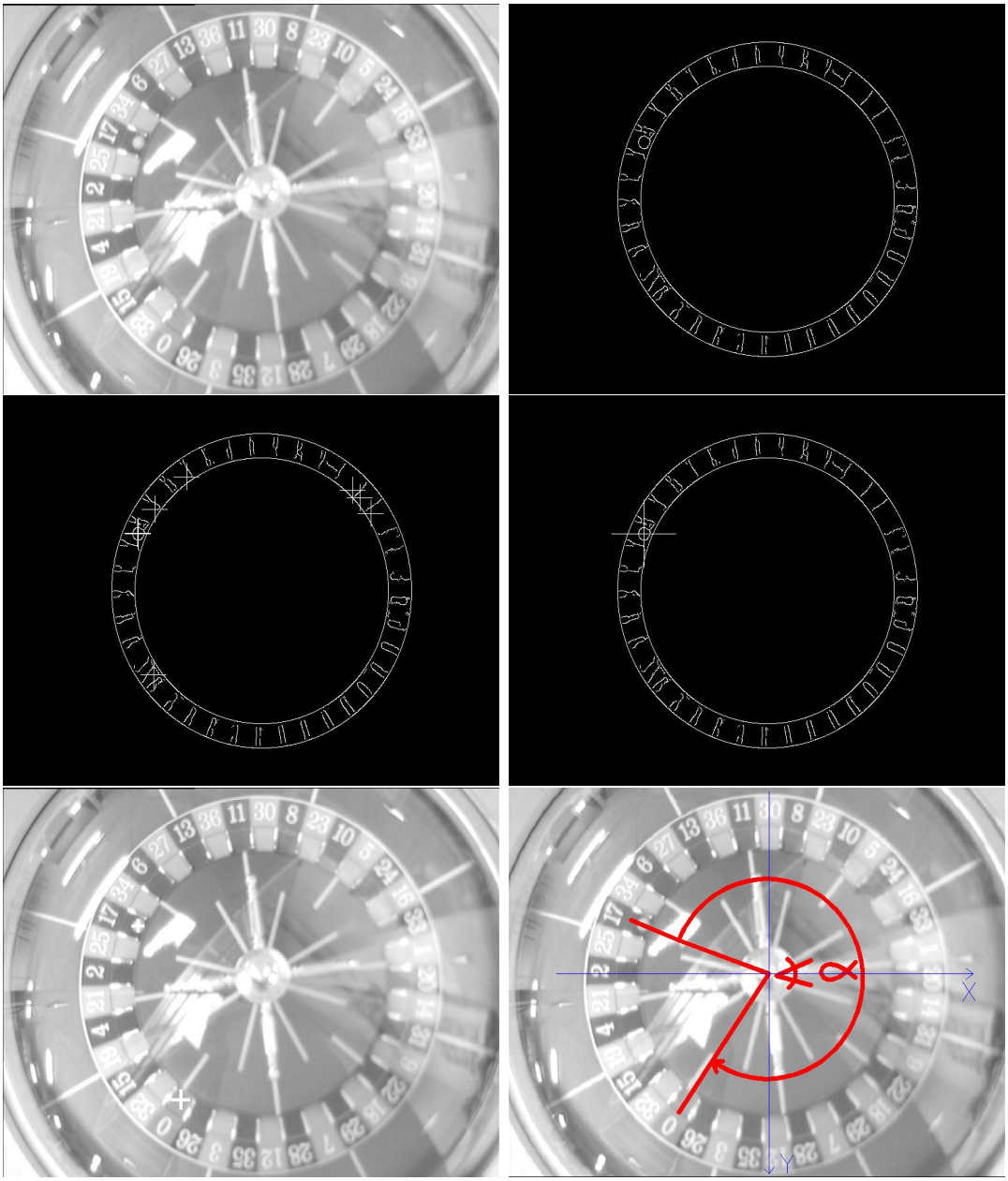
in pozicijah, na katerih se nahaja kroglica, ko pade na številko, posredujejo se tudi parametri za detekcijo robov (Cannyjev detektor robov) in detekcijo krožnice (Houghov transform) itd. Ti podatki se potem upoštevajo pri sistemu za detektiranje, na katero številko je padla kroglica, sam sistem se tako poenostavi in pridobi na hitrosti ter natančnosti. Najpomembnejše pravilo, ki sem se ga držal pri razvoju tega sistema, je bilo poskušati čimbolj poenostaviti algoritme, saj se s tem pripomore h hitrosti detekcije, kar pripomore k delovanju sistema v realnem času.
Drugi podsistem, podsistem za detektiranje številke, na katero ja padla kroglica, uporabi vse podatke, ki mu jih je posredoval kalibracijski podsistem, in poskuša ugotovoti, kam je padla kroglica.
V nadaljevanju sledi opis sistema za detektiranje, na katero številko je padla kroglica pri igralniški ruleti. Za opis kalibracijskega podsistema glej dodatek A.
Podsistem za ugotavljanje, na katero številko je padla kroglica, je sestavljen iz naslednjih korakov (glej slike 4):

1. Izvede se detekcijo robov po območju številčnice (Cannyjev detektor robov).
2. Poišče se možne pozicije kroglice (z vzorcem krožnice).
3. Ugotovi se, ali je kroglica padla, in določi se njena točna pozicija (Houghov algoritem).
4. Poišče se pozicija ničle.
5. Poišče se kot med pozicijo kroglice in ničlo.
6. Na podlagi kota med pozicijo kroglice in ničlo se določi, katera številka je padla.

4.1 Gaussovo glajenje

Slika, ki jo zajamemo s kamero, vsebuje tudi šum, ki nas moti pri nadaljnji obdelavi slike, zato ga poskušamo odstraniti iz slike.
Poznamo več vrst šuma na intenzitetnih slikah npr. Gaussov šum, sol in poper šum (impulzivni šum) itd. Šum je lahko aditiven ali pa multiplikativen. V pomanjkanju informacije o šumu in v večini primerov predpostavljamo, da je šum bel, Gaussov in aditiven. Če je šum aditiven velja, da je naključen signal $n(i, j)$ dodan pravim vrednostim slikovnih elementov:

$$I^{'}(i, j) = I(i, j) + n(i, j) \tag{4.1}$$



Slika 4.1: Od leve proti desni so predstavljeni koraki algoritma: Slika zajeta iz kamere, ki predstavlja vhod v algoritem, slika, ki je izhod iz detekcije robov, pozicije (narisane kot majhni križci), na katerih je najverjetneje kroglica, izbrana točna pozicija kroglice (velik križec), pozicija kroglice na ruleti in detektirana pozicija ničle ter kot med pozicijo kroglice in ničlo, s pomočjo katerega določimo katera številka je padla.

Koliko šuma je na sliki v primerjavi s signalom, nam pove signal to noise ratio (SNR). Če je šum Gaussov, potem je $n(i,j)$ naključna spremenljivka porazdeljena po Gaussovi porazdelitvi s fiksno standardno deviacijo in je dodana slikovnim elementom I(i,j), njene vrednosti so popolnoma neodvisne druga od druge.
Zdaj vemo, kaj je šum in kakšen je Gaussov šum. Preostane nam, da še povemo, kako Gaussov šum odpraviti s slike. To naredimo tako, da izračunamo konvolucijo vsakega slikovnega elementa na sliki z 2-D matriko, ki predstavlja vzorčene vzorce realnega 2-D Gaussovega jedra z določenim σ. Odpravljanje Gaussovega šuma s slike po tej metodi zahteva kvadratičen čas. K sreči je Gaussovo jedro separabilno (glej [1] str. 58) in je potreben čas za filtriranje le linearen. Tako dobimo isti rezultat, če namesto konvolucije z 2-D Gaussovim jedrom z določenim σ, delamo konvolucijo najprej po vrsticah, nato pa na novi sliki, ki smo jo dobili, delamo konvolucijo še po stolpcih z 1-D Gaussovim jedrom z enakim σ.
Kako pa določimo velikost maske W? Za pridobitev diskretne Gaussove maske je potrebno vzorčiti zvezno Gaussovo porazdelitev, ki je določena s σ. Diskretni Gaussov filter mora biti take širine W, da pokriva večino območja pod Gaussovo krivuljo. Zadostna izbira širine diskretnega Gaussovega filtra je naslednja:

$$W = 5\sigma \tag{4.2}$$

Z vzorčenjem Gaussove krivulje dobimo realna števila. Vendar pa lahko spremenimo dobljeni filter v takega, ki vsebuje samo cela števila, kar pripomore k zmanjšanju časovne kompleksnosti. To naredimo tako, da normaliziramo vrednosti v filtru na tak način, da je najmanjša vrednost v filtru 1. Nato zaokrožimo rezultat in delimo z vsoto vseh vrednosti v filtru.
Slabe strani Gaussovega filtra so predvsem zameglitev (blur) in nepopolno odstranjevanje šuma sol in poper. Dobra stran Gaussovega filtra je predvsem hitrost, ki je posledica separabilnosti in da Fourierjev transform Gaussove krivulje nima sekundarnih valov.

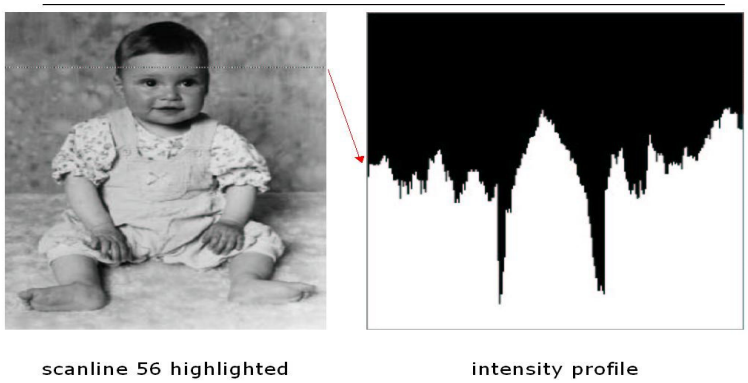
4.2 Računanje gradientov na intenzitetnih slikah

$$[1 \quad 0 \quad -1] \qquad\qquad [1 \quad 0 \quad -1]^T \tag{4.3}$$

Pri naši implementaciji sistema je potrebno izračunati odvoda v vseh točkah intenzitetne slike v smeri $\mathbf{x}$ in v smeri $\mathbf{y}$. Kako to narediti? $\mathbf{J_x}$ predstavlja odvod v smeri $\mathbf{x}$ in $\mathbf{J_y}$ predstavlja odvod v smeri $\mathbf{y}$. Želimo torej izračunati odvode intenzitetne slike $\mathbf{I}$ v vseh slikovnih elementih $p(i,j)$, tako da lahko to zapišemo kot $\Delta\mathbf{I} = [\frac{\rho\mathbf{I}}{\rho x}, \frac{\rho\mathbf{I}}{\rho y}]^T$. Odvod $\mathbf{J_x}$ izračunamo tako, da izračunamo konvolucijo vrstic slike z masko 4.3. Analogno temu, izračunamo odvod $\mathbf{J_y}$ tako, da izračunamo konvolucijo

stolpcev slike s transponirano masko 4.3 (za dokaz in nadaljnjo razlago glej [1], appendix A.2, str. 313).

4.3 Detekcija robov in Cannyjev detektor robov



Slika 4.2: Skenirana označena vrstica (levo) in intezitetna variacija te vrstice (desno).

Robne točke so preprosto slikovni elementi, na katerih oz. okoli katerih slikovne vrednosti močno (ostro) varirajo (glej sliko 4.3). Ostre variacije slikovnih vrednosti niso samo posledica dejanskih robov predmetov na sliki, ampak so tudi posledica šuma, ki nastane pri zajemanju slike iz kamere. Zato je naloga detekcije robov poiskati na sliki robove, ki niso posledica šuma, ampak predmetov na sliki. Dober detektor robov mora izničiti večino variacij, ki so posledica šuma.
V računalniškem vidu je detekcija robov tipično sestavljena iz treh korakov:

1. Odpravljanje šuma iz slike (Noise Smoothing)
   Odpravi čimveč šuma iz slike, ne da bi pokvaril resnične robove. Če nimamo podatkov o šumu, predpostavljamo, da je šum Gaussov.

2. Poudarjanje robov (Edge Enhancement)
   Poišči filter, ki ima velik odziv na robnih točkah in majhen odziv drugod. Tako so lahko robovi locirani kot lokalni maksimum odziva na filter.

3. Lokalizacija robov (Edge Localization)
   Ugotovi, kateri lokalni maksimumi so resnični robovi in kateri so posledica šuma na sliki. Lokalizacija je sestavljena iz dveh korakov:

   - Tanjšanje robov na debelino enega slikovnega elementa (nonmaximum suppression).

- Upragovljenje (tresholding):
  Definira se minimalna vrednost odziva na filter, da lahko proglasimo lokalni maksimum za robno točko.

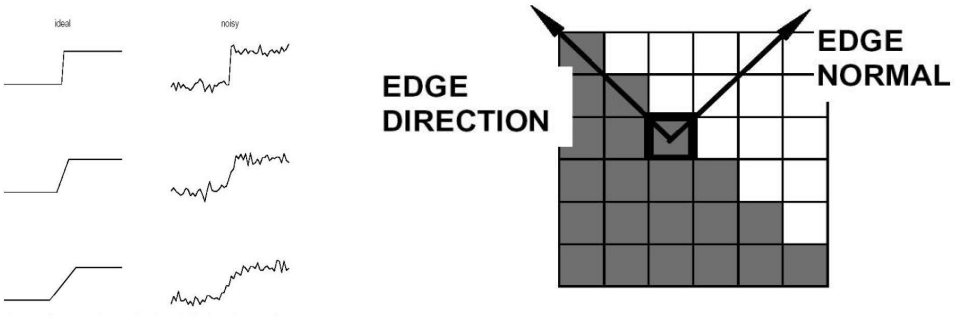
Obstaja več vrst detektorjev robov. Nekateri izmed njih so:

- Cannyjev detektor robov
  V praksi se ga največ uporablja. Možno ga je matematično formalizirati in je v matematičnem smislu optimalen. Je relativno manj občutljiv na šum kot ostali detektorji robov.

- Robertsonov detektor robov
  Je zelo občutljiv na šum in je relativno enostaven za implementacijo. Njegova glavna prednost je hitrost.

- Sobelov detektor robov
  Je bolj občutljiv na šum kot Cannyjev detektor in hkrati manj občutljiv kot Robertsonov detektor. Razlika med Robertsonovim in Sobelovim detektorjem robov je samo v filtru za poudarjanje robov (edge enhancing filter). Sobelov detektor upošteva širšo okolico slikovnega elementa, ko išče odziv filtra na slikovni element. Zato pride tudi manj do izraza šum na sliki, kot pri Robertsonovem detektorju robov.

4.4 Cannyjev detektor robov



Slika 4.3: Stopničasti rob (step edge), levo, in normala na rob, desno.

Cannyjev detektor robov je eden najpogosteje uporabljanih detektorjev robov. Je relativno malo občutljiv na šum in ima v ozadju dobro matematično podlago. Za

formalizacijo problema detektiranja robov glej [1]. Pri detekciji robov s Cannyjevim detektorjem robov in pri naši implementaciji se uporabljata normala na rob in stopničast model roba, ki sta prikazana na sliki 4.3.

4.4.1 Implementacija Cannyjevega detektorja robov

Sedaj bomo opisali podalgoritme, ki sestavljajo Cannyjev detektor robov in so bili že nakazani v opisu problema detektiranja robov na intenzitetnih slikah. Cannyjev algoritem detektiranja robov je sestavljen iz naslednjih podalgoritmov:

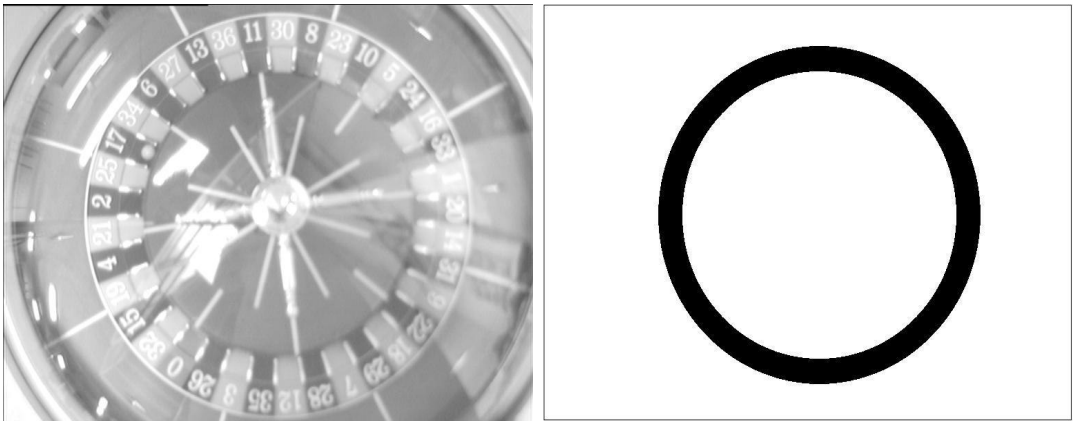
Algoritem CANNY_EDGE_DETECTOR

Dana je slika **I**. Izvedi naslednje korake:

1. Izvedi CANNY_ENHANCER na sliki **I**.
2. Izvedi NONMAX_SUPPRESSION na sliki, ki jo vrne CANNY_ENHANCER.
3. Izvedi HYSTERESIS_THRESH na sliki, ki jo vrne NONMAX_SUPPRESSION.

Ker je moj problem detektiranja številke, na katero je padla kroglica, zelo specialen in imamo že poznane nekatere podatke, npr. znano je območje, na katerem nas zanima detekcija robov, so vsi podalgoritmi detektiranja robov prilagojeni temu problemu. S tem se zmanjša čas, ki je potreben za detekcijo robov, kar pripomore k uresničevanju cilja, da sistem deluje v realnem času, istočasno se zmanjša tudi verjetnost napake.
Ob prvem zagonu sistema za detektiranje številke, na katero je padla kroglica, se inicializirajo določene strukture, ki so podane v obliki matrik. Te strukture se izračunajo samo enkrat in se v nadaljnjih zagonih sistema ne računajo več. V bistvu so to slike, ki so sestavljene samo iz dveh čnobelih barv, bele (vrednost slikovnega elementa 255) in črne (vrednost slikovnega elementa 0). Te slike se izračunajo na podlagi podatkov, ki so izhod iz kalibracijskega podsistema in so shranjeni v datoteki 'Settings.cfg'. Ena izmed takih slik je tudi slika, ki določa območje, na katerem se bo izvedla detekcija robov (glej sliko 4.4). Robovi se računajo samo na območju, kjer ima ta slika črne slikovne elemente (vrednost 0). To nam zagotavlja, da se časovna kompleksnost zmanjša, saj ni potrebno nikakršno računanje, temveč le primerjanje slikovnih elementov, če je vrednost slikovnega elementa 0.

V prvem koraku detekcije robov se izvede Gaussovo glajenje, da se znebimo šuma, za katerega predpostavljamo, da je bel, Gaussov in aditiven. Nato se izračunata gradienta J_x in J_y za vsak slikovni element zglajene slike. Ko sta gradienta izračunana, se izračuna še moč roba (edge strength).
Algoritem je naslednji:



Slika 4.4: Leva slika prikazuje zajeto sliko, ki je vhod v detekcijo robov. Desna slika prikazuje območje označeno s črnimi slikovnimi elementi, kjer se izvede detekcija robov na levi sliki.

Algoritem CANNY_ENHANCER

Vhod je intenzitetna slika **I**, ki je pokvarjena s šumom, za katerega predpostavljamo, da je bel, Gaussov in aditiven. **G** je Gaussov filter s centrom v ničli in standardno deviacijo σ:

1. Izvedi Gaussovo glajenje na sliki **I** (glej razdelek 4.1). Dobimo novo sliko **J**:

$$\mathbf{J} = \mathbf{I} * \mathbf{G} \tag{4.4}$$

2. Za vsak slikovni element (i,j) na sliki **J** izvedi naslednje korake:

   (a) Preveri, če ima istoležni slikovni element na sliki, ki označuje območje detektiranja robov (slika 4.4), vrednost 0. Če to velja nadaljuj z naslednjim korakom (b), sicer preidi na naslednji slikovni element.

   (b) Izračunaj komponente gradienta J_x in J_y (glej 4.2) in jih shrani v matriki $\mathbf{J_x}$ in $\mathbf{J_y}$.

   (c) Oceni moč roba (edge strength), ki je definirana kot:

$$e_s(i,j) = \sqrt{J_x^2(i,j) + J_y^2(i,j)} \tag{4.5}$$

   in jo shrani v matriko E_s.

Izhod sta matriki gradientov za vsako točko v x smeri, J_x, in v y smeri, J_y, ter močnostna slika (strength image) E_s, ki vsebuje vrednosti moči robne točke (amplituda odziva na filter), $e_s(i,j)$. Parameter σ nastavimo v odvisnosti od tega, koliko je slika pokvarjena s šumom. S σ določamo tudi trgovanje med detekcijo in lokalizacijo robov.

Kot izhod iz algoritma CANNY_ENHANCER dobimo močnostno sliko, E_s, na kateri lokalni maksimumi, ki predstavljajo rob, niso izraziti, temveč so široki (več lokalnih maksimumov okoli pravega maksimuma). Široki so zato, ker so se okoli pravih robov (okoli lokalnega maksimuma odziva filtra) detektirali tudi robovi, ki so posledica šuma. V naslednjem koraku moramo te robove stanjšati na debelino enega slikovnega elementa. To naredi algoritem NONMAX_SUPPRESSION:

Algoritem NONMAX_SUPPRESSION

Vhod v algoritem NONMAX_SUPPRESSION je izhod iz algoritma CANNY_ENHANCER, močnostna slika, E_s, ter matriki gradientov na robove, J_x in J_y. V algoritmu nastopajo konstante NOT_EDGE (vrednost 0) in POSIBBLE_EDGE (vrednost 128).

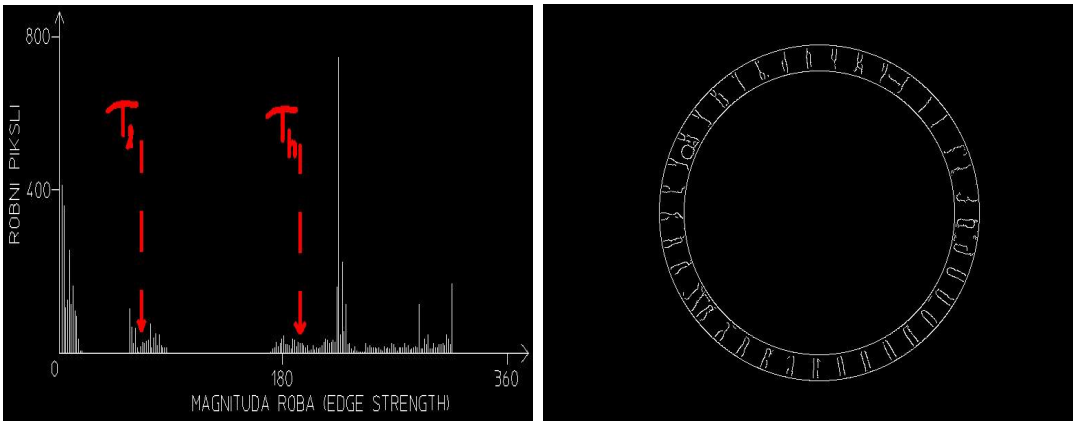
1. Za vsak slikovni element močnostne slike $E_s(i,j)$ izvedi naslednje korake:
2. Oceni orientacijo normale na rob v tem slikovnem elementu in sicer tako:

$$e_o = \arctan \frac{J_y(i,j)}{J_x(i,j)} \tag{4.6}$$

3. Poišči smer $d_k \in \{0^o, 45^o, 90^o, 135^o\}$, ki najbolje aproksimira izračunano smer normale na rob e_o (glej sliko 4.3).
4. Če je $E_s(i,j)$ manjše od vsaj enega od dveh sosedov tega slikovnega elementa vzdolž smeri normale na rob d_k, potem priredi $I_N(i,j) = NOT\_EDGE$ (zatri (suppress)), sicer priredi $I_N(i,j) = POSIBBLE\_EDGE$ in povečaj ustrezno polje v histogramu moči robov $\mathbf{H_s}$.

Izhod je slika $I_N(i,j)$, ki vsebuje stanjšane robne točke vzdolž normale na rob in histogram moči robov (edge strength histogram), $\mathbf{H_s}$ (glej sliko 4.5), ki pove, koliko robnih točk ima isto moč roba.

Slika $\mathbf{I_N}$, ki je izhod iz NONMAX_SUPPRESSION, še vedno vsebuje robove, ki so posledica šuma. Če bi se poskušali znebiti teh šumnih robnih točk tako, da bi proglasili za robne točke le točke, ki so nad nekim pragom, bi naleteli na dva problema:



Slika 4.5: Leva slika prikazuje histogram moči robov, ki se uporablja za določitev spodnjega (τ_l) in zgornjega (τ_h) praga, ki sta vhod v algoritem HYSTERESIS_TRESH. Desna slika je izhod iz detekcije robov in prikazuje robove detektirane na levi sliki 4.4 po določenem območju.

- Če postavimo nizek prag, bomo poleg pravih robnih točk sprejeli tudi točke, ki so posledica šuma.
- Če postavimo visok prag, bomo dobili fragmentirane robove (vzdolž roba ne bomo sprejeli šibkih pravih robnih točk, ki predstavljajo resničen rob).

Ta problem rešimo z adaptivnim upragovljenjem (hysteresis thresholding):

Algoritem HYSTERESIS_TRESH

Vhod je slika $\mathbf{I_N}$, ki je izhod iz NONMAX_SUPPRESSION, močnostna slika $\mathbf{E_s}$ in pragova τ_{lp} in τ_{hp}, ki sta izražena v odstotkih in za katera mora veljati $\tau_{lp} < \tau_{hp}$. τ_{hp} predstavlja odstotek robov, ki imajo manjšo moč kot je zgornji prag τ_h, τ_{lp} pa predstavlja odstotek praga τ_h in pomeni spodnjo mejo praga, do katerega sledimo robu (glej sliko 4.5).
Za vsako točko (i,j) iz slike $\mathbf{E_s}$ delaj:

1. Izračunaj praga τ_h in τ_l s pomočjo histograma moči robov $\mathbf{H_s}$ in procentov τ_{lp} ter τ_{hp} (glej sliko 4.5).
2. Poišči naslednjo neobiskano robno točko $I_N(i,j)$ (za tako točko velja $I_N(i,j) = POSSIBLE\_EDGE$), tako da velja $E_s(i,j) > \tau_h$
3. Začni v točki $E_s(i,j)$ in sledi verigi povezanih lokalnih maksimumov dokler velja $E_s(i,j) > \tau_l$. Vse obiskane točke si zapomni (priredi vrednost obiskanih točk na sliki $\mathbf{I_N}$ tako: $I_N(i,j) = EDGE$).

Izhod je slika $\mathbf{I_N}$, na kateri imajo resnične robne točke vrednost 255 (vrednost EDGE), točke ki niso robovi pa vrednost 0 (NOT_EDGE).

Algoritem HYSTERESIS_TRESH je nekoliko prirejen, saj ne potrebujemo opisov robov (edge descriptors), dovolj je že, da dobimo sliko, ki vsebuje resnične robne točke vhodne slike (to je slika $\mathbf{I_N}$). Tudi sledenje robu je implementirano tako, da se sledi vsem sosedom trenutne robne točke, dokler velja, da je $E_s(i,j) > \tau_l$ in se ne gleda normale na rob.
Na kratko povzemimo potek detekcije robov. Vhod v detekcijo robov predstavlja intenzitetna slika, ki je bila zajeta s kamero in je prikazana na levi sliki 4.5. Detekcija robov se izvede samo na območju, ki ga prikazuje desna slika 4.4. Ta slika se določi med kalibracijo sistema. Kot rezultat dobimo desno sliko 4.5.

4.5 Houghova transformacija

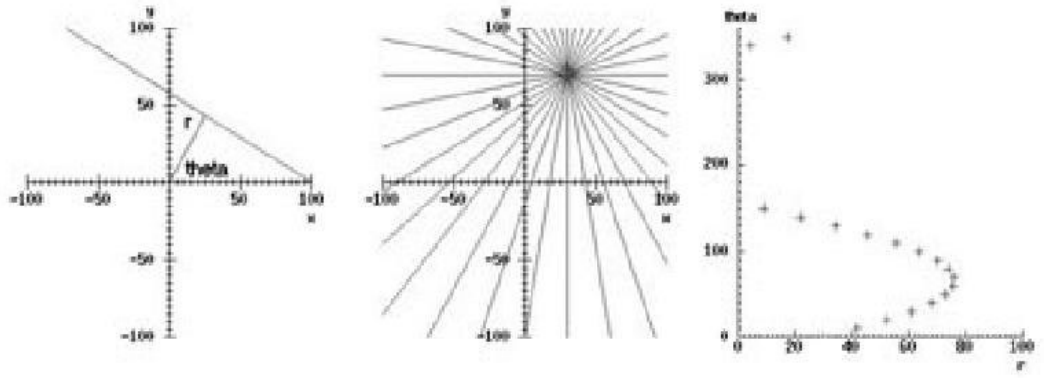
Houghova transformacija se uporablja za detektiranje vseh vrst krivulj, ki jih lahko zapišemo v parametrični obliki. Vhod v algoritem Houghove transformacije predstavlja slika robnih točk, ki jo dobimo kot izhod iz detektorja robov. Glavna ideja Houghove transformacije je preslikati težek problem detektiranja vzorca na sliki v enostaven problem iskanja maksimuma v parametričnem prostoru dane krivulje. Zaradi enostavnejšega razumevanja glavne ideje Houghove transformacije si poglejmo primer detekcije premice v polarni obliki:

$$\rho = x\cos\Theta + y\sin\Theta \tag{4.7}$$

Parameter ρ predstavlja razdaljo med premico in izhodiščem, parameter Θ pa predstavlja orientacijo premice (glej sliko 4.6).
Osnovna koraka Houghove transformacije za detekcijo premice sta:

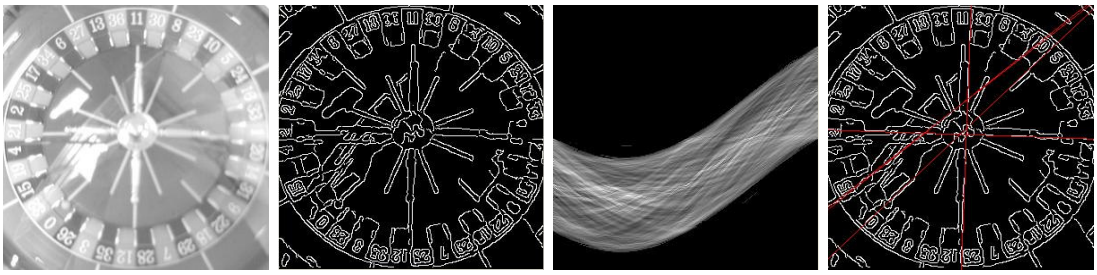
1. Pretvori problem detekcije premice v problem iskanja presečišča premic:
   Parametra ρ in Θ določata parametrični prostor premice in vsaka premica, definirana kot 4.7, je enolično predstavljena s parom (ρ, Θ) v parametričnem prostoru premice. Ravno tako vsaki točki na sliki ustreza sinusoidna krivulja v parametričnem prostoru, ki predstavlja vse možne premice skozi to točko (glej sliko 4.7). Točke, ki ležijo na isti premici, bodo v parametričnem prostoru predstavljene s sinusoidami, ki imajo skupno presečišče (ρ_1, Θ_1). To presečišče predstavlja premico, ki gre skozi vse te točke na sliki.

2. Pretvori problem iskanja presečišča premice v problem iskanja maksimuma (vrha) v parametričnem prostoru:
   Parametra ρ in Θ je potrebno diskretizirati na sprejemljivo resolucijo. Matrika $\mathrm{A} = \rho_\mathbf{d} \times \mathbf{\Theta_d}$ predstavlja akumulacijsko polje, ki predstavlja diskretiziran



Slika 4.6: Od leve proti desni: parametra (polmer r in kot Θ), ki določata premico v parametrični obliki, šop premic skozi robni element in predstavitev šopa premic v parametričnem prostoru.

parametrični prostor. Razdelitev parametričnega prostora na elemente akumulacijskega polja imenujemo kvantizacija. Vsak element tega polja, $A(\rho, \Theta)$, je na začetku inicializiran na 0. Za vsako točko iz slike povečaj vse števce $A(\rho_d, \Theta_d)$, ki jih določa sinusoida, ki jo ta točka definira v parametričnem prostoru. Torej je premica, ki gre skozi večino točk na sliki predstavljena kot vrh (peak) v parametričnem prostoru. Ta vrh se pojavi v presečišču, kjer se največ sinusoid, ki predstavljajo vse možne premice skozi točko na sliki, seka. Ker je akumulacijsko polje v primeru premice dvorazsežno, ga lahko prikažemo kot sivinsko sliko, s katere so lepo razvidni vrhovi akumulacijskega polja (glej sliko 4.7).



Slika 4.7: Od leve proti desni: vhodna slika, najdeni robovi, akumulacijsko polje in izbrane premice na sliki robov.

Razlog, da smo izbrali parametrično predstavitev premice (enačba 4.7) namesto kartezične enačbe $y = mx + n$ je v tem, da parametra m in n zavzameta neskončno

vrednosti na intervalu $[-\infty, +\infty]$ in ne moremo predstaviti vseh teh vrednosti z akumulacijskim poljem. V nasprotju s tem, sta parametra ρ in Θ končna in lahko predstavimo z njima vse premice na sliki ($\rho \in \left[0, \sqrt{M^2 + N^2}\right]$, $\Theta \in [0, \pi]$). Problem predstavlja tudi kvantizacija parametrov ρ in Θ, saj z njo izgubimo na natančnosti parametrov. S kvantizacijo parametrov določamo natančnost, več diskretnih vrednosti kot ima parameter, bolj natančna je Houghova transformacija. Na robni sliki je lahko predstavljenih tudi več premic, ki jih preprosto detektiramo tako, da poiščemo vse lokalne maksimume (vrhove) v akumulacijskem polju. Houghov algoritem je glasovalni algoritem, ker vsaka točka na robni sliki glasuje za svojo(e) premico(e). Šum na robni sliki in omejena natančnost detektorja robov (predvsem zaradi pikselizacije (največja resolucija je en slikovni element)) prispeva k temu, da vrhovi v akumulacijskem polju niso izraziti, ampak imajo v svoji okolici več manjših vrhov. Rešitev temu je upragovljenje maksimumov, to pomeni, da določimo najmanjši prag τ_{max}, katerega morajo vrhovi v akumulacijskem polju, ki jih proglasimo za maksimume, preseči.
Podajmo sedaj še Houghov algoritem za detekcijo premice, ki je bil delno opisan zgoraj:

Algoritem HOUGH_LINES

Vhod je slika robov, $E(i, j)$, ki je izhod iz detektorja robov. Seznama $\rho_{\mathbf{d}}$ in $\theta_{\mathbf{d}}$ predstavljata diskretizirane vrednosti parametrov ρ in Θ, $\rho \in \left[0, \sqrt{M^2 + N^2}\right]$, $\Theta \in [0, \pi]$, število R, ki predstavlja število elementov seznama $\rho_{\mathbf{d}}$, in T, ki predstavlja število elementov seznama $\mathbf{\Theta_d}$.

1. Diskretiziraj parametra ρ in Θ s takimi kvantizacijskimi koraki $\delta\rho$ in $\delta\Theta$, ki zagotavljata sprejemljivo resolucijo in obvladljivo velikost.

2. Naj bo $A(R, T)$ akumulacijsko polje, ki vsebuje celoštevilske števce. Inicializiraj vse elemente akumulacijskega polja A na 0.

3. Za vsako točko $E(i, j)$, za katero velja $E(i, j) = EDGE$ (torej je robna) delaj naslednje:

   - Za $h \in [1...T]$ delaj naslednje:
     (a) Naj bo $\rho = i \cos \Theta_d(h) + j \sin \Theta_d(h)$.
     (b) Poišči indeks $k \in \rho_d$, ki najbolje aproksimira realno vrednost ρ.
     (c) Povečaj števec $A(k, h)$ za ena.

4. Poišči vse lokalne maksimume (k_p, h_p), tako da velja $A(k_p, h_p) > \tau$, kjer je τ vnaprej določen prag.

Izhod je množica parov $(\rho_d(k_p), \Theta_d(h_p))$, ki opisuje premice, detektirane na vhodni robni sliki **E**.

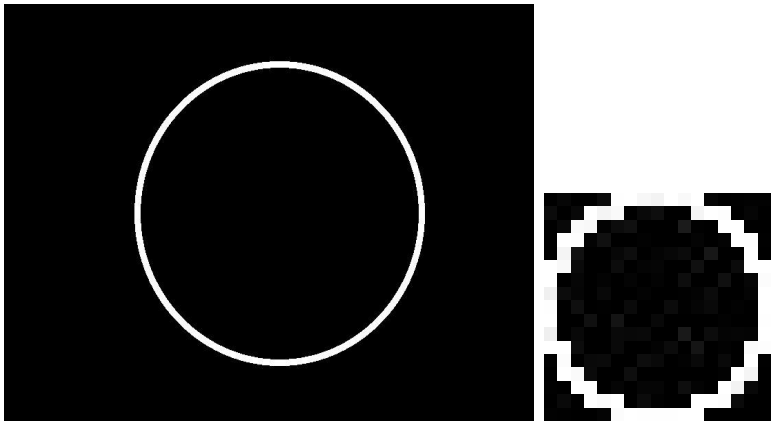
Dobre lastnosti algoritma Houghove transformacije so naslednje:

- Robustnost.
- Relativna neobčutljivost na šum na sliki.
- Neobčutljivost na delna zakrivanja oz. sence na sliki. Kljub temu, da je krivulja delno zakrita, lahko še vedno dovolj robnih točk glasuje zanjo, da je izbrana.

Kljub zgoraj navedenim dobrim lastnostim Houghove transformacije, obstajajo tudi pomembne slabe lastnosti algoritma. Najpomembnejše slabe lastnosti so:

- kvantizacija: Bolj veliko kot je akumulacijsko polje, tem boljši je približek najdene krivulje, hkrati pa to pomeni večjo časovno kompleksnost. Ravno tako velja obratno, da manjše kot je akumulacijsko polje, slabši je približek najdene krivulje, boljša pa je časovna kompleksnost algoritma.
- prag: Kako (kje) določiti prag, koliko točk mora glasovati za krivuljo, da bo izbrana.
- več parametrijske krivulje: pri teh krivuljah ima akumulacijsko polje več dimenzij, kar bistveno poveča časovno kompleksnost. Časovna kompleksnost iskanja maksimuma pri algoritmu Houghove transformacije je N^P, kjer je N število diskretnih (kvantizacijskih) vrednosti parametrov, P pa je število parametrov krivulje, ki jo iščemo.

Iz dosedaj povedanega bi lahko sklepali, da je Houghov algoritem najboljša rešitev za iskanje kroglice na sliki, ko ugotavljamo, na katero številko je padla kroglica. To je tudi res, vendar ima Houghov algoritem veliko, za naš problem pomembnih, negativnih lastnosti, da bi ga lahko samostojno uporabili za iskanje kroglice. Najpomembnejša negativna lastnost je velika časovna zahtevnost. To bi takoj privedlo do počasnega delovanja, kar ni v skladu z zahtevo, da sistem deluje v realnem času. Časovna zahtevnost iskanja maksimuma pri Houghovem transformu za iskanje krožnice pri znanem radiju R je N^2, ker iščemo vrednosti dveh parametrov, (a, b), ki predstavljata središče krožnice na sliki. N je število diskretnih vrednosti parametrov (kvantizacijske vrednosti). V našem primeru, v katerem nas zanima natančnost enega slikovnega elementa, bi ob predpostavki, da imamo kvadratno sliko širine 500 slikovnih elementov, N bil enak 500 in bi bila časovna zahtevnost samo za iskanje



Slika 4.8: Na levi strani je slika, ki določa območje, na katerem se nahaja kroglica, ko je padla na številko. Na tem območju se išče kroglico. Desna slika je slika povečanega vzorca (pattern) kroglice, s katerim iščemo najboljše kandidate (pozicije) za kroglico.

maksimuma pri Houghovi transformaciji enaka 500^2, kar je pa očitno mnogo preveč.

Zaradi počasnosti Houghove transformacije sem bil prisiljen uvesti izboljšave, ki pripomorejo h hitrejšemu delovanju, čeprav tudi malo na škodo uspešnosti detekcije kroglice. Prva izmed takih izboljšav je slika (interna struktura je matrika, inicializira se samo ob prvem zagonu sistema), prikazana na sliki 4.8 (levo). Ta slika je potrebna tudi zato, da vemo, kdaj je padla kroglica na številko. Na sliki je narisan bel krog. Slikovni elementi (beli) tega kroga predstavljajo območje, kjer se nahaja kroglica, ko je padla na številko (glej tudi sliko kalibracije A.2). Iskanje kroglice se izvede samo na slikovnih elementih, ki so beli na tej sliki (krožnica). Odgovor na vprašanje, zakaj ne bi raje računali točk kroga, na katerih bi kasneje iskali krožnico, je preprost. Tega ne delamo zato, ker bi morali uporabiti funkciji cos in sin za vsako točko na tej krožnici. Ti dve funkciji sta časovno kar precej kompleksni in časovna kompleksnost sistema bi se s tem precej povečala, kar je zopet v nasprotju z zahtevo, da mora sistem delovati v realnem času. Druga izboljšava je uvedba vzorca kroglice, prikazanega na sliki 4.8 (desno). S takim vzorcem najprej iščem minimalno vsoto kvadratov ujemanja (definirana kot 4.8) tega vzorca kroglice po območju, ki ga določa leva slika 4.8. Zapomnim si najboljših N pozicij, kjer je bila kvadratična napaka ujemanja najmanjša. Po pozicijah, ki mi jih je predlagala metoda iskanja minimalne kvadratične napake ujemanja vzorca kroglice in območja na sliki, izvedem Houghovo transformacijo za iskanje krožnice na intenzitetni sliki. S tem, ko izvedem Houghovo transformacijo po predlaganih pozicijah, zagotovim, da je vzorec na sliki res podoben krožnici, kajti vzorec kroglice se lahko dobro ujema s sliko tudi če del slike, s katerim ga primerjamo, ni podoben krožnici (dobro ujemanje na črnih

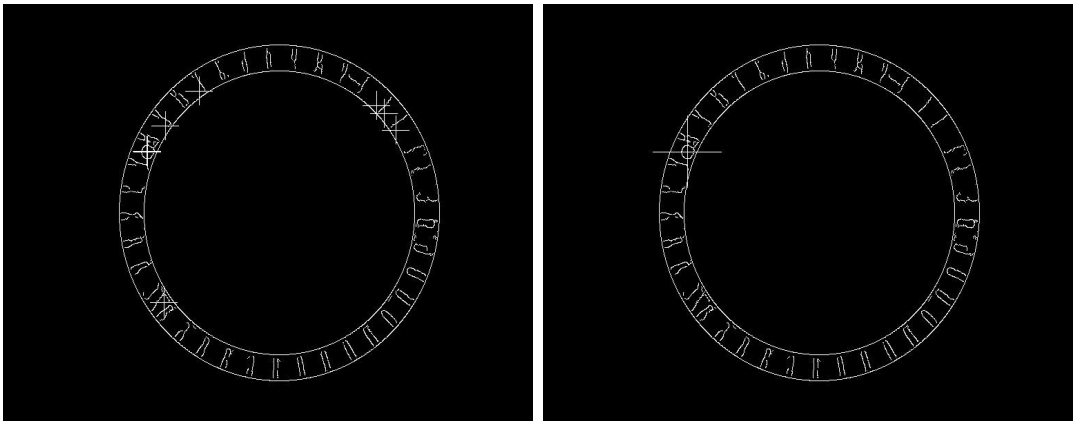
in slabo ujemanje na belih slikovnih elementih). Območje, kjer se izvede Houghova transformacija se še bistveno zmanjša in s tem se bistveno zmanjša tudi časovna kompleksnost predlaganega sistema.
Prvi korak v opisanem postopku je iskanje pozicij na sliki, ki se najbolje ujemajo z vzorcem 4.8 (desno). Testiranja so pokazala, da je bolje, kot iskati pozicijo ujemanja vzorca krožnice z določenim radijem, iskati pozicijo ujemanja vzorcev krožnice z različnimi radiji, ki malo (tipično 1 do 2 slikovna elementa) odstopajo od glavnega radija, določenega s kalibracijo (prikazan na sliki A.3). To je bolje zato, ker imamo opravka s problemom računalniškega vida, kjer je ogromno svetlobe (tudi zrcalnih odbojev). Ti svetlobni odboji tudi malo spreminjajo radij kroglice, čeprav je še vedno na robni sliki prikazana kot krožnica in temu je treba prilagoditi sistem. Še ena taka izboljšava, na katero moramo biti pozorni je, da si ne zapomnimo vseh najboljših pozicij krožnice na robni sliki, ki jih je predlagala metoda iskanja najmanjših napak ujemanja vzorca krožnice z deli robne slike. Zapomnimo si le tiste pozicije, na katerih se vzorec krožnice ujema s sliko v odstotkih vsaj za $Min\_Patt\_Match\_Perc$ odstotkov (določen s kalibracijo, glej sliko A.4). Ta izboljšava je potrebna zaradi tega, ker je algoritem Houghove transformacije zelo neobčutljiv na šum. Če bi v takem primeru imeli zelo šumno robno sliko (veliko robov, ki so posledica šuma), na kateri kroglice sploh ne bi bilo in bi algoritem iskanja najmanjše kvadratične napake ujemanja vzorca krožnice s sliko predlagal pozicije, ki se zelo slabo ujemajo z vzorcem, bi algoritem Houghove transformacije zaznal krožnico z določenim radijem, za katero bi glasovalo zelo veliko robnih (šumnih) točk. To se pri sistemu detektiranja številke, na katero je padla kroglica, zgodi zaradi svetlobnih odbojev. Z omejitvijo, da se morajo najboljše pozicije ujemanja vzorca krožnice s sliko, ujemati vsaj v $Min\_Patt\_Match\_Perc$ odstotkov robnih točk, se izognemo temu problemu, saj mora biti vzorec na sprejeti poziciji na sliki res podoben krožnici.

Algoritem iskanja N najboljših pozicij, na katerih je kvadratična napaka ujemanja vzorca krožnice z deli intenzitetne slike minimalna je definiran takole:

Algoritem FIND_BEST_N_BALL_POSITIONS

Vhod je slika robnih točk, **E**, slika, ki predstavlja območje iskanja kroglice, **S** (prikazano na sliki 4.8 (levo)), seznam vzorcev krožnice **Patt** (vzorec je prikazan na sliki 4.8 (desno)), minimalen procent ujemanja vzorca krožnice $Min\_Patt\_Match\_Perc$, in konstanta N, ki predstavlja število najboljših pozicij krožnice.

- Naj bo *worst* najslabši element seznama najboljših pozicij. Za vse točke s slike robnih točk E, za katere velja, da istoležni slikovni element na sliki območja iskanja kroglice ni črn, $S(i, j) \neq 0$, delaj:



Slika 4.9: Leva slika prikazuje najbolj perspektivne pozicije centra krožnice (prikazane kot križi na sliki), ki jih je na vhodni robni sliki 4.5 predlagal algoritem FIND_BEST_N_BALL_POSITIONS. Desna slika prikazuje pozicijo (označena kot velik križ), ki jo je izbral algoritem HOUGH_FIND_CIRCLE_PROPOSED_POS, ki uporablja Houghovo transformacijo.

(a) Za vsak vzorec krožnice $\mathbf{P} \in \mathbf{Patt}$ izvedi zanko med (b) in (d):

(b) Inicializiraj števec *sqr_err* na 0. Naj bo (X, Y) točka, ki predstavlja središče krožnice na sliki vzorca P. Izračunaj vsoto kvadratov ujemanja vzorca krožnice P s središčem v trenutni točki (i, j), ki je definirana z naslednjo formulo:

$$sqr\_err = \sum_{h=-\frac{P_h}{2}}^{+\frac{P_h}{2}} \sum_{w=-\frac{P_w}{2}}^{+\frac{P_w}{2}} (E(i+h, j+w) - P(Y+h, X+w))^2 \quad (4.8)$$

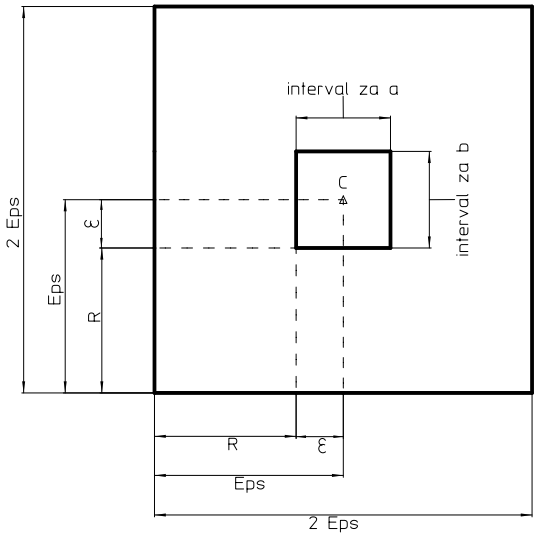
(c) Če se trenutni vzorec krožnice P ujema vsaj v $Min\_Patt\_Match\_Perc$ odstotkov s trenutno pozicijo na sliki in če je $sqr\_err < worst$ nadaljuj z (d), sicer preidi na drugi vzorec krožnice.

(d) Vpiši trenutno predlagano pozicijo krožnice (i, j) v seznam najboljših pozicij krožnice, in sicer tako:

- Če je seznam predlaganih pozicij poln, vpiši trenutno pozicijo v ta seznam na mesto, kjer je najslabši, $worst$, element in poišči nov najslabši element.
- Če seznam ni poln, vpiši element na zadnje mesto in po potrebi popravi najslabši element (če je trenutni element nov najslabši element).

Izhod je seznam **Best_Positions** dolžine N, ki vsebuje najbolj perspektivne pozicije krožnice na sliki.

Kot izhod iz algoritma FIND_BEST_N_BALL_POSITIONS dobimo seznam pozicij **Best_Positions**, ki vsebuje najboljše potencialne kandidate za center krožnice (prikazane na levi sliki 4.9, kot majhni križci). Če ta seznam ni prazen, služi kot vhod v naslednji algoritem, algoritem HOUGH_FIND_CIRCLE_PROPOSED_POS, ki s pomočjo Houghove transformacije za iskanje krožnice s poznanim radijem R, poišče krožnico v okolici ϵ okoli predlaganih pozicij **Best_Positions**, za katero glasuje največ robnih točk iz slike robnih točk **E**. Algoritem je sledeč:



Slika 4.10: Prikaz področja delovanja Houghovega algoritma za iskanje krožnice z začetnim predlaganim središčem v točki C in radijem R iz algoritma HOUGH_FIND_CIRCLE_PROPOSED_POS. Prikazano področje je izsek iz slike robnih točk. Išče se središče krožnice (a, b), ki je znotraj notranjega kvadrata širine 2ϵ.

Algoritem HOUGH_FIND_CIRCLE_PROPOSED_POS

Vhod je slika robnih točk **E**, seznam najbolj perspektivnih pozicij krožnice **BestPositions**, ki je izhod iz FIND_BEST_N_BALL_POSITIONS, radij krožnice R, ter ϵ, ki pove za koliko slikovnih elementov se lahko dejansko središče krožnice razlikuje od predlaganega središča krožnic iz seznama **BestPositions**.
Spremenljivka *BestMatch* pove število točk, ki so glasovale za trenutno najboljše središče krožnice **Center**, ki ga je predlagal Houghov algoritem.

- Inicializiraj Eps = R+ ϵ.
- Za vsako središče **C** $\in$ *BestPositions* delaj:

- Inicializiraj vse elemente akumulacijskega polja $A(i, j)$ na 0.
- Na podpodročju slike robnih točk, **E** , ki je prikazano na sliki 4.10 kot zunanji kvadrat in za katero velja $x \in [C_x - Eps, C_x + Eps]$ in $y \in [C_y - Eps, C_y + Eps]$, išči središče krožnice z radijem R in začetnim središčem v točki (a_c, b_c), za katero velja $a_c \in [C_x - \epsilon, C_x + \epsilon]$ in $b_c \in [C_y - \epsilon, C_y + \epsilon]$. Območje za (a_c, b_c) je prikazano kot notranji kvadrat na sliki 4.10. Enačba za iskanje središča krožnice (a, b) s Houghovim algoritmom pri znanem radiju R je definirana kot:

  $$b = y - \sqrt{R^2 - (x - a)^2} \tag{4.9}$$

  Za vsako robno točko (x, y) iz območja, ki ga določa zunanji kvadrat na sliki 4.10, je potrebno izračunati b s pomočjo enačbe 4.9, za vsako vrednost a iz intervala $a \in [C_x - \epsilon, C_x + \epsilon]$, skladno z definicijo Houghovega transforma. Zanimajo nas samo b, ki padejo v interval $b \in [C_y - \epsilon, C_y + \epsilon]$. Če je b iz tega intervala, se element akumulacijskega polja $A(a, b)$ poveča za 1.
- Poišči element akumulacijskega polja z indeksi (a_c, b_c), za katerega velja, da je vrednost akumulacijskega polja $A(a_c, b_c)$ največja izmed vseh vrednosti akumulacijskega polja A (za ta element je glasovalo največ točk).
- Če je vrednost akumulacijskega polja $A(a_c, b_c)$ večja od $BestMatch$, priredi $BestMatch = A(a_c, b_c)$, $Center_x = a_c$ in $Center_y = b_c$.
- Nadaljuj z naslednjo pozicijo iz seznama $BestPositions$.

Izhod iz algoritma je središče krožnice, **Center**, z danim radijem R na sliki robnih točk **E**. Izhod je tudi $BestMatch$, ki pove, koliko robnih točk je glasovalo za to središče krožnice.

Kot izhod iz algoritma HOUGH_FIND_CIRCLE_PROPOSED_POS dobimo center krožnice (prikazan kot velik križ na desni sliki 4.9), ki ga je izbral Houghov transform in število $BestMatch$, ki predstavlja število točk, ki so glasovale za to središče. Število $HoughMatch$ je prag, ki je določen s kalibracijo (glej sliko A.4) in pove, koliko robnih točk mora glasovati za določeno središče, da ga sprejmemo. Če je število točk $BestMatch$, ki so glasovale za dobljeno središče krožnice večje od števila $HoughMatch$, privzamemo, da je kroglica padla na številko in da je algoritem HOUGH_FIND_CIRCLE_PROPOSED_POS predlagal pravo središče krožnice. S tem se izognemo lokalnim vrhovom na akumulacijskem polju Houghove transformacije za iskanje krožnice, ki so posledica šuma.

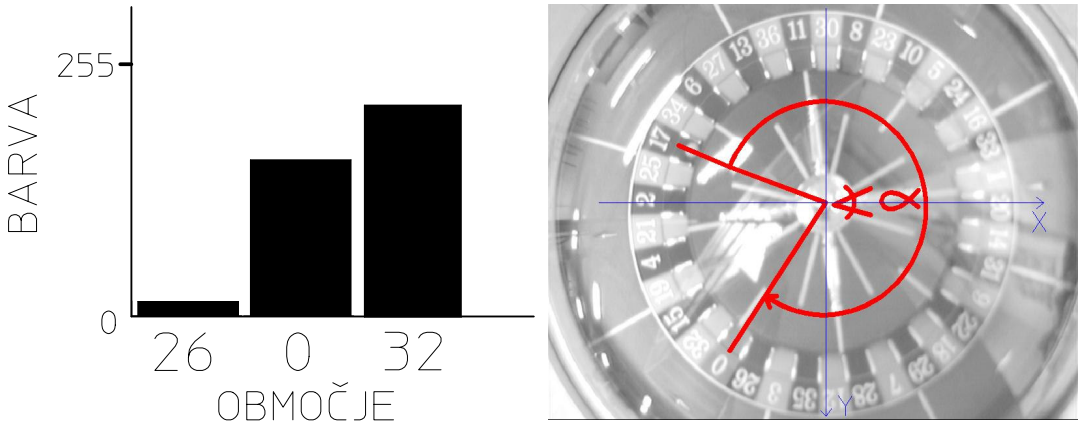
4.6 Določanje številke, na katero je padla kroglica

Sedaj vemo, kje se nahaja kroglica. Postavlja se vprašanje, kako ugotoviti, na katero številko je padla. V glavnem se ponujata dve možnosti:

1. Iz številke na številčnici.
2. S pomočjo svetlo-temnih predalčkov (sockets) ob številki, v katere pade kroglica. Ugotovi se kot med kroglico in ničlo ter se potem s pomočjo tega kota določi, katera številka je padla.

Prva možnost, da bi ugotavljal, na katero številko je padla kroglica, s pomočjo številke ob številčnici, vnese v sistem dodatne slabe lastnosti. S to metodo bi postal sistem še bolj občutljiv na šum, saj bi morala biti številka lepo vidna na sliki, da bi se jo dalo razbrati iz slike. To pomeni, da bi detektor robov moral zelo dobro izločiti šum iz slike. Ker pa imamo opravka s problemom računalniškega vida, kjer nastopa veliko svetlobe in s tem tudi šuma na sliki, bi bilo to zelo težko doseči. Verjetnost napake sistema bi se s tem bistveno povečala. Druga slaba lastnost prve možnosti je, da bi se sistem upočasnil, kar je v nasprotju s predpostavko, da naj sistem deluje v realnem času. Tranformacije, potrebne za predprocesiranje številke, ki bi jo poskušali razpoznati, so časovno potratne. Poleg tega bi več časa vzela tudi detekcija robov, saj bi jo bilo treba izvesti na večjem območju slike (po številkah na številčnici).
Ideja druge možnosti, ugotavljanje številke s pomočjo svetlo-temnih predalčkov, je sledeča. Najprej je treba ugotoviti pozicijo ničle. Predalčki, v katere pade kroglica ob številkah rulete, so izmenjaje temno-svetli (glej desno sliko 4.11). Le pri ničli so v pozitivni smeri koordinatnega sistema na sliki 4.11 po vrsti (glej graf 4.11): črn (številka 26), svetlejši (številka 0) in najsvetlejši (številka 32.). Iskanje ničle se torej spremeni v iskanje vzorca prikazanega na grafu 4.11. Ko je pozicija ničle znana, se določi kot med kroglico in ničlo ter se nato iz tabele, ki vsebuje številke in njihove kote od ničle, prebere, katera številka je padla. V nasprotju s prvo metodo, ta metoda ni toliko občutljiva na svetlobo, saj kljub svetlobnim variacijam temnejši predalčki ostajajo temnejši od svetlejših predalčkov. Tako se verjetnost napake sistema bistveno zmanjša v nasprotju s prvo metodo. Ta metoda je tudi časovno veliko manj zahtevna kot prepoznavanje številk. Tudi detekcijo robov je potrebno izvesti samo na območju, kjer so predalčki in ne na celotni številčnici. Ta metoda torej bolj pripomore k cilju delovanja sistema v realnem času, kot prva, istočasno je tudi verjetnost napake pri drugi metodi manjša kot pri prvi metodi. Problem pri drugi metodi predstavljajo edino zrcalni odboji svetlobe po predalčkih. Če se svetloba zrcalno odbija od območja, kjer se nahaja ničla, potem se lahko zgodi, da vzorca z grafa 4.11 ne bomo našli in posledično ničle ne bomo detektirali.
Pri primerjanju dobrih in slabih lastnosti obeh možnosti se izkaže, da je veliko boljša

druga možnost, ki sem jo izbral za implementacijo. Osnovni algoritem za iskanje kota med kroglico in številko (glej desno sliko 4.11) je sledeč:



Slika 4.11: Slika levo prikazuje graf vzorca, ki ga iščemo po številčnici, da dobimo ničlo. Slika desno prikazuje kot α med pozicijo kroglice in ničlo.

Algoritem FIND_BALL_TO_ZERO_ANGLE

Vhod je originalna slika **I**, položaj kroglice na sliki **BallPos** in središče številčnice rulete **Center**.

1. Izračunaj koordinate točke **BallPos** glede na koordinatni sistem, ki ima izhodišče v središču številčnice rulete **Center**, in osi paralelne koordinatnemu sistemu slike (glej desno sliko 4.11):

$$\begin{aligned} X &= BallPos_x - Center_x \\ Y &= BallPos_y - Center_y \end{aligned} \tag{4.10}$$

2. Izračunaj kot t_{Ball}, ki predstavlja kot pozicije kroglice (X, Y), v pozitivni smeri, in sicer tako:
   - Če je $X > 0$ in $Y = 0$, potem $t_{Ball} = 0$
   - Če je $X = 0$ in $Y > 0$, potem $t_{Ball} = \frac{\pi}{2}$
   - Če je $X < 0$ in $Y = 0$, potem $t_{Ball} = \pi$
   - Če je $X = 0$ in $Y < 0$, potem $t_{Ball} = \frac{3\pi}{2}$

- Sicer:

$$t_{Ball} = \begin{cases} \arctan\frac{Y}{X} & X \geq 0 \\ \arctan\frac{Y}{X} + \pi & X < 0 \end{cases} \tag{4.11}$$

3. N je število številk na številčnici. δ je kot med številkami na številčnici, izračunan po naslednji formuli:

$$\delta = \frac{N}{2\pi} \tag{4.12}$$

Socket je vektor dolžine N. Inicializiraj $t = t_{Ball}$ in $i = 0$. Dokler ni najdena ničla na številčnici oz. dokler velja $t \leq t_{Ball} + 2\pi$ (glej desno sliko 4.11) izvajaj naslednje korake:

- R je razdalja med izhodiščem in pozicijo kroglice, (x, y) predstavlja položaj znotraj predalčka (socket) ob številki, v katerega pade kroglica in je kot med pozicijo kroglice (X, Y) in njim, enak t:

$$\begin{aligned} x &= Center_x + R * \cos(t) \\ y &= Center_y + R * \sin(t) \end{aligned} \tag{4.13}$$

- Seštej slikovne elemente v okolici točke (x, y) in priredi $Socket(i) = \Sigma_Q(x, y)$, kjer je Q okolica točke (x,y)
- Če velja pogoj (glej levo sliko 4.11, iščemo vzorec, v katerem se nahaja številka 0):

$$Socket(i-2) \leq Socket(i-1) \leq Socket(i) \tag{4.14}$$

potem smo našli pozicijo ničle (pozicija $i - 1$). Izračunaj kot med ničlo in pozicijo kroglice v pozitivni smeri, kot je prikazano na desni sliki 4.11 in se ustavi:

$$\alpha = (i-1)\delta \tag{4.15}$$

- Povečaj t in i: $t = t + \delta$, $i = i + 1$.

4. Če v prejšnji zanki nismo dobili številke 0, preveri, ali je kroglica na kateri izmed zaporednih številk 26, 0 ali 32, in sicer tako:

- Če je $Socket(i-2) < Socket(i-1) < Socket(i-3)$, potem je številka 32, sicer
- Če je $Socket(i-1) > Socket(1) > Socket(2)$, potem je številka 26, sicer

- Ce je $Socket(i-1) < Socket(1)$, potem je številka 0, sicer
- Ničle nismo dobili, vrni $FALSE$

Izhod je kot v pozitivni smeri α med položajem kroglice in ničlo (glej desno sliko 4.11).

Kot izhod iz algoritma FIND_BALL_TO_ZERO_ANGLE dobimo kot α med pozicijo kroglice in ničlo (glej desno sliko 4.11). S pomočjo tega kota in tabele, ki vsebuje številke in kote v pozitivni smeri med temi številkami in številko 0, določimo, katera številka je padla. Tabela, ki vsebuje številke in njihove kote od ničle, se inicializira ob prvem zagonu sistema, ko se številke preberejo iz datoteke 'RouletteNumbers.txt'. S tem smo ugotovili, katera številka je padla in opisali celotno implementacijo sistema.

Poglavje 5

Rezultati

Za potrebe testiranja sem posnel bazo posnetkov, nad katerimi sem nato testiral sistem za detekcijo številke, na katero je padla kroglica pri igralniški ruleti. Posnel sem tudi učno oz. kalibracijsko množico posnetkov, nad katerimi sem skalibriral sistem. Upošteval sem, da mora biti učna množica neodvisna od testne množice, zato posnetki, nad katerimi sem sistem kalibriral, niso bili uporabljeni pri testiranju nad testno množico. Parametre sistema, ki sem jih določil s kalibracijo nad učno množico, sem uporabil za testiranje nad testno množico.

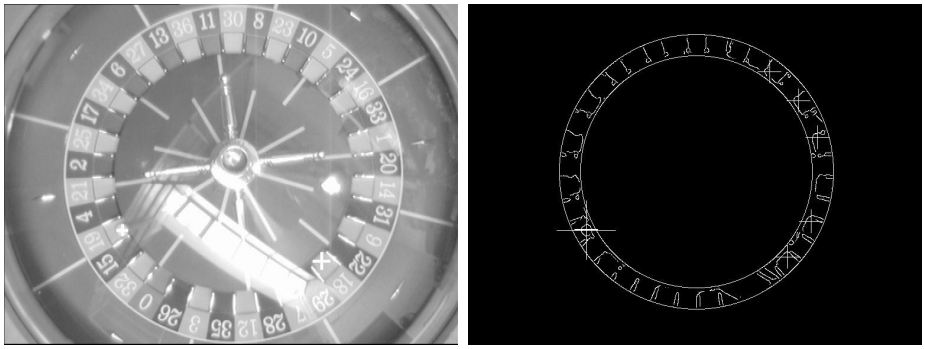
Predpostavka za vhod v algoritem za detekcijo, na katero številko je padla kroglica

Rezultati testiranja nad bazo posnetkov			
Št. posnetkov	Zadetki	Napake	Uspešnost
42	35	7	83.33%

Tabela 5.1: Rezultati testiranja nad testno množico posnetkov.

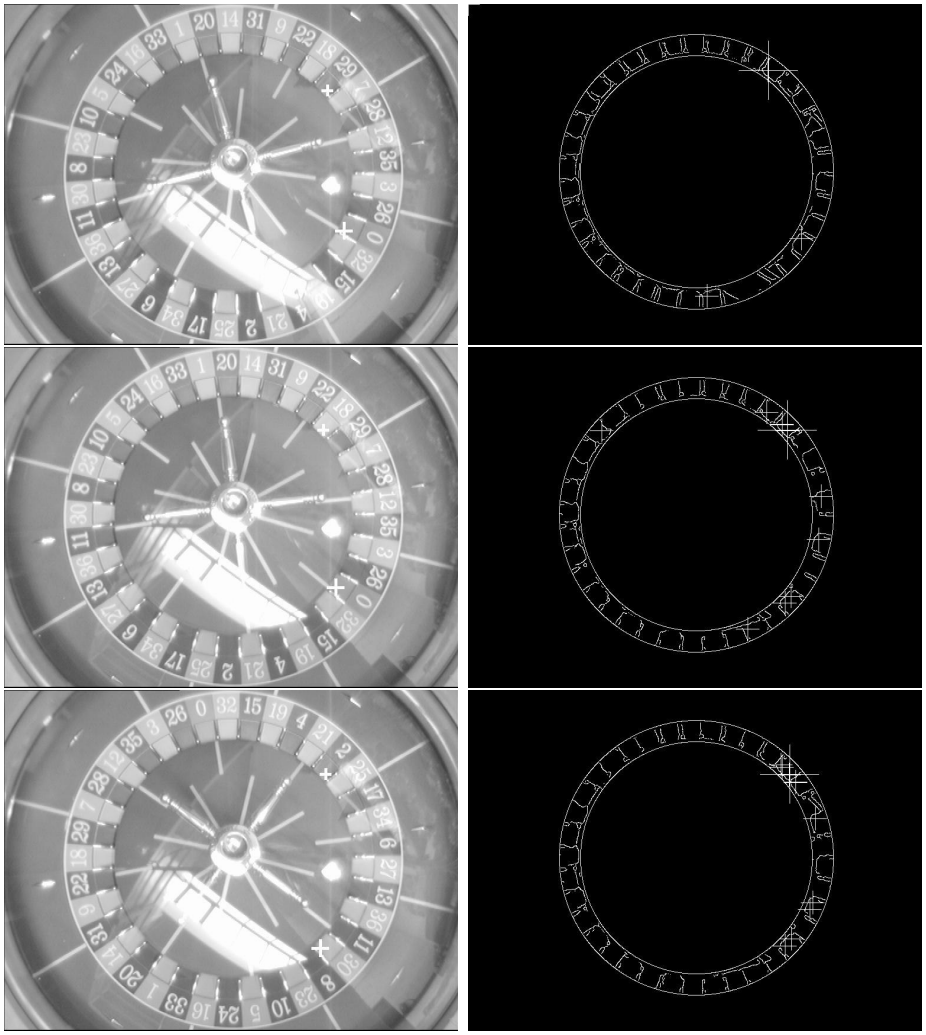
pri igralniški ruleti, je, da mora biti vhodna slika dovolj velika. To je težko doseči z zajetim kompresiranim posnetkom, npr. AVI, ker večina video kartic kompresira posnetke manjše resolucije, npr. 320 x 240 slikovnih elementov. Zaradi tega nisem uporabil kompresiranih posnetkov pri testiranju. Upošteval sem, da je posnetek zaporedje slik. Posnetke sem naredil tako, da sem s kamero, ki je uporabljena v sistemu, opisanem v pričujočem delu, posnel zaporedje slik, ki predstavlja posnetek. Tako sem pridobil posnetke velikosti 750 x 580 slikovnih elementov in zahtevi po dovolj velikem posnetku je bilo zadoščeno. Vsak tak posnetek predstavlja zaporedje slik od začetka vrtenja rulete do tedaj, ko kroglica pade na številko in jo obstoječi sistem avtomatske igralniške rulete pobere iz rulete. Ko kroglica pade na številko, obsoječi sistem igralniške rulete počaka tri obrate kolesa rulete, nakar kroglico pobere iz rulete.

Posnetki, uporabljeni pri testiranju, so bili narejeni časovno zaporedno, eden za drugim, in predstavljajo igro avtomatske rulete v določenem časovnem intervalu.
Algoritem dosega nad učno množico 10 posnetkov 100% točnost. Rezultati nad učno množico so prikazani v dodatku B.0.1. Testna množica vsebuje 42 posnetkov. Nad testno množico je algoritem dosegel 83.33% točnost (glej tabelo 5.1). Pravilni rezultati testiranja nad testno množico so prikazani v dodatku B. V tem razdelku se bomo osredotočili na napačne rezultate algoritma pri testiranju nad testno množico in bomo pogledali, kaj so bili razlogi za napake. Napačni rezultati testiranja nad testno množico bodo tudi nakazali šibke točke algoritma.



Slika 5.1: Napaka pri detekciji ničle. Levo je originalna slika, na kateri je z majhnim križem označena detektirana pozicija kroglice, z velikim križem pa pozicija ničle. Desno je slika robov leve slike. Z majhnimi križci so prikazane najverjetnejše pozicije kroglice, z velikim križem je označena tista pozicija izmed predlaganih, ki jo izbere Houghov transform.

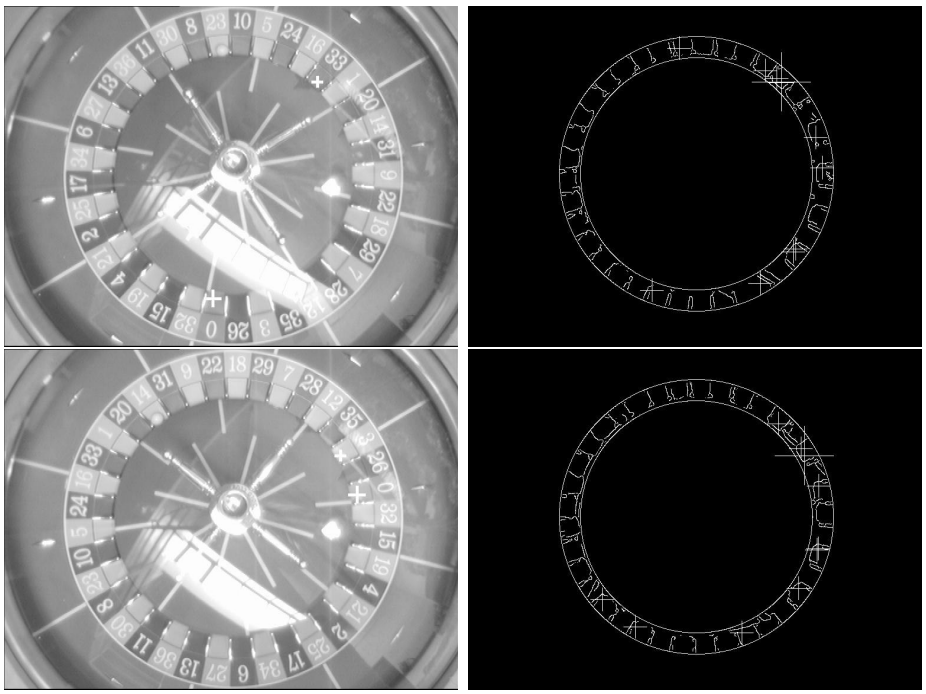
Napake algoritma so predstavljene s sliko, na kateri je z majhnim debelejšim križcem prikazana detektirana pozicija kroglice, z velikim debelejšim križem pa detektirana pozicija ničle. Poleg vsake slike, na kateri je prikazana napačna detekcija kroglice oz. ničle, je tudi robna slika, na kateri so z majhnimi križci prikazane možne pozicije kroglice, ki jih predlaga iskanje najmanjše kvadratične napake ujemanja slike z vzorcem kroglice in velik križ, ki predstavlja pozicijo kroglice, ki jo iz možnih pozicij kroglice izbere algoritem Houghovega transforma. Problem svetlobnih odbojev od steklenega pokrova rulete je prikazan pri napaki na sliki 5.1. Kroglica je bila pravilno detektirana, torej je del za detekcijo kroglice pravilno ugotovil točno pozicijo kroglice. Problem je pri detekciji ničle. Zrcalni svetlobni odboj se nahaja na delu števílčnice rulete, kjer je ničla. Zaradi tega je postalo črno polje svetlejše in detektiral se je vzorec, ki predstavlja ničlo in posledično je bila pozicija ničle napačno detektirana. Taka vrsta napak je najbolj kritična pri sistemu, opisanem v



Slika 5.2: Napake pri detekciji kroglice.

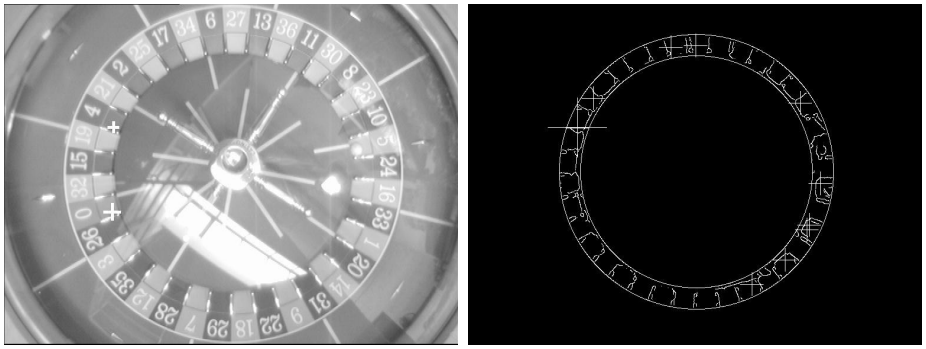
tem diplomskem delu, ampak se na srečo dogaja bolj redko. Rešitev bi bila lahko taka, da bi čakali, da se kroglica večkrat detektira in bi potem kot pravo številko, na kateri je kroglica, proglasili številko, ki se je največkrat detektirala. Tako bi se ena napačna detekcija številke znotraj niza enakih detekcij presegla. Po drugi strani, bi ta rešitev privedla do zakasnjenega odzivanja sistema, ker bi minilo nekaj časa od trenutka, ko je kroglica padla na številko, do trenutka, ko bi sistem proglasil številko, na kateri je kroglica. To bi zelo ogrozilo cilj delovanja v realnem času in hitrega odzivanja sistema, kar je zelo pomembno za igro igralniške rulete. Poleg tega se kroglica nahaja na številčnici le malo časa, odkar pade na številko, ker jo sistem avtomatske igralniške rulete hitro odstrani iz številčnice. Zato je vprašljivo,

koliko detekcij kroglice bi uspeli dobiti v tem kratkem času in vprašljivo je, ali bi bil niz detekcij kroglice dovolj dolg, da bi iz njega lahko karkoli sklepali. Zaradi vsega tega in zaradi dejstva, da se ta napaka ne zgodi pogosto, opisane ideje nismo implementirali.
Vzrok za napako pri detekciji številke, na katero je padla kroglica, na slikah 5.2 je napačna detekcija kroglice. Kroglica se je detektirala, čeprav sploh še ni padla na številko. Za odpravljanje te napake so dve možnosti, in sicer povečanje praga minimalne napake ujemanja vzorca kroglice s sliko ali pa povečanje praga za detekcijo kroglice s Houghovim algoritmom. Obe rešitvi povzročita, da se kroglica v nekaterih drugih posnetkih ne bo detektirala, ker bodo pragovi previsoki.



Slika 5.3: Napaka pri detekciji kroglice. Previsoko postavljen zgornji prag τ_h, ki je vhod v Cannyjev detektor robov. Kroglica na sliki robov ni vidna.

Vzrok za napačno detekcijo kroglice na slikah 5.3 je previsoko postavljen zgornji prag τ_h, pri Cannyjevem detektorju robov. Odziv filtra za detektiranje robov je na robovih kroglice premajhen. Znižanje tega praga odpravi problem pri tej konkretni napaki, obenem pa spusti preveč šuma na robno sliko in povzroči napačne detekcije kroglice pri drugih primerih. Ker je kontrast med svetlimi polji številčnice rulete in kroglico majhen, je potrebno pazljivo nastaviti pragove pri detekciji robov. Če je prag premajhen, spustimo preveč šuma na sliko, če je pa prevelik, na svetlejših



Slika 5.4: Napaka pri predlaganju najverjetnejših pozicij kroglice. Pozicija kroglice ni bila predlagana, čeprav je vidna iz slike robov.

poljih težje detektiramo kroglico.
Na sliki 5.4 je prikazana napaka pri detekciji kroglice. Algoritem za predlaganje možnih pozicij kroglice na sliki ni predlagal pozicije, kjer je kroglica. To se je zgodilo zato, ker je prag ujemanja vzorca kroglice s sliko postavljen previsoko ali pa ker si zapomnimo premalo najboljših možnih pozicij kroglice. Rešitvi sta dve: zmanjšanje praga ujemanja vzorca kroglice ali pa povečanje seznama možnih pozicij kroglice. Boljša je druga rešitev, ki sicer privede tudi do počasnejšega delovanja sistema, saj se Houghova transformacija večkrat izvede.
Kot je razvidno iz zgoraj opisanih vzrokov napačnih detekcij kroglice, je pravilna nastavitev parametrov sistema zelo pomembna. Nastavitev parametrov je odvisna od zunanjih pogojev, v katerih sistem deluje. Težko je doseči, da so parametri optimalno nastavljeni. Prevelika prilagoditev enega parametra za določen primer detekcije ponavadi pomeni, da se detekcija nekega drugega primera poslabša. Pri testiranju so bili parametri sistema nastavljeni tako, da je sistem dosegel 100% točnost nad učno množico, kar pa ne zagotavlja, da so bili nastavljeni optimalno. Namen testiranja ni bil optimalno nastaviti parametre, ampak prikazati šibke točke sistema.
Pri testiranju se je izkazalo, da je delovanje sistema v realnem času uresničeno. Hitrost delovanja algoritma je odvisna od nastavljenih parametrov sistema, predvsem od števila najboljših pozicij kroglice, ki si jih zapomnimo, kajti na vseh teh pozicijah se sproži Houghov tranform, ki je časovno zelo potraten. Drugi parameter, ki vpliva na hitrost delovanja, je tudi širina krožnice, po kateri iščemo kroglico. Ožja kot je, hitreje deluje sistem. Na slikovnih elementih krožnice namreč sprožimo iskanje najboljšega ujemanja slike z vzorcem krožnice, kar je tudi časovno potratno. Pri testiranju je sistem porabil povprečno 0.3 sekunde za obdelavo ene zajete slike.

Poglavje 6

Zaključek in nadaljnje delo

Kot je razvidno iz rezultatov, opisani sistem ne dosega točnosti avtomatske rulete, ki je implementirana s pomočjo elektronike in senzorjev in ki dosega več kot 99% točnost detekcije. Namen pričujočega dela ni bil narediti ekvivalenten sistem s pomočjo računalniškega vida, ampak le raziskati možnosti uporabe računalniškega vida v opisanem primeru, ter s tem tudi v drugih podobnih problemih pri igralniških igrah. Opisani sistem bi se dalo izboljšati z izboljšavami, ki bodo opisane v nadaljevanju tega poglavja.
Prva izboljšava opisanega sistema, za detekcijo številke, na katero je padla kroglica, bi bila zamenjava kamere, ki podpira PAL video standard z dražjo digitalno kamero. S tem bi pridobili na vertikalni resoluciji slike in podvajanje vrstic ne bi bilo več potrebno, poleg tega pa bi bilo možno dobiti več kot 25 slik na sekundo.
Največji problemi, ki se pojavljajo v opisanem sistemu, so povezani s svetlobnimi odboji. Svetlobne odboje močno poveča predvsem steklen pokrov rulete. Zaradi tega bi nadaljnje delo na opisanem sistemu moralo temeljiti predvsem na odpravljanju svetlobnih odbojev. Možne izboljšave v smeri zmanjšanja svetlobnih odbojev so naslednje:

- Uporaba digitalne barvne kamere namesto analogne črnobele kamere.
- Uporaba HSI barvnega prostora namesto RGB barvnega prostora pri barvni sliki.
- Uporaba polarizacijskega filtra.

Barvna kamera potrebuje več svetlobe za zajemanje slike kot črnobela kamera in je zato manj občutljiva na svetlobne odboje kot črnobela kamera. Ker imamo opravka s problemom, kjer nastopa ogromno svetlobe, ki predstavlja problem, bi ta lastnost barvne kamere lahko koristila sistemu, opisanem v tem diplomskem delu in bi se tako občutljivost na svetlobo malo zmanjšala.

Če bi uporabljali barvno kamero namesto črnobele kamere, bi lahko uporabili HSI barvni prostor namesto RGB barvnega prostora. Pretvorbo iz RGB v HSI barvni prostor dobimo po enačbi [6]:

$$\begin{aligned} H &= atan\left(\frac{\sqrt{3}(G-B)}{(R-G)+(R-B)}\right) \\ S &= 1 - \frac{\min(R,G,B)}{I} \\ I &= \frac{R+G+B}{3} \end{aligned} \tag{6.1}$$

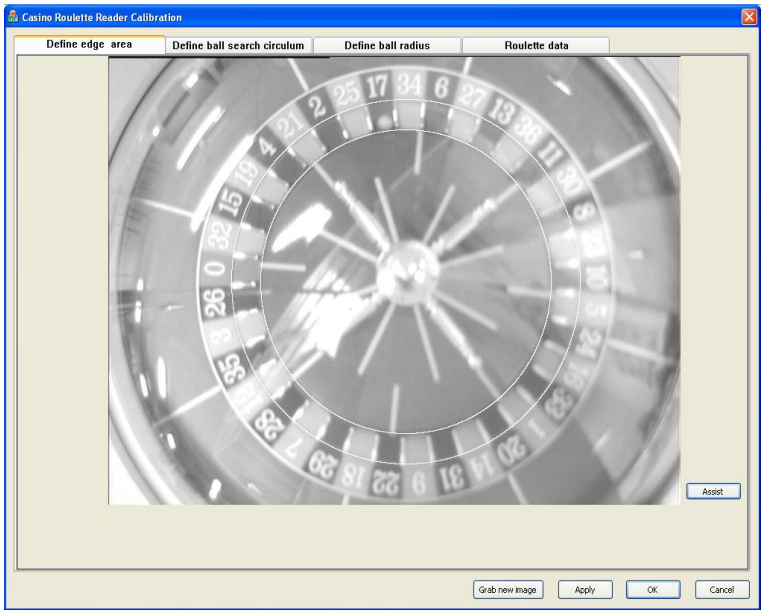
Pri HSI barvnem prostoru predstavlja komponenta H barvo, komponenta S pove kako čista oz. globoka je barva in komponenta I predstavlja količino svetlobe. Iz tega je očitno, da bi velik del svetlobe lahko izločili iz slike za ceno malo počasnejšega delovanja sistema, ker je pretvorba 6.1 časovno kompleksna. Tako bi lahko dobili iz slike samo barvno informacijo brez svetlobnih odbojev. Barvna informacija je v HSI barvnem prostoru predstavljena s komponento H, ki ima na žalost, najbolj kompleksen izračun pri pretvorbi iz RGB prostora v enačbi 6.1.
Svetlobne odboje bi lahko zmanjšali tudi z uporabo polarizacijskega filtra, ki bi ga postavili med ruleto in kamero. Polarizacijski filter je posebna leča, ki omili svetlobne odboje.

Dodatek A

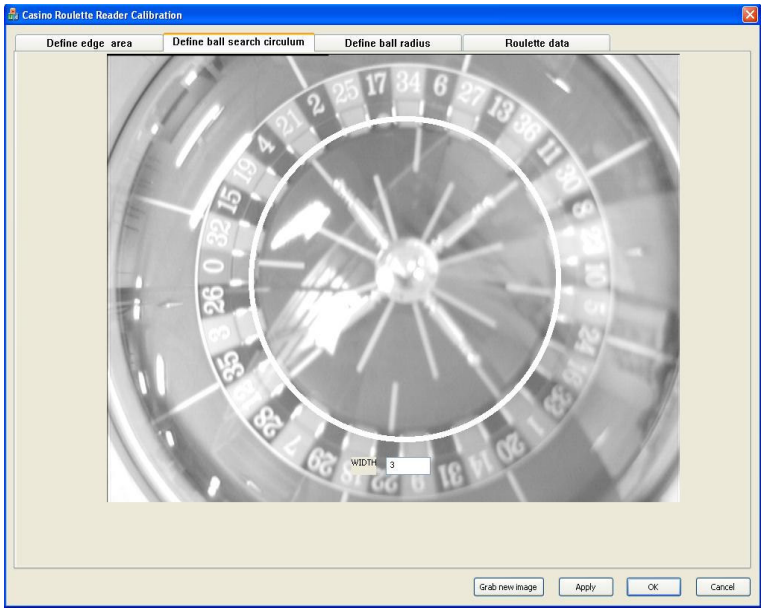
Kalibracijski podsistem

Kalibracijski podsistem je vglavnem interaktiven, delno je pa tudi avtomatiziran. Glavni del tega podsistema je uporabniški vmesnik (GUI), preko katerega se vnesejo potrebni podatki v sistem. Implementiran je kot povsem ločen program, ki na koncu shrani nastavitve v tekstovno datoteko 'Settings.cfg'.
Slika se zajame s kamere s pritiskom na gumb "Grab new image". Najprej uporabnik določi zunanjo in notranjo krožnico, ki omejujeta območje, na katerem se bo izvedla detekcija robov (slika A.1). To naredi tako, da z miško dvakrat klikne, nakar se mu pojavi krožnica, ki jo nato raztegne na željen radij tako, da drži stisnjen levi miškin gumb na krožnici. Center označene krožnice lahko premika s stisnjenim desnim miškinim gumbom. Trenutna krožnica se mu prikazuje na zaslonu, to je narejeno tako, da se riše bela krožnica na prazno (črno) podlago. Ko se slika izrisuje na zaslon, se na mestih, kjer je krožnica, riše bele slikovne elemente, ne pa slikovnih elementov prikazane slike. Zaželjeno je, da si uporabnik pri tem pomaga z gumbom "Assist". S pritiskom na ta gumb se v ozadju sproži Houghova transformacija, ki poskuša pozicionirati krožnico na zunanji ali notranji del številčnice. Zunanji in notranji del številčnice sta na sliki zarobljena, tako da Houghov transform lahko išče krožnico, ki se najbolj prilega. Zaradi prostorske in časovne kompleksnosti Houghovega transforma, mora najprej uporabnik nastaviti krožnico ± 10 slikovnih elementov od roba, ker nato Houghov transform išče krožnice, ki so zamaknjene le za 10 slikovnih elementov od trenutne pozicije ter imajo radij le za 20 slikovnih elementov različen od trenutnega. Če je slika preslaba, da bi Houghov transform pravilno pozicioniral krožnico, jo mora uporabnik sam pozicionirati. Naslednji korak je določiti, kje naj sistem išče kroglico. To določimo s krožnico, kot je prikazano na sliki A.2. Na tej krožnici se mora nahajati kroglica, ko je padla na številko. Določimo tudi širino krožnice, da razširimo področje iskanja žogice. S tem skrajšamo čas iskanja žogice ter s tem pohitrimo sistem. Ko sistem za iskanje žogice najde žogico na tej poziciji, privzame, da je kroglica padla na številko.
Nato določimo radij kroglice. To naredimo tako, da kliknemo na kroglico na sliki

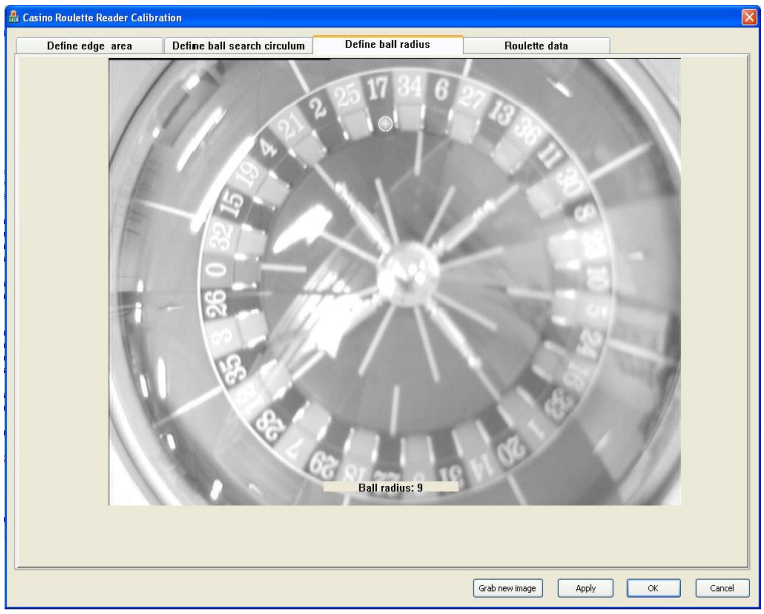
(glej sliko A.3), nakar se sproži Houghov transform za iskanje kroglice z radijem med 0 in 50 slikovnih elementov na območju 100x100 slikovnih elementov s središčem v točki, na katero smo kliknili. Tako se določi radij kroglice. Na sliki se tudi izriše krožnica, ki je bila najdena (glej sliko A.3). Preostane nam še določiti preostale parametre sistema (glej sliko A.4), kot so: Gaussova σ, zgornji in spodnji prag za detekcijo robov, koliko vzorcev kroglice uporabljamo, koliko najboljših pozicij, kjer naj bi bila kroglica, si zapomnimo pred izvedbo Houghovega transforma, koliko točk mora glasovati za krožnico, da proglasimo najdeno kroglico in minimalen procent ujemanja vzorca kroglice s sliko, da si zapomnimo možno pozicijo kroglice. Glavni rezultat kalibracijskega podsistema je datoteka 'Settings.cfg', v kateri so zapisane vse tukaj nastavljene nastavitve. Ta datoteka je vhod v drugi podsistem, sistem za detektiranje, na katero številko je padla kroglica.



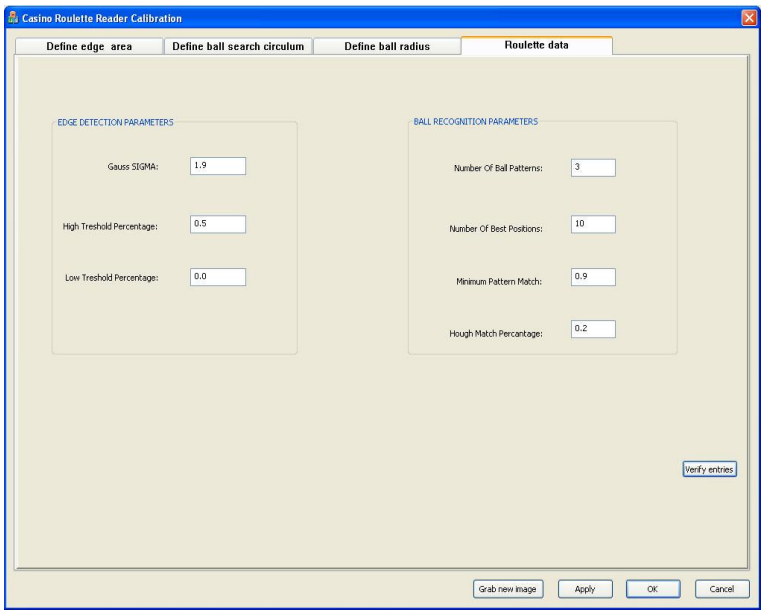
Slika A.1: Označevanje številčnice rulete, kjer se sproži detekcija robov. Območje med notranjo in zunanjo belo krožnico določa območje, kjer je številčnica in kjer se bo izvedla detekcija robov.



Slika A.2: Označevanje krožnice, na kateri se nahaja kroglica, ko je padla na določeno številko. To je območje iskanja kroglice. Debelino krožnice in s tem območja vpišemo v polje "WIDTH".



Slika A.3: Določitev radija kroglice. S klikom na kroglico na sliki se sproži Houghov transform, ki avtomatsko določi natančno pozicijo kroglice in radij kroglice.



Slika A.4: Slika prikazuje podajanje vrednosti parametrov sistema. Vrednosti, ki se določajo so npr. σ, spodnji in zgornji prag detekcije robov itd.

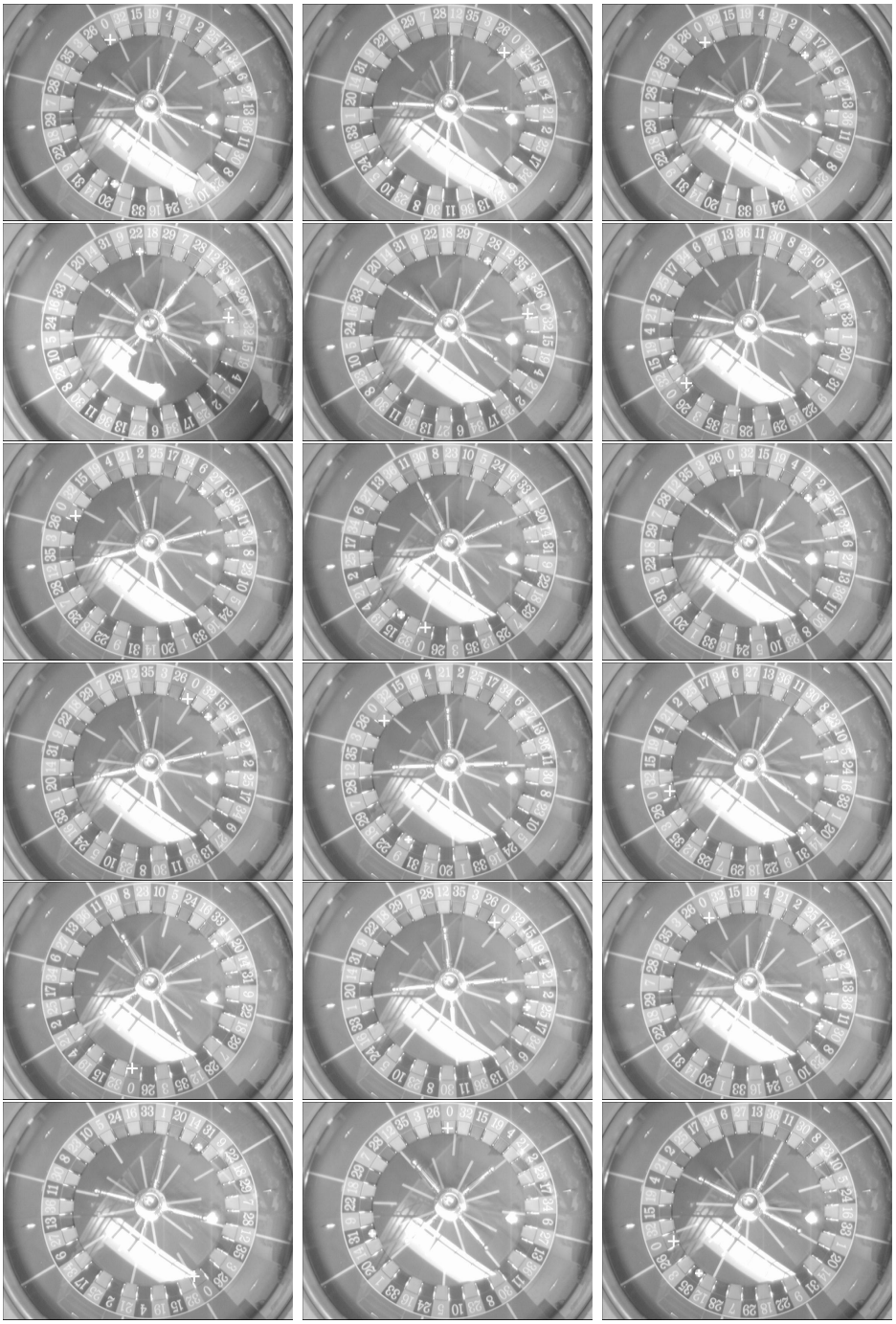
Dodatek B

Pravilni rezultati nad testno množico

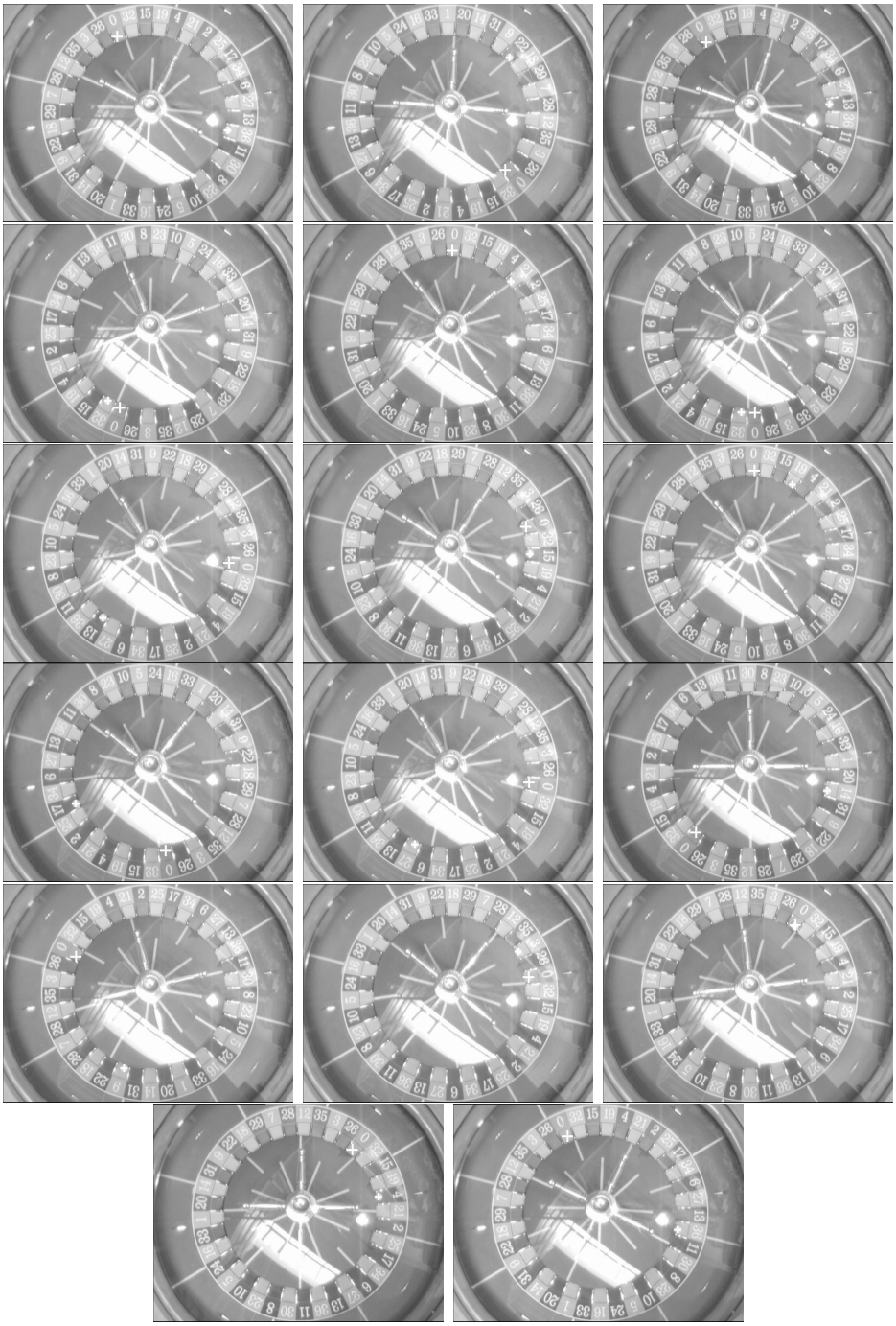
Na slikah B.1 in B.2 so prikazani pravilni rezultati testiranja nad testno množico. Z velikim križem je prikazana detektirana pozicija ničle, z malim križem pa detektirana pozicija kroglice. Negativni rezultati testiranja so prikazani v poglavju 5. Pri testiranju so bile uporabljene naslednje nastavitve:
Središče rulete $Center$ v točki $(386, 283)$, notranji radij kroga $Radius$, ki omejuje območje, kjer se izvede detekcija robov je bil enak $Radius = 195$, zunanji radij $Radius1$ območja detekcije robov je bil enak $Radius1 = 233$, radij kroga, ki označuje območje, kjer je kroglica, ko pade na številko, je bil enak $BallSearchRadius = 212$, širina te krožnice je bila $BallSearchCirculumWidth = 4$, radij kroglice je bil enak $BallRadius = 9$, Gaussova σ je bila $\sigma = 1.9$, $\tau_h = 0.5$, $\tau_l = 0$, število vzorcev kroglice $N\_Pattern = 3$, število najboljših pozicij kroglice $N\_BestPos = 10$, minimalno ujemanje z vzorcem kroglice, da ga sprejmemo v seznam najboljših pozicij kroglice $Min\_Patt\_Match = 0.9$ in odstotek točk krožnice, ki morajo glasovati za določeno pozicijo kroglice pri Houghovem transformu $HoughTresh = 0.2$.

B.0.1 učna množica

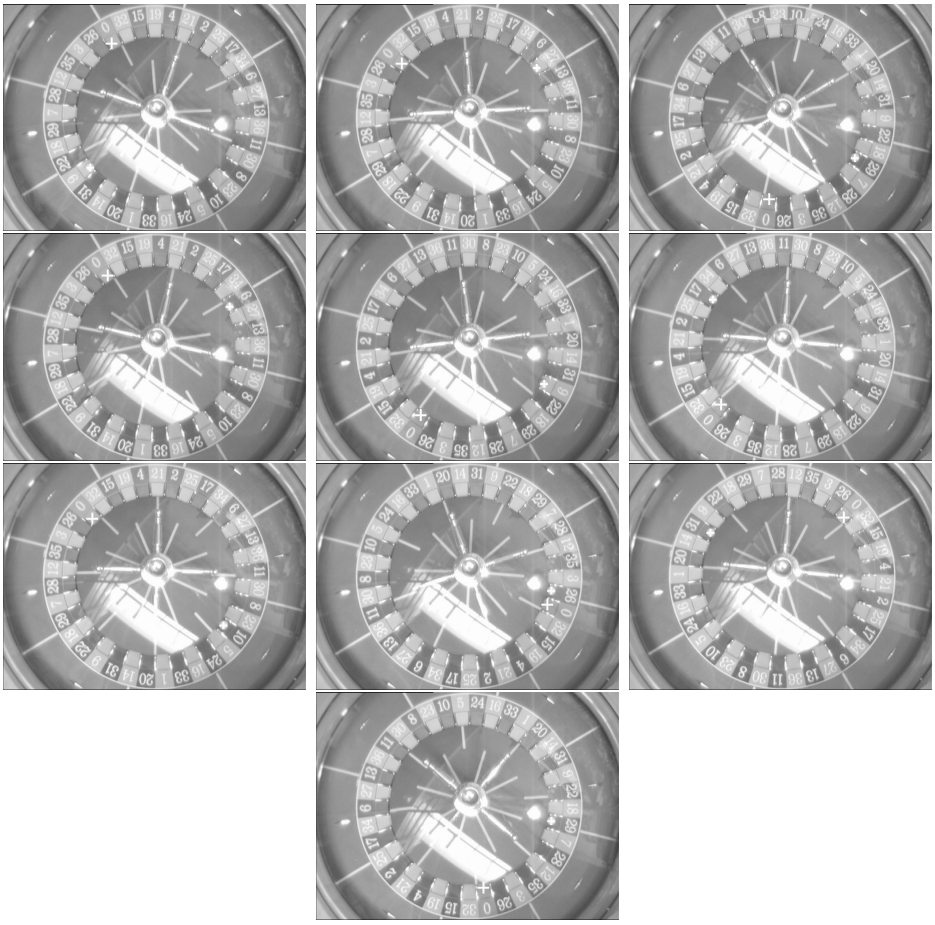
Na sliki B.3 je prikazana učna množica oz. množica, uporabljena za kalibracijo sistema. Na slikah so označeni rezultati sistema opisanega v pričujočem diplomskem delu (označena je pozicija kroglice in pozicija ničle). Algoritem dosega 100% točnost nad učno množico.



Slika B.1: Pravilni rezultati algoritma nad testno množico.



Slika B.2: Drugi del pravilnih rezultatov testiranja algoritma nad testno množico. Z majhnim križcem je označena pozicija kroglice, z velikim križem pa pozicija ničle.



Slika B.3: Rezultati testiranja nad učno množico. Algoritem dosega 100 odstotno točnost nad učno množico.

Zahvala

Za mentorstvo se zahvaljujem prof. dr. Francu Solini, univ. dipl. inž. Zahvaljujem se tudi podjetju GOLD CLUB d.o.o., ki mi je omogočilo izvedbo diplomskega dela. Največja zahvala gre pokojni mami in očetu, ki sta mi študij omogočila ter me pri njem podpirala.

Literatura

[1] E. Trucco, A. Verri: *Introductory Techniques for 3-D Computer Vision*, Prentice-Hall, 1998.

[2] I.N. Bronstein, K.A. Semendjajew, G. Musiol, H. Muhlig: *Matematični Priročnik*, 2. predelana in dopolnjena izdaja, Tehniška založba Slovenije, 1997.

[3] P. Peer: *Avtomatsko Iskanje Človeških Obrazov na Slikah*, diplomsko delo, 1998.

[4] M. Piccardi, T. Jan: *Recent Advances in Computer Vision*, The Industrial Physicist, vol. 9, no. 1, pp.18-21, Feb/Mar. 2003, American Institute of Physics.

[5] P. Courtney, P. Bottcher: *ECVision White Paper on Inustrial Applications of Cognitive Vision*, European Research Network for Cognitive Computer Vision Systems, April 2003.

[6] M. Lalonde and Y. Li: *Road Sign Recognition*, Collection scientifique et technique, CRIM-IIT-95/09-35, Centre de recherche informatique de Montreal, August 1995.

[7] Internet:

- *http://www-2.cs.cmu.edu/afs/cs/project/cil/ftp/html/vision.html*, Domača stran za računalniški vid.
- *http://www.spt.fi/eutist/*, Industrijski problemi v računalniškem vidu.
- *http://www.cs.ubc.ca/spider/lowe/vision.html*, Komercialne aplikacije računalniškega vida.
- *http://homepages.inf.ed.ac.uk/rbf/HIPR2/index.htm*, Opisi metod računalniškega vida.
- *www.casino-baden-baden.de/ e/spiele1.htm*, Igre na srečo.

Izjava o samostojnosti dela

Izjavljam, da sem diplomsko delo izdelal samostojno pod mentorstvom prof. dr. Franca Soline, univ. dipl. inž. Sodelavce, ki so mi pri delu pomagali, sem navedel v zahvali.

Tadej Žgur

Ljubljana, april 2005