Programska oprema za Preizkus z devetimi zatiči

David Zindović

Univerza v Ljubljani, Fakulteta za elektrotehniko, Tržaška 25, 1000 Ljubljana, Slovenija
E-pošta: dz0321@student.uni-lj.si

Povzetek. V tem dokumentu je predstavljen pristop in rezultati pri izdelavi programske rešitve za procesiranje stanja v naprednem preizkusu z devetimi zatiči (NHPT). Pri tem so bile uporabljene različne metode glajenja in upragovljanja slike, zaznavanja oblik in pridobivanja podatkov iz slike. Programska rešitev, ki jo opisuje ta članek, temelji na razpoznavanju števila zatičev v posodici, razpoznavanju števila postavljenih zatičev v mreži lukenj ter razpoznavanju stanja prekrivanja posodice zatičev z roko, iz česar nato ugotovi potek dogajanja in anotira videoposnetek.

Ključne besede: anotacija, NHPT, opencv, python, zatič

Abstract

This document presents the approach and results in the development of a software solution for processing the state of the advanced nine hole pin test (NHPT), using various methods of image bluring and thresholding, shape detection and image data acquisition. Based on the number of detected pins in the bowl, the number of pins placed in the grid of holes and the state of the bowl being covered by the hand, it determines the course of actions and annotates the video.

Keywords: annotation, NHPT, opencv, python, pin

1 Uvod

V sklopu izziva pri predmetu Robotski vid smo prejeli nalogo, da izdelamo programsko opremo za sistem, ki spremlja stanje pri preizkusu z 9 zatiči (angl. Nine Hole Pin Test – NHPT). Namen omenjenega preizkusa je ocena spretnosti dominante in nedominantne roke, kar se lahko uporabi pri vrednotenju nevrodegenerativnih bolezni.
Zaradi napredka na področju strojne opreme in programskih orodij je mogoče ovrednotiti tovrstne poskuse z minimalno količino strojne opreme in tipal [2]. Programsko opremo je v določenih primerih mogoče zasnovati tako, da zahteva le barvno kamero, kar lahko občutno zmanjša stroškovne zahteve po strojni opremi in s tem proizvodne stroške izdelka.

2 Metodologija

Za razvoj primerne programske opreme sem se odločil za uporabo metod, za katere sem izvedel na vajah za predmet Robotski vid. Za realizacijo svoje rešitve sem uporabil videoposnetka istega poskusa z obeh kamer. V tem poglavju bom opisal ovire in rešitve osnovnega problema ter nekaterih podproblemov le tega.
Moj pristop zajema razpoznavanje števila zatičev v posodici, števila postavljenih zatičev v mreži lukenj in stanja prekrivanja posodice zatičev z roko, iz česar nato ugotovi potek dogajanja in anotira videoposnetek.

2.1 Opis izziva

V sklopu izziva je bil cilj izdelati programsko opremo, ki lahko delom videoposnetka izvedbe NHPT preizkusa dodeli dogajanje oz. opis izbranega dela poskusa. Tekom preizkusa lahko uporabnik zatič prijema, prenaša, odlaga ali pa ima prazno roko.
Te štiri dogodke je potrebno pravilno razvrstiti tekom videoposnetka in izvoziti JSON datoteko, ki zajema podatek o časovnih oknih (v frame-ih oz. slikah videoposnetka) in o dogodku, ki se na le teh dogaja.
Pri preizkusu je potrebno devet zatičev, enega za drugim prenesti, iz posodice z zatiči v eno izmed lukenj v 3x3 mreži lukenj z eno roko. Ko so vsi zatiči postavljeni v luknje, jih mora uporabnik pobrati ven iz lukenj in odložiti nazaj v posodico enega za drugim (z isto roko). Poskus je potrebno izvesti z vsako roko ločeno.

2.2 Podrobnejši opis problematike

Osrednja problematika izziva, glede na moj pristop, je zaznavanje oblik. Pri tem pomaga dejstvo, da so zatiči temnih barv na svetli podlagi (v nadaljevanju plošča), ki pa je na nekoliko temnejši mizi.
Zatiči so valji, ki jih uporabnik prime v roke in jih s tem zakrije. Posledično je razpoznavanje barve in oblike med prenašanjem zatiča zahtevnejše. Prav tako se oblika zatiča spremeni glede na njegovo pozicijo. V posodici, v kateri začne vseh devet zatičev, so oblike in barve

popolnoma vidne. Ko pa je zatič postavljen v mrežo devetih lukenj je del zatiča neviden, ker je v luknji. Seveda se tekom poskusa spreminja orientacija vseh zatičev.

2.2.1 Oblika zatiča

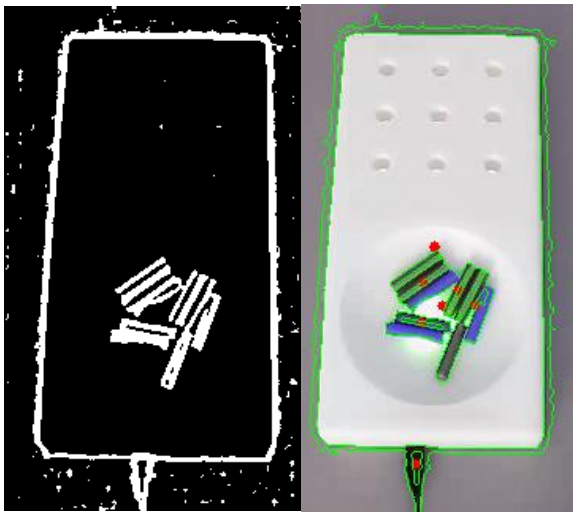
Ker se oblika zatiča spreminja, sem se odločil zaznavati zatiče le v stabilnih stanjih, torej v luknjah in posodici. Pri tem sem upošteval, da je mreža lukenj bližje eni kameri, medtem ko je posodica z zatiči bližje drugi kameri. Tekom poskusa težko pridobimo vse podatke iz enega videoposnetka, zato sem se odločil uporabiti obe kameri. Tako sem procesiral posnetek iz kamere, ki je bližje 3x3 mreži, in pridobil podatke o številu in poziciji zatičev izven posodice. S procesiranjem videoposnetka druge kamere, ki je bližje posodici, sem pa pridobil podatek o številu zatičev v posodici ter podatek o prisotnosti roke nad posodico.
Če bi želeli celotno dogajanje procesirati s pomočjo le enega videoposnetka, bi lahko naleteli na težavo: če bi bila roka bližje kameri, iz katere zajemamo posnetek, stanja na bolj oddaljenem delu plošče ne bi mogli zanesljivo zaznati, saj bi roka lahko zakrivala velik del pogleda na ploščo.

2.2.2 Osvetlitev zatičev

Pri sivinskih slikah zatičev v posodici se pojavi izrazita osvetlitev zatičev na sredini (na "vrhu" ležečega valja). Posledično pri upragovljanju srečamo rezultat, da namesto enega temnega lika najdemo dva tanjša temna lika s svetlim tankim likom med njima. Primer opisanega problema je viden na sliki 1 na spodnjem levem zatiču.

Slika 1: Zglajena slika z urejeno osvetljenostjo in kontrastom

Omenjena težava je zelo izrazita, ko so zatiči v posodici en zraven drugega, saj je stičišče med dvema zatičema zelo temne barve. Posledično je pri iskanju kontur prihajalo do tega, da je program našel le osvetljene oz. izrazito temne dele konture. Pogosto je zaznal tudi število zatičev v posodici kot število stičišč med zatiči.



Slika 2: Rezultat zaznavanja oblik pri prilagodljivem upragovljanju

2.2.3 Pozicija zatičev

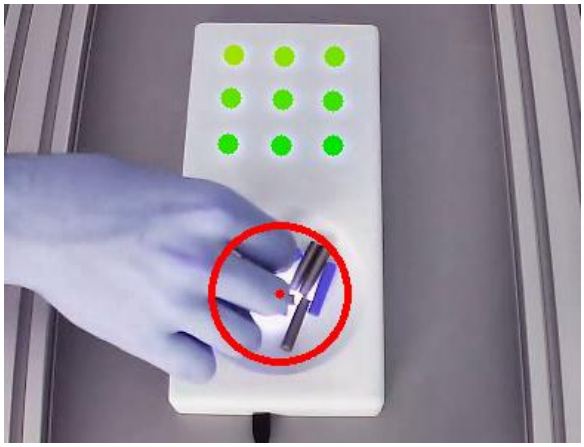
Pri vstavljanju zatičev se spremeni navidezna oblika zatiča, vendar obstaja možnost, da bi zatič izgledal primerljivo tudi med prenosom zatiča.
Posledično je potrebno iskati le zatiče, ki so bili uspešno postavljeni na pravilno pozicijo (t. j. v luknjo v 3x3 mreži lukenj). Za to je potrebno poznati pozicije lukenj, ki jih lahko najdemo s pomočjo iskanja osvetljenih lukenj na začetku poskusa.

2.2.4 Premik plošče

V primeru hitenja uporabnika lahko pride do premika plošče ali pa spremembe njene orientacije (ali pa obojega). V tem primeru se spremeni želena pozicija zatičev in pozicije lukenj v 3x3 mreži lukenj.
Premiku oz. zasuku plošče je zahtevno slediti tekom celotnega poskusa, saj lahko robove zanesljivo zaznamo le na začetku in na koncu, ko roka ne pokriva plošče. Ko roka pokriva ploščo, se spremeni njena kontura. Alternativna rešitev je sledenje določenemu delu plošče (npr. oglišču plošče) s pomočjo metode, kot je Lucas-Kanade za optično sledenje.

2.2.5 Pozicija posodice

Za pridobitev pozicije posodice sem izkoristil dejstvo, da je posodica vedno enako oddaljena od mreže lukenj za zatiče. Upošteval sem, da je sredina posodice vzporedna s sredinskim stolpcem mreže lukenj, središče pa je oddaljeno približno dva do tri vertikalne razmake od najbližje luknje za zatič.
Polmer posodice sem zaokrožil na dolžino enega horizontalnega razmaka v mreži lukenj za zatiče.



Slika 3: Zaznane luknje zatičev in središče ter območje posodice

2.3 Rešitve

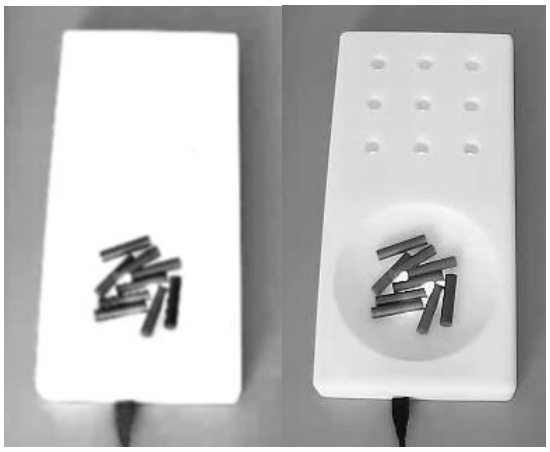
Za ovire, navedene v prejšnjem podpoglavju, sem poskusil različne rešitve, vendar so bile le nekatere učinkovite. Vmes bom izpostavil težave, ki sem jih poskušal rešiti s predstavljeno metodo.
V poglavju Rezultati bom predstavil primerjavo med različnimi metodami, ki sem jih uporabil za reševanje iste problematike.

2.3.1 Sivinska slika

Za enostavnejšo obdelavo sem barvno sliko pretvoril v sivinsko sliko, kar je RGB vrednosti pretvorilo v vrednosti 0-255. Za to sem uporabil funkcijo cv.cvtColor iz knjižnice OpenCV [1]. Sivinska slika je potrebna za realizacijo večine ostalih rešitev.

2.3.2 Upragovljanje

Za izkoriščenje kontrasta med zatiči in ploščo sem preizkusil uporabo navadnega in prilagodljivega upragovljanja s pomočjo vgrajenih funkcij v knjižnici OpenCV [1]. Pri navadnem upragovljanju sem nastavil zgornjo mejo, s pomočjo katere je program ločil temne dele slike od svetlih in ustvaril binarno sliko. Pred upragovljanjem sem moral na sliki manipulirati kontrast in svetlost s pomočjo funkcije cv.convertScaleAbs in alfa ter beta parametroma.
Prilagodljivo upragovljanje (angl. Adaptive Threshold) se je izkazalo za neprimerno za mojo problematiko, saj je mejo nastavljalo prenizko.
Kot alternativo za upragovljanje sem poskusil uporabljati tudi Sobelovo in Laplacovo jedro, saj sem želel izpostaviti prehode med svetlimi in temnimi konturami. Uporaba Laplacovega jedra je bila boljša od uporabe Sobelovega jedra, saj sem lahko tako na enostavnejši način spremljal spremembo v vseh smereh. Vendar tudi uporaba Laplacovega jedra ni bila dovolj koristna za nadaljnjo obdelavo.
Za osvetlitev sem poskusil uporabiti tudi CLAHE metodo in Histogram Matching, da bi razporedil osvetljenost po sliki, v upanju da bom lahko tako enostavneje zaznal temne objekte v posodici. Nobena od omenjenih dveh metod ni prišla prav, saj je bila "neugodna osvetlitev" prisotna le na določenih delih določenih objektov na sliki (svetel pas na sredini zatiča). Kot najprimernejši pristop za manipulacijo osvetljenosti in kontrasta se je izkazalo nastavljanje pravilnih alfa (2) in beta (-100) parametrov. Rezultat omenjenih parametrov je viden na sliki 4 (levo), ki predstavlja boljšo osnovo za upragovljanje, v primerjavi z osnovno sivinsko sliko (slika 4, desno).



Slika 4: Primerjava osvetljenosti in kontrasta

2.3.3 Glajenje

Ker je na slikah in videih z nižjo ločljivostjo prisoten šum, je priporočljivo slike zgladiti (znižati nenaden prehod med vrednostmi). S tem namenom sem preizkusil različne načine zameglitve z različnimi velikosti jeder. Primerjal sem uporabo MedianBlur in GaussianBlur funkcije iz OpenCV knjižnice [1], pri čemer se je izkazal GaussianBlur kot najbolj koristen za moj primer. Velikosti jeder sem med preizkušanjem spreminjal med 3×3, 5×5 in 9×9, pri čemer se je kot najprimernejša izkazala velikost 5×5.

2.3.4 Zaznavanje robov

Za zaznavanje robov sem uporabil Cannyevo metodo, ki temelji na zaznavanju prehodov glede na spodnjo in zgornjo mejno vrednost. Rob je zaznan, kadar prehod preseže spodnjo in nato zgornjo mejo oziroma obratno.
V primeru, da vrednosti preidejo čez eno izmed mej in ostanejo v vmesnem območju, določitev roba ni nujna.
Pred zaznavanjem je priporočljivo sliko pretvoriti v sivinsko in jo zgladiti, saj tako preprečimo napačno zaznavanje robov zaradi nenadnih prehodov, ki so lahko posledica šuma.

2.3.5 Zaznavanje oblik

Pri zaznavanju oblik sem uporabil funkcijo FindContours iz knjižnice OpenCV [1], saj ta na podlagi slike robov (npr. rezultat Cannyevega algoritma) določi konture. Za uspešno zaznavanje sem moral sliko primerno pripraviti s pomočjo rešitev iz prejšnjih poglavij.

2.4 Logika za dodeljevanje dogodkov

S pomočjo opisanih rešitev je moj program lahko prepoznal število pravilno vstavljenih zatičev in njihovo pozicijo, stanje roke nad posodico (če jo roka zakriva ali ne) in število zatičev v posodici (ko roka ne zakriva posodico).
Na ta način sem lahko zaznal spremembe, kot je na primer število vstavljenih zatičev, ter sprožil zapis dogodka.
Dogodki namreč v primeru navadnega poskusa sledijo zaporedju: pobiranje zatiča → prenos zatiča → odlaganje zatiča → prazna roka → pobiranje zatiča ...
Za namen bolj berljivega programa, sem poskus razdelil v dve fazi, in sicer faza "vstavljanja" in faza "odvzemanja". S prvo fazo sem označil, da se dogaja prenašanje zatičev iz posodice v mrežo lukenj, z drugo fazo pa sem označil, da se dogaja prenašanje zatičev iz mreže lukenj v posodico (pobiranje).
V fazi vstavljanja lahko prehode med dogodki opišemo s spremembami. Če pride do pokrivanja sklede z roko se je končal dogodek "prazna roka" in se je začelo pobiranje zatiča. Če roka več ne pokriva sklede, in se je zmanjšalo število zatičev, se je končalo pobiranje zatičev in se je začel prenos zatiča. Če pride do spremembe števila zatičev v mreži lukenj pomeni, da se je končal prenos zatiča in se je začel dogodek "odlaganje zatiča".
Ker moja programska oprema deluje na podlagi sprememb, ni mogoče vključiti realnočasnega dodeljevanja dogodkov oziroma ugotavljanja dogajanja na podlagi ene same slike (brez podatkov o preteklosti).
Posledično nabira podatke in beleži začetke in konce akcij, pri čemer za pobiranje (samo v fazi odvzemanja) in odlaganje (v obeh fazah) privzame, da trajata le eno sliko. Tako ob spremembi zabeleži konec zadnjega dogodka in doda trenutni dogodek (pobiranje ali odlaganje zatiča), ki je dolg eno sliko.

2.5 Podproblemi

Iz nabora podproblemov sem izbral nekaj težav, za katere sem ugotovil, da so rešljive z mojim pristopom. Rešitve zanje so opisane v tem podpoglavju.

2.5.1 Aktivna roka

Preizkus mora uporabnik izvesti prvo z uporabo ene roke, nato pa še z uporabo druge roke v ločenem poskusu. Za zaznavanje aktivne roke sem privzel, da bo uporabnik vedno sedel na enakem mestu. Tako sem v svoj program vključil logiko, ki gleda iz katere strani se približa roka plošči (gledano iz kamere, ki ima mrežo lukenj za zatiče bližje sebi).
Za zaznavanje prisotnosti roke sem od osnovne slike odštel sliko, na kateri sem sklepal, da bo roka prisotna (100. slika oz. frame). V rezultatu razlike nato štejem število svetlih polj v levi in desni polovici območja iskanja okoli plošče, ki je vnaprej določeno in stacionarno.

Slika 5: Primerjava osnovne slike (levo), slike z iztegnjeno roko (sredina), ter razlike (desno)

V primeru, da ima desna polovica večje število svetlih točk kot leva polovica, lahko zaključim, da uporabnik uporablja desno roko. V obratnem primeru pa levo.
Tekom preizkušanja sem se odločil za uporabo upragovljanja zaradi temnih barv, ki se pojavijo na sivinski sliki z iztegnjeno roko. Zaradi njih program težje loči med mizo in roko, kar sem rešil z upragovljanjem, to pa je vidno na skrajno desnem delu slike 5.

2.5.2 Barva zatičev

Zaznavanje barv program opravi tako, da procesira HSV vrednosti znotraj zaznanih kontur zatičev, pri čemer na podlagi vrednosti barve (V) določi eno izmed barv v vnaprej določenem naboru.
Ker sem na videoposnetkih zasledil, da zatiči niso po trije vsake barve, zapisane rešitve v zaključni rešitvi nisem uporabil.
V primeru, da bi bili po trije zatiči vsake barve, bi prilagodil izpis trenutnega stanja tako, da bi prikazoval barvo vsakega vstavljenega zatiča, saj sem razumel, da naj bi tako izgledal pravilen napredni test. Trenutna rešitev zmore le izpisovanje 3x3 tabele, ki predstavlja 3x3 mrežo lukenj za zatiče (gledano iz kamere, ki je bližje 3x3 mreži), pri čemer je v tabeli zajeto trenutno stanje mreže na videoposnetku (pozicije zatičev oz. zasedene luknje).

2.5.3 Prehiter začetek poskusa

Za zaznavanje prehitrega začetka poskusa sem privzel, da bi se poskus moral začeti, ko se vključijo lučke v mreži lukenj za zatiče in ne prej. Z namenom detekcije tega podproblema sem shranil sliko, ko sem zaznal vklop omenjenih lučk.
V primeru, da uporabnik prehitro začne poskus, bo na omenjeni sliki že prisotna roka. Posledično se to vidi pri razliki med sliko, na kateri je roka iztegnjena, in sliko, ko so se vključile lučke. V primeru prehitrega poskusa bo razlika med slikama minimalna (manjša od nastavljene meje), pri čemer bom razliko procesiral na enak način kot v poglavju "Aktivna roka" (z odštevanjem).

2.5.4 Sledenje premiku plošče

Pri iskanju rešitve za zaznavo premika oz. zasuka plošče sem privzel, da bodo spremembe na majhnem časovnem intervalu (med slikama) majhne.
Zasuku oz. premiku plošče sem želel prvotno slediti tako, da bi izbral določene točke robov plošče, ki bi jim sledil. V primeru spremembe lokacije teh točk bi poiskal transformacijsko matriko, ki bi jo nato uporabil za prilagoditev položajev lukenj za zatiče s pomočjo vgrajene funkcije warpAffine iz knjižnice OpenCV [1].

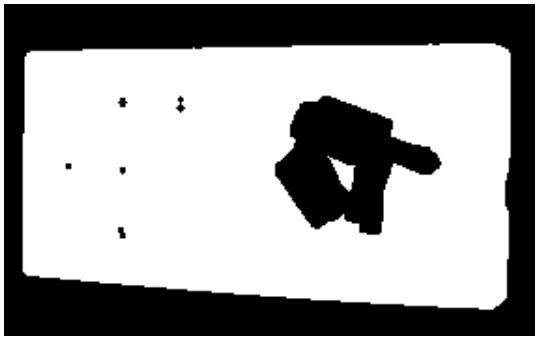
To se je izkazalo za neučinkovito, saj program ne more vedno videti vseh točk roba plošče zaradi pokrivanja z roko.
Kot alternativno rešitev sem preizkusil optično sledenje, pri čemer sem sledil izbrani točki na sliki, in sicer točki blizu oglišča plošče stran od uporabnika, saj tega dela roka nikoli ne zakrije tekom poskusa. Sledenje točki se je izkazalo za nezanesljivo, saj je pri manjšem premiku program zaznal enako točko drugje na sliki. Poskusil sem tudi slediti na zglajeni sliki, saj sem domneval da šum na sliki vpliva, vendar ni pomagalo.
Domnevam, da bi bilo sledenje delu oz. območju slike učinkovitejše, vendar nisem uspel napisati funkcije za takšno rešitev.

3 REZULTATI

V tem poglavju želim primerjati rezultate različnih rešitev za izbrano težavo.

3.1 Zaznavanje zatičev v posodici

Za namen zaznavanja zatičev v posodici sem primerjal Cannyevo metodo in navadno upragovljanje. Pri Cannyevi metodi sem preizkušal različne meje, predvsem sem menjaval načine pripravljanja slike za obdelavo s Cannyevo metodo. Rezultati so, ne glede na pripravo v najboljšem primeru, slučajno kazali pravilno število zatičev, vendar je kot zatič zaznal bodisi temne dele med zatiči ali pa svetle dele na sredinah zatičev. Zaradi tega sem moral pogosto prilagajati pogoje za zaznavo konture zatiča, vendar so bili rezultati nestanovitni oziroma nekonsistentni.
Ovira je bila tudi ta, da sem primerjal površino zaznanih kontur, ki pa je bila majhna, saj je zaznal program le del zatiča, ali pa velika, ker je zaznal več zatičev kot eno konturo.
Pri navadnem upragovljanju sem predpostavil, da bodo v posodici zatiči vedno ležali en zraven drugega in nikoli en na drugem. V primeru, da bi zatiči bili odloženi en na drugega, bi se njihovo število povečalo, medtem ko bi površina ostala enaka oz. bi se minimalno spremenila.
Uspel sem z navadnim upragovljanjem izoblikovati konturo, ki je zaobjela vse zatiče. Za tem sem s štetjem števila polj znotraj konture pridobil informacijo o površini devetih zatičev in s tem omogočil zanesljivo štetje zatičev v posodici. Vrednosti polj znotraj konture sem štel tako, da sem znotraj znanega območja posodice (slika 2) štel koliko polj ima vrednost 0 (črna barva). Pri manjšem številu zatičev se je pojavljala ovira, saj je program zaznal premajhno površino glede na pričakovano površino enega zatiča in je tako število zatičev bilo napačno. Za kompenzacijo sem upošteval faktor, ki je za zadnjih pet zatičev upošteval vsakič manjšo zahtevo za površino. Ta faktor program primerno nazaj povečuje, ko oseba na videoposnetku zatiče polaga nazaj v posodico.



Slika 6: Rezultat upragovljanja slike s ploščo in devetimi zatiči

3.2 Zaznavanje roke nad posodico

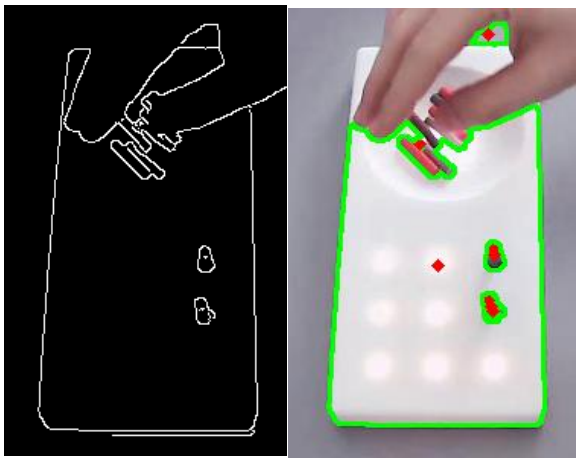
Zaznavanje roke nad posodico sem dodal s pomočjo enakega upragovljanja, kot pri zaznavanju zatičev v posodici. Razlika je ta, da v primeru, ko algoritem zazna veliko število zatičev (deset ali več), je površina temnega območja v posodici občutno večja, kar se zgodi le, ko roka prekriva posodico. Prekrivanje posodice vključuje tudi dele prstov, ki prispevajo k površini temno obarvane konture (slika 3).
Program stanje beleži v spremenljivki, ki vpliva na dodeljevanje dogodkov oz. proženje spremembe.

Slika 7: Rezultat upragovljanja slike, na kateri roka pokriva posodico

3.3 Zaznavanje zatičev v mreži lukenj

Za zaznavanje vstavljenih zatičev pripravim sliko za Cannyevo metodo, s pretvorbo v sivinsko sliko in z glajenjem. Po uporabi Cannyeve metode zaznane robove odebelim s pomočjo cv.dilate funkcije, za učinkovitejše zaznavanje kontur.
Pri konturah nato poiščem več parametrov, izmed katerih uporabim le razmerje (angl. aspect ratio) in krožnost (angl. circularity). Z določanjem pogojev za razmerje (med 1 in 3) , krožnost (nad 0.3) in površino (med 90 in 170) ugotovim ali je najdena kontura zatič v luknji ali ne. Poleg omenjenih pogojev mora biti središče zaznane konture znotraj območja, okoli središča ene izmed lukenj za zatiče. Informacijo o pozicijah lukenj za zatiče program pridobi na začetku poskusa, ko se lučke v luknjah prižgejo.

Pri tem problemu sem moral pogoje večkrat spremeniti, saj je v primeru prekrivanja mreže lukenj z roko program pogosto zaznal del roke kot zatič. V primeru, da bi zaznana kontura (del roke) imela središče blizu luknje za zatič, bi to lahko rezultiralo v napačno zaznanem zatiču.



Slika 8: Primerjava zaznanih robov in oblik

4 DISKUSIJA

Program z omenjenimi rešitvami in logiko je imel pri preizkušanju mero uspešnosti med 30 in 60 %, pri čemer sem to meril kot količnik med številom pravilnih anotacij in številom vseh anotacij.

Pri primerjavi referenčne anotacije in ustvarjene anotacije sem odkril, da se na večih točkah v večini primerih zgodi zamik anotiranih dogodkov. Trajanje dogodka je približno enako referenčnemu, vendar, ker pride zamik, in je pripisan napačen dogodek, to prispeva k slabši meri uspešnosti. Sklepam, da je razlog za to nezanesljivost ključnih dejavnikov (število zatičev, stanje roke nad posodico ipd.), ki v primeru manjše napake lahko proži dogodek prezgodaj ali pa ga konča prepozno.

Menim, da bi lahko logiko izboljšal tako, da bi podaljšal čas do potrditve nekega stanja, pri čemer bi tekom tega več spremljal roko in njeno pozicijo. Sklepam, da bi omenjena ideja lahko rešila tudi dodaten podproblem, saj bi z informacijo o poziciji ter smeri gibanja roke lahko več ugotovitev vpeljal v samo logiko.

V primeru, da bi želel svojo rešitev uporabljati, bi dodal tudi pregleden uporabniški vmesnik, saj imam trenutno le konfiguracijsko datoteko, v kateri lahko spreminjam zastavice za prikaz sprotnih stanj različnih delov programa (npr. stanje zatičev, prikaz ob spremembi, stanje v posodici itn.).

LITERATURA

[1] OpenCV. Pridobljeno s https://opencv.org/.

[2] D. Trihleb, V. Roman and V. Artem, "Increasing performance of image processing algorithms by computing during reception," *2021 International Conference "Nonlinearity, Information and Robotics" (NIR)*, Innopolis, Russian Federation, 2021, pp. 1-4, https://doi.org/10.1109/NIR52917.2021.9666076.