Poročilo projekta pri predmetu Robotski Vid

Maruša Kerpan, Jernej Rogelj, Timotej Klemenčič

Univerza v Ljubljani, Fakulteta za elektrotehniko, Tržaška 25, 1000 Ljubljana, Slovenija
E-pošta: mk9255@student.uni-lj.si, jr0318@student.uni-lj.si, tk8435@student.uni-lj.si

Povzetek.
Naša naloga je bila načrtati sistem za avtomatsko štetje časa pri “Nine Peg Hole” testu. Glavna ideja testa je vstavitev devetih palčk iz neurejenega stanja v urejeno stanje. To je iz posodice za palčke v devet enakomerno razporejenih lukenj ter nazaj. Pomemben je čas v katerem testna oseba opravi to nalogo. Iz časa vstavljanja in izstavljanja vseh palčk lahko predvidevamo zdravstveno stanje osebe. Če je na primer oseba doživela kap, bo vstavljanje in izstavljanje precej počasnejše in dosti bolj “nerodno” kot vstavljanje zdrave osebe.

Ključne besede: python, open cv, robotski vid, histogram, nine peg hole test

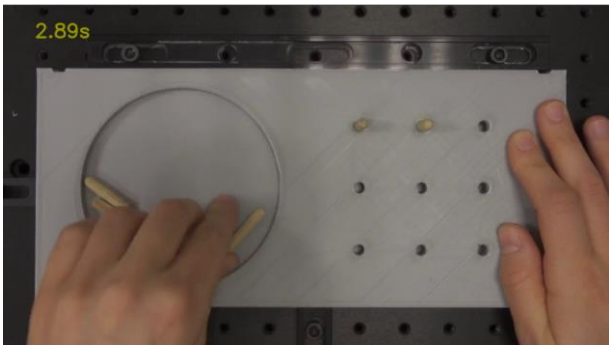
Report of a project at Robot Vision course

Our task was to design a system for automatic counting of time in the "Nine Peg Hole" test. The main idea of the test is to insert nine sticks from the unordered state to the ordered state. That is from a stick holder into nine evenly spaced holes and back. The time in which the test person performs this task is crucial. From the time needed to insert and remove all of the sticks, we can predict the health condition of the tested subject. For example, if a person has had a stroke, test time would be much slower and looked "clumsier" than the healthy persons test.

Keywords: python, open cv, robot vision, histogram, nine peg hole test

1 UVOD

Naloge smo se lotili s pomočjo programskega jezika Python in odprtokodne knjižnice Open CV. Uporabili smo videe, ki so nam bili posredovani skupaj z navodili za projekt na fakulteti.
Najprej smo se lotili analiziranja samega poteka “Nine Peg Hole” testa. Ta poteka tako, da merjena oseba iz neurejene posodice uredi palčke v enakomerno razmaknjene luknjice na drugi strani plošče. Preizkus se konča, ko oseba iztakne vse palčke iz luknjic nazaj v posodico. Čas vtikanja in iztikanja palčk se ves čas štopa. Začetek štetja časa napoči ob prihodu roke na sceno, konec pa po tem, ko v posodico pade zadnja iztaknjena palčka. Ta koncept je bil uporabljen v izvedbi naše naloge. Ves čas smo spremljali histogram na območju posodice. Ko se je ta močno spremenil, predvidevamo, da je roka vstopila v sceno. Tako vemo, pri katerem slikovnem okvirju lahko začnemo šteti čas. Med tem ves čas spremljamo stanje polnosti lukenj. Ko je vseh 9 lukenj praznih in se histogram na sceni posodice ponovno spremeni, vemo slikovni okvir končanja naloge. Sedaj nam preostane samo, da izpišemo čas med tema dvema okvirjema.



Slika 1: Scena "Nine Peg Hole" testa.

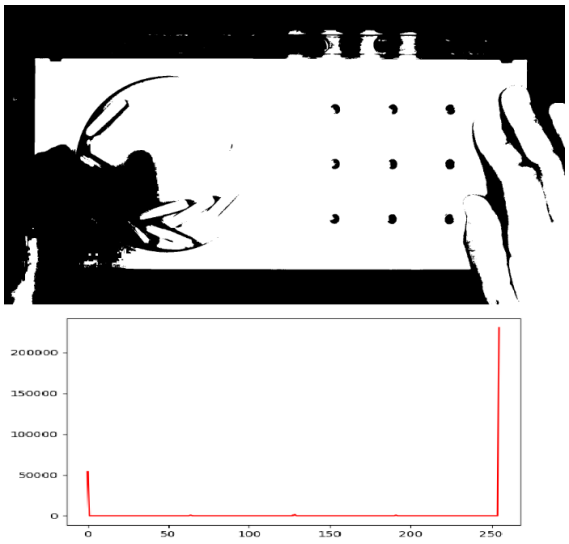
2 PROJEKT

V kodi najprej naložimo vse potrebne knjižnice, nato naložimo video datoteko ter jo shranimo v globalno spremenljivko, nato iz meta podatkov video datoteke pridobimo njeno hitrost – število sličic na sekundo in to shranimo v globalno spremenljivko. Sledi deklaracija ostalih potrebnih globalnih spremenljivk zatem pa osnovna programska zanka.

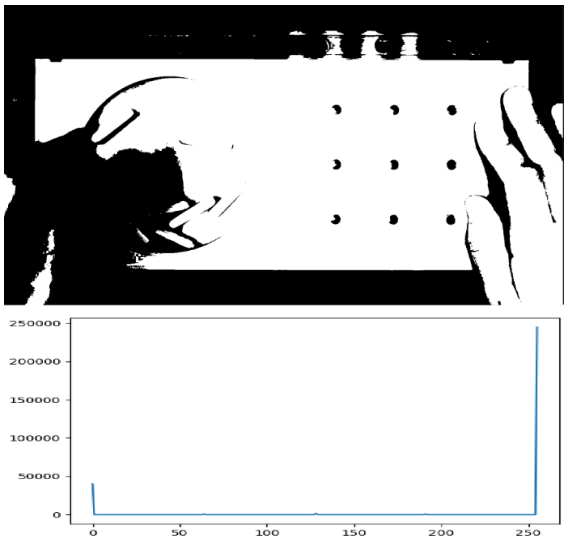
2.1 Zaznava roke in začetek štetja časa

Za zaznavanje kdaj pride roka v vidno polje posnetka smo se odločili uporabiti metodo primerjanja histogramov. Ko se zgodi ta dogodek, se začne merjenje časa. Ker imamo video s stotimi okvirji na sekundo, smo med sabo primerjali vsak peti okvir, saj so bile razlike v histogramih za vsak naslednji okvir premajhne za zaznavanje kdaj se slika spremeni. Uporabili smo if stavek s pogojem frameNum%5==0, kjer se

spremenljivka frameNum poveča za ena ob koncu vsake ponovitve glavne kode. Če je bil ta pogoj izpolnjen, smo pod spremenljivki H1 in H2 shranili histograma trenutne slike in slike iz pred petih ponovitev. Za tem smo s pomočjo metode cv2.compareHist in cv2.HISTCMP_HELLINGER primerjali histograma. Metoda nam vrne vrednosti med 0 in 1. V primeru da se histograma popolnoma prekrivata imamo vrednost 0, če sta pa popolnoma različna dobimo vrednost 1. V našem primeru je bila razlika v histogramih dovolj velika, ko je bila spremenljivka compareHist večja od 0.01. Ob tem dogodku, smo pod spremenljivko start_time shranili trenutni okvir posnetka in končali s pregledovanjem histogramov. Ker smo čas merili na podlagi tega koliko okvirjev se je zamenjalo med začetkom in koncem izvajanja testa, smo morali na koncu od končnega števila okvirjev odšteti začetnega in to število deliti z sličicami na sekundo.



Slika 2: Histogram scene preden roka vstopi v sceno.

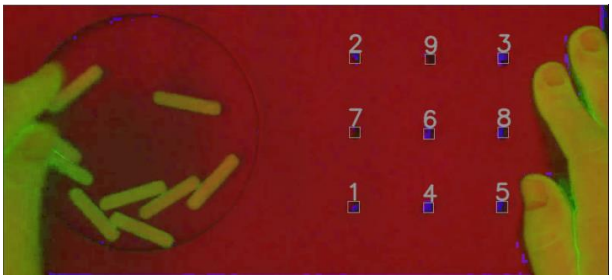


Slika 3: Histogram scene po tem ko roka vstopi v sceno.

Slika 2 prikazuje histogram in sceno preden roka pride v sceno. Torej preden začnemo šteti čas. Slika 3 pa prikazuje slikovni okvir, ob katerem zaznamo roko in začnemo šteti čas. Ob koncu stavka, ki ga ponovimo le vsake 5 okvirjev, pod spremenljivko *previousFrame* shranimo trenutno sliko. Na ta način se ob naslednjem izvajanju tega dela kode za nasljednji histogram, uporabi slika shranjena 5 okvirjev prej.

2.2 Zaznavanje lukenj

V osnovni programski zanki najprej pridobimo prvo sličico videa ter indikator, če je to zadnja sličica. Nato opravimo potrebne modifikacije te sličice in sicer ji spremenimo velikost ter shranimo posebej v upragovljeni ter HSV obliki. Iz upragovljene oblike z pomočjo *Houghes circles* poiščemo luknjice na prvi sličici videa (parametre nastavimo tako, da zazna le kroge v nekih mejah velikosti), zaznane kroge preštejemo in preverimo, če smo jih našli 9, če jih nismo nam terminal izpiše napako ter ustavi video. V vseh naslednjih sličicah videa izračunamo območje na njej kjer se nahaja vsaka posamezna luknja ter preverimo povprečno vrednost V-spremenljivke v HSV obliki te sličice. Če imajo v nekem trenutku vsa območja lukenj vrednost večjo od nekega pred nastavljenega parametra, postavimo indikator da so vse polne. Nato čakamo, da se vse luknje izpraznijo kar pomeni, da nehamo štopati test, to naredimo tako, da preverjamo če so vrednosti v vseh območjih manjše od nekega parametra. Ob tem program ustavimo in ta nam v terminal napiše pretečeni čas od začetka merjenja.



Slika 4: Primer zaznanih praznih lukenj v HSV obliki.

Slika 5: Primer nekaj polnih in nekaj praznih lukenj v HSV obliki.

3 ZAKLJUČEK IN MOŽNE IZBOLJŠAVE

Naloga je bila zanimiva, predvsem pa izziv na novem področju študija. Na začetku smo imeli nekaj težav z zaznavanjem lukenj, zato smo se naloge lotili na različne načine. Ko smo odpravili to težavo, smo preizkusili nekaj različnih metod za primerjavo histogramov, saj je bila naša začetna ideja, da bi roko zaznavali preko spremembe le-teh. Nalogo bi lahko rešili tudi na kakšen drugi način, na primer, da bi roko zaznavali na podlagi določene barve. Nadaljnje izboljšave trenutnega programa bi lahko bile usmerjene predvsem v zaznavanje senc, ki lahko potencialno zmotijo naš sistem, da zazna zapolnjene luknje ali izpraznjene luknje, ko se ta dogodek v resnici še ne zgodi. Nadgradili bi lahko nalogo tako, da bi štopali posebej čas vstavljanja in izstavljanja, ali pa da bi primerjali čas leve in desne roke med sabo.

LITERATURA

[1] Dokumentacija open cv: https://pypi.org/project/opencv-python/
[2] Iskanje krogov: https://docs.opencv.org/master/da/d53/tutorial_py_houghcircles.html
[3] Primerjava histogramov: https://docs.opencv.org/3.4/d8/dc8/tutorial_histogram_comparison.html
[4] Nine hole peg test: Mathiowetz, Virgil, et al. "Adult norms for the nine hole peg test of finger dexterity." *The Occupational Therapy Journal of Research* 5.1 (1985): 24-38.