SISTEM ZA TESTIRANJE UPORABNIŠKEGA VMESNIKA VGRADNEGA RAČUNALNIŠKEGA SISTEMA

Marko Vačovnik[1], Janez Perš[2]

[1]Gorenje d. d., Razvoj elektronike
E-pošta: marko.vacovnik@gorenje.com
[2]Laboratorij za strojno inteligenco
Fakulteta za elektrotehniko, Univerza v Ljubljani

POVZETEK: *Članek opisuje avtomatski sistem za testiranje uporabniškega vmesnika. Pri tem se osredotočamo na preverjanje vmesnikov vgradnih računalniških sistemov. Glavni cilj aplikacije je preverjanje pravilnosti prikaza receptov in nastavitev za avtomatsko pečenje na vgradni pečici. Za preverjanje pravilne umestitve sličic jedi je uporabljen primerjalnik deskriptorjev na osnovi histograma orientiranih gradientov, za preverjanje izpisanega besedila pa odprtokodno orodje za optično razpoznavanje znakov – Tesseract-ocr.*

1. UVOD

Uporabniški vmesniki so že dalj časa eno najbolj vročih področij razvoja elektronskih naprav, saj za končnega uporabnika predstavljajo edini način interakcije z napravo. Uporabniki si pri uporabi želijo barvite in pestre interakcije, zato lahko kompleksnost uporabniškega vmesnika hitro naraste. Kompleksnejša izvedba botruje večji verjetnosti pojavljanja napak, zaradi česar nastne večja potreba po testiranju. To pa pomeni vse višjo proizvodno ceno izdelka.

Testiranje uporabniških vmesnikov je dolgotrajen postopek, ki ga je potrebno v polnosti izvesti ob vsaki spremembi programske ali strojne opreme. Testiranje samo predstavlja kar 50-60% celotne cene razvoja programske opreme, zato je smiselno čim večji del testov avtomatizirati.

1.1 Avtomatsko testiranje uporabniških vmesnikov

Področje bi lahko razdelili na dva pola. Pri tem so predstavniki prvega pola invazivni, predstavniki drugega pa neinvazivni pristopi. Razlika med njima je to, da prvi neposredno posegajo v programsko kodo uporabniškega vmesnika, drugi pa ne. Iz tega je jasno, da morajo invazivni teči znotraj platforme, ki teste v ozadju izvaja, medtem ko lahko neinvazivni tečejo tudi na drugi napravi v kolikor je ta povezana s testirano napravo.

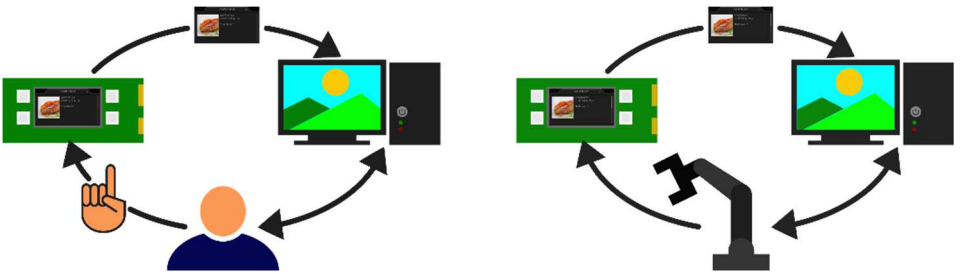
Neinvazivne pristope lahko razdelimo še na pristope za preverjanje s pomočjo skipt in postopke posnemi-ponovi (ang. *Record-playback*). Pri prvih strokovnjak, ki ustvarja testne scenarije napiše program, ki komunicira direktno z uporabniškim vmesnikom, pri drugih pa poteka v prvi fazi snemanje uporabnikovih interakcij z vmesnikom. Celotna seja interakcije se posname in se lahko kasneje ponovi avtomatsko.

Naš cilj je v začetku bil avtomatizacija testnih postopkov za testiranje uporabniškega vmesnika pečice, ki so najbolj zamudni in periodični. Preverjanje receptov, prikazanih na zaslonu uporabniškega vmesnika pečice je zelo dolgotrajen postopek, saj uporabniški vmesnik vsebuje na šestih različnih modelih skupno preko 300 receptov jedi v tridesetih različnih jezikih. Pri tem je potrebno pozorno prebrati celotno besedilo celotnega recepta – zank za znakom. Poleg receptov je časovno potratno tudi preverjanje pravilnosti prikazanega predvidenega časa pečenja na zaslonu z nastavitvami za pečenje jedi, za katere so na voljo recepti. Pri tem je potrebno spreminjati količino ter želeno intenziteto zapečenosti jedi, s čimer se spreminja tudi predviden čas pečenja.

Ker je elektronika pečice »svet zase«, pridejo v poštev samo algoritmi za avtomatsko testiranje, ki temeljijo na neinvazivnih pristopih. Interakcijo uporabnika s pečico je s pomočjo testne naprave – v našem primeru PC – nemogoče zaznati, zato odpadejo tudi metode *posnemi in ponovi*. Za naš primer je praktično izvedljiva samo metoda za avtomatsko preverjanje z uporabo skript, saj je scenarij periodičen in enostavno obvladljiv.

1.2 Pregled arhitekture uporabljenega sistema

Naš sistem neposredno iz elektronskega vezja pridobi sliko, prikazano na zaslonu pečice, ki jo ustrezno obdela in preveri, če je prikaz skladen z dokumentacijo. Za predobdelavo in segmentacijo slike je uporabljena odprtokodna knjižnica OpenCV [1], pri čemer je celoten sistem spisan v obliki skript v programskem jeziku Python. S pomočjo optičnega razpoznavalnika znakov preverjamo pravilnost prikazanega besedila, s pomočjo histograma orientiranih gradientov (HOG) pa pravilnost prikazanih sličic. Arhitektura našega sistema je nekako še najbolj podobna arhitekutri, ki jo opisujejo avtorji v delu [2].



Slika 1: Orodje za testiranje uporabniškega vmesnika samo z uporabo računalniškega vida (levo), in popolnoma avtomatizirano testiranje brez potrebe po operaterjevi interakciji (desno).

Na sliki 1 je prikazana shema sistema, kjer je viden potek interakcij med napravami in operaterjem. Ker smo se v tej fazi v glavnem posvečali avtomatizaciji s pomočjo računalniškega vida, sistem še ni v celoti avtomatiziran. Eden od ciljev za prihodnost je v model vključiti še kartezični robotski manipulator, ki bi upravljal z napravo namesto operaterja.

2. ZAJEM IN SEGMENTACIJA SLIKE

2.1 Zajem slike

Kot omenjeno, sliko pridobimo neposredno iz elektronskega vezja pečice. Osebni računalnik, na katerem teče testna aplikacija je s testirano napravo povezan preko protokola RS232, pri čemer je hitrost nastavljena na 2 Mbit/s, brez paritetnega bita in z enim stop bitom. Zajemamo sliko iz zaslona, velikega 480×272, ki je zapisana v formatu RGB565. Iz tega sledi, da je minimalen čas prenosa slike daljši od 1.3 s.

Ko sliko pridobimo, na testni napravi pretvorimo njen format v RGB888, ker je ta format bolj prikladen za analizo slike.

2.2 Segmentacija slike

Če želimo, da naš sistem za avtomatsko testiranje doseže dober rezultat, je potrebno pridobljeno sliko kvalitetno segmentirati. Segmentacijo izvršimo v petih korakih.

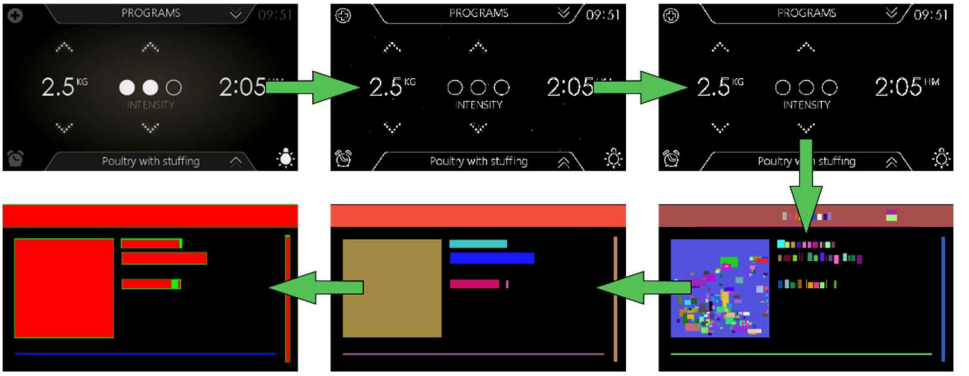
Za namen hitrejše in enostavnejše segmentacije sliko najprej pretvorimo v sivinsko. Sivinsko sliko tvorimo kot kombinacijo vseh treh kanalov RGB slike. Na dobljeni sivinski sliki nato s pomočjo Laplaceovega filtra in upragovljanja določimo robove. Za detekcijo robov smo testirali tudi Cannyjev detektor in Sobelov operator, a je pri teh prišlo do zamikanja robnih točk, zato slednjih pristopov v orodju nismo uporabili.

S pozitivnim pragom upragovljeno sliko, ki predstavlja robove s pozitivnim gradientom nato obdelamo tako, da jo pošljemo čez filter, ki iz nje odstrani majhna področja, ki so bila posledica šuma ozadja. To lahko storimo na dva načina. Prvi omogoča odstranjevanje premajhnih področij na podlagi minimalne površine, drugi pa na podlagi minimalne višine in širine. V obeh primerih filter za delovanje potrebuje označena sklenjena področja, ki predstavljajo robove. S pomočjo algoritma za označevanje povezanih komponent [3] poiščemo 8-povezljiva področja, kar nam omogoča lažje in hitrejše filtriranje. Nato na filtrirani sliki preostala področja združimo s pomočjo morfoloških operacij in uokvirjanja. Najprej preostala povezana področja uokvirimo in na sliki izvedemo dilacijo. Nato spet izvedemo uokvirjanje in na koncu še erozijo. Pri tem je zelo pomembna vellikost jedra, saj ne želimo med seboj povezati področij, znotraj katerih se nahajajo različne grafike.

Zadnji korak segmentacije je še na enak način poiskati področja, ki so temnejša od ozadja. Na enak način, kot je opisano zgoraj opravimo obdelavo slike, le da pri upragovljanju slike

robov uporabimo negativ filtrirane sivinske slike z Laplaceovim filtrom. Nato sliki s pozitivnimi in negativnimi robovi prileženo eno na drugo in od področij, ki se prekrivajo vedno izberemo manjše oz. objeto področje.

Vmesni rezultati posameznih korakov so prikazani na sliki 2. Prva vrstica slik prikazuje prve korake segmenacije zaslona z nastavitvami pečenja, druga pa končne na zaslonu z receptom. Na zadnji sliki je z modro barvo označeno področje na zaslonu, ki je temnejše od ozadja.



Slika 2: Vmesni rezultati posameznih korakov segmentacije. Začetni koraki na sliki prikazujejo segmentacijo zaslona z nastavitvami pečenja (prva vrstica), končni pa segmentacijo zaslona z receptom (druga vrstica).

3. METODE

Ustrezno segmentirano sliko zasona za tem ustrezno analiziramo. Najprej klasificiramo regije znotraj posameznih področij, nato pa jih ustrezno z vsebino še nadalje segmentiramo ali pa ustrezno preverimo pravilnost njihove vsebine.

3.1 Klasifikacija področij

Klasifikacijo področij se izvaja na podlagi vektorjev značilk, ki predstavljajo posamezno področje. Osnova za gradnjo vektorjev značilk so momenti kotur znotraj področij. Računamo 24 momentov [4] za vsako konturo, nato pa dobljene vektorje povprečimo. Povprečnega predstavnika področja nato še logaritmiramo in tako dobljeni vektor odvajamo. Za še boljšo ločljivost med področji momentom dodamo še nekaj značilk na osnovi analize barv znotraj področij ter dimenzij in položaja področja znotraj slike zaslona.

Področja na podlagi izpeljanih vektorjev značilk delimo v 7 razredov: ikone, slike, drsnike, razmejevalnike, glave/noge, področja z besedilom in področja, ki jih pri pregledovanju ignoriramo.

3.2 Šivanje slik

Večina receptov je predolga, da bi jih bilo mogoče prikazati na enem zaslonu, zato je celoten recept razširjen na več zaslonov, uporabnik pa si lahko celoten recept ogleda s pomočjo drsnika. Drsnik je tako tudi pogoj za zagon algoritmov za šivanje slik v našem sistemu. Algoritem od operaterja zahteva, da mu »pokaže« celotno sliko recepta. To naredi tako, da obdela izrezek zaslona z drsnikom in na podlagi tega določi, kateri del celotnega recepta je že videl. Ko ima shranjene sličice celotnega recepta, jih glede na njim lastno pozicijo drsnika uredi po vrsti od zgornje proti spodnji. Nato se za posamezni sosednji sliki zaslonov izvede iskanje najboljšega prekrivanja s pomočjo RMS razlike področja prekrivanja. Ko so določena optimalna prekrivanja za celotno množico posnetih sličic, se sličice na ta mesta prilepijo ena na drugo.

3.3 Razpoznavanje besedila

Za razpoznavanje besedila na zaslonu pečice je uporabljeno odprtokodno orodje za optično razpoznavanje besedila Tesseract-ocr [5]. Z njim preverimo vsa področja, ki jih je klasifikator razpoznal kot področja z besedilom.

Optični razpoznavalnik deluje tako, da najprej sliko po svoje predobdela, na njej poišče vrstice z besedilom ter k njim prileže kubični polinom. Za tem izvede detekcijo fiksne širine znakov in če ta ni detektirana, začne z iskanjem besed pri besedilih s proporcionalno širino znakov. To izvede tako, da najprej s pomočjo točk kandidatk razdeli povezane znake, za tem pa izvede asociiranje zlomljenih znakov na podlagi izpeljanih značilk in klasifikacije s pomočjo vpogledne tabele (ang. *look-up table*). Na koncu izvede še lingvistično analizo za besede, ki imajo nizko mero zaupanja. Klasifikator je adaptivnega tipa, kar pomeni, da se skozi celotno razpoznavo strani še dodatno uči in nato adaptiran še enkrat preveri celotno besedilo.

Orodje sicer omogoča učenje dodatnih slogov pisav, a smo se kljub temu odločili, da bo hitrejša pot do izboljšanih rezultatov s pomočjo predobdelave sličic z besedilom. Pri tem sta najpomembnejši operaciji izravnava širine razmakov med znaki in besedami in skaliranje besedila na ustrezno velikost.

3.4 Preverjanje slik in ikon

Ustreznost slik in ikon se preverja s pomočjo deskriptorjev na podlagi histogramov orientiranih gradientov [6]. Enostavnejša rešitev se v začetku zdi zgolj razlika med sličico na zaslonu in sličico, ki jo preverjamo iz urejene baze z dokumentacijo, a je celotno preverjanje mnogo bolj robustno, če se temu ozognemo. Zajete in obrezane sličice se namreč po velikosti lahko razlikujejo od originalov, so pa tudi zajete v barvnem formatu RGB565 in pretvorjene v format RGB888, v katerem so izvirniki. Pri pretvarjanju izvirnikov v format RGB565 se nekaj informacije izgubi in prav lahko bi prišlo do situacije, da bi pri preverjanju napravili napako.

4. REZULTATI

Predstavili bomo rezultate, kot jih v našem sistemu za testiranje uporabniškega vmesnika dosegajo posamezna metode.

4.1 Klasifikator področij

Za izbiro ustreznega klasifikatorja je bila najprej opravljena 20-delna križna validacija na setu 289 izrezkov področij. Testirali smo uspešnost klasifikatorjev na podlagi nevronskega omrežja, Gaussovih mešanic in metode podpornih vektorjev. Kljub temu, da je nevronsko omrežje doseglo najboljši rezultat, zaradi določenih pomanjkljivosti kasneje ni bilo uporabljeno. Za klasifikacijo smo uporabili klasifikator, ki temelji na metodi podpornih vektorjev in ima naučene ločilne meje za en razred proti enemu.

Nadalje je bil klasifikator testiran še na novih 246 različnih sličicah področij. Pri tem je napačno klasificiral 3.16% področij. Od tega je 62.5% področij z besedilom zamenjal za ikone, 25% ikon za besedilo in 12.5% področij s sliko za glavo/nogo.

4.2 Šivanje slik

Ob neustreznem prekrivanju sosednjih izrezkov zaslona lahko šivanje slik odpove. To se zgodi v primeru, da si bo prvih par vrstic slikovnih elementov dovolj podobnih, da bo ujemanje dodeljeno zadnji od njih, pravo ujemanje pa bo, zaradi rahle zatemnitve slike na zgornjem in spodnjem delu zaslona z receptom, prezrto.

4.3 Razpoznavalnik besedila

Razpoznavalnik besedila smo testirali na štirih recepturah, pri čemer smo vnesli napake v 40 besed. Pri generaciji napak smo pazili, da je bil njihov spekter čim bolj poln. Napake smo vnašali na začetek in na konec besede, menjali smo velike in male črke ter ločila in številke, zamenjali pa smo tudi nekaj celotnih besed.

Razpoznavalnik je pri tem napako naredil le v kontekstu »734 g«, kjer je »g« zamenjal za »9«.

4.4 Primerjalnik slik in ikon

Za testiranje primerjalnika slik in ikon smo uporabili testno množico desetih pravilno in desetih napačno umeščenih sličic, pri tem smo izbrali samo najbolj podobne si pare. Primerjalnik se je v vseh primerih odrezal 100% pravilno.

5. NADALJNJE DELO

V prihodnosti imamo namen sistem narediti povsem avtonomen in namesto operaterja uporabiti robotski manipulator. Ker se je za ozko grlo sistema izkazalo šivanje slik imamo

namen nadgraditi algoritme in uporabiti kompleksnejše iskanje ujemanja na podlagi deskriptorjev.

LITERATURA

1. G. Bradski, The OpenCV Library, *Dr. Dobb's Journal of Software Tools*, 2000.

2. Z. Yu, P. Xiao, Y.Wu, B. Liu in L.Wu, A novel automated gui testing echnology based on image recognition, *High Performance Computing and Communications;IEEE 14th International Conference on Smart City; IEEE 2nd International Conference on Data Science and Systems (HPCC/SmartCity/DSS), 2016 IEEE18th International Conference on Data Science and Systems*, str. 144–149, IEEE, 2016.

3. C. Grana, L. Baraldi in F. Bolelli, Optimized Block-based Connected Components Labeling with Decision Trees, *IEEE TRANSACTIONS ON IMAGE PROCESSING*,vol. 19, no. 6, str. 1596–1609, 2010.

4. C. Steger, On the Calculation of Moments of Polygons, *FGBV–96–04*, Forschungsgruppe Bildverstehen, Informatik IX, Technische Universität München, Orleansstrase 34, 81667 München, Germany, Avgust 1996.

5. R. Smith, An Overview of the Tesseract OCR Engine, *Proc. Ninth Int. Conference on Document Analysis and Recognition (ICDAR)*, str. 629–633, 2007.

6. N. Dalal, B. Triggs, Histograms of oriented gradients for human detection. *Computer Vision and Pattern Recognition, 2005, CVPR 2005, IEEE Computer Society Conference on*, vol. 1, str. 886-893, IEEE, 2005.