Univerza v Mariboru
Fakulteta za elektrotehniko,
računalništvo in informatiko

Jernej Mlakar

OVERITVENI SISTEMI ZA OSEBNO UPORABO

Diplomsko delo

Maribor, april 2013

Diplomsko delo univerzitetnega študijskega programa

OVERITVENI SISTEMI ZA OSEBNO UPORABO

Študent: Jernej Mlakar

Študijski program: UN ŠP – računalništvo in informatika

Smer: Informatika

Mentor: doc. dr. Marko Hölbl

Maribor, april 2013

Univerza v Mariboru
Fakulteta za elektrotehniko, računalništvo in informatiko

Številka: 93593769
Datum in kraj: 30. 05. 2012, Maribor

Na osnovi 330. člena Statuta Univerze v Mariboru (Ur. l. RS, št. 01/2010)
izdajam

SKLEP O DIPLOMSKEM DELU

1. **Jerneju Mlakarju,** študentu univerzitetnega študijskega programa RAČUNALNIŠTVO IN INFORMATIKA, smer Informatika, se dovoljuje izdelati diplomsko delo pri predmetu Dostopnost in zaščita podatkov.

2. **MENTOR: doc. dr. Marko Hölbl**

3. **Naslov diplomskega dela:**
OVERITVENI SISTEMI ZA OSEBNO RABO

4. **Naslov diplomskega dela v angleškem jeziku:**
AUTHENTICATION SYSTEMS FOR PERSONAL USE

5. Diplomsko delo je potrebno izdelati skladno z "Navodili za izdelavo diplomskega dela" in ga oddati v treh izvodih (dva trdo vezana izvoda in en v spiralo vezan izvod) ter en izvod elektronske verzije do 30. 05. 2013 v referatu za študentske zadeve.

Pravni pouk: Zoper ta sklep je možna pritožba na senat članice v roku 3 delovnih dni.

Dekan:

Obvestiti:
- kandidata,
- mentorja,
- odložiti v arhiv.

ZAHVALA

Za pomoč in vodenje pri izdelavi diplomskega dela se iskreno zahvaljujem mentorju doc. dr. Marku Hölblu.
Posebna zahvala velja staršem, ki so mi omogočili študij in mi vsa ta leta stali ob strani. Njihova pomoč in spodbuda sta mi omogočili uresničitev zastavljenih ciljev.
Rad bi se zahvalil še vsem prijateljem za vso razumevanje in podporo.

OVERITVENI SISTEMI ZA OSEBNO UPORABO

Ključne besede: overjanje, biometrija, prepoznavanje obrazov, prstni odtis.

UDK: 004.352.24:57.087.1(043.2)

Povzetek: *Diplomsko delo predstavlja pregled glavnih overitvenih pristopov za osebno uporabo. Najprej smo opisali osnovne koncepte overjanja ter obravnavali tri glavne načine overjanja: kaj nekdo ve, kaj nekdo je in kaj nekdo ima. Predstavili smo tudi prednosti in slabosti teh metod in raziskali, katera je najbolj primerna za osebno rabo. Posebej smo se osredotočili na prepoznavanje obrazov, kjer smo tudi opisali različne vrste algoritmov, ki jih sistemi uporabljajo. Na koncu pa smo implementirali metodo prepoznavanja obrazov s pomočjo algoritma Eigenobraz.*

AUTHENTICATION SYSTEMS FOR PERSONAL USE

Key words: authentication, biometrics, face recognition, fingerprint.

UDK: 004.352.24:57.087.1(043.2)

Abstract: *The thesis presents an overview of the main authentication systems for personal use. At first, we have intruduced the basic concepts of authentication and adressed that there are three main methods of authentication: what one knows, what one is and what one has. We have identified the advantages and disadvantages of these methods and researched which one would be the most suitable for personal use. In particular, we had focused on facial recognition, where we also described the various types of algorithms used by the systems. At the end, we have implemented method for face recognition with the help of algorithm Eigenface.*

KAZALO VSEBINE

KAZALO SLIK

KAZALO TABEL

UPORABLJENE KRATICE

PIN	Personal Identification Number
MIT	Massachusetts Institute of Technology
FAR	False Acceptance Rate
FRR	False Reject Rate
USB	Universal Serial Bus
RFID	Radio Frequency Identification
API	Application Programming Interface
PCA	Principal Component Analysis
UML	Unified Modeling Language
DNK	Deoksiribonukleinska kislina
NIST	National Institute of Standards and Technology
GUI	Graphical User Interface
JMF	Java Media Framework

1 UVOD

V modernem svetu je zanesljivo varovanje ena od najbolj zaželenih dobrin. Brez zanesljivega varovanja so ogrožene številne vsakdanje aktivnosti, kot na primer zaščita osebnih računalnikov, mobilnih telefonov, internetnih storitev, preprečevanje tatvin, poneverb pri finančnih transakcijah ipd. Ljudje se nenehno srečujemo z raznimi vdori ali nepooblaščenimi vstopi v računalniške sisteme, kjer nam lahko ukradejo ali zlorabijo podatke in dokumente, s čimer lahko še kako škodujejo posamezniku oz. podjetju. Pa naj bo to prijatelj, ki želi pregledati vaš Facebook račun, ali nepridipravi, ki ti želi ukrasti geslo do spletne banke. Prav zaradi tega je v današnjem času množične uporabe računalnikov zelo pomembno, da so ti podatki varni pred nepooblaščeno uporabo in zlorabo.

1.1 Opredelitev problema

Overjanje (ang. authentication) uporabnikov je osnovna in ena izmed najbolj pomembnih vrst zaščite pred nepooblaščenim dostopom in posledično nepridipravi. Overjanje pomeni, da se preverja, ali je nekdo res tisti, za katerega se izdaja (človek ali sistem). V ta namen obstajajo različne metode, ki jih delimo na tri glavne skupine: kaj nekdo ima, kaj nekdo ve in kaj nekdo je. V diplomski nalogi se bomo osredotočili na overitvene sisteme za osebno uporabo, se pravi sisteme, ki jih lahko uporablja vsak posameznik za zaščito svojih podatkov in jih lahko najdemo na vsakem osebnem računalniku, posebej pa nas bodo zanimali sistemi kaj nekdo je, kamor spada tudi področje prepoznavanja obrazov.

1.2 Predstavitev ciljev diplomskega dela

Glavni cilji diplomskega dela so, da podrobno preučimo različne overitvene pristope, metode in sisteme za osebno uporabo, opredelimo njihove prednosti in slabosti, ter da s pomočjo brezplačne programske opreme in brezplačnimi ter odprto-kodnimi rešitvami izdelamo aplikacijo v okolju Java, ki bo s pomočjo spletne kamere znala prepoznati obraz in bo na tak način overjala uporabnika. Aplikacija bo tako omogočala dostop samo pooblaščenim osebam in bo primerna za vsakdanjo uporabo.

1.3 Predstavitev predpostavk in omejitev diplomskega dela

V aplikaciji za prepoznavanje obrazov bomo uporabljali odprto-kodne rešitve in že obstoječe knjižice, ki jih bomo vključili v izdelano rešitev, saj so algoritmi za prepoznavanje obrazov zapleteni in je smiselno uporabiti že obstoječe rešitve.

Tako bomo na začetku opisali kaj je overjanje in predstavili osnovne koncepte. Nato bomo podrobno pregledali glavne metode overjanja, jih opisali in predstavili njihove prednosti in slabosti. Na koncu bomo implementirali še rešitev v Javi, in sicer prepoznavanje obrazov s pomočjo algoritma Eigenobraz (ang. Eigenface).

V diplomsko nalogo ne bomo vključili overitvenih metod, ki se jih uporablja v podjetjih, bankah in drugih ustanovah, ker za osebno uporabo niso primerne. V omenjeno skupino sodijo prepoznavanje očesne mrežnice, geometrija roke, primerjava DNK (deoksiribonukleinska kislina) ipd. Tako se bomo osredotočili le na izbrane, najbolj razširjene metode.

1.4 Predstavitev osnovnih metod dela

Pri preučevanju predstavljene problematike bomo uporabili naslednje metode in tehnike:

- Študijo virov in literature, kjer bomo preučili domačo in tujo literaturo.
- Študij že obstoječih rešitev, kjer bomo podrobno preučili rešitve, katere bi lahko uporabili na našem primeru.
- Načrtovanje, izdelava in testiranje programske rešitve, kjer bomo implementirali delujoč program za prepoznavanje obrazov.

2 OVERJANJE

Kamorkoli gremo, nenehno identificiramo in overjamo kogarkoli že vidimo. Na primer, prijatelja bi v nakupovalnem središču identificiral tako, da bi iskal značilnosti, ki so ti znane. Ali je oseba moški ali ženska? Kakšne barve so njegove oči, lasje? Ali nosi poznan pulover? Ko bo ta prijatelj videl, da ga gledate, vas bo verjetno pozdravil po imenu. Z uporabo vašega prejšnjega znanja ste overili to osebo kot vašega prijatelja. Ali pa ste lahko prepričani, da imate pravo osebo? Verjetno ne 100 %, ampak smo ublažili tveganje, tako da smo prišli do zadovoljivega nivoja udobja z njegovo/njeno identiteto.

Zgornji scenarij je zelo podoben scenariju računalnika, kadar hočejo uporabniki dostop. Na računalnik bi naj dostopali samo legitimirani uporabniki. Da pa zvemo, ali je uporabnik legitimiran ali ne, je računalnik priskrbljen z uporabniškim imenom in metodo overitve.

Overjanje je, v smislu informacijske varnosti, nabor metod, ki jih uporabimo, da vzpostavimo zahtevo po identiteti kot pravilno. Poudariti je treba, da overjanje samo vzpostavi to zahtevo kot pravilno in nam ne pove, kaj lahko uporabnik nato dela, katere pravice ima – za to je zadolženo drugo opravilo imenovano avtorizacija.

V smislu overjanja obstaja mnogo kategorij metod, ki jih lahko uporabljamo. Na vsako kategorijo se sklicujemo kot faktor, znotraj katerega so različne metode. Ko poskušamo overiti zahtevo po identiteti, je bolje, da uporabimo čim več faktorjev, saj bodo tako rezultati natančnejši [2].

Obstajajo trije glavni načini overitve oz. trije faktorji:

1. Overjanje na podlagi kaj veš.
2. Overjanje na podlagi kaj imaš.
3. Overjanje na podlagi kaj si.

Tem overjanjem pogosto rečemo trije stebri overitve. Lahko so uporabljeni posamezno ali ločeno, za še močnejšo overitev [4].

Multifaktorno overjanje uporablja enega ali več zgoraj omenjenih faktorjev. V večini primerov tej metodi pravimo kar dvofaktorna metoda, saj po navadi uporabljamo dva faktorja. Pogost primer multifaktornega overjanje je bančni avtomat. Imamo namreč nekaj, kar vemo, našo PIN (ang. Personal Identification Number) kodo, in nekaj, kar imamo − našo bančno kartico. Odvisno od faktorjev, ki jih izberemo, lahko sestavimo močnejše ali šibkejše multifaktorne overitvene metode, vendar veliko metod ni praktičnih za implementacijo. Na primer, če bi namesto PIN kode uporabili metodo DNK. To bi bila zelo močna metoda za overjanje, ampak predraga in nepraktična.

Medsebojno overjanje se nanaša na overitvene mehanizme, kjer morata obe stranki overiti druga drugo. V standardnih overitvenih procesih, ki so enosmerni, se mora uporabnik overiti strežniku, da mu dokaže, da lahko dostopa do sredstev, ki jih strežnik ponuja. V medsebojnem overjanju pa se mora poleg uporabnika overiti tudi strežnik. Takšna vrsta overjanja je pogosta pri digitalnih certifikatih. Medsebojno overjanje lahko uporabimo tudi v kombinaciji z multifaktornim overjanjem, pri čemer se le to uporablja le na uporabnikovi strani [2].

2.1 Overjanje na podlagi kaj veš

Nekaj, kar veš, je zelo pogost overitveni faktor. Pod to metodo spadajo oblike, pri katerih je potrebna samo naša pamet oziroma znanje. Za overjanje uporabljamo gesla, PIN kode, varnostna vprašanja ipd. Overjanje na podlagi kaj veš je najbolj pogosta oblika overjanja za osebno uporabo. Ne potrebujemo namreč nobene druge opreme, kot so kamere, razni bralniki ali podobno. Vse, kar potrebujemo, je naš računalnik in tipkovnica. V nadaljevanju bomo opisali osnovne metode overjanja na podlagi kaj veš, kot so uporabniško ime in geslo, fraze, PIN kode in enkratna gesla ter predstavili njihove prednosti, omejitve in slabosti.

UPORABNIŠKO IME IN GESLO

Geslo je skrivna beseda ali vrsta znakov, ki je uporabljena za identifikacijo uporabnika, s čimer dokažemo identiteto le tega, ali za dostop do sistema, informacij ali drugih virov in je zaradi svoje enostavnosti najbolj priljubljen mehanizem za overjanje uporabnikov.

Pojavljati so se začela že v rimskih časih, kjer so vsak večer stražarji za dostop prejeli geslo, ki je bilo zapisano na leseni deski. Prva računalniška gesla pa so se začela uporabljati leta 1961 na MIT-u (ang. Massachusetts Institute of Technology). Poleg osnovnega gesla poznamo tudi nekaj izpeljank. Najbolj znana je prav gotovo tako imenovana PIN koda, ki jo uporabljamo predvsem pri bančnih karticah in pri odklepanju telefonov. Poznamo pa tudi razna grafična gesla, matrična gesla ipd. [8].

Gesla uporablja večina od nas, ki redno uporabljamo računalnik. V kombinaciji z uporabniškim imenom nam bo geslo po navadi dovolilo dostop do računalniškega sistema, telefona ali podobne naprave. Gesla, čeprav nudijo overjanja samo na en faktor, predstavljajo relativno visok nivo varnosti.

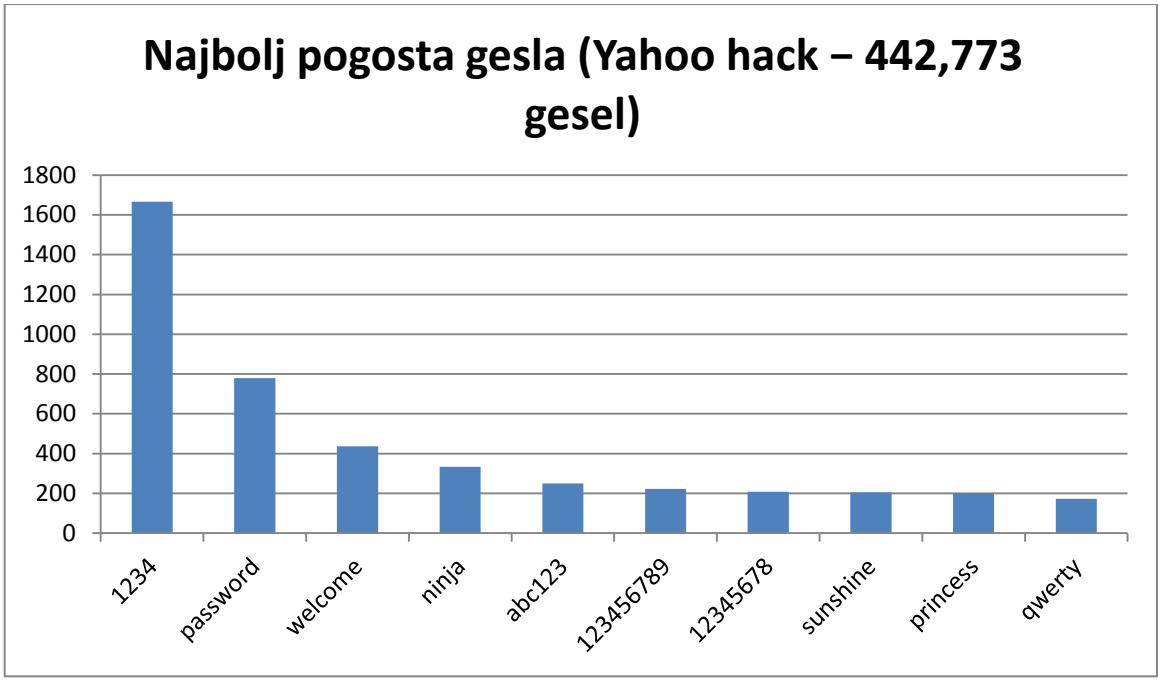
Kadar opisujemo geslo kot močno, ne podamo takojšnje točne informacije o čem govorimo. Boljši izraz bi mogoče bil kompleksno, ker s tem predstavimo pomemben koncept pri gradnji in izbiri gesel. Če sestavimo geslo, ki ima vse črke majhne in je dolgo osem znakov, lahko z orodjem za odkrivanje gesel le to odkrijemo v minuti ali dveh, če imamo dovolj zmogljiv računalnik, na katerem se bo to orodje poganjalo. Če pa uporabimo isto geslo z osmimi znaki, vendar z velikimi in malimi črkami, bo to orodje potrebovalo približno šest dni. Če pa v geslo dodamo še številke, lahko traja tudi več kot 25 dni. Torej, če uporabljamo priporočeno metodo za gradnjo gesel, bomo uporabili velike črke, male črke, številke, simbole ipd. Tako bomo, kljub temu da si ga bo potencialno težje zapomnili, kot na primer $sU&qw!3, imeli geslo, za katero bo potrebno več let, da se ga odkrije s povprečnimi metodami.

Poleg izbire močnih gesel pa moramo biti pozorni tudi na prakso dobre higiene gesel. Problem z močnimi gesli je namreč, da si jih je težko zapomniti, kar pa velikokrat privede do tega, da si geslo zapišemo na primer pod tipkovnico ali pod ekran. To pa seveda popolnoma izniči sam namen gesel, če kdo brska po naši mizi in to geslo najde.

Še en problem gesel je sinhronizacija le teh – na kratko, da uporabljamo isto geslo vsepovsod. Če uporabljamo isto geslo za naš račun spletne pošte, Facebook, forume in povsod drugod, moramo paziti, da nam ne odkrijejo kakšnega, kajti s tem ogrožamo vse sisteme, kjer uporabljamo isto geslo. Vse kar mora napadalec narediti je, da pogleda naše

uporabniško ime in najde mesta, kjer je le to uporabljeno in tako začne uporabljati naše privzeto geslo. Ko pa je enkrat napadalec v našem računu spletne pošte, je igre že konec [2].

Obstaja kar nekaj načinov za pridobivanje in krajo gesel. Najbolj znani sta napad z grobo silo (ang. brute force attack), kjer preizkusimo vse možne kombinacije znakov in če se katera ujema, imamo naše geslo in pa napad s slovarjem (ang. dictionary attack), kjer imamo datoteko oziroma program, ki vsebuje najbolj znana gesla in razne variacije le teh. Če osebo dovolj dobro poznamo, lahko gesla tudi preprosto ugibamo, ali pa jih iščemo v raznih datotekah in programih [8]. Slika 1 prikazuje najbolj pogosta gesla, s katerimi uporabniki zavarujejo svoje račune.



Slika 1: Najbolj pogosta gesla [15]

Nekaj priporočil, ki se jih moramo držati pri izbiri gesel:

- ne zaupamo jih nikomur,
- izbiramo čim bolj kompleksna gesla,
- ne uporabljamo raznih imen, znanih fraz, pregovorov ipd.,
- nikamor jih ne zapisujemo,

- redno jih menjamo,
- ne uporabljamo enakega gesla za vse račune,
- pazimo, da nas med vpisom gesla nihče ne opazuje.

FRAZA

Fraza (ang. passphrase) je zaporedje besed ali drugega teksta za kontroliranje dostopa do računalniškega sistema, programa ali podatkov. Fraza je po uporabi podobna geslom, ampak je po navadi daljši za dodatno varnost. Fraze so pogosto uporabljene za kontroliranje dostopa in operacij kriptografskih programov in sistemov. Moderni koncept fraze bi naj leta 1982 izumil Sigmund N. Porter.

Če najdemo besede, ki jih uporabimo v frazi v slovarju – posebej takšnem, ki je lahko kot elektronski vnos v kakšnem programu – potem je ta fraza veliko bolj ranljiva na napad s slovarjem. To je posebej velik problem, če je v knjigi fraz in navedkov najdena celotna fraza. Potreben trud (v času in denarju) lahko naredi to nepraktično, še posebej, če uporabimo dovolj besed in jih čim bolj naključno izberemo. Število kombinacij, ki bi jih morali testirati pod pravimi pogoji, naredi napad s slovarjem tako težak, da je praktično neizvedljiv. Že samo, če izberemo eno besedo, ki je ne moremo najti v slovarju, ogromno poveča moč naše fraze [19].

Fraze se razlikujejo od gesel. Geslo je po navadi kratko – šest do deset znakov. Takšna gesla so lahko zadovoljiva za razne aplikacije, kot so prijava v računalnik ipd., ampak gesla po navadi niso dovolj varna kot ključi za samostojne varnostne sisteme. Fraze so po navadi močnejše in so očitno boljša izbira v teh primerih. Kot prvo, so po navadi veliko daljša (od 20 do 30 znakov), tako da odpade napad s silo. Drugič, če so dobro izbrana, jih ne bomo našli v nobenem slovarju, tako da skoraj ni možnosti za napad s slovarjem. In nazadnje, lahko jih sestavimo tako, da si jih lažje zapomnimo kot gesla, brez da si jih zapisujemo, kar zmanjšuje možnost, da jih kdo najde [18].

Tipični nasveti za izbiro dobre geslovne fraze:

- dovolj dolga, da jo težko uganemo,

- ne sme biti navedek iz literature, znanih oseb ipd.,
- mora biti težko uganljiva po intuiciji – tudi za nekoga, ki te dobro pozna,
- da je lahka za zapomniti in natipkati,
- za boljšo varnost lahko nanesemo lahko zapomnljivo kodiranje,
- ne smemo ga uporabljati na več straneh hkrati.

Ena izmed metod za izbiro močne fraze je uporaba kocke, da izberemo naključne besede iz spiska. Čeprav lahko takšen spisek besed krši pravilo, da ne sme biti iz slovarja, pa celotna varnost sloni na tem, da obstaja ogromno število besed na seznamu. Na primer, če imamo 7776 besed in jih naključno izberemo šest, potem je 7776^2 = 221073919720733357899776 kombinacij.

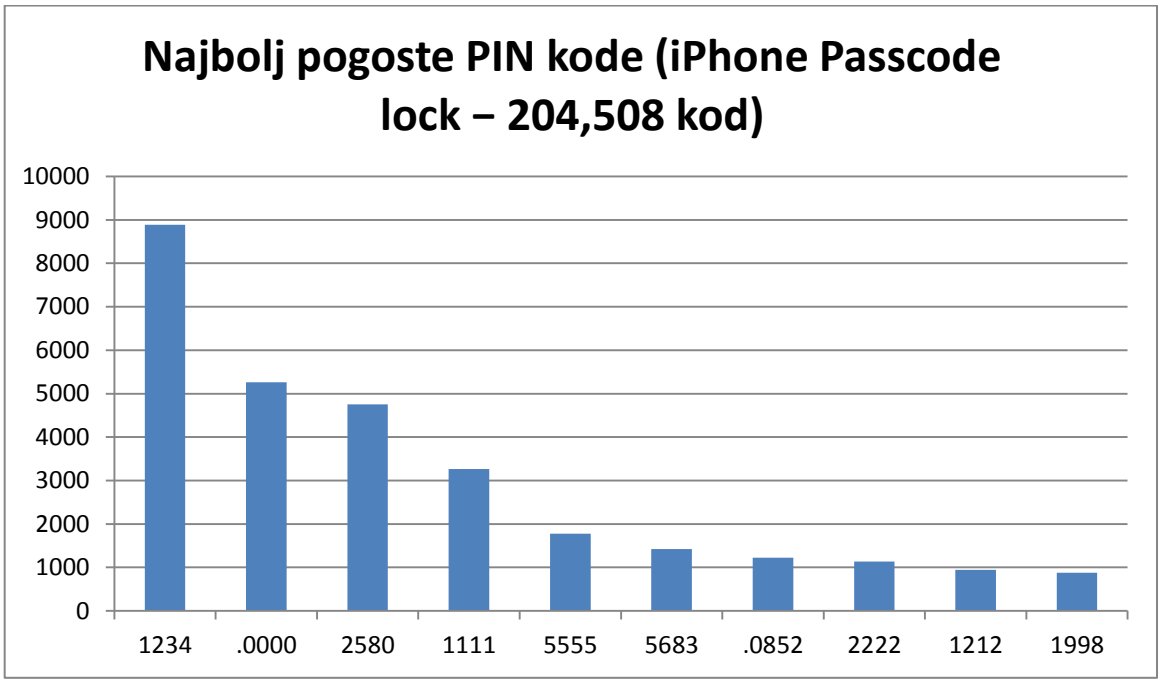
Drugi način za dobro geslovno frazo je, da izberemo dve frazi, spremenimo eno v akronim in jo vključimo v drugo. Na primer − Hitra rjava lisica skoči čez lenega psa postane hrlsčzp. To pa sedaj vstavimo v − Sedaj je čas, da vsi dobri možje pomagajo državi, in tako nastane − Sedaj je čas, da vsi dobri hrlsčlp pomagajo državi [17].

PIN KODE

Osebna identifikacijska številka oz. PIN koda je skrivno geslo sestavljeno iz številk, ki si ga delita uporabnik in sistem, kateri je uporabljen, da overi uporabnika. Po navadi mora uporabnik podati nezaupno uporabniško identifikacijo ali žeton in zaupno PIN kodo, da lahko dostopi do sistema. Uporabnik ima omogočen dostop le, če se vpisana številka ujema s številko v sistemu. PIN kode so najpogosteje uporabljene na bankomatih in pri plačilih z bančnimi oz. kreditnimi karticami [20].

Koncept PIN kod izvira od izumitelja bankomata, John Stepherd-Barrona. Nekega dne, leta 1967, ko je razmišljal o tem, kako bi lahko banke bolj učinkovito izdajale denar, se mu je posvetilo, da bi bil najboljši način v obliki modela avtomatov. Za overjanje si je Stepherd-Barron najprej zamislil šest-številčno kodo, njegova žena pa je hotela štiri-številčno, kar je tudi postala najbolj razširjena dolžina. Mednarodni standard za PIN dovoljuje kode od štiri do dvanajst številk, vendar zaradi uporabnih razlogov PIN koda naj ne bi presegala šest številk [21].

Poleg bančnih avtomatov je zelo pogost način uporabe PIN kode tudi pri mobilnih telefonih, kjer je koda shranjena na SIM kartici. Na sliki 2 lahko vidimo seznam najbolj pogostih PIN kod.



Slika 2: Najbolj pogoste PIN kode [16]

ENKRATNO GESLO

Enkratno geslo je geslo, ki je veljavno samo eno sejo oziroma transakcijo. Ta gesla se izognejo velikemu številu pomanjkljivosti, ki so povezane s tradicionalnimi (statičnimi) gesli. Največja pomanjkljivost, ki jo odpravljajo enkratna gesla, je možnost za ponovitveni napad (ang. replay attack). To pomeni, da potencialni vsiljivec, ki bo uspel posneti enkratno geslo, ki je že bilo uporabljeno, le tega ne bo mogel zlorabiti, ker ne bo več veljavno. Slabost enkratnih gesel pa je, da si jih je težko zapomniti, zato potrebujejo dodatno tehnologijo za svoje delovanje.

Generirni algoritem enkratnih gesel po navadi uporablja principe psevdonaključnosti in naključnosti. To je potrebno, ker bi bilo drugače lahko ugotoviti naslednje enkratno geslo, in sicer z opazovanjem prejšnjih. Obstajajo različni načini za generiranje enkratnih gesel:

- Temelječi na časovni-sinhronizaciji med overitvenim strežnikom in uporabnikom, ki poda geslo (varnostni žetoni).
- Uporaba matematičnega algoritma, ki generira novo geslo na podlagi prejšnjih gesel (uporaba funkcije f … f(s) → f(f(s)) → f(f(f(s)))).
- Uporaba matematičnega algoritma, kjer novo geslo temelji na izzivu (npr. naključno število, ki ga izbere overitveni strežnik) in/ali števcu.

Obstaja tudi veliko različnih načinov, preko katerih se uporabnik zaveda, katero enkratno geslo mora uporabiti. Nekateri sistemi uporabljajo posebne elektronske žetone, ki jih uporabnik nosi s sabo in generirajo enkratna gesla, ter jih prikazujejo na majhnem zaslonu. Drugi sistemi so sestavljeni iz programske opreme, ki teče na uporabnikovem mobilnem telefonu. Spet nekateri generirajo enkratna gesla na strežniku in jih pošiljajo uporabniku po npr. tekstovnih sporočilih. Poznamo pa tudi takšne, kjer so gesla natisnjena na papirju in jih mora uporabnik nositi s sabo [22].

2.2 Overjanje na podlagi kaj si

Overjanje na podlagi kaj si, z drugimi besedami rečemo tudi biometrično overjanje. Ti načini overjanja sicer niso tako pogosti kot gesla ali PIN kode, vendar so še vedno zelo razširjeni predvsem, kadar želimo še večji nivo varnosti in delujejo skupaj z overjanjem na podlagi kaj veš.

Biometrični način identifikacije posameznika pomeni individualno obravnavanje človekovih fizičnih lastnosti ali značilnosti obnašanja, in zajem ter shranjevanje tega vzorca (imenovan tudi živi vzorec) v standardni podatkovni obliki. Ta vzorec se v postopku identifikacije primerja z vzorcem (imenovan tudi shranjeni vzorec ali podpis), ki temelji na istih značilnostih in je shranjen v varnostnem sistemu. Primerjava obeh vzorcev potrdi ali zavrže identiteto posameznika.

Pri tovrstni identifikaciji je pozornost usmerjena na majhno število fizičnih značilnosti, ki so lastne izključno eni osebi. Mednje spadajo barva glasu, način hoje, značilnosti obraza, vzorec šarenice, odtis dlani in prstov (DNK pri tem ni vključena, ker je vzorčenje DNK

počasno in pomeni poseg v telo). Najbolj dozorel, napreden in najrazvitejši način je preverjanje na osnovi prstnih odtisov [23].

V diplomski nalogi bomo največ prostora namenili biometričnemu overjanju, saj smo ga zavoljo praktičnega dela, kjer smo implementirali metodo za prepoznavanje obrazov, tudi najbolj spoznali. Razložili bomo, kaj sta odobritvena identifikacija napačne osebe ali FAR (ang. False Acceptance Rate) in neodobrena identifikacija pravilne osebe ali FRR (ang. False Reject Rate), ter opisali prednosti in slabosti glavnih metod, kot so overjanje s prstnimi odtisi, prepoznavanje obrazov, dinamično preverjanje podpisa in dinamika tipkanja. Metode, ki pa za osebno uporabo niso primerne, bomo izpustili.

2.2.1 Biometrija

Biometrično prepoznavanje (ali biometrija) se nanaša na avtomatizirano prepoznavanje posameznikov, na podlagi njihovih fizičnih in vedenjskih značilnosti. Biometrija tvori močno vez med osebo in njegovo identiteto, kajti biometričnih lastnosti ne moremo zlahka deliti, izgubiti ali ponarediti. Ravno zato je biometrično overjanje boljše in bolj odporno na socialne inženirske napade kot ostale metode overjanja, npr. žetoni ali gesla. Ker biometrično overjanje potrebuje, da je uporabnik prisoten v času overjanja, lahko odvrne uporabnika, da dela lažne trditve zavračanja. Še več, le biometrija lahko omogoča negativne identifikacijske funkcionalnosti, kjer lahko ugotovimo, če je neka oseba vpisana v sistem, čeprav trdi, da ni. Zaradi teh karakteristik je biometrično overjanje pozdravljeno kot naravna, zanesljiva in je nezamenljiva komponenta kakršnega koli identifikacijskega sistema.

Biometrični sistem je računalniški sistem, ki implementira algoritme biometričnega prepoznavanja. Tipični biometrični sistem je sestavljen iz modulov zaznavanja, pridobivanja značilnosti in ujemanja. Biometrični senzorji ujamejo ali slikajo biometrične značilnosti posameznika in ustvarijo njegovo digitalno predstavitev. Po navadi je opravljeno še preverjanje kakovosti, ki zagotovi, da so vzorci lahko zanesljivo procesirani za nadaljnjo obdelavo. Modul pridobivanja značilnosti zavrže nepotrebne in odvečne informacije iz pridobljenih vzorcev, in pridobi samo potrebne informacije, uporabljene za

potrebe ujemanja. Med ujemanjem pa je pridobljen biometrični vzorec primerjan z referenčnimi vzorci, ki so že shranjeni v bazi.

Na splošno ima biometrični sistem dve fazi operacij: vpis in prepoznavanje. Vpis se nanaša na fazo, pri kateri sistem shrani biometrične informacije osebe v bazo. Te informacije so lahko v obliki predloge (lastnosti pridobljene iz biometričnega vzorca ali parametri matematičnega modela, ki najbolje predstavijo pridobljene lastnosti) ali kar sam biometrični vzorec (npr. slika obraza ali prstnega odtisa). V veliko aplikacijah se zraven biometričnega vzorca shrani še nekaj informacij o osebi (ime, ID številka …). Če pa ni nobene informacije o osebi (npr. prstni odtis iz kraja zločina), sistem avtomatsko dodeli oznako, po kateri se kasneje sklicujemo. V fazi prepoznavanja sistem prebere uporabnikove biometrične značilnosti, pridobi lastnosti in jih primerja z referenco biometričnih informacij, ki so že v bazi. Visok rezultat podobnosti med obema podatkoma pomeni, da je uporabnik overjen oziroma identificiran.

Biometrični sistemi prepoznavanja tipično nudijo dve različni vrsti funkcionalnosti:

- Verifikacija (»Ali je oseba tista, za katero trdi da je?«). Na primer, oseba trdi, da je Janez Novak in ponudi svoj prstni odtis; sistem nato ali sprejme ali pa zavrne zahtevo, na podlagi primerjave vzorcev. Mnogo komercialnih aplikacij, od fizičnih (vstop v stavbo) in logičnih (prijava v računalnik) nadzorov vstopa, do transakcij na bankomatih in podobne, je primer uporabe sistema za potrebe verifikacije.
- Identifikacija (»Ali je ta oseba v bazi?«). Ko sistem dobi vzorec, nato le tega primerja z ogromnim številom ostalih vzorcev v bazi. Nato pa sta lahko dva scenarija. Pozitiven, kar pomeni, da ga sistem prepozna, ali negativen, se pravi, ga ne prepozna. Primeri takšnih sistemov za identifikacijo so nadzor vstopa brez PIN kod, izdaja vozniških dovoljen ipd.

Čeprav se zdi biometrija kot očitna tehnologija za robustno osebno overjanje in je že uspešno vpeljana v mnogo trgov, to še vedno ni neprebojna metoda avtomatskega prepoznavanja oseb. Biometrija bi potrebovala boljše rešitve v treh temeljnih ovirah: uspešnost prepoznavanja, varnost sistema in vprašanja zasebnosti.

V teoriji od biometričnega sistema pričakujemo, da ko mu predstavimo vzorec, se bo na podlagi tega pravilno odločil, vendar pa v praksi lahko naredi sistem dva osnovna tipa napak. FAR − sistem nepravilno napove uspešno ujemanje med vzorcem in predlogo v bazi. Se pravi osebi, ki ni prava in ni overjena, ima dovoljen dostop v sistem. FRR − biometrični sistem napačno napove neujemanje med vzorcem in predlogo v bazi. V tem primeru ima oseba, ki bi morala imeti dostop, le tega zavrnjenega.

Obstajajo trije glavni razlogi za napačne odločitve biometričnega sistema:

- Zelo podobna biometrična vzorca dveh različnih oseb.
- Slaba kvaliteta vzorca ali slaba interakcija uporabnika s senzorjem, kar rezultira v velikih variacijah biometričnih značilnosti istega uporabnika.
- Sprememba v biometričnih značilnostih uporabnika od shrambe vzorca in do danes (staranje) [3].

2.2.2 Biometrija na podlagi fizičnih karakteristik

Pri biometriji na podlagi fizičnih karakteristik se osredotočamo na značilnosti našega telesa, ki so unikatna samo nam. Sem spadajo prstni odtisi, značilnosti našega obraza, naš glas ipd.

OVERJANJE S POMOČJO PRSTNIH ODTISOV

Prstni odtis je odtis, ki ga naredi koža blazinice človeškega prsta. Še nikoli niso našli dveh ljudi, niti enojajčnih dvojčkov, z enakim prstnim odtisom, zato se prstni odtisi uporabljajo za dokazovanje istovetnosti osebe. To je najbolj napreden, dozorel in najrazvitejši način za biometrično overjanje. Najbolj pogost je na področju kriminalistike, za osebno uporabo pa se je najbolj razširil kot bralnik na prenosnem računalniku.

Prstni odtisi so trenutno najzanesljivejši način preverjanja identitete oseb, kar nam kažejo tudi dolgoletne raziskave in izkušnje. Še vedno namreč veljajo kot nedvomen dokaz identitete posameznika in večina današnjih biometričnih sistemov temelji ravno na prepoznavanju prstnih odtisov.

Fiziološko je prstni odtis konfiguracija grebenov s porami, ki jih delijo doline. Ležijo na ožilju, neposredno pod kožo. Morfologija (oblika) prstnega odtisa je povezana s specifičnimi toplotnimi in električnimi značilnostmi kože. To pomeni, da lahko za zajem podobe prstnega odtisa uporabimo svetlobo, toploto ali električno napetost (ali kombinacijo vseh). Prstni odtis nastane že pri razvoju zarodka in se s starostjo osebe ne spremeni, temveč raste v svoji prvotni obliki in po končani rasti osebe ostane nespremenjen. Tudi, če je poškodovan, se obnovi v prvotno obliko.

Majhen odstotek populacije (npr. rudarji in majhno število glasbenikov) ima prstne odtise zaradi stalnega trenja, poškodovane. V razvitih državah je ta odstotek zanemarljiv in ne predstavlja omembe vredne težave za identifikacijske sisteme, ki temeljijo na prstnih odtisih.

Obstaja več algoritemskih metod, za zajem značilnosti vzorca prstnega odtisa. Najbolj razširjene metode temeljijo na prepoznavanju vzorca ali izvlečku minucij (ang. minutiae). V primeru algoritmov, ki temeljijo na minucijah, je prstni odtis sestavljen iz grobih značilnosti, kot so zanke, loki in zasuki, ter drobnih značilnosti (minucije), kot so predvsem bifurkacije – razdelitve (ang. bifurcation), zaključki grebenov in delte (združevanja v obliki črke Y). Prstni odtis ima med 30 in 40 minucij. Značilnost vsake od njih je tip (bifurkacija, delta ali zaključek), položaj (koordinate), in usmerjenost (orientacija). Skupek značilnosti minucij lahko da predlogo za prstni odtis. Če so značilnosti natančno zajete, je možnost, da bi imela dva prstna odtisa enake značilnosti, izjemno nizka.

Elektronski zajem slik in algoritmi za razpoznavanje vzorcev so danes dovolj razviti, da vzorec prstnega odtisa avtomatsko obdelajo in shranijo. Za ta postopek v več primerih obstajajo tudi standardi. Ti standardi obstajajo za vzorce, ki temeljijo na minucijah. Najbolj uporabljen je standard, ki ga v ZDA predpisuje NIST (ang. National Institute of Standards and Technology). Oklepanje standardov je kljub temu ovira za fleksibilnost razvijalcev algoritmov in omejuje njihovo intelektualno lastnino. Tako naletimo na razkorak med natančnostjo in hitrostjo, ko gre za standardizacijo procesa in oklepanjem standarda.

Tehnologija bralnikov prstnih odtisov

Za zajem prstnih odtisov so na trgu številne tehnologije. Najbolj znane so: optična, kapacitivna, radijska, tehnologija tlaka, mikro-elektro-mehanična in toplotna.

- Optična: Za odčitavanje podobe prstnega odtisa so uporabljene digitalne kamere.
- Kapacitivna: Ko položimo prst na množico točk, občutljivih na spremembe električne napetosti, se razlike v napetosti med dolinami (pretežno zrak) in grebeni (pretežno voda) zapišejo kot slika.
- Radijska: Če prst obsevamo z radijskimi valovi nizke intenzitete, deluje kot oddajnik in razlike v oddaljenosti med dolinami in grebeni so razpoznavne kot množica ustrezno usmerjenih točkovnih anten.
- Tlačna: Točkovna množica, občutljiva na tlak, je sestavljena iz piezo-električnih elementov, ki zajemajo vzorec grebenov, ko nanjo položimo prstni odtis.
- Mikro-elektro-mehanična: Mikro-elektro-mehanična metoda je ostala na stopnji med raziskavo in razvojem ter med uporabo v različnih aplikacijah.
- Termična: Piro-električni material lahko razliko v temperaturi spremeni v določeno napetost [23].

Nevarnosti

Tveganja pri uporabah novih tehnologij moramo vedno preveriti. Zapomniti si moramo razmerje med povečano varnostjo in zmanjšanim udobjem za stranke. Če povečamo varnost, se bo zmanjšalo udobje. Čeprav tukaj lahko nastane tudi paradoks, kajti npr. biometrične naprave so kompleksnejše od up. imena in gesla, a vendar so lahko za uporabnika bolj priročne, saj si le ta ne potrebuje zapomniti svojega gesla. To pa pomeni, da so tudi bolj varne.

- **Napad na fizični prst:** To je vrsta napada, ki najbolj odmeva. Vsi smo že videli v filmih, ko zločinec ponaredi žrtvin prstni odtis. To se po navadi naredi s kopijo odtisa, ali tudi z odstranitvijo prsta samega.

- **Uporaba artefaktov (ang. artifacts):** Kot smo videli pri napadih na fizični prst, so odtisi oz. artefakti, ki jih pustimo za seboj, lahko izkoriščeni. Ta vrsta napada se osredotoča na odtise, ki jih pustimo na sami napravi za branje. Popolnoma logično je, da ko se dotaknemo naprave, za seboj pustimo neko sled in ravno to sled lahko izkoristimo, da nas biometrični sistem overi. Da to deluje, moramo pretentati senzor, da misli, da bere dejanski prst in ne samo to sled oz. artefakt. Ta vrsta napada bi lahko delovala na optičnih in kapacitivnih bralnikih, a ne na RF bralnikih, kajti ti potrebujejo živo kožo pod samim odtisom.
- **Napad na komunikacijske kanale:** Če napadalec ne more ogroziti sistema na točki zbiranja, je naslednji logični korak, da ogrozi njegove komunikacijske poti. Če je lahko spremenjena poslana informacija, tako da je lahko napačno pozitivna, ali da se zgodi napačna zavrnitev, potem je napadalec uspel. Da se to naredi, mora napadalec fizično vdreti v linijo med napravo in računalnikom. On/ona bi morala namestiti programsko opremo na računalnik (trojanskega konja), da bi prestregla predlogo pred lokalnim ali oddaljenim primerjanjem.
- **Napad na predlogo:** Če se pomaknemo po verigi navzgor in če ne moremo ogroziti same komunikacije, potem se lahko lotimo ogroziti kar samo shranjeno predlogo. Da spremenimo predlogo, se lahko napadalec loti medija, kjer je predloga shranjena, naprave, ki zagotavlja predlogo, ali predlogo samo, ko je v prehodu do gostitelja.
- **Napad na pomožni sistem:** V nobenem biometričnem sistemu ni 100 % pokritosti uporabniške baze. Še dodatno, nekateri uporabniki bodo imeli biometrične pomanjkljivosti, ki bodo potrebovale drugačen faktor overjanja. Ti pomožni sistemi so prav tako odprti za napade. Napadalec se bo vedno lotil najšibkejšega dela sistema in po navadi je to pomožni sistem [4].

Do odobrene identifikacije napačne osebe bi pri prstnih odtisih prišlo, če bi sistem ustvaril sliko z npr. 100 točkami razpoznavnosti, ki bo shranjena v bazi in bi služila za primerjavo, nato pa bi se hotela identificirati druga oseba, ki bi imela enake točke, in bi sistem po algoritmu prepoznal to osebo kot pravilno, čeprav ni. Do zavrnitve osebe, ki bi morala imeti dostop, pa bi prišlo, kadar se razpoznavne točke prstnega odtisa ne ujemajo s tistimi,

ki so že v bazi. To se lahko zgodi, kadar npr. napačno postavimo prst oz. imamo kakšne poškodbe na prstu [1].

Prednosti: preprostost, več proizvajalcev in različnih izdelkov, takojšnja identifikacija, večja varnost kot pri geslih.

Slabosti: lahko jih ponaredimo, manjša varnost kot pri nekaterih drugih metodah (šarenici in mrežnici), hitro pride do zavrnitve zaradi umazane podlage bralnika [5].

OVERJANJE S POMOČJO PREPOZNAVANJA OBRAZOV

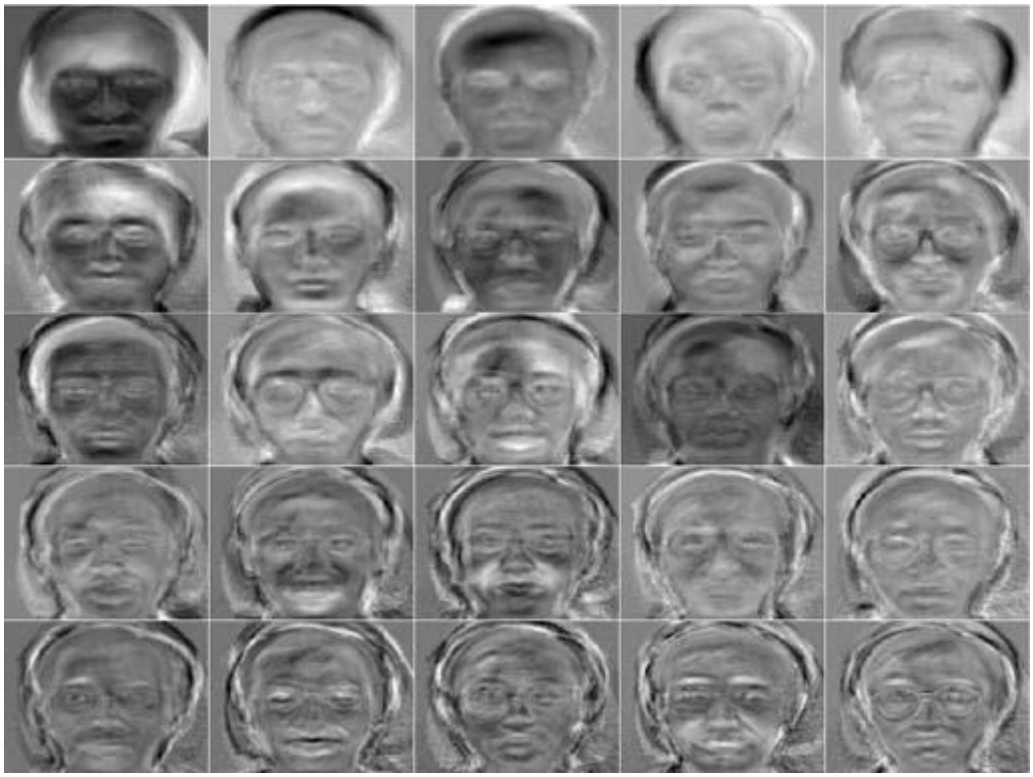
Ker bomo v nadaljevanju naše diplomske naloge implementirali metodo prepoznavanja obrazov s pomočjo algoritma Eigenobraz, bomo poglavju o overjanju s pomočjo prepoznavanja obrazov namenili največ časa. Tako bomo opisali glavne algoritme za prepoznavanje, jih primerjali, ter predstavili odobreno identifikacijo napačne osebe in neodobreno identifikacijo pravilne osebe.

Obraz je sestavljen iz več značilnih mikro in makro elementov. Makro elementi vključujejo usta, nos, oči, ličnice, brado, čelo in ušesa. Mikro značilnosti pa vključujejo razdaljo med makro elementi in velikostjo samih elementov. Prav tako nevidno za naše oči je dejstvo, da naš obraz oddaja toploto, ki jo lahko merimo s pomočjo infrardeče kamere. Vse te značilnosti lahko uporabimo pri sistemih obrazne biometrije za identifikacijo in overjanje posameznika.

Slike obraza lahko ujamemo kot živo sliko ali kot fotografijo ali video. Nekateri algoritmi sicer ne bodo podpirali uporabo fotografije ali videa, ker potrebujejo globino in druge tipe meritev. Kamere, ki so trenutno v uporabi za namene varnosti, so enake kot tiste na osebnih računalnikih. So poceni in so dovolj majhne, da jih lahko postavimo kamorkoli. Prav tako ne potrebujejo kakšne posebne povezave, saj jih večinoma podpira USB (ang. Universal Serial Bus).

Eigenobraz

Eigenobraz temelji na patentni tehnologiji iz MIT-a in v prevodu pomeni »tvoj/svoj obraz«. Algoritem deluje na 2D slikah v sivini (slika 3), iz katerih je nato izvlečen eigenobraz. Obraz je nato preslikan na vrsto eigenvektorjev (ang. eigenvectors), ki so pravzaprav matematične lastnosti, ki opisujejo edinstveno geometrijo posameznega obraza, kar pa ustvari biometrično predlogo. Ta predloga je nato primerjana z ustvarjenim eigenobrazom. Stopnja variance med predlogo in referenčnim eigenobrazom pa nato določi zadetek. Manjša kot je ta varianca, večja je možnost zadetka.



Slika 3: Eigenobrazi [25]

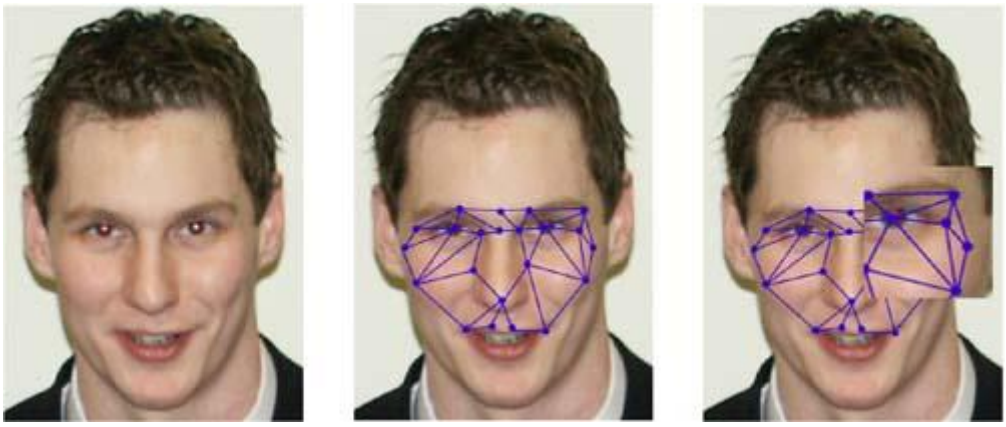
Algoritem Eigenobraz je nekako edinstven, kadar opravlja identifikacije ena-k-mnogo. Za ustvarjanje referenčne predloge, na katero primerjamo živo predlogo, zgradi sestavljanko vseh vpisanih obrazov. To pomeni, da vsakič, ko je nov obraz dodan v bazo, se ta referenčna predloga posodobi. Večinoma sistemov, ki temeljijo na eigenobrazih, naredi referenčno predlogo iz 100−150 obrazov.

Podrobneje bomo algoritem Eigenobraz razložili v kasnejših poglavjih.

Analiza lokalnih značilnosti

Analiza lokalnih značilnosti je bila razvita s strani Dr. Josepha Aticka, Dr. Paula Griffina, in Dr. Normana Redlicha iz Visionics Corporation-a. Ta analiza uporablja makro značilnosti obraza kot referenčne točke. Algoritem prvo locira obraz od njegove okolice. Nato so z uporabo sprememb v senčenju okoli sebe locirane referenčne točke. Ko je enkrat najdena sprememba v senci, je določena kot točka sidranja. Ko so najdene vse točke sidranja, algoritem ustvari trikotnike, ki povežejo skupaj vse točke sidranja. Koti med trikotniki so nato zmerjeni in ustvarjena je 672-bitna predloga. Če je sprememba v intenzivnosti svetlobe ali orientaciji, lahko to pomeni spremembo v sencah, kar bi pomenilo drugačno predlogo. Ko je branje živega obraza končano, je ustvarjena nova predloga z uporabo analize lokalnih značilnosti; ta nova predloga je nato primerjana z referenčno predlogo. Višji kot je procent primerjave, višja je verjetnost, da se bosta živa in referenčna predloga ujemali.

Slika 4 prikazuje obraz, na katerega je nanesena analiza lokalnih značilnosti. Najprej je obraz ločen od ozadja, nato pa je na vrsti analiza. Zadnja slika prikazuje pobližano sliko analize lokalnih značilnosti – razlike v senci so očitne.



Slika 4: Analiza lokalnih značilnosti [4]

Nevronska mreža

Algoritem nevronske mreže je oblikovan po sinapsah in nevronih v človeških možganih. Z ustvarjanjem umetne živčne mreže lahko rešimo probleme s pomočjo učenja ali treninga mreže. Da učimo mrežo, le to nahranimo z vrsto slik, ki imajo identificirane že svoje makro značilnosti. Poleg teh slik pa so v učni niz dodane tudi naključne slike. Te naključne slike pripomorejo, da se mreža nauči, kaj ni del obraza. Nato pa, ko se mreža začne učiti, dodajamo v sistem nove obraze, ki še nimajo identificiranih svojih makro značilnosti. Neidentificirani obrazi, ki jih ni uspelo prepoznati, so ponovno vrnjeni v sistem z identificiranimi značilnostmi.

Algoritem nevronske mreže je sestavljen iz naslednjih delov:

- Prepoznavanje obraza in okvirjanje: Ko je obraz pretvorjen v sliko, mora biti ločen od ozadja, nato pa je okvirjen in pretvorjen v pravo velikost.
- Vnosni nivo algoritma: V tej točki je slika obraza pretvorjena v slikovne pike (ang. pixel), da ustreza specifikacijam velikosti algoritma živčne mreže. Če je vnosni medpomnilnik 20 x 20 slikovnih pik in je slika iste velikosti, potem vsaka slikovna pika pomeni natančno en vnosni nevron.
- Sprejemna polja: Ko je slika prevedena v nevrone, so ti nevroni preslikani na sprejemna polja. Preslikanje sprejemnih polj je po navadi izbrano, da prikažejo glavne karakteristike obraza. Na primer, sprejemni nevroni so grupirani tako, da so lahko vnosni nevroni enakomerno razdeljeni v kvadrate in nato preslikani na en nevron. Tako bi nastalo veliko kvadratno področje za preslikanje, kjer so lahko izolirane glavne značilnosti obraza. Od tu naprej lahko dodatni sprejemni nevroni zavzamejo različne stopinje in oblike vnosnih nevronov, da pomagajo izolirati makro značilnosti, kot so nos, usta, oči in ušesa.
- Skrite enote: Skrite enote imajo razmerje ena-na-ena nevron/sprejemna polja. Na ta način lahko skrita enota določi, če je bila locirana prava značilnost.
- Izhod: Rezultiran izhod skrite enote, je en sam izhodni nevron. Glede na prej izbrani prag, lahko ta nevron podaja pozitivno ujemanje ali negativno ujemanje.

Avtomatsko procesiranje obraza

Avtomatsko procesiranje obraza je najenostavnejši algoritem prepoznavanja obraza. Deluje tako, da meri velikost makro značilnosti in razdaljo med njimi. Rezultirana razmerja, ki so ustvarjena, so potrebna za izdelavo obrazne predloge. Ko so enkrat razmerja izračunana, so predloge urejene glede na različna primarna razmerja. Na primer, glede na razdaljo med očmi ali glede na širino ustnic.

Primerjava algoritmov

Na obrazno biometrijo in njeno uporabo lahko zelo vplivajo razmere, v katerih je uporabljena. Mi se bomo osredotočili na biometrijo za osebno uporabo, katera je predvsem na prenosnikih in osebnih računalnikih, uporablja pa se doma/v pisarnah in kjer so svetlobne razmere konstantne, prav tako pa se bo uporabnik identificiral 3 do 4 krat na dan, kar pomeni, da mora biti algoritem zadosti hiter.

Eigenobraz: Algoritem Eigenobraz je dokaj hiter s svojimi iskanji. Po navadi potrebuje dobro svetlobo in da v kamero gledamo z obrazom obrnjenim povsem naprej. Ne deluje dobro z variacijami pri obraznimi izrazi, kar pomeni, da je najbolje, da uporabnik poda obraz, ki je vedno brez izraza, kar pa ni vedno mogoče. Poleg tega, ni konstanten pri ljudeh, ki včasih nosijo očala, ali imajo občasno brado. To pomeni, da bi moral uporabnik ponovno registrirati svoj obraz, če ima očala ali brado. Eigenobraz je dober splošno-namenski algoritem za uporabo, kadar je struktura uporabnikov relativno kontrolirana in pod določenimi pogoji. Dejstvo, da lahko očala in brada vplivata na sposobnost algoritma, lahko vodi do veliko napačnih zavrnitev (FRR), kar pomeni, veliko klicev do uporabniške pomoči in posledično nezadovoljstva uporabnikov.

Analiza lokalnih značilnosti: Ta analiza uporablja makro obrazne značilnosti, skupaj s strukturo kosti in spremembami med sencami za določanje točk sidranja. Kot takšen je veliko bolj odpustljiv, ko je govora o slabših svetlobnih pogojih in uporabnikih, ki imajo občasno očala ali brado. Tudi obrazni izraz ni tako kritičen, ker se struktura kosti pri tem ne spremeni. Tolerira tudi, kadar obraz ni povsem obrnjen naprej. Ker so za točke sidranja uporabljene makro značilnosti in struktura kosti, glava tudi ne potrebuje povsem mirovati,

da se začne primerjava. Na splošno je torej analiza lokalnih značilnosti zelo primerna kot overjanje za osebno uporabo.

Živčna mreža: Živčna mreža uporablja metodo učenja, da nauči mrežo kako prepoznati razlike na obrazih. Kot takšna je zelo dobra, da izolira obraz iz kompleksnega ozadja. Medtem ko so nekatere pisarne bolj neurejene kot druge, je na splošno pisarniško okolje urejeno. Algoritem prav tako potrebuje veliko bazo slik, da je neodvisen in je zato lahko počasnejši pri procesiranju podanega obraza za overjanje. Poleg tega živčna mreža potrebuje obraz obrnjen povsem naprej in dobro svetlobo. Medtem ko živčna mreža deluje zelo dobro v kompleksnih okoljih, naše domače okolje oz. pisarna ne potrebujeta takšnega nivoja prefinjenosti. Zato prednosti algoritma niso dovolj velike, da bi ga uporabljali v okoljih, ki nas zanimajo.

Avtomatsko procesiranje obraza: Avtomatsko procesiranje obraza je zelo hiter in učinkovit algoritem. Uporablja makro meritve in velikosti kombinirane s povezovanjem, zaradi česar je zmagovalec v hitrosti. Kar vemo je, da bodo variacije v obraznih izrazih vplivale na meritvah kot so širina ustnic in razdaljo med drugimi makro značilnostmi. Deluje pa zelo dobro v prostorih s slabo svetlobo, kar pa za naša okolja ni noben plus, saj so po navadi dobro osvetljena. Poleg tega lahko ima tudi velik odstotek napačnega sprejetja (FAR).

Priporočeni algoritmi: Iz naše analize algoritmov lahko vidimo, da je analiza lokalnih značilnosti najbolj primerna za overjanje za osebno uporabo [4].

Nevarnosti

Kot smo že omenili, so lahko celo ljudje pretentani, da mislijo, da poznajo obraz, ki ga v resnici ne. Zato je verjetno, da je lahko računalniški biometrični sistem prav tako pretentan. Sicer je na splošno znano, da obrazna biometrija ne ponuja istega nivoja napačnega sprejetja (FAR) kot ostale biometrične metode, ima pa zato druge privlačne atribute. Večinoma ljudi je pripravljena to metodo uporabljati vsak dan, je zelo dobro sprejeta in deluje pod nizkimi stroški. Vendar kot vse ostale metode, je tudi prepoznavanje obrazov lahko mogoče prevarati.

- **Napad na fizični obraz:** Obrazna biometrija je pasivna. To pomeni, da je lahko vaš biometrični vzorec vzet, brez da bi se vi zavedali le tega. Obstaja kar nekaj metod, ki prikazuje, kako so lahko predstavljeni obrazi in od njih je tudi odvisno, kako lahko te tehnike prevaramo: 2D slika, 2D slika z izrezanimi za oči, ponovitev ujetega videa …
- **Uporaba artefaktov:** Ker je obrazna biometrika pasivna in ne potrebuje uporabnika, da aktivno odda meritve, torej ni fizičnega kontakta med uporabnikom in bralnikom. To pomeni, da so artefakti, ki jih pusti bralnik drugačni od tistih, ki jih pusti bralnik prstnega odtisa. Obrazni artefakti so po navadi v obliki slik, ki jih je uporabljal sistem med zajemom. Kot takšne lahko podajo veliko informacij in podatkov za ponovitev napada na biometrični sistem [4].

FAR se pri prepoznavanju obraza zgodi, kadar se ujemata obraza dveh popolnoma različnih oseb. To se lahko zgodi pri enojajčnih dvojčkih ali pri osebah, ki so si zelo podobne, in sicer zato, ker tehnologija ni zajela dovolj točk, na podlagi katerih se oseba identificira. FRR pa se zgodi zaradi posledic sprememb obraza, pa naj bo to lepotna operacija, staranje, osvetlitev, frizura, očala, brada ali preprosto drug izraz na obrazu [1].

Prednosti: brez fizičnega kontakta z napravo, preprostost, uporaba standardnih kamer, cenovno ugodna.

Slabosti: napačna identifikacija dvojčkov, problemi s svetlobo (če je pretemno ali presvetlo), kretnje obraza lahko povzročijo zavrnitev, dlake in očala lahko povzročijo zavrnitev [5].

OSTALE METODE

Pri biometriji na podlagi fizičnih karakteristik obstaja še kar nekaj metod, ki pa niso tako primerne za osebno uporabo, bodisi, ker je oprema predraga ali pa so nepraktične. Med te metode spadajo bralnik šarenice, bralnik mrežnice, geometrija roke, odtis dlani ali stopal ipd.

2.2.3 Biometrija na podlagi vedenjskih karakteristik

Biometrija na podlagi vedenjskih karakteristik preučujejo naše vedenje in navade in na podlagi le teh overi uporabnika. Sama izvedba te vrste overjanja ni preveč kompleksna, a vendar med uporabniki metodi dinamično preverjanje podpisa in dinamika tipkanja nista preveč priljubljeni, zato ju bomo razložili le na kratko.

DINAMIČNO PREVERJANJE PODPISA

Dinamično preverjanje podpisa je avtomatizirana metoda preučevanja posameznikovega podpisa. Uporablja posebno pisalo in podlago, po katerem uporabnik piše. Ta tehnologija preuči dinamiko, kot sta hitrost, smer in pritisk pisanja, prav tako pa meri čas, kako dolgo se dotikamo podlage in celoten čas pisanja. Na žalost je preverjanje podpisa ena izmed najmanj zanesljivih metod, obstaja namreč veliko načinov, kako lahko ponarejevalci ponaredijo podpis [6].

Dinamičen podpis je dokaj lahko ponarediti, a vseeno mora vsiljivec točno poznati vzorec podpisa (tehniko, hitrost, pritisk). FRR pa se lahko zgodi, če oseba spremeni tehniko pisanja, če ima kakšno poškodbo ali bolezen (Parkinsonova).

Prednosti: odporen proti vsiljivcem, zlahka zamenjamo podpis ipd.

Slabosti: povišanje napak zaradi neskladnosti podpisov, ljudje niso navajeni podpisovanja na tablice [5].

DINAMIKA TIPKANJA

Dinamika tipkanja je avtomatizirana metoda preverjanja posameznikove tehnike tipkanja po tipkovnici. Ta tehnika preučuje dinamiko tipkanja, kot sta hitrost in pritisk tipk, čas tipkanja posamezne besede in čas med pritiski posameznih tipk. Algoritmi te tehnologije se še vedno razvijajo, da bi zagotovili večjo robustnost in boljšo delovanje [6].

FAR bi bil mogoč, če bi se kakšna oseba zelo dobro naučila ali uganila vaš vzorec pisanja, a za to je zelo majhna možnost. Program pa bi nam lahko zavrnil dostop, če bi s časom spremenili vzorec tipkanja in se tako ne bi ujemal s predlogo, ki smo jo ustvarili ob vzpostavitvi sistema. Druga stvar pa je kakšna bolezen ali hujša poškodba, ki bi spremenila vzorec [1].

Prednosti: ne potrebujemo strojne opreme, skupna uporaba z uporabniškim imenom, hitra identifikacija, nizka cena.

Slabosti: vzorec tipkanja se lahko s časom spremeni, programi, kateri lahko posnemajo tipkanje, različne tipkovnice, tehnologija še ni zaupljiva [5].

2.3 Overjanje na podlagi kaj imaš

Metode overjanja na podlagi kaj imaš ne uporabljamo pogosto pri osebni uporabi, a vendar predvsem pri internetnem bančništvu, se uporabljajo žetoni, s pomočjo katerih se vpišemo v našo e-banko oziroma opravimo nakup. Tudi nakupi preko Monete so kar razširjeni. V to skupino overjanja spadajo oblike, pri katerih je potrebna neka fizična naprava, ki jo mora imeti uporabnik in s pomočjo katere se lahko overimo. Za overjanje tako uporabljamo razne žetone, pametne kartice, mobilne naprave ipd.

VARNOSTNI ŽETONI

Varnostne žetone uporabljamo za elektronsko dokazovanje posameznikove identitete. Žeton uporabimo poleg ali namesto gesla, da dokažemo, da je nekdo, kar trdi, da je. Žeton torej deluje kot elektronski ključ za dostop do nečesa.

Nekateri žetoni lahko shranijo kriptografske ključe, kot na primer digitalni podpis, ali biometrične podatke, kot na primer minucije prstnega odtisa. Nekateri so v zelo odpornem ohišju, medtem ko spet drugi lahko vsebujejo majhne tipkovnice, ki omogočajo vpis PIN kode, ali preprost zaslon, ki začne generacijsko rutino, ki nato na zaslonu prikaže generirano številko. Posebni primerki vsebujejo USB priključek, RFID (ang. Radio

Frequency Identification) funkcije ali brezžični oz. modri zob (ang. Bluetooth) vmesnik, za potrebe prenosa generiranega zaporedja številk h klientu.

Uporaba žetonov

Najenostavnejši varnostni žetoni ne potrebujejo kakšne povezave z računalnikom. Klient vpiše številke s pomočjo lokalne tipkovnice, ki je na žetonu, po navadi skupaj s PIN kodo, kadar je zahtevana. Ker pa nismo povezani z overitvenim strežnikom, pomeni, da so takšni žetoni ranljivi na napad posrednika (ang. man in the middle attack).

Drugi žetoni se povežejo na računalnik preko brezžičnih tehnologij, kot na primer modrega zoba. Takšni žetoni prenesejo zaporedje številk k lokalnemu klientu ali bližnji dostopni točki.

Alternativno, že kar nekaj let so zelo dosegljivi žetoni v obliki mobilnih naprav, ki komunicirajo z uporabo zunajpasovne signalizacije kanalov (npr. glas, sms). Tako kot fizično nepovezani žetoni, so tudi tile ranljivi na napad posrednika.

Obstajajo pa tudi žetoni, ki jih priključimo direktno v računalnik. Za te moramo:

1. Povezati žeton z računalnikom s primerno vhodno napravo.
2. Vnesti PIN, če je potrebno.

Nato pa bo, odvisno od vrste žetonov, računalnik z žetona prebral ključ in opravil kriptografske operacije na njem, ali pa pustil žeton v strojni opremi, da opravi te operacije.

Shranjevalni žetoni

Shranjevalni žetoni so po navadi sestavljeni iz pametnih kartic in USB žetonov. Na vsakem žetonu je edinstvena informacija, ki identificira nosilca žetona. Če računalniški sistem sprejme samo žeton za potrebe overjanja, potem je lahko identificiran kdor koli, ki ima ta žeton. Zato, kadar se želimo identificirati z žetonom, le tega najprej ustavimo, nato pa vnesemo še geslo za dodatno zaščito. Ta vrsta metode se imenuje multifaktorna metoda

overitve. Samo, da vemo geslo, torej ne bo dovolj za overitev, imeti moramo oboje skupaj. Večina ljudi je seznanjena z multifaktornim preverjanjem iz bankomatov: bančna kartica je shranjevalni žeton, PIN pa je geslo. Je pa slabost tega načina, da si mora uporabnik zapomniti dve stvari namesto ene – kje je žeton in kakšno je geslo [4].

Dinamični žetoni

Tudi dinamični žetoni so lahko v več oblikah, kar pa jih razlikuje od shranjevalnih žetonov je, da generirajo enkratno overitveno kodo. Ta koda je lahko v obliki izziva, poslanega iz računalnika in odziva od žetona, ali registracije žetona in časovno odvisnega odzivnega ključa. Podobno kot shranjevalni žeton, tudi tukaj sam žeton ni dovolj, potrebno je še geslo, kar je zopet neprijetnost za uporabnika [4].

Slika 5: Različni žetoni [26]

Vrste žetonov

- Žetoni brez povezave (ang. disconnected tokens): Nimajo ne fizične, ne logične povezave do odjemalčevega računalnika. Običajno ne potrebujejo posebne vhodne enote in namesto tega uporabljajo vgrajen zaslon za prikaz informacij, ki jih uporabnik vnese ročno preko tipkovnice ali številčnice.

- Žetoni s povezavo (ang. connected tokens): So žetoni, ki morajo biti fizično povezani na odjemalčev računalnik. Ti žetoni avtomatsko oddajajo overitvene informacije, ko vzpostavimo fizično povezavo in tako odpravijo potrebo za ročni vnos informacij. Najbolj pogosti žetoni s povezavo so pametne kartice in USB žetoni.
- Brezkontaktni žetoni (ang. contactless tokens): Za razliko od žetonov s povezavo, le ti tvorijo logično povezavo z odjemalčevim računalnikom, a ne potrebujejo fizične povezave. Odsotnost potrebe po fizičnem kontaktu jih naredi bolj priročne, kot žetone brez in s povezavo. Kot rezultat so brezkontaktni žetoni popularni kot oblika vstopa brez ključa, kjer je uporabljena oblika RFID za oddajanje overitvene informacije od žetona. Obstaja pa velika skrb glede varnosti teh ključev, odkrili so namreč, da se da RFID zapise z lahkoto ponarediti, prav tako pa v primerjavi z USB žetonom baterija traja manj časa [9].
- Žetoni enojnega vpisa (ang. single sign-on tokens): Tu so uporabljeni žetoni, ki imajo zmožnost shranjevanja gesel. Uporabnik ima tako možnost, da z enim uporabniškem imenom in geslom pride do različnih aplikacij v sistemu, vendar samo do njegovega varnostnega nivoja. Za naslednji nivo je potrebno novo geslo oz. nove metode [10].
- Žetoni mobilnih naprav (ang. mobile device tokens): Uporabljajo mobilne naprave, kot so pametni telefoni in tablične računalnike kot overitvene naprave. To zagotavlja varno multifaktorno overitev, ki ne zahteva od uporabnika, da s seboj nosi še dodatno fizično napravo [9].

Ranljivosti

Največja ranljivost pri katerikoli posodi za gesla (ang. password container) je, da izgubimo posebno napravo s ključem oz. pametnim telefonom z integriranimi funkcijami ključa. Če se nam to zgodi, ne moremo v določenemu časovnemu obdobju nič storiti. Pred tem pa se lahko zavarujemo s pomočjo posebne elektronske vrvice, senzorjev ali alarmov. Fizično nepovezani žetoni in žetoni v obliki mobilnih naprav so ranljivi na napad posrednika. To pomeni, da se prevarant obnaša kot posrednik med sistemom in uporabnikom ter s tem pridobi vrednost na žetonu, kar pomeni, da mu je dovoljen dostop [24].

3 PROGRAMSKA REŠITEV ZA PREPOZNAVANJE OBRAZOV

3.1 Načrtovanje

Dandanes je med uporabniki daleč najbolj priljubljeno overjanje s pomočjo uporabniških imen in gesel, vendar kot smo opisali, ima ta metoda kar nekaj pomanjkljivosti, zato smo v diplomski nalogi združiti gesla s prepoznavanjem obrazov. Ko smo razmišljali, v kakšen projekt bi lahko vključili takšno rešitev, smo prišli na idejo o rezervacijah teniških ur preko računalnika. V lasti imamo namreč tenis igrišča in vemo, da je model, ki ga prakticirajo klubi, zelo neroden. Rezervacije so možne le na list papirja, ki se nahaja ob tenis igriščih, kar pa prinaša veliko slabosti. List lahko kdo ukrade, se lažno napiše, lahko se poškoduje ipd. Največji problem pa je, če nismo ob igriščih in nas nekdo pokliče, da želi igrati tenis, mi pa ne vemo točno, ali je takrat prosto ali ne.

Tako bi naša aplikacija omogočala rezervacije preko spleta, kjer bi se vsak uporabnik z registracijo vpisal v bazo, nato pa bi lahko z nekaj kliki rezerviral določeno uro, ostali uporabniki pa bi točno videli, kdaj je še prost kakšen termin. Ker pa bi lahko samo z uporabniškim imenom in geslom prav tako prišlo do zlorab (neznana oseba se registrira od doma, nato pa rezervira ure, katerih se nima namena udeležiti) smo rezervaciji dodali še sliko. Tako bi lastniki igrišč točno vedeli, katera oseba se je registrirala. Zmanjšala pa bi se tudi možnost napačne prijave, saj bi ob vsaki prijavi poleg uporabniškega imena in gesla prepoznaval tudi obraz osebe.

Celotno kodo in algoritem za prepoznavanje obrazov smo želeli napisati sami, vendar smo ob nadaljnjem preučevanju spoznali, da so ti algoritmi izjemno kompliciran, tako da smo poiskali odprto-kodni algoritem in ga vključiti v naš projekt. Po preučevanju različnih algoritmov smo se odločili za algoritem Eigenobraz, katerega smo podrobno opisali v naslednjih poglavjih.

Ker je sam algoritem napisan v Javi, smo tudi celoten program napisali v tem programskem jeziku. Tako se nam na začetku programa odpre začetni zaslon, kjer imamo

možnost prijave v sistem kot navaden uporabnik, kot administrator ali pa novo registracijo. Torej, če nas še ni v sistemu, se moramo prvo registrirati. Ob registraciji vpišemo naše ime in priimek ter geslo, preko spletne kamere pa se zajame naša slika, katero bomo kasneje uporabili pri prepoznavi. Nato imamo možnost vstopa v program, in sicer sistem preveri naše uporabniško ime in geslo ter primerja našo sliko s slikami v bazi. Če se vsi trije parametri ujemajo, lahko vstopimo v program, kjer imamo osnovne opcije registracije ure in preklic registracije. V primeru pa, da smo administrator, imamo na voljo še ostale možnosti kot so brisanje uporabnikov, nastavljanje datuma tednov terminov in brisanja vseh terminov.

V nadaljevanju se bomo posvetili samo delu aplikacije, ki prepoznava obraze. Opisali bomo cel postopek od implementacije posnetkov spletne kamere, do samega delovanja algoritma Eigenobraz.

3.2 Posnetki spletne kamere, detekcije gibanja in sledenje obrazu

Preden smo v naš projekt vključili algoritem za prepoznavanje obrazov, smo morali pripraviti okolje, kamor ga bomo implementirali. Tako smo na začetku morali generirati posnetke spletne kamere, nato pripraviti kamero, da je zaznala gibanje in na koncu še, da je sledila obrazu. Šele ko so vse komponente brezhibno delovale, smo začeli s prepoznavanjem obrazov.

Posnetki spletne kamere

Spletna kamera je odlično orodje za ustvarjanje novih oblik vnosa uporabnikov. Ima jo praktično vsak prenosni računalnik, tako da prepoznavanje obrazov preko nje postaja vse bolj priljubljena oblika overjanja. Na najnižjem nivoju kamera nenehno dostavlja slike (na primer roke ali obraza) v fazo procesiranja, ki nato zaposluje vrsto tehnik za procesiranje slik, ki pridobijo informacije o spreminjajoči se sceni.

Aplikacija uporablja za zajem slike iz spletne kamere JMF (Java Media Framework) − Performance Pack for Windows v 2.1.1e [7]. Cilj je, da zajamemo slike kar se da hitro in jih zaporedno prikazujemo v panelu.

Detekcija gibanja

Detekcija gibanja je zelo razširjena predvsem na področju varnosti, kjer kamere nenehno snemajo nek zavarovan prostor in ob neavtoriziranemu gibanju takoj sprožijo alarm oziroma pošljejo obvestilo pooblaščeni osebi. Na podoben način bo delovala tudi naša aplikacija, le da bo za nas to le vmesna točka do prepoznavanja obrazov.

Detekcija bo pri nas delovala tako, da se bodo okvirji (ang. frames) nenehno risali na panel s pomočjo kode iz prejšnjega poglavja. Nato bo detektor gibanja analiziral zaporedne okvirje in izpostavil kakršne koli spremembe na sliki. To bo prikazano z okvirjem v centru gravitacije gibanja.

Algoritem detekcije bo implementiran z uporabo računalniške knjižice OpenCV [11]. OpenCV je realno-časovna računalniška knjižica z zelo obsežno funkcionalnostjo. Leta 1999 jo je kot C knjižico razvilo podjetje Intel, leta 2009 pa je bila izdana nova verzija s C++ vmesnikom. Osnovne funkcije za obdelavo slik vključujejo filtriranje, detekcija robov, vzorčenje in interpolacijo, ter barvno pretvorbo. Za uporabo te knjižice v Javi pa potrebujemo še JavaCV [12], ki je ovojnica okoli OpenCV-ja in ustvari vmesnik okoli OpenCV-jevega C++-evega API-ja (ang. Application Programming Interface).

Sledenje obrazu

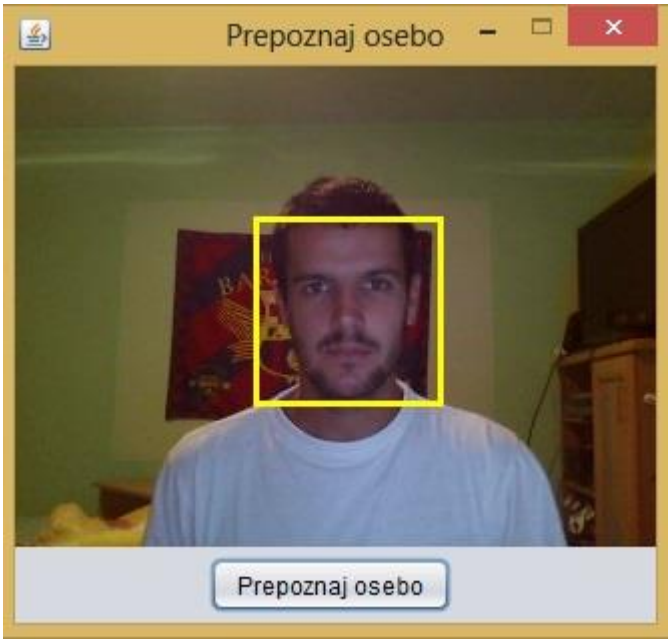
Sama detekcija gibanja za nas ni dovolj. Ko zna program enkrat slediti gibanju, ga moramo pripraviti do tega, da zazna, kaj je obraz in kaj ne. To bomo naredili tako, da bomo detektorju, ki zazna gibanje iz prejšnjega poglavja, dodali funkcionalnost, da med posameznimi okvirji analizira in najde še obraze. Tako bo program sledil obrazom.

Detekcijo obrazov bomo izvedli s klasifikatorjem Haar, ki je že naučen, da najde obrazne značilnosti. Za učenje tega klasifikatorja je sicer potrebno kar veliko časa, vendar mi uporabljamo kar klasifikator, ki je del OpenCV knjižice, tako da lahko preskočimo to fazo. S tem klasifikatorjem lahko brez problema najdemo tudi več obrazov hkrati, ampak samo, če so vsi obrazi dobro vidni. Obstaja pa lahko problem napačne identifikacije obrazov – program lahko namreč izpostavi dele slike, ki sploh niso obrazi.

3.3 Načrtovanje in implementacija prepoznavanja obrazov

Po tem, ko smo razvili programsko opremo, ki lahko zazna in sledi obrazu, ki se giblje pred kamero, moramo sedaj temu obrazu dodati še ime, in sicer s tehniko imenovano eigenobrazi.

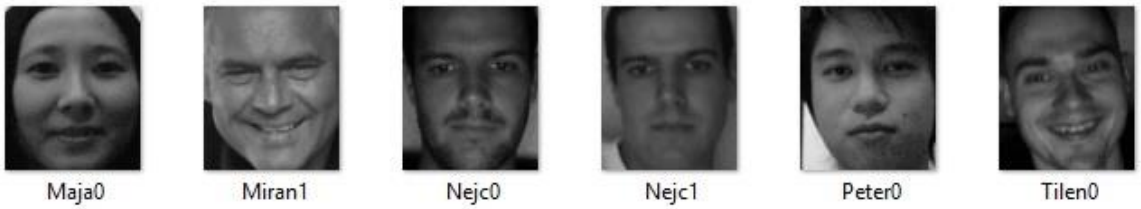
Rezultat je grafični uporabniški vmesnik (Graphical User Interface − GUI) kot ga prikazuje slika 6.



Slika 6: Uporabniški vmesnik za prepoznavanje obrazov

Ko uporabnik pritisne gumb »Prepoznaj osebo«, je trenutno izbrani obraz primerjan z naborom slik, ki so že shranjene (imenovane trening (ang. training) oz. učne slike), nato pa vrne ime in meritev razdalje povezano s učno sliko, ki je najbližji zadetek tej naši novi sliki.

Proces prepoznavanja se zanaša na učne slike s pripadajočimi imeni. Tipične učne slike so prikazane na sliki 7.



Slika 7: Učne slike

Pomembno je, da so vse učne slike izrezane in orientirane na podoben način, tako da so variacije med slikami samo zaradi obraznih razlik in ne zaradi ozadja ali položaja obraza. Mora biti enotnost med resolucijo, velikostjo in svetlostjo slik. Uporabno je, če je vključenih več slik istega obraza z različnimi izrazi, kot so smejanje ali mrščenje. Ime vsake osebe je zapisano v imenu vsake slike. Na primer, oseba Nejc je predstavljena z dvema slikama, imenovanima Nejc0 in Nejc1.

Te učne slike ustvarijo eigenobraze, ki so kompoziti učnih slik, ki izpostavijo elemente, ki so različni med obrazi. Tipični eigenobrazi, ki so ustvarjeni iz učnih slik, so prikazani na sliki 8. Zaradi svojega čudnega izgleda eigenobrazom včasih rečemo tudi obrazi duhov.

Slika 8: Eigenobrazi

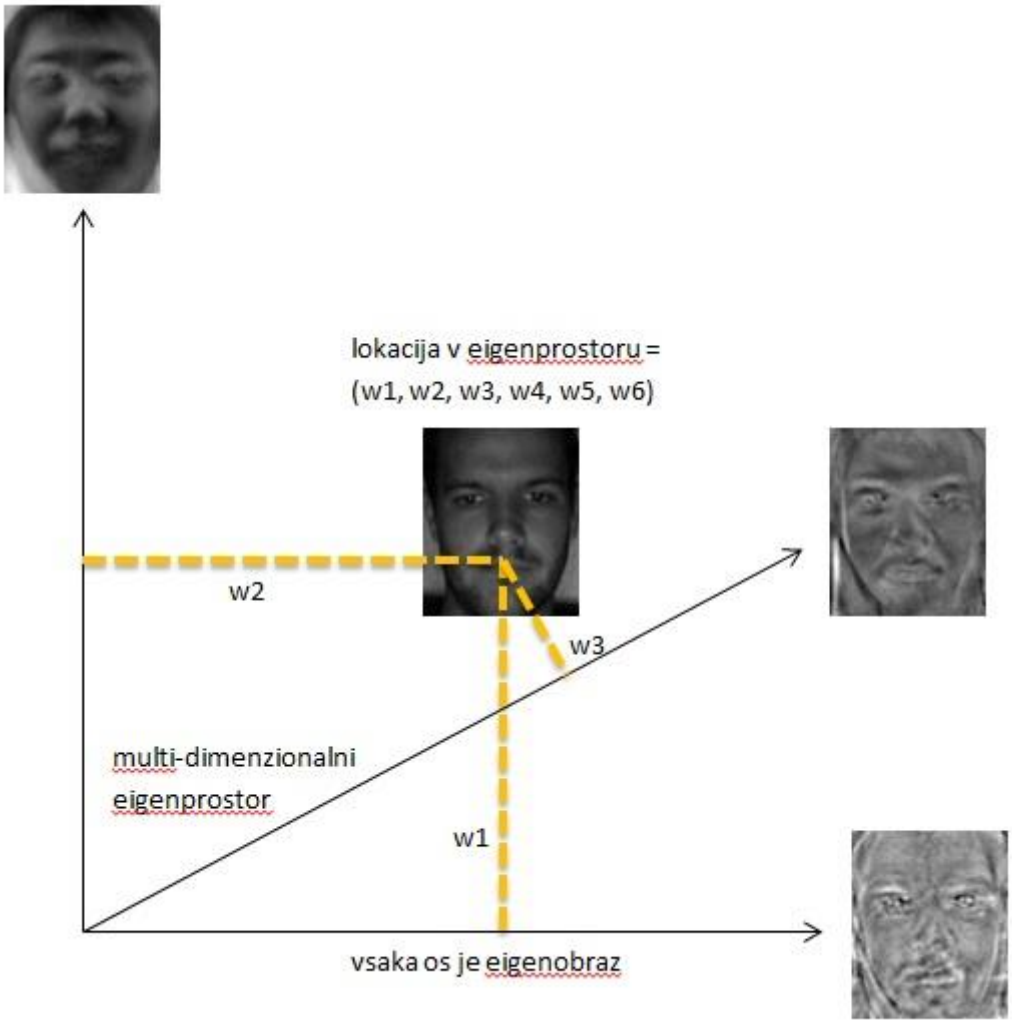
Vsaka učna slika je lahko predstavljena kot uteženo zaporedje eigenobrazov:

$$učna\ slika\ =$$
$$uteženo\ zaporedje\ \{eigen\_0, eigen\_1, eigen\_2, eigen\_3, eigen\_4, eigen\_5\}$$

Ideja je, da je učna slika lahko razgrajena na uteženo vsoto več eigenobrazov in da so vse uteži shranjene v zaporedje.

Niso pa vsi eigenobrazi isto pomembni – nekateri bodo vsebovali bolj pomembne obrazne elemente za razlikovanje med slikami. To pomeni, da po navadi ni potrebno uporabiti vse generirane eigenobraze, da se prepozna obraz. To dovoljuje, da je slika predstavljena z manjšim zaporedjem utežmi (npr. samo s tremi in ne s šestimi). Slabost tega pa je, da manj pomembne obrazne lastnosti ne bodo vključene v proces prepoznavanja.

Drug način razumevanja odnosa med eigenobrazi in slikami je ta, da je vsaka slika postavljena v multidimenzionalen eigenprostor (ang. eigenspace), čigar osi so eigenobrazi (slika 9).

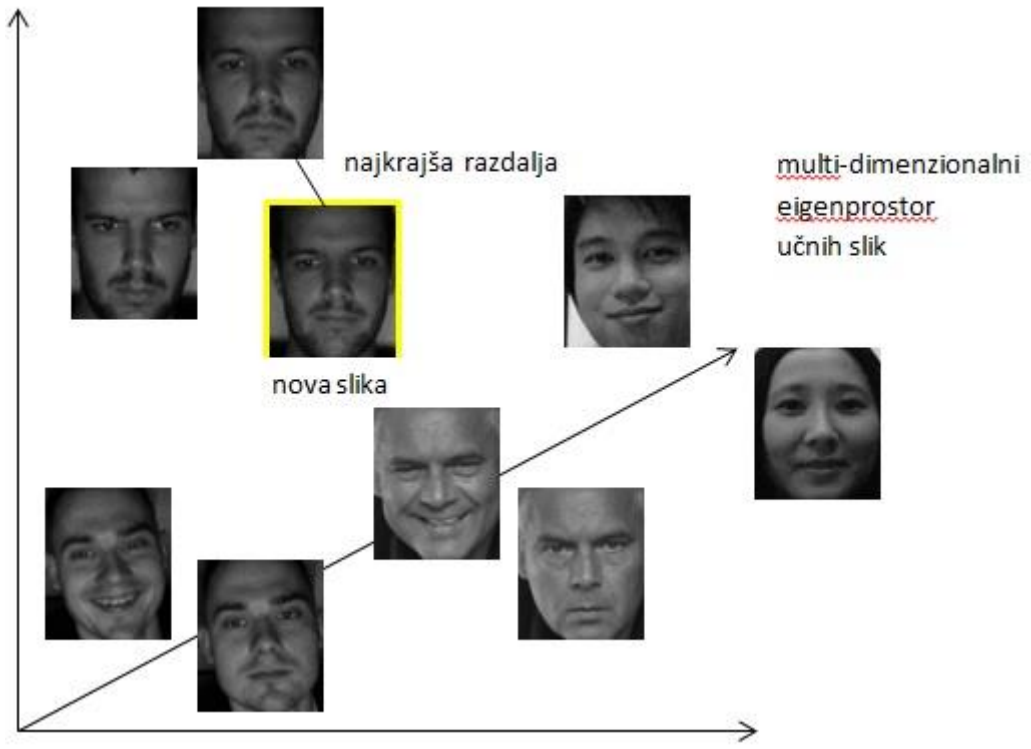


Slika 9: Slika v eigenprostoru

Težo lahko sedaj razberemo kot koordinate slike v eigenprostoru (če je 6 koordinat, bi moralo biti 6 osi).

Po fazi učnih slik pride na vrsto samo prepoznavanje. Slika novega obraza je razgrajena v eigenobraze s pripadajočimi utežmi. Ko pa dobimo zaporedje uteži, jih primerjamo s pripadajočim zaporedjem učnih slik in ime, katero je najbližji zadetek, je uporabljeno kot identifikacija.

Drugače lahko fazo prepoznavanja razložimo s pojmi eigenprostora. Nova slika s svojimi koordinatami (utežmi) je postavljena v eigenprostor. Potem pa najdemo najbližjo učno sliko glede na razdaljo med slikami, kot kaže slika 10.



Slika 10: Prepoznavanje slike v eigenprostoru

Čeprav v tej razlagi uporabljamo obraze, ni nobenih omejitev, da ta tehnika ne bi veljala tudi na drugih vrstah slik. V tem primeru se imenuje eigenslikovno (ang. eigenimage) prepoznavanje. Eigenslike delujejo najbolje na objektih, ki imajo redno strukturo in so sestavljene iz več pod-komponent. Obrazi so dobra izbira, ker imajo vsi zelo podobno kompozicijo (dvoje oči, nos in usta), ampak z različnimi variacijami med njimi (razdalja med očmi ipd.).

1 Generiranje eigenobrazov

Eigenobrazi (in eigenslike) so generirani z uporabo posebne tehnike imenovane analiza glavnih komponent (ang. Principal Component Analysis − PCA), ki izpostavi podobnosti in razlike v podatkih. V samo matematiko se ne bomo spuščali, raje se bomo zanesli na vizualno intuicijo delovanja analize.

Najprej pa bomo poskušali razložiti analizo glavnih komponent z uporabo dveh naborov numeričnih podatkov – nabor x vrednosti in nabor y vrednosti, kot je prikazano v tabeli 1.

Tabela 1: Podatki x in y

x	y
2.5	2.4
0.5	0.7
2.2	2.9
1.9	2.2
3.1	3.0
2.3	2.7
2.0	1.6
1.0	1.1
1.5	1.6
1.1	0.9

Naš cilj je, da izpostavimo podobnosti in razlike med tema dvema naboroma podatkov na numerični način. Predvidevamo, da ste seznanjeni s statističnimi meritvami, kot sta povprečje in standardni odklon (meritev razširjenosti podatkov okoli povprečja). Disperzija je še ena meritev razširjenosti in je samo kvadrat standardnega odklona:

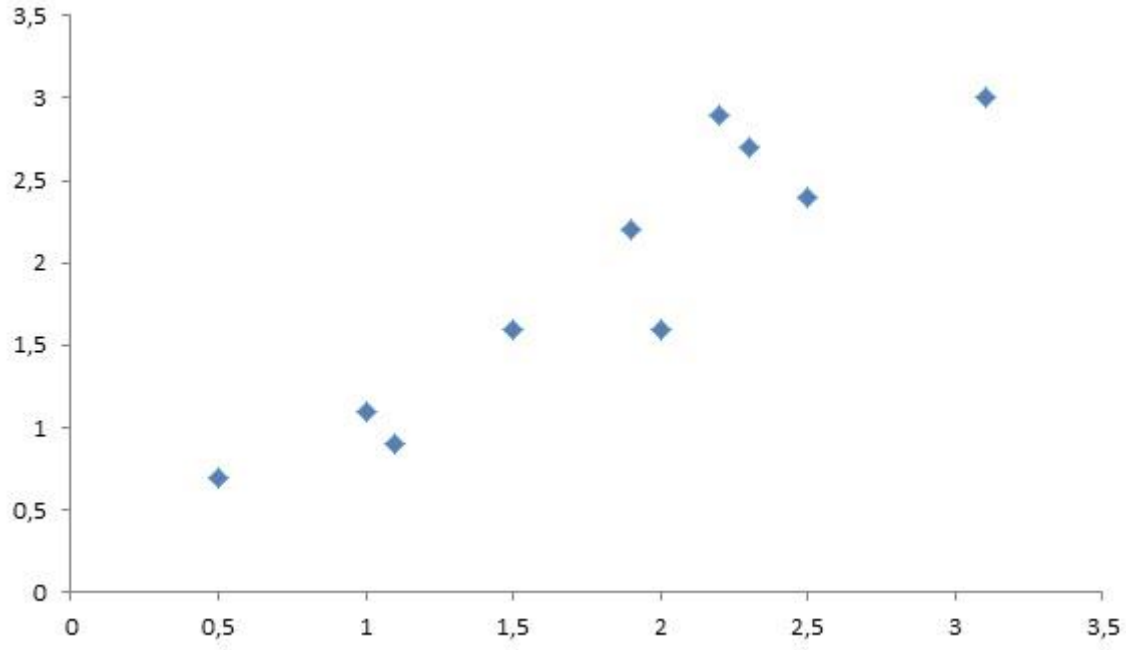
$$\mathrm{var}(x) = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(x_i - \bar{x})}{(n-1)} \tag{1}$$

n je število podatkov (10 v tabeli), $\bar{x}$ pa je povprečje podatkov.

Na žalost nam povprečje in disperzija pokažeta samo informacije o obliki podatkov v posameznem naboru (samo za nabor x); mi pa hočemo količinske razlike med nabori (na primer med x in y). Takšna meritev pa je kovarianca – to je generalizacija enačbe za disperzijo, ki primerja dva nabora podatkov.

$$\mathrm{cov}(x, y) = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{(n-1)} \qquad (2)$$

Če uporabimo kalkulator in uporabimo funkcijo cov() za podatke x in y v tabeli, dobimo rezultat 0.6154. Pomemben del je predznak: negativen pomeni, da ko se en nabor podatkov zmanjšuje, se drugi zvišuje. Če pa je pozitiven (kot v našem primeru), potem se oba nabora podatkov zvišujeta skupaj. Preprosta vizualna potrditev tega primera je, da narišemo pripadajoče x in y podatke na graf v obliki točk, kot prikazuje slika 11.



Slika 11: Podatki x in y

Če imajo podatki več kot dva nabora (na primer x, y in z), potem je kovarianca izračunana med vsemi pari naborov, kar rezultira v $\mathrm{cov}(x, y)$, $\mathrm{cov}(x, z)$ in $\mathrm{cov}(y, z)$. Ni nam potrebno

računati $\mathrm{cov}(y, x)$, $\mathrm{cov}(z, x)$ in $\mathrm{cov}(z, y)$, ker definicija enačbe pomeni, da je $\mathrm{cov}(A, B)$ enaka $\mathrm{cov}(B, A)$.

Standardni način za shranjevanje kovariančnih vrednosti med več nabori podatkov je v matriki, kjer podatki postanejo indeksi vrstic in stolpcev. Na primer, kovarianca med podatki x, y in z, bi bila matrika 3 x 3:

$$\begin{pmatrix} \mathrm{cov}(x,x) & \mathrm{cov}(x,y) & \mathrm{cov}(x,z) \\ \mathrm{cov}(y,x) & \mathrm{cov}(y,y) & \mathrm{cov}(y,z) \\ \mathrm{cov}(z,x) & \mathrm{cov}(z,y) & \mathrm{cov}(z,z) \end{pmatrix}$$

Kot vidimo, dobimo po glavni diagonali kovarianco s samim sabo, kar je enako, kot odklon v teh podatkih (to lahko vidimo, če primerjamo $\mathrm{var}(x)$ in $\mathrm{cov}(x, x)$). Matrika je prav tako simetrična okoli glavne diagonale.

Naši podatki x in y bi bili predstavljeni s kovariančno matriko 2 x 2:

$$\begin{pmatrix} 0.6166 & 0.6154 \\ 0.6154 & 0.7166 \end{pmatrix}$$

1.1 Od kovariance do eigenvektorjev

Kovariančna matrika je uporaben način za prikaz razmerja med nabori podatkov, ampak z uporabo matrike za izračun eigenvektorjev in eigenvrednosti lahko pridobimo še več informacij.

Eigenvektor je navaden vektor. Ko ga pomnožimo z dano matriko, spremeni le svoj obseg (magnitudo); eigenvrednost pa je le ime za ta obseg. Kot primer, bomo uporabili matriko:

$$\begin{pmatrix} 2 & 3 \\ 2 & 1 \end{pmatrix}$$

Eigenvektor za to matriko je $\begin{pmatrix} 3 \\ 2 \end{pmatrix}$, ker ko ga pomnožimo s to matriko, nam vrne ta isti vektor, pomnožen s 4:

$$\begin{pmatrix} 2 & 3 \\ 2 & 1 \end{pmatrix} \times \begin{pmatrix} 3 \\ 2 \end{pmatrix} = \begin{pmatrix} 12 \\ 8 \end{pmatrix} = 4 \times \begin{pmatrix} 3 \\ 2 \end{pmatrix}$$

4 je eigenvrednost za $\begin{pmatrix} 3 \\ 2 \end{pmatrix}$ eigenvektor.

Eigenvektorje lahko najdemo samo v kvadratnih matrikah, a še tu ne pri vsaki. Dana n x n matrika, ki ima eigenvektor, bo imela n vektorjev. Na primer, naš 2 x 2 primer zgoraj ima 2 eigenvektorja (in ustrezni eigenvrednosti).

Razmerje eigenvektorjev lahko matematično zapišemo kot:

$$Av = \lambda v \tag{3}$$

A je kvadratna matrika, v je eigenvektor za A in skalar λ je njegova eigenvrednost.

Za njihovo uporabo v analizi glavnih komponent bi morali biti eigenvektorji normalizirani, tako da bi vsi imeli isto dolžino. Na primer, eigenvektor od zgoraj, $\begin{pmatrix} 3 \\ 2 \end{pmatrix}$, ima dolžino brez enote $\sqrt{3^2 + 2^2} = \sqrt{13}$. Vektor smo delili s to vrednostjo, da smo dobili dolžino v enotah.

$$\begin{pmatrix} 3 \\ 2 \end{pmatrix} \div \sqrt{13} = \begin{pmatrix} 3/\sqrt{13} \\ 2/\sqrt{13} \end{pmatrix}$$

Morda se sprašujete, kako so eigenvektorji (in njihove eigenvrednosti) izračunani za dano matriko? Za 2 x 2 ali 3 x 3 matike je dokaj lahko. Detajli so razloženi v knjigah linearne algebre (kot npr. »Elementary Linear Algebra« od Howard Antona). Računanje je bolj zapleteno za večje matrike, ampak na srečo bomo uporabljali Colt matematično knjižnico [13], ki bo opravljala to delo.

Zgoraj smo že generirali kovariančno matriko:

$$\begin{pmatrix} 0.6166 & 0.6154 \\ 0.6154 & 0.7166 \end{pmatrix}$$

Ker je to matrika 2 x 2, ima dva eigenvektorja in dve eigenvrednosti.

$$\begin{pmatrix} -0.7352 \\ 0.6779 \end{pmatrix} \text{ in } 0.049$$

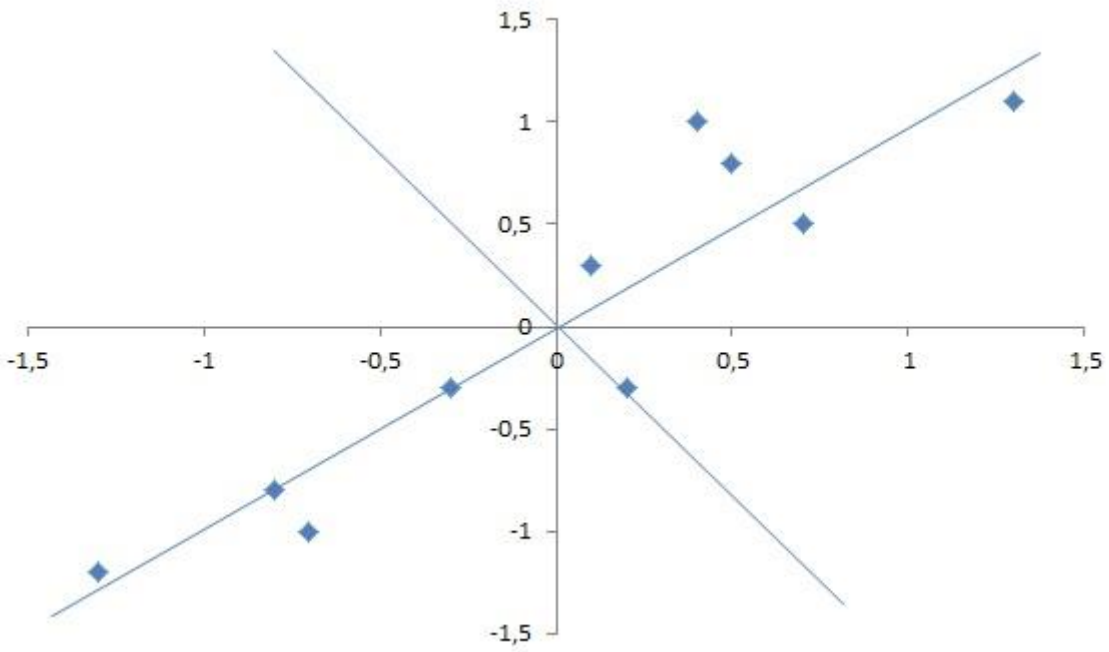
$$\begin{pmatrix} 0.6779 \\ 0.7352 \end{pmatrix} \text{ in } 1.284$$

Oba vektorja imata dolžino v enotah.

To poglavje smo začeli, tako da smo rekli, da eigenvektorji in eigenspremenljivke razkrivata več informacij o razmerju med nabori podatkov; v našem primeru med x in y podatki iz tabele. Torej kaj natančno nam razkrijeta?

1.2 Uporaba eigenvektorjev za analizo glavnih komponent

Vrnimo se h grafu, ki v obliki točk prikazuje podatke x in y. Na naslednji sliki je od x-a in y-a odšteto povprečje, tako da so točke prestavljene v center okoli izhodišča (podatki so bili normalizirani).

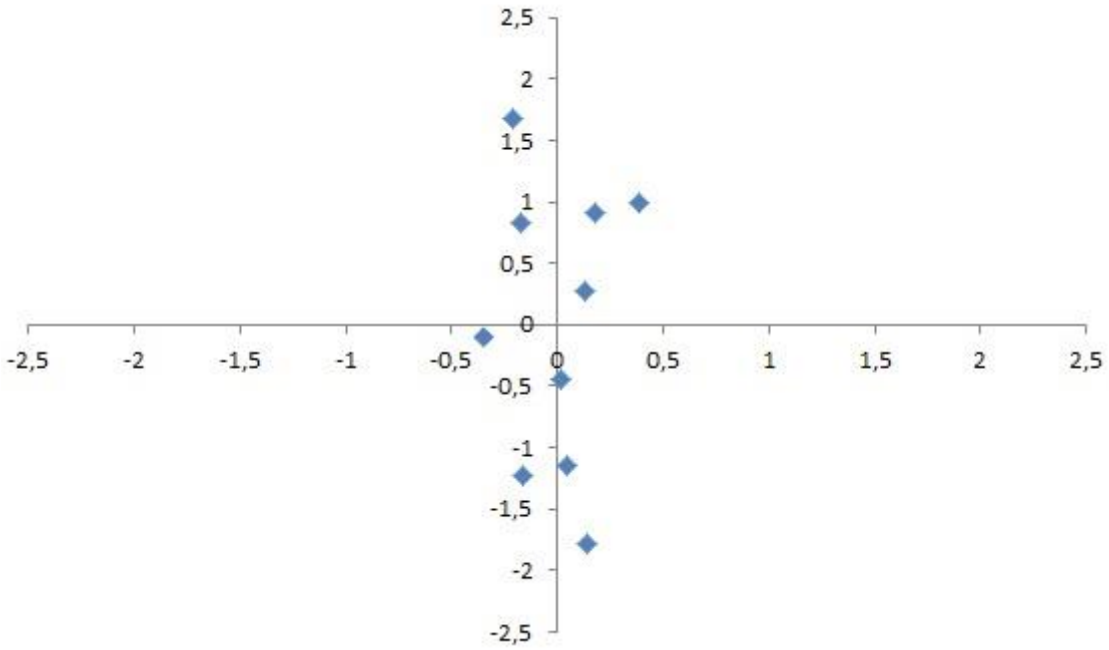


Slika 12: Normalizirani podatki x in y z eigenvektorji

Dva eigenvektorja sta lahko dodana v graf kot premici, ki sta pretvorjeni iz vektorjev v enačbi. $\begin{pmatrix} -0.7352 \\ 0.6779 \end{pmatrix}$ postane $y = \frac{-0.7352}{0.6779}$ x in $\begin{pmatrix} 0.6779 \\ 0.7352 \end{pmatrix}$ postane $y = \frac{0.6779}{0.7352}$ x. Normalizirani podatki in dve enačbi sta prikazani v grafu kot modre črte.

Zgornja slika kaže, kako eigenvektorji izpostavijo razmerja med nabori podatkov – vektorji nakazujejo, kako so nabori podatkov razpršeni po koordinatnem prostoru. Od dveh eigenvektorjev (označena z modro), je linija $y = \frac{0.6779}{0.7352}$ x najbolj uporabna, ker so podatki večinoma razpršeni po tej liniji. To je lahko potrjeno numerično, če pogledamo eigenvektorjeve pripadajoče eigenvrednosti. Eigenvektor $y = \frac{0.6779}{0.7352}$ x je boljši indikator razpršenosti, ker je njegova pripadajoča eigenvrednost (1.284) večja od obeh eigenvrednosti. Druga linija $y = \frac{-0.7352}{0.6779}$ x uporabno pripomore pri prikazu informacij s tem, da pokaže, kako so podatki razporejeni levo in desno od glavnega eigenvektorja (linije $y = \frac{0.6779}{0.7352}$ x), ampak je vseeno manj pomembna komponenta razpršitve podatkov, kar kaže tudi njena manjša eigenvrednost (0.049).

Eigenvektor z najvišjo eigenvrednostjo (v našem primeru linija $y = \frac{0.6779}{0.7352}$ x oziroma vektor $\begin{pmatrix} 0.6779 \\ 0.7352 \end{pmatrix}$) se imenuje glavna komponenta. Pravkar smo uporabili informacije eigenvektorja in eigenvrednosti, da izvedemo analizo glavnih komponent na naboru podatkov. Vsi eigenvektorji, pridobljeni iz matrike, so pravokotni. To pomeni, da je možno zavrteti (in mogoče preslikati) podatke tako, da postanejo eigenvektorji poravnani z osmi. Če se odločimo, da zavrtimo in preslikamo glavno komponento $\begin{pmatrix} 0.6779 \\ 0.7352 \end{pmatrix}$, da je poravnana z y-osjo, bi to izgledalo takole:



Slika 13: Obrnjena in preslikana verzija slike 12

Pravzaprav nam ni potrebno preslikati podatkov, vendar so te točke rezultat »Colt eigenvektorjeve funkcije« in smo se zato odločili, da jih ne spreminjamo.

Na zgornji sliki vidimo, da je koordinata podatka sestavljena iz razdalj točke od eigenvektorske osi. Na primer, točka najbližje izhodišču je $(0.13, 0.27)$. Lahko bi govorili tudi, da ima vsaka točka težo relativno k vsakemu eigenvektorju, kar tvori njeno zaporedje uteži $\{0.13, 0.27\}$.

Obe notaciji (koordinate ali uteži) sta lahko uporabljeni pri eigenobrazih. Zgoraj imamo predstavljeno uteženo zaporedje šestih eigenobrazov. Lahko je pa tudi podatkovna točka v šest-dimenzionalnem eigensprostoru, kot na sliki 9, kjer je vsaka os definirana z eigenobrazom (ali eigenvektorjem).

Pomaga, da si zapomnimo, kako bodo ti podatki uporabljeni kasneje, ko pridobimo nov podatek (nov obraz) in se moramo odločiti, kateremu od že pridobljenih podatkov (učnih slik) je najbolj podoben. Obstaja kar nekaj merilcev razdalje, ki jih lahko uporabimo, najbolj enostavna pa je Evklidova razdalja, ki je ravna črta med točkama v prostoru.

V tej fazi prepoznavanja bi lahko primerjal nove podatke s podatki, ki so pokazani na sliki 13, ampak v primerih iz pravega sveta želimo po navadi zmanjšati velikost podatkov. Cilj je, da pospešimo opravilo prepoznavanja, medtem ko ohranimo zadostne podatke, da ločimo med točkami, ko jih primerjamo z novimi podatki.

Poanta je v tem, da ohranimo vse podatkovne točke, a zmanjšamo dimenzije eigenprostora. To zmanjša velikost podatkov tako, da odstranimo nekaj koordinatnih osi, kar je isto, kot da bi odstranili nekaj eigenvektorjev. Zanima nas, katere eigenvektorje naj odstranimo? Odstraniti moramo tiste, ki imajo najmanjši vpliv na razpršenost podatkov, kar zvemo s pogledom na eigenvektorjem pripadajoče eigenvrednosti.

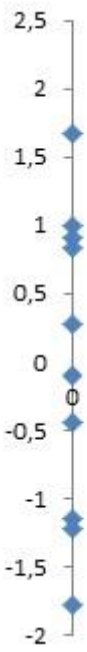
Podatki, ki so prikazani na sliki 13, imajo naslednje eigenvektorje in eigenvrednosti:

$$\begin{pmatrix} -0.7352 \\ 0.6779 \end{pmatrix} \text{ in } 0.049$$

$$\begin{pmatrix} 0.6779 \\ 0.7352 \end{pmatrix} \text{ in } 1.284$$

Prvi eigenvektor je bil zavrten na x os, drugi pa na y os.

Prvi eigenvektor pripomore najmanj k razpršenosti informacij zaradi svoje majhne eigenvrednosti (tudi če pogledamo sliko 13). Če ga zavržemo, so podatkovne točke projecirane na y osi, x os pa izgine, kar prikazuje spodnja slika.



Slika 14: Podatki iz slike 13 projektirani na y os

Ta pristop pomeni, da smo razpolovili obseg shranjenih podatkov za vsako podatkovno točko, ampak so podatki še vedno dovolj razpršeni, tako da so nove podatkovne točke lahko primerjane s pomočjo meritev Evklidove razdalje. V naslednjem poglavju si bomo pogledali primer tega.

Na žalost pa odstranitev eigenvektorja ni vedno tako pozitivna. Odstranitev pomeni, da je razpršitev informacij izgubljena in to je lahko kritično, če so originalne točke zelo skupaj. Na primer, slika 13 vsebuje tri točke blizu $y == 1$, ki so razpršene vzdolž x osi. Ko pa je ta os odstranjena, so te točke na sliki 14 veliko bližje skupaj in jih je tako težje ločiti.

2 Glavne komponente analize

V naslednjem poglavju bomo razložili implementacijo poenostavljene verzije algoritma analize glavnih komponent. Java direktno ne podpira potrebnih statističnih in linearnih algebraičnih operacij, zato pa obstaja veliko knjižic, ki opravljajo ravno to delo. Mi bomo uporabili paket Colt.

Colt podpira multidimenzionalna polja, operacije linearne algebre, vključujoč eigenvektorje in ogromno matematike ter statistike. Slabost pa je, da je paket kar star (2004) in o njem ni skoraj nič informacij, razen njegova API dokumentacija. Po pravici, smo ga uporabili, ker je paket Javafaces [14], ki ga bomo uporabljali kasneje, zgrajen nad njim.

Podatke, s katerimi smo računali zgoraj, lahko preverimo s testnim primerom, kjer s pomočjo Colt knjižice izračunamo kovariančno matriko, ki je uporabljena za izračun eigenvektorjev in eigenvrednosti. Novi podatki so nato primerjani s spremenjenimi učnimi podatki z uporabo meritve Evklidove razdalje.

Kompleksnost računanja eigenvektorjev in eigenvrednosti so shranjene znotraj EigenvalueDecomposition razreda, ki je inicializiran s kovariančno matriko in ima get() metode za dostop do eigenvektorjev in skalarjev (getV() in getRealEigenvalues()).
Učni podatki in novi podatki so preoblikovani znotraj transformAllData(), ki povprečno-normalizira (ang. mean-normalized) podatke in jih zavrti, tako da glavna komponenta (najpomembnejši eigenvektor) sovpada z y osjo. Z drugimi besedami, podatki so spremenjeni, tako da izgledajo kot tisti na sliki 13 (čeprav program ne prikaže nobenega grafa).

Najmanjša Evklidova razdalja med novimi in učnimi podatki je izračunana v metodi minEuclid(), ki vrača indeks pozicije najbližjega podatka v naboru učnih podatkov.
Novi podatek kaže na točko $(2.511, 2.411)$, katera je najbližje učni točki $(2.5, 2.4)$, ki pa je na indeksu 0 v trainingData[][]. Metoda transformAllData() uporablja vse eigenvektorje, ampak bi bilo lažje, da bi uporabili samo glavni vektor, $\begin{pmatrix}0.6779\\0.7352\end{pmatrix}$, tako da bi bili podatki projektirani na y os, kot na sliki 14.

2.1 Kovariančna matrika

Kovariančna matrika za x, y in z, bi bila 3 x 3 matrika:

$$\begin{pmatrix} \mathrm{cov}(x,x) & \mathrm{cov}(x,y) & \mathrm{cov}(x,z) \\ \mathrm{cov}(y,x) & \mathrm{cov}(y,y) & \mathrm{cov}(y,z) \\ \mathrm{cov}(z,x) & \mathrm{cov}(z,y) & \mathrm{cov}(z,z) \end{pmatrix}$$

cov() je vgrajena funkcija v knjižnjici Colt, v Descriptive.covariance(), tako da metoda calcCovarMat() porabi večinoma časa, da gradi matriko.

Koveriančna matrika za moje učne podatke je: $\begin{pmatrix} 0.6166 & 0.6154 \\ 0.6154 & 0.7166 \end{pmatrix}$.

2.2 Uporaba eigenvektorjev in eigenspremenljivk

Eigenvektorji in eigenspremenljivke pridobljene iz kovariančne matrike, so vrnjene v DoubleMatrix2D in DoubleMatrix1D podatkovne strukture v Colt-u, katere so enostavne za manipuliranje in tiskanje. Njihove vrednosti so natisnjene kot:

Eigenvektorji:
2×2 matrika
$-0.735179 \;\; 0.677873$
$\;\; 0.677873 \;\; 0.735179$

Eigenvrednosti:
1×2 matrika
$0.049083 \;\; 1.284028$

Vektorji in skalarji so navedeni v vrstnem redu stolpca, tako da je glavna komponenta v drugem stolpcu $\begin{pmatrix} 0.6779 \\ 0.7352 \end{pmatrix}$, ker je njena eigenvrednost 1.284028.

Naša metoda reportBiggestEigen() v matriki eigenvrednosti išče največjo vrednost in nato vrne njeno pozicijo v stolpcu. Z njo lahko izberemo glavno komponento v eigenvektorjih.

2.3 Prepoznavanje novih podatkov

Novi podatki imajo koordinate (2.511, 2.411), in »prepoznavanje« pomeni najdbo učne podatkovne točke, ki je najbližje tej koordinati.

Nova koordinata je dodana k že obstoječimi učnimi podatki v transformAllData(), ki preoblikuje vse podatke na načine prikazane na sliki 12 in 13 – prvo so podatki povprečno-normalizirani, nato zavrteni in preslikani, tako da je glavna komponenta poravnana z y osjo. Ta poravnanost je dosežena z uporabo normaliziranih eigenvektorjev, ki preoblikujejo podatkovne točke z matričnim množenjem:

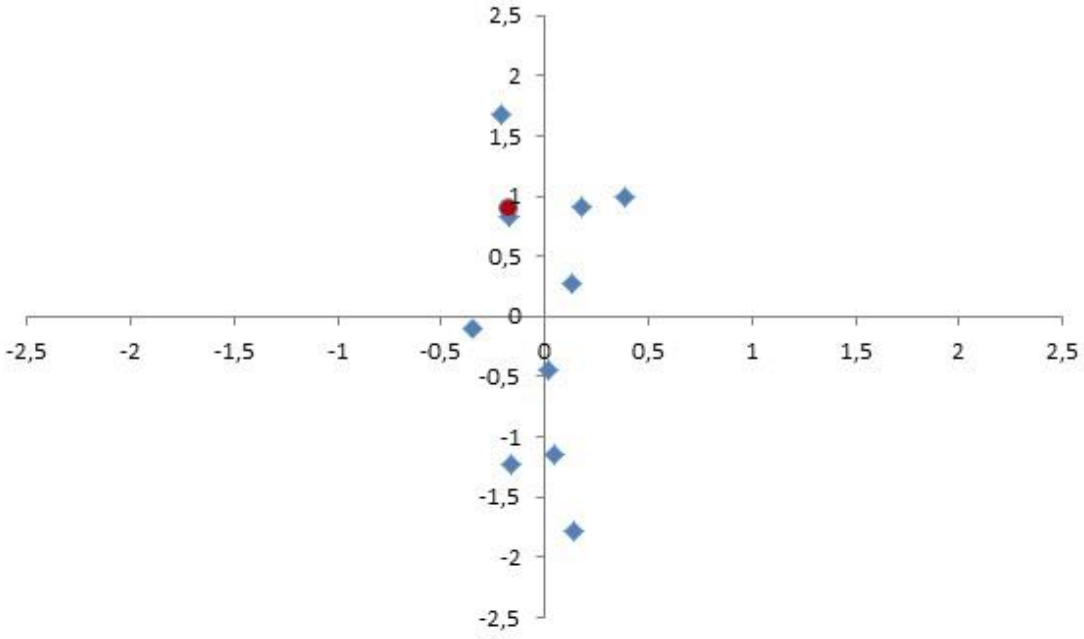
$$\text{Preoblikovani podatki} = \text{preneseni\_eigenvektorji} \times \text{podatkovna\_matrica}$$

Eigenvektorji so shranjeni v stolpčnem redu, ki mora biti spremenjen v vrstični red s prenosom, in sicer preden izvedemo matrično množenje.

Novi podatki so dodani v prvi stolpec nove matrike, imenovane matNewData, učni podatki pa v druge stolpce. Matrika je nato povprečno-normalizirana, vektorji pa so preneseni. Preoblikovani podatki so ustvarjeni z množenjem prenesenih vektorjev s podatki. Matrika, ki jo dobimo:

$$\begin{pmatrix} -0.1758 & -0.1751 & 0.1429 & 0.3844 & 0.1304 & -0.2095 & 0.1753 & -0.3498 & 0.0464 & 0.0177 & -0.1627 \\ 0.8435 & 0.8280 & -1.7776 & 0.9922 & 0.2742 & 1.6758 & 0.9130 & -0.0991 & -1.1446 & -0.4380 & -1.2238 \end{pmatrix}$$

Prvi stolpec so koordinate novih preoblikovanih podatkov, medtem ko so ostali stolpci učni podatki. Slika teh podatkov je prikazana spodaj (novi podatek je obarvan rdeče).



Slika 15: Nov podatek v primerjavi z obstoječimi

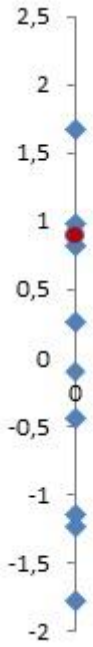
Slika 15 je enaka kot slika 13, le da je dodana še nova točka, ki jo najdemo tik nad točko (− 0.1751, 0.8280), ki je pravzaprav učna točka (2.5, 2.4). Z drugimi besedami, nova podatkovna točka (2.511, 2.411) je najbližje učni podatkovni točki (2.5, 2.4).

Zdi se kot velika dela samo za primerjavo nekaj točk, ampak prava prednost eigenvektorjev je, ko postanejo dimenzionalnosti teh točk zelo velike (tukaj je dimenzionalnost 2, ko pa gledamo slike, je le ta tudi do 40.000).

Z večjimi dimenzijami pa prihajajo potrebe, da pospešimo proces primerjave. Dimenzije preoblikovanih podatkov lahko preprosto zmanjšamo z uporabo manj eigenvektorjev. Na primer, manjši eigenvektor je lahko odstranjen iz eigenVecs v DoubleMatrix2D, še preden je dodan v transformAllData(). Ta metoda bi potem vrnila le eno vrstico podatkov:

$$(0.8435\ \ 0.8280\ -1.7776\ \ 0.9922\ \ 0.2742\ \ 1.6758\ \ 0.9130\ -0.0991\ -1.1446\ -0.4380\ -1.2238)$$

To je isto kot slika točk na glavni komponentni osi, kot jo prikazuje spodnja slika.



Slika 16: Nov podatek projektiran na y os

Slika 16 je podobna sliki 14, ampak je tu vključen nov podatek v obliki rdeče točke. Slika prikazuje potencialni problem z zmanjšanjem dimenzionalnosti: težje se je odločiti, katera točka je najbližja novemu podatku.

Še en faktor pa je enačba Evklidove razdalje. Uporablja pragovno vrednost za definiranje, kdaj sta dve točki blizu skupaj. Če pa je vrednost prevelika, potem je lahko več točk znotraj sprejemljive razdalje od nove podatkovne točke.

2.4 Računanje meritve razdalje

Knjižica Colt vsebuje številne funkcije za merjenje razdalje. Naša minEuclid() metoda kliče Colt-ovo Statistic.distance() metodo nad 2 x 11 matriko prikazano na sliki 15. Ta izračuna razdaljo med točkami v matriki, in vrne simetrično 11 x 11 kvadratno matriko. Mi potrebujemo samo prvo vrsto (ali stolpec), kjer so razdalje med prvo točko (v našem primeru novim podatkom) in ostalimi točkami v matriki (učnimi podatki). Pomembna vrstica je tako:

(0 0.0156 2.6404 0.5795 0.6464 0.8330 0.3578 0.9586 2.0005 1.2961 2.0674)

Prvi element v vrstici lahko zanemarimo, saj je to razdalja med novo točko in samim seboj. Ostale vrstice pa moramo pregledati, tako da najdemo najmanjšo vrednost, ki je v našem primeru kar druga vrednost v vrstici. To je razdalja do prve vrednosti v učnih podatkih, (2.5, 2.4).

MinEuclid() najde drugo najmanjšo razdaljo v prvi vrsti matrike razdalj, matDist, tako da jih sortira v naraščajočem vrstnem redu. Nato uporabi to vrednost razdalje, da išče skozi nesortirano matriko, tako da nam lahko vrne indeks pozicije te vrednosti.

3 Od eigenvektorjev do eigenobrazov

Do sedaj smo razlagali analizo glavnih komponent z uporabo manjše kolekcije podatkov x-ov in y-ov. Dobra novica je, da je od takšnega tipa informacij, pa do informacij sestavljenih iz slik, zelo preprost korak.

V bistvu vse, kar potrebujemo je, da pretvorimo vsako sliko v podatkovno točko, in potem lahko uporabljamo isti algoritem analize glavnih komponent.

Pretvorba slike v podatkovno točko je stvar obravnave slike kot 2D polje, in preslikane v 1D vektor, kot kaže spodnji primer.

$$\begin{pmatrix} p_{11} & \cdots & p_{1N} \\ \vdots & \ddots & \vdots \\ p_{N1} & \cdots & p_{NN} \end{pmatrix} \dashrightarrow \begin{pmatrix} p_{11} \\ \vdots \\ p_{1N} \\ p_{21} \\ \vdots \\ p_{2N} \\ \vdots \\ p_{NN} \end{pmatrix}$$

Če je slika velikosti 200×200 v sivini, potem bodo vse podatkovne točke v 40.000 dimenzionalnem prostoru. To v teoriji ni problem, ampak vodi v zelo velike matrike v knjižnici klicev metod. Na primer, kovariančna matrika mora primerjati vsako dimenzijo, in bi bila tako velika 40.000×40.000 (1.6×10^9). To bi vodilo h generiranju 40.000 eigenvektorjev (in njihovih eigenvrednosti). Velika večina teh eigenvektorjev bi imela zelo

male eigenvrednosti, kar bi pomenilo, da niso potrebni v primeru iskanja razlik s podatkovnimi točkami slik.

Na splošno, če ima slika v sivini N pikslov na kvadrat, potem bo generiranih N^2 eigenvektorjev. V naslednjih razlagah bomo predvidevali, da imamo M slik.

Pametni način za zmanjšanja števila eigenvektorjev in velikosti matrik sta predstavila Turk in Pentland v njunem delu »Face Recognition using Eigenfaces«. Njuna ideja je uporabiti manipulacijo nad matrikami, da drastično zmanjšamo število eigenvektorjev. Poanta je, da razstavimo kovariančno matriko (ki je $N^2 \times N^2$) na dve matriki ($N^2 \times M$) in ($M \times N^2$), in ju nato zamenjamo ter pomnožimo, da dobimo to veliko manjšo matriko velikosti ($M \times M$). Za nanos razgradnje eigenvektorjev na to novo matriko bo potrebno veliko manj dela, saj bo generiranih samo M eigenvektorjev.

Vsak od teh M eigenvektorjev bo dimenzije M, tako da jih bo potrebno pretvoriti nazaj v M eigenvektorje z originalno dimenzijo N^2. To storimo z uporabo hitrega matričnega množenja.

Končni rezultat je M eigenvektorjev, kar je v primerjavi s prvotnimi N^2 eigenvektorji veliko manj. V praksi po navadi potrebujemo še manj kot M eigenvektorjev za prepoznavo nove slike; po navadi je dovolj že $\frac{3xM}{4}$ ali $\frac{M}{2}$.

Teh M eigenvektorjev lahko gledamo kot osi v eigenprostoru. Prav tako je mogoče, da pretvorimo vsakega od ($N^2 \times 1$) vektorja v sliko velikosti ($N \times N$), kar je tudi razlog, da se imenujejo eigenobrazi (ali obrazi duhov), kot kaže slika 8.

4 Implementacija eigenobraz prepoznave

Eigenobraz prepoznava je tako uporabna, da že obstaja odlična Java aplikacija, ki jo izvaja – Javafaces, ki jo je razvil Sajan Joseph [12].

Trenutna verzija je opremljena z lepim uporabniškim vmesnikom, ampak mi smo uporabili prejšnjo verzijo, ki deluje s pomočjo komandne vrstice. Prav tako smo kar spremenili

kodo, in sicer smo jo razdelili na dva velika razreda: BuildEigenFaces in FaceRecognition. Kot imeni že predlagata, razred BuildEigenFaces generira eigenvektorje in eigenvrednosti od učnih podatkov, medtem ko razred FaceRecognition sprejme nove podatke in poišče učno sliko, ki se z njo najbolj ujema.

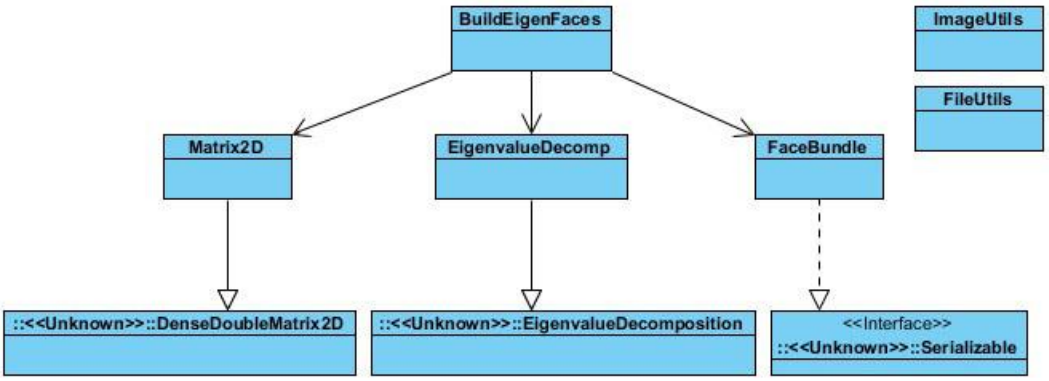
V nadaljevanju bomo pregledali spremenjeno Javafaces kodo, ampak ne bomo razložili vsake metode. Razlaga, ki sledi, se koncentrira na kodo povezano z analizo glavnih komponent in ne na standardne stvari, kot so branje in pisanje v datoteke.

5 Faza Gradnje

Vsaka učna slika je shranjena v datoteko, v poddirektoriju imenovanem trainingImages (slika 7). Obstaja več izidov, ko zaženemo BuildEigenFaces, najpomembnejša pa je binarna datoteka imenovana eigen.cache (eigen.predpomnilnik). Predpomnilnik je serializiran FaceBundle objekt, ki shrani izračunane eigenvektorje in eigenvrednosti, skupaj s številnimi drugimi informacijami o učnih slikah.

Dva druga izhoda pa sta dva nova direktorija: eigenfaces/ in reconstructed/, ampak nobeden izmed njiju ni potreben za samo prepoznavanje, saj je vse, kar potrebujemo, shranjeno v eigen.cache. Ta direktorija sta uporabna, da pregledamo sam proces gradnje. Eigenfaces/ vsebuje vse eigenobraze shranjene kot sliko (slika 8), medtem ko je reconstructed/ ustvarjen z BuildEigenFaces, ki z eigenvektorji in utežmi poustvari učne slike, kot nekakšno ponovno preverjanje natančnosti kalkulacij eigenvektorjev.

Razredi vključeni v fazo gradnje so prikazani v UML razrednem diagramu na spodnji sliki.



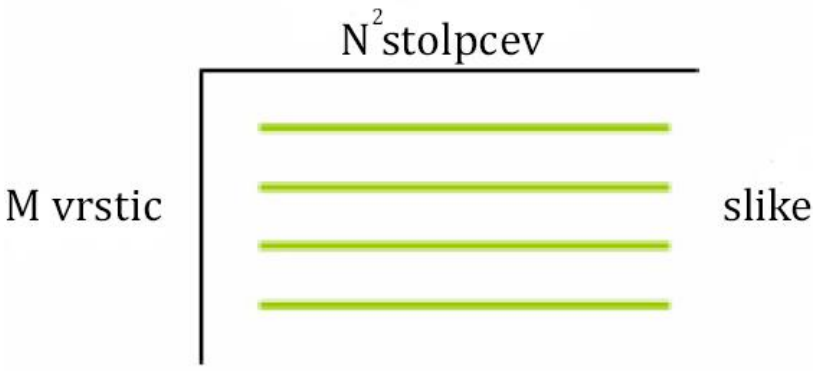
Slika 17: Razredi vključeni v fazo gradnje eigenobrazov

Pogosto uporabljene metode za manipulacijo nad datotekami in slikami so shranjene v razredih FileUtilss in ImageUtils. Razred zgornjega nivoja (ang. top-level) je BuildEigenFaces, ki uporablja poenostavljeno različico Colt-ovih DenceDoubleMatrix2D in EigenvalueDecomposition razredov, s tem da naredi podrazrede Matrix2D in EigenvalueDecomp. Razred FaceBundle je uporabljen za ustvarjanje serializiranega objekta, ki je shranjen v eigen.cache na koncu gradnje.

Gradnja paketa programske opreme

Večina pomembne kode analize glavnih komponent kliče metoda makeBundle() v razredu BuildEigenFaces. Ta ustvari eigenvektorje/eigenvrednosti FaceBundle objekta za specifične datoteke učnih slik, in prav tako shrani vsak eigenvektor kot sliko v eigenfaces/.

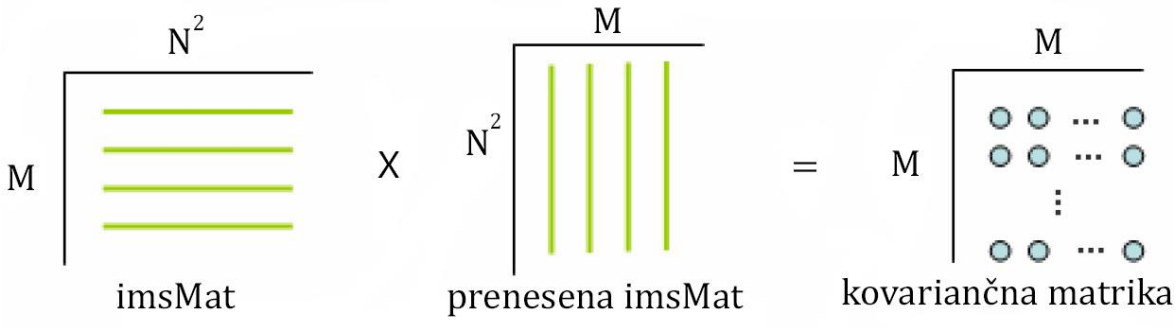
Vse slike so naložene v eno polje z metodo FileUtils.loadTrainingIms() in nato pretvorjene v 2D matriko z metodo convertToNormMat(). Pomaga, da imamo vizualno idejo iz česa je sestavljena ta imsMat matrika. Predvidevamo, da je vsaka slika $N \times N$ v sivini in da imamo M učnih slik, potem bo matrika izgledala podobno kot spodnja slika.



Slika 18: Vizualizacija imsMat matrike v makeBundle()

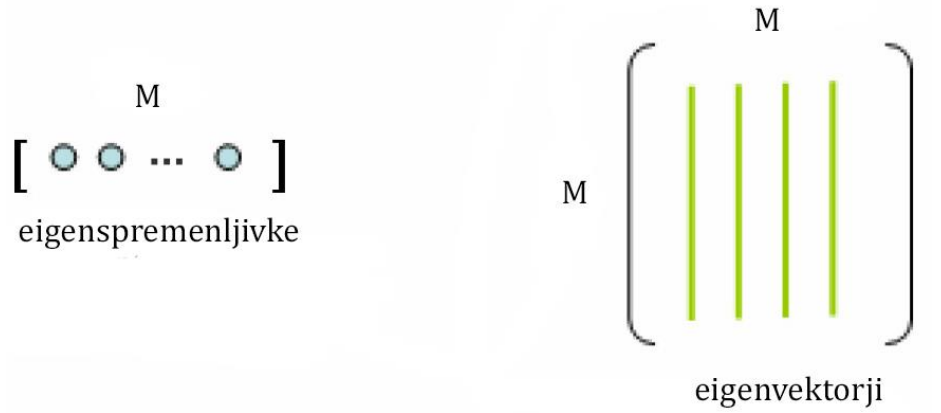
Za vsako vrstico je shranjena ena slika (ki je dolga N^2) in imamo M vrstic. Metoda convertToNormMat() normalizira matriko, kar poenostavi kovarianco in računanje eigenvrednosti, ki ga bomo izvedli kasneje.

Koda Javafaces ne uporablja Colt-ove Descriptive.coveriance() metode; namesto nje odšteje povprečje učnih slik od vsake slike in nato izračuna kovarianco z množenjem matrik slik s preneseno matriko slik. Koda zaposluje Turk in Pentland-ov trik za množenje $M \times N^2$ matrike z $N^2 \times M$ matrike za ustvarjanje manjše kovariančne matrike velikosti M x M, kot kaže spodnja slika.



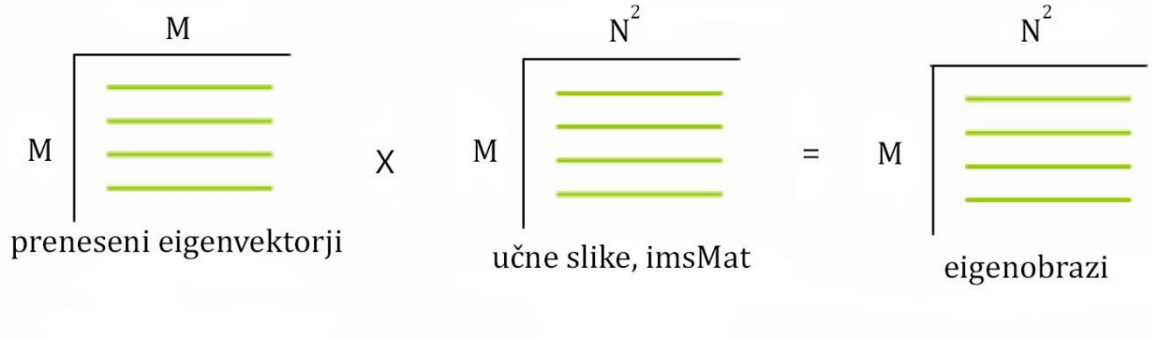
Slika 19: Računanje M×M kovariančne matrike

Eigenvektorji in eigenvrednosti so izračunani s klicem razreda EigenValueDecomp, ki je dokaj enostaven podrazred Colt-ovega EigenvalueDecomposition razreda. Eigenspremenljivke in eigenvektorji so sortirani v padajočem vrstnem redu z metodo sortEigenInfo(), ki spremeni ta dva polja, tako da izgledata podobno kot spodnja slika.



Slika 20: Polje eigenspremenljivk in eigenvektorjev

Poanta na zgornji sliki je, da je matrika eigenvektorjev velika M x M, kar pomeni, da še ne vsebuje vseh eigenvektorjev za to sliko (ki jo bomo od sedaj naprej imenovali eigenobrazi). Ker je vsaka slika vektor velikosti $N \times N$, mora imeti eigenobraz za sliko dimenzijo N^2. Zato moramo pretvoriti $M \times M$ eigenvektorsko matriko na sliki 20 v $M \times N^2$ eigenobrazovsko matriko, kot prikazuje slika 21.



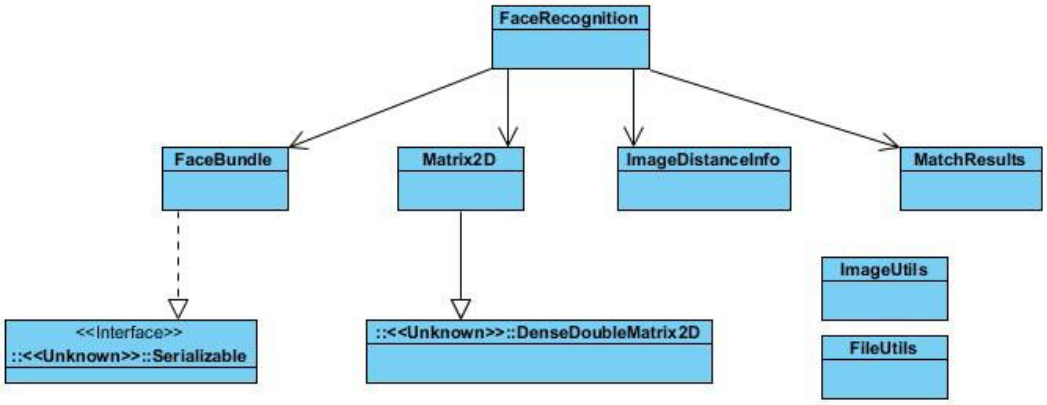
Slika 21: Pretvorba eigenvektorjev v eigenobraze

Transformacija je narejena v metodi getNormEgFaces(), ki pomnoži prenesene eigenvektorje z matrikami slik, da dobimo eigenobraze.

Na koncu makeBundle() metode, so eigenobrazi izpisani, vsak od njih je shranjen kot slika v eigenfaces/ direktoriju (kot na sliki 8). Še bolj pomembno pa je, da je kreiran objekt FaceBundle, ki vsebuje preračunane eigenobraze, eigenspremenljivke in učne slike.

6 Prepoznavanje nove slike

Drugi pomemben del naše spremenjene verzije Javaobrazov je koda, ki bere novo sliko (nov podatek) in se odloča, katera izmed učnih slik ji je najbolj podobna. Razred najvišjega nivoja je FaceRecognizer, ki uporablja številne iste podporne razrede kot BuildEigenFaces razred. Razredi so prikazani na spodnji sliki.

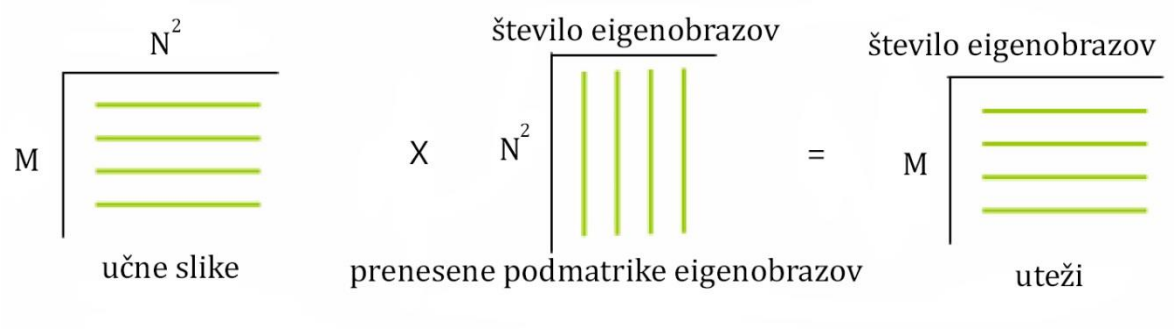


Slika 22: Razredi uporabljeni pri prepoznavanju nove slike

FaceRecognizer je kar različen od BuildEigenFaces, ampak si delita informacije preko eigen.cache datoteke. FaceRecognizer se začne z nalaganjem predpomnilniške datoteke in jo pretvori nazaj v FaceBundle objekt. Nato naloži novo sliko z uporabo metod iz razredov FileUtils in ImageUtils.

Konstruktor FaceRecognizer razreda uporablja metodo calcWeights() iz FaceBundle-a za kreiranje uteži za učne slike. To je narejeno že prej, kot pa v času gradnje, ker šele med samo prepoznavo obraza uporabnik pove število eigenobrazov, ki bodo uporabljeni med procesom prepoznave.

Potrebno število eigenobrazov je dobavljeno v spremenljivki numEFs (število eigenobrazov) in rezultirano polje uteži je izračunano z množenjem matrike slik s podmatriko eigenobrazov. Množenje je ilustrirano na spodnji sliki (spomnimo se, da predvidevamo, da je slika velika $N \times N$ pikslov in da imamo M učnih slik).



Slika 23: Ustvarjanje uteži za učne slike

En način za razumevanje teh uteži so kot nove koordinate od M učnih slik, potem ko so bile zavrtene, tako da je numEF eigenvektorjev poravnanih z osmi.

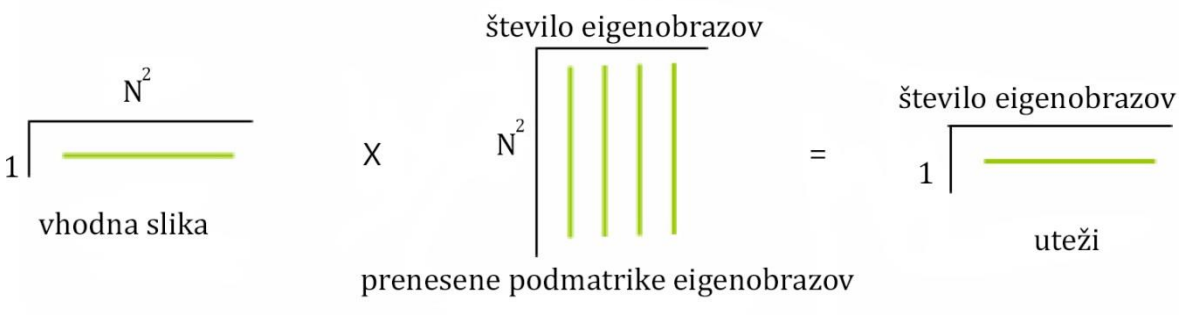
6.1 Iskanje zadetka

Kritična metoda v razredu FaceRecognizer je findMatch(), ki upravlja primerjavo nove slike s učnimi slikami. Zaposluje eigenobraze in eigenvektorje pridobljene iz objekta FaceBundle.

Nova slika je pretvorjena v enodimenzionalno polje pikslov dolžine N^2, nato je pretvorjena v normalizirano matriko z odšteto povprečno sliko obraza (imMat). To je v bistvu ista transformacija, ki je nanesena na učne slike na začetku metode BuildEigenFaces.makeBundle().

Slika je nato preslikana v eigenprostoru z metodo getImageWeights(), ki vrne njene rezultirane koordinate (ali uteži) kot matriko imWeights.

Samo določeno število numEf eigenobrazov so uporabljeni kot osi. Matrično množenje je prikazano spodaj:



Slika 24: Ustvarjanje uteži vhodne slike

Množenje je podobno tistemu v FaceBundle.calcWeights(). Razlika je, da je na sliki 24 samo ena slika preslikana, medtem ko so na sliki 23 preslikane vse.

FindMatch() izračuna Evklidovo razdaljo med novo sliko in učnimi slikami s primerjavo utežmi nove slike (v imWeights) z utežmi učnih slik. GetDists() vrne polje vsot kvadratne Evklidove razdalje med utežjo vnesene nove slike in utežmi vseh učnih slik; funkcija ne uporablja Colt-ove Statistics.distance() metode.

Najkrajša razdalja v polju je najdena s pomočjo metode getMinDistInfo, ki vrne tudi indeks pozicije ustrezne učne slike, ovite v ImageDistanceInfo objekt. Ta indeks je uporabljen, da pogledamo ime datoteke te učne slike. Skupaj sta ime in razdalja vrnjena v zgornjem nivoju kot MatchResult objekt.

6.2 Izvajanje prepoznave obraza

Ob prepoznavanju, ki se izvaja v razredu FaceRecognizer(), nov podatek primerjamo z vsemi slikami v naboru učnih slik in tako dobimo sliko, ki je najbolj podobna temu podatku. V našem primeru je to slika »Nejc1.pgn« (Ime »Nejc« je pridobljeno iz imena datoteke, vsa imena so namreč sestavljena iz imena in številke), meritev razdalje med njima pa je 0.4840.

Slika 25: Nov podatek in najbolj podobna slika »Nejc1.pgn«

Problem pri tem pristopu je interpretiranje pomena vrednosti razdalje. Kako blizu pravzaprav je 0.4840? Na žalost je to odvisno od posameznega nabora učnih slik, tako da moramo rezultirajoče eigenobraze testirati z več slikami, da lahko ugotovimo ali je 0.4840 zelo podobno, ali samo malo podobno.

Ta problem se lahko pojavi, če zaženemo kodo še enkrat, ampak s sliko, ki je ni v učnem naboru. Sliki »Miran.pgn je najbližje slika »Tilen.pgn«, in sicer je meritev razdalje med njima 0.6648.

Slika 26: Slika »Miran.png« in »Tilen.pgn«

Seveda to ujemanje ni pravilno, ampak kako lahko to zaznamo v kodi? Vse, na kar se lahko zanašamo, je meritev razdalje, ki je 0.6684. Očitno je, da če razdalja doseže takšno vrednost, moramo javiti, da ni zadetka, ampak je točni prag te vrednosti odvisen od učnega nabora in tipa slik, ki jih prepoznavamo.

7 Generiranje učnih slik

Učne slike generiramo pri registraciji nove osebe, in sicer ko vpišemo naše ime in geslo ter stisnemo gumb »Registriraj«. Ko ga pritisnemo, sproži boolean funkcija spremenljivko saveface v true, in sicer s klicem saveFace() metode v razredu FacePanel. Če je v trenutni sliki zaznan obraz in je saveFace true, potem je poklicana metoda clipSaveFace(), da shrani obraz v datoteko.

ClipSaveFace() dobi celotno sliko spletne kamere, nato jo pa sam obreže, spremeni velikost in jo naredi v sivini. Uporablja globalni faceRect kvadrat za obrezovanje.

SaveClip() spremeni velikost odrezka (ang. clip), tako da je standardne že prej definirane velikosti, primerne za prepoznavo obrazov, jo pretvori v sivino in jo potem zopet odreže, da se prepriča o standardni velikosti. Na koncu je slika shranjena v savedFaces/.

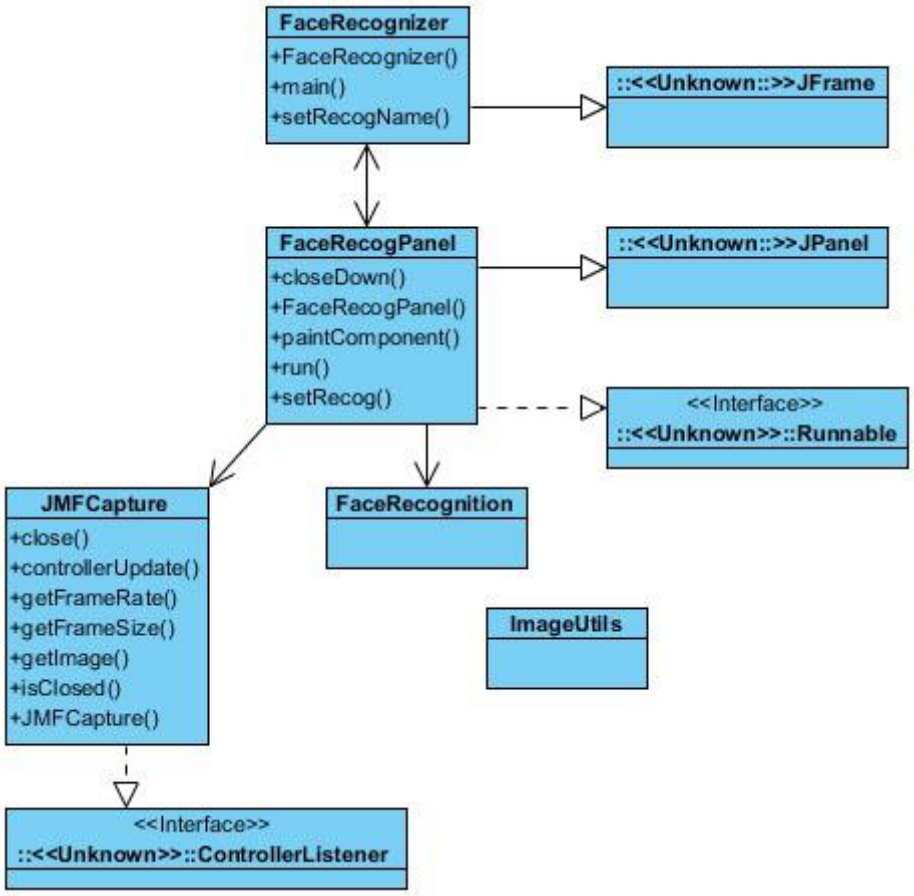
Ko je enkrat shranjenih več obrazov, morajo biti njihove datoteke kopirane v aplikacijo BuildEigenFaces, ki jih nato uporabi kot učne slike. Nujno je, da se datoteke preimenujejo, tako da vsebujejo ime osebe, ki ji sledi edinstvena številka.

8 Uporabniški vmesnik za prepoznavanje nove slike

Uporabniški vmesnik za prepoznavanje je že prikazan na sliki 6.

Ko uporabnik pritisne gumb »Vstop v program«, je trenutno izbrani obraz podan v razred FaceRecognition, nato pa vrne ime slike, ki ji je najbolj podobna (in razdaljo) in gre v program.

Spodaj prikazujemo UML diagram za aplikacijo.



Slika 27: Razredni diagram za FaceRecognizer

Prepoznava

Metoda Run FaceRecogPanel prikazuje trenutno sliko spletne kamere vsakih par milisekund in kliče TrackFace(), da najde obraz (in ga prepozna) v drugi niti. Spremenljivka recognizeFace je nastavljena na true, ko uporabnik pritisne »Vstop v program« − gumb na uporabniškem vmesniku. Glavna naloga metode recogFace() je, da uporabi pravokotnik za rezanje in izreže obraz iz slike spletne kamere. FaceRect je zaposlena znotraj sinhroniziranega bloka, ker je lahko posodobljena isti čas v drugih nitih. Metoda recogFace() poda izrezek za prepoznavo v matchClip(). MatchClip() se začne tako, da obreže in spremeni velikost sliki. Namesto, da bi klicala metodo saveImage(), poda sliko k FaceRecognition objektu, in sicer k faceRecog. Objekt FaceRecognition pa je kreiran v FaceRecogPanel-ovem konstruktorju. FaceRecognition.match() vrne objekt MatchResult, ki vsebuje ime in meritev razdalje [27].

3.4 Predstavitev programske rešitve Teniški organizator

Kot smo že omenili na začetku tega poglavja, smo prepoznavanje obrazov vključili v aplikacijo, s katero lahko osebe preko spleta rezervirajo ure na teniškem igrišču. Zaenkrat aplikacija deluje na Windows platformi, zaradi velike popularnosti tabličnih računalnikov pa imamo v prihodnje namen razviti to aplikacijo tudi za operacijski sistem Android.

Naše razvojno okolje je bil Eclipse Java EE, za bazo pa smo uporabili Oracle-ovo mySql Workbench. Samo overjanje uporabnikov pa poleg uporabniškega imena in gesla torej deluje s pomočjo prepoznavanja obrazov na osnovi algoritma Eigenobraz.

Ob zagonu programa se nam odpre začetni zaslon, kjer imamo na voljo registracije novega uporabnika in vpis v program. Če nas sistem še nima v bazi, se torej najprej registriramo in sicer tako, da vpišemo naš priimek in geslo, isti čas pa spletna kamera posname našo sliko, ki potem služi za prepoznavanje. Priimek in geslo se shranita v bazo, slika pa v mapo trainingImages. Ko smo enkrat vpisani v bazo, se lahko prijavimo. Ob prijavi vpišemo priimek in geslo, sistem pa preko kamere opravi še prepoznavanje obraza, kot smo ga opisali zgoraj. Če se v vseh parametrih ujemamo (priimek, geslo in slika), lahko vstopimo v program, v nasprotnem primeru smo zavrnjeni in nimamo dostopa. Seveda je možno, da smo prava oseba in nas program na podlagi napačnega prepoznavanja obraza zavrne, vendar se to ne zgodi pogosto.

V samem programu pa imamo možnosti rezervacije, preklica in ogleda terminov. Če je termin prost in kliknemo nanj, ga rezerviramo. Na termin, ki smo ga kliknili, se tako shrani naš priimek. Če je termin zaseden z naše strani in nanj pritisnemo, potem ga prekličemo, se pravi, se iz določenega termina zbriše naš priimek. Če pa ga je rezerviral drug uporabnik, katerega priimek je tudi izpisan na gumbu, s klikom ne naredimo nič. Uporabnike in termine lahko namreč briše le administrator. Pod »ogled terminov« si lahko termine bolj pregledno tudi ogledamo, če pa želimo, lahko le te tudi natisnemo. V primeru, da se odločimo, da ne bomo nikoli več igrali pri določenemu klubu, pa se lahko tudi izbrišemo iz baze.

V aplikaciji imamo še veliko prostora za dodajanje novih funkcionalnosti. Dodamo lahko na primer prijave na turnirje, prijave na tečaje, prikazujemo razne novice, rezultate ipd. Tako bi lahko aplikacijo kot komponento vključili v spletno stran kateregakoli teniškega kluba.

Kot administrator pa imamo na voljo nekaj več funkcionalnosti. Lahko namreč izbrišemo poljubnega uporabnika, tako da ga izberemo iz seznama vseh uporabnikov. Spreminjamo lahko datume, saj potrebujemo, da se v aplikaciji vsak teden izpiše pravi datum. Tedensko pa moramo tudi »počistiti« vse termine, tako da so vsi prosti in lahko uporabniki na novo rezervirajo ure.

Slika 28: Teniški organizator

4 SKLEP

Varno overjanje je še vedno eden izmed največjih problemov v računalništvu. To smo spoznali tudi med izdelovanjem diplomske naloge, saj smo pri vseh metodah našli pomanjkljivosti in slabosti, katerih se uporabniki poizkušajo izogniti. Na podlagi raziskovanja smo prišli do rezultatov, da je overjanje najmočnejše, če je večfaktorno. V tem primeru imajo nepridipravi neprimerljivo več dela, če želijo dostop, kajti pretentati morajo dve ali več metod, ki skrbijo za pravilen dostop do sistema oz. podatkov.

Tako tudi naša rešitev, uporablja večfaktorno overjanje. Večinoma sistemov za osebno uporabo v današnjem času ima namreč kot edino vrsto overjanja samo uporabniško ime in geslo. Z malo dela pa bi lahko te sisteme posodobili, tako da bi imeli zraven še prepoznavanje obrazov. Nenazadnje ima že praktično vsak računalnik spletno kamero, s katero bi to prepoznavanje omogočili.

To rešitev prepoznavanja smo nato vključili program, in ker imamo v lasti tenis igrišča, smo izbrali rezervacijo tenis igrišč preko interneta. Problema, da posamezni uporabnik ne ve, kdaj je kakšna ura prosta, pa nimajo samo igralci tenisa. Podobno moramo vedno po telefonu klicati tudi za ostale športe. Če bi bilo to vse na internetu, bi bilo mnogo lažje, preprosto kliknemo na gumb in že smo izbrali željen termin. Ker pa lahko samo z uporabniškim imenom in geslom pride do zlorab, smo dodali še prepoznavanje obrazov, na podlagi katerega lahko lastnik igrišč točno ve, kdo je kdo in lahko tej osebi prepreči igranje v primeru, da si je večkrat rezervirala kakšno uro, pa potem ni prišla.

Pri samem prepoznavanju obrazov pa smo opazili tudi kar nekaj pomanjkljivosti. Ob registraciji osebe bi bilo namreč optimalno, če bi vsak uporabnik naredil čim več slik z različnimi izrazi, brki, očali ipd. Sam algoritem najbolj pravilno namreč prepozna osebo ravno, če ima veliko slik za primerjati, v nasprotnem primeru rad naredi napako in prepozna napačno osebo. Prav tako smo imeli težave s tem, da smo podali pravo vrednost za prag meritve razdalje med dvema slikama. Ta prag smo nastavili na 0.5, tako na naših testnih primerih algoritem deluje kar se da pravilno, a vendar včasih imata dve osebi, ki nista isti, to razdaljo manjšo kot 0.5, kar da seveda napačne rezultate. Zgodilo se je pa tudi, da je imela ista oseba meritev razdalje nad 0.5 (na testni sliki ima oseba drug izraz, očala

ipd. …). Seveda je tudi to napačen rezultat, ampak so te napake tako redke, da jih lahko zanemarimo.

Na splošno smo spoznali, da je overjanje na podlagi prepoznavanja obrazov zadovoljiv in varen način overjanja, posebej, če ga vključimo skupaj s kakšno drugo vrsto in prepričani smo, da bo v prihodnje vse več podobnih aplikacij, kot smo jo razvili sami v tej diplomski nalogi.

5 VIRI IN LITERATURA

1. Chirillo J., Blaul S., Implementing biometric security, Indianapolis, Wiley Red Books, 2003.
2. Andress J., The Basics of Information Security, Waltham, Syngress, 2011.
3. Anil K. Jain, Biometric Authentication, dostopno na: http://www.scholarpedia.org/article/Biometric_authentication [21. 3. 2013].
4. Reid P., Biometrics for network security, New Jersey, Prentice Hall PTR, 2003.
5. Vacca J. R., Biometric tehnologies and verification systems, Butterworth-Heinemann/Elsevier, Amsterdam, 2007.
6. Palm Print and Footprint Identification, dostopno na: http://biometrics.pbworks.com/w/page/14811351/Authentication%20technologies#PalmPrintAndFootprintIdentification [21. 3. 2013].
7. Java Media Framework, dostopno na: http://www.oracle.com/technetwork/java/javase/download-142937.html [21. 3. 2013].
8. Passwords, dostopno na: http://en.wikipedia.org/wiki/Password [21. 3. 2013].
9. Security Token, dotopno na: http://en.wikipedia.org/wiki/Security_token [21. 3. 2013].
10. Single Sign On Authentication, dostopno na: http://www.authenticationworld.com/Single-Sign-On-Authentication/ [21. 3. 2013].
11. OpenCV, dostopno na: http://opencv.willowgarage.com [21. 3. 2013].
12. JavaCV, dostopno na: http://code.google.com/p/javacv/ [21. 3. 2013].
13. Colt, dostopno na: http://acs.lbl.gov/software/colt/ [21. 3. 2013].
14. JavaFaces; dostopno na: http://code.google.com/p/javafaces/ [21. 3. 2013].
15. Top 10 Passwords, dostopno na: http://www.zdnet.com/the-top-10-passwords-from-the-yahoo-hack-is-yours-one-of-them-7000000815 [21. 3. 2013].
16. Top 10 PIN codes, dostopno na: http://www.macrumors.com/2011/06/13/ten-most-common-iphone-passcodes-revealed/ [21. 3. 2013].
17. Passphrase, dostopno na: http://www.iusmentis.com/security/passphrasefaq/ [21. 3. 2013].
18. Passphrases vs. Passwords, dostopno na: http://technet.microsoft.com/library/cc512609 [21. 3. 2013].

19. Bonneau J., Shutova E., Linguistic propertise of multi-word passphrases, dostopno na: https://www.cl.cam.ac.uk/~jcb82/doc/BS12-USEC-passphrase_linguistics.pdf [21. 3. 2013].
20. ID number, dostopno na: http://webb-site.com/articles/identity.asp [21. 3. 2013].
21. ATM Machine, dostopno na: http://news.bbc.co.uk/2/hi/business/6230194.stm [21. 3. 2013].
22. One Time Password, dostopno na: http://en.wikipedia.org/wiki/One-time_password [21. 3. 2013].
23. Prstni odtis, dostopno na: http://sl.wikipedia.org/wiki/Prstni_odtis [21. 3. 2013].
24. Security Token, dostopno na: http://bits.blogs.nytimes.com/2012/06/25/computer-scientists-break-security-token-key-in-record-time/ [21. 3. 2013].
25. EigenFaces, dostopno na: http://www.cs.princeton.edu/~cdecoro/eigenfaces/eigenfaces.jpg [21. 3. 2013].
26. Security Tokens, dostopno na: http://upload.wikimedia.org/wikipedia/commons/thumb/d/db/SecurityTokens.CryptoCard.agr.jpg/640px-SecurityTokens.CryptoCard.agr.jpg [21. 3. 2013].
27. Face Recognition, dostopno na: http://fivedots.coe.psu.ac.th/~ad/jg/?? [21. 3. 2013].

Univerza v Mariboru
Fakulteta za elektrotehniko, računalništvo in informatiko

IZJAVA O AVTORSTVU

diplomskega dela

Spodaj podpisani/-a **Jernej Mlakar,**

z vpisno številko **93593769,**

sem avtor/-ica diplomskega dela z naslovom:

OVERITVENI SISTEMI ZA OSEBNO UPORABO

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal/-a samostojno pod mentorstvom (naziv, ime in priimek)

  doc. dr. Marko Hölbl

- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela

- soglašam z javno objavo elektronske oblike diplomskega dela v DKUM.

V Mariboru, dne 4.4.2013

Podpis avtorja:

Univerza v Mariboru
Fakulteta za elektrotehniko, računalništvo in informatiko

IZJAVA O USTREZNOSTI DIPLOMSKEGA DELA

Podpisani mentor **Marko Hölbl** izjavljam, da je

Študent **Jernej Mlakar** izdelal

diplomsko delo z naslovom:

OVERITVENI SISTEMI ZA OSEBNO UPORABO

v skladu z odobreno temo diplomskega dela, Navodili o pripravi diplomskega dela in mojimi navodili.

Datum in kraj: Podpis mentorja:

Maribor, 4.4.2013

Univerza v Mariboru

Fakulteta za elektrotehniko, računalništvo in informatiko

IZJAVA O ISTOVETNOSTI TISKANE IN ELEKTRONSKE VERZIJE ZAKLJUČNEGA DELA IN OBJAVI OSEBNIH PODATKOV DIPLOMANTOV

Ime in priimek avtorja:	**Jernej Mlakar**
Vpisna številka:	**93593769**
Študijski program:	**UN ŠP – računalništvo in informatika**
Naslov zaključnega dela:	**Overitveni sistemi za osebno uporabo**
Mentor:	**doc. dr. Marko Hölbl**

Podpisani **Jernej Mlakar** izjavljam, da sem za potrebe arhiviranja oddal elektronsko verzijo zaključnega dela v Digitalno knjižnico Univerze v Mariboru. Zaključno delo sem izdelal-a sam-a ob pomoči mentorja. V skladu s 1. odstavkom 21. člena Zakona o avtorskih in sorodnih pravicah dovoljujem, da se zgoraj navedeno zaključno delo objavi na portalu Digitalne knjižnice Univerze v Mariboru.

Tiskana verzija zaključnega dela je istovetna z elektronsko verzijo elektronski verziji, ki sem jo oddal za objavo v Digitalno knjižnico Univerze v Mariboru.

Podpisani izjavljam, da dovoljujem objavo osebnih podatkov, vezanih na zaključek študija (ime, priimek, leto in kraj rojstva, datum zaključka študija, naslov zaključnega dela), na spletnih straneh in v publikacijah UM.

Datum in kraj:

Maribor, 4.4.2013

Podpis avtorja: