Ansiktsigenkänning med eigen faces

Ett projektarbete i TNM034 vid Linköpings Universitet

Olivia Enroth, Linus Hjeltman, Fredrik Burmester

15 december 2020

Innehåll

1 Inledning

Denna rapport täcker utvecklingen av ett program för ansiktsigenkänning utvecklat i MATLAB. Arbetet är ett projekt i kursen TNM034 vid Linköpings Universitet och syftet är att utveckla ett program som utifrån en bild på ett okänt ansikte kan avgöra om det ansiktet finns med i en databas med ansikten.

Det finns ett antal krav på programmet. Programmet ska kunna klassificera ett ansikte i en bild som är:

- Roterad +/- 5 grader
- Skalad +/- 10%
- Intensitetsförändrad +/- 30%
- En kombination av de tre ovanstående

1.1 Dataset

Datasetet som används i detta projektet är en delmängd av en databas som Stanford University tagit fram och som i detta fall delades in i två mängder: DB0 och DB1. DB1 innehåller 16 porträttbilder på olika individer som tillhör träningsmängden, vilken bildar databasen som nya bilder jämförs med. DB0 innehåller 4 porträttbilder på personer som inte finns med i träningsmängden, vilken är en testmängd för att se till att inte klassificera fel.

2 Metod

Implementationen av programmet är uppdelad i 3 olika delmoment som beskrivs i detta avsnitt: förbehandling av bilden, att urskilja ansiktet och klassificering av bilden.

I det första steget sker en förbehandling av bilderna. Syftet är att både inbilden, alltså bilden på det okända ansiktet, och bilderna i databasen ska ha ett liknande utseende så att gränsvärden kan väljas för att kunna urskilja hudtoner. Både ljusintensitet och färgkorrigering sker med intensitetskompensationen *white patch*.

Efter förbehandlingen skapas en *face mask* som beskriver vart ansiktet befinner sig genom att hitta de pixlar som representerar hudtoner. För att kunna jämföra inbilden på det okända ansiktet med ansiktena i databasen krävs att ögonens position är kända för alla bilder. Positionen tas fram via en kombination av en *face mask* och en *eye map*. När positionen för ögonen är kända är nästa steg att beskära alla bilder så att de blir kvadratiska. Alla bilder beskärs så att upplösningen blir densamma samt att avståndet mellan ögonen blir densamma. När ansiktet och positionen av ögonen är normaliserade kan *eigen faces* beräknas som används vid klassificeringen.

Klassificeringen av ett nytt ansikte mot databasen sker med metoden *Principal component analysis* (PCA) [5]. I denna metod skapas *eigen faces* som utnyttjas för att skapa vikter för varje bild i databasen samt för inbilden, vilket möjliggör jämförelse mellan dessa två bilder. Vikterna jämförs mot inbildens vikter och ett beslut tas om skillnaden mellan bilderna är liten nog för att klassificera det nya ansiktet i inbilden som en del av databasen.

Den bild som används genom rapporten för att demonstrera vad som händer i de olika stegen av implementationen visas i Figur 1 innan behandling. Behandlingen som utförs på denna bild sker på alla bilder i databasen samt på inbilden som testas mot databasen.

Figur 1: Den obehandlade inbilden som jämförs mot databasen

2.1 Förbehandling av bilderna

För att algoritmen som hittar ansiktet i bilden ska fungera korrekt krävs det att bilderna först behandlas så att vitbalansen och intensiteten i bilderna från databasen är normaliserade. Detta går att göra på flera olika sätt. Två lösningar implementerades för att sedan kunna välja den lösning som gav bäst resultat i efterkommande delmoment av implementationen.

2.1.1 Gray world intensitetskompensation

Gray world baseras på antagandet att medelvärdet av intensiteten i de tre kanalerna i en RGB-bild ska vara lika. Om medelvärdet i de tre kanalerna är tillräckligt lika redan i originalbilden modifieras inte bilden. Om medelvärdena däremot skiljer sig skalas två av kanalerna med skalfaktorerna givna i ekvation 1,

$$\begin{aligned} \alpha &= \frac{G_{avg}}{R_{avg}} \\ \beta &= \frac{G_{avg}}{B_{avg}} \end{aligned} \tag{1}$$

där G_{avg} anger medelvärdet för G-kanalen, R_{avg} anger medelvärdet för R-kanalen och B_{avg} anger medelvärdet för B-kanalen.

De nya värdena för R-, G- och B-kanalen ges av $\hat{R}_{sensor}$, $\hat{G}_{sensor}$ och $\hat{B}_{sensor}$ enligt ekvation 2,

$$\begin{aligned} \hat{R}_{sensor}(x,y) &= \alpha R_{sensor}(x,y) \\ \hat{G}_{sensor}(x,y) &= G_{sensor}(x,y) \\ \hat{B}_{sensor}(x,y) &= \beta B_{sensor}(x,y) \end{aligned} \tag{2}$$

där R_{sensor}, G_{sensor} och B_{sensor} är de orörda kanalerna.

I figur 2 visas bilden efter att *gray world* är applicerad.

Figur 2: Bilden efter intensitetskompensation med gray world

2.1.2 White patch intensitetskompensation

White patch baseras på antagandet att den ljusaste delen av bilden bör vara helt vit [3]. Utifrån detta påstående skalas de olika kanalerna med skalfaktorerna givna i ekvation 3,

$$\begin{aligned} \alpha &= \frac{G_{max}}{R_{max}} \\ \beta &= \frac{G_{max}}{B_{max}} \end{aligned} \tag{3}$$

där R_{max}, G_{max} och B_{max} är intensiteten i en liten del av bilden. I detta projekt valdes denna del som 5% av de olika kanalerna.

Skalningen sker sedan enligt ekvation 4,

$$\begin{aligned} \hat{R}_{sensor}(x,y) &= \alpha R_{sensor}(x,y) \\ \hat{G}_{sensor}(x,y) &= G_{sensor}(x,y) \\ \hat{B}_{sensor}(x,y) &= \beta B_{sensor}(x,y) \end{aligned} \tag{4}$$

där R_{sensor}, G_{sensor} och B_{sensor} är R-, G- och B-kanalen innan skalningen och $\hat{R}_{sensor}$, $\hat{G}_{sensor}$ och $\hat{B}_{sensor}$ är kanalerna efter skalningen.

Figur 3: Bilden med white patch intensitetskompensation

2.2 Urskilja ansiktet

När intensiteten i bilden korrigerats är nästa steg att hitta ansiktet i bilden genom att ta fram en *face mask*. En *face mask* anger var ansiktet är och används i kombination med en *eye map* för att med hög säkerhet kunna säga vart ögonen befinner sig i bilden. Innan man kan jämföra inbilden med ansiktena i databasen krävs det även att bilderna roteras och skalas.

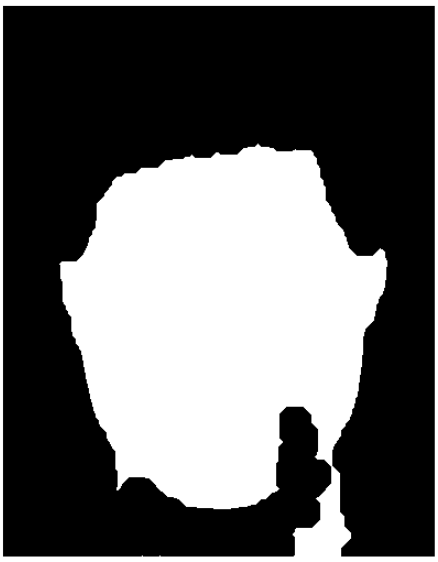
2.2.1 Face mask

För att ta fram en *face mask* krävs att programmet hittar de pixlar som representerar hud. Hudtoner kan skilja sig väldigt mycket åt, dock skiljer sig hudtoner mer i intensitet och mindre i krominans [2]. Därför används färgrymden YCbCr, där krominansen särskiljs från luminansen.

Två kopior av RGB-bilden skapas, dessa konverteras till färgrymderna HSV och YCbCr. Dessa kopior används sedan för att skapa maskerna. Om en pixel i bilden inte uppfyller kriterierna i HSV samt YCbCr kanalerna sätts det pixelvärdet till 0. Pseudokoden nedan beskriver denna behandling av bilden med de värden som användes i detta projekt.

```
1 if (H_xy > 0.9 || H_xy < 0.85 || Cb_xy > ...
      130 || Cb_xy < 100)
2     P_xy = 0;
3 end
```

Bilden görs sedan binär för att kunna användas som mask, vilken visas i figur 4.



Figur 4: Den binära facemasken

Figur 5: Den binära eye mapen

2.2.2 Eye map

En *eye map* skapas för att bestämma var ögonen befinner sig i bilden. För att hitta ögonen kan man utnyttja variationerna i krominans och luminans kring ögonen [4].

Bilden omvandlas till färgrymden YCbCr, där Y-kanalen beskriver luminansen i bilden och Cb- och Cr-kanalerna beskriver krominansen. Det tas fram två olika masker som därefter kombineras.

Luminansmasken tas fram enligt ekvation 5,

$$Mask_L = \frac{Y(x,y) \oplus g_\sigma(x,y)}{Y(x,y) \ominus g_\sigma(x,y) + 1} \tag{5}$$

där $g_\sigma(x,y)$ anger strukturelementet som används vid erosion och dilation; $\oplus$ och $\ominus$. $Y(x,y)$ anger Y-kanalen i punkten (x,y).

Krominansmasken som tas fram från Cb- och Cr-kanalerna och som kombineras med Luminansmasken för att få fram en eye map, ges av ekvation 6,

$$Mask_K = \frac{C_b^2 + \tilde{C_r}^2 + \frac{C_b}{C_r}}{3} \tag{6}$$

där $\tilde{C_r}$ är Cr-kanalen inverterad.

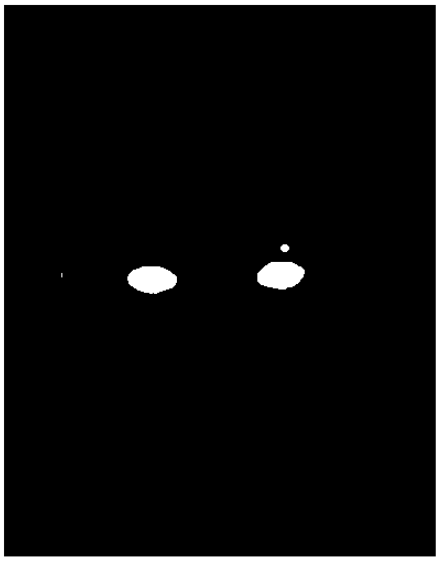
De båda maskerna kombineras sedan via elementvis multiplikation enligt ekvation 7.

$$EyeMap = Mask_L . * Mask_K \tag{7}$$

Därefter normaliseras samt trösklas bilden för att få fram en binär *eye map* som kan ses i figur 5.

2.2.3 Face mask och eye map i kombination

För att endast behålla ögonen i masken används den framtagna *face masken* och *eye mapen* i en kombination, som kan ses i figur 6.



Figur 6: Kombinationen av face mask och eye map

Genom att utnyttja det som är känt för inbilden kan falska positiver för ögon och ansikte elimineras med hård klippning. Eftersom ansiktet förväntas vara i mitten av bilden kan pixlarna längs med kanterna klippas bort för att förbättra maskerna.

För att försäkra att ögonmasken valde just ögonen sattes gränser för hur nära, alternativt hur långt ifrån, avståndet var mellan ögonen. Dessutom undersöktes höjdskillnaden mellan ögonen.

2.2.4 Rotation och skalning

Både för att kunna utesluta oönskade objekt i bakgrunden av bilden samt för att kunna jämföra ansiktenas egenskaper, krävs det att ögon, näsa och mun befinner sig på samma plats i bilden. Bilden roteras och skalas för att ögonen alltid ska befinna sig vågrätt i förhållande till varandra och med ett fast pixel-antal mellan pupillerna. Den roterade och skalade bilden kan ses i figur 7

Figur 7: Den roterade bilden

2.3 Klassificering

För att kunna avgöra om inbilden matchar ett ansikte i databasen måste en klassificering göras. Varje bild i databasen får vikter som representerar hur lik inbilden är medianen av alla bilder i databasen. Med hjälp av dessa vikter kan en jämförelse göras mot ett nytt ansikte som testas mot databasen.

Varje bild i databasen jämförs med en medianbild och ett antal vikter skapas. En bild jämförs sedan med vikterna för att avgöra om det är samma ansikte i bilden eller inte.

2.3.1 Egenansikten och PCA

För att kunna jämföra ansikten skapas först en medianbild μ utifrån alla bilder i databasen enligt ekvation 8,

$$\mu = \frac{1}{M}\sum_{i=1}^{M} x_i \tag{8}$$

där x_i är en bild i databasen och M är antalet bilder. Därefter subtraheras medianbilden från varje bild i databasen. Detta skapar en matris A vars normaliserade egenvektorer med högst egenvärde korresponderar till de delarna av A som påverkar bildens utseende mest. Vikterna $w \in IR^{i\times j}$ för en bild beräknades genom att ta varje kolumn i matris A multiplicerat med de egenvektorerna enligt ekvation 9 och sparas i Ω som visas i 10.

$$w = u^T A \tag{9}$$

$$\Omega = \sum_{i=1}^{M} w_i \tag{10}$$

För att hitta vilken bild i databasen som matchar inbilden jämförs vikterna med varandra som visas i 11. Där det minsta e motsvarar den bild i databasen som mest liknar inbilden.

$$e_j = \|\Omega - \Omega_j\| \tag{11}$$

Dock kommer det alltid finnas en bild med kortast avstånd, därför behövs ett gränsvärde som anger hur mycket en bild får avvika för att den klassificeras som ett ansikte som inte finns i databasen. Med gränsvärdet satt, returnerar programmet id-numret för den bild i databasen som matchar inbilden samt har ett värde e under gränsvärdet. Om inbilden har ett för högt e-värde returnerar programmet en nolla, vilket representerar att ingen matchning skett.

3 Resultat

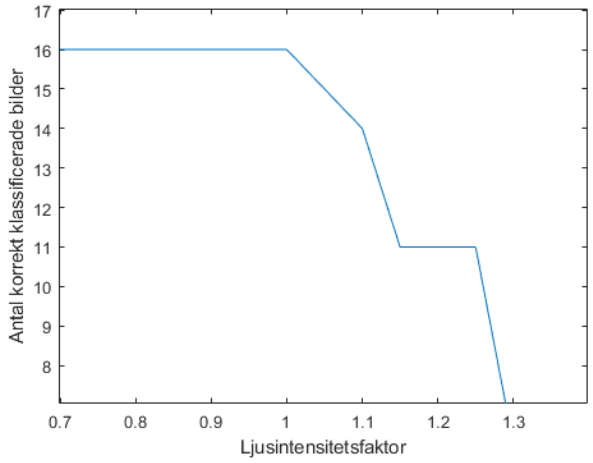
I avsnitt 2 presenterades två olika sätt att kompensera för intensiteten i en bild; *gray world* och *white patch*. Det visade sig att metoden *white patch* gav bättre resultat för alla bilder i databasen, medan *gray world* gav väldigt varierande resultat mellan olika bilder. Därför valdes *white patch* som den metod som användes vid förbehandling av bilderna.

Oavsett om *white patch* gav ett bättre resultat än *gray world*, var det inte ett felfritt sätt att korrigera intensiteten i bilden. Programmet klarade av att klassificera alla ansikten i DB1 oavsett om de var roterade +/- 5 grader, skalade +/- 10% eller intensitetsförändrade med upp till -30%, men när inbilden istället var intensitetsförändrad med upp till +30% var det många bilder som inte kunde klassificeras. Resultatet av testkörningen där inbilderna intensitetsförändrats med 30% kan ses i tabell 1.

Tabell 1: Resultat vid +30% ljushetsförändring.

Inbild	Klassificering	Skillnad mellan vikter
1	1	8,0131
2	2	9,9219
3	3	11,5739
4	4	7,3119
5	5	6,4624
6	6	4,9300
7	7	6,0659
8	0	28,6736
9	0	27,2327
10	0	20,6845
11	11	11,4093
12	12	11,7041
13	0	20,4265
14	0	26,9902
15	15	10,0064
16	16	3,4776

I figur 8 visas tydligt hur antalet korrekt klassificerade bilder beror av intensiteten i bilden. Då ljusintensiteten höjs med 30% kan programmet inte längre klassificera bilderna.



Figur 8: Korrekt antal klassificerade bilder i DB1 mot ljusintensitet

Figur 9 visar ett exempel på en bild med en höjd ljusintensitet.

Figur 9: Bilden med en intensitetsförändring på +30%

För att testa hur bra programmet presterade gentemot kraven som ställts, vilka listades i avsnitt 1, utfördes ett test där bilderna från DB1 behandlades och sedan jämfördes mot de obehandlade bilderna i DB1 med 500 iterationer per bild (totalt 8000 iterationer). Behandlingen som gjordes var slumpmässig inom intervallen +/- 5 graders rotation, +/- 10% skalning och +/- 30% intensitetsförändring. Programmet klarade av att klassificera 91,47% av de behandlade bilderna med noll felklassificeringar.

Testresultatet av de 4 bilderna från DB0 jämförda mot databasen DB1 kan ses i tabell 2. Inga av ansiktena i DB0 återfinns i DB1 och programmet lyckades klassificera dem utan problem.

Tabell 2: Resultat med 4 bilder från DB0 (ansikten som inte finns i databasen)

Inbild	Klassificering	Skillnad mellan vikter
1	0	25.5945
2	0	25.9531
3	0	21.4888
4	0	29.0944

4 Diskussion

Det var svårt att sätta ett bra gränsvärde då ett litet gränsvärde hade nekat en korrekt klassificering och ett för stort hade släppt igenom felaktiga klassificeringar. För att sätta ett bra gränsvärde felsöktes koden där observationen som gjordes var att ansikts- samt ögon-masken gjorde sitt rätt, problemet var ljusintensiteten i bilderna. Normaliseringen av ljuset var inte tillräcklig. Det går inte att kompensera för en för hög ljusintensitet om en pixel är helt vit, då all information som fanns där har gått förlorad.

Att välja gränsvärde för vad som är en godkänd klassificering var väldigt viktigt. Från början valdes gränsvärdet till det största värdet på en rätt klassificerad bild från DB1. Vi upptäckte snabbt att det gränsvärdet var för högt och ett striktare gränsvärde behövdes. Anledningen var, som nämndes i avsnitt 2, att en bild alltid kommer ha ett lägsta gränsvärde, trots att det är väldigt högt jämfört med andra gränsvärden. Därför valdes det nya gränsvärdet till ett värde som aldrig felklassificerade en bild i DB1 samt alltid klassificerade bilderna från DB0 som ej tillhörande databasen. Det gränsvärdet gjorde att några bilder bedömdes inte finnas i DB1 trots att de återfanns i databasen, men vi ansåg att det var bättre att aldrig klassificera fel och säga att personen inte tillhör, än att klassificera en in-bild som fel person.

Vid vidareutveckling av projektet skulle andra metoder för klassificering kunna testats, exempel på detta är så kallade *fisher faces*. Där är tanken att ljussättning inte är lika kritisk som den är i metoden med *eigen faces* [6]. Det hade även varit intressant att vidare undersöka olika typer av masker för att hitta ansikten och hudtoner, även om maskerna som togs fram i detta projekt var tillräckligt bra för att uppfylla de krav som fanns på programmet.

4.1 Slutsats

Ett program för ansiktsigenkänning implementerades utifrån några specifika krav på programmet; Programmet skulle klara av att klassificera en bild som var roterad +/- 5 grader, skalad +/- 10% och intensitetsförändrad med +/-30%. De första två kraven uppfylldes var för sig, men programmet klarade inte av att klassificera bilder med en positiv intensitetsförändring på upp till 30%. När rotation, skalning och intensitetsförändring kombinerades lyckades programmet klassificera 91,47% av fallen, och de klassificeringar

som inte lyckades berodde på intensitetsvariationer mellan bilderna.

Att klassificera ansikten med *eigen faces* är möjligt så länge inbilden följer de stränga krav som algoritmen klarar av. Metoden PCA är dock känslig för skillnader i intensitet. För att få en ökad korrekt klassificering krävs noggrann förbehandling och normalisering.

Referenser

[1] Wikipedia, 2020, *Color normalization*, https://en.wikipedia.org/wiki/Color_normalization Hämtat: 2020-12-15.

[2] Jose M. Chaves-González, Miguel A. Vega-Rodríguez, Juan A. Gómez-Pulido,Juan M. Sánchez-Pérez, *Detecting skin in face recognision systems: A colour spaces study*, Digital signal processing, 20(3) p.806-823, 2009.

[3] Rein-Lein Hsu, Mohamed Abdel-Mottaleb, Anil K. Jain, *Face Detection in Color Images*, IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, 24(5), 2002.

[4] M. Shafi, P.W.H. Chung, *A hybrid method for eyes detection in facial images*, Proceedings of World Academy of Science, Engineering and Technology International Conference on Computer Science Singapore, (32)99-104, 2008.

[5] Sergio Bacallado, Jonathan Taylor, *Principal Components Analysis* Stanford University, https://web.stanford.edu/class/stats202/notes/Unsupervised/PCA.html Hämtad: 2020-12-15.

[6] Peter N. Belhumeur, Joao P. Hespanha, and David J. Kriegman, *Eigenfaces vs. Fisherfaces: Recognition Using Class Specific Linear Projection*, IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, 19(7), 1997.