Linköpings universitet
TEKNISKA HÖGSKOLAN

TNM034 Avancerad Bildbehandling

29 november 2012

Hans-Christian Helltegen Alexander Johansson

Sammanfattning

Detta är en projektrapport för kursen TNM034- Avancerad Bildbehandling. Uppgiften för projektet var att identifiera ansikten ur en databas genom att implementera valfri metod för automatiskt ansiktsigenkänning i Matlab. Metoden som används i denna rapport är Eigenfaces. Genom att läsa in hela databasen med bilder som ska kunna identifieras och generera egenansikten för varje bild. När en bild ska identifieras mot databasen så beräknas det euklidiska avståndet mellan bildens egenansikte och alla andra bilders egenansikten i databasen. De bilderna med kortast avstånd mellan varandra är mest lika varandra.

Innehållsförteckning

Figurer

1

Syfte

Syftet med projektet är att implementera ett matlabprogram för automatisk ansiktsigenkänning. Programmet ska kunna identifiera en bild och returnera rätt id utifrån en databas av bilder samt returnera falskt om bilden inte existerar i databasen.

2

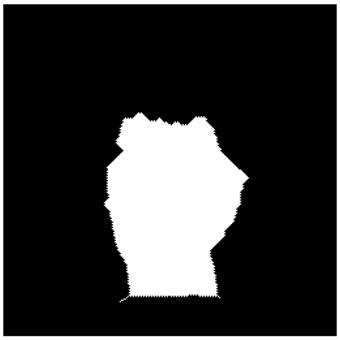
Metod

2.1 Förbehandling

För att alla bilder ska vara i samma format måste bilderna från databasen och bilden som ska identifieras förbehandlas. Bilderna förbehandlas för att programmet ska klara av manipulerade bilder av olika slag så som rotation, translation, tonvärdesändring och skalning. Detta görs genom att först generera en mask för ansiktet i bilden som baseras på hudfärgen (Figur 2.1), när masken har applicerats på bilden så återstår bara ansiktet. För att klara av tonvärdesändringar på bilderna så konverteras bildens färgrymd från RGB till HSV för att sedan spara H kanalen för att sedan konvertera bilden från HSV till YCbCr.

Figur 2.1: Bild efter att huden hittats

När denna konvertering är gjord kollas varje pixel i bilden mot tröskelvärden i H-,Cb- och Cr-kanalen och tar bort pixlar som inte ligger inom intervallen. Masken förbättras sedan genom en rad olika morfologiska operationer (Figur 2.2).



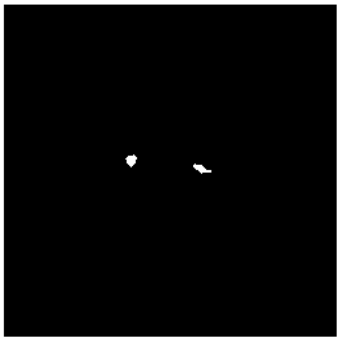
Figur 2.2: Bild på masken

För att sedan hitta ögonen i den maskade bilden så används ekvation 2.1 [1]. Resultatet ses i figur 2.3.

$$EyeMapC = \frac{1}{3}(C_b^2 + (1 - C_r)^2 + (C_b/C_r)) \tag{2.1}$$

Figur 2.3: Bild på EyeMapC

Bilden i figur 2.3 bearbetas sedan för att få ut ögonen genom att kombinera figur 2.2 och 2.3. Genom att utföra tröskling och morfologiska operationer så uppnås figur 2.4.



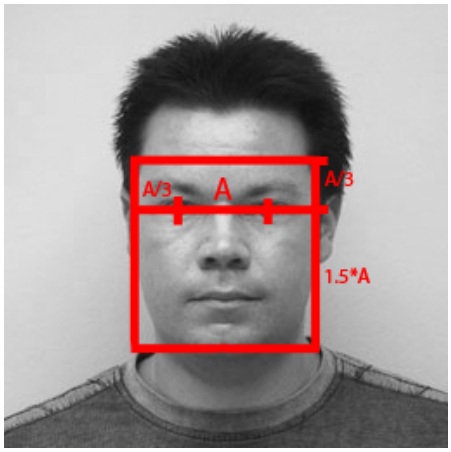
Figur 2.4: Bild på färdig ögonmask

När ögonen är identifierade roteras bilden så att ögonen är vinkelräta mot varandra (Figur 2.5 och 2.6).

Figur 2.5: Ögon hittade, innan rotation

Figur 2.6: Efter rotation

Efter att bilden har roterats så beskärs bilden genom att använda avståndet mellan pupillerna och sedan ansiktssymmetri (Figur 2.7 och 2.8).



Figur 2.7: Symmetri i ett ansikte efter pupillavstånd

Figur 2.8: Bild efter beskärning

2.2 Eigenfaces

Metoden bygger upp ett träningset med hjälp av bildernas egenvektorer på personer som ska identifieras. Sedan jämnförs de inkommande bildernas egenvektorer mot träningsdatan. För att identifiera egenskaper hos de olika ansiktena så konstrueras ett medelansikte för alla bilder i träningsdatan. Detta görs genom att ta medelvärdet av varje pixel och lägga ihop till en bild (figur 2.9).

Figur 2.9: Medelansiktet

För att sedan hitta den vektor som bäst beskriver fördelningen av datan i bilden används ekvationerna 2.2 och 2.3 [2]. Vektorn är vald så att λ_k maximeras.(u_k och λ_k är egenvektorer från kovariansmatrisen)

$$\lambda_k = \frac{1}{M}\sum_{n=1}^{M}(u_k^T\Phi_n)^2 \tag{2.2}$$

$$u_l^T u_k = \delta_{lk} = \begin{Bmatrix} 1 & om\ l = k \\ 0 & annars \end{Bmatrix} \tag{2.3}$$

Koviariansmatrisen räknas ut med ekvationen 2.4, där Ψ är medelansiktet och I innehåller alla bilder.

$$\begin{gathered} \Phi_i = I_i - \Psi \\ A = \{\Phi_1, \Phi_2, ..., \Phi_n\} \\ C = AA^T \end{gathered} \tag{2.4}$$

När egenvektorerna är uträknade kan ett egenansikte skapas utifrån egenvektorerna för varje bild. 2.10.

Figur 2.10: Egenansikte för en bild

2.3 Identifiering

När en bild ska identifieras mot träningsdatan förbehandlas den som beskrivet i sektion 2.1. När bilden har förbehandlats subtraheras Ψ för att hitta skillnaden. Skillnaden multipliceras sedan med egenvektornerna ur kovariansmatrisen för att hitta den relevanta datan i bilden. Varje värde representerar en vikt och sparas i en vektor Ω. För att sedan hitta vilken bild som matchar bilden som ska identifieras bäst så minimeras det euklidiska avståndet(ekvation 2.5).

$$\varepsilon_k = \|\Omega - \Omega_k\|^2 \tag{2.5}$$

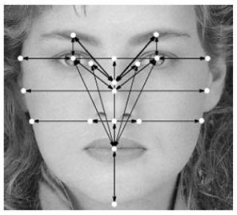
Bilden är en träff om det euklidiska avståndet är under den valda tröskeln. Om bilden som skickats in har ett större euklidiskt avstånd än tröskeln så anses det inte finnas i träningsdatan. Om bilden är en träff så retuneras ID:t på ansiktet som har det kortaste euklidiska avståndet till bilden.

3

Andra metoder

3.1 Identifiering med vinklar

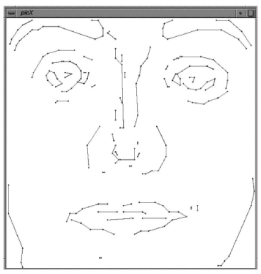
En annan metod för att identifiera ett ansikte är att bilda en triangel mellan ögonen och mitten på munnen (figur 3.1).Vinklarna i ansiktet kommer vara unika för varje ansikte och kan på så sätt användas för identifiering. Fördelen med att använda denna metod är att förhållandet mellan vinklarna alltid är samma även om bilden har roterats eller skalats. En nackdel med den här metoden är att det blir högre krav på att hitta ögon och mun och detta medför att metoden inte är lika robust som tex LEM(Line edge map).



Figur 3.1: Identifiering med hjälp av vinklar

3.2 Line Edge Map

Line edge map bygger på att ta ut alla linjer i ansiktet med hjälp av tex sobelfilter (figur 3.2). Alla linjer jämförs sedan genom att använda LHD (Line Segment Hausdorff Distance). LHD är ett sätt att jämnföra former mellan olika line maps. Denna metod är mer tolerant mot störningar i bilden eftersom den använder aproximeringar istället för exakta positioner.

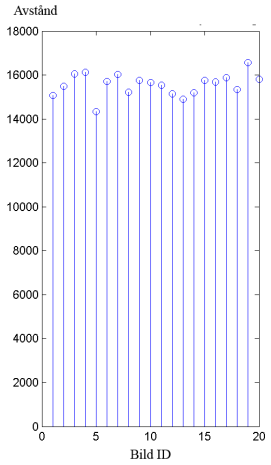


Figur 3.2: Line Edge Map

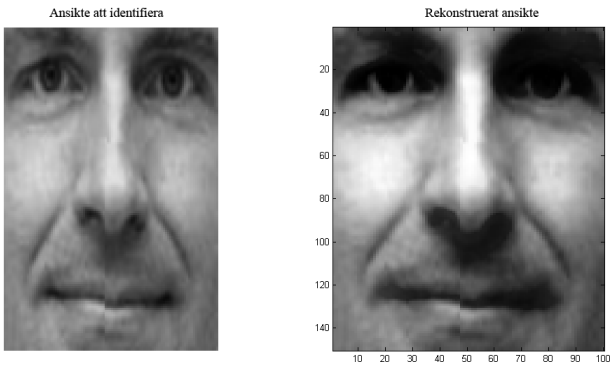
4

Resultat

Implementationen klarar av att identifiera alla bilder i träningsdatan samt bilder som manipulerats enligt kraven. Alla ansikten ur "db2" (ansikten som inte är med i träningsdatan) får en nolla som ID, dvs att de inte matchar något ansikte i träningsdatan. I figur 5.1 kan de euklidiska avstånden från inbilden till alla bilder i träningsdatan ses. Den bilden med lägst avstånd är det ID:t som kommer retuneras, om avståndet ligger under tröskelvärdet. Figur 4.2 visar den förbehandlade bilden som ska identifieras och bilden rekonstruerad från dess egenansikte.



Figur 4.1: Euklidiska avstånd för en bild



Figur 4.2: Inbilden bredvid den rekonstruerade bilden

5

Diskussion

Detektering av ögon hade varit simplare om upplösningen på bilderna hade varit högre. Då hade ögonen haft fler pixlar och man hade kunnat använda en till ögonmask som baserats på luminansen. Den ögonmasken hade sedan kombinerats med den andra ögonmasken för att få distinktare ögon. Valet av metod för klassificeringen av ansikten var inte ultimat då vi endast hade tillgång till en bild per person i träningsdatan, då egenansikten bygger på att man har flera bilder per person. Nedan ses en jämförelse av olika metoder och av olika sorters databaser. Som det kan utläsas så klarar egenansikten uppgiften bra mot den ena databasen fast får betydligt lägre procent igenkända ansikten i den andra. Detta beror på att Bern-databasen ändrar endast på tre variabler(huvudposition,storlek och kontrast) medans AR-databsen ändrar ansiktsuttryck, skymmer vissa delar av ansiktet och har olika sorters belysning.

	Bern database			AR database		
Method	EM	Eigenface	LEM	EM	Eigenface	LEM
Recognition rate	96.7%	100%	100%	88.4%	55.4%	96.4%

Figur 5.1: Jämförelse mellan olika metoder

Litteraturförteckning

[1] Rein-Lien Hsu, Mohamed Abdel-Mottaleb, Anil K. Jain, *Face Detection in Color Images*. IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, 2002. s. 699–700

[2] Santiago Serrano, `http://www.pages.drexel.edu/~sis26/Eigenface%20Tutorial.htm`, 2012-11-29.