Gezichtsdetectie

Research & Developement Pilot

Christiaan Thijssen
Wouter Gereadts
Kevin Reintjes

Inhoud

Inleiding

In ons onderzoek gaan we verschillende gezichtsdetectie methoden bekijken en implementaties van deze methoden testen. Hiervoor moeten we eerst een aantal zaken vaststellen, zoals wat we precies onder een gezicht verstaan en welke applicaties we willen gaan testen. Dit gebeurd in de eerste vier onderzoeksvragen. In de laatste drie onderzoeksvragen worden onze resultaten en conclusies behandeld.

Gezichtsdetectie is een belangrijke stap in het proces van gezichtsherkenning. Je moet namelijk eerst weten waar het gezicht zich bevind, voordat je kan gaan proberen het te herkennen. In deze pilot hebben we daarom als onderwerp gezichtsdetectie gekozen. Eventueel kunnen we later in een vervolg onderzoek ons richten op gezichtsherkenning.

Gezichtsdetectie en gezichtsherkenning zijn onderdelen van de Artificiële Intelligente. Voor mensen is het detecteren van een gezicht bijzonder makkelijk, we kunnen dit al van kinds af aan en hebben er eigenlijk nooit echt bij stil gestaan (het detecteren gebeurd bijna vanzelf). Voor een computer is dit echter een stuk ingewikkelder. Er zijn ingewikkelde algoritme voor nodig om een gezicht te detecteren en een computer heeft in sommige gevallen ook meer tijd nodig om een gezicht te detecteren. In dit onderzoek willen bepalen ‘hoe goed’ computers precies zijn in het detecteren van gezichten en hoeveel tijd ze hier gemiddeld voor nodig hebben. Eventueel willen we in een vervolg onderzoek vaststellen op welke punten computers precies vastlopen en hoe deze punten te verbeteren zijn.

Welke gezichtsdetectie methoden / applicaties zijn er allemaal precies op dit moment?

We omschrijven hier niet alle gezichtsdetectie methoden, omdat dit er te veel zijn en dit te veel tijd zou kosten. Hierdoor zou het onderzoek veranderen in opzoekwerk en hebben we minder tijd over voor het echte onderzoek. Vandaar dat er gekozen is om een globale opsomming te maken van veel gebruikte gezichtsdetectie methoden en applicaties.

Methoden:

- *Gezichten detecteren in afbeeldingen met een vaste achtergrond.*
  Dit is de meest simpele vorm van gezichtsdetectie. Als er ergens op de afbeelding een of meerdere gezichten bevinden en de achtergrond van de afbeelding is effen (en er zijn geen andere objecten op de achtergrond), dan is het weghalen van de achtergrond voldoende om het gezicht te detecteren. Kijk naar de coördinaten van wat er overblijft en de positie van het gezicht is bekend.
  Voordelen: Makkelijk te implementeren, werkt erg snel.
  Nadelen: Werkt alleen op afbeelding met een aantal gezichten en verder geen objecten en een effen achtergrond.

- *Gezichten detecteren op kleur.*
  Bij kleurenafbeelding zijn gezichten gemakkelijk te herkennen aan de typische kleur van gezichten. Als er ergens in de afbeelding een bepaalde ronde vlek (het gezicht) in de typische kleur van gezichten bevindt, is de kans groot dat daar het gezicht zit.
  Voordelen: Deze methode is ook redelijk makkelijk te implementeren en werkt ook redelijk snel. Werkt ook op afbeeldingen waar andere objecten aanwezig zijn.
  Nadelen: Werkt niet goed voor alle kleuren gezichten en verschillende belichtingstechnieken.

- *Gezichten detecteren door beweging.*
  Bij video's zijn gezichten over het algemeen makkelijk te detecteren door het zoeken naar bepaalde bewegingen. Een gezicht zal nooit helemaal stil staan, vandaar dat in de theorie een gezicht altijd gedetecteerd zal kunnen worden door beweging.
  Voordelen: Werkt voor bijna alle gezichten.
  Nadelen: Werkt alleen bij video's en niet bij losse afbeeldingen. Kan ook andere bewegende objecten als gezicht detecteren.

- *Gezichten detecteren op basis van typische gezichtskenmerken.*
  Deze methode is het meest effectiefst, maar tegelijk ook het meest ingewikkeld om te implementeren. Deze methode let op bepaalde typische kenmerken die gezichten vertonen. Zo zal een gezicht (bij een normale foto) altijd in bepaalde hoeken staan en

nooit verder gedraaid kunnen zijn dan een bepaald limiet (in verband met de beperkingen van de nek). Ook hebben gezichten een bepaalde afstand tussen onderdelen met een andere kleur (of andere grijswaarde in zwart-wit afbeeldingen). Bijvoorbeeld de ogen staan altijd ongeveer een bepaalde afstand van elkaar. Hetzelfde geldt over het algemeen voor de neus, de mond en de oren.
Om deze typische kenmerken aan te leren wordt er vaak gebruik gemaakt van een neuraal netwerk, wat geleerd wordt wat een gezicht is en wat niet.
Voordelen: Meest effectieve vorm van gezichtsdetectie. Deze methode maakt gemiddeld het minste fouten.
Nadelen: Leren van typische kenmerken kost veel tijd. Veel werk om de methode te implementeren.

Over het algemeen wordt er een combinatie van meerdere methoden gebruikt om de applicatie zo effectief mogelijk te maken.

Applicaties:

We omschrijven hier slechts de applicaties die door ons getest worden en niet alle applicaties. Dit zullen over het algemeen applicaties zijn die gratis te downloaden waren en het liefst open-source zijn (of een library hebben zodat we gemakkelijk een test omgeving kunnen maken in Java of C++).

- Fdlib_windows: Deze applicatie is geschreven in C++ en bestaat uit een windows library met functies om voor een bepaalde afbeelding het aantal gezichten te detecteren en de X- en Y-coördinaten van die gezichten te geven. Er zit ook een bestand fdtest.exe wat gebruikt kan worden om de gezichtsdetectie methode snel te testen (als je er niet zelf een programma om heen wilt schrijven).
  http://www.kyb.mpg.de/bs/people/kienzle/facedemo/facedemo.htm

- JNI2OpenCV: Deze applicatie is geschreven in Java en is een wrapper voor de (in C++ geschreven) OpenCV applicatie. Deze wrapper bied ook de mogelijkheid om het aantal gedetecteerde gezichten en de begin- en eind X- en Y-coördinaten terug te geven.
  http://www.drhu.org/freeDownload.asp

- FaceDetect: Deze applicatie is geschreven in C++ en is een programma om gezichten te herkennen en deze te omlijnen met een vierkant. Dit programma is tevens in staat de postie van de neus, mond en ogen te detecteren en te omlijnen. Dit programma zette in onze globale test erg goede resultaten neer, maar is helaas niet open source en heeft ook geen library. Dit programma bestaat slechts uit een executable wat het testen erg lastig maakt (het geeft ook geen coördinaten van het gezicht terug, het omlijnt deze alleen). Vandaar dat we besloten hebben dit programma niet uitgebreid

te testen.
http://www.iis.fraunhofer.de/bf/bv/kognitiv/biom/dd.jsp

Wat zijn de criteria van wat een gezicht genoemd mag worden en wat niet?

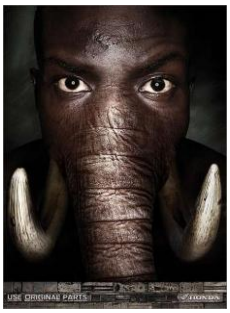
Voor het testen welke methode het beste gezichten kan herkennen moeten we eerst definiëren wat nou precies tot een gezicht geschaard mag worden en wat niet. Voor elke foto uit de set moet er namelijk bepaald worden of het een gezicht betreft en hier mogen geen onduidelijkheden over bestaan. Daarom gaan we nu criteria op stellen welke gezichten een goede gezichtsdetectie methode zou moeten herkennen als een gezicht en welke juist niet.

Het belangrijkste criterium dat we stellen is dat een gezicht als een mensengezicht gedetecteerd dient te worden als het een vooraanzicht van een mensenhoofd betreft dat door een mens, met het sinds de geboorte aangeleerde vermogen om gezichten te herkennen, wordt herkent. Als het dus duidelijk is dat de afbeelding een foto van een mensengezicht betreft (of dat er op een of andere manier een mensengezicht deels te zien is) dan heeft de methode voor gezichtsdetectie een juiste analyse gedaan als het een positief resultaat geeft.

Ook rekenen we natuurgetrouwe 3d gegenereerde afbeeldingen van mensengezichten tot de als gezichten te herkennen foto's omdat wij mensen erg veel moeite moeten doen om te zien dat het geen foto van een mens is maar een door de computer gegenereerd beeld. De intentie bij deze afbeeldingen was dan om zo goed mogelijk een gezicht na te maken zodat het verschil tussen een echt gezicht en de afbeelding vrijwel niet te zien of detecteren zou zijn. Vandaar dat we deze afbeeldingen ook tot de gezichten rekenen.

De tegenhanger van de natuurgetrouwe 3d tekeningen van gezichten zijn de afbeeldingen van gezichten die met opzet niet natuurgetrouw zijn weergegeven. Dit kunnen mensengezichten betreffen waarvan een eigenschap zodanig is veranderd dat het bij mensen niet voorkomt of zeer simpele tekeningen (of andere representaties) van gezichten. Hiervan kunnen mensen direct zien dat het geen echt mensengezicht betreft, maar een representatie van een gezicht dat gemaakt is uit ander materiaal (welk vaak ook meteen te herkennen is). Deze afbeeldingen dienen dus niet als gezicht gedetecteerd te worden.

Voorbeelden hiervan zijn:



Op het moment dat er een foto van een mens met een masker voor zijn gezicht wordt getest op de aanwezigheid van gezichten, dient de methode een negatief resultaat te geven. Het gezicht is namelijk verborgen achter een (niet natuurgetrouwe) representatie van een gezicht en er is dus op de foto zelf geen gezicht te zien. Als er echter een foto wordt ingevoerd van een gezicht dat is geschminkt of op een andere wijze is aangepast (piercings en tatoeages), dan dient de gezichtsdetectie methode wel een positief resultaat te geven.

Hoofden of Koppen van dieren hebben over het algemeen redelijk veel gelijkenis met mensengezichten. Dit is vooral het geval bij apen. Dierenhoofden of koppen worden echter niet tot gezichten gerekend. Wij als mensen zien direct dat het geen mens betreft, ook al heeft het dier ogen, een neus, een mond en haar. Omdat mensen zo snel het onderscheid kunnen zien, rekenen we dit niet tot de gezichten. Bij dieren dienen de detectiemethodes dus een negatief resultaat te geven.

Welke afbeeldingen zijn geschikt om deze methoden te testen.

Voor het testen van de verschillende gezichtsdetectie methodes moeten we een testset foto's hebben. We hebben besloten dat er redelijk betrouwbare resultaten uit de proeven moeten komen als we een totale hoeveelheid van ongeveer 4000 foto's als testset nemen.

Deze 4000 foto's bestaan uit:

1824 portret of zijaanzicht foto's uit een database (genaamd FERET) met foto's van mensen die vrijwillig foto's hebben laten maken voor de wetenschap die onderzoek doet naar gezichtsdetectie of gezichtsherkenning. Deze database is vrij te gebruiken voor onderzoeken zoals dit onderzoek en is aan te vragen op de site: http://face.nist.gov/colorferet/
Deze foto's zouden de methodes voor het grootste deel moeten herkennen omdat deze foto's opnames van gezichten van dichtbij zijn. De foto's zijn allemaal zo gemaakt dat de gezichten zich op een a twee meter van de camera bevonden (zoals bij Portretfoto's gebruikelijk is). De gezichten beslaan dus in de meeste gevallen de gehele foto. Dit zijn de meest duidelijke foto's van gezichten. Ook zitten er portretfoto's bij die van de zijkant van de personen genomen zijn. Hierdoor is maar een helft van het gezicht echt duidelijk te zien en hier zullen de gezichtsdetectie methodes waarschijnlijk meer moeite mee hebben deze te detecteren en de juiste coördinaten van het gezicht te geven.
1954 "garbage" foto's zonder gezichten of iets dat met gezichten te maken heeft (dieren of maskers). Deze set foto's bestaat uit random foto's die van internet gedownload zijn. Het zijn foto's van gebouwen, natuur of andere dingen. Deze foto's bevatten geen enkel gezicht, maar het is wel mogelijk dat de methodes in de samenstelling van de afgebeelde dingen misschien wel een gezicht detecteert. Om te bepalen in welke mate de methodes een foutief positief resultaat geven hebben we deze grote hoeveelheid "garbage" foto's (foto's die niet specifiek zijn) toegevoegd. De methodes zouden in het meest positieve geval dus de gehele set niet als gezicht te detecteren. Bij deze reeks foto's zullen de methodes niet zo veel fouten maken omdat het niet vaak voor zal komen dat de patronen van gezichten, waar de methodes naar zoeken, toevallig in de foto's voorkomen.
241 Dierengezichten van verschillende dieren. Deze dieren hebben ook een hoofd of kop en een, voor elke diersoort specifiek, gezicht. We hebben bij onze criteria besloten dat een hoofd of kop van een dier niet tot een gezicht gerekend mag worden. De set dient in het meest positieve geval volledig afgekeurd te worden bij het detecteren van gezichten. Deze set foto's zullen waarschijnlijk snel een foutief positief resultaat opleveren omdat de hoofden of koppen van dieren meestal toch wel de combinatie van (een soort) ogen, mond en neus bevatten. Deze hoofden of koppen zullen dus erg lastig te onderscheiden zijn van gezichten.
73 Niet realistische afbeeldingen van gezichten. Deze afbeeldingen zijn op een of andere manier een representatie van een gezicht. Denk aan abstracte schilderijen van gezichten met hun expressies. Volgens onze criteria voldoen tekeningen, schilderijen of beelden van

gezichten niet aan de eisen van een gezicht. In het meest gunstige geval zullen de methodes deze afbeeldingen dus niet als gezicht detecteren.
28 Vreemde of deels verborgen gezichten. Deze gezichten zijn gedraaid, deels of geheel verhult door een masker of een ander voorwerp. Deze gezichten zijn een stuk moeilijker te herkennen dan de 1500 portret of zijaanzichten die we ook in de test hebben opgenomen.

De testset bestaat dus voornamelijk uit foto's van de eerste twee genoemde categorieën. Deze foto's zijn voor ons als mens gemakkelijk in te delen in de categorieën foto's met en foto's zonder gezichten. De laatste 3 genoemde categorieën zijn een veel kleiner deel van de testset, maar vormen waarschijnlijk wel een echte uitdaging voor de detectie methodes.

Testmethode

We hebben besloten om het testen van de applicaties met behulp van een Java interface te doen. Dit gezien de grote hoeveelheid afbeeldingen die getest moeten worden. Als eerste stap hebben we alle afbeeldingen gesorteerd op de hierboven omschreven categorieën en voor elke categorie een aparte directory aangemaakt. De door ons geschreven Java interface doet vervolgens de volgende zaken:

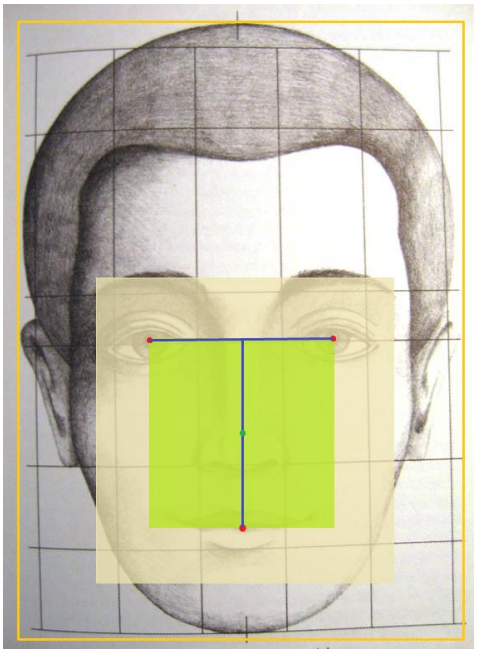
1. Een door ons aangewezen directory wordt geopend.
2. Vervolgens worden alle afbeeldingen in deze directory geopend.
3. Indien nodig wordt een afbeelding naar .bmp formaat geconverteerd.
4. De afbeelding wordt vervolgens door de verschillende applicaties geleid. Deze worden aangeroepen als Java klasse (zoals bij JNI2OpenCV) of als executable.
5. De Java interface houd bij hoeveel tijd de huidige applicatie nodig heeft om met een resultaat te komen.
6. Het resultaat wordt opgeslagen in een MySQL database. Er wordt per afbeelding opgeslagen of er gezichten zijn gevonden en zo ja, hoeveel gezichten op welke coördinaten in de afbeelding (X en Y coördinaten). Ook wordt de benodigde tijd die de applicatie nodig had om met een resultaat te komen opgeslagen. Verder wordt opgeslagen tot welke categorie de afbeelding behoort.

Na het uitvoeren van deze test interface hebben wij de beschikking over een database met test resultaten. Hier kunnen wij (door middel van MySQL query's) gemakkelijk resultaten uithalen. Verder kunnen we onze database met test resultaten koppelen aan de database van FERET om de coördinaten van de gezichten gevonden door onze applicaties te vergelijken met de coördinaten van de gezichten in de FERET database en de afwijkingen te berekenen.

De soort coördinaten uit de FERET database komt echter niet helemaal overeen met de coördinaten die de programma's als uitvoer geven. De FERET database heeft namelijk zowel x als y coördinaten voor beide ogen en de mond. We weten dus voor elke foto uit de FERET database precies wat de coördinaten van de ogen en de mond van het getoonde gezicht zijn. De coördinaten die geteste programma's uitgeven zijn alleen coördinaten van het middelpunt van het gedetecteerde gezicht of een hoekpunt van het als gezicht herkende gebied en een grootte. Als het resultaat van het algoritme een hoekpunt betrof, hebben we dit in combinatie met de grootte omgerekend naar de coördinaten van het middelpunt van het als gezicht herkende gebied. Dit middelpunt zou dus bij een juiste detectie van het gezicht gelijk moeten zijn aan het punt dat met de x coordinaat precies tussen de x

coördinaten van de in de FERET aangegeven ogen zit en met de y coordinaat precies tussen het gemiddelde van de y coordinaten van beide ogen en de y coordinaat van de mond zit.

De volgende afbeelding illustreert de samenhang tussen de coördinaten uit de FERET database (met rode punten aangegeven) en de coördinaten die als resultaat naar voren komen tijdens tests met de gezichtsdetectie programma's.



De groene punt geeft het hierboven beschreven middelpunt van de 3 coordinaten uit de FERET database aan. Dit punt is het beste resultaat dat een gezichtsdetectie algoritme kan opleveren. Natuurlijk is het nooit zo dat een algoritme zo'n coordinaat altijd precies goed heeft of helemaal fout. Vandaar dat we ook hier criteria voor moeten opstellen. Wat rekenen we nu tot een goede detectie en wat rekenen we nu tot een foutieve detectie? We hebben besloten dat de uitkomst van de algoritmes, wat dus het middelpunt van het gezicht zou moeten zijn (groene punt), in het gebied tussen de ogen en de mond horen te zitten. Dit gebied is in de afbeelding met groen gearceerd. Omdat we ook de detectie van een gezicht met een middelpunt die niet helemaal tussen de ogen en mond liggen goed willen rekenen, hebben we besloten om ook nog een marge te stellen. De zandkleurige arcering in de afbeelding toont het gebied dat we ook nog goed zullen rekenen als coördinaten van het middelpunt, dit omdat het gezicht in het geheel bij deze coördinaten als uitkomst eigenlijk wel gedetecteerd is, maar niet goed in te schatten was wat de proporties van het gezicht waren en waar de ogen en mond precies zaten. Deze marge zal 150% van het (groen gearceerde) gebied tussen de ogen en de mond zijn. Als de coördinaten die de algoritmes als uitkomst geven binnen deze marge vallen, heeft het algoritme een juiste detectie gedaan.

De algoritmes geven naast coördinaten van het middelpunt van het gezicht ook nog een grootte van het gezicht als resultaat. Deze grootte is zowel de lengte als breedte van het gebied waarin het algoritme een gezicht herkent. Het algoritme geeft dus de informatie op basis waarvan je een vierkant zou kunnen tekenen die de ogen en de mond (de voornaamste kenmerken van een gezicht) omlijnen. Voor deze grootte hebben we besloten dat deze maximaal het hele hoofd mag omlijnen. Dit komt overeen met 200% het gebied tussen de ogen en de mond (op de tekening aangegeven met een oranje lijn). Ook moet deze grootte minimaal de grootte van het gebied tussen de ogen en de mond hebben (de groene arcering).

Als de resultaten van het algoritme aan bovenstaande grenzen voldoen, wordt het resultaat goedgekeurd, anders niet.
Als foto's meerdere gezichten bevatten, rekenen we elk goed gedetecteerd gezicht als een goed resultaat, elk fout gedetecteerd gezicht als een foutief resultaat.
Mochten er op een foto's geen gezichten zijn, dan is het resultaat van de detectie goed op het moment dat er geen resultaten van gedetecteerde gezichten worden gegeven. Worden er wel (meerdere) resultaten gegeven dat er gezichten gevonden zijn (die er dus eigenlijk helemaal niet zijn) dan wordt de foto als een geheel als een foutieve detectie beschouwd.
Mocht een foto maar een gezicht bevatten en er meerdere positieve resultaten volgens het algoritme zijn, dan telt het resultaat alleen als een goede detectie op het moment dat er een van de coördinaten voldoen aan de bovengenoemde eisen.

Welke gezichtsdetectie methode detecteert het hoogste aantal gezichten in onze test.

Zoals beschreven bij 'Testmethode' hebben wij na het uitvoeren de test de beschikking over een uitgebreide MySQL database met resultaten. Deze database is te bezichtigen op http://www.onii.org/phpmyadmin met gebruikersnaam: kevin en wachtwoord: woei.

We hebben de resultaten onderverdeeld in secties van afbeeldingen. Hierbij hebben wij de volgende secties gehanteerd:

1. portretten uit de FERET database, 1824 afbeeldingen *.
2. Streekfoto's zonder gezichten, 1826 afbeeldingen.
3. Streekfoto's met dieren, 143 afbeeldingen.
4. Afbeeldingen met dieren gezichten, 98 afbeeldingen.
5. Vervaagde / onduidelijke gezichten, 11 afbeeldingen *.
6. Overige afbeeldingen zonder gezichten, 128 afbeeldingen.
7. Tekeningen van gezichten, 73 afbeeldingen.
8. Vreemde / verborgen gezichten, 17 afbeeldingen *.

* = In deze sectie van afbeeldingen zou de gezichtsdetectie methode volgens onze criteria een gezicht moeten kunnen vinden.

Uit onze database hebben wij met behulp van een aantal query's de volgende resultaten gekregen:

FDlib windows

Sectie	Totaal aantal gezichten gevonden	Gemiddeld aantal gezichten gevonden per afbeelding.
1*	1897	1.0400
2	2468	1.3516
3	246	1.7203
4	82	0.8367
5*	2	0.1818
6	32	0.2500

7	36	0.4932
8*	12	0.7059

Uit deze resultaten blijkt dat de FDlib in elke afbeelding van de FERET sectie gemiddeld iets meer dan één gezicht ontdekt. Dit is positief, aangezien in elke afbeelding in de Feret sectie ook precies één gezicht staat. Merkwaardig genoeg detecteert FDlib echter in de sectie 2, 3 en 4 per afbeelding gemiddeld ook bijna of meer dan één gezicht, terwijl deze afbeeldingen juist geen gezichten bevatten. Op dit vlak doet FDlib het bij de FERET sectie erg goed, maar in de overige secties doet FDlib het aanzienelijk minder goed.

Sectie	Totaal aantal afbeeldingen waarin een gezicht werd gevonden	Percentage van afbeeldingen waarin een gezicht werd gevonden.	Correct
1*	1557	85.3618%	85.3618%
2	974	53.3406%	46.6594%
3	90	62.9371%	37.0629%
4	44	44.8980%	55.1020%
5*	2	18.1818%	18.1818%
6	20	15.6250%	84.3750%
7	24	32.8767%	67.1233%
8*	9	52.9412%	52.9412%

Uit deze resultaten is af te leiden in hoeveel van de afbeeldingen per sectie FDlib_windows een gezicht vond. Hoe hoger het percentage in een sectie met een * en hoe lager het percentage in een sectie zonder * hoe beter. Een hoog percentage in een sectie afbeeldingen met gezichten betekend namelijk dat FDlib vaak een correct positief resultaat terug geeft. Een laag percentage in een sectie afbeeldingen zonder gezichten betekend dat FDlib vaak een correct negatief resultaat teruggeeft. Beiden gevallen zijn positief.

Als er juist een laag percentage in een sectie afbeeldingen met gezichten is betekend dit dat FDlib vaak een verkeerd negatief resultaat teruggeeft. Een hoog percentage in een sectie afbeeldingen zonder gezichten zou betekenen dat FDlib vaak een verkeerd positief resultaat terug geeft. Deze laatste twee gevallen zouden er op wijzen dat de gezichtsdetectie methode niet correct functioneert.

Uit onze test is gebleken dat FDlib voornamelijk bij de FERET sectie erg goed scoort. In 85% van de gevallen wordt een correct positief resultaat terug gegeven. In de andere gevallen doet FDlib het aanzienelijk slechter. Zo geeft FDlib in de sectie streekfoto's zonder gezichten toch in 53% van de gevallen een verkeerd positief resultaat terug. Eigenlijk scoort FDlib alleen in de FERET sectie goed, maar in de andere secties bedroevend slecht.

JNI2OpenCV

Sectie	Totaal aantal gezichten gevonden	Gemiddeld aantal gezichten gevonden per afbeelding.
1*	1845	1.0115
2	200	0.1095
3	17	0.1189
4	20	0.2041
5*	2	0.1818
6	4	0.0313
7	22	0.3014
8*	11	0.6471

Aan het gemiddeld aantal gezichten gevonden per afbeeldingen valt op dat dit getal in de FERET sectie heel dicht tegen de één aan ligt. Dit is positief, aangezien elke afbeelding in de FERET sectie ook precies één gezicht bevat. Wat verder opvalt is dat het gemiddeld aantal gezichten gevonden in afbeeldingen zonder gezichten (secties 2, 3, 4, 6 en 7) een stuk lager is. Het lijkt er op dat JNI2OpenCV redelijk in staat is het aantal gezichten in een afbeelding te bepalen (in ieder geval het onderscheid tussen één gezicht en nul gezichten).

Sectie	Totaal aantal afbeeldingen waarin een gezicht werd gevonden	Percentage van afbeeldingen waarin een gezicht werd gevonden.	Correct
1*	1804	98.9035%	98.9035%
2	172	9.4195%	90.5805%
3	16	11.1888%	88.8112%
4	1	13.2653%	86.7347%

5*	2	18.1818%	18.1818%
6	4	3.1250%	96.8750%
7	19	26.0274%	73.9726%
8*	10	58.8235%	58.8235%

Deze resultaten kunnen hetzelfde geïnterpreteerd worden als de resultaten van FDlib_windows. Wat gelijk opvalt aan deze resultaten is het bijzonder hoge percentage van afbeeldingen uit de FERET sectie waarin een gezicht werd gedetecteerd. Wat tevens opvalt is dat dit percentage in afbeeldingen zonder gezichten juist een stuk lager is. Beiden constateringen zijn positief en houden in dat dit algoritme goed in staat is afbeeldingen met gezichten te onderscheiden van afbeeldingen zonder gezichten. Gezien de percentages kunnen we concluderen dat JNI2OpenCV beter in staat is om afbeeldingen met gezichten te onderscheiden van afbeeldingen zonder gezichten dan FDlib_windows.

Welke gezichtsdetectie methode detecteert het hoogste aantal gezichten op de juiste locatie in onze test.

Van de FERET database zijn de coördinaten van de ogen, mond en neus bekend. Hieruit hebben we de coördinaten van het midden van het gezicht bepaald (zie het kopje 'Testmethode') en gekeken of dit overeen kwam met wat de gezichtsdetectie methoden als coördinaten gaven. We hadden echter alleen de beschikking over deze gegevens bij de FERET sectie, bij de overige secties met gezichten (5 en 8) hebben we daarom geen resultaten over of het gezicht ook op de juiste locatie is gedetecteerd.

Met behulp van een PHP script hebben we een methode opgesteld om door middel van de gegevens in de database te controleren of het gezicht op de juiste locatie is gevonden. Dit onderzoek gaf de volgende resultaten:

FDlib_windows heeft bij 52.3026% van de afbeeldingen uit de FERET sectie een gezicht gedetecteerd op de juiste locatie. Dit houd in dat van de afbeeldingen waarin FDlib een gezicht vond (85.3618%) er bij 61.2717% ook op de juiste locatie een gezicht gedetecteerd werd.

JNI2OpenCV heeft bij slechts 0.1645% van de afbeeldingen uit de FERET sectie een gezicht op de juiste locatie gevonden. Het valt op dat dit bijzonder weinig is (terwijl JNI2OpenCV het op de andere vlakken juist veel beter deed dan FDlib_windows). De verklaring hiervoor is waarschijnlijk dat JNI2OpenCV een gezicht veel ruimer neemt. FDlib_windows geeft namelijk als formaat van het gezicht de breedte van de tussen de uiteinden van de ogen. JNI2OpenCV neemt een gezicht echter veel breder en hoger. Omdat dit buiten onze marges valt heeft JNI2OpenCV zo'n slecht resultaat op dit punt. Het blijkt echter dat (na handmatige controle van de afbeeldingen) JNI2OpenCV het midden van een gezicht wel goed detecteerd alleen het formaat telkens met een bepaalde factor te groot inschat. JNI2OpenCV is dus in principe prima te gebruiken als je er rekening mee houd dat JNI2OpenCV de gezichten ongeveer twee keer zo ruim neemt dan de afstand tussen het uiterst linkse punt van het linker oog en het uiterst rechtse punt van het rechter oog. Volgens onze criteria zou JNI2OpenCV op dit punt bedroevend slecht scoren. Echter blijkt uit ons kleine handmatige onderzoek dat JNI2OpenCV prima bruikbaar is als je rekening houd met het feit dat JNI2OpenCV gezichten erg ruim neemt.

Welke gezichtsdetectie methode is het snelst.

Het is ons helaas niet gelukt om geldige resultaten te krijgen wat betreft de tijdsmeting. De oorzaak hiervan is als volgt: We hebben alle testen onder Linux gedaan, echter kregen we de linus variant van FDlib niet aan de praat. Vandaar dat we de Windows executabel gebruikt hebben en deze via het programma Wine hebben laten werken. Uiteindelijk hebben we in voor onze test applicatie zoveel wrappers en / of andere programma's gebruikt alleen om de verschillende methoden in dezelfde interface te laten werken, dat geldige tijdsmetingen niet meer te doen waren. We hebben echter voor de volledigheid het toch bijgehouden en een gemiddelde tijd per afbeelding berekent:

FDlib_windows heeft gemiddeld 3.748 seconden per afbeelding nodig.
JNI2OpenCV heeft gemiddeld 1.688 per afbeelding nodig.

Hieruit blijkt dat FDlib_windows aanzienlijk trager is dan JNI2OpenCV. Echter is dit waarschijnlijk voornamelijk te wijten aan het gebruik van Wine dan aan de applicatie zelf. Een korte (handmatige) test met FDlib_windows in Windows toonde aan dat FDlib_windows juist bijzonder snel is en vaak binnen een halve seconde de gezichten al gedetecteerd heeft (en het resultaat naar het scherm heeft weggeschreven). FDlib_windows lijkt in Windows over het algemeen ook sneller te werken dan JNI2OpenCV.

Conclusie

We zijn bijzonder verrast met de goede resultaten van JNI2OpenCV. JNI2OpenCV is prima in staat om afbeeldingen met gezichten te onderscheiden van afbeeldingen zonder gezichten. Het detecteren van de juiste locatie gaat ook prima, het detecteren van de juiste hoogte en breedte echter een stuk minder. Deze wordt (volgens onze criteria) veels te ruim genomen. Desondanks is JNI2OpenCV prima bruikbaar als je met dit punt rekening houd.

FDlib_windows scoorde daarentegen een stuk slechter. Er wordt volgens ons te vaak een incorrect resultaat terug te geven. Ook het detecteren van gezichten op de juiste locatie gaat nog niet geweldig. Slechts van 52.3026% van de afbeeldingen in de FERET database werd de juiste hoeveelheid gezichten en de juiste locatie van deze gezichten door FDlib_windows terug gegeven. FDlib_windows heeft voornamelijk moeite met afbeeldingen zonder gezichten. Het lijkt erop dat dit algoritme per se een gezicht wilt vinden en vind daardoor vaak in afbeeldingen zonder gezichten op onlogische locaties toch een gezicht. De oorzaak hiervan is bij ons niet precies bekend (en is ook lastig te achterhalen, aangezien de source-code van FDlib_windows niet beschikbaar is). FDlib_windows lijkt op het eerste gezicht (als je deze in Windows draait tenminste) wel een stuk sneller te zijn dan JNI2OpenCV. Echter hebben we dit niet uitgebreid getest en kunnen we dit dus niet met zekerheid vast stellen. Bovendien is het de vraag of deze extra snelheid wel opweegt tegen de grote hoeveelheid fouten die deze methode maakt.

Helaas waren wij niet in staat meerdere gratis gezichtsdetectie methoden (die ook automatisch getest zouden kunnen worden) te vinden. Vandaar dat ons onderzoek zicht beperkt heeft tot FDlib_windows en JNI2OpenCV. Aangezien we slechts twee methoden getest hebben is het lastig om een gemiddelde te bepalen, bijvoorbeeld hoeveel procent van de afbeeldingen detecteren de verschillende methoden gemiddeld. Het is natuurlijk wel mogelijk om een gemiddelde uit twee waarden te berekenen, het nut hiervan zou echter niet zo groot zijn.

Mochten er in de toekomst echter nog meer gezichtsdetectie methoden beschikbaar zijn, dan is het voor ons relatief simpel om deze te testen (aangezien we al verschillende sets van afbeeldingen hebben en een Java interface om deze afbeeldingen snel en automatisch te testen). Als vervolg voor het vak Research & Developement gaan we waarschijnlijk een poging doen een eigen gezichtsdetectie methode te schrijven. Als we hier in slagen kunnen we deze methode ook relatief gemakkelijk laten testen door onze Java interface en de resultaten vergelijken met FDlib_windows en JNI2OpenCV.

Literatuur

- Stuart Russell, Peter Norvig, *Artificial Intelligence A Modern Approach*, Pearson Education Inc, New Jersey, 2003
- Facial Recognition Technology (FERET) Database National Institute of Standards and Technology
  http://www.nist.gov/humanid/colorferet/
-
- Face Detection techniques
  http://www.facedetection.com/facedetection/techniques.htm
- Face Detection Software
  http://www.facedetection.com/facedetection/software.htm
- FDlib_windows
  http://www.kyb.mpg.de/bs/people/kienzle/facedemo/facedemo.htm
- Face Detection System
  http://franck.fleurey.free.fr/FaceDetection/index.htm
- JNI2OpenCV
  http://www.drhu.org/freeDownload.asp
- FaceDetect
  http://www.iis.fraunhofer.de/bf/bv/kognitiv/biom/dd.jsp