13-2-2020

Implementatieplan

Een plan van implementatie

Nathan Houwaart en Jochem van Weelde
V2A

Inhoud

Doel

Het doel

Het doel van dit onderzoek is een Offline gezichtsherkenning toepassing ontwikkelen met hoge accuraatheid.

Facebook gebruikt gezichtsherkenning om te vragen aan gebruikers wie deze persoon is. De gebruiker kan deze persoon “taggen” in de foto. Doordat dit offline kan gebeuren (Niet real-time) is accuraatheid belangrijker dan snelheid.

Door gezichten te analyseren in foto’s kunnen die ook gecategoriseerd worden op personen.

De kwestie:

Als er een foto op een online website wordt geüpload van personen, moet de gebruiker een makkelijke manier tot zijn beschikking hebben om de personen in de afbeelding te “taggen”

De gewenste situatie:

Als een persoon een foto uploadt, moet het programma een blokje om gezichten tekenen zodat de gebruiker die de foto uploadt, makkelijk personen kan “taggen” in de foto’s.

Methoden

Uit onderzoek (Nadernejad, Shirifzadeh, & Hassanpour, 2008) blijkt dat er verschillende methoden zijn om edge detection toe te passen. Een aantal verschillende edge detection algoritmen zijn onder andere (maar niet beperkt tot):

- The Marr-Hildreth Edge Detector
- The Canny Edge Detector
- The Local Threshold and Boolean Function Based Edge Detection
- Color Edge Detection Using Euclidean Distance and Vector Angle
- Color Edge Detection using the Canny Operator
- Depth Edge Detection using Multi-Flash Imaging

Er zijn ook onderzoeken (Maini, R., & Aggarwal, H. (2009) & Nadernejad, E., Sharifzadeh, S., & Hassanpour, H. (2008). gedaan naar de voor en nadelen van verschillende edge detection technieken. Hieronder volgt een overzicht:

Methode	Voordelen	Nadelen
The Marr-Hildreth Edge Detector	Populair, makkelijk te implementeren	Defect in de hoeken, bochten en waar de grijsniveau-intensiteitsfunctie varieert.
Canny edge detection	Betere detectie in noisy afbeeldingen	Duurt lang op te optimaliseren, vereist veel rekenkracht
The Local Threshold and Boolean Function Based Edge Detection	Minder sensitief voor belichting en kan goed overweg met noise in de afbeelding	Produceert minder duidelijke lijnen dan canny edge
Color Edge Detection Using Euclidean Distance and Vector Angle	Neemt kleur mee in het bepalen van edges.	Methode mist details in de afbeelding
Color Edge Detection using the Canny Operator	Neemt kleur mee in het bepalen van edges waardoor het meer edges vind	Duurt lang op te optimaliseren, nog meer rekenkracht dan greyscale canny edgde
Depth Edge Detection using Multi-Flash Imaging		Heeft moeite met edges dat op dezelfde diepte staan of met dieptes ver weg in de afbeeling

Uit meerdere onderzoeken (Maini, R., & Aggarwal, H. (2009) & (Nadernejad, Shirifzadeh, & Hassanpour, 2008) & Shrivakshan, G. T., & Chandrasekar, C. (2012). blijkt dat canny edge de superieure edge detection techniek is voor het detecteren van edges. Deze methoden zal dan ook toegepast worden voor de opdracht. Deze onderzoeken geven echter wel aan dat canny edge detection een stuk zwaarder is om uit te voeren. Dit probleem is echter te verwaarlozen aangezien de kwestie waarvoor edge detection gebruikt wordt, een offline kwestie is. Hierbij is accuraatheid belangrijker dan snelheid. Daarnaast kan de threshold altijd worden aangepast om minder belangrijke edges weg te halen of juist meer detail over te laten voor een hogere accuraatheid.

Keuze

Er zullen een tweetal methoden worden geïmplementeerd in dit onderzoek. De The Marr-Hildreth Edge Detector wordt geïmplementeerd vanwege de simpliciteit. Daarnaast zal ook de Canny Edge methode worden geïmplementeerd. Ondanks de extra rekenkracht die dit algoritme gebruikt, blijft volgens onderzoeken de Canny Edge detecion superieur aan de andere methoden. Daarnaast maakt voor deze kwestie de extra rekenkracht niet uit.

De twee geïmplementeerde edge detection methoden zullen ook met elkaar worden vergeleken.

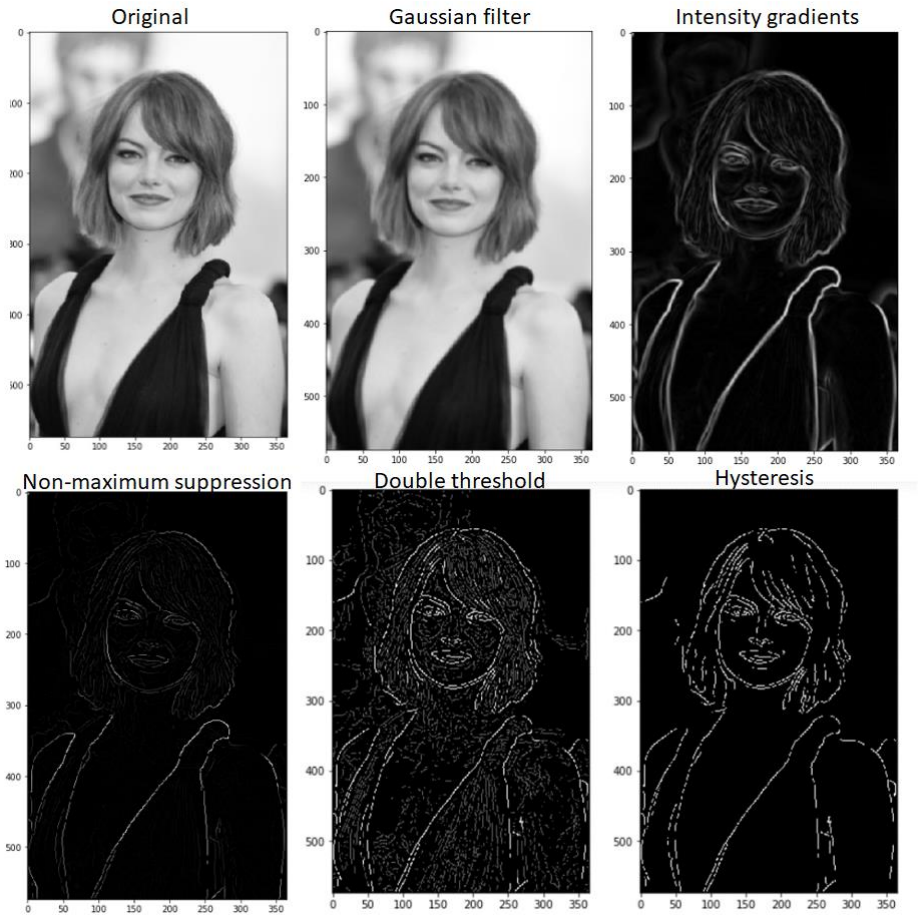
Canny edge detection is een lastige methode om te implementeren, maar is erg populair. Mede dankzij uitgebreide stappenplannen (Roodt, Y., Visser, W., & Clarke, W. A. (2007)) achten we de kans hoog dat het lukt om deze methode te implementeren.

De The Marr-Hildreth Edge Detection is makkelijk om te implementeren en moet geen probleem zijn.

Implementatie

Om Canny edge de implementeren moeten de volgende stappen op volgorde worden uitgevoerd:

1. Pas een **Gaussian filter** toe om een foto egaler te maken. Hierdoor wordt ruis verwijderd van de foto.
2. Zoek de **intensiteitsgradiënten**. Hiermee kunnen randen worden gevonden. Deze randen kunnen verticaal, horizontaal en diagonaal gezocht worden. Door alle grote veranderingen witter te tekenen ontstaan er randen in het plaatje.
3. Pas **niet maximum suppressie** toe. De randen van het vorige plaatje worden met deze techniek dunner.
4. Een **dubbele drempel filter** toepassen. Door lage intensiteit pixels helemaal zwart te maken en hoge intensiteit pixels helemaal wit te maken (Of precies andersom), ontstaat er een foto met super scherpe felle randen. Met behulp van een dubbele drempel, is er aan het einde verschil in de sterkste randen (wit), de gemiddelde randen (grijs) en de zwakste (zwart).
5. De rand volgen met **Hysteresis**. Door lijnen te volgen kunnen zwakke randen toch worden meegenomen doordat het een sterke rand verlengt en waarschijnlijk een echte rand is. Zwakke randen die helemaal alleen staan, worden weggehaald.



Evaluatie

Tijdens het testen van de twee edge detection technieken zullen er meerdere soorten afbeeldingen worden gebruikt. Samen met de testafbeeldingen, zullen ook afbeeldingen met een minder witte achtergrond en ruis worden geanalyseerd. Daarnaast zullen ook afbeeldingen met meerdere personen worden geanalyseerd om te kijken of het algoritme dan nog steeds goed werkt.

Benodigde input:

- Testafbeeldingen (paspoortfoto's) https://www.shutterstock.com/nl/search/passport+photo
- Afbeeldingen met meer ruis / minder witte achtergrond

In dit onderzoek willen we aantonen dat canny edge detection beter is (accurater) in het detecteren van gezichten en het opnieuw herkennen van een gezicht wat het systeem.

Stappenplan

De volgende stappen zullen worden ondernomen om aan te tonen dat canny edge detection beter is dan de Marr-Hildreth Edge Detector:

1. Één van de twee edge detection methoden wordt geselecteerd
2. Honderd testafbeeldingen (paspoortfoto's) worden door de geselecteerde edge detection methode gehaald
3. Er wordt gekeken naar hoeveel gezichten de applicatie succesvol kan detecteren
4. De gedetecteerde afbeeldingen zullen worden opgeslagen in de applicatie
5. De eerder gedetecteerde afbeeldingen worden opnieuw door de edge detector gehaald
6. De applicatie moet als het goed is dezelfde persoon detecteren en de corresponderende opgeslagen afbeelding laten zien
7. Noteren hoe 'zeker' de applicatie is van de macht van database en afbeelding
8. Deze tests worden herhaald voor de andere edge detection methode met dezelfde honderd testafbeeldingen
9. Aan het einde van de tests worden de resultaten vergeleken

Om een accuraat beeld te kunnen schetsen, wordt dit stappenplant meerdere keren uitgevoerd met telkens 100 andere afbeeldingen.

Extra Criterium

Er wordt ook getest of het algoritme met ruis kan omgaan.

1. Alle afbeeldingen die zijn herkent door het algoritme worden verzameld.
2. Voeg een klein beetje ruis toe aan alle foto's
3. Check of het algoritme nog werkt op de foto's met ruis en schrijf de resultaten op.
4. Herhaal stap 2 en 3 totdat alle foto's niet meer worden herkent.

Herhaal ook dit stappenplan met steeds een andere set foto's. Uiteindelijk heeft een van de twee algoritmes het beter volgehouden. Er kan dan worden geconcludeerd welk algoritme beter om kan gaan met ruis.

Hoe wordt bepaald welke methode beter is dan de andere?

Er wordt per methode gekeken naar het aantal gezichten dat succesvol herkend wordt.

Als een methode meer gezichten kan herkennen, is deze methode beter voor de bovengenoemde kwestie.

Daarnaast wordt gekeken naar de ‘zekerheid’ van een match van een foto uit de database.
Wanneer een methode slechter is in het detecteren van een gezicht, maar beter is in het herkennen van opgeslagen gezichten is deze methode minder geschikt voor de bovengenoemde kwestie, maar beter geschikt voor een eventuele persoons herkenning applicatie.

Hypothese
Er wordt verwacht dat Canny Edge beter gezichten kan detecteren en ook opgeslagen afbeeldingen beter kan matchen.

Literatuurlijst

Acharjya, P. P., Das, R., & Ghoshal, D. (2012). Study and comparison of different edge detectors for image segmentation. *Global Journal of Computer Science and Technology*.

Kaur, S., & Singh, I. (2016). Comparison between edge detection techniques. *International Journal of Computer Applications*, *145*(15), 15-18.

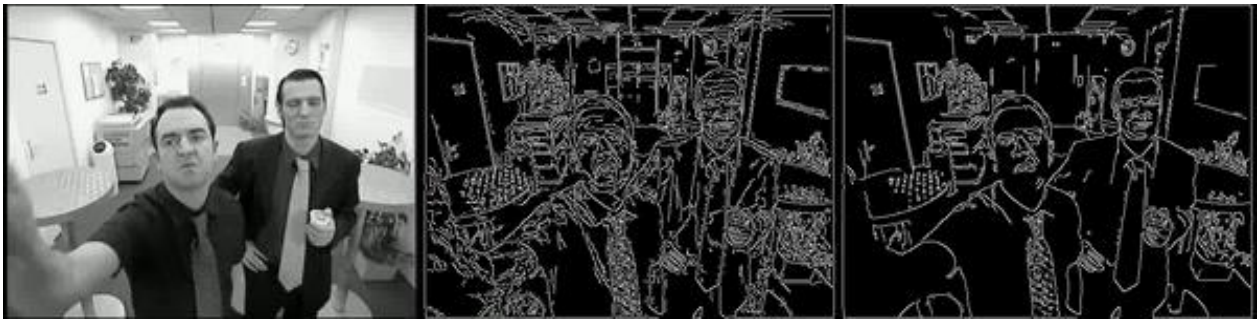
Maini, R., & Aggarwal, H. (2009). Study and comparison of various image edge detection techniques. *International journal of image processing (IJIP)*, *3*(1), 1-11.

Nadernejad, E., Sharifzadeh, S., & Hassanpour, H. (2008). Edge detection techniques: evaluations and comparisons. *Applied Mathematical Sciences*, *2*(31), 1507-1520.

Roodt, Y., Visser, W., & Clarke, W. A. (2007). Image processing on the GPU: Implementing the Canny edge detection algorithm.

Shrivakshan, G. T., & Chandrasekar, C. (2012). A comparison of various edge detection techniques used in image processing. *International Journal of Computer Science Issues (IJCSI)*, *9*(5), 269.

Bijlagen



Figuur 1: Links: origineel, midden: canny met lage threshold, rechts: canny met hoge threshold

Het huidige edge detection algoritme dat in de library opgenomen is, is de laplican methode.

De reden waarom je iets gaat implementeren

Er is iets dat opgelost moet worden. Er is een probleem.

- De kwestie

Er is een situatie die je graag wilt

- De gewenste situatie

De reden waarom je iets gaat implementeren Er is iets dat opgelost moet worden, er is een probleem De kwestie Er is een situatie die je graag wilt De gewenste situatie

Wat doet het algoritme nu:

Gebruikt een matrix met de volgende waardes

```
// this is the kernel -> it gets instantiated
cv::Mat ThoroughBushThoroughBrier = (cv::Mat_<float>(9, 9) <<
    0, 0, 0,  1,  1,  1, 0, 0, 0,
    0, 0, 0,  1,  1,  1, 0, 0, 0,
    0, 0, 0,  1,  1,  1, 0, 0, 0,
    1, 1, 1, -4, -4, -4, 1, 1, 1,
    1, 1, 1, -4, -4, -4, 1, 1, 1,
    1, 1, 1, -4, -4, -4, 1, 1, 1,
    0, 0, 0,  1,  1,  1, 0, 0, 0,
    0, 0, 0,  1,  1,  1, 0, 0, 0,
    0, 0, 0,  1,  1,  1, 0, 0, 0
);
```

Artikel Hoe doe je canny in python: https://towardsdatascience.com/canny-edge-detection-step-by-step-in-python-computer-vision-b49c3a2d8123

(Jochem) Random paper ff gevonden later onderzoeken:
https://pdfs.semanticscholar.org/8b5a/2944dd13bbb6c8a45598e38b3a7af2cf72c9.pdf