— Report P&O CW 2019–2020 Task ST2.2 —

Department of Computer Science – KU Leuven 10 maart 2020

Team 12	
Martijn Debeuf	22h
Toon Sauvillers	22h
Seppe Van Steenbergen	11h

Inhoudsopgave

1 Inleiding

Het vorige verslag behandelde de kleurdetectie, met focus op de te herkennen kleur en de invloeden van omgevingsfactoren. Bij het herkennen van kleuren spelen oneffenheden in de foto's ook een rol, het verslag doet hiernaar verder onderzoek. Een foto is onderhevig aan verschillende factoren. Zo zal een artificiële intelligentie, die nu vaak standaard is ingebouwd in smartphones, de foto al bewerken. Lichtinval, een ander scherm of zelfs een vuil scherm, maakt dat de kleuren niet egaal worden weergegeven.

In sectie 2 komen verscheidene filters aan bod die de oneffenheden en ruwheden in een foto kunnen reduceren. Het zijn deze filters waar het onderzoek zich op focust. Op welke manier doen ze dit, hoe zijn deze geïmplementeerd... Vervolgens diept het verslag de toegepaste methode uit, waarna het meer uitleg geeft over de bevindingen van dit onderzoek. Een eerste zeer belangrijke factor is de tijdsconsumptie van de filter. Weegt de toeneming van uitvoeringstijd op tegen de verbetering van het resultaat? Daarnaast kan gekeken worden naar foto's die initieel niet correct werden gedetecteerd. Zorgt het toepassen van een filter ervoor dat deze wel correct wordt gedetecteerd? Welke filter geeft dan het beste resultaat? Zorgt het aanpassen van de kerngrootte voor betere detectie, en heeft dit effect op de uitvoeringstijd? Ten slotte wordt gekeken of er bevindingen toepasbaar zijn binnen het project.

2 Achtergrond

Er bestaat natuurlijk een breed assortiment aan fotofilters. De benodigde code voor dit onderzoek is geschreven in Python. Daarom is gekozen om vier filters uit de bibliotheek van OpenCV te gebruiken (zie sectie 3). Een eerste toegepaste filter is de *Gaussian Blur*, gevolgd door *Median Blur* en *Mean Blur*. Deze methodes kijken naar de pixels rond een centrale pixel om zijn kleurwaarde te bepalen. Ten slotte is er ook een minder bekende ontruismethode gebruikt, namelijk de *Fast Non-Local Mean Denoising*. Deze werkt anders dan de eerste drie, zie sectie 2.2.

2.1 Methodes met kerngrootte

De volgende methodes maken gebruik van een kerngrootte. Concreet betekent dit dat het algoritme zal kijken naar een aantal rondomliggende pixels om de nieuwe kleurwaarde te bepalen. Bij een kerngrootte van drie zal er een vierkant met zijde drie rond de pixel getrokken worden. Elk algoritme gaat deze pixels anders interpreteren om een nieuwe waarde toe te kennen.

2.1.1 Gaussian Blur

De *Gaussian Blur* zal kijken naar de omliggende pixels. Degenen die verder liggen, zullen minder invloed hebben dan degenen die dichterbij liggen. Een pure *Gaussian Blur* zou naar alle omliggende pixels kijken, ze zullen namelijk allemaal een bijdrage leveren. Door de zware berekeningen die hiermee gepaard gaan, kijkt deze blur enkel naar de omliggende pixels binnenin de door de kerngrootte gedefinieerde omgeving. [1]

2.1.2 Median Blur

Binnen de kern sorteert dit algoritme de pixels van klein naar groot, vervolgens wordt de middelste waarde (mediaan) geselecteerd. De kleur van de centrale pixel van de kern

krijgt deze waarde vervolgens toegeschreven.

2.1.3 Mean Blur

Hier gebeurt hetzelfde als bij de *Median Blur* enkel neemt het algoritme hier het gemiddelde in plaats van de mediaan.

2.2 Fast Non-Local Mean Denoising

Dit algoritme verdeelt de afbeelding in gelijke blokken, deze worden met elkaar vergeleken. Gelijkende blokken worden bij elkaar geplaatst. In elk van deze groepen wordt dan het gemiddelde bepaald. Vervolgens krijgt elke pixel, in elke kern, de gemiddelde waarde toegekend. [2]

3 Methode

Het onderzoek is volledig geautomatiseerd geweest aan de hand van Python. Binnen Python is gebruik gemaakt van de bibliotheek OpenCv om de filters toe te passen op de afbeeldingen. [3] De testen zijn uitgevoerd op een gegevensbank met 670 afbeeldingen van schermen. Net zoals in het vorige onderzoek is gebruikgemaakt van verschillende omgevingsfactoren. Deze zijn echter buiten beschouwing gelaten in het onderzoek.

Om de kwalitatieve data te kwantificeren is gewerkt met een scoresysteem. Een foto wordt voordat deze geanalyseerd wordt, onderverdeeld in een aantal blokken. Daarna worden binnen elk blok tien procent van de pixels willekeurig gekozen. Het onderverdelen in blokken zorgt ervoor dat de pixels mooi verspreid liggen binnen de afbeelding. Daarna worden de verschillende methodes op de foto toegepast. Om uiteindelijk een score aan de afbeeldingen toe te wijzen worden enkel de eerder willekeurig bepaalde pixels bekeken. Deze blijven wel dezelfde voor verschillende filters. De score geeft weer hoe dicht de kleuren van de samples bij de theoretische kleur liggen, met 0 de volledig andere kant van het kleurspectrum en 1 de theoretische kleur zelf.

4 Bevindingen

In dit deel bekijkt het verslag of het al dan niet interessant is om de afbeeldingen te filteren. De eerste belangrijke indicator was tijdsconsumptie. Ook de filters onderling zijn vergelijkbaar met betrekking tot de herkenning van de kleur. Vervolgens kan gekeken worden of de kerngrootte een invloed heeft op de resultaten. Ten slotte gaat het verslag na of filteren effectief het detecteren van de kleur positief beïnvloedt.

4.1 Tijdsconsumptie

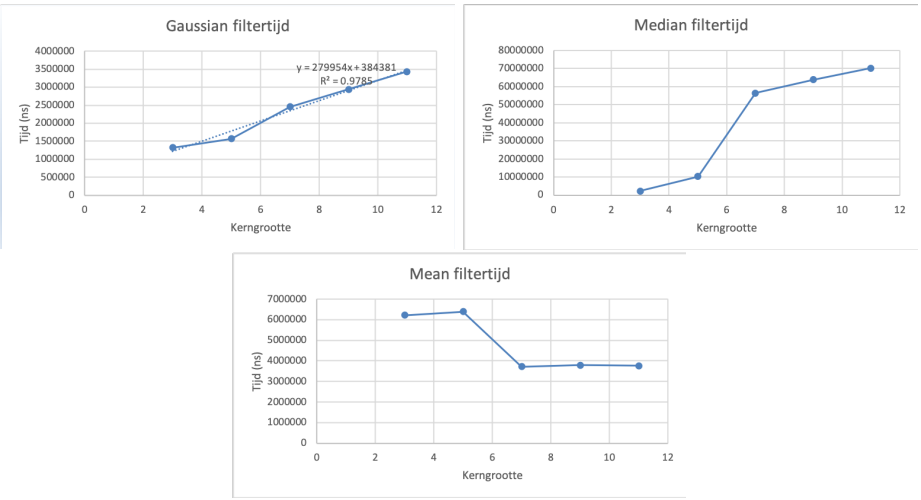
Bij het kijken naar figuur 1 valt meteen op dat *Fast Non-Local Mean Denoising* niet efficiënt is. Dit resultaat ligt binnen de verwachtingen door de manier waarop deze methode te werk gaat, zie sectie 2.2. De waarde blijft nagenoeg constant doordat deze methode geen kerngrootte heeft, hij is trager met grootteorde drie ten opzichte van de andere filters. Om deze reden zal deze filter niet opgenomen worden in het project.

Voor de andere filters kunnen hun individuele tijdsgrafieken, zie figuur 2, van dichterbij bekeken worden. Bij het bekijken van deze tijdsgrafieken, is bij de *Gaussian filter* een duidelijk lineair verband tussen zijn kerngrootte en de tijdsconsumptie zichtbaar. Dit wordt bevestigd door de determinatiecoëfficiënt van 97,85 procent van de lineaire trendlijn.

	Gemiddelde tijd (ns)			
Kerngrootte	Gaussian	Median	Mean	Fast NLM Denoising
3	1319454	2318204	6212660	2944694425
5	1571094	10311129	6388270	2768823310
7	2457046	56403157	3711259	2747259786
9	2936336	63908330	3791051	2784639628
11	3436375	70302982	3761732	2783860072

Figuur 1: Gemiddelde tijd nodig om filter op afbeelding toe te passen.

Verder zijn er nog twee opvallende zaken; enerzijds zorgt een grotere kern ervoor dat de *Median filter* veel meer tijd nodig heeft. De reden hiervoor is dat hoe groter de kern, hoe meer elementen pixelwaarden gesorteerd moeten worden van klein naar groot. Bij de *Mean filter* zou hetzelfde gedrag verwacht worden. Rond kerngrootte zeven is er echter een sprong, de verklaring hiervoor is niet gevonden en eist verder onderzoek.



Figuur 2: Tijdsgrafieken voor de verschillende filters

Gaussian en *Medain blur* geven de beste resultaten, gekeken naar de tijdscomponent. De *Fast Non-Local Mean Denoising* is door dezelfde component afgeschreven.

4.2 Juiste herkenning

Vervolgens bekijkt het verslag of het toepassen van een filter op een onjuist gedetecteerde foto, een al dan niet positief effect heeft. Hiervoor maakt het onderzoek een onderscheid tussen afbeeldingen die initieel juist of initieel fout zijn gedetecteerd. In figuur 3 worden de proporties weergegeven die minder scoren na het toepassen van een filter op een initieel fout gedetecteerde afbeelding. Dit geeft dus het percentage foto's weer die slechter scoort na het filteren ervan.

Het is opmerkelijk hoe dicht alle waarden bij elkaar liggen. Ze leunen allen zeer dicht aan tegen de 100%. Dit wil zeggen dat de filters de resultaten enkel verslechteren bij een al fout gedetecteerde foto. Afbeeldingen die initieel slecht scoren zullen na de filtering

	Filterscore <= ongefilterde score na foute initiële detectie			
Kerngrootte	Gaussian	Median	Mean	Fast NLM Denoising
3	99.43%	99.43%	99.43%	98.86%
5	99.43%	100.00%	99.43%	99.43%
7	99.43%	100.00%	100.00%	98.86%
9	99.43%	100.00%	99.43%	99.43%
11	99.43%	100.00%	99.43%	98.86%

Figuur 3: Proportie slechter gedetecteerde foto's na foute initiële detectie.

geen beter, tot fouter resultaat opleveren.

Daarnaast zijn er degene die initieel juist gedetecteerd zijn. Uit figuur 4 wordt al snel duidelijk dat er hier een verband is met kerngrootte. Indien de kerngrootte zeven à negen bedraagt, zullen alle afbeeldingen een hogere score hebben en dit bij alle geteste filters. Bij een grootte van drie zal ongeveer acht procent van de foto's een hogere score hebben. Kernen van grootte vijf en zeven zorgen voor een middelmaat van respectievelijk om en bij de 40 en 63 procent. Tussen de filters onderling zijn bij gelijke kerngroottes geen significante verschillen op te merken, de waarden zijn zo goed als gelijk. Het slechtere resultaat bij een kerngrootte van 11 is te wijten aan een te grote verspreiding van de oneffenheden.

Deze resultaten wijzen erop dat gebruik van filters bij de herkenning in *ScreenCaster* geen meerwaarde zullen hebben. Bij een foute detectie verbeteren de filters niets en bij een juiste detectie is er nog kans op een slechter resultaat.

	Filterscore <= ongefilterde score na correcte initiële detectie			
Kerngrootte	Gaussian	Median	Mean	Fast NLM Denoising
3	92%	93%	92%	93%
5	58%	58%	58%	58%
7	0%	0%	0%	0%
9	0%	0%	0%	0%
11	37%	37%	37%	36%

Figuur 4: Proportie slechter gedetecteerde foto's na foute initiële detectie.

4.3 Gefilterd en ongefilterd

Om het uiteindelijke oordeel te vellen of filteren al dan niet een verbeterende factor kan zijn binnen het project, doet het verslag nog een laatste vergelijking. Namelijk kijken naar het verschil in juist gedetecteerde afbeeldingen per gebruikte methode, alsook zonder het gebruik van een filter.

Ook hier zijn de waarden nagenoeg gelijk, van de 670 afbeeldingen worden er bij elke toegepaste filter en gebruikte kerngrootte gemiddeld 495 correct gedetecteerd. Dit wijkt slechts af met maximaal één. Hieruit volgt alweer dat een filter toepassen nagenoeg geen effect heeft op het detecteren van de kleuren.

	Hoeveelheid juist gedetecteerde afbeeldingen van de 670				
Kerngrootte	Ongefilterd	Gaussian	Median	Mean	Fast NLM Denoising
3	496	496	496	496	495
5	495	495	496	495	495
7	495	495	496	496	494
9	495	495	496	495	495
11	495	495	496	495	494

Figuur 5: Aantal juist gedetecteerde foto's van de 670, gegeven de gebruikte filter.

5 Besluit

Verschillende filters zijn getest en vergeleken, dit op vlak van tijdsconsumptie en aan de hand van een toegekende score. Die score is berekend op basis van de gedetecteerde waarde van willekeurig gekozen pixels. Uit de resultaten kan geconcludeerd worden dat het toepassen van een filter geen significante verbetering teweegbrengt. De tijdsconsumptie van de filters weegt niet op tegen de prestatiewinst van het algoritme. De foto's bevatten echter maar kleine en wijdverspreide oneffenheden, dit onderzoek zegt niets over grotere vormen hiervan.

5.1 Toepassingen op het project

Door het opvallende resultaat dat bij kleine oneffenheden een filter enkel nadelig is, zal de gebruikte filtermethode in de *ScreenCaster* herbekeken worden. Bij deze foto's is er echter een grotere vorm van oneffenheid, dit vereist specifieker onderzoek. Indien bij de applicatie een filter geen meerwaarde blijkt te hebben, zal deze geschrapt worden met een sneller reagerende applicatie tot gevolg.

Referenties

[1] R. Fisher, S. Perkins, A Walker, and E Wolfart. Gaussian smoothing, 2003.

[2] Kota Yamaguchi. Non-local means image denoising, Nov 2017.

[3] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.