— Verslag P&O CW 2019–2020 Taak 4 —

Department Computerwetenschappen – KU Leuven 4 november 2019

Team 12	
Frédéric Blondeel	1h
Martijn Debeuf	38h
Toon Sauvillers	h
Dirk Vanbeveren	25h
Bert Van den Bosch	32h
Seppe Van Steenbergen	37h

Inhoudsopgave

1 Introductie

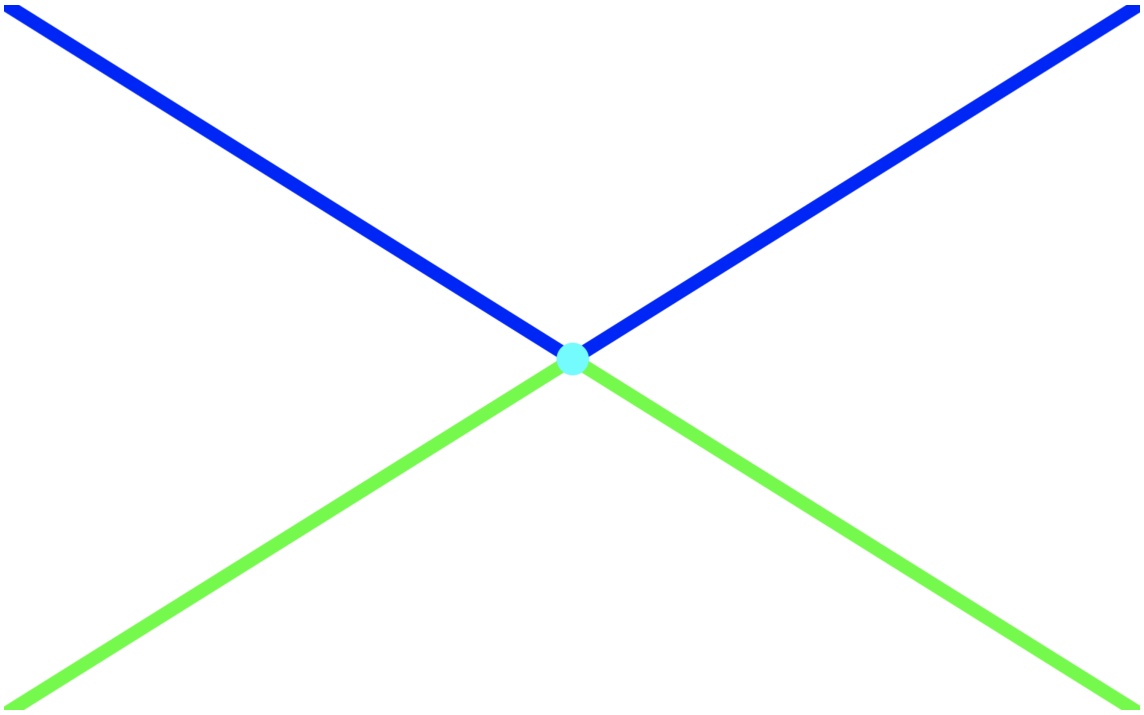
Schermen detecteren terwijl er overlap mogelijk is. Dit verslag behandeld de overgang van "Basic screendetection"naar "Advanced screendetection". Al snel werd duidelijk dat de eerste methode niet meer voldoende zou zijn voor deze nieuwe uitdaging. De achtergrond die tijdens de te nemen foto wordt getoond op de schermen werd gewijzigd alsook de methode om de hoeken te detecteren moest uitgebreid worden. Aanpassingen, keuzes, veronderstellingen en voorwaarden die gemaakt zijn worden behandeld.

2 Algoritmen

2.1 Algemene uitleg

De manier van aanpak werd dus veranderd. In plaats van een border langsheen de rand van het scherm te creëren is geopteerd geweest om de diagonalen te verbinden (een groene en een blauwe) en op het snijpunt wordt een lichtblauwe bol weergegeven, zie figuur 1.

Door deze aanpassing wordt het makkelijker om te achterhalen tot welk island een los deel van een scherm behoort. De huidige voorwaarden om een scherm te kunnen detecteren zijn dat twee aanliggende hoeken en het middelpunt zichtbaar moeten zijn. Achter het kruis wordt nog steeds de barcode geplaatst om de schermen te identificeren. Een offset werd toegevoegd tijdens het vormen van de islands omdat zo de ruis verwijderd wordt rond de overgangen van de achtergrondkleuren.Hieronder volgt een opsomming van de aangepaste methoden die toegepast worden op de invoer afbeelding in chronologische volgorde. Ongewijzigde methoden worden achterwege gelaten in deze opsomming.



Figuur 1: Nieuwe schermachtergrond

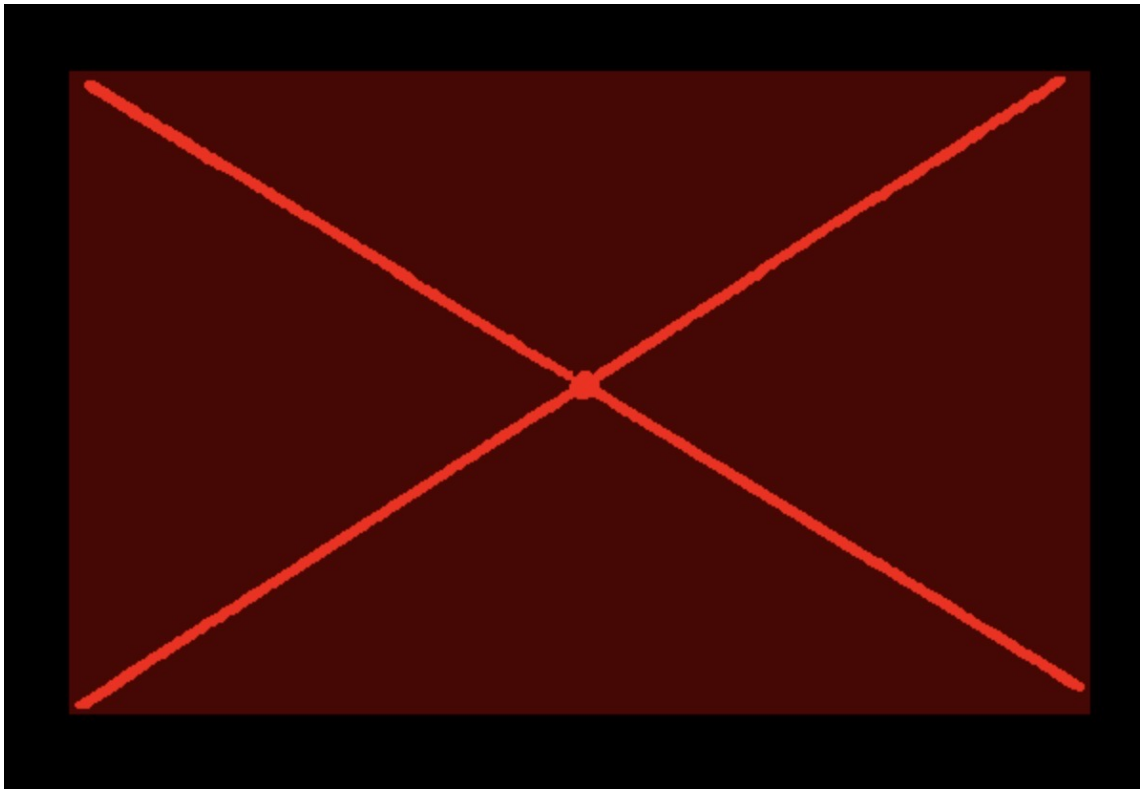
2.2 Kleuren masker

Deze methode werkt nog steeds hetzelfde als bij de vorige taak. Wel werd een extra kleur toegevoegd voor het middelpunt. Groene pixels krijgen een waarde 1, blauwe 2 en lichtblauwe 3 in de matrix. Tijdscomplexiteit van de methode blijft door deze aanpassing onveranderd, nog steeds moeten alle pixels overlopen worden.

2.3 Find Islands

De detectie van een scherm start nog steeds met het zoeken van eilanden van pixels die niet zwart zijn. Bij overlap van schermen en dus mogelijks losliggende stukken scherm in onze matrix, wordt het zeer belangrijk om bij te houden tot welke island een niet zwarte pixel wel degelijk hoort. Hiervoor krijgt elke gestarte island een ID (met tussensprongen van vier om nog steeds de kleuren te kunnen onderscheiden in onze matrix). Wanneer tijdens de floodfill dan een pixel gevonden wordt die bij een reeds gestartte island zou moeten horen, wordt de waarde van deze pixel geïncrementeerd met de desbetreffende ID. [1] Bijvoorbeeld stel dat er reeds een island was en er een tweede ontdekt wordt. Dan krijgt deze tweede island een ID met waarde 3. Een groene pixel die dan tot deze island behoort zal waarde 4

hebben. Eens alle pixels gevonden zijn wordt omtrekkend kader opgesteld aan de hand van de minimum en maximum x– en y–waarden 2.



Figuur 2: Island met kader gevormd na de floodfill

.

2.4 Bepaling middelpunt

Om het middelpunt binnen een kader van een island te bepalen wordt de gemiddelde x– en y–waarde bepaald van alle pixels waarvan de waarde gelijk is aan de island ID plus 2. Pixels die het middelpunt vormen hadden namelijk een lichtblauwe kleur.

2.5 Find Corners

Eerst wordt nagegaan of het scherm voornamelijk recht of gekanteld is ten opzichte van de foto. Hiervoor wordt langs de linkerkant van het kader nagegaan hoe hoog de standaarddeviatie van witte pixels is. Indien hieruit blijkt dat het scherm recht ligt, wordt kant per kant afgegaan welke pixels wit zijn en van deze een gemiddelde positie genomen. Indien het scherm gekanteld blijkt te zijn wordt hetzelfde proces herhaald maar dan gebeurt dit diagonaal. De gevonden hoeken zullen zich dan altijd op de rand van het kader bevinden.

2.6 Clean Corners

De vorige methode zal vier hoeken teruggeven. De mogelijkheid bestaat dat wanneer een scherm geroteerd is, twee hoeken rond hetzelfde hoekpunt gevonden worden. Dit geeft eigenlijk geen problemen? Hierdoor is het namelijk duidelijk dat het scherm niet volledig

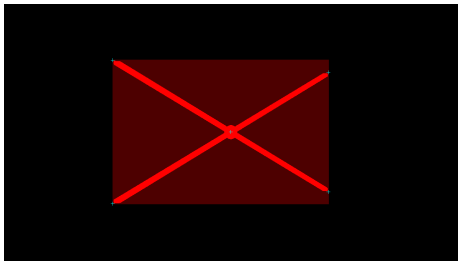
zichtbaar is. Deze methode zal nagaan of twee hoekpunten naar het zelfde punt verwijzen. Indien dit zo is wordt het gemiddelde van de twee opgeslaan als het desbetreffende punt en de andere op "null"geplaats. Daarna moet nog bepaald worden welke de linkse,rechse,onder en boven hoeken zijn. Om deze systematisch bij te houden wordt gebruik gemaakt van een dictionary met als keys "LU","RU","RD" en "LD". Deze toewijzing gebeurt aan de hand van de punten die op “null “ geplaatst zijn geweest.

2.7 Reconstructie

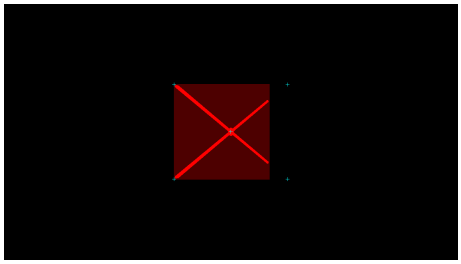
Indien na het controleren van de hoeken nog steeds vier hoeken worden overgehouden, wordt nagekeken of de afstand van overstaande hoeken tot het middelpunt ongeveer gelijk is. Indien dit niet het geval is, dit betekend dat het scherm niet volledig zichtbaar is, maar het kruis toch mooi op de rand van het kader liggen, wordt het punt waarvan de afstand tot het middelpunt het grootst is gepuntspiegeld rond de oorsprong. In het geval dat er minder dan vier punten gevonden zijn (2 is het minium, zie onze voorwaarden). Wordt gekeken welke hoek null is en wat de overstaande hoek van deze is zodanig dat, net zoals hiervoor vermeld, deze hoek kan gepuntspiegeld worden. Deze methode is dus ook in staat om te melden wanneer er niet aan de voorwaarden voldaan is geweest. De reconstructie zal volgens de meetkunde slechte resultaten geven wanneer de hoek waaronder de foto getrokken wordt te groot is. Hiervoor is al een oplossing maar deze is nog niet geïmplementeerd (zie secties 3 en 4 voor meer).

3 Testen

Het grootste probleem bij overlappende schermen is de verloren informatie te recupereren en te reconstrueren. Een gevonden hoek wordt volgens het reconstructie algoritme enkel correct bevonden als hij minstens voldoende dicht bij het middelpunt van het scherm ligt tegenover zijn overstaande hoek. De gebruikte threshold is dat de afstand van een hoek tot het middelpunt minimaal 70 procent van zijn overstaande hoek moet zijn. Als dit niet het geval is zal de puntspiegeling van zijn overstaande hoek gebruikt worden. Om te testen of deze verhouding en puntspiegeling stand houdt onder verschillende omstandigheden, zijn er in het 3D-software pakket Blender scenes gemaakt om een realistische situatie te creëren. Omdat er gewerkt kan worden met een echte camera en rotatie volgens drie dimensies, kon het algoritme nauwkeurig getest worden. Door de grootte van het scherm en de afstand van de camera tegenover het scherm constant te houden, kon de invloed van de rotatie van het scherm en de focal length van de camera als variabele ingesteld worden om zo telkens de resultaten van het reconstructie algoritme te testen. De testreeksen werden verdeeld als volgt: per focal length werd het scherm van 0 to 90 graden per 5 gedraaid. Omdat de meest voorkomende smartphone cameras een focal lenght hebben van 25mm tot 35mm werd de focal length getest van 20mm tot 40mm met een stap van 2mm. Er werden dus in totaal 180 beelden gemaakt en getest. De resultaten liggen niet al te ver af van de verwachtingen. Het algoritme houdt zeer goed stand bij relatief kleine kleine vervormingen door de focal length, zeker omdat de kleinst geteste focal length van 20mm wel wat kleiner is dan de de gemiddelde camera. Als het scherm weg begint te draaien geeft het reconstructie algoritme vaak een reconstructie waar er eigenlijk geen nodig is. Dit gebeurde gemiddeld rond de 60 tot 65 graden. Dit is waar een enkele threshold te kort schiet om een onderscheid te maken of het scherm niet volledig zichtbaar is, of relatief scherp gedraaid.



Figuur 3: Correct resultaat



Figuur 4: Foute reconstructie

4 Valkuilen

De grootste valkuil in het detecteren is de reconstructie die nog gevoelig is voor de hoek waaronder de foto genomen is. De meest voor de hand liggende oplossing is werken met een drempel die afhankelijk is van de grootte van het scherm om te kijken wanneer een punt moet gereconstrueerd worden of niet. Deze aanpak lijkt niet ideaal en daarom wordt het volgende voorgesteld. Om later een foto te kunnen weergeven op de schermen moet een transformatiematrix opgesteld worden per scherm. Hieruit kan dan al een transformatiematrix opgesteld worden die de driehoek transformeert naar de originele afmetingen van het scherm. Daarna kunnen de ontbrekende hoekpunten bepaald worden.

5 Besluit

Het overschakelen naar een nieuwe methodes voor screen- en corner-detection bracht wel wat uitdagingen met zich mee. Onze huidige resultaten geven voor veel gevallen een bruikbaar resultaat, maar er is wel duidelijk ruimte voor verbetering om meerdere situaties aan te kunnen. De methodes waar we al oog op hebben voor vebetering zijn de maskeer functies en het reconstructie algoritme. Als deze nog verfijnd en op punt gesteld kunnen worden zal er een grote verbetering van onze huidige resultaten zijn, maar ook een veel breder spectrum van opstellingen aankunnen.

Referenties

[1] Wikipedia floodfill. https://nl.wikipedia.org/wiki/Floodfill-algoritme.