Implementatieplan titel

Jullian van Doorn en Jari van Dam 19-3-2018

Doel

De implementatie moet edges zo snel mogelijk kunnen detecteren uit afbeeldingen. Er moet een hoger percentage gezichten juist worden herkent binnen een bepaalde tijdsframe. Dit betekent dat de snelheid van deze implementatie hoger moet zijn.

Methoden

Roberts cross

Roberts cross is een vrij oud algoritme. Dit algoritme is daarom ook heel snel want toentertijd waren computers lang niet zo snel. Het algoritme is dus ook heel simpel qua karakter. Het zou mogelijk zijn om deze methode te implementeren zodat de edges wellicht wel van lagere kwaliteit zijn in slechtere afbeeldingen. Als er een stroom van afbeeldingen is of een afbeelding met een hoog contrast, zouden de edges mogelijk voldoende zijn om gezichten te herkennen.

Bron: https://en.wikipedia.org/wiki/Roberts_cross

Sobel operator

De Sobel operator is vijf jaar jonger dan de Roberts Cross. Deze operator verschilt met de Roberts Cross op twee manieren. De matrix van de Sobel operator is 3x3 i.p.v. 2x2 en deze matrix is overigens ook separabel. Dus in principe kan de snelheid hoger zijn doordat gebruik gemaakt kan worden van multithreading en er kan meer geheugen ingezet worden dan bij de Roberts Cross methode.

Bron: https://en.wikipedia.org/wiki/Sobel_operator

Prewitt operator

De Prewitt operator lijkt op de Sobel operator. Het enige verschil is de matrix van de operator. Dit leidt dus niet naar verschillen in memory consumption of snelheid. Het kan alleen uitmaken in gevallen waar een edge zou word gedetecteerd waar dat niet zou moeten en andersom. Dit hangt af van het plaatje en algoritme. In theorie zou de Sobel operator een meer smoothing effect hebben. Dit kan een voordeel zijn en tegendeel.

Bron: https://en.wikipedia.org/wiki/Prewitt_operator

Kirsch operator

De Kirsch operator berekent voor acht verschillende richtingen het reliëf. Dit betekent meteen dat deze operator 8 keer meer operaties moet doen dan de voorgaande drie methodes. Daarentegen kan deze operator wel precies de edges te detecteren. Maar de accuratie-performance benefit is vergeleken met simpelere algoritmes minimaal.

Bron: https://en.wikipedia.org/wiki/Kirsch_operator

Canny edge

De Canny edge detector is een algoritme met meerdere stappen. Meestal wordt van te voren de input afbeelding geblurd, zodat de noise uit de afbeelding gaat. Daarna wordt een operator als die van Robert of Sobel toegepast, om de eerste edges te detecteren. Dan worden alle edges verdunt zodat ze maar één pixel breed zijn en er dus maar één representatie van de edge is. Vervolgens wordt gekeken of de weak edges, edges met een lage intensiteit, verbonden zijn met een strong edge,een edge met een hoge intensiteit, zo niet wordt deze edge genegeerd.

Bron: https://en.wikipedia.org/wiki/Canny_edge_detector

Keuze

Om gezichten sneller te herkennen is er gekozen om een stukje accuratie in te ruilen voor snelheid. Dit zal gedaan worden door Canny edge detection uit de gezichtsherkenning te halen en hiervoor in de plaats een goedkoop algoritme zoals Sobel te implementeren. Dit algoritme is overigens ook makkelijk te implementeren.

Implementatie

De Sobel operator wordt geïmplementeerd binnen `IntensityImage* StudentPreProcessing::stepEdgeDetection(const IntensityImage&).` Er worden geen nieuwe hulp klassen toegevoegd omdat die alleen maar C++ overhead bezorgen en het lastig is om dit weg te optimaliseren. Er zal moeten worden getuned met parameters van de Sobel operator om gezichtsherkenning te kunnen laten werken.

Evaluatie

De eigen implementatie zal getest worden op snelheid en geheugengebruik. De snelheid zal gemeten worden door de uitvoertijd van de code te vergelijken met de OpenCV. Dit kan aan de hand van een timer library die telt hoeveel nanosecondes er voorbij zijn gegaan tussen het begin en het einde van de edge-detection en thresholding.

De geheugengebruik moet worden gemeten door middel van een memory usage profiler. Visual Studio heeft deze ingebouwd dus het is goed mogelijk om deze te gaan gebruiken. Het is belangrijk om het geheugengebruik te meten om te achterhalen of deze techniek bruikbaar is binnen embedded systems. Embedded systems hebben vaak beperkte resources dus moet er moeten worden geoptimaliseerd voor snelheid en geheugengebruik.

Er wordt verwacht dat wanneer de parameters juist zijn getuned dat de snelheid en geheugengebruik van de student-implementatie beiden de OpenCV implementatie overtreffen. Waarbij het geheugengebruik minder is dan OpenCV evenals de gemiddelde tijd om één gezicht te herkennen.