1. Implementatieplan titel

1.1. Namen en datum

Jessy Visch
01-05-2018

1.2. Doel

Het doel van deze odpracht is de huidige implementatie van edge detection te vervangen door een eigen implementatie. De output van de implementatie moet zodanig op het origineel lijken dat de volgende stappen van het programma nog steeds blijven werken.

1.3. Methoden

Er zijn meerdere methoden om edges te detecteren.

Sobel
Als eerste is er de Sobel operator. Een sobel operator is samengesteld bestaat uit 2 kernels/filters: 1 voor de X en 1 voor de Y. Deze kernels zien er als volgt uit:

-1	0	+1
-2	0	+2
-1	0	+1

x filter

+1	+2	+1
0	0	0
-1	-2	-1

y filter

Deze 2 filters worden gecombineerd. Om deze kernel te faciliteren moet er gebruikt worden gemaakt van een blur: Hierbij kan gebruik worden gemaakt van de bovengenoemde meanfilter of een gaussian blur. De edges worden in het output plaatje aangeduid in het wit.

Laplacian
De tweede operator die kan worden gebruikt is de Laplacian Operator. Het mooie aan deze operator is dat hij op verschillende manieren kan worden gebruikt. Je kan een simpele standaard 3 bij 3 kernel gebruiken met een losse blur erbij. Je kan de kernel ook uitbreiden tot een grotere (bijvoorbeeld 7 bij 7) kernel met een 'ingebouwde' blur.

0	−1	0
−1	4	−1
0	−1	0

3x3 laplacian

Blur
Om de afbeelding te blurren bestaan er meerdere blurs. Waaronder de Box blur, de median filter en de gaussian blur.

1.4. Keuze

Er is gekozen voor de laplacian kernel. Het vermoeden bestaat dat omdat lapliacian slechts 1 kernels is dit efficienter is dan meerdere kernels te gebruiken bij bijvoorbeeld Sobel. Daarnaast wordt er ook geexperimenteerd met verschillende blurs. Allereerst wordt er een median filter gebruikt, omdat deze zeer simpel is. Wanneer dit niet voldoet wordt er gebruik gemaakt van een gaussian blur. Om te zorgen dat de vervolgstappen van het programma de afbeelding correct kunnen gebruiken, wordt er een zero-crossings functie geschreven die alle edges omzet naar 0 of 255 afhankelijk van hun huidige waarde.

1.5. Implementatie

Om bewerkingen via Kernels op afbeeldingen faciliteren zijn de studentKernel en studentPreProcessing classes gemaakt. In deze classes zijn onder andere functies voor tresholding, Convolutie en zeroCrossing geimplementeerd. Daarnaast is er een StepEdgeDetection functie waarin het alles bij elkaar komt en de verschillende bewerkingen op de afbeeldingen worden uitgevoerd.

1.6. Evaluatie

Om te bewijzen dat de geschreven implementatie correct is worden alle afbeeldingen van de testset door het programma gebruikt en wordt de output hiervan vegeleken met de output van de huidige implementatie. Daarnaast wordt ook de tijd die het kost om de edges te detecteren gemeten en vergeleken met de bestaande implementatie. De verwachting is wel dat de eigen implementatie niet sneller gaat zijn dat de huidige (OpenCV) implementatie.