1. Edge detection

1.1. Namen en datum

Sander Kolman
Alexander Freeman
4-5-2016

1.2. Doel

Voordat localisatie en extractie gedaan kan worden, moet er preprocessing worden uitgevoerd. De onderdelen die wij implementeren zijn edge detection en thresholding.

1.3. Methoden

Om edge detection te implementeren zijn er meerdere methoden.

Laplacian

De eerste is de, in de les uitgelegde, Laplacian operator. Laplacian Edge Detection heeft voordelen en nadelen. Het werkt door middel van een standaard convolutie kernel en een methode om edges te halen uit de output hiervan. Deze kernel kan 3x3 zijn, maar er zijn ook kernels waar bijvoorbeeld al de blurring inzit. Deze heet ook wel de LoG (Laplacian of Gaussian). Verder zijn er ook bijvoorbeeld ‘normale’ kernels van 5x5 , 7x7 of hoger. De output van de kernel is een plaatje waarin de edges zijn aangegeven met een overgang van zwart naar wit.

Laplacian gebruikt de afgeleide van de afgeleide. Dit zorgt ervoor dat het erg gevoelig is voor noise. Het is echter makkelijk te implementeren en detect met maar 1 pass van een convolutie kernel edges die diagonaal, verticaal en horizontaal zijn.

Sobel

Een andere vorm is de Sobel operator. Deze werkt op basis van de eerste afgeleide en gebruikt 2 kernels: één voor de x-as en één voor de y-as, die daarna worden samengevoegd. Dit zijn ook standaard convolutie kernels en deze hebben een plaatje waarbij de edges wit zijn als output. Hieruit kunnen dan de edges worden gehaald. Het is niet zo gevoelig voor noise als de Laplacian edge detection, maar het versterkt het nog steeds. Dit zorgt ervoor dat blurring ook nodig zal zijn.

Canny edge detection

De laaste vorm die we bekeken hebben is de canny edge detection. Het is een algorithme met meerdere stappen dat verschillende technieken combineert, waaronder de Sobel operator. Het zorgt voor minimale edges, die zo dun mogelijk zijn. Deze edges worden ook als het ware doorgetrokken, waardoor minder duidelijke edges die een verlengde zijn van een duidelijke edge ook mee worden genomen. Dit heeft als gevolg dat de edges die gevonden zijn, erg duidelijk zijn. Ook zorgt dit doortrekken ervoor dat false positives voorkomen worden. Dit is handig als er veel noise is. Dit is echter een lastig algorithme om te implementeren, wat meerdere technieken vereist.

Blurring

Voor de blurring die nodig is bij de Sobel en de Laplacian operators zijn er meerdere mogelijkheden. De makkelijkste 3 om te implementeren zijn de median filter, de gaussian blur en de box blur.

1.4. Keuze

Wij hebben gekozen voor de laplacian operator. We hebben voor deze gekozen om meerdere redenen. Allereerst is het maar één kernel die gebruikt hoeft te worden voor de edge detectie zelf. Dit zorgt ervoor dat het waarschijnlijk efficienter zal zijn, dan de sobel kernels. Daarnaast waren we hier al mee aan het experimenteren en waren we al aardig ver met de implementatie ervan.

Verder gaan we eerst experimenteren met blurring. We willen eerst kijken of box blurs voldoende zijn om een goed resultaat te behalen. Indien dat niet zo is, gaan we experimenteren met gaussian blurs, in de vorm van de LoG.

Aangezien dit aan moet sluiten op een bestaande codebase, moet ook bekeken worden wat de beste methode is om de edges uit het resultaat te halen. Bij de laplacian operator wordt dit gedaan door middel van zero crossings, het punt waar de sign overgaat van + naar - of omgekeerd. Misschien kan de code hier echter niet mee omgaan, aangezien de default implementatie niet hetzelfde resultaat teruggeeft als voorbeelden van de laplacian op internet..

1.5. Implementatie

Om kernels toe te passen op onze afbeeldingen, gebruiken we de geschreven StudentKernel class. Deze class heeft methods om bijvoorbeeld convolution, dilation en erosion op een afbeelding toe te passen.

Daarnaast schrijven we een method om de edges uit het resultaat van de edge detection te halen. Dit willen we eerst dus proberen met zero crossings, maar als dat niet lukt proberen we het door middel van de afstand naar de ‘geen edge’ value. In dit geval is dat 127. Hiermee halen we edges in beide richtingen uit het plaatje en combineren we dit tot één afbeelding.

1.6. Evaluatie

Om te controleren of onze edge detection aansluit op de bestaande codebase, gaan we de plaatjes van de test-set langs en kijken we hoeveel plaatjes in de standaard routine worden gedetecteerd. Dit vergelijken we met de standaard implementatie. Daarnaast willen we kijken of onze implementatie langzamer is en hoeveel dat verschilt.
Daarnaast willen we controleren wat het verschil is qua output van ons algorithme en die van de default implementatie. Dit doen we aan de hand van een visuele inspectie van de output met de test-set als input.