Implementatieplan Practica Vision

Edge Detection

8 maart 2018

Door:

- Kiet van Osnabrugge
- Wiebe van Breukelen

Inhoud

1.1. Doel

Het doel van onze implementatie is om meer bruikbare informatie uit de edges te halen om het mogenlijk te maken een extra verwerking toe te passen op de edges afbeelding. Denk hierbij aan het berekenen van hoeken en lijndiktes met behulp van de edge gradienten. Met deze informatie is het bijvoorbeeld mogelijk om onbelangrijke edges te kunnen negeren en onbekende edges in te vullen. (Dit valt buiten de scope van deze opdracht.)

1.2. Methoden

Laplacian edge detection (standaard implementatie):

Levert alle edges op maar is erg gevoelig voor ruis (dit is iets te verbeteren met behulp van Gaussian ruis). Met Laplacian edge detection is het niet mogelijk om een richting van een edge te bepalen (met Prewitt en Sobel is dit wel mogelijk).

Prewitt edge detection

Edge detection voor de x en y gradient met lager threshold (relatief lagere waardes in vergelijking met het middelpunt van de kernel). De gradienten die bepaald worden met behulp van de verticale en horizontale kernel worden gebruikt om de hoek van de edges te bereken. Levert (mogelijk te veel onnodige edges) edges op met een richting.

Sobel edge detection

Edge detection voor de x en y gradient met hoger threshold (in tegenstelling tot Prewitt). Deze gradienten worden gebruikt om de hoek van de edge te bereken. Levert (minder onnodige edges dan prewitt) edges op met een richting.

1.3. Keuze

Wij kiezen voor de Prewitt en de Sobel edge detection methodes, omdat deze beiden met de hulp van dezelfde verwerkingsstap verbetert kunnen worden (Canny). Het verschil tussen Prewitt en Sobel is de threshold, Prewitt haalt veel sneller een edge uit het plaatje dan Sobel. Dit resulteert waarschijnlijk in dat de eerste stap langzamer voltooid zal worden, omdat er meer edges zijn. We weten niet welke van de twee methoden een beter beeld op levert. Dit hangt

mogelijk af het geteste plaatje. In bepaald lichtsituaties zou de ene methode relatief beter functioneren dan de andere.

1.4. Implementatie

Onze implementatie komt in de klasse `StudentPreProcessing` te staan. Wij gaat de methode `stepEdgeDetection en stepThresholding` realiseren. Hierbij wordt de functionaliteit opgedeeld in verschillende functie's, namelijk:

- calcGradientX: Bereken gradient in de X-richting. Wordt gebruikt voor edge detection.
- calcGradientY: Bereken gradient in de Y-richting. Wordt gebruikt voor edge detection.
- calcMeanCenterPixel: Bereken de gemiddelde pixelwaarde voor het centrum (afbeelding – randen) van een afbeelding. Wordt gebruikt voor thresholding.
- calcMeanCornerPixel: Bereken de gemiddelde pixelwaarde voor de randen van de afbeelding. Wordt gebruikt voor thresholding.

Zoals in de sectie keuze vernoemd zouden wij eventueel Canny willen implementeren, maar een mogelijk risico is dat de data die door Canny geleverd wordt niet goed geïntegreerd kan worden in de bestaande implementatie.

1.5. Evaluatie

- **Robuustheid**: De resolutie van de afbeelding negatief veranderen en de uitkomst beoordelen aan de hand van het originele beeld. We willen op deze manier vergelijken of Prewitt/Sobel beter functioneren bij verschillende resoluties.
- **Snelheid**: Onderzoeken of Prewitt of Sobel de hoogste verwerkingssnelheid heeft. Met behulp van runtime timers kan bepaald worden welke methode sneller is. De snelheid van het edge detecie proces en het thresholding proces zal genomen worden om snelheid verschillen beter te kunnen verklaren. Dit wordt ook getest bij verschillende resoluties.