Implementatieplan Vision

Rik Buimer, Kyle Taylor Parkins 20 juni 2017

Doel

In deze opdracht wordt een studentimplementatie ontwikkeld om onderzoek te doen naar Edge Detection. Hierbij wordt er gebruik gemaakt van kernels en onderzoek gedaan naar de beste manier.

Methoden

Er zijn meerdere kernel filters beschikbaar. De volgende hebben wij uitgelicht:

- Canny. Het meest efficiënt.
- Sobel
- Prewitt
- Laplacian.

Keuze

We hebben de keuze gemaakt voor het Laplacian filter. Dit is tijdens de lessen goed uitgelegd, beheersen wij enigszins en er zijn vrij concrete voorbeelden te vinden op internet.

Hierbij is tevens een thresholding functie gemaakt om de ruis in de resulterende afbeelding te verminderen. Dit maakt ook de lijnen wat duidelijker.

Implementatie

De threshold functie is vrij compact, zie de volgende code:

```
IntensityImage * StudentPreProcessing::stepThresholding(const IntensityImage &image) const {
    IntensityImage* IM = ImageFactory::newIntensityImage(image.getWidth(), image.getHeight());
    int size = image.getWidth() * image.getHeight();
    int threshold = 220; // drempelwaarde, kan aangepast worden
    for (int i = 0; i < size; i++) {
        Intensity in = image.getPixel(i);

        if (in > threshold) {
            IM->setPixel(i, 0);
        }
        else {
            IM->setPixel(i, 255);
        }
    }
    return IM;
}
```

De volgende code is gerealiseerd voor de laplacian edge detectie.

```
IntensityImage * StudentPreProcessing::stepEdgeDetection(const IntensityImage &image) const {
    IntensityImage* IM = ImageFactory::newIntensityImage(image.getWidth(), image.getHeight());
    int offset = 0; // offset voor randen
    int kernelWidth = 3; // breedte van de kernel
    int blockSize = 3;
    int imageWidth = image.getWidth();
    int pixSum = 0; // som van pixelwaarden
    int pixVal = 0;

    // loop door de image heen
    for (int x = offset; x < (image.getWidth() - offset); x++) {
        for (int y = offset; y < (image.getHeight() - offset); y++) {
            pixSum = 0;
            // loop door kernel heen per pixel
            for (int kx = 0; kx < kernelWidth; kx++) {
                for (int ky = 0; ky < kernelWidth; ky++) {
                    pixVal = 0;
                    if (kernel[ky * kernelWidth + kx] == 0) {
                        continue;
                    }
                    for (int bx = 0; bx < blockSize; bx++) {
                        for (int by = 0; by < blockSize; by++) {
                            // bereken de waarde na het toepassen van de kernel
                            pixVal += image.getPixel( (x - offset + bx + (kx * blockSize)) +
                                    (imageWidth * (y - offset + by + (ky * blockSize))));
                        }
                    }
                    pixSum += pixVal * kernel[ky * kernelWidth + kx];
                }
            }
            // set pixelwaarde binnen itensity range
            if (pixSum < 0) {
                pixSum = 0;
            }
            else if (pixSum > 255) {
                pixSum = 255;
            }
            // bouw nieuwe intensityimage
            IM->setPixel(x + imageWidth * y, pixSum);
        }
    }
    return IM;
}
```

Evaluatie

De code voor de edge detectie is nu klaar. We hopen met het laplacian filter goede resultaten te behalen tijdens het meten. De resultaten van deze testen zullen worden opgenomen in het meetrapport.