1. Implementatieplan edge detection

1.1. Namen en datum

Daniël van den Berg - 1636304
Hylco Uding - 1635936

Dinsdag 26 Mei 2015

1.2. Doel

Het doel van de implementatie "edge detection" is om een afbeelding te creëren die alleen de edges bevat. Deze afbeelding zal zwart-wit zijn. Wit is geen edge, zwart is wel een edge.

1.3. Methoden

Voor edge detection zijn meerdere methoden. Dit zijn onder andere: Laplacian, prewitt, sobel, etc. Dezen gaan allemaal uit van een kernel die de edge-waarde zal teruggeven. Deze waarde zal dan door middel van thresholding onderscheiden moeten worden van de plekken waar geen edges zijn.

1.4. Keuze

Wij kiezen voor de implementatie voor laplacian. Dit doen we omdat dit de makkelijkste methode is om te implementeren. Ook werkt deze methode goed met integers, wat een voordeel is voor (normale) computers die niet floating-point georienteerd zijn.

1.5. Implementatie

Wij implementeren deze laplacian als volgt:

```
const int size = 9;
	const int halfSize = floor(size / 2.0f);
	int mask[9][9] = { { 0, 0, 0, 1, 1, 1, 0, 0, 0 },
	{ 0, 0, 0, 1, 1, 1, 0, 0, 0 },
	{ 0, 0, 0, 1, 1, 1, 0, 0, 0 },
	{ 1, 1, 1, -4, -4, -4, 1, 1, 1 },
	{ 1, 1, 1, -4, -4, -4, 1, 1, 1 },
	{ 1, 1, 1, -4, -4, -4, 1, 1, 1 },
	{ 0, 0, 0, 1, 1, 1, 0, 0, 0 },
	{ 0, 0, 0, 1, 1, 1, 0, 0, 0 },
	{ 0, 0, 0, 1, 1, 1, 0, 0, 0 } };
```

```
IntensityImage * res = ImageFactory::newIntensityImage(image.getWidth(),
image.getHeight());
	for (int x = halfSize; x < image.getWidth() - halfSize; x++){
		for (int y = halfSize; y < image.getHeight() - halfSize; y++){
			int value = 0;
			for (int _x = 0; _x < size; _x++){
				for (int _y = 0; _y < size; _y++){
					value += image.getPixel(x + _x - halfSize, y + _y -
halfSize) *mask[_x][_y] ;
				}
			}
```

```
                    value = value*2 + 127;
                    if (value < 0){
                          value = 0;
                    }
                    else if (value > 255){
                          value = 255;
                    }
                    res->setPixel(x, y, value);
             }
       }

       for (int x = 0; x < image.getWidth(); x++){
             for (int y = 0; y < halfSize; y++){
                    res->setPixel(x, y, 0);
                    res->setPixel(x, image.getHeight() - y - 1, 0);
             }
       }

       for (int y = 0; y < image.getHeight(); y++){
       for (int x = 0; x < halfSize; x++){
                    res->setPixel(x, y, 0);
                    res->setPixel(image.getWidth() - x - 1,y, 0);
             }
       }
       return res;
```

Deze implementatie loopt over de volledige afbeelding heen. Voor elke pixel loopt hij ook door de gehele max heen. De waarden telt hij bij elkaar op, wordt verdubbeld om duidelijker onderscheid te maken tussen wel een edge en geen edge, en er wordt 127 bij opgeteld. Dit om te zorgen dat grijs het middenpunt is, in plaats van zwart.

1.6. Evaluatie

Als evaluatie zal een visuele inspectie gedaan worden.