1. Week 5. 02-06-2015 (Scaling)

1.1. Week5. 02-06-2015

Gemaakt door Thijs van Tiem & Jos Roijakkers

1.2. Doel

Het doel van deze opdracht is de vision afbeelding te scalen naar een nieuw formaat. Hiervoor moet je kijken naar de mogelijkheden en een keuze maken. Aan de hand van je gemaakte keuze ga je de scaling toepassen met het algoritmen dat je hebt gekozen.

Het totale doel van de opdracht is meer te weten te komen over scaling.

1.3. Methoden

Er zijn een aantal verschillende onderzochte methoden om scaling toe te passen namelijk:

- Zero order interpolatie ook wel nearest neighbor. Deze manier kijkt naar de naast liggende pixel en beslist aan de hand van zijn 'buurman' de kleur van de nieuwe pixel.
- First order interpolatie ook wel bilinear interpolation. Deze manier pakt voor de nieuwe pixel het gewogen gemiddelde van 4 naastliggende pixels.
- Higher order interpolatie ook wel third-order/bicubic interpolation. Deze manier pakt voor de nieuwe pixels het gewogen gemiddelde van de 16 naastliggende pixels

1.4. Keuze

De keuze is gevallen op Zero order interpolatie doormiddel van backwards mapping. Deze keuze is gemaakt nadat er eerst vooraf onderzoek is verricht naar de verschillende mogelijkheden en hun implementaties. Uit dit onderzoek kwam Zero order naar voren als een gemakkelijke en prima werkende methode en is de keuze gemaakt om dit eerst eens te proberen. Daarnaast was hier de meeste informatie over te vinden op het web.

1.5. Implementatie

De zero interpolatie is als volgt toegepast in de code: eerst wordt een nieuwe intensity image aangemaakt deze gaan we dan scalen. We geven de scale aan die we uiteindelijk willen hebben als constanten. Er wordt een dubbele for-lus gemaakt om de x en de y pixels op te halen en hier wordt dan de nearest neighbor berekening overheen gezet(auto Xold = round(x / schaal) hier wordt dus gekeken naar de oude pixel waarde en hier wordt dan x of y gedeeld door de schaal om aan de nieuwe waarde te komen voor de pixel.(round houdt in dat het wordt afgerond zodat er geen getallen achter de komma kunnen voorkomen.) Deze nieuwe waardes worden dan geretourneerd in het nieuwe plaatje.
De code ziet er zo uit:

```
auto vergrootObject = ImageFactory::newIntensityImage(image.getWidth(),
image.getHeight());
	const int scaleX = 300;
	const int scaleY = 300;

	double inputSize = image.getWidth()*image.getHeight(); double realSize =
	static_cast<double>(scaleX)*static_cast<double>(scaleY);
	double schaal = 1.0 / sqrt(inputSize / realSize);
```

```
        vergrootObject->set(static_cast<int>(image.getWidth()*schaal),
        static_cast<int>(image.getHeight()*schaal));
        for (int x = 0; x < vergrootObject->getWidth(); ++x){
                for (int y = 0; y < vergrootObject->getHeight(); ++y){
                        auto Xold = round( x / schaal);
                        auto Yold = round( y / schaal);
                        vergrootObject->setPixel(x, y, image.getPixel(Xold, Yold));
                }
        }
        return vergrootObject;
}
```

1.6. Evaluatie.

Tijdens het implementeren zullen er verschillende personal tests worden uitgevoerd om te achterhalen wat geïmplementeerd wordt ook echt werkt. Zodra dit positief resultaat geeft wordt er gekeken of de code verbeterd kan worden binnen een reëelen tijd.. Uiteindelijk wordt er getest of de zelf gemaakte code beter of slechter functioneert dan de default code van de leraar en word dit gedocumenteerd.