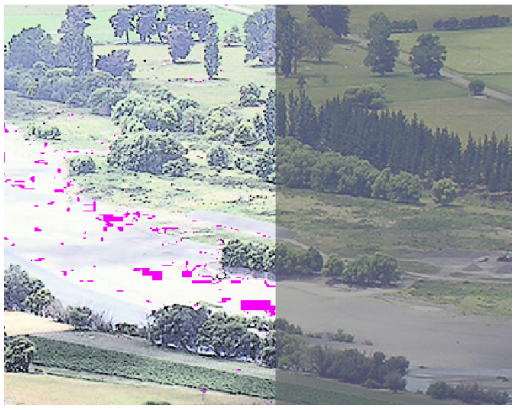
Verslag Globale filters

Jonas Van Der Donckt

25 mei 2018

1 Belangrijke beslissingen code

In de eerste versie van de code werd het resultaat in figuur 1 verkregen. Hierin kan gezien worden dat de witte delen van de afbeelding artefacten bezitten. Dit kwam doordat er niet gecontroleerd werd of de delta waarde nul was. Er werd namelijk in latere stukken code door de delta gedeeld. Het resultaat van deze deling is niet voorspelbaar indien delta nul is. Na de aanpassing van de code (zie hieronder), werd het resultaat in figuur 2 verkregen.



Figuur 1: Artefacten

Figuur 2: Geen artefacten

```
1    ...
2    const unsigned char max = std::max(r, std::max(g, b));
3    const unsigned char min = std::min(r, std::min(g, b));
4    float delta = max - min;
5    if (delta == 0)   // to make sure we don't devide by zero
6        delta = 1;
7    ...
```

2 Vergelijking uitvoeringstijd CPU-GPU

Er werden 6 resultaten opgenomen. De uitvoertijden zijn samengevat in onderstaande tabel.

	gem CPU	**stdv CPU**	**gem GPU**	**stdv GPU**	**(gem) CPU / GPU**
histogram	116166	2 334	17 899	786	6.49

Tabel 1: Alle gemidelden en standaard deviaties in us

Het valt op dat er een significante hoeveelheid tijd gewonnen wordt door het algoritme uit te voeren op de GPU.

De uitvoertijd (voor de GPU) kan verkleind worden door nog een kernel toe te voegen die eerst de frequenties van de workgroups bepaalt. Daarna worden deze frequentie tabellen (op de CPU) gesommeerd. Zo wordt de frequentie tabel van de globale afbeelding berekend. Hierna zijn alle stappen hetzelfde als de huidige GPU code.

Deze lagere uitvoertijd zou aan het feit liggen dan de CPU niet meer over elke pixel moet itereren.