Meetrapport snelheid bilineaire interpolatie bij schaling afbeeldingen

Chris Smeele (1681546), Philippe Zwietering (1685431)

04-04-2017

Contents

1 Doel

We willen in dit onderzoek kijken wat het verschil is in snelheid van twee verschillende implementaties van bilineaire interpolatie om afbeeldingen te schalen. Hierbij willen we graag het verschil bekijken tussen onze eigen implementatie en de implementatie die al wordt gebruikt in de bestaande implementatie van dit project, die gebruik maakt van de OpenCV library. Het verschil in snelheid kan gemakkelijk gemeten worden door het verschil in tijd te meten tussen het begin en einde van de schalingsstap in beide implementaties.

1.1 Hypothese

Wij denken dat onze eigen implementatie sneller zal zijn omdat deze multi-threading toepast, waardoor er meerdere rijen tegelijkertijd kunnen worden geschaald.

2 Werkwijze

Onze code voor de bilineaire interpolatie en het schalen staan in appendix 7.1.

Voor het verschil in snelheid tussen de twee algoritmes is gekeken naar de gemiddelde tijdsduur voor een test afbeelding om te worden geschaald. We hebben de volgende testafbeelding bekeken, deze afbeelding geeft voor beide algoritmen een succesvolle gezichtsherkenning:

Om het tijdsverschil tussen het begin en einde van de schalingsstap te bepalen is gebruik gemaakt van de chrono namespace van de standaard library van c++ 2014. Aan het begin en einde van de schalingsstap wordt gekeken wat de tijd is met de `now()` methode op een `std::chrono::steady_clock` en het verschil hiertussen wordt uitgeprint op het scherm in milliseconden, met 6 cijfers achter de komma.

Figure 1: Onbwerkte testafbeelding

3 Resultaten

We hebben beide implementaties meerdere keren gedraaid op Chris zijn laptop. Hieruit krijgen we de volgende resultaten, tabel 1. De significantie van dit resultaat is platform afhankelijk en daarom zal de nauwkeurigheid per systeem afwijken. We gebruiken echter de `steady_clock`, en zoals te zien is in de c++-manual is deze specifiek bedoeld voor het meten van tijdsintervallen. We gaan er daarom vanuit dat de nauwkeurigheid minstens goed genoeg is om het verschil in microsecondes goed te kunnen onderscheiden. Dit betekent dat alleen de eerste drie cijfers achter de komma betekenis hebben en dat we de rest weg kunnen gooien. We controleren at compile time dat de resolutie van de tijdmeting nauwkeurig genoeg is, met behulp van een `static_assert`. Dan krijgen we:

Table 1: Tijdsduur verschillende implementaties

	OpenCV algoritme	Eigen algoritme
Nummer	Tijdsduur (ms)	Tijdsduur (ms)

4 Verwerking

5 Conclusie

6 Evaluatie

7 Appendix

7.1 Code bilineaire schaling

```
Hier komt code
```