Conversion of RGB images to Intensity images

Niels de Waal (1698041), Jasper Smienk(1700502)

April 15, 2018

Contents

1 Target

The assignment of TICT-V2VISN1-13 at the HU, which we have chosen to undertake consists of two different components. The first one consists of writing a image shell which has to ability to extract or modify data from a given image. The second assignment is to convert an RGB image to an intensity/grey-scale image.

We will be mostly focusing on the second assignment. This is because we are interested in comparing a few different approaches to this problem.

For the second assignment we will be doing two measuring tests. The first will cover speed across multiple methods of grey-scaling. The second test will revolve around effects on the cpu cache across the different methods of grey-scaling.

Both can be of interest. This could be because testing multiple methods of grey-scaling, including testing the speed at which this can be done could have effects when grey-scaling becomes an integral part of a system. This could happen in a real-time face recognition system, where grey-scaling has to be done every frame.

2 Methods

In this section we will discuss several methods of grey-scaling an image.

Let us refer to pixel i as part of image P, where $i \in \{0, \ldots, (P_{width} * P_{height}) - 1\}$. Every i has a set of (R, G, B) where R, G and B have a range $\{0, \ldots, 255\}$.

2.1 Methods for grey-scaling

In this sections a few methods for grey-scaling will be discussed. There are far too many methods to compare them all, thus in this article we will discuss these three:

- Averaging
- Luma
- Decomposition

2.1.1 Averaging

This will be considered as the naive approach to grey-scaling. Here we take the RGB values and using the average of them to calculate the grey-scale value. With the values: $\forall i \in P$ the grey-scale is $(i_R + i_G + i_B) \div 3$.

This technique has the advantage that it is very easy to implement and cheap to run. It has the disadvantage that it is not a very good at giving a good representation of a grey-scale image for human eyes.

2.1.2 Luma

The luma method fixes the problem that averaging has regarding the correctness for the human eye. Luma gives different priorities to different channels. The values for the different channels are described in [1]. With the values: $\forall i \in P$ the grey-scale is $i_R * 0.2126 + i_G * 0.7152 + i_B * 0.0722$.

With this method we could lose speed in comparison to averaging, because of the multiple floating point operations that have to be done each pixel. The upside is that this method delivers better images, because they are suited for the human eye.

2.1.3 Decomposition

Decomposition is the simplest method of grey-scaling. This method of grey-scaling has the advantage that it is very cheap to implement and run. The disadvantage is that, depending on the version used, it can result in a vastly lighter or darker image. With the values: $\forall i \in P$ the grey-scale is $\max(i_R, i_G, i_B)$ or $\min(i_R, i_G, i_B)$.

This method is not adjusted for the human eye.

2.2 Cache

Cache is an imported part of modern cpu architectures. Cpu cache can significantly increase the speed of piece of software. This is because it eliminates the time required of waiting for an value from memory. It does this by loading values from memory ahead of time and writing them to the on-die memory inside the cpu.

The algorithms and processes by which the cpu decides what values to load is far beyond the scope of this document.

3 Choice

In this section we will shortly discuss why we chose the afore mentioned methods for grey-scaling.

3.1 Grey-scaling

There are any number of different methods for grey-scaling. Far too many to be explored in this assignment. That is why we will examine the three methods mentioned above. We chose these methods of grey-scaling for the following reasons:

- Simple
- All methods operate on a per pixel basis
- Because of the per pixel operations, the methods have a high chance of being run in parallel
- Although the operations are similar they still differ enough that a difference should me measurable

4 Implementation

In this section we present the implementations of the different methods.

4.1 Grey-scaling

What follows are the different methods of grey-scaling, implemented in the framework required for this assignment.

The average method:

```
IntensityImage * StudentPreProcessing::stepToIntensityImage(const
    RGBImage &image) const {
       auto* intensityImage = new
           IntensityImageStudent(image.getWidth(), image.getHeight());

       for (int i = 0; i < (image.getWidth() * image.getHeight()); i++)
           {

              //Average
              intensityImage->setPixel(i, ((image.getPixel(i).r +
                  image.getPixel(i).g + image.getPixel(i).b) / 3));

       }
       return intensityImage;
}
```

The luma method:

```
IntensityImage * StudentPreProcessing::stepToIntensityImage(const
    RGBImage &image) const {
       auto* intensityImage = new
           IntensityImageStudent(image.getWidth(), image.getHeight());

       for (int i = 0; i < (image.getWidth() * image.getHeight()); i++)
           {

              //Luma
              intensityImage->setPixel(i, (image.getPixel(i).r * 0.2126
                  + image.getPixel(i).g * 0.7152 + image.getPixel(i).b
                  * 0.0722));


       }
       return intensityImage;
}
```

The decomposition method:

```
IntensityImage * StudentPreProcessing::stepToIntensityImage(const
    RGBImage &image) const {
       auto* intensityImage = new
           IntensityImageStudent(image.getWidth(), image.getHeight());

       for (int i = 0; i < (image.getWidth() * image.getHeight()); i++)
           {

              //Decomposition; max
              //intensityImage->setPixel(i,
                  std::max({image.getPixel(i).r, image.getPixel(i).g,
                  image.getPixel(i).b}));

              //Decomposition; min
              intensityImage->setPixel(i,
                  std::min({image.getPixel(i).r, image.getPixel(i).g,
                  image.getPixel(i).b}));


       }
       return intensityImage;
}
```

5 Evaluation

In this section we will discuss the 2 different aspects of the grey-scaling methods that will be measured.

5.1 Speed

We will be measuring and comparing the speed of the already discussed methods for grey-scaling. This could be of benefit to anyone wanting to run facial recognition in real-time. These kinds of systems need to process every frame at a relatively high frame-rate. This means that if some time could be saved on the processing of each frame, it would increase the time available to process the rest of a frame.

The speed will be measured by measuring the time required to run through the process of either just grey-scaling, or the full facial recognition process. Our hypothesis is available in our measurement report.

5.2 Cache behaviour

This one ties in with the speed measurements. Here we will attempt to find out which of the methods for grey-scaling have the most benefit from the cache. This will be measured by comparing the amount of cache hits and misses. This will give an idea about the efficiency of the methods with regards to cpu caches.

When one method seems to better synergy with the cache, those results could be tied to the results from the first measurement report. We won't be doing that in our measurement report as it will be outside the scope of the measurements presented in those reports.

References

[1] ITU-R, Recommendation ITU-R BT.709-6, http://www.itu.int/dms_pubrec/itu-r/rec/bt/R-REC-BT.709-6-201506-I!!PDF-E.pdf

[2] Texas Instruments, https://en.wikipedia.org/wiki/TI_Advanced_Scientific_Computer

[3] Intel, Pentium processor with MMX, https://www.intel.com/content/www/us/en/intelligent-systems/previous-generation/embedded-pentium-mmx.html

[4] khronos, OpenCL home page, https://www.khronos.org/opencl/