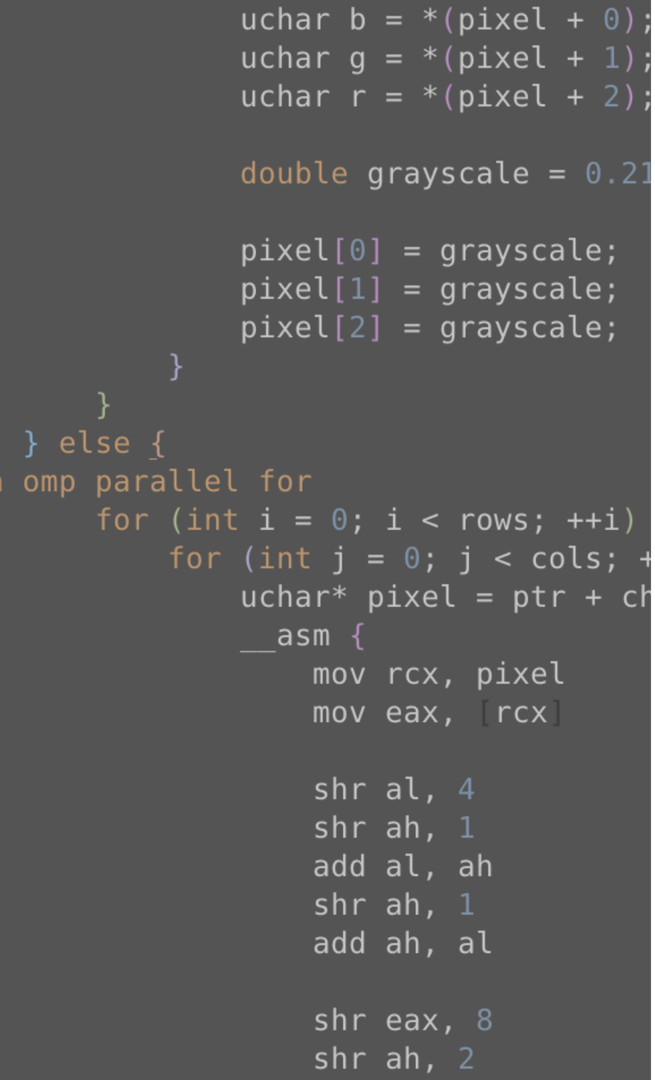


```
            uchar b = *(pixel + 0);
            uchar g = *(pixel + 1);
            uchar r = *(pixel + 2);

            double grayscale = 0.21

            pixel[0] = grayscale;
            pixel[1] = grayscale;
            pixel[2] = grayscale;
        }
    }
} else {
omp parallel for
    for (int i = 0; i < rows; ++i)
        for (int j = 0; j < cols; +
            uchar* pixel = ptr + ch
            __asm {
                mov rcx, pixel
                mov eax, [rcx]

                shr al, 4
                shr ah, 1
                add al, ah
                shr ah, 1
                add ah, al

                shr eax, 8
                shr ah, 2
```

Andrey Borisov

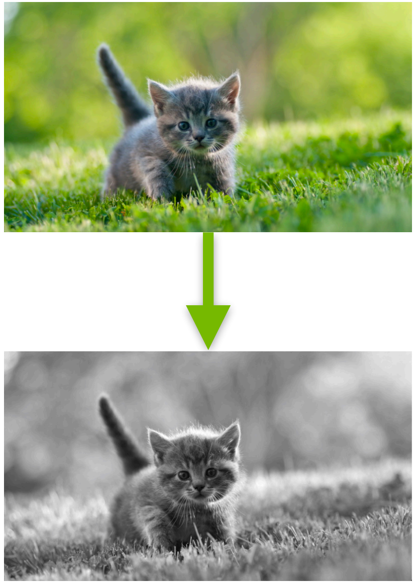
Aufgabe 4B

RGB zu Grayscale, Helligkeit und Histogramm

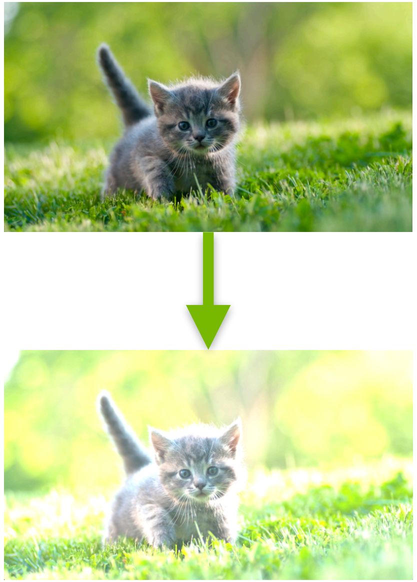
05.01.2021

Hochschule für Technik
und Wirtschaft Berlin
University of Applied Sciences

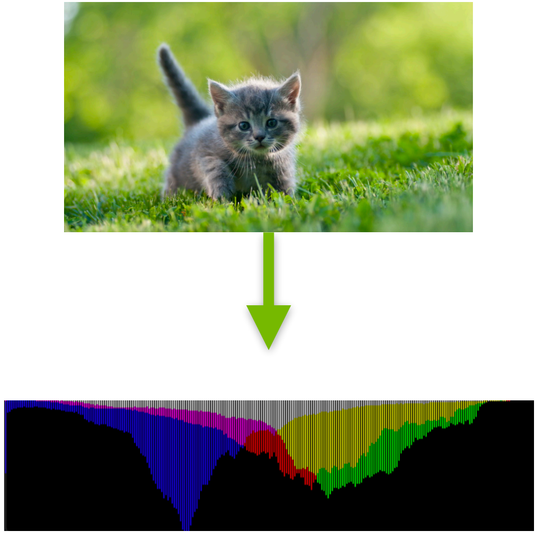
Aufgabenstellung:



RGB zu Grayscale



Helligkeit



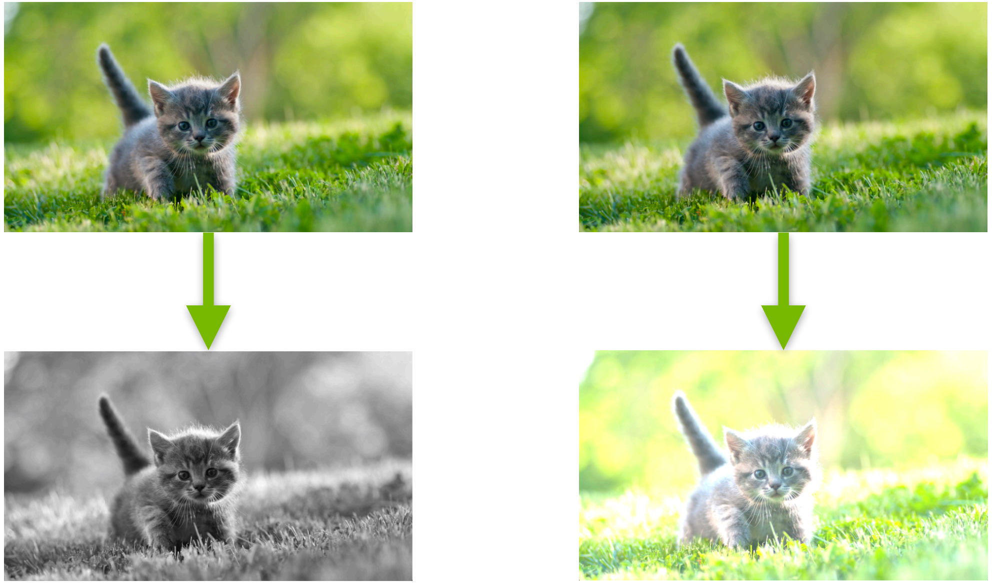
Histogramm

Werkzeuge:

- C++
- OpenMP
- OpenCV

RGB -> Grayscale und Helligkeit

Parallelisierung:

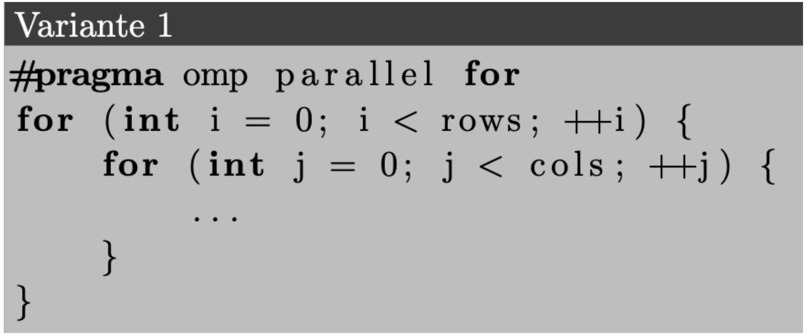


RGB zu Grayscale

Helligkeit

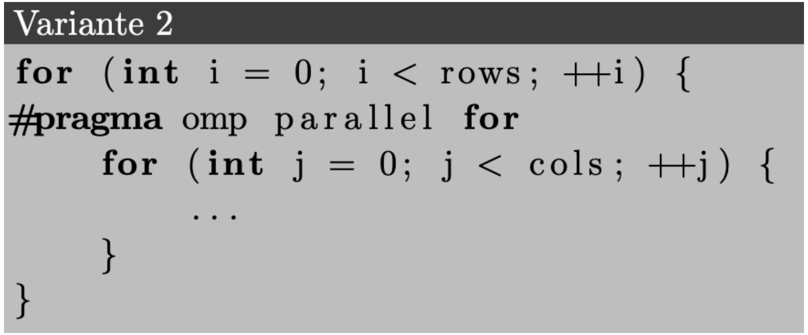
RGB -> Grayscale und Helligkeit

Parallelisierung:



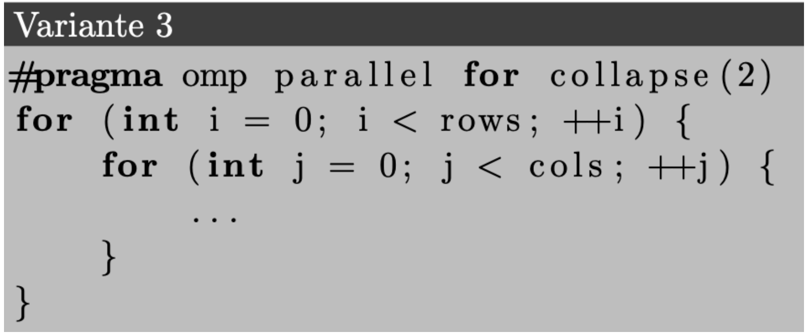
Variante 1

```
#pragma omp parallel for
for (int i = 0; i < rows; ++i) {
    for (int j = 0; j < cols; ++j) {
        ...
    }
}
```



Variante 2

```
for (int i = 0; i < rows; ++i) {
#pragma omp parallel for
    for (int j = 0; j < cols; ++j) {
        ...
    }
}
```

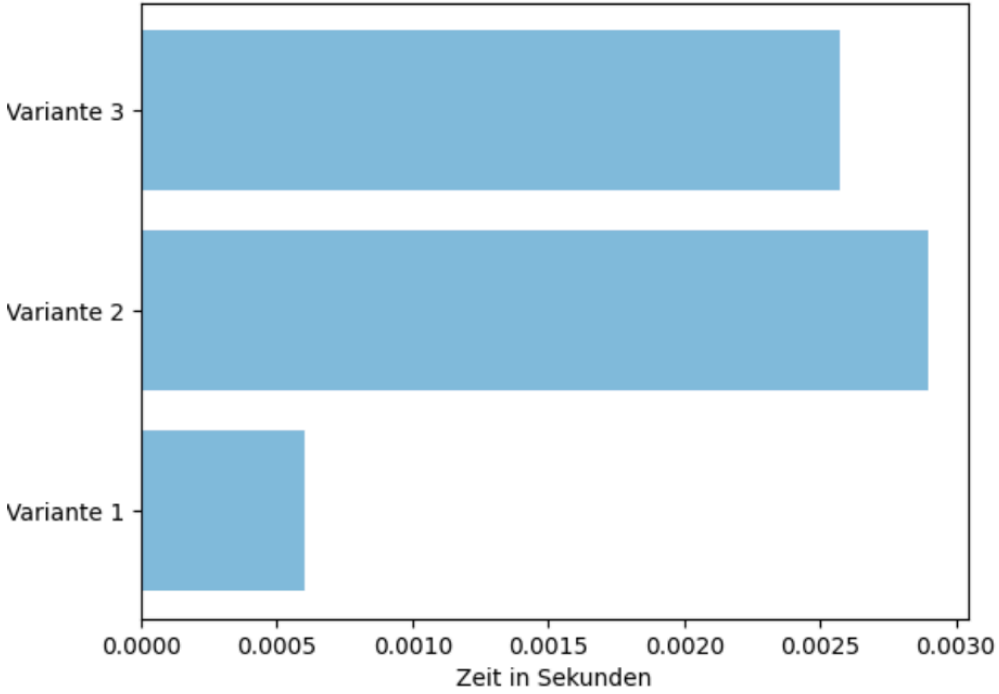


Variante 3

```
#pragma omp parallel for collapse(2)
for (int i = 0; i < rows; ++i) {
    for (int j = 0; j < cols; ++j) {
        ...
    }
}
```

RGB -> Grayscale und Helligkeit

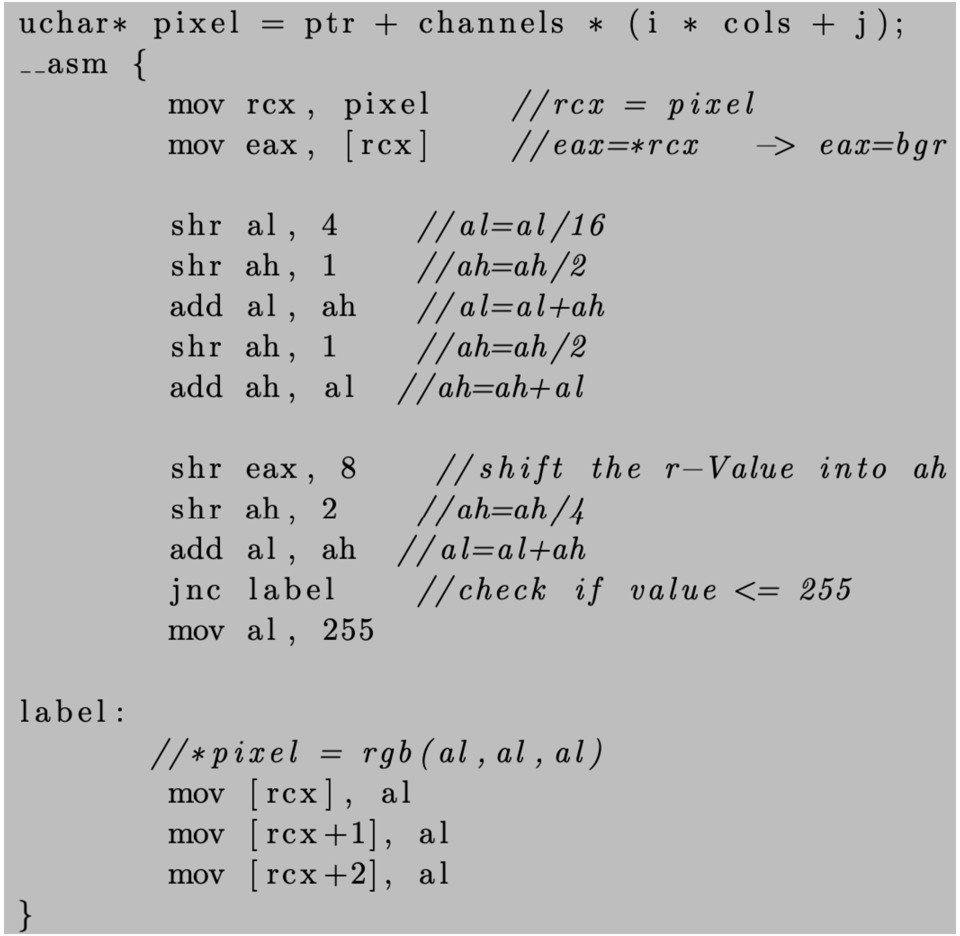
Parallelisierung:



RGB -> Grayscale

Implementierung:

$$gray = r/4 + (g/2 + g/4) + b/16$$ **<=>**

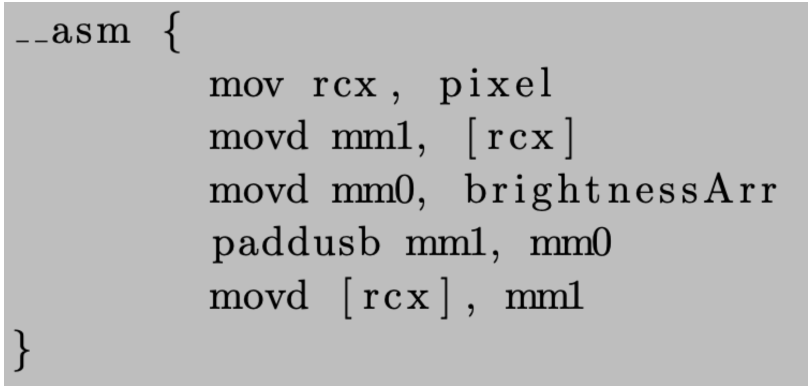


```
uchar* pixel = ptr + channels * (i * cols + j);
__asm {
        mov rcx, pixel    //rcx = pixel
        mov eax, [rcx]    //eax=*rcx   -> eax=bgr

        shr al, 4    //al=al/16
        shr ah, 1    //ah=ah/2
        add al, ah   //al=al+ah
        shr ah, 1    //ah=ah/2
        add ah, al  //ah=ah+al

        shr eax, 8    //shift the r-Value into ah
        shr ah, 2    //ah=ah/4
        add al, ah  //al=al+ah
        jnc label    //check if value <= 255
        mov al, 255

label:
       //*pixel = rgb(al,al,al)
        mov [rcx], al
        mov [rcx+1], al
        mov [rcx+2], al
}
```

Helligkeit

Implementierung:

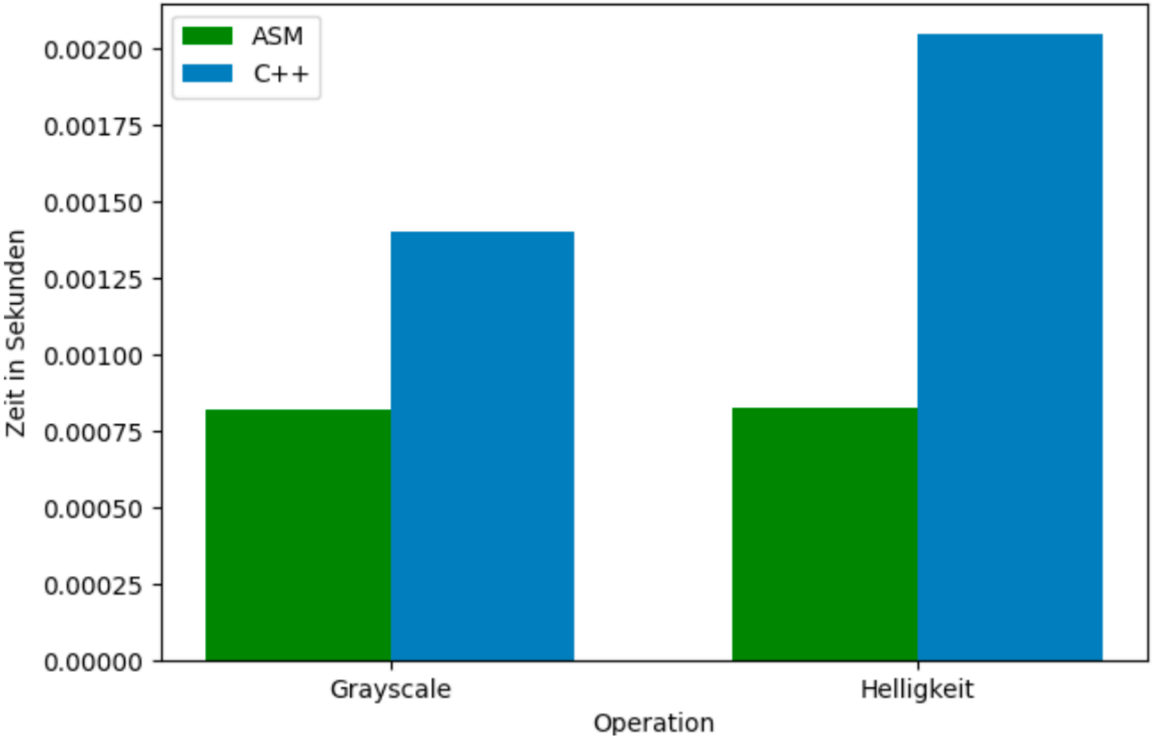
$pixel = rgb(r + \alpha, g + \alpha, b + \alpha)$ **<=>**



```
__asm {
          mov rcx, pixel
          movd mm1, [rcx]
          movd mm0, brightnessArr
          paddusb mm1, mm0
          movd [rcx], mm1
}
```

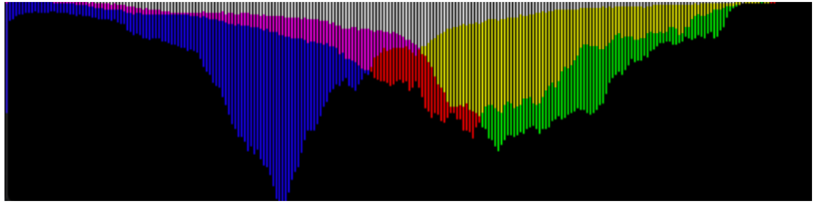
Assembler vs. C++

Implementierung:



Histogramm

Parallelisierung:



Histogramm

Histogramm

Parallelisierung:

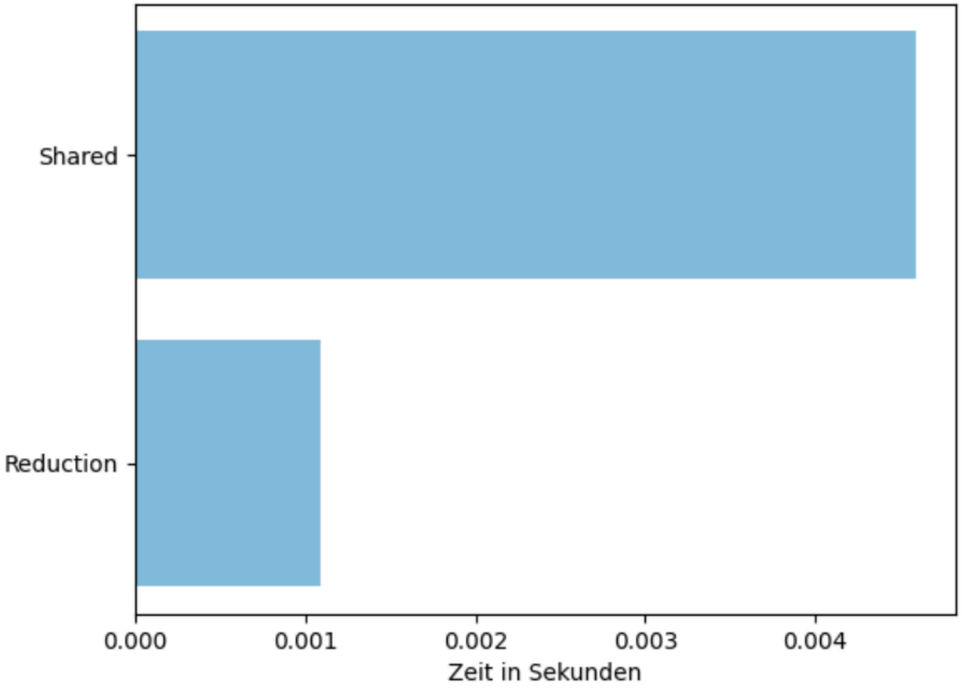
```
#pragma omp parallel for shared(%varName%)
```

Oder

```
#pragma omp parallel for reduction(%operation% : %varName%)
```

Histogramm

Parallelisierung:

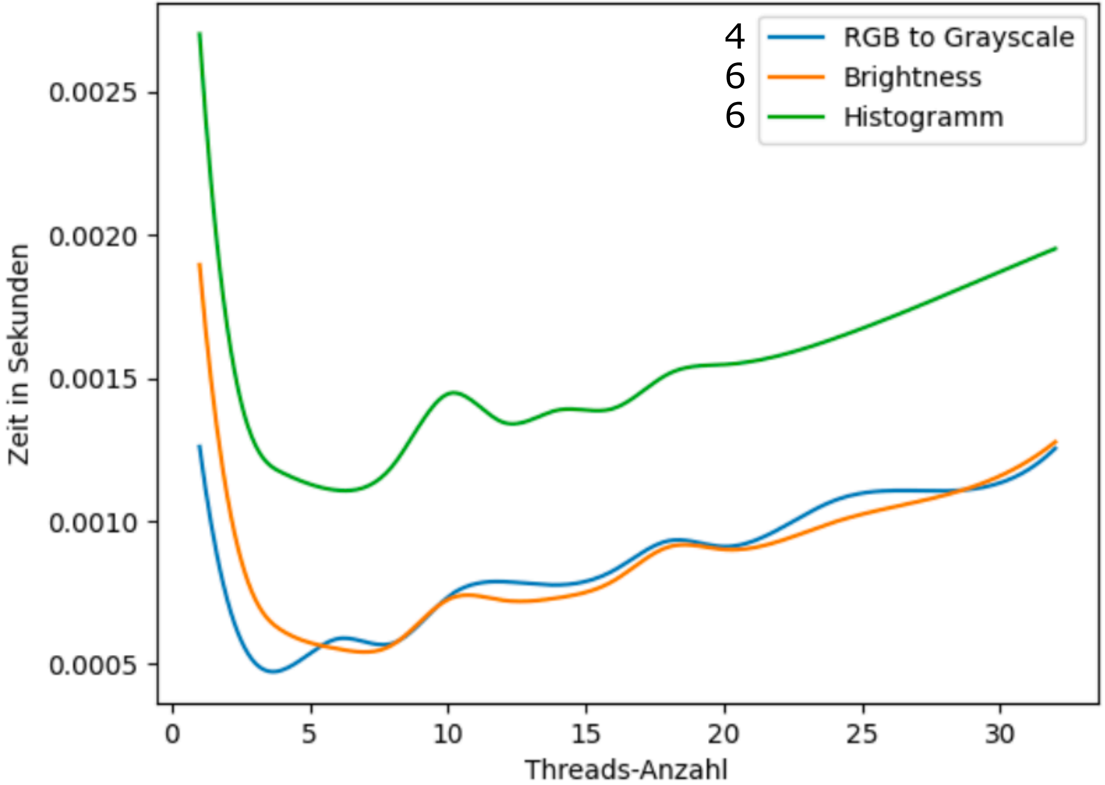


Threads-Anzahl:

Welche Anzahl an Threads ist optimal?

i5 Prozessor der 10. Generation (4 Cores, 8 Threads)

Threads-Anzahl:

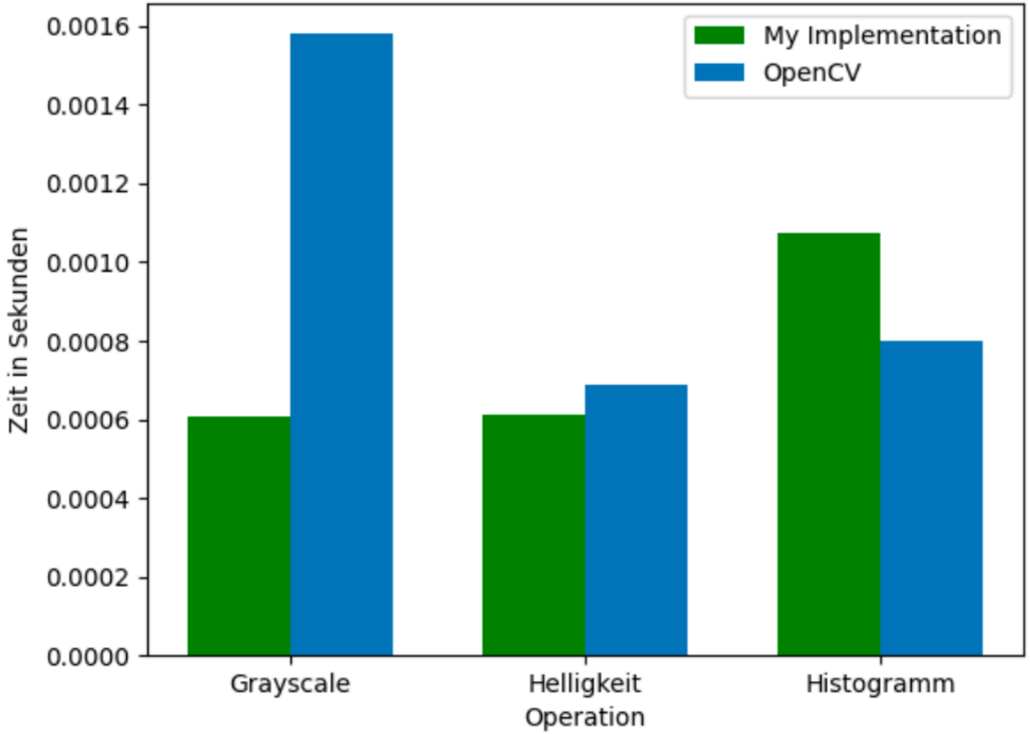


1-Thread vs. Optimale Threads-Anzahl

Threads-Anzahl:

RGB zu Grayscale:	162%
Helligkeit:	242%
Histogramm:	144%

Vergleich mit anderen Implementierungen:



Hochschule für Technik
und Wirtschaft Berlin

University of Applied Sciences

www.htw-berlin.de