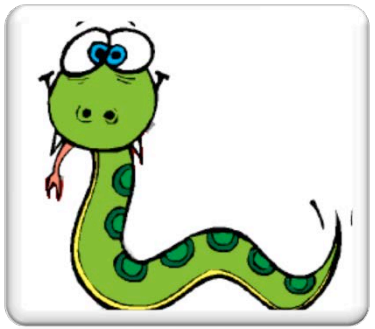
KOGS WINTER of CODE



FIRST YEAR 2014

Die Aufgabe

- Implementierung elementarer, häufig benötigter Bildverarbeitungsfunktionen (siehe folgende Folien)
- Zeitmessung der Laufzeit (**timeit**)
- Kompletter Code muss Python ausgeführt werden!
- C/C++, OpenCL, CUDA, Threading und andere Interpreter sind erlaubt!
- Python-Version 2.X-kompatibel

Das Ziel

- Vergleich der Möglichkeiten zur effizienten Bildverarbeitung in Python
- Erstellung von Empfehlungen aufgrund der Statistiken
- Untersuchung der Laufzeit, Arbeitsspeicher erst einmal irrelevant!
- Referenz: Numpy

Setting (1)

- Alle folgenden Bildverarbeitungsoperationen sollen auf folgenden Bilddaten ausgeführt werden:
  - Grauwertbilder (2D-Arrays)
  - Bildgrößen: 2048x2048 Pixel, 4096x4096 Pixel
  - Pixeldatentypen: UInt8, float(32)

→ Pro Test: 4 Variationen!

- Bei insgesamt 7 Operationen: 28 Tests pro Team
- Bei 5 Teams (+1 Referenz): 168 Ergebnisse

Setting (II)

- Falls externer (z.B. C/C++) Code verwendet wird, wird die Compilezeit des Wrappers (1x) mitgezählt!
- Aber: Alle Tests müssen in Python je 100x laufen!
- Relative Verringerung des Compile-Aufwands
- 3 Typen von Bildverarbeitungsverfahren:
  - Map/reduce
  - Filter
  - Transformationen

Map/reduce

- Algorithmus 1 (Summierung):

$Img\_dest = Img\_src1 + Img\_src2$

- Algorithmus 2 (Schwellenwert):

$Img\_dest = Img\_src > Scalar$

- Algorithmus 3 (Histogramm, Annahme: $-1 < I(x,y) < 256$)

$Hist(i) = card(\{(x,y) \mid I(x,y) = i\}) \text{ für } -1 < i < 256$

Filter (Faltung)

- Faltung (5x5)

```
For (int y=2; y!=height-2; ++y)
        For (int x=2; x!=width-2; ++x)
                For (int j=-2; j!=3; ++j)
                        For (int i=-2; i!=3; ++i)
                                Img_dest(x,y)
                                 = Img_src(x,y)*mask(i+2,j+2)
```

Gauß'sche Glättung ($\sigma = 1$, näherungsweise)

Mask(i,j):

$$\frac{1}{273}$$

1	4	7	4	1
4	16	26	16	4
7	26	41	26	7
4	16	26	16	4
1	4	7	4	1

Median-Filter

```
For (int y=2; y!=height-2; ++y)
        For (int x=2; x!=width-2; ++x)
                list = []
                For (int j=-2; j!=3; ++j)
                        For (int i=-2; i!=3; ++i)
                                list.append(Img_src(x+i, y+j))
                Img_dest(x,y) = median(list)
```

Subsampling (Binning)

- Verkleinern des Bildes (w,h) → (w/2,h/2)

```
For (int y=1; y<height; y+=2)
        For (int x=1; x<width; x+=2)
            Img_dest(x/2,y/2)  = (    Img_src(x-1,y-1)
                                    + Img_src(x-1,y)
                                    + Img_src(x,y-1)
                                    + Img_src(x,y))/4
```

Bildtransformationen (Rotation)

- Transformation der Bildkoordinaten, nicht der Bildintensitäten!
- Beispiel, für einen Winkel α :

Img_dest(r_x,r_y) = Img_src(x,y)

mit: $(r\_x,r\_y)^T = R_\alpha * (x,y)^T$

und: $$R_\alpha = \begin{pmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{pmatrix}.$$

- Bi-linear interpolieren!

Zusammenfassung

1. Bildaddition
2. Schwellenwertbildung
3. Histogrammberechnung
4. Gauß'sche Glättung (5x5)
5. Medianfilter (5x5)
6. Binning (2x2->1x1)
7. Rotation

Abgabe

- Abgabe bis zum 31.3.2014
- Preise für die schnellsten Implementationen
- Format: Archiv mit allen Dateien und Hinweise auf benötigte Python-Pakete
- Gut dokumentierter Code
- Definition der einzelnen Funktionen und mehrfacher Aufruf
- Zur Zeitmessung bitte **timeit** verwenden!

Viel Erfolg!

Times to beat [s] (2x2,66GHz Xeon Dual Core,5 GB RAM):

Quelltext verwendet numpy/scipy

Verfügbar auf der Homepage!

	small, uint8	small, float	large, uint8	large, float
Summierung	0,52	3,96	2,52	22,84
Schwellenwert	0,58	1,09	2,75	4,81
Histogramm	19,76	25,46	79,04	101,75
2D-Faltung	37,26	19,88	149,15	90,00
Median-Filter	213,82	210,31	856,19	851,17
Subsampling	1,61	4,95	5,90	19,52
Rotation	71,83	74,86	286,27	324,76