Belegaufgaben für den Kurs “Programmierkonzepte und Algorithmen“

Inhalt

Aufgaben

Sie haben zur Auswahl insgesamt 20 Aufgaben. Jedes Team soll nur eine Aufgabe auswählen.

Die Aufgabestellung finden Sie in Tabelle 1 (Bildverarbeitung) oder Tabelle 2 (Big Data).

1. Bildverarbeitung

Eingabe: eine JPG oder PNG Datei

Aufgabe: die Bilddatei einlesen, mit der entsprechenden Technologie parallel oder verteilt verarbeiten, die Ergebnisse anzeigen und speichern.

Technologien: *MPI*, *OpenMP*, *CUDA*, *OpenCL*

Programmiersprachen: C/C++/C#/Java/usw.

Allgemeine Hinweise:

1. Die Datei kann in die Anwendung als *RGB Matrix* geladen werden (zwei- oder dreidimensionalen Arrays mit Werten von 0 bis 255). Für *File Load, File Save* oder *Show Image* Operationen sind die Bibliotheken **pnglite**, **libpng**, **OpenCV** usw. zugelassen.
2. Diese Matrix muss in eine andere Matrix entsprechend der Aufgabe konvertiert werden. Die Konvertierung soll als Matrix-Verarbeitung entsprechend der Aufgabestellung (MPI, OpenMP, CUDA, OpenCL) parallelisiert erfolgen, ohne OpenCV oder ähnlichen Bibliotheken zu nutzen.

   Pixelzugriff (`Mat.at()`), Kanalverschmelzung (`cv::merge()`) und Kanaltrennung (`cv::split()`) mit OpenCV ist auch erlaubt.
3. Aus der resultierenden Matrix soll wieder eine PNG-, JPG- oder TIF-Datei erstellt werden.
4. Als Test-Dateien können sie die Bilder aus **diesem Datensatz** nutzen. Auch eigene Bilder sind erlaubt (aber nehmen Bilder mehreren Größen).
5. Vergleichen Sie Ihre Anwendung (die Ausführungszeiten für unterschiedliche Eingangsdateiengröße, Komplexität, Programmieraufwand) mit einer App ***ohne*** Parallelisierung sowie ggf. mit ***schon vorhandener*** Implementierung (z.B. die OpenCV).

Hinweis: OpenCV erlaubt direkter Vergleich zwischen zwei Bildern durch Subtraktion, z.B.:

```
cv::Mat diff_image = your_image - opencv_image;
```

Aufgaben:

(nach Schwierigkeit geordnet)

1. Farbraumkonvertierung (*color space conversion*):
    a. RGB → Grayscale
    b. RGB → YCbCr
    c. RGB → HSV

2. Helligkeit und Kontrast (Brightness and Contrast)
    a. Helligkeit
    b. Kontrast
    c. Normalisierung
    d. Histogrammberechnung

3. Gleitfensterverarbeitung (*sliding-window processing*):
    a. Emboss-Algorithmus
    b. Morphologische Dilatation (auf dem Grayscale Bild!)
    c. Box-Weichzeichner (Simple Blur oder nach Wunsch – Gaussian Blur)

Aufgabenauswahl:

Aufgabe	**Beschreibung**	**OpenMP**	**MPI**	**CUDA**	**OpenCL**
Aufgabe 1	RGB → Grayscale, Helligkeit+Kontrast		✓		✓
Aufgabe 2	Normalisierung, Histogrammberechnung	✓	✓		
Aufgabe 3	RGB → HSV, Box-Weichzeichner			✓	
Aufgabe 4	RGB → YCbCr, Morphologische Dilatation				✓
Aufgabe 5	Normalisierung, Emboss-Filter		✓	✓	
Aufgabe 6	RGB → HSV, Histogrammberechnung	✓	✓		
Aufgabe 7	RGB → Grayscale, Morphologische Dilatation			✓	
Aufgabe 8	RGB → Grayscale, Box-Weichzeichner				✓
Aufgabe 9	RGB → YCbCr, Helligkeit+Kontrast		✓	✓	
Aufgabe 10	RGB → Grayscale, Emboss-Filter				✓

Tabelle 1. Aufgaben für parallelisierte Bildverarbeitung

Anweisung:

Sie wählen eine Aufgabe in der Tabelle aus, gucken die benötigte Technologie in der Kopfzeile, und die Algorithmen in die Tabellenzelle.

> ### *Hinweis:*
>
> *In der Belegaufgabe, das **wichtigste** Teil ist die **Parallelisierung/Verteilung**, und nicht das Bildverarbeitungsalgorithmus. Sie müssen nicht zu viel Zeit und Mühe verschwenden, um ein besserer Algorithmus zu finden oder implementieren. Verwenden lieber mehr Zeit auf die Parallelisierung und/oder Verteilung.*

2. Verarbeitung von Big Data

Eingabe: ein Korpus von Texten

Aufgabe: Texten einlesen, verteilt mit Hadoop oder Spark verarbeiten, Ergebnisse speichern.

Technologien: Hadoop oder Spark

Programmiersprachen: Java/Python/Scala/R/usw.

Allgemeine Hinweise:

1. Die Dateien müssen als Text geladen werden (getrennt oder zusammen).
2. Hadoop oder Spark (entsprechend der Aufgabe) müssen für die Analyse benutzt werden
3. Als Test-Dateien benutzen Sie **dieses Datensatz**.
4. Fur alle Aufgaben (außer №5) sollen die Stoppwörter (z.B. "*a*", "*an*", "*the*", "*of*", "*und*", "*mit*", "*la*", "*u*", "*же*" usw.) ausfiltern. Nutzen Sie z.B. eine von diesen Kollektionen:
    a. https://github.com/6/stopwords-json
    b. https://github.com/Alir3z4/stop-words
    c. https://github.com/stopwords-iso
5. ***Empfehlung***: um besser die Vorteile von Hadoop/Spark zu sehen, sollen Sie alle Texte mehrmals kopieren, bis die Datensatzgröße hunderte von Megabytes oder Gigabytes erreicht.

Aufgabenauswahl:

	Technologien	
Aufgaben	**Hadoop**	**Spark**
Type-Token-Ratio	**Aufgabe 11**	**Aufgabe 12**
Sprachvergleichung	**Aufgabe 13**	**Aufgabe 14**
Bigram-Analyse	**Aufgabe 15**	**Aufgabe 16**
Sprachstil-Fingerprint	**Aufgabe 17**	**Aufgabe 18**
Text-Clustering	—	**Aufgabe 19**
Ähnliche Wörter	—	**Aufgabe 20**

Tabelle 2. Aufgaben für Big Data

Aufgaben:

1. **Sprachliche Vielfalt messen (Type-Token-Ratio pro Sprache)**
   a. Zähle pro Sprache die Gesamtzahl und die Anzahl einzigartiger Wörter
   b. Berechne die Type-Token-Ratio.
   c. Gib die Werte pro Sprache aus und vergleiche sie.

2. **Sprachvergleichung**:
   a. Ermitteln Sie für jede Sprache die Verteilung der Wortlängen und vergleichen Sie die durchschnittliche bzw. mediane Wortlänge; geben Sie außerdem das längste Wort jeder Sprache als Tupel „Sprache – Wort – Länge“ aus.

3. **Häufigste Wortpaare pro Sprache (Bigram-Analyse)**
   a. Erzeuge für jede Sprache alle aufeinanderfolgenden Wortpaare.
   b. Zähle deren Häufigkeit.
   c. Gib die Top-20 Wortpaare pro Sprache aus.

4. **Sprachstil-Fingerprint (Stoppwort-Anteil pro Sprache)**
   a. Bestimme pro Sprache den Anteil der Stoppwörter an allen Wörtern.
   b. Gib den Stoppwort-Anteil je Sprache aus und vergleiche die Ergebnisse.

5. **Text-Clustering**:
   a. Alle Texte einer gewählten Sprache in Vektoren (z. B. mit Word2Vec oder Sentence Embeddings) umwandeln, und führen anschließend ein Clustering (z. B. K-Means) durch, um thematisch ähnliche Bücher zu gruppieren und die charakteristischen Wörter oder Titel jedes Clusters zu identifizieren.

6. **Ähnliche Worter**:
   a. Viel Sprachen haben ähnliche Wörter (als Lehnwörter oder „falsche Freunde“).
   b. Finden Sie alle Wörter (außer den Stoppwörter wie z.B. „*the*“, „*mit*“, „*la*“, „*же*“, usw.) in gegebenen Texten, die auch in Texten aus mehreren anderen Sprachen vorkommen.
   c. *Hinweis:* Da die inhaltliche Bedeutungsanalyse der Wörter sehr aufwendig und nicht Teil dieses Kurses ist, genügt es, Wörter mit exakt gleicher Schreibweise zu zählen (z.B. "hand" [En] und "Hand" [De]).

Anforderungen

Für das Bestehen der Belegaufgaben müssen die folgenden drei Anforderungen erfüllt:

1. Eine öffentliche Präsentation mit Folien (5+ Seiten) für 5-10 Minuten halten
2. PowerPoint Folien vorlegen

> *In der Präsentation **weniger Zeit** auf die Erklärung von Algorithmen (Blur, Farbräume, usw) verwenden, und **mehr Zeit** auf die Technologie, und Parallelisierung bzw. Verteilung.*

3. Dokumentation (5-10 Seiten) als PDF vorlegen
4. Quellcode (ZIP, TAR oder GitHub, GitLab, BitBucket, usw.) vorlegen

Die Dokumentation muss am mindestens die Folgende enthalten:

1. Kurze Beschreibung und Erklärung der Aufgabe
2. Detaillierte Lösungsbeschreibung
3. Code-Fragmente mit einer Textbeschreibung
4. Screenshots für die Ergebnisse und/oder Zwischenergebnisse
5. Ausführliche Teste der Anwendung
6. Graphen und Diagrammen für die Leistung und vergleichende Laufzeit
7. Kurzes Fazit

Hinweise

Algorithmus für die Konvertierung von RGB in HSV:

Es gibt andere Algorithmen, aber ein von den existierenden:

1. Teilen die `R`, `G` und `B` Werte durch 255 (Normalisierung)

```
R = R / 255.0, G = G / 255.0, B = B / 255.0
```

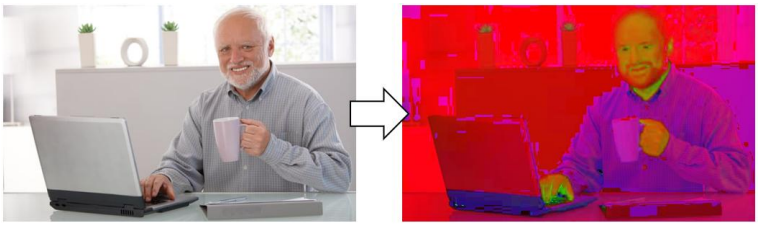
2. Berechnen `cmax` und `cmin` und ihre Differenz

```
cmax = max(R, G, B)
cmin = min(R, G, B)
diff = cmax - cmin
```

3. Berechnen Hue (Farbwert):
   a. Wenn `cmax` und `cmin` sind gleich, ist `H = 0`
   b. Wenn `cmax` gleich `R`, dann `H = (60 * ((G - B) / diff) + 360) % 180`
   c. Wenn `cmax` gleich `G`, dann `H = (60 * ((B - R) / diff) + 120) % 180`
   d. Wenn `cmax` gleich `B`, dann `H = (60 * ((R - G) / diff) + 240) % 180`

4. Berechnen Saturation (Farbsättigung):
   a. Wenn `cmax` gleich 0, dann `S = 0`
   b. Andernfalls, `S = (diff / cmax) * 255`

5. Berechnung von Value (Hellwert): `V = cmax * 100`



RGB HSV

Umrechnung zischen RGB und Grayscale:

$$\text{Grau: } G = 0.21 \cdot R + 0.72 \cdot G + 0.07 \cdot B$$

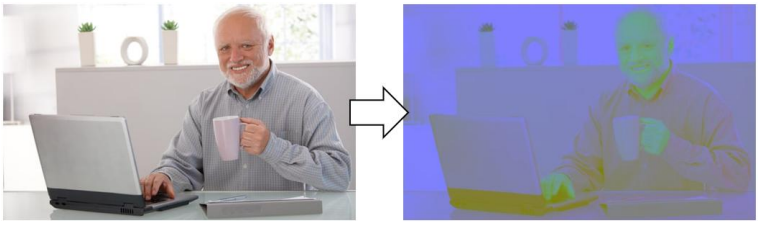
RGB Grau

Umrechnung zwischen RGB und YCbCr:

$$Y = 16 + (65.481 \cdot R + 128.553 \cdot G + 24.966 \cdot B)$$

$$C_B = 128 + (-37.797 \cdot R - 74.203 \cdot G + 112.0 \cdot B)$$

$$C_R = 128 + (112.0 \cdot R - 93.786 \cdot G - 18.214 \cdot B)$$



RGB YCbCr

Farbnormalisierung:

So normalisiert man die Farbwerte in einem Bild:

1. Maximaler Wert in dem ganzen Bild finden
2. Alle Werte in dem Bild durch maximalen Wert teilen (alle Werte sind jetzt in Bereich $[0;\ 1]$)
3. Alle Werte in Bereich $[0;\ 1]$ wieder mit 255 multiplizieren (oder 179 für H in HSV)

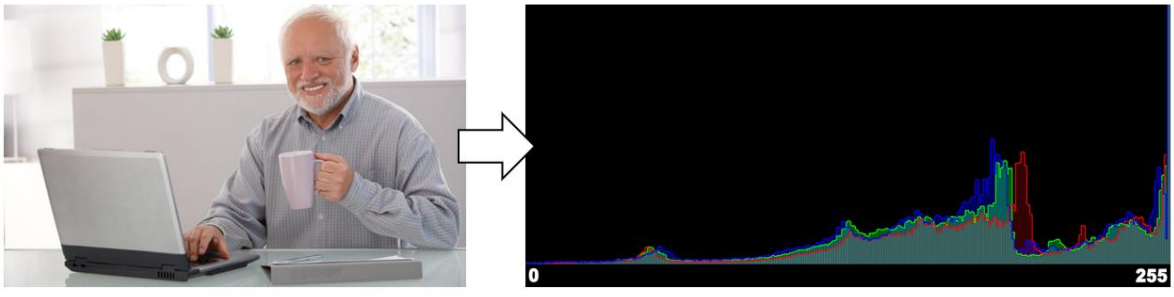
Histrogrammberechnung

So baut man ein Histogramm aus einem Bild:

1. Ein Array H (Histogramm) von Null-Elementen erzeugen, mit der Länge *N,* wobei *N* ist der maximaler Wert in dem Bild.
2. Für jedes Element i des Bildes Img, inkrementieren das Element in dem Array H mit dem Index $H[Img[i]]$

Pseudokode:

$$N = \max(Img)$$
$$H = [0] \times N$$
$$for\ i,j\ \text{in}\ Img\text{:}$$
$$\quad H\big[Img[i,j]\big] \mathrel{+}= 1$$



RGB Histogramm

(Rechts ist nur eine graphische Darstellung einer Histogramm und ist nicht pflichtig)

Helligkeit und Kontrast

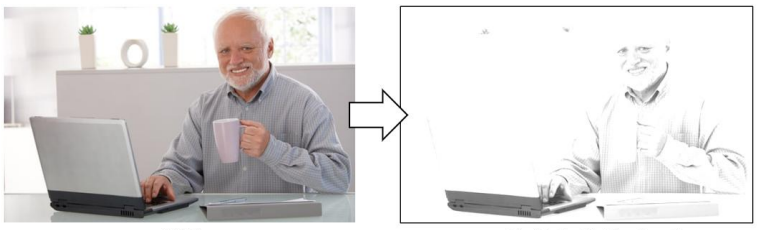
Die Helligkeit und Kontrast eines Bildes kann man mit der folgenden Formel anpassen:

$$Img'[i,j] = \alpha \cdot Img[i,j] + \beta$$

Wobei,

- i und j zeigen an, daß sich das Pixel in der i-ten Zeile und j-ten Spalte befindet.
- Die Parameter $\alpha > 0$ und β werden oft als Verstärkungs- und Vorspannungsparameter bezeichnet. Manchmal wird gesagt, dass diese Parameter den *Kontrast* bzw. die *Helligkeit* steuern.

Also, für *Kontrast*, multiplizieren allen Elementen des Bildes mit einem Wert α, und für Helligkeit, summieren allen Elementen mit einem Wert β.



RGB Helligkeit+Kontrast

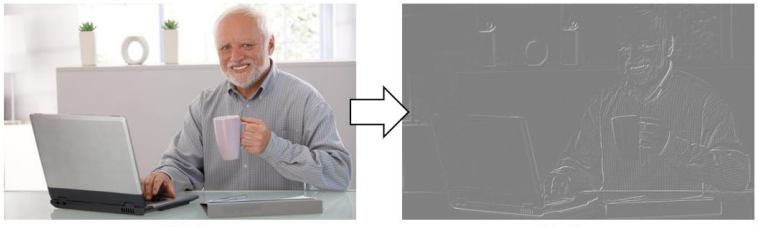
Emboss

Für jeden Pixel finden die Different zwischen seinen roten, blauen und grünen Komponenten und vergleichen es mit seinem oben-linken Nachbarn. Aus den drei Differenzen finden die, deren absoluten Wert der gröste ist. Es soll ein Integer `diff` in Bereich [-255; +255] sein. Berechnen ein grauen Wert aus dieser Differenz wie folgend:

```
gray = 128 + diff
if gray > 255:
   gray = 255
if gray < 0:
   gray = 0
```

Machen jetzt ein neues RGB-Pixel aus diesem grauen Wert.

```
pixel = [gray, gray, gray]
```



RGB Emboss

Hinweis: "emboss" heißt "**prägen**" auf Englisch, also das Ergebnis sieht ähnlich aus.

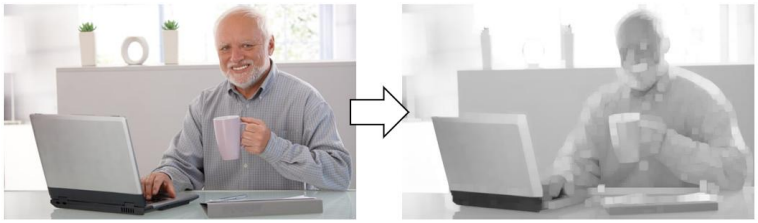
Gaußcher Weichzeichner

Mehr können Sie auf Wikipedia lesen.

RGB Blur

Dilatation

Die morphologische Dilatation machen Sie auf dem grayscale-Bild. Mehr können Sie hier lesen.



RGB Dilatation