Übung zur Vorlesung Video-Inhaltsanalyse

Blatt 1 – Schnitterkennung

Allgemeine Hinweise

Verwenden Sie zur Programmierung der Aufgaben 1 und 2 JAVA oder C++ (die angegebenen Funktionsdeklarationen sind jeweils in C++ angegeben).

JAVA bietet deutlich mehr Funktionen zum Laden, Verwalten und Speichern von Bildern. Verwenden Sie die in JAVA verfügbaren Routinen, um die JPEG-Bilder zu laden.

Für **C++** ist die Klasse ***Image*** als Sourcecode verfügbar, um Bilder im PPM-Format einzulesen und auf die Bilder zuzugreifen. Folgende Funktionen stehen in ***Image*** zur Verfügung:

`Image (int width, int height, int band)`	Erzeugung eines Bildes anhand der Breite, Höhe und Anzahl von Farbkanälen
`Image (string filename)`	Laden eines Bildes im PPM-Format aus einer Datei
`resize (int width, int height, int band)`	Ändern der Größe eines Bildes
`clear (unsigned char value)`	Bildinhalt mit value überschreiben
`read (string filename)`	Einlesen eines Bildes
`write (string filename)`	Speichern eines Bildes

Aufgabe 1 – Klasse ConvertColorspace

Entwerfen Sie eine Klasse *ConvertColorspace*, die ein RGB-Bild in ein YUV-Bild umwandelt bzw. umgekehrt.

- Lesen Sie die Beschreibung des YUV-Formats (YUV-Format.pdf).
- Welche maximalen und minimalen Werte können bei der Umwandlung eines RGB-Bildes in den YUV-Farbraum auftreten?
- Implementieren Sie eine Klasse *ConvertColorspace* sowie folgende Methoden der Klasse*:*
  - `void RGB2YUV (const Image &src, Image &dest);`
  - `void YUV2RGB (const Image &src, Image &dest);`

  Addieren Sie den Wert *127.5* auf die U und V Komponente, um negative Werte zu vermeiden. Subtrahieren Sie vor der Umwandlung von YUV nach RGB diesen Wert zunächst.
- Testen Sie mit Hilfe der Bilder ***test1***, ***test2*** und ***test3***, ob die Umwandlung zwischen dem RGB- und YUV–Format verlustbehaftet ist. Wandeln Sie dazu ein Bild in das YUV-Format um und wieder zurück. Vergleichen Sie das Bild mit dem ursprünglichen Bild. Wie hoch ist die höchste Pixelabweichung zwischen beiden Bildern?
- Erzeugen Sie ein Bild der Größe 256x256 und setzten Sie alle Pixel des ersten Farbkanals auf 128. Setzen Sie die Pixel im zweiten Farbkanal entsprechend der jeweiligen Spaltennummer und im dritten Farbkanal auf die jeweilige Zeilennummer. Wandeln Sie das Bild mittels *YUV2RGB* um und speichern Sie das YUV-Bild direkt ab. Was wird dargestellt?

Aufgabe 2 – Klasse ImageDistance

Entwerfen Sie die Klasse ***ImageDistance***, um einen Wert zur Beschreibung des Unterschiedes zweier Bilder zu berechnen. Implementieren Sie anschließend die folgenden 3 Funktionen:

```
// sum of absolute difference
void SAD (Image &img1, Image &img2, double &diff);

// histogram difference
void HD (Image &img1, Image &img2, double &diff);

// standard deviation
void SD (Image &img1, Image &img2, double &diff);
```

- Die Histogrammdaten sollen innerhalb der Funktion *HD* in einem Feld oder Vektor gespeichert werden. Welche Dimension hat der Vektor?
- Laden Sie die beiden Graustufenbilder ***test4a*** und ***test4b*** und berechnen Sie die Differenzen für alle drei Distanzmaße. Was fällt Ihnen auf?
  Hinweis: Da es sich um Graustufenbilder handelt, brauchen Sie nur den ersten Farbkanal berücksichtigen.
- Welche Arten von Schnitten können mit der jeweiligen Funktion erkannt werden?

Aufgabe 3 – Beurteilung der Qualität der Schnitterkennung

- Was bedeuten die Qualitätsmaße *Präzision* und *Vollständigkeit*? Warum wird neben der Präzision und Vollständigkeit noch ein *F1*-Wert eingeführt?
- Geben Sie ein einfaches Beispiel an, bei dem trotz eines sehr hohen Wertes für die Präzision der F1-Wert sehr gering ist. Ist es möglich, ein entsprechendes Beispiel für die Vollständigkeit zu finden?

Aufgabe 4 – Kantenerkennung

- Geben Sie die Komplexität folgender Verfahren zur Berechnung der Differenz zweier Bilder an:
  - Summe der absoluten Pixeldifferenzen
  - Histogramme
  - Standardabweichung
  - Kantenbasierter Kontrast
  - ECR