Übung zur Vorlesung: Image and Video Processing

Blatt 1 – Image Similarity

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 Bitmap-Bilder (BMP) 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.
- 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 – Komplexität

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

Documentation: YUV

Quelle: Wikipedia, the free encyclopedia (en.wikipedia.org).

YUV is the color space used in the PAL system of television broadcasting which is the standard in most of Europe and some other places. Y stands for the luminance component (the brightness) and U and V are the chrominance (color) components. The YCbCr or YPbPr color space, used in computer component video, is derived from it (Cb/Pb and Cr/Pr are simply scaled versions of U and V), and is sometimes inaccurately called "YUV".

YUV signals are created from an original RGB (red, green and blue) source. The weighted values of R, G and B are added together to produce a single Y signal, representing the overall brightness, or luminance, of that spot. The U signal is then created by subtracting the Y from the blue signal of the original RGB, and then scaling; and V by subtracting the Y from the red, and then scaling by a different factor. This can be accomplished easily with analog circuitry.

The following equations can be used to derive *Y*, *U* and *V* from *R*, *G* and *B*:

$$Y = 0.299R + 0.587G + 0.114B$$

$$U = 0.492(B - Y) = -0.147R - 0.289G + 0.436B$$

$$V = 0.877(R - Y) = 0.615R - 0.515G - 0.100B$$

Here, *R*, *G* and *B* are assumed to range from 0 to 1, with 0 representing the minimum intensity and 1 the maximum.

- If $[R \;\; G \;\; B]^T = [1 \;\; 1 \;\; 1]$ then $[Y \;\; U \;\; V]^T = [1 \;\; 0 \;\; 0]$. In other words, the top row coefficients sum to unity and the last two rows sum to zero.

Luminance/chrominance systems in general

The primary advantage of luminance/chrominance systems such as YUV is that they remain compatible with black and white analog television. The Y signal is essentially the same signal that would be broadcast from a normal black and white camera (with some subtle changes), and the U and V signals can simply be ignored. When used in a color setting the subtraction process is reversed, resulting in the original RGB color space.

Another advantage is that the signal in YUV can be easily manipulated to deliberately discard some information in order to reduce bandwidth. The human eye actually has fairly low color resolution: the high-resolution color images we see are processed by the visual system by combining the high-resolution black and white image with the low-resolution color image. Using this information to their advantage, standards such as NTSC reduce the amount of signal in the chrominance considerably, leaving the eye to recombine them. For instance, NTSC saves only 11% of the original blue and 30% of the original red, throwing out the rest. Since the green is already encoded in the Y signal, the resulting U and V signals are substantially smaller than they would otherwise be if the original RGB or YUV signals were sent. This filtering out of the blue and red signals is trivial to accomplish once the signal is in YUV format.

However this process, obviously, reduces the quality of the image. In the 1950s when NTSC was being created this was not a real concern because common equipment could not display images any better than the quality of the signal they were already receiving. But today a modern television can display more information than is contained in these lossy signals. This has led to a number of attempts to record images with as much of the YUV signal as possible, including S-Video on VCRs. YUV was also used as the standard format for common video compression algorithms such as MPEG-2, which is used in digital television and for DVDs. The professional CCIR 601 uncompressed digital video format also uses the YUV color space, for compatibility with previous analog video formats, which can then be easily mixed into any output format needed.

YUV is a versatile format which can be easily combined into other legacy video formats. For instance if you amplitude-modulate the U and V signals onto quadrature phases of a subcarrier you end up with a single signal called **C**, for *chroma*, which can then make the YC signal that is S-Video. If you mix the Y and C signals, you end up with composite video, which almost any television can handle. All of this modulating can be accomplished easily in low-cost circuitry, while the *demodulation* is often very difficult indeed. Leaving the signal in the original YUV format thus made DVDs very simple to construct, as they could easily downmix to support either S-video or composite and thus guarantee compability with simple circuits, while still retaining all of the original information from the source RGB signal.