Dokumentation: Abschlussprojekt in Visual Computing 1

Lilian Hung

2. August 2017

Inhalt

Liste der Beispiele

1 Farbhistogramme

Bereits im Verlauf der Veranstaltung wurde ein Histogramm für den RGB-Farbraum implementiert. Die Ergebnisse (mAP-Werte) für dieses Feature waren verglichen mit der Durchschnittsfarbe (Mean-Color) und der später implementierten Farbsignatur relativ gut. Eine Idee für das abschließende Projekt war es, ein Histogramm im YCoCg-Farbraum als Feature zu verwenden.

1.1 Der YCoCg-Farbraum

Der Farbraum kann durch eine Transformation des RGB-Farbraums beschrieben werden. Die Farbaufteilung besteht aus einem Helligkeitswert, Y oder lum, und zwei Chrom-Werten, Co und Cg. Dabei steht Co für "chromiance orange" und Cg für "chromiance green". Die Transformation ist folgendermaßen definiert:

- $Y = 1/4 \cdot R + 1/2 \cdot G + 1/4 \cdot B$
- $Cg = (-1/4) \cdot R + 1/2 \cdot G + (-1/4) \cdot B$
- $Co = 1/2 \cdot R + (-1/2) \cdot B$

Die Werte von Y befinden sich im Bereich zwischen 0 und 1, während die Werte von Co und Cg sich im Bereich von $-0,5$ und $0,5$ befinden. [1]

Die Aufteilung der Farben für das YCoCg-Histogramm wurde des weiteren, nicht wie das RGB-Histogramm in einem simplen Würfel aufgeteilt, sondern die Luminanz-Werte wurden auf der Y-Achse durch eine einstellbare Zahl geteilt, während die Co und Cg-Werte in einem Kreis liegend betrachtet wurden und darin über Winkel und einen Radius geteilt (Siehe Abbildung 1). Die Idee dieser Aufteilung war, dass sich die Farben so "sinnvoller" auf das Histogramm verteilen ließen und somit der Vergleich der Histogramme bessere Ergebnisse liefert.

Trotz verschiedener Experimente mit unterschiedlichen Anzahlen von Bins auf den drei verschiedenen Dimensionen, war der mAP-Wert, der zwischen $0,4170$ und $0,4237$ (siehe Beispiel 1) für die Bilder im "Test_10×5"-Ordner, schlechter als der Wert des RGB-Histogramms.

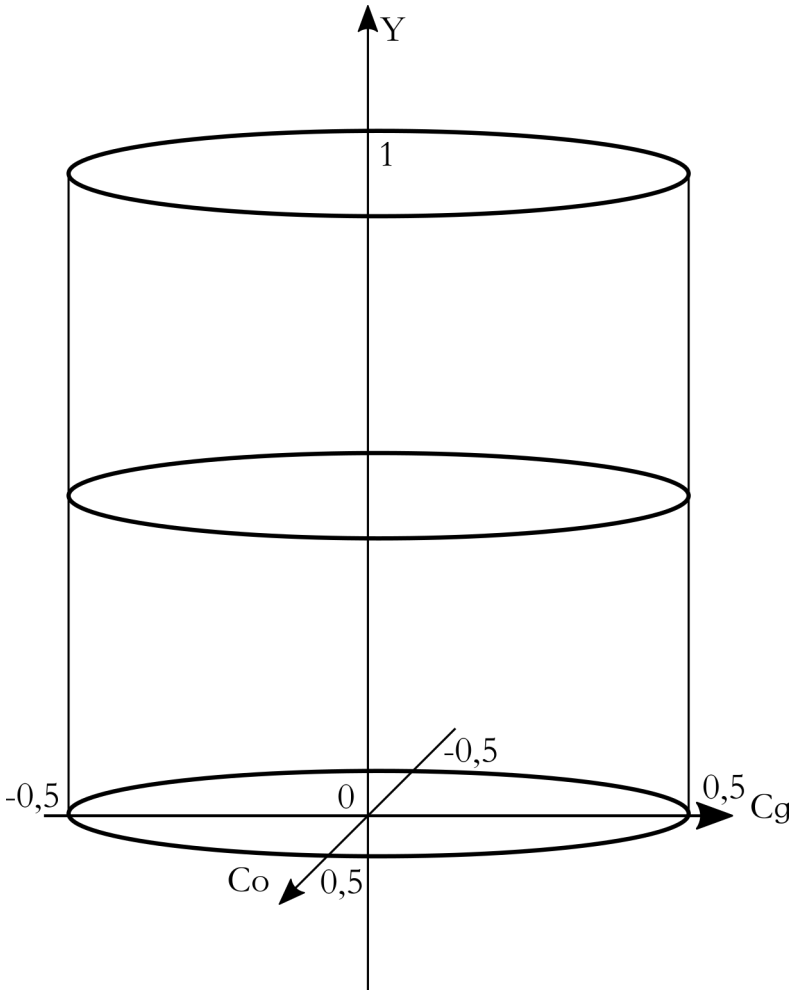

Abbildung 1: YCoCg-Farbmodell

```
bins: 1
MAP: 0.4217 took 12ms for feature factory ColorHistogramYCoCg
bins: 2
MAP: 0.4237 took 16ms for feature factory ColorHistogramYCoCg
bins: 3
MAP: 0.4170 took 17ms for feature factory ColorHistogramYCoCg
bins: 4
MAP: 0.4170 took 62ms for feature factory ColorHistogramYCoCg
bins: 5
MAP: 0.4170 took 20ms for feature factory ColorHistogramYCoCg
bins: 6
MAP: 0.4170 took 21ms for feature factory ColorHistogramYCoCg
bins: 7
MAP: 0.4170 took 30ms for feature factory ColorHistogramYCoCg
```

Beispiel 1: Ergebnisse des YCoCg-Histogramms für Bilder aus "Test_10×5"

2 Kantenerkennung

Ein Feature, welches ebenfalls viel versprechend ist, ist ein Kantenhistogramm. Eine mögliche Umsetzung davon ist das MPEG7-Edge-Histogram.

2.1 MPEG7-Edge-Histogramm

Das MPEG7-Edge-Histogramm ist so aufgebaut, dass ein gegebenes Bild in 4×4 Regionen zerlegt wird. Diese Regionen werden wiederum in $2n \times 2n$ Bildblöcke unterteilt. Die Bildblöcke werden auch wieder aufgeteilt in 2×2 Bereiche. Für jeden dieser Bereiche wird eine durchschnittlicher Grauwert ermittelt, mit dessen Information dann über eine Funktion ein Wert zwischen -2 und 2 errechnet wird. Über die Anordnung der Werte wird dann festgestellt, ob der Bereich eine vertikale, horizontale, 45 oder 135 Grad diagonale oder ungerichtete Eigenschaft hat.

Die Umsetzung des MPEG7-Edge-Histogramms wurde für dieses Projekt etwas abgewandelt und beim Feststellen der Bereichswerte über eine Funkti-

on nur appoximiert umgesetzt, da eine konkrete Funktionsdefinition fehlte. Im implementierten an das MPEG7-Edge-Histogramm angelehten Feature-Vektor wird das gegebene Bild in $n \times n$ verschiedene Blöcke aufgeteilt. Für diese Bildblöcke wird jeweils für seine 2×2 Bereiche ein durchschnittlicher Grauwert ermittelt. Dieser Grauwert wird mit dem durchschnittlichen Grauwert aller 2×2 Bereiche verglichen (siehe Code-Ausschnitt in Beispiel 2). Wenn zum Beispiel die Felder 0 und 1 heller und die Felder 2 und 3 dunkler als der durchschnittliche Grauwert sind, wird dem Bildblock eine horizontale Eigenschaft zugewiesen.

```
1  private static int getEdgeIndex(int meanGrey, int grey1, int
       grey2, int grey3, int grey4){
2      if((meanGrey < grey1 && meanGrey < grey2 && meanGrey >
       grey3 && meanGrey > grey4)
3           || (meanGrey > grey1 && meanGrey > grey2 && meanGrey
       < grey3 && meanGrey < grey4) ){
4        //assume horizontal edge
5        return 0;
6      }
7      else if((meanGrey < grey1 && meanGrey > grey2 && meanGrey
       < grey3 && meanGrey > grey4)
8           || (meanGrey > grey1 && meanGrey < grey2 && meanGrey
       > grey3 && meanGrey < grey4) ){
9        //assume vertical edge
10       return 1;
11     }
12     else if(meanGrey > grey1 && meanGrey > grey4){
13       //assume 45 degrees diagonal edge
14       return 2;
15     }
16     else if(meanGrey > grey2 && meanGrey > grey3){
17       //assume 135 degrees diagonal edge
18       return 3;
19     }
20     else {
21       //assume non-directional edge
22       return 4;
23     }
24   }
```

Beispiel 2: `getEdgeIndex`-Methode der `EdgeHistogram`-Klasse

Es ist erkennbar, dass die übergeordnete Unterteilung des Bildes in 4×4 Regionen in der Implementierung, relativ unkompliziert noch zu ergänzen wäre und in Kombination mit einer anderen Funktion, um die Ausrichtung des Bildblocks zu ermitteln, die Ergebnisse noch verbessert werden können. Mit der Bisherigen Implementierung liegt der beste mAP-Wert bei $0,5192$ mit einer Unterteilung des Bildes in 17×17 Bereiche (siehe Beispiel 3).

```
res: 2
MAP: 0.3894 took 13ms for feature factory EdgeHistogram
res: 3
MAP: 0.4204 took 19ms for feature factory EdgeHistogram
res: 4
MAP: 0.4009 took 14ms for feature factory EdgeHistogram
res: 5
MAP: 0.4346 took 23ms for feature factory EdgeHistogram
res: 6
MAP: 0.4584 took 20ms for feature factory EdgeHistogram
res: 7
MAP: 0.4266 took 18ms for feature factory EdgeHistogram
res: 8
MAP: 0.4402 took 15ms for feature factory EdgeHistogram
res: 9
MAP: 0.4819 took 20ms for feature factory EdgeHistogram
res: 10
MAP: 0.4628 took 12ms for feature factory EdgeHistogram
res: 11
MAP: 0.4517 took 10ms for feature factory EdgeHistogram
res: 12
MAP: 0.4555 took 10ms for feature factory EdgeHistogram
res: 13
MAP: 0.4390 took 12ms for feature factory EdgeHistogram
res: 14
MAP: 0.4664 took 13ms for feature factory EdgeHistogram
res: 15
MAP: 0.4631 took 11ms for feature factory EdgeHistogram
res: 16
MAP: 0.4542 took 14ms for feature factory EdgeHistogram
res: 17
MAP: 0.5192 took 10ms for feature factory EdgeHistogram
res: 18
MAP: 0.4938 took 14ms for feature factory EdgeHistogram
res: 19
```

```
36 MAP: 0.4719 took 14ms for feature factory EdgeHistogram
37 res: 20
38 MAP: 0.4910 took 10ms for feature factory EdgeHistogram
```

Beispiel 3: Ergebnisse des Edge-Histogramms für Bilder aus "Test_10 × 5"

3 Zusammenführen von Feature-Vektoren

Hinter dem Zusammenführen von verschiedenen Feature-Vektoren steht die Idee, durch die Verwendung und Vergleich unterschiedlicher Features, mehr Informationen über die Bilder zu erhalten und vergleichen zu können und somit bessere Suchergebnisse zu erhalten.

3.1 Evolutionsstrategie

Die Evolutionsstrategie ist eine Optimierung der Zusammenführung der verschiedenen Feature-Vektoren, in der die für die Zusammenführung verwendeten Feature-Vektoren unterschiedlich gewichtet werden. Das Finden der optimalen Gewichtung erfolgt "*evolutionär*" Die Gewichtung von zwei verschiedenen Feature-Vektoren F_1 und F_2 erfolgt durch einen Wert alpha, wobei alpha zwischen 0 und 1 liegt. Der erste Feature-Vektor F_1 wird mit alpha multipliziert während der zweite F_2 mit $(1-\alpha)$ multipliziert wird. Anschließend werden die gewichteten Feature-Vektoren-Werte zu einem neuen Feature-Vektor zusammengeführt (z.B. durch Addition).

Um die Gewichtung auf das Zusammenführen von drei Vektoren F_1, F_2, F_3 zu erweitern, kann ein weiterer Wert β' eingeführt werden. Auch β' liegt im Bereich von 0 bis 1. Der Wert von β' wird mit $(1-\alpha)$ multipliziert, um den absoluten prozentuellen Anteil, β, festzustellen. Letztlich wird der Feature-Vektor F_1 mit alpha, F_2 mit β und F_3 mit $(1-\alpha-\beta)$ multipliziert und im Anschluss zusammengeführt, wieder zum Beispiel per Addition der gewichteten Vektoren.

Da die Vektoren F_1, F_2, F_3 unterschiedliche Längen bzw. Anzahl von Einträgen haben können, werden beim Zusammenführen die Werte der Vektoren, wenn sie existieren, addiert (oder eine andere Operation ausgeführt) oder – ansonsten – angenommen, dass sie 0 sind und addiert oder – gleichbedeutend

– es werden an der Stelle, an der es keine entsprechenden Werte gib, keine Werte addiert. Der zusammengeführte Vektor hat die Länge des längsten Feature-Vektors von F_1, F_2 und F_3.

- $0 < \alpha < 1$
- $\beta = (1 - \alpha) \cdot \beta'$, wobei $0 < \beta' < 1$
- $\gamma = 1 - \alpha - \beta$

```
1  bins: 4 alpha: 0.3 beta: 0.3
2  MAP: 0.7184 took 67ms for feature factory FeatureCombination
3  bins: 5 alpha: 0.3 beta: 0.4
4  MAP: 0.7186 took 27ms for feature factory FeatureCombination
5  bins: 5 alpha: 0.3 beta: 0.6
6  MAP: 0.7081 took 16ms for feature factory FeatureCombination
7  bins: 5 alpha: 0.3 beta: 0.1
8  MAP: 0.7037 took 17ms for feature factory FeatureCombination
9  bins: 5 alpha: 0.5 beta: 0.4
10 MAP: 0.7174 took 17ms for feature factory FeatureCombination
11 bins: 5 alpha: 0.8 beta: 0.4
12 MAP: 0.7041 took 16ms for feature factory FeatureCombination
13 bins: 5 alpha: 0.1 beta: 0.4
14 MAP: 0.7201 took 17ms for feature factory FeatureCombination
15 bins: 5 alpha: 0.1 beta: 0.8
16 MAP: 0.6789 took 12ms for feature factory FeatureCombination
17 bins: 5 alpha: 0.1 beta: 0.2
18 MAP: 0.7118 took 22ms for feature factory FeatureCombination
```

Beispiel 4: `FeatureCombination`-Ergebnisse für Bilder aus "Test_10×5". Die Features "Thumbnail", "RGB-Histogram" und "YCoCg-Histogram" wurden kombiniert.

Tests mit dem Bildset im Ordner "Test_10 × 5" haben gezeigt, dass eine Kombination der Feature-Vektoren einen insgesamt besseren mAP-Wert liefert (siehe 4). Der Beste Wert der Experimente liegt bei $0,7201$ für fünf Bins für die Histogramme im YCoCg- und RGB-Farbraum und einer im Beispiel nicht sichtbaren Thumbnail-Auflösung von 4×4. Die Anteile der Features waren $\alpha = 0,1$ für das YCoCg-Histogramm, $\beta' = 0,4$ und somit $\beta = 0,36$ für den Thumbnail und letztlich $\gamma = 0,54$ für das RGB-Histogramm.

Literatur

[1] Farbraum YCoCg, https://en.wikipedia.org/wiki/YCoCg, 29.07.2017.

[2] Dong Kwon Park, Yoon Seok Jeon, and Chee Sun Won. 2000. Efficient use of local edge histogram descriptor. In Proceedings of the 2000 ACM workshops on Multimedia (MULTIMEDIA '00). ACM, New York, NY, USA, 51-54. DOI=http://dx.doi.org/10.1145/357744.357758