# Aufgabe 3: MPEG

***Werkzeuge:*** *Eclipse 3.6.2 Helios*

***Zwischenstand:*** *08.06.2011, 13:00 Uhr*
***Abgabetermin:*** *20.06.2011, 13:00 Uhr*

**Bevor Sie beginnen**

Stellen Sie eine Verbindung mit *\\nas2\MMS* her. Falls Sie über SFTP zugreifen, verbinden Sie sich mit *serv9.inf.tu-dresden.de* und wechseln Sie dann in das Verzeichnis */zbv/mms*. Legen Sie dort unter */3_MPEG/* ein neues Unterverzeichnis an und benennen Sie dieses nach Ihrer Matrikelnummer. Legen Sie Ihre Ergebnisdateien in diesem Verzeichnis ab.

**Allgemeine Hinweise**

Die für diese Aufgabe benötigten Quelldateien finden Sie auf der Übungswebsite.

Neben den Quelldateien steht Ihnen eine Framework-Anwendung mit grafischer Benutzeroberfläche zur Verfügung, mit deren Hilfe Sie überprüfen können, ob Ihre Algorithmen korrekt funktionieren. Die Anwendung nutzt Ihre Algorithmen, um Bilder umzuwandeln bzw. Makroblockdateien zu erzeugen. Anschließend versucht die Anwendung, aus den Ergebnissen ein Bild zu rekonstruieren. Dieses sollte dann dem Originalbild entsprechen. Haben Sie einen Fehler in Ihren Algorithmen, werden als Resultat Bildfehler oder Exceptions in der Kommandozeile auftreten.

## *Alle Ergebnisse müssen folgende Anforderungen erfüllen:*

- Der Quellcode sollte so strukturiert und kommentiert sein, dass er für die Bewertung nachvollziehbar ist.
- *Alle verwendeten Bestandteile müssen entweder Public Domain sein oder die erforderlichen Nutzungsrechte müssen schriftlich vorliegen.*

# Aufgabe 3.1

## *Perspektivwechsel: Umwandlung von RGB in $YC_bC_r$*

Als einer der ersten Arbeitsschritte bei der MPEG-Codierung ist es notwendig, die Repräsentation des Bildes vom bekannten RGB-Farbmodell in $YC_bC_r$ zu überführen. Durch die Trennung von Helligkeits- und Farbinformationen wird es möglich, diese mit unterschiedlicher Genauigkeit zu speichern.

Ihre Aufgabe ist es, den Algorithmus zu ergänzen, der diese Umwandlung vornimmt. Lesen Sie dazu die Pixel nacheinander aus und erzeugen Sie die entsprechenden $YC_bC_r$-Werte. Verwenden Sie dazu die unten angegebene Vorschrift. Führen Sie anschließend eine Auflösungsreduktion der Farbebenen $C_b$ und $C_r$ im Format 4:2:0 (siehe Vorlesung) durch.

$$\begin{pmatrix} Y \\ C_b \\ C_r \end{pmatrix} = \begin{pmatrix} 0 \\ 128 \\ 128 \end{pmatrix} + \begin{pmatrix} 0{,}299 & 0{,}587 & 0{,}114 \\ -0{,}1687 & -0{,}3313 & 0{,}5 \\ 0{,}5 & -0{,}4187 & -0{,}0813 \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix}$$

Die dabei entstehende Datenstruktur kann wie folgt visualisiert werden:

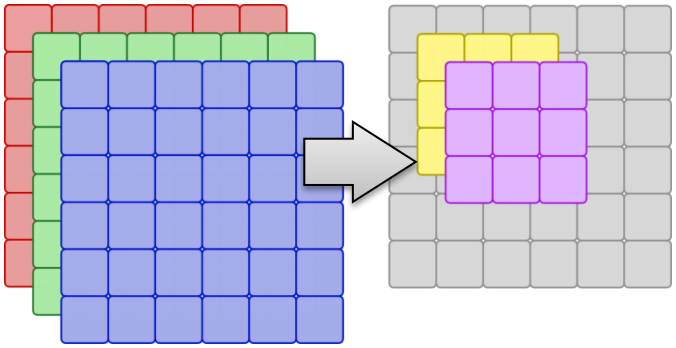

**Abbildung 1:** Umwandlung einer RGB-Pixelmatrix in $YC_bC_r$

| **Quelldateien** | **Abgabedateien** | **Bewertung** |
|---|---|---|
| - Exercise1.java<br>- frame01.jpg<br>- frame02.jpg | - Exercise1.java | 30 P |

# Aufgabe 3.2

## *Blocksatz: Umwandlung der Datenstruktur in Makroblöcke*

Um Bildausschnitte in späteren Bildern wiederzufinden, müssen die Einzelbilder nun noch jeweils in einheitliche Bestandteile, sogenannte Makroblöcke (16x16 Pixel) unterteilt werden. Legen Sie dazu eine Matrix von Makroblöcken entsprechend der Bildgröße an. Ordnen Sie nun jedem Makroblock die zuvor erzeugten $YC_bC_r$-Werte zu. Generieren Sie nun mithilfe der mitgelieferten Framework-Anwendung aus den Bildern frame01.jpg und frame02.jpg entsprechende „Makroblockdateien“ **frame01.blk** und **frame02.blk**.

| **Quelldateien** | **Abgabedateien** | **Bewertung** |
|---|---|---|
| - Exercise2.java<br>- frame01.jpg<br>- frame02.jpg | - Exercise2.java<br>- frame01.blk<br>- frame02.blk | 20 P |

# Aufgabe 3.3

## *Puzzle in Motion: Blocksuche und Differenzkodierung*

Ausgehend von den zuvor ermittelten Makroblöcken soll nun das Bild frame02.jpg als sogenanntes P-Frame gespeichert werden. Implementieren Sie dazu ein *Blocksuchverfahren Ihrer Wahl* (z.B. eine heuristische zweidimensionale logarithmische Suche, Vorlesung Seite 19). Als Ähnlichkeitsmaß verwenden Sie der Einfachheit halber die summierte Differenz der $YC_bC_r$-Werte eines Blocks. Als Schwellwert für einen „ähnlichen" Block verwenden Sie bitte den Wert *9000*.

Erzeugen Sie anschließend eine Matrix von Differenz-Makroblöcken. Nutzen Sie dazu folgende Regeln:

1) *Wurde ein ähnlicher Block im vorherigen Bild gefunden, speichere die relativen Koordinaten des Originalblocks sowie eine Differenzmatrix für die $YC_bC_r$-Ebenen.*
2) *Wurde kein ähnlicher Block im vorherigen Bild gefunden, speichere den Makroblock „normal", d.h. wie in Aufgabe 3.2 ermittelt.*

Wenden Sie Ihren Algorithmus nun auf frame02.jpg an, um daraus einen P-Frame zu frame01.jpg zu erzeugen. Speichern Sie den von Ihrem Algorithmus erzeugten Differenz-Frame mithilfe der mitgelieferten Framework-Anwendung als **p-frame.blk**.

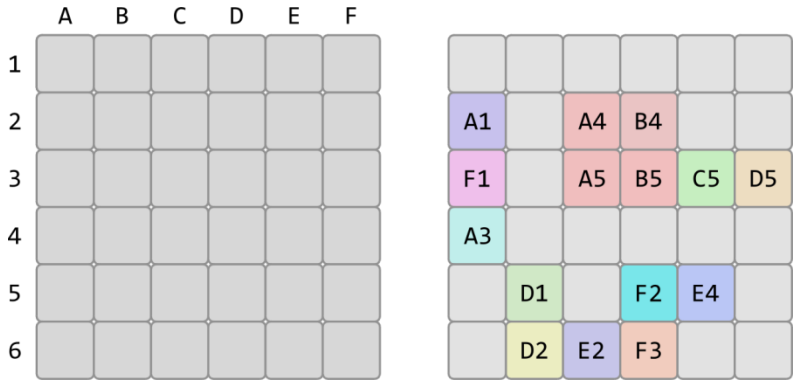


**Abbildung 2:** Erster Frame einer Sequenz (links) und zweiter Frame differenzkodiert (rechts). Graue Makroblöcke werden „normal" gespeichert; Farben stehen beispielhaft als Maß der Differenz zum referenzierten Block aus dem Vor-Bild

**Lösungshinweis:** Es kommt es nicht darauf an, alle Blöcke zuzuordnen. Ihr Algorithmus sollte jedoch zumindest 30% der Blöcke wiederfinden können, um eine Datenreduktion zu ermöglichen.

**Quelldateien**
- Exercise3.java
- frame01.jpg
- frame02.jpg

**Abgabedateien**
- Exercise3.java
- p-frame.blk

**Bewertung**
50 P