Aufgaben zu Muxing und Transmuxing

Lösen Sie die nachfolgenden Aufgaben und bereiten Sie diese bis zum nächsten Lehrveranstaltungstermin vor. Unterstrichene Aufgaben sind nach Möglichkeit während der Lehrveranstaltung zu lösen.

LB-MTM 01.

a) Verwenden Sie eine H.264-kodierte Videodatei (ohne Container, d.h. nach Annex B) aus einer Ihrer bisherigen Laboraufgaben (alternativ können Sie auch eine entsprechend kodierte Datei aus anderen Quellen verwenden). Muxen Sie diese mit *ffmpeg* in einen MP4-Container und ermitteln Sie den Containeroverhead, indem Sie die Größe der gemuxten Datei mit jener der Ausgangsdatei vergleichen.

b) Wie erhöht sich der Overhead aus a), wenn Sie eine Datei mit unterschiedlichen Kodierparametern verwenden?

c) Wie erhöht sich der Overhead aus a), wenn Sie zusätzlich eine „MP3"-Tonspur in den Container muxen? Verwenden Sie der Einfachheit halber eine beliebige „MP3"-Datei, die durch grobes Schneiden mit *Audacity* oder anderen Werkzeugen in etwa dieselbe Länge wie die Videodatei hat.

LB-MTM 02.

a) Analysieren Sie die Datei aus LB-MTM 01. c) mit *MP4Muxer* (`http://www.videohelp.com/tools/MP4Muxer`) und versuchen Sie, den Inhalt je einer audiospurspezifischen, einer videospurspezifischen und einer anderen Box mit Hilfe der Vorlesungsunterlagen zu deuten.

b) Analysieren Sie die in a) gewählten Boxes mit *Mp4 Explorer* (`https://mp4explorer.codeplex.com/`) und heben Sie die Unterschiede zu Ihren Ergebnissen aus a) hervor.

LB-MTM 03.

a) Transmuxen Sie den Inhalt des DivX-Containers aus LB-TK 02. mit *ffmpeg* in einen MP4-Container. Bestimmen Sie den Unterschied der Dateigrößen und schließen Sie daraus, welcher der beiden Container für die vorliegenden Daten besser geeignet ist.

b) Bestimmen Sie den absoluten Overhead der beiden Container aus a), indem Sie die enthaltenen Datenströme demuxen und deren Gesamtgröße als Basiswert für die Overheadberechnung verwenden.

c) Finden Sie zumindest ein weiteres Containerformat, das die Datenströme aus b) aufnehmen kann und bestimmen Sie dessen Overhead.

d) Vergleichen Sie die Overheads der Container aus a) und c) und schließen Sie daraus, welcher der beiden Container für vorliegenden Daten besser geeignet ist.

LB-MTM 04.

a) Schreiben Sie ein C-Programm, das *libx264* (vgl. unten) zur H.264-Kodierung von Rohdaten aus einer IYUV-Datei verwendet. Verwenden Sie zum Einlesen der Rohdaten Ihren Code aus LB-VK 04. und die Sequenzen aus LB-VK 01. Der Pfad der zu kodierenden IYUV-Datei sowie deren Bildbreite und -höhe sollen als Kommandozeilenparameter in ebendieser Reihenfolge angegeben werden können.

b) Stellen Sie das Muxingbeispiel aus LB-MTM 01. a) nach, indem Sie Ihr Programm aus a) derart erweitern, dass die H.264-kodierten Daten an *Libav* weitergereicht werden, um sie in einen MP4-Container zu muxen, der abschließend in eine Datei geschrieben wird.

Ergänzung zu LB-MTM 04.: Verwendung von *libx264*

libx264 ist ein Teil von *x264*, dessen Download und Installation in der „Anleitung zu *x264*“ erläutert ist. Bei der Kompilierung von *x264* wird *libx264* im Regelfall automatisch erstellt. Zur Beschleunigung des Kompiliervorganges kann *libx264* auch separat, d.h. insbesondere ohne die Kommandozeilenapplikation *x264*, erstellt werden:

```
./configure --enable-pic --enable-static --disable-cli
```

Eine noch striktere Konfiguration erlaubt eine weitere Beschleunigung des Kompiliervorganges sowie eine Reduktion der finalen Dateigröße, indem die für das vorliegende Programmierbeispiel nicht erforderlichen Features deaktiviert werden:

```
./configure --enable-pic --enable-static --disable-avs
--disable-swscale --disable-lavf --disable-ffms --disable-gpac
--disable-interlaced --disable-cli --disable-opencl
--chroma-format=420 --bit-depth=8
```

Die Verwendung von *libx264* ist in `x264.h` ausführlich dokumentiert. Zudem enthält `x264.c` (im *x264*-Repository) ein **sehr** umfangreiches Beispiel als Anhaltspunkt.