ffmpeg

Merge to files:

```
ffmpeg -i input_video.mp4 -i input_audio.m4a -c copy output_file.mp4
```

Convert Videos to smaller files:

```
#!/bin/bash

# Pfad zum Verzeichnis mit den Originalvideos
VIDEO_DIR="./"
# Zielverzeichnis für die komprimierten Videos
OUTPUT_DIR="./converted"

# Erstelle das Zielverzeichnis, falls es nicht existiert
mkdir -p "$OUTPUT_DIR"

# Durchlaufe alle .mp4 Dateien im Verzeichnis
for original in "$VIDEO_DIR"/*.MP4; do
```

```
    # Dateiname ohne Pfad
    filename=$(basename -- "$original")
    # Vollständiger Pfad zur Ausgabedatei
    output="$OUTPUT_DIR/$filename"

    echo "Komprimiere: $original -> $output"

    # Führe ffmpeg aus, um das Video zu komprimieren
    ffmpeg -i "$original" -c:v libx265 -crf 23 -preset medium -c:a copy "$output"
done

echo "Kompression aller Videos abgeschlossen."
```

Anleitung: Videos in Einzelbilder aufteilen mit `ffmpeg`

Ziel

Mit dieser Anleitung können Sie Videos in Einzelbilder aufteilen. Dies kann nützlich sein, wenn Sie beispielsweise eine Struktur-aus-Bewegung-Rekonstruktion mit Tools wie COLMAP durchführen möchten.

Voraussetzungen

- `ffmpeg` muss auf Ihrem System installiert sein. Wenn es noch nicht installiert ist, können Sie es mit dem folgenden Befehl installieren:

  ```
  sudo apt install ffmpeg
  ```

Schritte

1. **Bash-Skript erstellen und an die richtige Stelle verschieben**
   Führen Sie den folgenden Befehl aus, um das Skript zu erstellen und es direkt nach `/usr/local/bin/split-videos` zu schreiben:

```
echo '#!/bin/bash

# Überprüfen Sie, ob ffmpeg installiert ist
if ! command -v ffmpeg &> /dev/null; then
    echo "ffmpeg ist nicht installiert. Bitte installieren Sie es zuerst."
    exit 1
fi

# Standardwerte für Eingabe- und Ausgabepfade
INPUT_PATH="."
OUTPUT_PATH="."

# Optionen mit getopts verarbeiten
while getopts "i:o:" opt; do
    case "$opt" in
    i) INPUT_PATH="$OPTARG" ;;
    o) OUTPUT_PATH="$OPTARG" ;;
    *) echo "Ungültige Option: -$OPTARG" >&2; exit 1 ;;
    esac
done

# Gängige Videoformate
VIDEO_FORMATS=("*.mp4" "*.avi" "*.mkv" "*.mov" "*.flv" "*.wmv")

# Durchlaufen Sie alle gängigen Videoformate im angegebenen Eingabeverzeichnis
shopt -s nullglob
for format in "${VIDEO_FORMATS[@]}"; do
    videos=($INPUT_PATH/$format)
    for video in "${videos[@]}"; do
        # Entfernen Sie die Dateierweiterung, um den Ordnernamen zu erhalten
        dir_name="${video##*/}"
        dir_name="${dir_name%.*}"

        echo "Verarbeite Video: $video"

        # Erstellen Sie ein Verzeichnis im angegebenen Ausgabepfad mit dem Namen des Videos
```

```
        mkdir -p "$OUTPUT_PATH/$dir_name"

        # Teilen Sie das Video in Einzelbilder auf und speichern Sie sie im entsprechenden Verzeichnis
        ffmpeg -i "$video" -q:v 2 "$OUTPUT_PATH/$dir_name/frame_%04d.jpg"
        echo "Bilder gespeichert in: $OUTPUT_PATH/$dir_name/"
    done
done



echo "Alle Videos wurden erfolgreich in Einzelbilder aufgeteilt!"' | sudo tee /usr/local/bin/split-videos > /dev/null
```

2. **Skript ausführbar machen**
   Machen Sie das Skript mit dem folgenden Befehl ausführbar:

   ```
   sudo chmod +x /usr/local/bin/split-videos
   ```

3. **Skript ausführen**
   Sie können das Skript jetzt von überall auf Ihrem System ausführen:

   ```
   split-videos
   ```

   Oder mit spezifischen Eingabe- und Ausgabepfaden:

   ```
   split-videos -i /path/to/videos -o /path/to/output
   ```

Hinweis

- Das Skript sucht automatisch nach den gängigsten Videoformaten, sodass Sie nichts anpassen müssen.

Normalize-Audio:

Just normalize

create file `normalize_wav.sh`

```
#!/bin/bash

# Verzeichnis mit den WAV-Dateien
input_dir="$1"

# Zielverzeichnis für die normalisierten Dateien
output_dir="${input_dir}/loudness-normalized"

# Erstelle das Zielverzeichnis, falls es nicht existiert
mkdir -p "$output_dir"

# Durchlaufe alle WAV-Dateien im Eingabeverzeichnis
for wav_file in "$input_dir"/*.WAV; do
    # Dateiname ohne Verzeichnis
    filename=$(basename "$wav_file")

    # Normalisierte Datei im Zielverzeichnis
    normalized_file="${output_dir}/${filename}"

    # Normalisieren der Lautstärke
    ffmpeg -i "$wav_file" -af "loudnorm=I=-23:LRA=7:TP=-2" "$normalized_file"

    echo "Normalisiert: $filename -> $normalized_file"
done

echo "Alle Dateien wurden normalisiert und nach $output_dir verschoben."
```

run `zsh normalize_wav.sh ./`

or

`bash normalize_wav.sh ./`

with to pass noize remover:

```
#!/bin/bash

# Verzeichnis mit den WAV-Dateien
```

```
input_dir="$1"

# Zielverzeichnis für die normalisierten Dateien
output_dir="${input_dir}/loudness-normalized"

# Temporäres Verzeichnis für die rauschgefilterten Dateien
temp_dir="${input_dir}/temp-noise-removed"

# Erstelle die Verzeichnisse, falls sie nicht existieren
mkdir -p "$output_dir"
mkdir -p "$temp_dir"

# Durchlaufe alle WAV-Dateien im Eingabeverzeichnis
for wav_file in "$input_dir"/*.WAV; do
    # Dateiname ohne Verzeichnis
    filename=$(basename "$wav_file")

    # Temporäre Datei für rauschgefilterte Datei
    temp_file="${temp_dir}/${filename}"

    # Normalisierte Datei im Zielverzeichnis
    normalized_file="${output_dir}/${filename}"

    # Erstelle eine temporäre Rauschprofildatei
    noise_profile="${temp_dir}/${filename}.prof"

    # Extrahiere eine kurze Rauschprobe und erstelle das Rauschprofil
    sox "$wav_file" -n trim 0 1 noiseprof "$noise_profile"

    # Rauschentfernung mit sox
    sox "$wav_file" "$temp_file" noisered "$noise_profile" 0.21

    # Überprüfen, ob die Datei erfolgreich erstellt wurde
    if [ -s "$temp_file" ]; then
        # Lautstärke normalisieren mit ffmpeg
        ffmpeg -i "$temp_file" -af "loudnorm=I=-23:LRA=7:TP=-2" "$normalized_file"
        echo "Rauschentfernung und Normalisierung: $filename -> $normalized_file"
    else
        echo "Fehler bei der Rauschentfernung: $filename"
    fi
```

```
    # Entferne die temporäre Rauschprofildatei
    rm "$noise_profile"
done

# Entferne das temporäre Verzeichnis und die darin enthaltenen Dateien
rm -r "$temp_dir"

echo "Alle Dateien wurden rauschgefiltert, normalisiert und nach $output_dir verschoben."
```

you need sox:

```
apt install -y sox
```

Revision #10
Created 22 August 2023 10:26:42 by joscha.mijailovic
Updated 5 November 2024 00:57:34 by joscha.mijailovic