Processing of 3D Signals (Video Sequences)

DSP Lab 4

Table of contents

1 Objective

Students should understand and be able to operate with video data in Matlab.

2 Theoretical aspects

2.1 Representation of video sequences

A video sequence is just a sequence of images displayed with speed high enough to "fool the eye" into thinking it sees a continuous movement. As a consequence, a video sequence can be represented as an array of images. The images are known as "frames".

Note: there is a difference between how a video sequence is stored to disk (with compression) and the raw uncompressed data. Here we talk about the uncompressed video data.

Grayscale video

For simplicity, we consider only grayscale video in this laboratory. Since a grayscale image is a matrix, a grayscale video can be represented as a 4D tensor with dimensions (H, W, 1, NoF), where:

- H = height of video (number of rows)
- W = width of image (number of columns)
- NoF = number of frames (number of distinct images in the video).

For example, V(20, 50, 1, 7) is the pixel value for the pixel at row 20, column 50, in the 7th frame.

Color video

In case of a color video, the data structure is a tensor with dimensions (H, W, 3, NoF). We have three components for the RGB values.

For example, V(20, 50, 2, 7) is the Green value (2) of the pixel at row 20, column 50, in the 7th frame of the video.

2.2 Matlab functions for video handling

There are multiple ways of handling video data in Matlab, depending on how it is represented. In the following, the recommended way is to work with 4D tensors.

In the following we provide code templates for working with video data.

Creating and playing videos as 4D tensor

A video is a 4_ tensor of shape (H, W, 1, NoF) (grayscale) or (H, W, 3, NoF) (color).

We play the video with `implay()`.

Code template below:

```
height = ...; % desired height
width  = ...; % desired width
NoF    = ...; % desired number of frames
```

```
% an array of size height x width x 1 x NoF:
video  = zeros(height, width, 1, NoF);
for i = 1:NoF
    video(:,:,:,i) = ... the frame number i ... ;
end

% Play the sequence
implay(video);

% Fix: ensure we don't have any value larger than 1, it crashes Matlab
video(video > 1) = 1;

% Save file to disk
aviObj = VideoWriter('OutputVideo.avi', 'Uncompressed AVI');
aviObj.open();
aviObj.writeVideo(video);
aviobj.close();
```

[Alternative] Creating and playing videos using Video objects

In this approach, we work with specially crafted Matlab objects for handling video.

```
% Prepare data structure for a new video file in grayscale
height = ...; % desired height
width  = ...; % desired width
NoF    = ...; % desired number of frames
video = struct('cdata', zeros(height,width,1,'uint8'), ...
   'colormap',colormap(gray(256)));

% Put each frame in the video data structure
for i = 1:NoF   % how many frames we want
    video(i).cdata = ... put here the i-th image ...;
end

% Play the sequence
implay(video);

% Save the video to disk
aviObj = VideoWriter('OutputVideo.avi', 'Uncompressed AVI');
open(aviObj);
```

```
for i = 1:numel(video)
   % Fix: ensure we don;t have any value larger than 1, it crashes Matlab
   video(i).cdata (video(i).cdata > 1) = 1;

   % Save to disk
   writeVideo(aviObj,ofmov(i).cdata);
end
close(aviObj);
```

2.3 Loading and processing frames from an existing video file

```
v = VideoReader(['FisierVideo.avi']);
height = v.Height;           % get height of the video frames
width  = v.Width;            % get width of the video frames
NoF    = v.NumberOfFrames;   % get total number of frames in the video

% Process every frame in the video
for i = 1:NoF
    frame = v.read(i);              % read frame number i
    ... do stuff ...
end
```

3 Exercises

1. Load the `Lena` image, convert it to a grayscale image, convert it to `double` type, adapt the values to the $[0, 1]$ range, and display it.
2. Create a video sequence by scrolling the Lena image circularly to the right, by 3 pixels at every frame. Display the video at 25fps and save it to disk.
3. Create another video sequence by progressively changing the average luminosity of the image from 0 to 1. The video sequence should last exactly 4 seconds at a frame rate of 25fps.
4. Load the video file `veh.mp4`. Convert each frame to grayscale, `double` type, and range $[0, 1]$. Display the video sequence.
5. **Background/foregound extraction**. Make a new video sequence as follows:

   output_frame = (1 - α) * previous_output_frame + α * current_input_frame

   Set $\alpha = 0.99$.

a. Display the video sequence. What happened? What kind of filter is this? (Hint: rewrite the equation in the usual way (with $x[n]$ and $y[n]$))

b. Create another video sequence as the difference between the original sequence and the sequence from a).

- Display the video sequence. What happens?
- Deduce the equation of this system. What kind of filter is this?

4 Final questions

TBD