UE VIDEOVERARBEITUNG 2017W

EXERCISE 3

GENERAL

The following filters have to be implemented for the first exercise:

1) **distortion_scratches.m**
2) **distortion_vineagar.m**
3) **distortion_grain.m**
4) **distortion_burn.m**

The filter functions have to be implemented in the above-mentioned files. All filters have a different amount of parameters, which control the properties of the filters. You can expand the number of parameters for every filter but you should be able to justify each new parameter. During the implementation of each filter, meaningful parameter values that result in authentic looking results have to be found.

All the files which are needed for implementing each filter (distortion_*.m) are in the /src directory. By calling the function exercise3.m all implemented filters are applied on an image sequence.

Example 1: **exercise3('../images/', '../output/');**
applies all filters to the frames frameXXXX.png in the directory '../images' and stores the resulting images in the directory '../output'.

Example 2: **exercise3('../images/', '../output/', start_frame, end_frame);**
processes only frames 'start_frame' to 'end_frame'.

Example 3: **exercise3({'../images/','../images/'}, '../output/');**
reads all images from 2 input directories '../images1' and '../images2' (in this order) and applies the filters to all images.

Example 4: **help exercise3**
displays help text for each file in the command window (in this example for the file exercise1).

SUBMISSION

The deadline for exercise3 is **18.12.2017**.

The submission must include:

- Commented Matlab source files of all filters.
- Following fields have to be completed at the beginning of each filter:
  - Short description of the implementation.
  - Short summary of the physical background/reason of the effect.
    For example: What causes Sepia discoloration?
  - Meaningful parameter values.
- One picture of every filter that demonstrates the operation of the filter.

You will get points for

- Your implementation
- The description of your filters
- The mention of meaningful parameter values

All files have to be uploaded as a .zip file (UE_GROUPx_EXERCISEy.zip) on TUWEL. Only one submission is needed per group.

NOTE: Check your submission for completeness because points will only be given to the submitted files!

GENERAL INFORMATION

DATA CONVERSION

You can use the function convert_video_to_images to extract frames from a video of your choice and save them as .png files. Be warned that this process might lead to a large consumption of disk space if the movie is very long.

Furthermore, the function convert_images_to_video does the opposite and combines all images from a directory into a movie. This can be used to create the final movie after the filters were applied.

IMPLEMENTATION OF THE FILTERS

All exercises are handed out with a Matlab framework. In this exercise the framework consists of the file exercise3.m. The framework automatically takes care of reading, saving and displaying the images. To each filter in this exercise, the struct video will be passed as the first argument. It has the following structure:

video.original	Original image in RGB
video.filtered	Filtered image in RGB
video.original_frame_nr	Frame number of this frame in the input directory (is not modified by any buffer!)
video.frame_nr	Frame number of this frame (can change if frames are inserted/removed by a filter)

When implementing a filter the following has to be taken into account:

1) Each filter works on the field video.filtered and saves the result there again. In this way, all filters can work sequentially on the same image.
2) To provide access to previous frames a frame buffer is installed. Thus, two previous frames can be addressed. You can access the current frame over video.frame(1).filtered, the previous one over video.frame(2).filtered.

video.frame(1).filtered	current frame i
video.frame(2).filtered	frame i-1
video.frame(3).filtered	frame i-2

3) You can perform any conversion of the input information. However, EVERY filter MUST save its results in the RGB format (RGB values between 0 and 255). Thus, no distinction and multiple implementations for different image and color formats are needed and the filters can be executed in an arbitrary order.

Example:

```
function filter_test(video)
  img = video.frame(1).filtered;
  …
  video.frame(1).filtered = img;
end
```

In order to pass information from one filter to another, the struct video will be expanded in the course of this lecture. Temporary filters, which operate on multiple images, require knowledge about the status of the filtering (What has been done so far? What has to be done in this step?).

HINT 1:

You can insert any additional fields in the struct video if you need them for the proper working of the filters.

HINT 2:

Matlab manages the images in a transposed way, which means that the image point (x,y) corresponds to the value (y,x) in the image structure. Each image in video.filtered and video.original has the format [H x W x C], where H represents the image height, W the image width and C the color depth. For the RGB color space C = 3 (R/B/G).

Here are some examples showing how to access the color information of the images:

Example 1:

```
r_info = video.filtered(4,3,1);        % x = 3, y = 4, c = 1
r_info = 34                            % red color information at the position (3,4)
```

Example 2:

```
rgb_info = video.filtered(4,3,1:3)     % x = 3, y = 3, c = 1:3
rgb_info = [34 12 120]                 % RGB values at the position (3,4)
```

Example 3:

```
r_info = video.filtered(1:10,1:10,1)   % x = 1:10, y = 1:10, c = 1
r_info = 10 x 10 matrix                % red color information at the positions (x = 1:10, y
                                                                                   1:10)
```

HINT 3:

Try all the filters in an arbitrary order to verify correct functioning.

Useful MATLAB links:

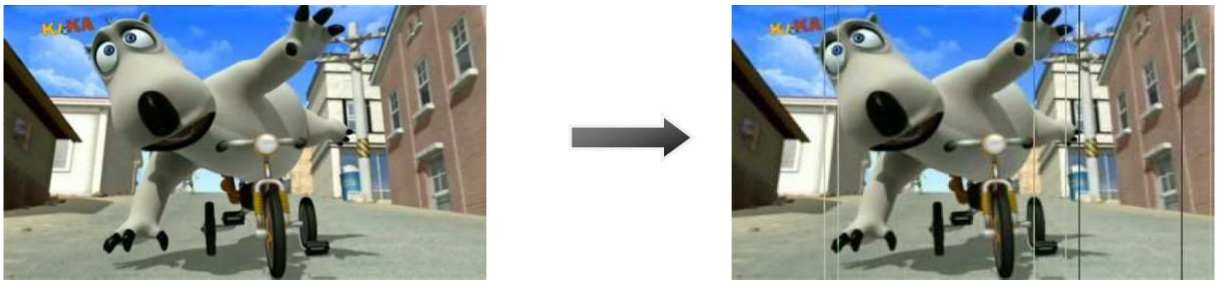
https://de.mathworks.com/help/matlab/ref/videoreader.html

https://de.mathworks.com/help/matlab/ref/videowriter.html

DISTORTION 1: DISTORTION_SCRATCHES

GENERAL INFORMATION

The vertical scratches that occur in old movies should be reproduced in this exercise.



IMPLEMENTATION

For this task, the following functions have to be implemented:

Function:	distortion_scratches
Input:	<video> struct, max amount of scratches
Output:	<video> struct with scratches
File:	distortion_scratches.m

1) Generate randomly distributed vertical scratches. The width of the scratches should be 1 px.
   ATTENTION: Check for every scratch if its position lies between the image boundaries.
2) The authenticity of the scratches is increased when the distance between the scratches appears to be random.
3) The color of the scratches can be bright as well as dark. Therefore, generate half of the scratches with bright intensities (e.g. values between 200 and 230) and the other half with dark intensities (e.g. values between 50 and 80). The scratches can be applied directly on the current frame. The actual brightness values are replaced with the randomly calculated values, which represent dark or bright scratches.
4) To get even more randomness into the image only some of the calculated scratches can be applied to the image.

REFERENCES

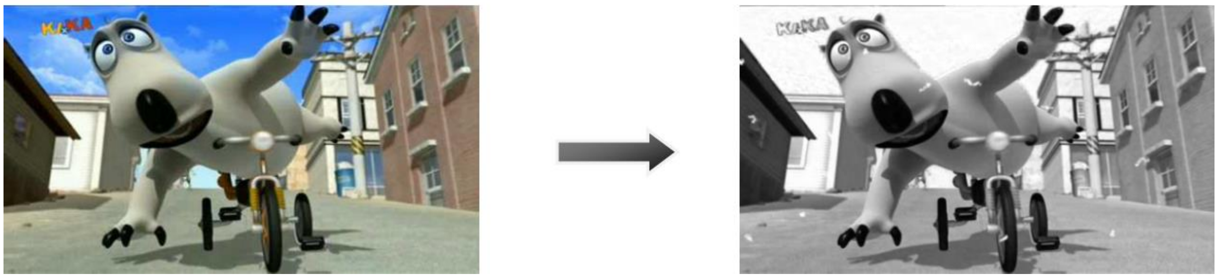
http://www.cgarena.com/freestuff/tutorials/ae/oldfilmlook/

http://www.nattress.com/Products/filmeffects/G_Film_Plus_RT.htm#10

DISTORTION 2: DISTORTION_VINEGAR

GENERAL INFORMATION

In this exercise, the vinegar damage should be reproduced. It is caused by poor storage of the films and is beside of mold and reddish one of the biggest damager for old movies.



IMPLEMENTATION

Firstly, this effect creates an empty alpha map for each frame (value = 0). The function generate_blob generates random blobs on the alpha map (connected points with alpha values = 1). Finally, alpha blending is used to apply the alpha map to the frames:

```
video.frame(1).filtered = (1-alpha) * video.frame(1).filtered + alpha * 1
```

A smoother transition between the original image and the alpha map can be achieved by applying an unsharp filter.

Following functions have to be implemented for this exercise:

Function:	generate_blob
Input:	2D Array 'map', x, y, blob_size
Output:	2D Array 'map' with a new blob
File:	distortion_vinegar.m

The function map = generate_blob(map, x, y, blob_size) generates a blob in the alpha map. The starting point of the blob is position (x, y):

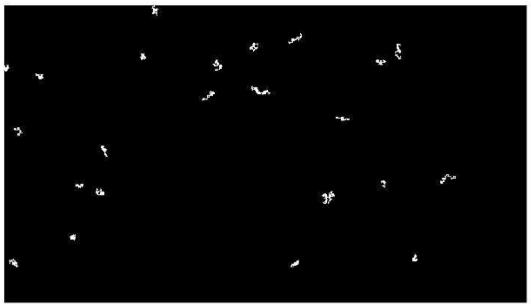
1. The point (x, y) is used as a seed point from which a random region with blob_size points is generated.
2. for i = 1:blob_size
   a. The current seed point is set to 1 on the alpha map.
   b. Generate a random direction vector (left, right, up, down).
   c. Go from the current seed point in this direction.

d. This is the new seed point.

ATTENTION: Check for each seed point whether its position is within the alpha map!

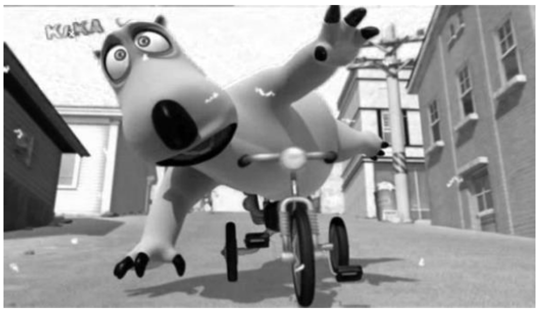
Function:	distortion_vinegar
Input:	<video> struct, number of blobs, maximal blob size
Output:	<video> struct with vinegar damage
File:	distortion_vinegar.m

1. Create a blank alpha map (value = 0).
2. Generate random blob positions (x1, y1), (x2, y2), ...
3. Repeated calls of the function generate_blob produce blobs at these points. Example of a finished alpha map with blobs:



4. Apply an unsharp filter on the final alpha map to get softer transitions. Example of a finished alpha map with an applied unsharp filter:

5. Use alpha blending to apply the alpha map to the frames:



REFERENCES

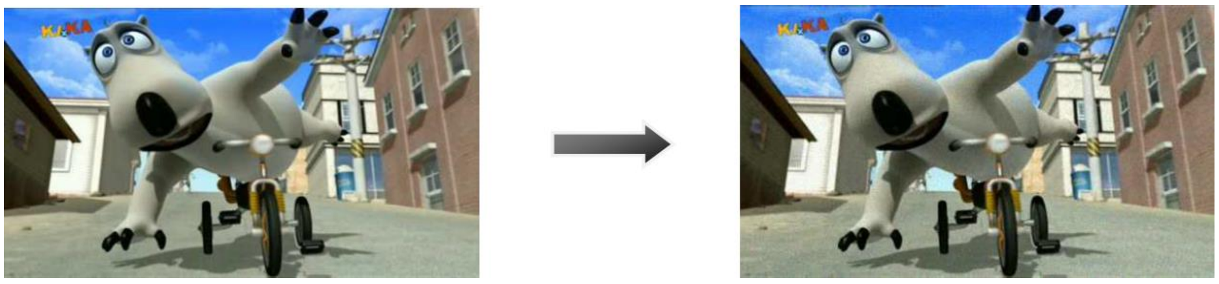
http://en.wikipedia.org/wiki/Cellulose_acetate_film#Decay_and_the_.22vinegar_syndrome.22

http://en.wikipedia.org/wiki/Alpha_compositing#Alpha_blending

DISTORTION 3: DISTORTION_GRAIN

GENERAL INFORMATION

The aim of this exercise is to reproduce film grain that is mainly caused by the physical characteristics of the recording material.



IMPLEMENTATION

For this example, the following functions have to be implemented:

Function:	distortion_grain
Input:	<video> struct, filter parameters
Output:	<video> struct with grain effect
File:	distortion_grain.m

1) Familiarize yourself with the MATLAB command imnoise.
2) Apply this filter to each of the three RGB channels in order to achieve the desired film grain.
3) Try different parameters for imnoise and select the one you think gives the best result.

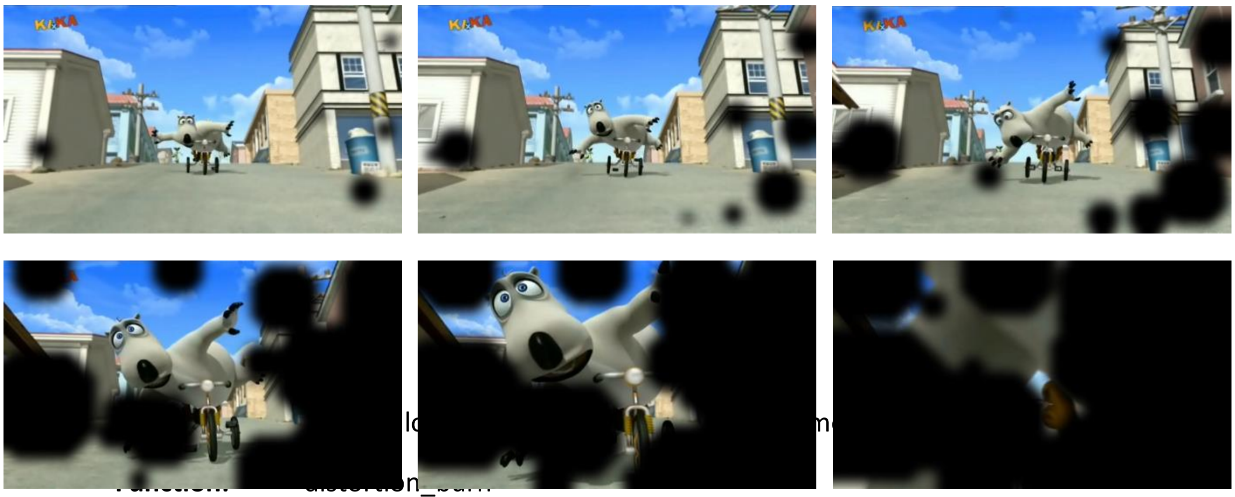
REFERENCES

http://en.wikipedia.org/wiki/Film_grain

DISTORTION 3: DISTORTION_BURN

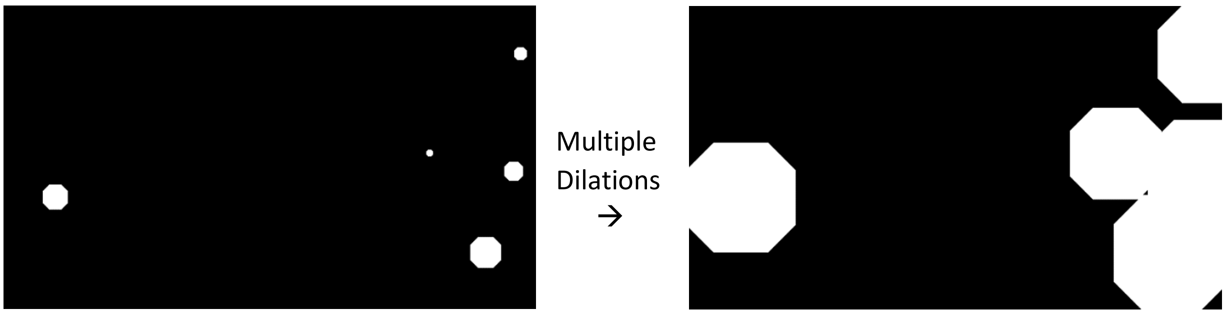
GENERAL INFORMATION

The aim of this exercise is to produce a simple effect where the film burns under the heat of the projector at the end of the movie.



Function:	distortion_burn
Input:	<video> struct, filter parameters
Output:	<video> struct with film burns
File:	distortion_burn.m

1) Determine when to start the burn effect.
2) Create an initial map with a single, randomly placed white pixel.
3) For each new frame
    1. Possibly place a new white pixel in the map. This could be done at random, so that not at every frame a new pixel is added.
    2. The map is dilated using a morphological operation with an appropriate structure element (e.g. disk).
    3. This result should be stored for the next frame.



4) The generated map should be inverted and blurred using a Gaussian filter.

5) For the final step, the map should be blended with the current frame to create dark spots.

REFERENCES

https://de.mathworks.com/help/images/ref/imdilate.html

http://en.wikipedia.org/wiki/Alpha_compositing#Alpha_blending