Exercise 3 – Merge

188.346 Videoverarbeitung UE 2016W

1 . Framework

In this exercise, the framework consists of the sub-directory */src*. The */src* directory includes the following files:

- add_shadow.m
- change_illumination.m
- create_video.m
- exercise3.m
- merge.m

You are supposed to complete this framework by implementing the tasks (3.a.-3.d.) described below. In order to accomplish this, you can find hints in the source files, such as this one:

```
%---------------------------------------------------------------------------
% Task a: Adjust brightness and resize foreground object
%---------------------------------------------------------------------------
```

You can start the framework by calling the function "exercise3" in the command window. For example: *exercise3('../fg_frames/','../bg_frames/','../fg_map/','../output/','png');*

The framework automatically takes care of reading and saving the frames.

2 . Submission

The deadline for exercise3 is **16.01.2017, 23:55.**

Your submission must include:

- the */src* directory including the commented matlab source files with the implemented tasks
- a pdf file with your answers on the two theoretical questions listed below
- last output frame - this is the frame with the number 129
- output frames packed to a video with a frame rate of 24 fps

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

To avoid any misunderstandings, the structure of the submission should look like this:

```
UE_GROUPx_EXERCISEy.zip
        /output
                frame00129.png
```

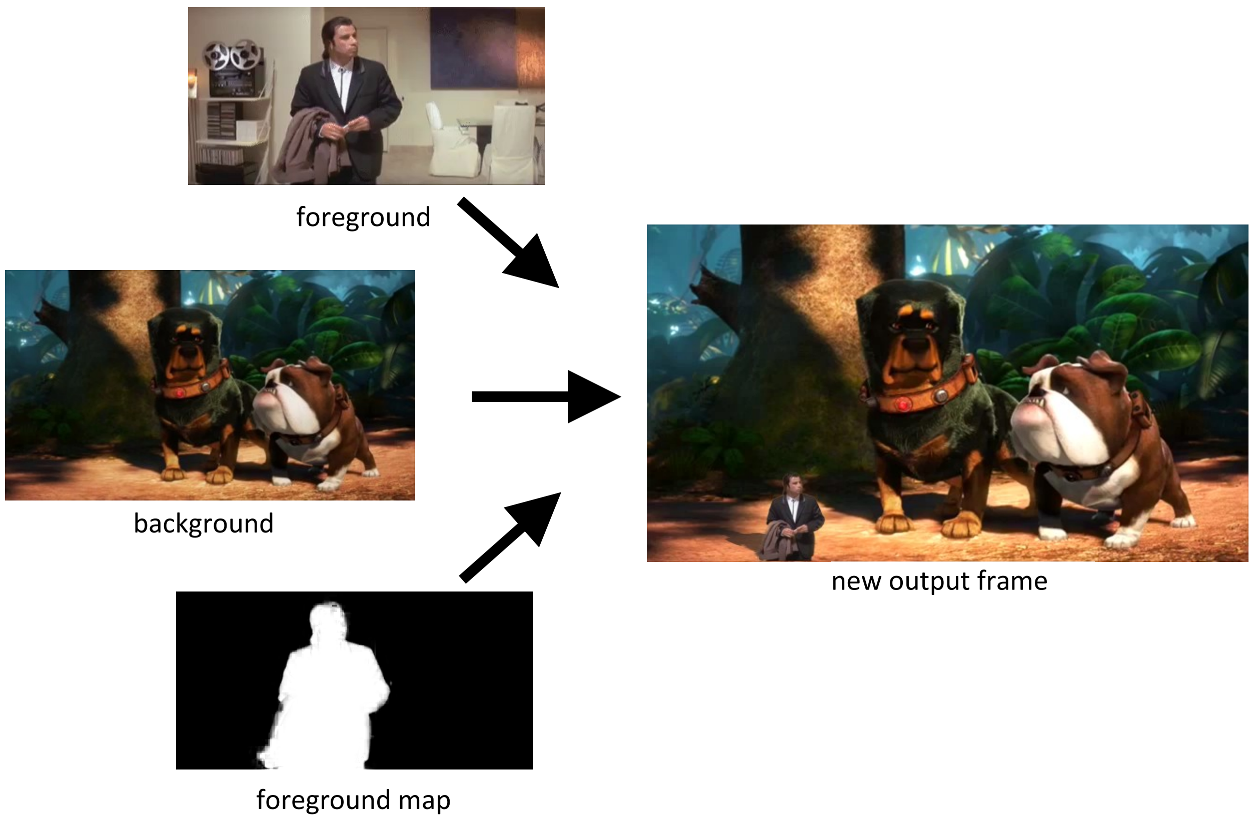
```
            output.avi
    /src
            add_shadow.m
            change_illumination.m
            create_video.m
            exercise3.m
            merge.m
    UE_GROUPx_EXERCISEy_theory.pdf
```

You will get points for:

- your implementation + meaningful comments
- your answers on the two theoretical questions
- output frame + output video
    - you also get points if you correct your solutions from exercise 1 and 2 to result in better input frames for this exercise

<u>NOTE</u>: Please make sure that your submission includes ALL required files because points will be given only on the submitted files! Also please make sure that your debugging code lines, like *"figure"*, *"imshow"* or "*subplot*" are not enabled in your final solution! Otherwise you will lose points.

3 . General Task



In the third exercise you will merge the foreground video frames, the foreground map output frames of the first exercise and the retimed background frames of the second exercise to one new video. Each of these inputs consists of 129 frames. So for each new frame of this new video you need three corresponding input frames. To do this, you first adjust the foreground and background frames, then add a shadow of the foreground object to the background frame, merge the new background with the foreground and finally create the output video.

Task a. Adjust brightness and resize foreground object

Implementation in:	change_illumination.m, exercise3.m
Input for this task:	foreground frame and foreground map
Output of this task:	resized and brightness adjusted foreground

First of all, you have to resize the foreground image and foreground map, so you can compute further calculations on a smaller image. You can choose any size that looks natural in proportion to the background (for example a height of 100 pixels). After this, adjust the brightness of the background, so it the lightning conditions of it are similar to the slightly brighter foreground. This is done by calling the function *“change_illumination”*. In this function, you have to convert the background frame into the HSV color space to easily modify one of the three channels that is responsible for the brightness of the colors. To accomplish this, take a look at the link http://en.wikipedia.org/wiki/HSL_and_HSV.
At the end of this function you have to convert the frame back to the RGB color space.

Task b. Add shadow of foreground object to background

Implementation in:	add_shadow.m, exercise3.m (call function *"add_shadow")*
Input for this task:	background frame and foreground map
Output of this task:	background frame with shadow of foreground

In the next step, the shadow of the foreground object is added to the background frame. To perform this task, you first compute a projection of the foreground map, so it looks like this:

Projected shadow of foreground map frame00001.png

This can be done by creating a spatial transformation structure with the MATLAB function *"maketform"*. This function needs two parameter values: a transformation matrix and a transformation type. You can use the following projective transformation matrix:

$$\begin{pmatrix} 10 & -3 & 0 \\ 10 & 5 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

With this created transformation structure you can call the MATLAB function *"imtransform"* with the foreground map, the transformation structure and the parameter "XYScale" with suitable scales. The "XYScale" parameter with a scaling vector (for example [12 22], if you use the transformation matrix above) is needed to specify the width and height of each output pixel of the transformation in X-Y space, so you don't have to scale it down after the transformation. With this step your shadow map is finished. Now you just have to merge it with the background. For an easier merge you first have to bring the shadow map to the same size as the background frame, so you generate a 2D-matrix in the size of the background image and paste the shadow on the position of the parameter values "xpos" and "ypos". After this you can finally compute the new background frame by making an addition of these three parts for each color channel:

1. [shadow map with a chosen value for the shadow color (this means instead of the ones in the shadow map you take a shadow color for each color channel)] * t
2. [shadow map with the background color (this means instead of the ones in the shadow map you take the background color at this position of each background color channel)] * (1 - t)
3. [(1 - shadow map) with the background color (this means instead of the zeros in the shadow map you take the background color at this position of each background color channel)]

t… a chosen transparent factor for the shadow (value between 0 and 1)

The first two parts are responsible for a transparent shadow, so for instance a transparent factor of 0.7 will result in an area consisting of 70 % shadow and 30 % background. The last part is for the other background areas with no foreground shadow – in other words, the areas which should result in 100 % of the background frame.

`Task c.` **Merge foreground and background**

Implementation in:	merge.m, exercise3.m (call function "*merge")*
Input for this task:	foreground frame, foreground map and new background frame with foreground shadow
Output of this task:	merge of background and foreground

In this step we merge the foreground object, the foreground map and the new background (the output of Task b.). This is done in the same way as the merging of the background and shadow in Task b. So you again resize the foreground map to the same size as the background frame by generating a 2D-matrix in the size of the background and paste the foreground map on the position of the parameter values "xpos" and "ypos". You do the same with the foreground frame, the only difference is that we use a 3D-matrix in this case. After this step, all 3 inputs (foreground map, foreground frame, background frame) have the same size and we can merge them in a similar way as in Task b. by taking the foreground if there is a one in the foreground map and taking the background if there is a zero.

`Task d.` **Create output video of all resulting frames**

Implementation in:	create_video.m, exercise3.m (call function "*create_video")*
Input for this task:	computed frames in output directory
Output of this task:	output video

The last step of this exercise is to create a video from the computed output frames. In the function "create_video" you have to initialize a VideoWriter object, set the framerate, prepare the object to write frames in it and finally write each computed output frame in the video object.

4 . Theoretical Questions

a. What is the difference between the MATLAB operators .* and * ? Demonstrate the difference on the basis of two simple examples.

b. What needs to be removed in the HSV color space, to eliminate the color information (to compute an image that contains only shades of gray)?