Exercise 1 – Segmentation

188.346 Videoverarbeitung UE 2016W

Framework

In this exercise, the framework consists of the two sub-directories */fg_frames* and */src*. In */fg_frames* you find all extracted frames of a given foreground video. The */src* directory includes the following files:

- boxfilter_vid.m
- colHist.m
- exercise1.m
- get_histograms.m
- guidedfilder_vid_color.m
- keepConnected.m
- segmentation.m

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

```
%--------------------------------------------------------------------------
% Task a: Filter user scribbles to indicate foreground and background
%--------------------------------------------------------------------------
```

You can start the framework by calling the function "exercise1" in the command window. For example: *exercise1('../fg_frames/','../output_fg_map/','png');*

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

HINT: A loop controlling how many frames should be read till the segmentation function is called is integrated in exercise1m. Caching too many frames may exceed your computer's memory, so you can adjust the caching size by modifying the parameter "loop_size".

Submission

The deadline for exercise1 is **05.12.2016, 23:55.**

Your submission must include:

- the */src* directory including the commented matlab source files with the implemented tasks
- the three additionally added frames in a directory */scribbled_frames*: one reference frame and two scribbled frames (see 3. task a.)
- a pdf file with your answers on the three theoretical questions listed below

- representative examples of your output: this means two foreground maps of different video frames in a directory */output,* to take a closer look on your computed output. One could be for example this one:

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

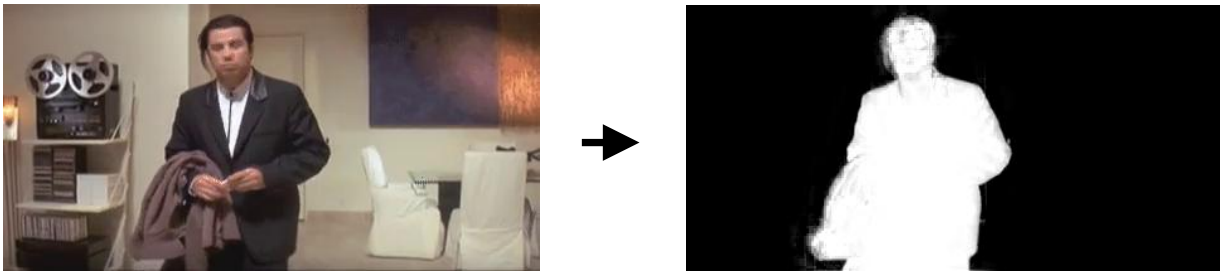
The structure of the submission should look like this:

<u>UE_GROUPx_EXERCISEy.zip</u>
- **/scribbled_frames** *(XXXX stands for your chosen reference frame)*
  - rframe-XXXX.png
  - sframe-XXXX.png
  - ssframe-XXXX.png
- **/output** *(YYYY and ZZZZ stands for two different output frames)*
  - frameYYYY.png
  - frameZZZZ.png
- **/src**
  - boxfilter_vid.m
  - colHist.m
  - exercise1.m
  - get_histograms.m
  - guidedfilder_vid_color.m
  - keepConnected.m
  - segmentation.m
- **UE_GROUPx_EXERCISEy_theory.pdf**

You will get points for:
- your implementation + meaningful comments
- answers of theoretical questions
- quality of output maps

<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.

General Task



Input Output

In the first exercise you will separate a specific foreground object from a given foreground video based on spatio-temporal Cost-Volume filtering. This video contains 43 frames which are included in the framework in the directory */fg_frames* as mentioned above. Within this task you have to detect which pixels in each frame of the video belong to the person in the middle of the video, in order to paste it - in the third exercise - in front of another background. You can perform the segmentation by following seven tasks (a.-g.):

Task a: Filter user scribbles to indicate foreground and background

Implementation in: get_histograms.m, exercise1.m (call function *"get_histograms")*
Output of this task: scribbled foreground and background pixels in reference frame

First you have to define which areas belong to the foreground and which to the background. This is done by providing three additional frames in the directory */fg_frames*. The first frame is a copy of one of the existing 43 frames and should be a reference frame of the foreground object, which should represent as many different colors of the foreground object as possible. The other two frames are copies of this reference frame marked with scribbles. In the first scribbled frame you mark the foreground and in the second one the background - this can be done by any image manipulation software. The file with the foreground scribbles has to start with "s", the file with the background scribbles with "ss" and the filename of the reference frame with "r" to differentiate them from the real frames. The following images show an example of the three additional frames plus filename convention:

rframe-00001.png sframe-00001.png ssframe-00001.png

With the reference frame and the frame with the foreground scribbles you can now extract the marked foreground pixels. The resulting matrix should be a binary map, where one means there was a scribble and zero means there was no scribble. In the same way you indicate the marked background pixels with the background scribbles.

Note: The process of defining the scribbles is very important for the quality of your solution. Try various scribble variations until you get the best result. As you can see in the picture on the previous page, there may be a few little holes in the foreground map because the background is structured and contains more than one color. This is ok as long as they are as small as seen on the output picture on the previous page.

Task b: Generate color models for foreground and background

Implementation in: get_histograms.m, exercise1.m (call function *"get_histograms")*
Output of this task: foreground and background histograms

With the information which pixels are marked you can now determine which color these pixels have. With the resulting pixel colors we now want to discover which colors are often used. For this purpose the framework provides a function called "*colHist*" that can be used. It returns a histogram of all red, green and blue values of one image. If you take a look at this function you can see how a specific red-green-blue combination is represented in the histogram.

For instance, the number of all black pixels (RGB value [0 0 0]) will be in the first bin, and the number of all white pixels (RGB value [255 255 255]) in the last bin. Take a look at the PDF-file "binned histogram" on TUWEL to get a better understanding of this topic.

Task c: Generate Cost-Volume

Implementation in: segmentation.m, exercise1.m (call function *"segmentation")*
Output of this task: Cost-Volume Matrix

In the next step we want to allocate the explicit red, green and blue values of the frames of our video to their corresponding probability values stored in the histograms that have been calculated before. This is done by pre-computing the related bin-ID of the color values and looking up the probability of this specific ID in the fore- and background histogram.

With these values it is now possible to generate a Cost-Volume. This means you are able to calculate certain costs of each pixel. These costs indicate if a pixel belongs to the background or to the foreground and are based on the comparison of the corresponding frequencies in the two normalized histograms:

$$cost = \frac{H_{foreground}}{(H_{foreground} + H_{background})}$$

$H_{foreground}$ …. normalized foreground histogram

$H_{background}$ ... normalized background histogram

Subsequently, after all costs have been set, we have to manually set the costs of pixels which don't occur in the for- or background respectively, because now their costs are "not-a-number". We can assign these pixels to the background and therefore set their costs to zero. This means a zero in the result stands for definitely background and a one for definitely foreground.

Task d: Extension of the guided image filter to a guided video filter

Implementation in: boxfilter_vid.m
Output of this task: guided video filter

Before you can use the guided filter in the next task as a video filter, you first have to adapt the already implemented 2D-filter in guidedfilter_vid_color.m to a guided video filter. To do this, you have to include not only the values over the Y- and X-axis, but also on the 3rd plane – the Z-axis. The number of frames that should be included is regulated by the parameter "rt" (practise has shown that a good value for "rt" on this foreground object is 1).

Task e: Filter Cost-Volume with guided video filter

Implementation in: segmentation.m
Output of this task: binary image map of filtered Cost-Volume Matrix

Using the Cost-Volume from task c for extracting the foreground using a threshold (assign for example all pixels to foreground, if costs > 0.5) would result in a segmentation that is spatially and temporally not coherent. This is primarily due to color ambiguities, missing colors in the color models or noise. To account for these problems, a smoothness assumption has to be implemented to propagate costs to neighbour pixels that are similar in terms of color. Therefore you have to reshape the linear Cost-Volume to a 3-dimensional-matrix and call the function "*guidedfilter_vid_color*" with suitable parameters. After this we can extract the foreground by using a threshold and can execute the other operations on the resulting binary image.

Note: A good default value for the threshold is 0.5. Try your solution with this value and modify it to check if your solutions get's better or worse. Also try to modify the parameters for the guided filter. For example, start with r=20, rt=1, eps=0.00001 and check your results with other values. Also test them with other scribble variations until you get the best result.

Task f: Delete regions which are not connected to foreground scribble

Implementation in: segmentation.m
Output of this task: binary map of foreground connected with scribbles

Another problem is still not solved: Regions which have the same color as our foreground object, are still counted to the foreground. To get rid of these sections, we remove regions that are disconnected from the scribbles by calling the function "*keepConnected*".

Task g: Guided feathering

Implementation in: segmentation.m
Output of this task: smooth foreground map

After eliminating unconnected regions there is still one last challenge. The hard cuts at the object borders should be replaced by a smooth transition. It has been shown that the spatial guided image filter approximates a matting algorithm when being applied to binary segmentations. So we have to apply the spatio-temporal guided filter a second time to the previously obtained binary segmentation volume.

Note: Again, try to modify the filter parameters (especially the filter radius r) to remove regions with noise or errors in it.

Introduction to MATLAB image representation

Matlab manages the images transposed, which means that the image point (x,y) corresponds to the value (y,x) in the image structure. Each image 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 to access the color information of the images:

image1(4,3,1) → x = 3, y = 4, c = 1 → output: red color value at position (3,4); e.g. 98
image1(4,3,1:3) → x = 3, y = 4, c = 1:3 → output: all three RGB values at position (3,4); e.g. [98 12 234]

Theoretical Questions

Please answer each question as precisely and accurately as possible (2-3 sentences)!

a. Explain the overall concept of Guided filtering and name an example beyond smoothing images.

b. If your foreground object is a black circle with a white square in the middle, does it matter if the white square is marked by the scribbles so that it is recognized as part of the foreground, or does the algorithm assign it to the foreground because it is surrounded by the rest of the foreground object (which is marked by scribbles)?

c. What assumption is used at Cost-Volume filtering for segmentation, and why?

d. When you calculate the Cost-Volume for a background, what does it mean when a certain pixels has high costs?