Lab 2 – Image Processing

Electrical measurement systems– ELA212

You are allowed to work in groups of 2. Prepare a **short** report answering the bold-faced questions below, complemented by code snippets (not in image format) and images from your MATLAB script. Assign yourselves to a lab group before uploading the report through Canvas.

1. MATLAB basics: read, write and display an image
    a. Load, show and save `img/bob.JPG` using MATLAB commands `imread, imshow, imwrite`.
    Note: MATLAB has good documentation. To find out more about a function type `help <function_name>` in the command window alternatively enter the function name in the search field in the upper right corner.
2. Image representation, intensity and RGB
The variable into which the image was loaded is an MxNx3 matrix, where M is the number of rows and N the number of columns. For `bob.JPG` the numbers are 1920x2580x3. This can be read out from the MATLAB Workspace window where all currently defined variables of the workspace are listed. The third dimension represents color, in this case 1=Red, 2=Green and 3=Blue (RGB). From the workspace it can also be noted that the data type is `unit8`, i.e. values are in the interval [0,255]. The value represents intensity where 0 is dark and 255 bright. Another common datatype is `double` with a data range [0.0, 1.0].
    a. Show and inspect the separate R, G and B color channels, for example I_R = I(:,:,1), and see how they correlate to the original color image. Each color-channel can be seen as an intensity image (grayscale). Compare the intensity of the green bag, Bob's blue trousers and the red box and wall between the different channels. The command `subplot` can be used to show several images in one figure. The command `figure` will spawn a new figure window.
    b. Convert the color image into intensity image (grayscale). How does it compare to the intensity images of the separate color channels?
3. Threshold and color space
    a. Threshold the intensity image. The result should be what is referred to as a binary or logical image (black and white). This is also sometimes referred to as a binary mask. Display the resulting mask.
    b. Use the binary mask to set pixels below the threshold in the intensity image to 0. Display the resulting image. Try different threshold values to study the different results. You have now created a basic high-pass filter.
    c. Now for the color image, similarly to before, create a filter that removes all but green pixels.
        i. **How did you create the filter? Did you get a good result, i.e. like the middle image of Figure 1? Why/why not? (Possibly there is a lot of white pixels – do not worry).**
        ii. **Suggest how to improve the result.**
    d. Start the color thresholder app (APPS tab) and load the Bob image, either directly from a workspace variable or file. You are now presented with the image in 4 different color spaces. Focus on RGB and HSV. Rotate the cubes and inspect the color separation in the 3D representation. Select HSV to continue to the next view to create and export the filter.

        i. **How do you create the green filter in this color space? Which parameter was the most important?**
        ii. **How does your filtered image compare to the previous RGB-based result?**

    e. Change the color of the green bag to blue, similarly to the color changed (rightmost) image of Figure 1.
        i. **How did you do this?**
        ii. **Show the resulting image**

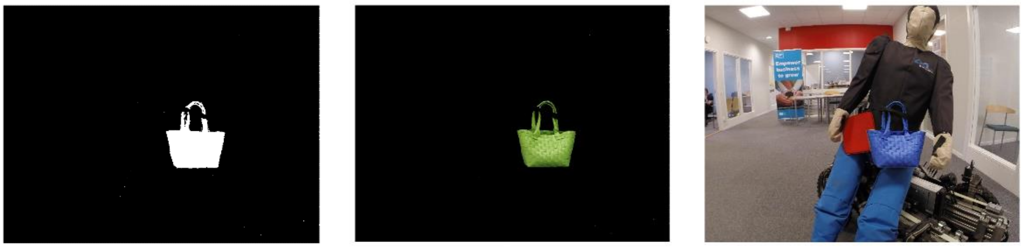

Figure 1. Examples: Filter mask, filtered image, colour changed

4. Spatial operator and convolution.
    a. Load the image img/calendar_speckle10.JPG. Note: convert the input image to `double` using `im2double`.
    b. Use the Sobel filter to create an edge image. Useful functions: `conv2(Image,kernel)` to perform convolution, `fspecial` to generate filters.
    c. The result is noisy – combine sobel with gaussian smoothing to reduce noise.
    d. Convolution is associative, compare A=(I*s)*g and B=I*(s*g)
        i. **Do they produce the same result?**
        ii. **Which one is quicker and why? Use tic, toc. Note: As the image is small loop the operation to get a better time estimation.**

```
tic
for i=1:1000
    perform the convolution
end
toc
```

5. Calibration and measurement.
    a. Estimate the camera parameters using the MATLAB camera calibrator app (APPS tab) and calibration images in img/calib. Checkerboard square size: 20mm. Use 3 coefficients for radial distortion. Export the parameters to your workspace and examine the content of the struct. Have a look at the intrinsic matrix. Does it look familiar? Also note the focal length for the following depth estimation. You may average the focal lengths in x and y direction.
    b. Images pen1.JPG and pen2.JPG there is a pen. Note the (barrel) distortion where straight lines are bent around the optical center. Measure the length of the pen in the images (number of pixels). You can use the data cursor in the MATLAB figure window. Given the actual length of the pen = 140mm, estimate the depth to the pen in both images using a pinhole camera model.
    c. Undistort the pen images using `undistortImage` with the estimated parameters
        i. You can also turn on full view to get a better understanding of the image warping: `undistortImage(I,cameraParams,'OutputView','full');`
        ii. Note that straight lines are now appear straight.

d. Repeat the depth estimation using undistorted images.

e. **What was the result of your 4 depth estimations? What do you conclude from these with respect to measurement and distortion?**

f. Bonus assignment: Create a filter to automatically detect the length of the pen. Use your knowledge from section 3. The function `regionprops` may be of use.