[redacted]

Lab 2 – subjective assessment / documentation

GUI

There are 3 functions in the file : detectLeaf(path), main() and slideThreshold(path). The solution is contained in detectLeaf. SlideThreshold is my attempt at doing something with other functions, but failing; results are unimpressive, so I didn't include them (though you can run it to see what it does, by replacing detectLeaf with slideThreshold within main()).

The script has a simple GUI that allows the user to adjust certain parameters for each window separately, although the default settings will produce the desired result most of the time – manual management is for the trickier photos.

The GUI compromises of 6 windows, 3 containing mask, boxed image and image with the mask applied, refreshed in a loop to reapply new configurations of sliders from the 3 remaining windows:

'sliders', which contain (top to bottom)

- the lower threshold for cv2.threshold (upper set to 255),
- 'kernel size' (for blurring; note that kernel_size+1 will be read as it has to be >0),
- 'iterations' for the number of iterations in erosion and dillations (set to 0 to skip it),
- and 'use sobel' to apply the Sobel transform and blurring before thresholding.

'rgb start', containing the RGB values for lower colour threshold

'rgb end', containing the RGB values for upper colour threshold

By default, Sobel and 'iterations' are set to 0 and thus not used (while they greatly improve the quality when well-adjusted, on average I noticed them to degrade the performance).

You need to specify the directory holding .jpg's within main(). This directory will also be used to store the output – use a copy for testing.

After running the program you have keys that can be pressed:

ESCAPE, to exit the program without saving the current result

BACKSPACE, to continue to the next picture without saving the current result

SPACE, to save the current result into the same directory as the original data, with the filename suffixed with result type.

Note that if the sliders are set incorrectly and do not produce a contour (i.e. the area of the largest contour is equal to 0) the program will stop execution. When that happens, change the slider positions to more reasonable ones and press any key retry. Keep the console in view to see when that happens – a relevant message will be printed.

Algorithm (per image)

The code is commented.

Read the image in color and grayscale. In a loop do the following:

1. Shallow copy the both grayscale and color img for this iteration.

2. Apply inRange: find pixels that are within the specified color range, by default [35,35,25] to [80,200,255], and apply an AND to remove them from the grayscale version. This cleans up the image a bit and helps greatly. As far as impact on the solution goes this option has perhaps the most importance, and if chosen correctly can single-handedly solve the task. That said, it is time-consuming to find that threshold correctly, and by default only a rough approximation of 'green' scale is used.
3. (disabled by default) apply the Sobel transform to find the gradient in grayscale img, and blur it based on that – makes the subsequent thresholding more accurate. Blurring done with kernel size set by the user – 2 by default.
4. Apply thresholding. By default in range [3,255]; the user can manually change the lower threshold if necessary.
5. (disabled by default) apply a series of erosions and dilations IT number of times, IT=0 by default, set by the user.
6. Attempt to find the contours by the cv2.findContours function, and choose the contour with the largest area found. **Note that bad input can result in all contours being of size 0, as mentioned beforehand.** Require the user to change it if it happens, and re-run the loop after key input.
7. Fill the largest contour with white, to find the mask.
8. Bound the largest contour in a rectangle.
9. Apply the mask to the image with bitwise AND.
10. Wait 100ms for a key input (ESCAPE, BACKSPACE, or SPACE) and start again if no input detected.

Results

All outputs are included with the documentations in a file. 'processed_user_adjusted_2' contains results where each image had its sliders adjusted manually; 'processed_default_settings' contains results where images were processed in bulk without adjusting the sliders. 'processed_user_adjusted' is an older folder which contains results from an earlier version (discussed below).

Subjective assessment

While using pixel color threshold is the most powerful pre-gradient thresholding of the options used, it is also the most invasive and most error-sensitive. With that in mind when you spend that 20 seconds and narrow the sliders down as much as possible the results will be incredibly accurate, as visible in the processed_user_adjusted_2 folder; however, getting that range even slightly off will heavily impact the overall result, so in the default setting the range is very wide, [35,35,25] to [80,200,255]. The 'natural' green is notably rich in green and red, but rarely includes blue – hence the blue upper thresh can be aggressively lowered (to 80). The leaves in sample data are in four different hues: dark green, lime green, something in-between, and saturated green (reflections of light on green), hence the wide range.

If you play with the sliders and use the Sobel transform and blurring you will notice that, like with color threshold, it produces a less accurate outline but removes noise, and thus improves the results when well-adjusted, but ruins them when its settings are even slightly off. Sobel, blurring, erosions and dilations help to polish out the edges, fill in the gaps, remove false positives extending from the leaf (ex the cable on the first picture, a keyboard on the other, etc).

Note that in the ‘processed_user_adjusted’ a slightly different version of the script was used – there, the largest contour was not being coloured in, and thus sometimes black holes would appear in the middle and other masks would be present in different spots (and also random small pockets of noise wouldn’t be present). In that version the largest contour wasn’t being properly coloured in (thus leaving closed holes in the mask), and smaller patches of noise weren’t being filtered out. This has since been fixed in the script, but I had a lot of data in that folder that I wasn’t willing to simply delete, as it highlighted the issue.