`http://www.cs.ucf.edu/~bagci`

[Programming Assignment] (2)
Computer Vision

Dr. Ulas Bagci • (Fall) 2016 • University of Central Florida (UCF)

Coding Standard and General Requirements

Code for all programming assignments should be **well documented**. A working program with no comments will receive **only partial credit**. Documentation entails writing a description of each function/method, class/structure, as well as comments throughout the code to explain the program flow. Programming language for the assignment is **Python**. You can use standard python built-in IDLE, or CANOPY for the working environment. Other commonly used IDLEs are the following: PyCharm Community Edition, PyScripter, CodeSculptor, Eric Python, Eclipse plus PyDev.

Following libraries will be used extensively throughout the course:

- PIL (The Python Imaging Library), Matplotlib, NumPy, SciPy, LibSVM, OpenCV, VLFeat, python-graph.

If you use CANOPY, make sure that you use version 2.7, which already includes many libraries. If you are asked to implement "Gaussian Filtering", you are not allowed to use a Gaussian function from a known library, you need to implement it from scratch.

Submit by **15th of October 2016**, 11.59pm.

Optical Flow [8 pts]

[4 pts] Implement Lucas-Kanade optical flow estimation, and test it for the two-frame data sets provided in the webcourses: basketball, grove, and teddy.

[4 pts] Implement Lucas-Kanade optical flow estimation algorithm in a multi-resolution Gaussian pyramid framework. After experimentally optimize number of levels for Gaussian pyramid, local window size, and Gaussian width, use the same data sets (basketball, grove, and teddy) to find optical flows, visually compare your results with the previous step where you don't use Gaussian pyramid.

Evaluation: Please create some random colors to show velocity vectors on the original images, and save them as outcomes of your program. Do the same step for each data set. Visit `http://opencv-python-tutroals.readthedocs.org/en/latest/py_tutorials/py_video/py_lucas_kanade/py_lucas_kanade.html` for a sample implementation of Lucas-Kanade using Python and OpenCV. Other Python sources are available too. You are allowed to use built-in functions such as Gaussian smoothing, convolution, gradient operations, corner detections, etc., but not the Lucas-Kanade implementation itself. For features: you are suggested to use OpenCV's **goodFeaturesToTrack**, or your own implementation of corner detection.

Gaussian Mean-Shift Clustering [7 pts]

Implement mean-shift clustering algorithm for the segmentation of the following RGB image: "input3.jpg". Set the parameters (h_s: spatial resolution, h_r: range resolution) according to the properties of the attached image. Briefly mention optimality of why and how you decide those parameters along with the output. There is no ground truth available for this image as this question is intended to be a blind-competitions among attendants. Best segmentation method will be decided and announced based TP rate, FP rate, and F-score (use isotropic Gaussian kernel). In the pseudo-code, $p(n|x)$ is posterior probability, and x's update means difference of x's values between consecutive iterations.

Data: Input image with N data points (pixels)

Result: Output image segmented via Gaussian mean shift

for $n \in \{1, ..., N\}$ **do**

$\quad x \leftarrow x_n$;

$\quad$ **while** $\underline{\text{Update of } x\text{'s} > \text{tolerance}}$ **do**

$\quad\quad \forall n : p(n|x) \leftarrow \dfrac{exp(-0.5.\|(x-x_n)/\sigma^2\|^2)}{\sum_{m=1}^{N} exp(-0.5.\|(x-x_m)/\sigma^2\|^2)}$;

$\quad\quad x \leftarrow \sum_{n=1}^{N} p(n|x) x_n$;

$\quad$ **end**

$\quad z_n \leftarrow x$;

end

Use connected component in z_n to identify clusters (built-in function);

Algorithm 1: Pseudo-Code for Gaussian Mean Shift Clustering