Fast Prototyping Exercise 2

Exercises, 8, 9, 10
CSC872
Pattern Analysis and Machine Intelligence

https://bidal.sfsu.edu/~kazokada/csc872/Segmentation_Data.zip

Fast Prototyping Exercise

• Fast Prototyping
  – Learn how to do a quick proof of concept by building a prototype
  – Correctness matters (no sloppy algorithm!)
  – Speed matters (no beautification!)
  – No perfect SE necessary
  – No copying of codes online.
  – Parameterization/Visualization/Experimentation
    – Find out what are free parameters in your algorithm whose value must be hand-picked by you
    – Learn how to view internal variable’s current values
    – Learn how to visualize your prototype’s results in plots/images etc
    – Tweak the parameter values and study your prototype’s behavior to understand the how algorithm works

• Group Work
  – You are encouraged to freely exchange ideas and codes
  – Contributions to others are as valuable as making your own work
Fast Prototyping Exercise

- Please upload your matlab codes thru iLearn forum for my grading and your playing!
  - First two exercises: Due on midnight of the day (just what you did during the exercise)
  - Third last exercise: Due on midnight next day (complete version with some doc/screen shots of running the code)
- 5% extra credit for those who completed all three algorithms in-class

Platforms

- MATLAB
  - MathWorks: http://www.mathworks.com/
- MATLAB @ SFSU
  - https://at.sfsu.edu/at-mathworks-matlab
- Various tutorials available online
Public Libraries

- OpenCV (Computer Vision)

- ITK (Medical Imaging)
  - http://www.itk.org/

- WEKA (Machine Learning)

Segmentation

- Image Segmentation
  - Label pixels according to the image intensity such that pixels with similar intensity have same label

  1) Intensity-based Features: use only proximity in intensities
     - Pixels placed far away can be grouped together due to similar value

  2) Spatio-intensity-based: Features use both space and intensity proximity
     - Segment a connected-components with similar intensity values!
Segmentation cond.

- Segmentation is a labeling process
- Edge-preserved smoothing
- Density-based smoothing

- Grouping of Modes

---

Paper 2

- D. Comaniciu, P. Meer,
- http://comaniciu.net
Data

- I provide a set of nine test images
  - [https://bidal.sfsu.edu/~kazokada/csc872/Segmentation_Data.zip](https://bidal.sfsu.edu/~kazokada/csc872/Segmentation_Data.zip)
- 3 Color images
  - Baboon, Lena, Pepper
  - A set of pixels with a 3D 8bit (0-255) RGB feature
  - Feature space is a 3D histogram of RBG colors (Color space) or 5D RBG-Space feature
- 6 Grayscale images
  - Baboon, Lena, Pepper, Barbara, Cameraman, Goldhill
  - A set of pixels with a 1D 8bit (0-255) feature
  - Feature space is a 1D histogram of intensity values or 3D intensity-Space feature

Mean shift

- “Conceptual” Steps
  1) Do KDE on $x_1, \ldots, x_N$ for $p(x)$
  2) Do Clustering of $x_1, \ldots, x_N$ according to the estimated $p(x)$
  3) Re-label each $x_i$ by its cluster center value

- Mean Shift
  - Adaptive step-size gradient-ascent in a feature space $x$
  - Convergent to nearest mode/peak $x^{mle}$
  - **No need for explicitly computing a density estimate!!!**
  - Bandwidth parameter must be hand-picked though
Algorithm

Vector Norm: \( \|x\| = \sqrt{x_1^2 + \cdots + x_d^2} \)

- Suppose we are given \( N \) samples \( x_1, \ldots, x_n, \ldots, x_N \)
- And we model \( p(x) \) by KDE with bandwidth \( h \)
- **Mean Shift Vector** defined at arbitrary location \( x \)
  - Compute arithmetic mean of the samples with a weight function \( g \)
    \[
    m(x; h) = \frac{\sum_{n=1}^{N} x_n g \left( \frac{\|x - x_n\|^2}{h} \right)}{\sum_{n=1}^{N} g \left( \frac{\|x - x_n\|^2}{h} \right)} - x
    \]
- With Epanechnikov Kernel, you get
  - We can simplify the above MS because you get a constant weight function
    \[
    g \left( \frac{\|x - x_n\|^2}{h} \right) = \begin{cases} C & \|x - x_n\| \leq h \\ 0 & \text{otherwise} \end{cases}
    \]
- With (isotropic) Gaussian Kernel
  - We have smooth KDE \( p(x) \) so we expect better behavior
    \[
    g \left( \frac{\|x - x_n\|^2}{h} \right) = \exp \left( -\frac{\|x - x_n\|^2}{h^2} \right)
    \]

Algorithm Cond.

- **Mean Shift Procedure**
  - Given \( N \) samples \( x_1, \ldots, x_p, \ldots, x_N \)
  - Iteratively compute \( y_1, \ldots, y_k, \ldots, y_K \) \( \rightarrow \) \( y_{\text{mle}} \) (until convergence)
    \[
    y_1 \leftarrow x_{\text{init}} \quad \text{This loops at each pixel}
    \]
    loop over \( k \)
    \[
    y_{k+1} = m(y_k, h) + y_k = \frac{\sum_{n=1}^{N} x_n g \left( \frac{\|y_k - x_n\|^2}{h} \right)}{\sum_{n=1}^{N} g \left( \frac{\|y_k - x_n\|^2}{h} \right)}
    \]
    \[
    y_k \leftarrow y_{k+1}
    \]
- **Stopping Criteria**
  \[
  \left\| \frac{m(x; h)}{h} \right\|^2 \leq TH^2
  \]
  This sums over the sample set
Hints

- First try grayscale image then color image next if you can
- Try small image size like 64 by 64 (should take about 1 min)
- How to make an output image by doing MS clustering?
  - Define a new image $J$ whose size is the same as the input $I$
  - For each pixel of the input image $I(x,y)$,
    - Initialize the iterator variable $y$ by the intensity of the pixel $y_1 = x_{	ext{init}} = I(x,y)$
    - Do the mean shift procedure shown in the previous slide $y_i \rightarrow y_{\text{mle}}^k$
    - Set the corresponding intensity value of the output image $J(x,y) = y_{\text{mle}}^k$
  - This is known as Mean Shift Filtering
- Free parameters to be hand-picked
  - Bandwidth $h$
  - Stop threshold $TH$
  - Max iteration $K$
- Think of how to group the convergence points?
- Think how to visualize the density and each mean shift step
- Think how to extend to color image

Useful MATLAB Codes

For Mean Shift
- vec = Matrix(:) colon operator to vectorize a matrix
- val = exp(), exponential function
- M = double(M), casting the data type to double
- figure, display a figure window
- Imagesc(IMG), display a matrix as an image (scaling the values to 8 bit range)