Spring 2020 - CS 477/577 - Introduction to Computer Vision

Assignment Ten

Due: 11:59pm (*) Wednesday, April 30.

(*) There is grace until 8am the next morning, as the instructor will not grade assignment before then. However, once the instructor starts grading assignments, no more assignments will be accepted.

Weight: Approximately 5 BONUS points

This assignment is OPTIONAL. You can hand it in to effectively make up for a previous missing assignment, or one you did not do well on, or even for bonus (but recall that the assignment grade will be capped at 75/70).

If you choose to take a break from handing in assignments, I suggest that you read it to make sure you know how you would proceed.

This assignment has minimal reporting requirements

This assignment can be done in groups of 2-3. You should create one PDF for the entire group, but everyone should hand in a copy of the PDF to help me keep track. Make it clear on the first page of your PDF who your group is, and a brief explanation of how you worked together. If the group consists of both graduate and undergraduate students, make it clear to what degree undergraduates contributed to any graduate parts or extra parts. Group reports are expected to be higher quality than individual ones.

General instructions

You can use any language you like for this assignment, but unless you feel strongly about it, you might consider continuing with Matlab.

You need to create a PDF document that provides needed meta information such as who was in your group, where you substantively deviated from the specification / suggests, conclusions that are not obvious from the images, etc. However, unlike previous assignments, you do not need to “write up” the assignment. You only need to provide image results. Captions are not required. While learning to write good reports is very important, we have done a lot of it so far, and this assignment supports a simpler format (mostly images).

Don’t forget to hand in your code as well.

Learning goals

- Learn the K-means algorithm as a basic objective function based clustering method
- Understand segmentation as clustering
- Understand the effect of features such as color, location, and texture
- Learn about the texton representation of texture (grads)

- Exercise the concepts of developing methods and associated constants on a training data set, and then applying that to a testing data set (grads).

Part A (undergrads and grads)

This part is mostly about clustering and segmenting images using clustering. Some of you might have implemented clustering using K-means, or the slightly more sophisticated method of using EM to fit a GMM, in some other course. You are welcome to reuse parts of such code, provided you wrote it yourself. While there are many implementations of K-means (and EM+GMM) available online (even in Matlab!) it is instructive to implement this kind of clustering at least once. But having done so, you do not need to do it again, unless you want to.

1. Implement K-means clustering. You should make K easy to set. At every iteration you must print out the value of the objective function (which should go down). Recall that the objective function is the sum of the squared errors from the mean of the assigned cluster. Use your implementation to segment the following three images based on color taken as a 3D feature vector, using k=5 and k=10 (also on D2L).
   http://kobus.ca/teaching/cs477/data/sunset.tiff.
   http://kobus.ca/teaching/cs477/data/tiger-1.tiff.
   http://kobus.ca/teaching/cs477/data/tiger-2.tiff.

   For visualizing the segmentations, set each segment to its mean color. You could spice it up adding region borders. One way to do so is to make any pixel that is **not** surrounded by pixels from the same cluster a distinctive color, perhaps adding its neighbors in one pass to make the border thicker.

   Note for subsequent parts. This is a reasonable way to visualize your segments, even if you did not use to color create the segments. For example, a segmentation based only on texture could be visualized this way, although there, using only the borders might be better as you can then see the texture within the region.

   Hint for debugging. Monitor the object function value. It must go down monotonically.

   Hand in code to generate the segmentations (6 total), and put the original and segmented images into your PDF "report" (**$**).

2. Add $\lambda*X$ and $\lambda*Y$ to your feature vector, where (X,Y) are the coordinates of the pixel, and λ is constant that you need to play with. Thus you now have five dimensional feature vector $(R,G,B,\lambda*X,\lambda*Y)$. If λ is zero, then you should get the same answer as for the previous question. For each image, provide two images that show the effect of increasing λ, which combined with the implicit result from question 1 for $\lambda=0$, tells a visual story (**$**). Assuming your R,G,B values are between 0 and 1, you could start with $\lambda=1$ and $\lambda=10$. You should be able to explain the results to yourself. You may have to push λ higher to have a significant effect.

3. Construct some texture features from a black and white version of the above three images. To get texture features, use your code from HW8 to find horizontal and vertical edges at two blurring sigmas (2 and 4). Consider a window size, W, centered on each image pixel in turn. Compute the mean squared response of some filters over the pixels of W, for some sigmas. This will result in a feature vector that captures some notion of texture for the location of W.

   Explore **(a)** using this feature vector alone; **(b)** together with the full color (RGB), and **(c)** with the spatial location feature in the previous question, for some choice for W. You do not need to provide images for more than one choice of W, but make it easy to experiment with so you can provide results for a choice of W that you like (I do not have a good suggestion at this point, but I would start with 30). Provide results for the three images for the three different feature vectors just described **($)**. If the three images provided do not behave interestingly enough, try some other ones!

Don't forget that K-means is sensitive to the range of your data, and so you will want to scale your features accordingly.

For more interesting results, you may want to apply the scaling trick to the color features (e.g., multiply the color features by some constant to tune the degree that color is used (much like tuning k above to tune the degree that spatial adjacency is used).

Part B (grad students, ugrads are eligible for a bonus)

4. Create a texton representation of texture from a modest size data base of images. If you have some idea of what kind of images you would like to use (perhaps related to your research, or perhaps the kind of images you like), by all means create/provide your own training and test set, but tell the TA about your choice. Otherwise, following a long tradition of texture in images work, let's use patterned animals, specifically tigers, as available here (also on D2L):
   http://kobus.ca/teaching/cs477/data/tigers_small.tar.
   If you find that more images might be helpful, you can use
   http://kobus.ca/teaching/cs477/data/tigers.tar.

   To create your textons you first have to get filter bank responses, which means you have to think about creating a filter bank, and then applying it. You can use functions provided by others to do so if you like. I suggest looking into the Matlab functions `gabor`() and `imgaborfilt`(). Having got your filter bank representations of your training images, you can use K-means for clustering to get textons. You will have to choose K via some combination of intuition and experimentation, where success might be defined based on the task below. I would start with K=50, but this is just a guess. Note that doing this might be slow on Matlab, so you may need to consider every tenth point or some other strategy to reduce computation.

   Having found your textons, the characterization of texture in your test image at a given point is the histogram of textons in a window W around the point. We could use this characterization to segment images based on texture, but instead, let us try to demonstrate how well things are working using a process that is analogous to finding a good texture window match in a database. Here, create a "database" of texture patch representations by dividing the training images into W sized non-overlapping blocks (ignore the "leftover" parts of the image due to that fact the number of rows and columns will not necessarily be a multiple of W). For each such patch create its histogram, and an index into the image, and the window within the image. Now, consider a test image. For each of its non-overlapping patches similarly defined, replace it by the best patch from the data base. Use the chi-squared measure to define "best". (You will need to deal with some bins both being zero in some way, such as ignoring them.) Provide the before and after image pairs for three images from the test set.

   Note. I do not know how well this will work, but I am curious to find out!

What to Hand In

As PDF document that provides needed meta information such as who was in your group, where you substantively deviated from the specification / suggests, etc., and the images asked for. Please help the TA out, by checking that the order of the images is the order that they were asked for. Also, hand in your code that creates those images.