Computer Vision - Cartoonization

You are responsible for understanding everything that you submit. Read through the entire homework.

How to submit your homework:

A. Create a directory named **HWNN_Lastname_Firstname**, (with the obvious substitutions).
B. Put everything in that directory:
   a. Your original input image, which should be about 640x480 or so in size.
   b. Your output image cartoon, in a *.png format.
   c. Your Matlab code, and
   d. Your write-up in PDF,
      which should include a copy of your output image results.
C. Zip up the entire directory, and submit it to the dropbox.

D. Please be sure you zip up the directory, and not just the files in the directory.
Test your by submission by unzipping it to a new directory. We've had some that do not, and they are difficult to grade.

The goal of a good cartoonization algorithm is to simulate what you would do if you were drawing a cartoon. It should preserve region details well enough that you can see what the subject is. However, it should not provide too many details.
This type of process is sometimes used as a pre-processing step to pre-segment an image before other processing.

Overview of Parts:
Using rgb2ind, in RGB with different values of K.
rgb2ind( ) returns a colormap. You will need to use this colormap when displaying images.

Then you will use kMeans explicitly, with a set list of pixels and their attributes.

Segmentation using kMeans:

Main Routine:
As before, create one file called HWNN_Lastname_Firstname, (with the obvious changes), and which sets the Matlab path to include 'TEST_IMAGES', and '../TEST_IMAGES', and ../../TEST_IMAGES, which runs all of the sub-parts.

In this homework, you will use some images provided, and a small colorful image of yourself that you provide.

1) **(1 pt) HWNN_ part1_Changing_K_only.m**
   Using rgb2ind, in RGB with different values of K, **be sure to set the option 'nodither'** for your work with rgb2ind() for this entire homework. rgb2ind( ) uses kMeans under the covers.

   rgb2ind(… 'nodither') returns a colormap. See the examples in the documentation.

   You need to use this colormap when displaying the image if you want it to look good.

   Consider the two images BALLS_FIVE_7537_shrunk* and HW_*_MacBeth_Regular*.
   Try various values of K from 5 to 256, with rgb2ind(… 'nodither').
   You don't need to try them all, you can skip values.

   - What was the smallest value of K you found acceptable for the BALLS_FIVE_*.jpg image?
   - What was the smallest value of K you found acceptable for the HW_08_MacBeth_Regular* image?
   - What value of K did you like best for the MACBETH_*.jpg image?

   One might expect that the MACBETH image will look better with very few colors because there are more regions and they are weirdly shaped. Or, one might expect exactly the opposite. What did you expect? What did you find? Which statement is wrong? Is there one value of K that makes a great difference between the BALLS and the MACBETH_* image?

   Include two resulting images in your *.pdf write-up for discussion.

2) **Using kmeans explicitly:**

a) **(1 pt) HWNN_part3a_DistanceWts.m:**
Remember, you have sample code to get you started on this.

Using k of 64, consider the Corel test image Corel_Image_198023*
Look at the documentation for the options for kmeans( ).
Try different parameters and different attributes to get the best clustering – segmentation that does not cross the edges.
Use the attributes { column, row, red, green, blue } for every pixel in the entire image.
You might sub-sample (resize) the input images by a reasonable amount for speed.

The values of column and row come from meshgrid( ). (See documentation.)

You want to tell kmeans( ) to return the cluster centers it found.
Then get the colors out of those cluster centers.

What distance weights did you find worked best for this image?
What color space worked best for you? rgb? rgb2hsv? rgb2lab? rgb2ycbcr?
Show your final two images in your *.pdf. Describe your findings.

b) **(1 pt) HWNN_part3a_ Euclidean_vs_CityBlock.m:**
Using k from 40 to 60, consider the images HW_08_MacBeth_Rotated* and HW_08_MacBeth_Regular*.
We seek the smallest value of k, for which there is a strong difference between the two distance metrics:

- SquaredEuclidean (or Euclidean), and
- CityBlockExplicitly use kmeans( ) with the same set of attributes and weights for both runs.
  Try using the cityblock distance versus the Euclidean distance.

  What do you notice about the different distance metrics?
  Show an example of the difference between SquaredEuclidean and CityBlock.

3) **(2 pt) HWNN_part4_portrait.m:**
Using what you learned, cartoonize your own portrait. It should be around 640 to 480 pixels in size, or smaller.
You will use different values for this part.

4) **(3 pt)** Once you have a good image cartoon, find the edges and add them back onto the image, but in black.
In other words, find only the edge pixels, and set them to black in the image.

Show your resulting cartoon.

5) **(2 pt) Conclusions:**
After you have written what you saw and decided in each of these sections, also write-up your over-all conclusions in a conclusions section. Write about three paragraphs summarizing your learning.

Describe your cartoonization solution. Did you do any noise removal? What parameters did you use?
Did you pre-process your image using any pre-processing?

Be sure to write good conclusions, which stand by themselves.
It is just fine if the conclusions repeat what you said earlier.
Use paragraphs, please!!

Then, just for completeness, send a copy of your cartoon to your parents. Parents like to see these things.