ARTcode experiment

In this project, we will take you onto the journey of learning and understanding ARTcode. ARTcode was one of our research projects. The goal of ARTcode is to create a code that serves as a barcode and can help users identify the contents of the barcode at the same time. The detailed information of ARTcode can be found at this address: http://dl.acm.org/citation.cfm?id=2971733 . You can click on "PDF" and obtain an explanation of ARTcode.

The prerequisite of this project includes K-means clustering and MATLAB programming. If you are not familiar with MATLAB, now is the time for you to grasp it.

Before we make any move, firstly we would like to give you a brief introduction on our ARTcode implementation MATLAB programs.

In ARTcode experiment folder, there are a number of MATLAB files, some data files, and a folder. The folder contains some testing images that you can work on. Of course you can put in your own images. Just don't forget to adjust the image size (128*128)!

The MATLAB files in ARTcode experiment folder generate an ARTcode. The main function is *encodeMain.m*. Don't panic, important information are provided in comments. Oh, another tip, you might want to read the questions first before you try to understand the MATLAB functions.

Are you ready for this journey? Let's get started!

1. In line 20 of *encodeMain.m*, there is a function *getpalette()*. However, you might already notice from the folder that this function is missing. It is for you to complete this function! But don't worry, you have a reference. You can refer to our paper. The corresponding part is in page 906 -> "Colored Dot Matrix and Shuffling" -> "Colored Dot Matrix Generation." -> "Phase 1. Color palette selection by clustering". All inputs and outputs of *getpalette()* are listed in our paper. To give you a better understanding, we specify here a rough outline of this function.

   This function has two or three inputs, namely, *picname*, *targetcolornum*, and *coor*. *picname* refers to the picture we are using and is a string recording the address of the image. *targetcolornum* refers to the number of colors in a color palette (number n in our paper). *coor* is an optional input. When *coor* is specified, the clustering algorithm should run only on the pixels determined by *coor*.

   The outputs of this function are *colors* and *power* (I know what I am saying, MATLAB allows multiple outputs). *colors* means the clustered colors for colored dot matrix generation in RGB (a n*3 uint8 matrix), and *power* means the ratio of corresponding color in the result image.

So *getpalette()* takes an image as input, and finds the *targetcolornum* colors that can best represent the image using K-means clustering (a kind tip: MATLAB has a build-in function of K-means).

Now you should know how to implement this function.

2. The next missing function is *data_hiding()* in line 37. This function embeds data into a binary vector. It corresponds to page 907 -> "Embedding Data with Minor Modifications". All you need to implement the function *data_hiding()* is provided there, for you to explore, so we won't waste the space explaining them.

   Still we would like to remind you that there are three inputs to this function. *image_serial* is a binary vector into which we embed data. *info* is the data we would like to embed into *image_serial. k* is embedding block size (in the paper it is denoted by *l*). The output of this function, *changed_serial*, is the embedded binary vector.

   Now it is your turn to complete ARTcode!

3. When you complete the above two functions, you launch *encodeMain.m* and expect to obtain an ARTcode. However, sometimes you may encounter an error in MATLAB (if this error does not appear, please run *encodeMain.m* for more times on all testing images until the error appears). The error information goes like,

```
Error in find2Color (line 3)
[color_num, ~] = size(codingcolors);

Output argument "color_0" (and maybe others) not assigned during call
to "E:\New folder\ARTcode-final code\find2Color.m>find2Color".

Error in generateCode (line 13)
        [color_0, color_1] = find2Color(cur, codingcolors);

Error in encodeMain (line 44)
        img_embedded = generateCode(dithered_img, img_colors,
        codingcolors, changed_pos, changed_serial_CRC, coor,
    shuffle_size);
```

   Well, as we said, this error may appear, and it may not appear. When the error doesn't appear, you can get an ARTcode. When the error appears, you cannot obtain an ARTcode. Why should there be an error like this? Why there is sometimes an error and in other times there is not? We can guarantee that it is not due to your RP (Renpin). So it is for you to answer the two questions.