Learning Objectives:

- Learn how to generate RGB image from raw image data
- Learn how different interpolation methods can be implemented.
- Learn how to analyze the outcome of different interpolation methods.
- Improve your skills in writing technical reports.

Organization and Tasks

The aim of this work is to implement three interpolation methods to create full RGB image from raw Bayes data. Even this is a laboratory project, most of the work can be done with any Matlab equipped modern computer. Group size can be one or two students. The project must be done during this academic year, which means that the project must be returned by latest at **Friday 1.5.2014 at 23:59**.

The basic knowledge of Matlab and programming is mandatory to succeed in this work. However, guidance for the work will be provided during the project according to the requests. If you have questions, you are welcome to visit in TE316 or send an email.

Overall the work can be divided in four tasks:

I. Laboratory session in TE407 (10%)
II. Implementation of the interpolation methods (50%)
III. Testing the outcome of the methods (20%)
IV. Documentation (20%)

Laboratory session

You will have to reserve a session time (see [1]) for your group via email from the assistant. The session takes 20-60 minutes, depending on the number of questions and on the extent of discussion. The purpose of the session is to familiarize you with:

a) additional information related to the topic (the methods) and
b) the data that you are going to process.

If something seems to be missing from this document, then we shall discuss that also in detail during the lab session. The used data will be distributed to you during the session, so take an USB memory with you to the session.

You will find the free laboratory times from the calendar [1]. The laboratory session will have a short discussion about the theory behind the work, the requirements of the work and the suggested working styles. The session is the best place to ask the questions about the work itself.

Before session familiarize yourself with the assignment. If you do not understand something, you should ask it during the laboratory session.

Raw data processing

To use the data provided in the laboratory session you need to implement the following algorithm as a function:

1. Open the file with fopen.
2. Read the data with fread function to some variable using the three inputs for fread.
3. Save the color data to the R, G and B layer matrixes from the read data. The original data is in the corresponding format as visualized in Figure 1. The final R, G and B matrixes should look like ones visualized in Figure 2.
4. The function call should be as: 
   \[
   [R,G,B] = \text{readimagefile}(\text{filename, iMsize, iMtype})
   \]

The image size for raw_imageX is [1008 1018] and data type to read is int16, but the values are 10bit little endian. The image size for testikuva.raw is [512 512] and the data type is uint8 with 8 bit little endian values. For this reason, the values the raw_imageX must be divided with $2^{10}-1$ and the testikuva with $2^{8}-1$ to normalize them.
Theory

Most of the digital cameras saves the data in RGB format, where all pixel values are formed with a color vector of red (R), green (G) and blue (B) components. These components are measured with the camera sensor by separated color cells. Single cell (pixel) of the sensor measures only one intensity at a time.

To measure three main colors a separated color filter arrays (CFA) are used with the sensor. For each cell (pixel) there will be a single color filter. The most common CFA is the Bayer filter, where red, green and blue filters are in group of four cells with two green and single red and blue cells (see raw data visualized at Figure 1 and 2). The green color is measured with two cells instead of one as the human eye is more sensitive to green color. The amount of cells in the sensor defines the amount of pixels in the output image. Thus in the raw data there is only one color value for each pixel. For example in 16 Mpix camera sensor the resolution can be $4928 \times 3264$ pixels.

Figure 1: Raw data from Bayesian CFA visualized without interpolation

Figure 2: Separated color data with the corresponding CFA
To be able to create a full RGB image from the sensor data the two missing color values of each pixel must be interpolated from surrounding pixels. Commonly the interpolation is done in 3x3 window over the pixel. Multiple interpolation methods exists for this task and the interpolation has significant affect to the output quality. For example, Figure 3 presents the results of nearest neighborhood and bilinear interpolation methods. In common compact cameras the interpolation is often implemented to the hardware and the camera saves the RGB-images only in some common image format like jpeg. In industry, the interpolation procedures are commonly implemented with software outside the camera. The DSLR and more advanced compact cameras offers normally the raw data format, which enables the possibility to external raw image data processing.

1. Nearest Neighborhood interpolation

The nearest neighborhood (NN) -interpolation is the simplest interpolation method. It has two advances over others: it is simple to implement and it is computationally fast. As a negative side the quality of the result is lesser than with more advanced methods.

Figure 4 presents the Bayer matrix before interpolation process. The pixels in the Figure 4 are numbered from 1-16 and named according to the Bayer filter. At the starting point only one value (red, green or blue) is known from each pixel. After interpolation each pixel should have all three RGB-values. In the following the pixels are named according to number and corresponding color, for example B1 is the blue value of pixel in the location 1.

In principle NN-interpolation is performed in 2x2 windows: The blue values in the corresponding locations of G2, G5 and R6 are defined by copying them from the pixel B1. The Red values for locations B1, G2 and G5 are formed similarly by copying them from R6. The Green pixel in B1 is copied from G2 and for pixel R6 the green value is copied from G5. The similar process is repeated in all 2x2 windows in the image. After the process, there exists three color layers which all have the same size as the original raw image. By combining these layers to one NxMx3 RGB matrix we have a full RGB color image.

2. Bilinear Interpolation

The Bilinear interpolation is one step more complex than NN -interpolation. In bilinear interpolation, pixel values are computed from surrounding pixels. This prevents the pixel artifacts effect, which is clearly visible in the NN-interpolation. In Bilinear interpolation the border pixels does not have all necessary pixels, which must be dealt somehow. One option is to forgot the border pixels, which creates smaller image to output, second option is to add zeros to surroundings of the image and so fill the missing pixels, third option is to mirror them from the border pixels and then use these values to interpolation. Here you can select the method freely by your own choice.

The bilinear interpolation procedure is following: in the case of Green pixels the value is always the average from surrounding four green pixels. For example G6 = (G2 + G5 + G7 + G10) / 4. The Blue and Red pixels are interpolated in similar manner as an average of nearest two or four pixels: With average of two pixels for example B2 = (B1+B3)/2 and B5 = (B1 + B9)/2. With average of four pixels for example: B6 = (B1 + B3 + B9 + B11) / 4. This procedure is then repeated over the whole image and results are combined to RGB image similarly as in the NN-interpolation.
3. Patterned Pixel Grouping Interpolation

Patterned Pixel Grouping (PPG) interpolation [2] is an advanced interpolation method which is designed to maintain the high image quality and minimize the interpolation artifacts from the output. It is used for example in raw image processing software Ufraw [3].

Implement the PPG interpolation according to the instructions of [2]. When implementing the algorithm note that instructions of [2] assume that the pixel in top left corner is red not blue as we have here in our data.

Result Quality Measurements

The first quality measurement to use here is the Mean Squared Error (MSE):

\[
MSE = \frac{1}{n} \sum_{i=1}^{n} (\hat{Y}_i - Y_i)^2
\]

Where the \( \hat{Y} \) is the ground truth and the \( Y \) is the result from the current method.

The second measurement is the Mean Absolute Error (MAE):

\[
MAE = \frac{1}{n} \sum_{i=1}^{n} |f_i - y_i|
\]

Where the \( f \) is the ground truth of the study and \( y \) is the result from the current method.

Test all three implemented methods with second dataset by comparing the result of the method to the ground truth of the data. Present the results in a single bar-chart.

Test also the computing times of your implementations with tic-toc mechanism of Matlab. Present and discuss the results of quality measurements and computing times in your report. Report also the amount of time you used with the whole project.

Writing the Report and Programming

Please write the report in PDF format such that it is printer friendly and user-friendly. The number of words is not important, but the content is. Include your answers for the preliminary questions in to the report. Keep in mind the question: Does this report explain and visualize my project achievements such that I myself would like to read it through and would also understand it? Keep the size of the report under 5Mb (do not include too many images and take care that image size is reasonable).

During the project, keep in mind the basics of good programming. The program code should be easy to read, well-commented and fast to run. It should also be intuitive to analyze and debug, if the results are not what was expected. This means that the programmer should also use descriptive variable names. Please check that your solution works in Matlab of TC303 / TC407 without any extra codes or toolboxes.

Implement the different interpolation methods in a separated matlab functions and then call them from the main script. In general, it is most useful to start by writing a script and, after a segment of code is ready and tested, wrap it into a function. Consider also the use of sub functions inside the main functions. Familiarize yourself with the Matlab debugger. It is highly useful when programming functions in Matlab. Details of function programming and debugger will be discussed during laboratory session in TE407. Take care that no errors are shown in any situation of use. Test that your function works in Windows (TC303) as well as in Linux (TC407)

References

[2] https://sites.google.com/site/chklin/demosaic/

Extra material: