|  |  |
| --- | --- |
| **Image processing** | **Task No. 1** |
| **Task variant: Group 1**  (all B, all G, N9, all E) | |
| **Day and time** November 8, 2022  **Academic year 2022/2023** | **Full name : Piotr Czapla 234751**  **Full name : Aleksandra Banasiak 234750**  The text in blue contains instructions about what should be put into appropriate sections of the report. The instructions must be followed and the blue text itself should be deleted afterwards. The layout of this template must be respected; otherwise the report will be rejected. In this section the following information should be given: the number of the task, its variant, day and time of the classes, academic year and the full names of the group members. |
| **Technical description of the application**  Description  In order to fulfill the requirements we decided to focus on programming our application in C++ programming languages as we had the basic knowledge from previous projects. Not to mention the research we performed to find the suitable library with open source and free usage for students. We decided to focus on features of the CImg Library which is a small and open-source C++ library for image processing and thanks to that we were able to create function which will be described precisely in further points of the report.  Additionally to create command line application we used available Program Options Parser Library which is a C++ command line arguments parser that supports the same set of options as GNU's getopt and thus closely follows the POSIX guidelines for the command-line options of a program. We used available resources from <https://github.com/badaix/popl> to implement our solution.  In this section the technical description of the application should be placed. In the case of using an external library, specify how it is used, what structures are used for storing the image in memory, etc. | |
| **Description of implementation of basic image operations**   * Description (B1) Image brightness modification     **Figure nr.1** Original picture, before editing    **Figure nr.2** Picture after execution of brightness method with value 100  To adjust the brightness of an image, we created the function called brightness\_modification which as a variable from the user takes a constant type integer by which the brightness of the picture should be increased. It changes the value of all pixels by a this handed over constant. Adding a positive constant to all of the image pixel values makes the image brighter. Similarly the user can subtract a positive constant from all of the pixel values to make the image darker.  void brightness\_modification(int constant) {  CImg<unsigned char> image("..\\..\\images\\lenac.bmp");if(constant >= 0)  {  for (int x = 0; x < image.width(); x++)  {  for (int y = 0; y < image.height(); y++)  {  float valR = image(x, y, 0) + constant;  float valG = image(x, y, 1) + constant;  float valB = image(x, y, 2) + constant;   if(valR <= 255) image(x, y,0) = valR;  else image(x, y,0) = 255;   if(valG <= 255) image(x, y,1) = valG;  else image(x, y,1) = 255;   if(valB <= 255) image(x, y,2) = valB;  else image(x, y,2) = 255;  }  }  }  else  {  for (int x = 0; x < image.width(); x++)  {  for (int y = 0; y < image.height(); y++)  {  float valR = image(x, y, 0) + constant;  float valG = image(x, y, 1) + constant;  float valB = image(x, y, 2) + constant;   if(valR >= 0) image(x, y,0) = valR;  else image(x, y,0) = 0;   if(valG >= 0) image(x, y,1) = valG;  else image(x, y,1) = 0;   if(valB >= 0) image(x, y,2) = valB;  else image(x, y,2) = 0;  }  }  }  image.save\_bmp("..\\..\\images\\brightness\_modification.bmp");  }     * Description (B2) Image contrast modification     **Figure nr.3** Picture after execution of contrast method with value of 200  Contrast is the difference between maximum and minimum pixel intensity in an image. Intending change the contrast of the picture we created the method called contrast\_modification which takes value of intensity as the variable from the user. After executing this method it changes the contrast of an image by changing the value of the max and min intensity pixel.  void contrast\_modification(int intensity){  CImg<unsigned char> image("..\\..\\images\\lenac.bmp");for (int x = 0; x < image.width(); x++) {  for (int y = 0; y < image.height(); y++) { float correction\_factor = 259\*(255+intensity)/255\*(259-intensity);  float valR = correction\_factor\*(image(x, y, 0)-128)+128;  float valG = correction\_factor\*(image(x, y, 1)-128)+128;  float valB = correction\_factor\*(image(x, y, 2)-128)+128;   if(valR <= 0)  {  image(x, y, 0) = 0;  }  else if (valR >= 255) image(x, y,0) = 255;   else image(x, y, 0) = valR;   if(valG <= 0){  image(x, y, 1) = 0;  }   else if (valG >= 255) image(x, y,1) = 255;   else image(x, y,1) = valG;   if(valB <= 0)  {  image(x, y, 2) = 0;  }  else if (valB >= 255) image(x, y,2) = 255;   else image(x, y,2) = valB;   }  }  image.save\_bmp("..\\..\\images\\contrast\_modification.bmp");}   * Description (B3) Negative     **Figure nr.4** Picture after execution of negative method  Negative transformation refers to subtracting pixel values from (L-1), where L is the maximum possible value of the pixel, and replacing it with the result. To negatively transform an image, we loop through the pixels using two for loops. If the image is RGB, the red, green, and blue values are subtracted from (L-1) and the result is stored in place of the values. In the case of greyscale images, the intensity of the pixels is subtracted instead. Negative transformation is done to bring attention to detail in the darker regions of the image.  void negative(){  CImg<unsigned char> image("..\\..\\images\\lenac.bmp");  for (int x = 0; x < image.width(); x++) {  for (int y = 0; y < image.height(); y++) {float valR = image(x, y, 0);  float valG = image(x, y, 1); float valB = image(x, y, 2);  float negative1 = 255-valR;  float negative2 = 255-valG;  float negative3 = 255-valB;  image(x, y, 0) = negative1;  image(x, y, 1) = negative2;  image(x, y, 2) = negative3;  }  }  image.save\_bmp("..\\..\\images\\negative.bmp}   * Description (G1) Horizontal flip     **Figure nr.5** Picture after execution of horizontal flip method  To define the term, Horizontal Flip is a data augmentation technique that takes both rows and columns of such a matrix and flips them horizontally. As a result, you will get an image flipped horizontally along the y-axis.  void horizontal\_flip(){  CImg<unsigned char> image("..\\..\\images\\lenac.bmp");  CImg<unsigned char> buffer = image;  for (int x = 0; x < image.width(); x++) {  for (int y = 0; y < image.height(); y++) {  buffer(x, y, 0) = image(image.width()-x, y, 0);  buffer(x, y, 1) = image(image.width()-x, y, 1);  buffer(x, y, 2) = image(image.width()-x, y, 2);  }  }  image = buffer;  image.save\_bmp("..\\..\\images\\horizontal\_flip.bmp");  }   * Description (G2) Vertical flip     **Figure nr.6** Picture after execution of vertical flip method  Vertical Flip flips algorithm works the same as described in horizontal flip with one distinction that image flips along the x-axis.  void vertical\_flip(){  CImg<unsigned char> image("..\\..\\images\\lenac.bmp");  CImg<unsigned char> buffer = image;  for (int x = 0; x < image.width(); x++) {  for (int y = 0; y < image.height(); y++) {  buffer(x, y, 0) = image(x, image.height()-y, 0);  buffer(x, y, 1) = image(x, image.height()-y, 1);  buffer(x, y, 2) = image(x, image.height()-y, 2);  }  }  image = buffer;  image.save\_bmp("..\\..\\images\\vertical\_flip.bmp");  }   * Description (G3) Diagonal flip     **Figure nr.7** Picture after execution of diagonal flip method  As we know the colored image can be represented as a 3 order matrix. The first order is for the rows, the second order is for the columns and the third order is for the layers, the pixel value will determine the color of the pixel based on the color format. Approach to implement diagonal flip is separating each layer, then flipping every layer up to down then flipping every layer right to left and finally taking transpose of the image.  void diagonal\_flip(){  CImg<unsigned char> image("..\\..\\images\\lenac.bmp");  CImg<unsigned char> buffer = image;  for (int x = 0; x < image.width(); x++) {  for (int y = 0; y < image.height(); y++) {  buffer(x, y, 0) = image(image.width()-x, image.height()-y, 0);  buffer(x, y, 1) = image(image.width()-x, image.height()-y, 1);  buffer(x, y, 2) = image(image.width()-x, image.height()-y, 2);  }  }  image = buffer;  image.save\_bmp("..\\..\\images\\diagonal\_flip.bmp");  }   * Description (G4) Image shrinking     **Figure nr.8** Picture after execution of shrink method  ????  We decided to use the Nearest neighbor Image Scaling algorithm as it is the simplest and fastest implementation of image scaling technique. The existing pixel values are the only information you have access to in order to generate a larger or smaller version of that image. For larger versions of the original image, you can take the original pixel values and place them analogically across the new specified dimension, so you can fill up your new canvas size, but there will be new (and vacant) pixel positions that you will have to generate values for, otherwise your image will look like a series of detached grid squares with a black border. In order to avoid that, the resampling algorithm has to fill that space.  However, note that nearest neighbor doesn't perform any interpolation for filling up the values of these newly generated pixels. It copies values to fill the required gap, so no new values are ever generated. In scaling up, it just determines through analogy the x and y positions for the existing pixels to be positioned at, and then copies their values to the vacant pixels that are nearest to those.  void shrink(float multiplier){  if (multiplier<=0){   }else if(multiplier>=1){   }else{  CImg<unsigned char> image("..\\..\\images\\lenac.bmp");  CImg<unsigned char> buffer (image.width()\*multiplier,image.width()\*multiplier,1,3,0);  for (int x = 0; x < buffer.width(); x++) {  for (int y = 0; y < buffer.height(); y++) {  buffer(x, y, 0) = image(x/multiplier, y/multiplier, 0);  buffer(x, y, 1) = image(x/multiplier, y/multiplier, 1);  buffer(x, y, 2) = image(x/multiplier, y/multiplier, 2);  }  }  image = buffer;  image.save\_bmp("..\\..\\images\\shrink.bmp");  }   }   * Description (G5) Image enlargement     **Figure nr.9** Picture after execution of enlargement method  void enlarge(float multiplier){  if (multiplier<=0){   }else if(multiplier<=1){   }else{  CImg<unsigned char> image("..\\..\\images\\lenac.bmp");  CImg<unsigned char> buffer (image.width()\*multiplier,image.width()\*multiplier,1,3,0);  for (int x = 0; x < buffer.width(); x++) {  for (int y = 0; y < buffer.height(); y++) {  buffer(x, y, 0) = image(x/multiplier, y/multiplier, 0);  buffer(x, y, 1) = image(x/multiplier, y/multiplier, 1);  buffer(x, y, 2) = image(x/multiplier, y/multiplier, 2);  }  }  image = buffer;  image.save\_bmp("..\\..\\images\\enlarge.bmp");  }   }  Here, a short description of implementation of the aforementioned basic operations should be placed. Attention should be paid (if possible) to the efficiency of this implementation w. r. t. both the time and the memory usage (these aspects should be described). | |
| **Description of implementation of noise reduction methods**   * The first method : Adaptive median filter * The second method : Arithmetic mean filter   A short description of implementation of the assigned (in the variant N) noise reduction methods should be placed in this section. Attention should be paid (if possible) to the efficiency of this implementation w. r. t. both the time and the memory usage (these aspects should be described). | |
| **Analysis of parameters of the noise reduction methods**  Results and conclusions  The results of the experiments related to changes in values of the parameters (if there are any) of the assigned noise reduction methods should be placed here. In this section the conclusions drawn from the experiments should be also placed. | |
| **Analysis of the noise reduction methods w. r. t. the possible applications for various types of noise**  Results and conclusions  Objective (coefficients E) and subjective conclusions related to the quality of the results obtained by the assigned noise reduction methods for different types of noise should be placed here. The images provided on the web page should be applied for this purpose. | |
| **Teacher's remarks**                This is a section for teacher's remarks for the laboratory group (please leave some free place). | |