IaMP_Lab2

2

Створено системою Doxygen 1.9.1

Розділ 1

Звіт з лабораторних робіт №2 та №3

за дисципліною "Обробка зображень та мультимедіа"
студента групи ПА-17-2
Панасенка Єгора Сергійовича
Кафедра комп'ютерних технологій
ФПМ, ДНУ, 2017-2018 навч.р.
Тема: "Точкові методи обробки зображень. Методи фільтрації зображень."
Варіант 17

Звіт доступний за посиланням
https://gaurapanasenko.github.io/unilab_opt/IaMP_Lab2/html/index.html.
Вихідний код доступний за посиланням
https://github.com/gaurapanasenko/unilab/tree/master/08/IaMP_Lab2

1.0.1 Постановка задачі лабораторної роботи 2

1.0.1.1 Завдання I.

Вибрати одну із фотографій. Перетворити її на напівтонове зображення (програмно або за допомогою існуючих додатків). Згенерувати два зображення: затемнене та висвітлене.

Розробити програму, в якій будуть виконуватись такі дії із завантаженим напівтоновим зображенням:

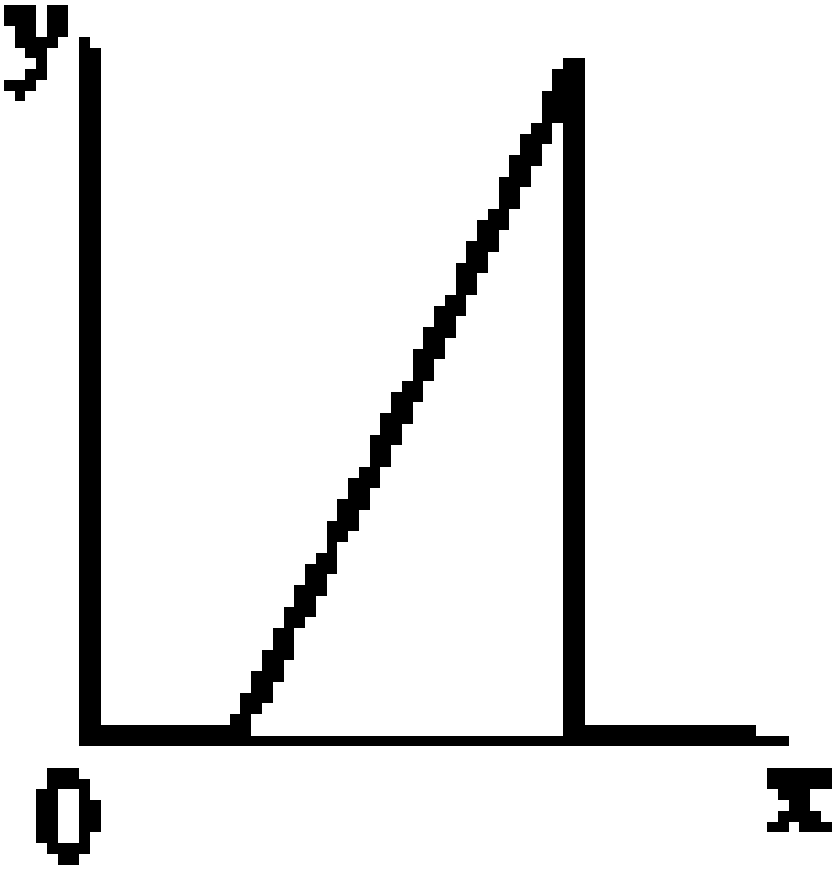
розрахунок та побудування гістограми; вирівнювання гістограми зображення, метод – за варіантами:

Варіант	Метод
1, 5, 9, 13, 17, 21, 25	Еквалізація гістограми

Показати, як працює розроблена програма на затемненому та на висвітленому зображеннях. Зробити висновки щодо реалізованого методу.

1.0.1.2 Завдання II.

Реалізувати програмно один з способів препарування напівтонового зображення (за варіантами):



Продемонструвати результати препарування напівтонового зображення для різних параметрів (на прикладі свого зображення).

1.0.2 Лабораторна робота 3

Розробити програму, в якій до зображення застосовується фільтр за варіантами:

Варіант	Метод фільтрації
4	Max-фільтр довільного розміру (з можливістю обрати розмір фільтру)

Показати, як працює розроблена програма на декількох фото. Зробити висновки щодо реалізованого методу.

1.0.3 Лабораторна робота 1

1.0.3.1 Завдання I.

Було розроблену програму на мові C++ та графіному інтерфейсі ImGui. Ця програма відкриває стандартні формати зображень перетворює їх у напівтонові зображення и еквалізує зображення. Також програма будує гістограми до зображень і здатна препарувати зображення. Продемонстуємо роботу програми.

Розділ 2

Алфавітний покажчик класів

2.1 Класи

Класи, структури, об'єднання та інтерфейси з коротким описом.

Розділ 3

Покажчик файлів

3.1 Файли

Повний список файлів.

Розділ 4

Класи

4.1 Клас Image

Stores pixels, width and height.

#include <image.h>

Загальнодоступні елементи

- Image (shared_ptr< const pixel_t[]> data, const int width, const int height)

  Image construcor. Stores reference to pixels, height and width of an image.
- shared_ptr< const int[256]> calcHistogram () const

  Calculates histogram of an image. Calculates histogram by red channel so Image must be grayscaled before.
- shared_ptr< const Image > toGray () const

  Convert image to grayscale. Still uses 24 bytes per pixel.
- shared_ptr< const Image > dissect (channel_t dissection[256]) const

  Dissect image. Dissecting image of every color channel except of alpha. For best result Image must be grayscaled before.
- shared_ptr< const Image > dilate (int params[2]) const

  Dilate image. Applies max filter. For this filter used mask with size (2∗params[0]+1)x(2∗params[1]+1)
- shared_ptr< const Image > erode (int params[2]) const

  Erode image. Applies max filter. For this filter used mask with size (2∗params[0]+1)x(2∗params[1]+1)

Загальнодоступні статичні елементи

- static Image fromFile (const char ∗path)

  Read image from file.

Загальнодоступні атрибути

- shared_ptr< const pixel_t[]> data

  Constant pixels array. Stores smart pointer to height ∗ width pixels of an image.
- const int width

  Width of an image.
- const int height

  Height of an image.

4.1.1 Детальний опис

Stores pixels, width and height.

Див. визначення в файлі image.h, рядок 32

4.1.2 Конструктор(и)

4.1.2.1 Image()

```
Image::Image (
            shared_ptr< const pixel_t[]> data,
            const int width,
            const int height )
```

Image construcor. Stores reference to pixels, height and width of an image.

Аргументи

data	Smart reference to array of pixels, size must be height ∗ width.
width	Width of an image.
height	Height of an image.

Див. визначення в файлі image.cpp, рядок 9

```
11     : data(data), width(width), height(height)
12 {
13 }
```

4.1.3 Опис методів компонент

4.1.3.1 calcHistogram()

```
std::shared_ptr< const int[256]> Image::calcHistogram ( ) const
```

Calculates histogram of an image. Calculates histogram by red channel so Image must be grayscaled before.

Повертає

Smart pointer to histogram.

Див. визначення в файлі image.cpp, рядок 24

```
25 {
26     std::shared_ptr<int[256]> histogram(new int[256]{0});
27     const pixel_t *in_data = data.get();
28
29     int size = height * width;
30     for (int i = 0; i < size; i++)
31         histogram[in_data[i][0]]++;
32
33     return histogram;
34 }
```

4.1.3.2 dilate()

shared_ptr< const Image > Image::dilate (
int params[2]) const

Dilate image. Applies max filter. For this filter used mask with size (2∗params[0]+1)x(2∗params[1]+1)

Аргументи

params	size of mask.

Повертає

Smart pointer to dilated image.

Див. визначення в файлі image.cpp, рядок 73

```
74 {
75     std::shared_ptr<pixel_t[]> out_data(new pixel_t[width * height]);
76     const pixel_t *in_data = data.get();
77     int i, j, k, l, kbegin, kend, lbegin, lend;
78 
79     for (i = 0; i < height; i++) {
80         for (j = 0; j < width; j++) {
81             channel_t mx = in_data[i * width + j][0];
82             kbegin = max(i - params[1], 0);
83             kend = min(i + params[1] + 1, height);
84             lbegin = max(j - params[0], 0);
85             lend = min(j + params[0], width - 1);
86             for (k = kbegin; k != kend; k++) {
87                 for (l = lbegin; l != lend; l++) {
88                     mx = max(mx, in_data[k * width + l][0]);
89                 }
90             }
91             for (k = 0; k < 3; k++) {
92                 out_data[i * width + j][k] = mx;
93             }
94             out_data[i * width + j][3] = 255;
95         }
96     }
97     return std::make_shared<Image>(out_data, width, height);
98 }
```

4.1.3.3 dissect()

shared_ptr< const Image > Image::dissect (
channel_t dissection[256]) const

Dissect image. Dissecting image of every color channel except of alpha. For best result Image must be grayscaled before.

Аргументи

dissection	New values for all possible channel values.

Повертає

Smart pointer to dissected image.

Див. визначення в файлі image.cpp, рядок 57

```
58 {
59     const int size = width * height;
60     std::shared_ptr<pixel_t[]> out_data(new pixel_t[size]);
61     const pixel_t *in_data = data.get();
62 
63     for (int i = 0; i < size; i++) {
64         for (int j = 0; j < 3; j++) {
65             out_data[i][j] = dissection[in_data[i][j]];
66         }
67         out_data[i][3] = 255;
68     }
69 
70     return std::make_shared<Image>(out_data, width, height);
71 }
```

4.1.3.4 erode()

shared_ptr< const Image > Image::erode (
int params[2]) const

Erode image. Applies max filter. For this filter used mask with size (2∗params[0]+1)x(2∗params[1]+1)

Аргументи

params	size of mask.

Повертає

Smart pointer to eroded image.

Див. визначення в файлі image.cpp, рядок 100

```
101 {
102     std::shared_ptr<pixel_t[]> out_data(new pixel_t[width * height]);
103     const pixel_t *in_data = data.get();
104     int i, j, k, l, kbegin, kend, lbegin, lend;
105 
106     for (i = 0; i < height; i++) {
107         for (j = 0; j < width; j++) {
108             channel_t mx = in_data[i * width + j][0];
109             kbegin = max(i - params[1], 0);
110             kend = min(i + params[1] + 1, height);
111             lbegin = max(j - params[0], 0);
112             lend = min(j + params[0], width - 1);
113             for (k = kbegin; k != kend; k++) {
114                 for (l = lbegin; l != lend; l++) {
115                     mx = min(mx, in_data[k * width + l][0]);
116                 }
117             }
118             for (k = 0; k < 3; k++) {
119                 out_data[i * width + j][k] = mx;
120             }
121             out_data[i * width + j][3] = 255;
122         }
123     }
124     return std::make_shared<Image>(out_data, width, height);
125 }
```

4.1.3.5 fromFile()

Image Image::fromFile (
const char ∗ path) [static]

Read image from file.

Аргументи

path	Path of image file.

Повертає

Image class, that also references to pixels.

Див. визначення в файлі image.cpp, рядок 15

```
16 {
17     int width = 0, height = 0;
18     pixel_t *data = (pixel_t *)stbi_load(path, &width, &height, NULL, COMP);
19     assert(data != NULL);
20     std::shared_ptr<pixel_t[]> ptr(data, stbi_image_free);
21     return {ptr, width, height};
22 }
```

4.1.3.6 toGray()

std::shared_ptr< const Image > Image::toGray ( ) const

Convert image to grayscale. Still uses 24 bytes per pixel.

Повертає

Smart pointer to grayscaled image.

Див. визначення в файлі image.cpp, рядок 36

```
37 {
38     int size = width * height;
39     std::shared_ptr<pixel_t[]> out_data(new pixel_t[size]);
40     const pixel_t *in_data = data.get(), *pixel;
41     unsigned char avg;
42 
43     for (int i = 0; i < size; i++) {
44         pixel = &in_data[i];
45         avg = (*pixel[0] + *pixel[1] + *pixel[2]) / 3;
46         //avg = fmax((int)pixel[0], (int)pixel[1]);
47         //avg = fmax((int)avg, (int)pixel[2]);
48         for (int k = 0; k < 3; k++) {
49             out_data[i][k] = avg;
50         }
51         out_data[i][3] = 255;
52     }
53 
54     return std::make_shared<Image>(out_data, width, height);
55 }
```

4.1.4 Компонентні дані

4.1.4.1 data

shared_ptr<const pixel_t[]> Image::data

Constant pixels array. Stores smart pointer to height ∗ width pixels of an image.

Див. визначення в файлі image.h, рядок 39

4.1.4.2 height

const int Image::height

Height of an image.

Див. визначення в файлі image.h, рядок 47

4.1.4.3 width

const int Image::width

Width of an image.

Див. визначення в файлі image.h, рядок 43

Документація цих класів була створена з файлів:

- image.h
- image.cpp

4.2 Клас ImageData

Image metadata class. Stores pointer to image, and histograms. Also stores maximum value of histogram.

#include <imagedata.h>

Загальнодоступні елементи

- ImageData (shared_ptr< const Image > image)
  ImageData constructor. Calulates and stores histogram based on image.
- shared_ptr< const Image > equalize () const
  Equalize image using histogram.

Загальнодоступні статичні елементи

- static shared_ptr< const float[256]> copyHistogram (shared_ptr< const int[256]> histogramI)
  Converts histogram from integer type to float.

Загальнодоступні атрибути

- shared_ptr< const Image > image
  Smart pointer to image.
- shared_ptr< const int[256]> histogramI
  Smart pointer to histogram with integer type.
- int maxHistogramI
  Maximum value of histogramI.
- shared_ptr< const float[256]> histogramF
  Smart pointer to histogram with float type.
- float maxHistogramF
  Maximum value of histogramF.

4.2.1 Детальний опис

Image metadata class. Stores pointer to image, and histograms. Also stores maximum value of histogram.

Див. визначення в файлі imagedata.h, рядок 10

4.2.2 Конструктор(и)

4.2.2.1 ImageData()

```
ImageData::ImageData (
            shared_ptr< const Image > image )
```

ImageData constructor. Calulates and stores histogram based on image.

Аргументи

image	Smart pointer to image.

Див. визначення в файлі imagedata.cpp, рядок 4

```
5     : image(image), histogramI(image->calcHistogram()),
6       maxHistogramI(*std::max_element(histogramI.get(), histogramI.get() + 256)),
7       histogramF(copyHistogram(histogramI)),
8       maxHistogramF(maxHistogramI)
9 {
10 }
```

4.2.3 Опис методів компонент

4.2.3.1 copyHistogram()

```
std::shared_ptr< const float[256]> ImageData::copyHistogram (
            shared_ptr< const int[256]> histogramI )   [static]
```

Converts histogram from integer type to float.

Аргументи

histogramI	Integer type histogram.

Повертає

Float type histogram.

Див. визначення в файлі imagedata.cpp, рядок 13

```
14 {
15     std::shared_ptr<float[256]> histogram(new float[256]{0});
16     std::copy(histogramI.get(), histogramI.get() + 256, histogram.get());
17     return histogram;
18 }
```

4.2.3.2 equalize()

std::shared_ptr< const Image > ImageData::equalize ( ) const

Equalize image using histogram.

Повертає

New equalized image.

Див. визначення в файлі imagedata.cpp, рядок 20

```
21 {
22     int width = image->width, height = image->height;
23     int size = width * height;
24     std::shared_ptr<pixel_t[]> data(new pixel_t[width * height]);
25     const pixel_t *in_data = image->data.get();
26     const int *histogram = histogramI.get();
27     int accum = 0;
28     int s[256];
29 
30     for (int i = 0; i < 256; i++) {
31         accum += histogram[i];
32         s[i] = 255 * accum / size;
33     }
34 
35     for (int i = 0; i < size; i++) {
36         channel_t cur = s[in_data[i][0]];
37         data[i][0] = cur;
38         data[i][1] = cur;
39         data[i][2] = cur;
40         data[i][3] = 255;
41     }
42 
43     return std::make_shared<Image>(data, width, height);
44 }
```

4.2.4 Компонентні дані

4.2.4.1 histogramF

shared_ptr<const float[256]> ImageData::histogramF

Smart pointer to histogram with float type.

Див. визначення в файлі imagedata.h, рядок 27

4.2.4.2 histogramI

shared_ptr<const int[256]> ImageData::histogramI

Smart pointer to histogram with integer type.

Див. визначення в файлі imagedata.h, рядок 19

4.2.4.3 image

shared_ptr<const Image> ImageData::image

Smart pointer to image.

Див. визначення в файлі imagedata.h, рядок 15

4.2.4.4 maxHistogramF

float ImageData::maxHistogramF

Maximum value of histogramF.

Див. визначення в файлі imagedata.h, рядок 31

4.2.4.5 maxHistogramI

int ImageData::maxHistogramI

Maximum value of histogramI.

Див. визначення в файлі imagedata.h, рядок 23

Документація цих класів була створена з файлів:

- imagedata.h
- imagedata.cpp

4.3 Клас Processor

Processing image. Calculates histogram, applies dissection or max filter and stores result image to texture.

#include <processor.h>

Загальнодоступні елементи

- Processor (shared_ptr< const ImageData > input)
  Processor constuctor.
- void updateDissection ()
  Calculates new image and stores it to texture.
- bool process_image (const char *name)
  Implements interface to manipulate fields.

Приватні дані

- shared_ptr< const ImageData > orig
  Original image with calculated histogram.
- shared_ptr< const ImageData > data
  Currently shown image on display.
- const Texture texture
  Texture used to draw with OpenGL.
- int dissection_x [2]
  Range of channel value where apply dissection.
- float dissection_y [2]
  Range of proportional coefficients between dissection values. Coefficients must be in range [0, 1] to make sure that new channel values will not out of [0, 255] range.
- channel_t dissection [256]
  New channel values for dissection for image.
- float dissectionF [256]
  New channel values for dissection for image in float type.
- int dilate_params [2]
  Size of max filter mask.
- int erode_params [2]
  Size of min filter mask.
- bool dissected
  Flag to apply dissection to the image.
- bool dilate
  Flag to apply dilation to the image.
- bool erode
  Flag to apply erosion to the image.

4.3.1 Детальний опис

Processing image. Calculates histogram, applies dissection or max filter and stores result image to texture.

Also have implements interface to manipulate parameters in this class with imgui.

Див. визначення в файлі processor.h, рядок 15

4.3.2 Конструктор(и)

4.3.2.1 Processor()

```
Processor::Processor (
            shared_ptr< const ImageData > input )
```

Processor constuctor.

Аргументи

input	Input image that will be processed by this class.

Див. визначення в файлі processor.cpp, рядок 7

```
8     : orig(input),
9       data(input),
10      dissection_x{100, 200},
11      dissection_y{0, 1},
12      dissection{0},
13      dilate_params{1, 1},
14      erode_params{1, 1},
15      dissected(false),
16      dilate(false),
17      erode(false)
18 {
19    updateDissection();
20 }
```

4.3.3 Опис методів компонент

4.3.3.1 process_image()

bool Processor::process_image (
const char * name)

Implements interface to manipulate fields.

Аргументи

name	name of window that will be shown.

Повертає

boolean value that shows that it is needed to update image.

Див. визначення в файлі processor.cpp, рядок 47

```
47                                          {
48    bool opened = true;
49    auto img = data->image;
50    auto tex_id = (void*)(intptr_t)(texture.id);
51    ImGui::PushID(name);
52
53    ImVec2 size(img->width, img->height);
54    ImGui::SetNextWindowSize(size, ImGuiCond_FirstUseEver);
55    ImGui::PushStyleVar(ImGuiStyleVar_WindowPadding, ImVec2(0,0));
56    ImGui::Begin(name, &opened, ImGuiWindowFlags_NoSavedSettings);
57    ImGui::Image(tex_id, ImGui::GetContentRegionAvail());
58    ImGui::End();
59    ImGui::PopStyleVar();
60
61    char info_name[128];
62    snprintf(info_name, 128, "Info %s", name);
63    ImGui::Begin(info_name, NULL, ImGuiWindowFlags_NoSavedSettings);
64    ImGui::Text("pointer = %ld", (intptr_t)tex_id);
65    ImGui::Text("size = %d x %d", img->width, img->height);
66    ImGui::PlotHistogram(
67              "##", data->histogramF.get(), 256, 0, "Histogram", 0.0f,
68              data->maxHistogramF, ImVec2(0, 100.0f));
69    bool changed = false;
70    ImGui::Spacing();
```

```
71     changed |= ImGui::Checkbox("Dissected", &dissected);
72     if (dissected) {
73         {
74             auto val = dissection_x;
75             changed |= ImGui::DragIntRange2("dissection x", val, val + 1, 1, 0,
76                                             256, "Min: %d", "Max: %d");
77         }
78         {
79             auto val = dissection_y;
80             changed |= ImGui::DragFloatRange2("dissection y", val, val + 1, 0.01,
81                                               0, 1, "Min: %.2f", "Max: %.2f");
82         }
83         ImGui::PlotLines("Lines", dissectionF, 256, 0, NULL, 0, 1.0f,
84                          ImVec2(0, 80.0f));
85     }
86     ImGui::Spacing();
87     changed |= ImGui::Checkbox("Dilate", &dilate);
88     if (dilate) {
89         changed |= ImGui::SliderInt2("dilate params", dilate_params, 0, 16);
90     }
91     changed |= ImGui::Checkbox("Erode", &erode);
92     if (erode) {
93         changed |= ImGui::SliderInt2("erode params", erode_params, 0, 16);
94     }
95     if (changed) updateDissection();
96     ImGui::End();
97
98     ImGui::PopID();
99     return opened;
100 }
```

4.3.3.2 updateDissection()

void Processor::updateDissection ( )

Calculates new image and stores it to texture.

Див. визначення в файлі processor.cpp, рядок 22

```
22                                  {
23     data = orig;
24     if (dissected) {
25         float diffY = dissection_y[1] - dissection_y[0];
26         int diffX = dissection_x[1] - dissection_x[0];
27         float diff = diffY / diffX;
28         float acc = dissection_y[0];
29         memset(dissection, 0, 256 * sizeof(channel_t));
30         memset(dissectionF, 0, 256 * sizeof(float));
31         for (int i = dissection_x[0]; i < dissection_x[1]; i++) {
32             acc += diff;
33             dissection[i] = acc * 255;
34             dissectionF[i] = acc;
35         }
36         data = make_shared<ImageData>(data->image->dissect(dissection));
37     }
38     if (dilate) {
39         data = make_shared<ImageData>(data->image->dilate(dilate_params));
40     }
41     if (erode) {
42         data = make_shared<ImageData>(data->image->erode(erode_params));
43     }
44     texture.update(*data->image);
45 }
```

4.3.4 Компонентні дані

4.3.4.1 data

shared_ptr<const ImageData> Processor::data [private]

Currently shown image on display.

Див. визначення в файлі processor.h, рядок 23

4.3.4.2 dilate

bool Processor::dilate [private]

Flag to apply dilation to the image.

Див. визначення в файлі processor.h, рядок 61

4.3.4.3 dilate_params

int Processor::dilate_params[2] [private]

Size of max filter mask.

Див. визначення в файлі processor.h, рядок 49

4.3.4.4 dissected

bool Processor::dissected [private]

Flag to apply dissection to the image.

Див. визначення в файлі processor.h, рядок 57

4.3.4.5 dissection

channel_t Processor::dissection[256] [private]

New channel values for dissection for image.

Див. визначення в файлі processor.h, рядок 41

4.3.4.6 dissection_x

int Processor::dissection_x[2] [private]

Range of channel value where apply dissection.

Див. визначення в файлі processor.h, рядок 31

4.3.4.7 dissection_y

float Processor::dissection_y[2] [private]

Range of proportional coefficients between dissection values. Coefficients must be in range [0, 1] to make sure that new channel values will not out of [0, 255] range.

Див. визначення в файлі processor.h, рядок 37

4.3.4.8 dissectionF

float Processor::dissectionF[256] [private]

New channel values for dissection for image in float type.

Див. визначення в файлі processor.h, рядок 45

4.3.4.9 erode

bool Processor::erode [private]

Flag to apply erosion to the image.

Див. визначення в файлі processor.h, рядок 65

4.3.4.10 erode_params

int Processor::erode_params[2] [private]

Size of min filter mask.

Див. визначення в файлі processor.h, рядок 53

4.3.4.11 orig

shared_ptr<const ImageData> Processor::orig [private]

Original image with calculated histogram.

Див. визначення в файлі processor.h, рядок 19

4.3.4.12 texture

const Texture Processor::texture [private]

Texture used to draw with OpenGL.

Див. визначення в файлі processor.h, рядок 27

Документація цих класів була створена з файлів:

- processor.h
- processor.cpp

Розділ 5

Файли

5.1 Файл image.cpp

```
#include <stb_image.h>
#include <algorithm>
#include "image.h"
```

Макровизначення

- #define STB_IMAGE_IMPLEMENTATION

5.1.1 Опис макровизначень

5.1.1.1 STB_IMAGE_IMPLEMENTATION

#define STB_IMAGE_IMPLEMENTATION

Див. визначення в файлі image.cpp, рядок 1

5.2 Файл image.h

```
#include <memory>
```

Класи

- class Image

  Stores pixels, width and height.

Макровизначення

- #define COMP 4

Визначення типів

- typedef unsigned char channel_t

  Channel type for image. Used unsigned char because our channel values are in range [0, 255] so has 8-bit per channel.
- typedef channel_t pixel_t[COMP]

  Pixel type. One pixel has COMP channels and channel defined by channel_t type. Used 8-bit chanel with values that in range [0, 255]. Pixel has 4 channels (red, green, blue and alpha) so takes 4 bytes.

5.2.1 Опис макровизначень

5.2.1.1 COMP

#define COMP 4

Size of pixel in bypes.

Name means "components" and was taken from stb_image.h header file library.

Див. визначення в файлі image.h, рядок 10

5.2.2 Опис визначень типів

5.2.2.1 channel_t

typedef unsigned char channel_t

Channel type for image. Used unsigned char because our channel values are in range [0, 255] so has 8-bit per channel.

Див. визначення в файлі image.h, рядок 19

5.2.2.2 pixel_t

typedef channel_t pixel_t[COMP]

Pixel type. One pixel has COMP channels and channel defined by channel_t type. Used 8-bit chanel with values that in range [0, 255]. Pixel has 4 channels (red, green, blue and alpha) so takes 4 bytes.

Див. визначення в файлі image.h, рядок 27

5.3 Файл imagedata.cpp

```
#include <algorithm>
#include "imagedata.h"
```

5.4 Файл imagedata.h

```
#include "image.h"
```

Класи

- class ImageData

  Image metadata class. Stores pointer to image, and histograms. Also stores maximum value of histogram.

5.5 Файл main.cpp

```
#include <string>
#include <memory>
#include <sstream>
#include <map>
#include "gui/app.h"
#include "processor.h"
```

Макровизначення

- #define PROJECT_NAME "IaMP_Lab2"

Функції

- int main (int, char **)

5.5.1 Опис макровизначень

5.5.1.1 PROJECT_NAME

```
#define PROJECT_NAME "IaMP_Lab2"
```

Див. визначення в файлі main.cpp, рядок 9

5.5.2 Опис функцій

5.5.2.1 main()

```
int main (
                int ,
                char **  )
```

Див. визначення в файлі main.cpp, рядок 13

```
14 {
15     App app(PROJECT_NAME);
16     map<string, shared_ptr<Processor» images;
17 
18     imgui_addons::ImGuiFileBrowser *file_dialog = app.getFile_dialog();
19     // Main loop
20     while (!app.should_closed())
21     {
22         app.begin_loop();
23         auto mode = imgui_addons::ImGuiFileBrowser::DialogMode::OPEN;
24 
25         if (file_dialog->showFileDialog(
26                 "Open File", mode, ImVec2(700, 310), ".png,.jpg,.bmp"))
27         {
28             string path(file_dialog->selected_path);
29             auto img = Image::fromFile(path.c_str());
30             auto pathImage = make_shared<Image>(img);
31             auto gray = make_shared<ImageData>(pathImage->toGray());
32             {
33                 stringstream ss;
34                 ss « "Original Image: " « file_dialog->selected_fn
35                    « " " « file_dialog->selected_path;
36                 auto data = make_shared<ImageData>(pathImage);
37                 auto proc = make_shared<Processor>(data);
38                 images.insert(make_pair(ss.str(), proc));
39             }
40             {
41                 stringstream ss;
42                 ss « "Gray Image: " « file_dialog->selected_fn
43                    « " " « file_dialog->selected_path;
44                 auto proc = make_shared<Processor>(gray);
45                 images.insert(make_pair(ss.str(), proc));
46             }
47             {
48                 stringstream ss;
49                 ss « "Equalization: " « file_dialog->selected_fn
50                    « " " « file_dialog->selected_path;
51                 auto data = make_shared<ImageData>(gray->equalize());
52                 auto proc = make_shared<Processor>(data);
53                 images.insert(make_pair(ss.str(), proc));
54             }
55         }
56 
57         for (auto image = images.begin(); image != images.end();) {
58             if (!image->second->process_image(image->first.c_str())) {
59                 image = images.erase(image);
60             } else {
61                 ++image;
62             }
63         }
64         app.end_loop();
65     }
66 
67     return 0;
68 }
```

5.6 Файл mainpage.dox

5.7 Файл processor.cpp

```
#include <stdio.h>
#include "processor.h"
#include "imgui.h"
```

5.8 Файл processor.h

```
#include <cstring>
#include "imagedata.h"
#include "gui/texture.h"
```

Класи

- class Processor

  Processing image. Calculates histogram, applies dissection or max filter and stores result image to texture.