UNIVERSITÀ DEGLI STUDI DI PADOVA

Lab: feedback, code design, open discussion

Stefano Ghidoni

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

INTELLIGENT AUTONOMOUS SYSTEMS LAB

Agenda

- Compiling, executing, managing your files
- Header files & co.
- Connecting with the system – paths & co.
- Code organization
- Functions
- Function arguments
- Variables and variable names

Thanks to: Davide Allegro, Leonardo Barcellona, Laura Braganolo, Matteo Terreran

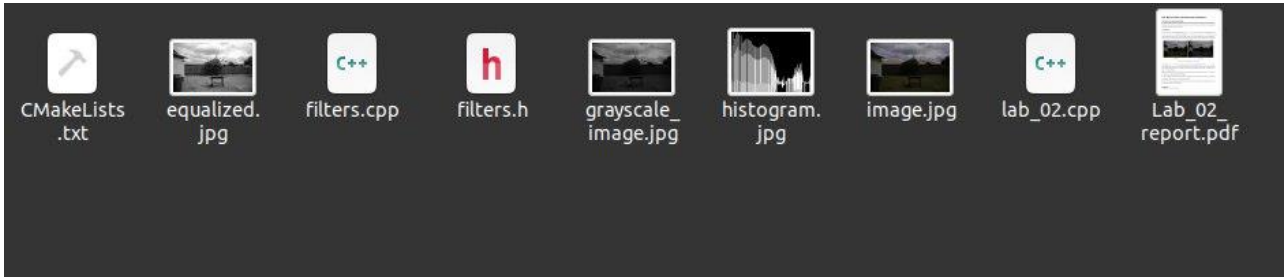
Submitting your files

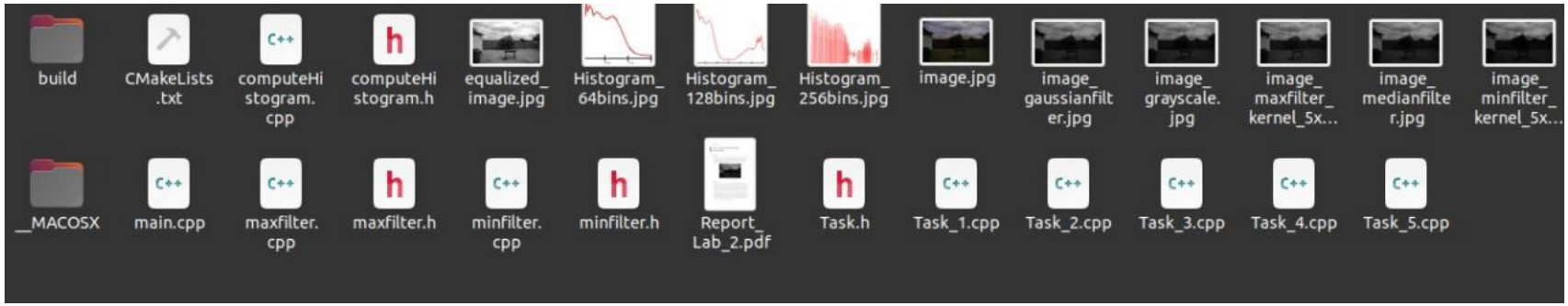
File submission

- Some submissions are made of a set of files
  - Better: one file (e.g., lab4.zip) containing a **directory** containing the whole project
- Some submissions include the binary files
  - They **should not** be included – why?

Project structure

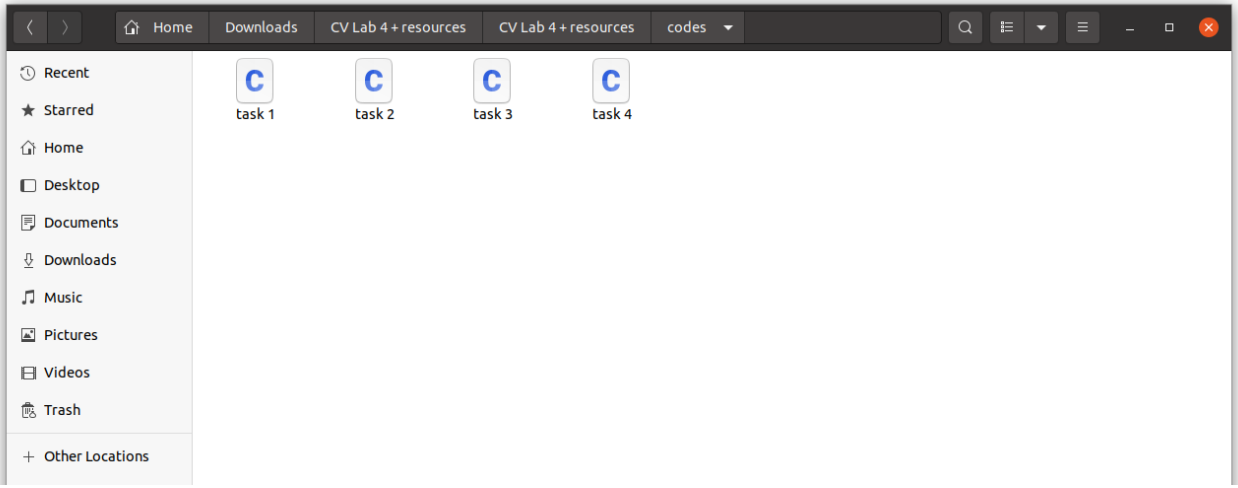
- Folders without structure





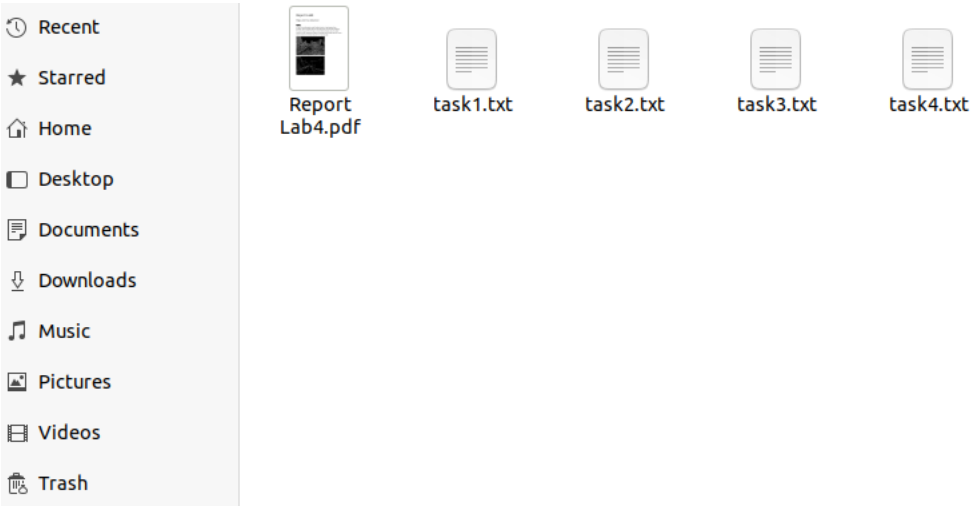
Project structure

- The project folder shall be organized using src and include folders
  - don't use other names (e.g., "codes" folder)



File extensions

- Use appropriate file extensions
  - Source code files **must not** have the .txt extension!



CMake

CMake commands

- Some CMake projects include absolute paths

```
set(Opencv_DIR /usr/local/lib/cmake/opencv4)
set(OpenCV_INCLUDE_DIRS /usr/local/include/opencv4)
set(OpenCV_LIBS /usr/local/bin)
```

  - Reason: find_package does not work
  - This might be caused by **wrong commands**, mainly due to upper/lower case

```
find_package(Opencv REQUIRED)
```

Wrong!

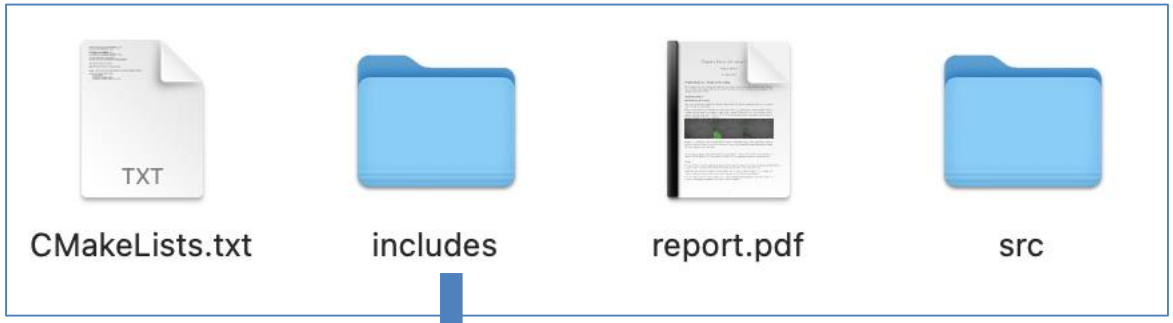
- Right spelling:

find_package(OpenCV REQUIRED)

Multiple CMakeLists.txt

- Some projects contain more CMakeLists.txt, one for each task
  - Better: use one single CMakeLists.txt with more add_executable commands

Project folder



Includes folder

Multiple CMakeLists.txt

- Be aware that you should not compile all the tasks in one single add_executable command
  - Why?

```
add_executable(${PROJECT_NAME} Task1.cpp Task3.cpp Task4.cpp)
```

CMake version

- Do not set a minimum required version that is too new unless you really need the features included in that version
  - You probably do not need them
- This prevents CMake to work on older software configurations without any need

```
cmake_minimum_required(VERSION 3.17)
```

Compiling, executing, managing your files

Code compilation

- The code you submit shall **compile and execute**
- You shall compile the code many times while coding
- Suggestion: compile every few new lines of code if you're not experienced

Header files & co.

Header files

- Header file
  - Includes function declaration(s): OK!
  - OpenCV header needed (cv::Mat is used!)
  - Include guards missing!

filters.h

```
cv::Mat filters(int const n, cv::Mat img, int filterType);
```

Include guards

- Include guards are directives used to protect the header from multiple inclusions

```
#ifndef MY_HEADER_H
#define MY_HEADER_H

// ...

#endif // MY_HEADER_H
```

Include guards

- Concept: each header file defines a dedicated constant
  - Name: related to the header file name
- First time: check if the constant is defined
  - If not: define & include the file
  - Otherwise: skip the header

```
#ifndef MY_HEADER_H
#define MY_HEADER_H

// ...

#endif // MY_HEADER_H
```

Include guards

- Alternative:

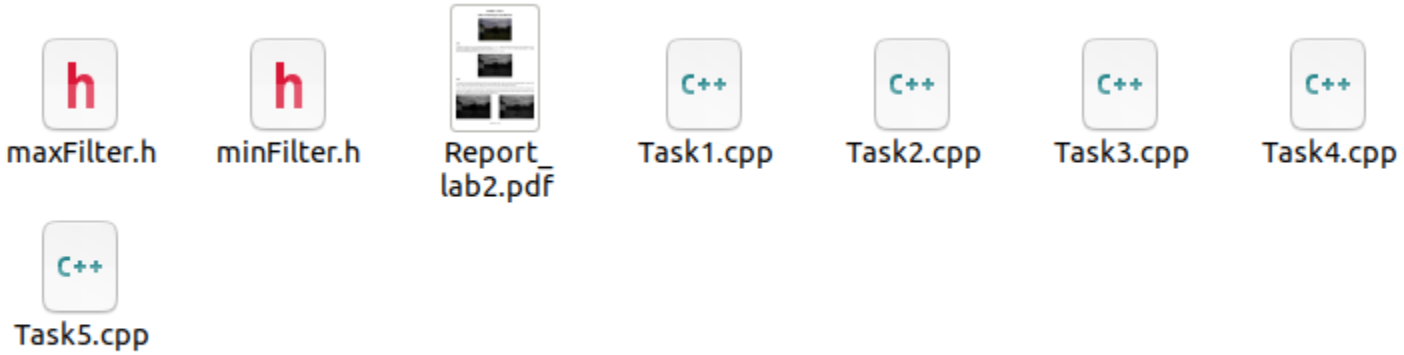
```
#pragma once                    // NON-standard!!!
```

- Frequently used, but it is not standard!

```
#pragma once
#include <opencv2/highgui.hpp>
#include "opencv2/imgproc.hpp"

//this function apply the min filter to origin. result is the resulting image
//kernel_size must be an odd number. if not, this function
//returns 0
int min(const cv::Mat& origin, cv::Mat& result, int kernel_size);

//this function apply the max filter to origin. result is the resulting image
//kernel_size must be an odd number. if not, this function
//returns 0
int max(const cv::Mat& origin, cv::Mat& result, int kernel_size);

//this function apply either the max or the min filter to the image origin
//the result is the resulting image. if selected_op = 0, it performs the
//min filter, otherwise the max one.
//it returns 0 if the given kernel_size is not an odd number
int max_min_filter(int selected_op, const cv::Mat& origin, cv::Mat& result, int kernel_size);

//functions used by max_min_filter
int max_min_test(int current, int new_value, int selected_op);
```

Header files

- Related function declarations shall be grouped into one header file
  - Why maxFilter and minFilter need two separate headers?



Header files

- Header files shall not contain function definitions

Task1.cpp | Task2.cpp | Task3.cpp | Task4.cpp | Task5.cpp | minFilter.h | maxFilter.h

```cpp
#include <iostream>
#include "opencv2/highgui.hpp"
#include "opencv2/imgcodecs.hpp"
#include "opencv2/imgproc.hpp"

void maxFilter(cv::Mat src, cv::Mat &dst, int kernel)
{
    int side = sqrt(kernel);
    int radius = side/2;

    int rr = src.rows;
    int cc = src.cols;


    for (int r = 0; r < rr; ++r)
    {
        for (int c = 0; c < cc; ++c)
        {
            int highest = 0;
            for (int i = -radius; i <= radius; ++i)
            {
                for (int j = -radius; j <= radius; ++j)
                {
                    int val = src.at<unsigned char>(r + i, c + j);
                    if (val > highest)
                        highest = val;
                }
            }
            dst.at<unsigned char>(r, c) = highest;
        }
    }
}
```

Header files

- A header file is missing here – why?

```
#include <opencv2/highgui.hpp>
#include "opencv2/imgproc.hpp"

//this function apply aeither the max or the min filter to the image origin
//the result is the resukting image. if selected_op = 0, it performs the
//min filter, otherwise the max one.
//it returns 0 if the given kernel_size is not an odd number
int max_min_filter(int selected_op, const cv::Mat& origin, cv::Mat& result, int kernel_size);

//function used by max_min_filter
int max_min_test(int current, int new_value, int selected_op);

//this function apply the max filter to origin. result is the resulting image
//kernel_size must be an odd number. if not, this function
//returns 0
int min(const cv::Mat& origin, cv::Mat& result, int kernel_size);

//this function apply the max filter to origin. result is the resulting image
//kernel_size must be an odd number. if not, this function
//returns 0
int max(const cv::Mat& origin, cv::Mat& result, int kernel_size);

//-----------------------------------------------------------------------------------------

int max_min_filter(int selected_op, const cv::Mat& origin, cv::Mat& result, int kernel_size){

    if((kernel_size % 2)==0) return 0; // wrong kernel size

    int a = (kernel_size-1)/2; // kernel_size = 2*a +1
    int right = a, left = a, up = a, down = a;
```

(Missing) header files

- You shall only include header files
  - .cpp files shall **never** be included

```
#include <opencv2/highgui.hpp>
#include <opencv2/imgproc.hpp>
#include <opencv2/imgcodecs.hpp>
#include "filters.cpp"


void toGrayscale();
void makeHistogram();
void equalizeHistogram();


int main(int argc, char** argv)
{
    toGrayscale();
    maxFilter(3);
    minFilter(1);
    medianFilter();
    gaussianFilter();
    makeHistogram();
    equalizeHistogram();
    return 0;
}
```

Including already included headers

- You can avoid including headers that are already included by other headers
  - core.hpp is included by other headers
  - This is not a mistake, however

```
#include "opencv2/core.hpp"
#include "opencv2/core/base.hpp"
#include "opencv2/core/hal/interface.h"
#include "opencv2/core/mat.hpp"
#include "opencv2/core/matx.hpp"
#include "opencv2/core/operations.hpp"
#include "opencv2/core/types.hpp"
```

Connecting with the system – paths & co.

Absolute paths

- Absolute paths shall be avoided
- Ok for the first trials, but:
  - You shall **change it before submitting** the code
  - You shall learn to **avoid such patterns** from the very beginning

```
int main(int argc, char** argv){

    // task1
    // show the original image
    Mat src,src1;
    Mat image = imread("/home/[redacted]/CLionProjects/computervision/cv_lab2/image.jpg");
    resize(image,image,Size(image.cols/2,image.rows/2));
    namedWindow("original image",WINDOW_GUI_EXPANDED);
    imshow("original image",image);
```

Absolute paths

- Absolute paths shall be avoided also in header inclusions

```
#include <opencv2/core/core.hpp>
#include <opencv2/highgui/highgui.hpp>
#include <opencv2/imgproc.hpp>
#include <opencv2/imgcodecs.hpp>
#include <C:\opencv\build\include\opencv2\core\mat.hpp>
#include <iostream>
```

Input arguments

- Whenever you use command-line arguments, you shall check them
  - They could be wrong or non-existent

```
12  int main(int argc, const char * argv[]) {
13      //TASK 1: converting the image to grayscale and saving it
14      Mat img = cv::imread(argv[1]);
15      Mat imgGray;
```

Checking input arguments

- This is **not enough** to handle the wrong input
  - Why?

```
int main(int argc, char** argv) {
  // safety check on argc
  if(argc < 2) {
    std::cerr << "Insufficient arguments, please provide the path of the image!" << std::endl;
  }

  // load the requested image
  cv::Mat img = cv::imread(argv[1], cv::IMREAD_COLOR);

  // a safety check on the image returned by cv::imread()
  if(img.data == NULL) {
    std::cout <<"The filename is wrong" << std::endl;
  }

  // show the input image
  cv::namedWindow("Original image");
  cv::imshow("Original image", img);
```

Checking input arguments

- This is **not enough** to handle the wrong input
  - Why?

```
Insufficient arguments, please provide the path of the image!
terminate called after throwing an instance of 'std::logic_error'
  what():  basic_string::_M_construct null not valid
Aborted (core dumped)
```

GUI

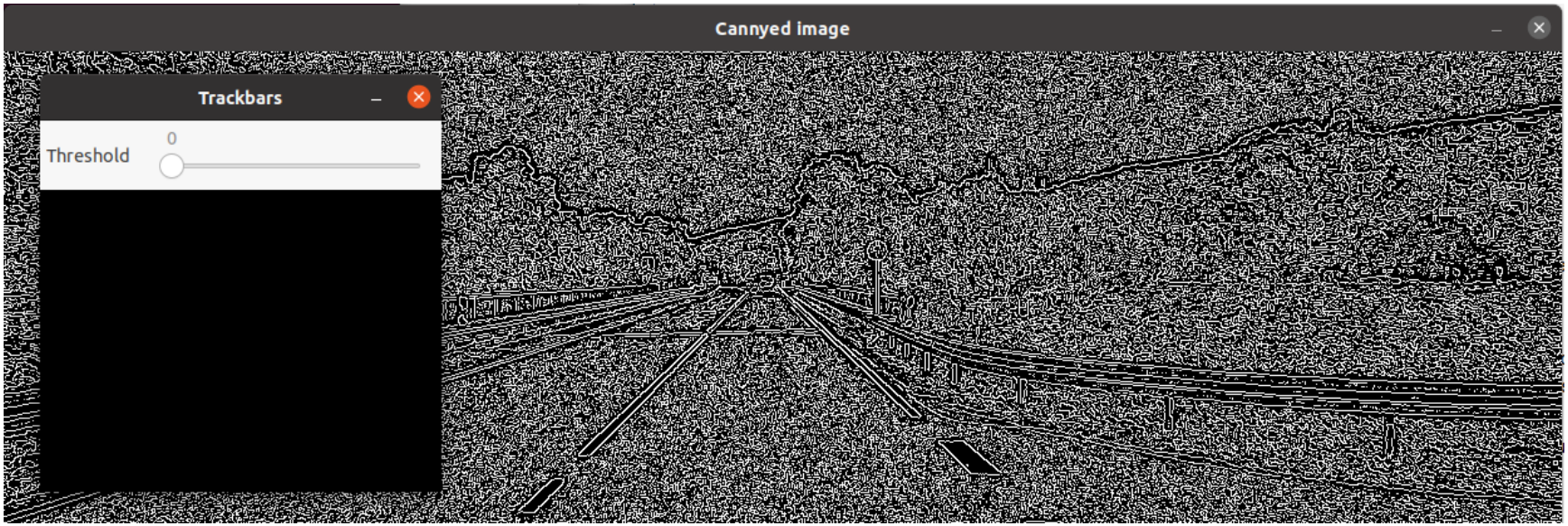
Connecting GUI elements

- Check the names of the windows and images
  - Did you really want to create a new namedWindow for the trackbar?

```
13  void onTrackbar(int, void*)
14  {
15    blur(source, imgCanny, Size(3, 3));
16    Canny(source, imgCanny, double(t), double(t * 3));
17    imshow("Cannyed image", imgCanny);
18  }
19
20  int main()
21  {
22    //TASK1
23    source = imread("street_scene.png");
24    imshow("Showing street_scene", source);
25    waitKey(0);
26    namedWindow("Trackbars");
27    createTrackbar("Threshold", "Trackbars", &t, 250, onTrackbar);
28    onTrackbar(0, 0);
29    waitKey(0);
```

Connecting GUI elements

- Check the names of the windows and images
  - Did you really want to create a new namedWindow for the trackbar?



Connecting GUI elements

- A better way: define the window names using const variables

```
int main(int argc, char** argv) {

    streetScene = imread("street_scene.png");
    if (!streetScene.data) {
        std::printf("No image data \n");
        return -1;
    }
    namedWindow("Street Scene", WINDOW_AUTOSIZE);
    imshow("Street Scene", streetScene);

    cv::cvtColor(streetScene, streetSceneGray, cv::COLOR_BGR2GRAY);
    namedWindow("Gray Street Scene", WINDOW_AUTOSIZE);
    imshow("Gray Street Scene", streetSceneGray);

    blur(streetSceneGray, streetSceneGray, Size(3,3));

    cvtColor(streetScene, HSVimg, COLOR_BGR2HSV);

    namedWindow(windowName, WINDOW_AUTOSIZE);
    createTrackbar("Canny Thresholds", windowName, &lowThreshold, maxLowThreshold, on_trackbar);
    createTrackbar("Sobel Threshold", windowName, &sobelLowThreshold, maxsobelLowThreshold, on_trackbar);
    on_trackbar(0, 0);

    waitKey(0);

    imwrite("images/t2_.png", edgesFiltered);

    return 0;
}
```

Code organization

Function definitions and declarations

- Functions shall be:
  - Declared in a header file
  - Defined in a .cpp file
  - (Commonly) used in a .cpp file that #includes the header file

Function definitions and declarations

- my_max_filter is
  - Declared in Utilities_lab2.h
  - Declared in the .cpp file
- The second declaration is useless and bad design

```
#include <opencv2/highgui.hpp>
#include <opencv2/core.hpp>
#include <opencv2/imgproc.hpp>
#include <opencv2/imgcodecs.hpp>

#include <iostream>
#include <vector>

#include "Utilities_lab2.h"

using namespace std;

cv::Mat make_gray(char* argv);

cv::Mat my_max_filter(cv::Mat& image, int kernel_size);
cv::Mat my_min_filter(cv::Mat& image, int kernel_size);
unsigned char find_max(cv::Mat& subimg);
unsigned char find_min(cv::Mat& subimg);

void median_filter(cv::Mat& image, int kernel_size);
void gaussian_filter(cv::Mat& image, int kernel_size);

void histogram(cv::Mat& image);

void equalized_histogram(cv::Mat& image);


int main(int argc, char** argv)
{
    //Task 1
    //cv::Mat img_gray = make_gray(argv[1]);

    //Task 2
    //cv::Mat max_filtered_image = my_max_filter(img_gray, 7);
    //cv::Mat min_filtered_image = my_min_filter(max_filtered_image, 5);

    //Task 3
    //median_filter(min_filtered_image, 7);
    //gaussian_filter(min_filtered_image, 7);

    //Task 4
    //histogram(img_gray);

    //Task 5
    //equalized_histogram(img_gray);
}
```

Code organization – functions

- Code shall be organized into functions
- Each function implements one well-defined processing, e.g.:
  - Apply a given filter to an image
  - Visualize an image
  - …
- It is bad design to mix more than one activity into a single function

Code organization

- min_filter does **many** things

```
int min_filter(int kernel_size ,cv::Mat image_grayscale)
{
   int pad = kernel_size/2;
   int img_height = image_grayscale.size().height;
   int img_width = image_grayscale.size().width;

   cout << img_height << " " << img_width << endl;  //print out the height and width of

   cv::Mat test(cv::Size(img_width + pad*2, img_height + pad*2), CV_8UC1, Scalar(255));
   std::cout << test.size().height << "  " << test.size().width << endl; //print out the

   for (int i = pad ; i < test.size().height - pad ; i++)
   { ...
   }

    double max = 0;
    double min = 0;


   Mat imgFinal(img_height,img_width,  CV_8UC1);
   cout<<imgFinal.size<<'\n';
   for (int i = 0 ; i < test.size().width - 2*pad ; i++)
   { ...
   }

    imwrite("../min_filter.jpg", imgFinal);
    imshow("Min filter on grayscale image", imgFinal);

    Mat gaussian_image(img_height,img_width,  CV_8UC1);
    Mat median_image(img_height,img_width,  CV_8UC1);


    GaussianBlur( imgFinal, gaussian_image , Size( kernel_size,kernel_size ), 0, 0 );
    medianBlur ( imgFinal, median_image, kernel_size );


    imwrite("../min_gaussian_filter.jpg", gaussian_image);
    imwrite("../min_median_filter.jpg", median_image);

    imshow("Min filter on grayscale image with gaussian filter", gaussian_image);
    imshow("Min filter on grayscale image with median filter", median_image);
    waitKey(0);
```

Code organization

- maxFilter loads an image
  - This should be in a different function

```
const int MAX_KERNEL_LENGTH = 13;


void maxFilter(int kernelSize)
{
    if (kernelSize % 2 != 0)
    {
        cv::Mat1b img = cv::imread("image_grayscale.jpg");
        cv::Mat1b padded;
        cv::Mat1b dst;
        int top, bottom, left, right;
        top = (int) (0.005*img.rows); bottom = top;
        left = (int) (0.005*img.cols); right = left;
        cv::copyMakeBorder(img, padded, top, bottom, left, right,

        dst = cv::Mat1b(img.rows, img.cols, uchar(0));

        for (int c = 0; c < img.cols; ++c)
        {
            for (int r = 0; r < img.rows; ++r)
            {
                uchar largest = 0;
                for (int i = -kernelSize; i <= kernelSize; ++i)
                {
                    for (int j = -kernelSize; j <= kernelSize; ++j
                    {
                        uchar val = padded(kernelSize + r + i, ker
                        if (val > largest) largest = val;
                    }
                }
                dst(r, c) = largest;
            }
        }


        cv::imshow( "Max filter", dst );
        cv::waitKey(0);
    }
}
```

Code organization

- Everything in the main
  - Low level + high level

```
int main(int argc, char** argv){

    // *** BEGIN TASK 1 ***
    // visualizing image
    cv::Mat image = cv::imread(argv[1]);
    cv::namedWindow("Colour image", cv::WINDOW_NORMAL);
    cv::imshow("Colour image", image);
    cv::waitKey(0);

    // converting to grayscale
    cv::Mat image_grey = image.clone();
    cv::cvtColor(image, image_grey, cv::COLOR_RGB2GRAY, 0);
    cv::namedWindow("Grayscale image", cv::WINDOW_NORMAL);
    cv::imshow("Grayscale image", image_grey);
    cv::waitKey(0);

    // saving image
    cv::imwrite("image_grayscale.jpg", image_grey);
    // *** END TASK 1 ***

    // *** BEGIN TASK 2 ***
    // min filter
    cv::Mat min_image(image_grey.rows, image_grey.cols, CV_8UC1);
    min_filter(image_grey, min_image, 5); // no removing cables
    cv::namedWindow("Min filter",cv::WINDOW_NORMAL);
    cv::imshow("Min filter", min_image);
    cv::waitKey(0);

    // max filter
    cv::Mat max_image(image_grey.rows, image_grey.cols, CV_8UC1);
    max_filter(image_grey, max_image, 5); // no removing cables
    cv::namedWindow("Max filter",cv::WINDOW_NORMAL);
    cv::imshow("Max filter", max_image);
    cv::waitKey(0);

    // *** END TASK 2 ***

    // *** BEGIN TASK 3 ***
    // median filter
    cv::Mat median_image = cv::Mat(image_grey.rows, image_grey.cols, CV_8U);
    cv::medianBlur(image_grey, median_image, 5);
    cv::namedWindow("Median filter", cv::WINDOW_NORMAL);
    cv::imshow("Median filter", median_image);
    cv::waitKey(0);
```

Code organization

Just one main function

```cpp
int main(int argc, char** argv)
{
  //---------------------------------------------- Setup ----------------------------------------------

  //check if input image is provided
  if (argc < 2)
  {
    cout << "Error: no input image" << endl;
    return -1;
  }

  //create folder for images
  if (!filesystem::exists("images"))
  {
    filesystem::create_directory("images");
  }

  //---------------------------------------------- Task 1 ----------------------------------------------

  //read input image
  cv::Mat img = cv::imread(argv[1]);
  cv::imshow("Input", img);

  //apply gaussian blur
  cv::Mat blur;
  cv::GaussianBlur(img, blur, cv::Size(9, 9), 0);

  //create canny
  cv::Mat canny;
  int lower = 60;
  int upper = 200;
  cv::Canny(blur, canny, lower, upper);

  //create window
  cv::namedWindow("Canny");

  //create trackbar
  cv::createTrackbar("Lower Threshold", "Canny", &lower, 255, NULL); //cause warning, &lower should be 0 but in this way is
  cv::createTrackbar("Upper Threshold", "Canny", &upper, 255, NULL); //cause warning, &upper ...

  //to avoid error when closing window with 'X'
  try {
    while (true)
    {
      //get trackbar values
      lower = cv::getTrackbarPos("Lower Threshold", "Canny");
      upper = cv::getTrackbarPos("Upper Threshold", "Canny");

      //apply canny edge detector
      cv::Canny(blur, canny, lower, upper);

      //show output image
      cv::imshow("Canny", canny);

      //close window on ESC
      if (cv::waitKey(10) == 27) break;
    }
    cv::destroyWindow("Canny");
  }
  catch (const std::exception& e)
  {
    cout << e.what() << endl;
  }


  //save canny
  cv::imwrite("images/canny.jpg", canny);

  //---------------------------------------------- Task 2-3 ----------------------------------------------

  //define line vectors
  vector<cv::Vec2f> r_lines;
  vector<cv::Vec2f> l_lines;

  //apply hough transform
  cv::HoughLines(canny, r_lines, 1, CV_PI / 180, 100, 0, 0, 0, CV_PI / 3);
  cv::HoughLines(canny, l_lines, 1, CV_PI / 180, 100, 0, 0, -CV_PI / 3, 0);
```

```cpp
  //take lines
  cv::Vec2f r_line = r_lines[0];
  cv::Vec2f l_line = l_lines[0];

  //define line points
  float r_rho = r_line[0];
  float r_theta = r_line[1];
  float l_rho = l_line[0];
  float l_theta = l_line[1];

  //draw lines
  for (int i = 0; i < canny.rows; i++)
  {
    for (int j = 0; j < canny.cols; j++)
    {
      if (i > l_rho / sin(l_theta) - j / tan(l_theta) && i > r_rho / sin(r_theta) - j / tan(r_theta))
      {
        img.at<cv::Vec3b>(i, j) = cv::Vec3b(0, 0, 255);
      }
    }
  }

  cv::imshow("Task 2-3", img);
  cv::waitKey(0);

  //save image
  cv::imwrite("images/task_2_3.jpg", img);

  //---------------------------------------------- Task 4 ----------------------------------------------

  //create gray from blurred
  cv::Mat gray;
  cv::cvtColor(blur, gray, cv::COLOR_BGR2GRAY);

  //create circle vectors
  vector<cv::Vec3f> circles;

  //apply hough transform
  cv::HoughCircles(gray, circles, cv::HOUGH_GRADIENT_ALT, 1.5, 20, 300, 0.9, 5, 50);

  //draw circles
  for (int i = 0; i < circles.size(); i++)
  {
    cv::Point center(cvRound(circles[i][0]), cvRound(circles[i][1]));
    int radius = cvRound(circles[i][2]);
    cv::circle(img, center, radius, cv::Scalar(0, 255, 0), -1, 8, 0);
  }

  cv::imshow("Task 4", img);
  cv::waitKey(0);

  //save image
  cv::imwrite("images/task_4.jpg", img);

  return 0;
}
```

Code organization

A whole task
in one single function

Low- and high-level mixed

```cpp
void task3(){

    cv::Mat canny_edges;
    cv::blur(img_gray, canny_edges, cv::Size(blur_k_size, blur_k_size));
    cv::Canny(canny_edges, canny_edges, threshold1, threshold2, sobel_size);

    std::vector<cv::Vec2f> lines;
    int hough_threshold = 150;
    cv::HoughLines(canny_edges, lines, 1, CV_PI/180, hough_threshold, 0, 0);

    bool left_line = false;     // used to draw just one line for each side of the road
    bool right_line = false;

    cv::Mat detected_road;      // image in which we draw the hough lines of the road
    img.copyTo(detected_road);
    cv::Mat final;              // image in which we fill the area representing the road
    img.copyTo(final);

    for(size_t i=0; i<lines.size(); i++ )
    {
        if(left_line && right_line)     // if we already have the two lines we can break the cycle
            break;

        float rho = lines[i][0];
        float theta = lines[i][1];
        int deg_theta = theta*360/(2*CV_PI);

        if(deg_theta > 40 && deg_theta < 50){       // left side
            if(left_line)
                continue;
            else
                left_line = true;
        } else if(deg_theta > 125 && deg_theta < 145){      // right side
            if(right_line)
                continue;
            else
                right_line = true;
        } else {
            continue;                       // if the line has not the theta expected for the two lines searched we don't dra
        }
```

```cpp
        cv::Point pt1, pt2;
        double a = std::cos(theta);
        double b = std::sin(theta);
        double x0 = a*rho;
        double y0 = b*rho;
        pt1.x = cvRound(x0 + 1000*(-b));
        pt1.y = cvRound(y0 + 1000*(a));
        pt2.x = cvRound(x0 - 1000*(-b));
        pt2.y = cvRound(y0 - 1000*(a));

        cv::line(detected_road, pt1, pt2, cv::Scalar(0,0,255), 3, cv::LINE_AA);
    }

    cv::namedWindow(window_hough_lines);
    cv::imshow(window_hough_lines, detected_road);
    cv::waitKey(0);

    std::vector<int> start(img.rows), end(img.rows);    // points delimiting the pixels between the two lines for each row
    int intersection_row = 0;                           // we have to consider the area between the lines only under their intersec
    // at the end of the following cycle, start[i] and end[i] (for i>=intersection_row) indicate first and last pixels of the road

    for(int i=0; i<detected_road.rows; i++){
        for(int j=0; j<detected_road.cols; j++){
            cv::Vec3b current = detected_road.at<cv::Vec3b>(i,j);

            if(current[0] == 0 && current[1] == 0 && current[2] == 255){
                if(start[i] == 0){
                    start[i] = j;
                    end[i] = j;
                }

                while(current[0] == 0 && current[1] == 0 && current[2] == 255){     // since the line has thickness = 3 there could
                    end[i] = j;
                    j++;
                    current = detected_road.at<cv::Vec3b>(i,j);
                }

                int new_end = end[i];       // used to check if the intersection is in the current line

                while(j<img.cols-1 && !(current[0] == 0 && current[1] == 0 && current[2] == 255)){
                    j++;
                    current = detected_road.at<cv::Vec3b>(i,j);
                    new_end = j;
                }

                while(current[0] == 0 && current[1] == 0 && current[2] == 255){     // since the line has thickness = 3 there could
                    end[i] = j;
                    j++;
                    current = detected_road.at<cv::Vec3b>(i,j);
                }

                if(new_end == img.cols-1)       // if new_end points to the last column it means that the previous cycles didn't fi
                    intersection_row = i;       // and then this road contains the intersection
                else
                    end[i] = new_end;           // otherwise new_end points to the last point in the row before the second line

                break;
            }
        }
    }

    for(int i=intersection_row; i<detected_road.rows; i++){ // we have to fill the area between the two lines from the intersection
        for(int j=start[i]; j<end[i]; j++){
            final.at<cv::Vec3b>(i,j) = cv::Vec3b(0,0,255);
        }
    }

    cv::imshow(window_hough_lines, final);
    cv::waitKey(0);

}
```

Code organization

- Mean BGR value calculated in the callback
  - Could be useful in other parts of the code

```
void OnClick::mouseHandler(int event,int cols,int rows, int flags,void* param){

    // select left click mouse event
    if(event == cv::EVENT_LBUTTONDOWN){
        cv::Mat* pclicked_img = (cv::Mat*) param;
        cv::Mat clicked_img = (cv::Mat)* pclicked_img;
        cv::Vec3b bgr_vector = clicked_img.at<cv::Vec3b>(rows,cols);

        int b_sum = 0;
        int g_sum = 0;
        int r_sum = 0;

        int pos_cols = 0;
        int pos_rows = 0;
        int n_pixels = 0;

        for(int i = 0; i < KERNEL_SIDE; i++){
                for(int j = 0; j < KERNEL_SIDE; j++){

                    // compute the position of the pixel in the kernel with respect to the whole image coords
                    pos_cols = cols - 1 + i;
                    pos_rows = rows - 1 + j;

                    // compute the sum of all the pixel values in the kernel
                    if(!(pos_cols > clicked_img.cols || pos_rows > clicked_img.rows )){
                            bgr_vector = clicked_img.at<cv::Vec3b>(pos_rows,pos_cols);
                            b_sum +=static_cast<int>(bgr_vector[0]);
                            g_sum +=static_cast<int>(bgr_vector[1]);
                            r_sum +=static_cast<int>(bgr_vector[2]);

                            // increment number of visited pixels
                            n_pixels++;
                    }
                }
        }

        // compute the average
        int avg_b = b_sum/n_pixels;
        int avg_g = g_sum/n_pixels;
        int avg_r = r_sum/n_pixels;

        // creation of the mask
        cv::Mat mask = Kernel::createMask(avg_b, avg_g, avg_r, clicked_img, 15);
        cv::imshow("mask", mask );

        std::vector<int> compression_params;
        compression_params.push_back(cv::IMWRITE_PNG_COMPRESSION);
        compression_params.push_back(9);
        cv::imwrite("robocup_mask.jpg", mask, compression_params);
    }
}
```

Code repetition

- If you do not define functions properly you'll end up repeting the code

```
std::vector<cv::Vec2f> linesPositive;
cv::HoughLines(edges, linesPositive, 1, CV_PI / 180, 150, 0, 0, CV_PI * 44 / 180, CV_PI * 46 / 180);

std::vector<cv::Vec2f> linesNegative;
cv::HoughLines(edges, linesNegative, 1, CV_PI / 180, 150, 0, 0, -CV_PI * 50 / 180, -CV_PI * 40 / 180);

for (size_t i = 0; i < linesPositive.size(); i++) {
    float rho = linesPositive[i][0], theta = linesPositive[i][1];
    cv::Point pt1, pt2;
    double a = cos(theta), b = sin(theta);
    double x0 = a * rho, y0 = b * rho;
    pt1.x = cvRound(x0 + 1000 * (-b));
    pt1.y = cvRound(y0 + 1000 * (a));
    pt2.x = cvRound(x0 - 1000 * (-b));
    pt2.y = cvRound(y0 - 1000 * (a));
    cv::line(Hough_imgPos, pt1, pt2, cv::Scalar(0, 0, 255), 3, cv::LINE_AA);
    cv::line(Hough_img, pt1, pt2, cv::Scalar(0, 0, 255), 3, cv::LINE_AA);
    cv::line(mask, pt1, pt2, cv::Scalar(0, 0, 255), 3, cv::LINE_AA);

}

double m1 = 0;
double c1 = 0;
for (size_t i = 0; i < linesPositive.size(); ++i) {
    float rho = linesPositive[i][0];
    float theta = linesPositive[i][1];
    double x = rho * cos(theta);
    double y = rho * sin(theta);
    m1 = -cos(theta) / sin(theta);
    c1 = y - m1 * x;
}

for (size_t i = 0; i < linesNegative.size(); i++) {
    float rho = linesNegative[i][0], theta = linesNegative[i][1];
    cv::Point pt1, pt2;
    double a = cos(theta), b = sin(theta);
    double x0 = a * rho, y0 = b * rho;
    pt1.x = cvRound(x0 + 1000 * (-b));
    pt1.y = cvRound(y0 + 1000 * (a));
    pt2.x = cvRound(x0 - 1000 * (-b));
    pt2.y = cvRound(y0 - 1000 * (a));
    cv::line(Hough_imgNeg, pt1, pt2, cv::Scalar(0, 0, 255), 3, cv::LINE_AA);
    cv::line(Hough_img, pt1, pt2, cv::Scalar(0, 0, 255), 3, cv::LINE_AA);
    cv::line(mask, pt1, pt2, cv::Scalar(0, 0, 255), 3, cv::LINE_AA);
}

double m2 = 0;
double c2 = 0;
for (size_t i = 0; i < linesNegative.size(); ++i) {
    float rho = linesNegative[i][0];
    float theta = linesNegative[i][1];
    double x = rho * cos(theta);
    double y = rho * sin(theta);
    m2 = -cos(theta) / sin(theta);
    c2 = y - m2 * x;
}

cv::Point2f intersection = findIntersection(m1, c1, m2, c2);
```

Code organization

- Some functions have a role that is not clear

```
//Task1
Mat convertTwoGrayScale(Mat image){
    Mat image_grayscale;
    //convert the image to grayscale
    cvtColor(image, image_grayscale, COLOR_RGB2GRAY);
    //save image to file
    imwrite("../image_grayscale.jpg", image_grayscale);

    return image_grayscale;
}
```

Code reuse

- If you know the functions offered by the library you avoid re-implementations
  - `cv::Mat::clone()` and `copyTo()` are available
  - Why a static function??

```
static void callBack(int , void* )
{
    cv::Mat copyofimg(streetWindow.rows, streetWindow.cols, CV_8UC3);
    for(int i = 0; i < streetWindow.rows; ++i)
    {
        for(int j = 0; j < streetWindow.cols; ++j)
        {
            copyofimg.at<cv::Vec3b> (i, j)[0] = streetWindow.at<cv::Vec3b> (i, j)[0];
            copyofimg.at<cv::Vec3b> (i, j)[1] = streetWindow.at<cv::Vec3b> (i, j)[1];
            copyofimg.at<cv::Vec3b> (i, j)[2] = streetWindow.at<cv::Vec3b> (i, j)[2];
        }
    }
    cv::imshow(STREET_WINDOW, copyofimg);
```

Function arguments

Arguments

- When an argument is an output, you should pass it using
  - References, or
  - Pointers
- Otherwise you write on the copy
  - cv::Mat are different, but only regarding the buffer – other elements (e.g. image size) work like variables

```
#include <opencv2/highgui.hpp>
#include <opencv2/imgproc.hpp>
#include <opencv2/imgcodecs.hpp>
#include <math.h>

void maxFilter(cv::Mat src, cv::Mat dst, int kernel_size);
void minFilter(cv::Mat src, cv::Mat dst, int kernel_size);
```

Arguments

- Input arguments passed by reference that shall not be modified must be declared const
  - E.g.: const cv::Mat& src

```
namespace filters {
    /**
    * Generates a filtered version of an image using a maxFilter of given size.
    *
    * Takes a source image and applies for each pixel a max filtering with a squa
    * dimension nxn where n is the integer passed as size parameter, and it saves
    * image in the destination matrix. The filtering process updates the value of
    * the maximum value of its neighbors inside the kernel centered in its positi
    *
    * @param src Source image
    * @param dst Destination image ( Filtered image )
    * @param size Size of the kernel
    */
    int maxFilter(cv::Mat& src, cv::Mat& dst, int size);

    /**
    * Generates a filtered version of an image using a minFilter of given size.
    *
    * Takes a source image and applies for each pixel a min filtering with a squa
    * dimension nxn where n is the integer passed as size parameter, and it saves
    * image in the destination matrix. The filtering process updates the value of
    * the minimum value of its neighbors inside the kernel centered in its positi
    *
    * @param src Source image
    * @param dst Destination image ( Filtered image )
    * @param size Size of the kernel
    */
    int minFilter(cv::Mat& src, cv::Mat& dst, int size);
}
```

Arguments

- Same problem
- Additionally: kernel_size is an int
  - Why is it passed as a const reference?

```
/*
 * Checking if the number is even.
 *
 * */
bool even(int number){
  if (number % 2 == 0) return true;
  else return false;
}

/*
 * Implementing a max filter that manipulate
 * the pixels directly.
 * Working only if kernel_size is odd, otherwise
 * the function does not process the image.
 *
 * */
void max_filter(cv::Mat& image, const int& kernel_size){
  if (even(kernel_size))
  {
    printf("Error: even kernel size.");
    return;
  }
  int range = kernel_size / 2;
  cv::Mat max_img = image.clone();

  for (int x = 0; x < max_img.rows; x++)
  {
    for (int y = 0; y < max_img.cols; y++)
    {
      int max = 0;
      for (int i = -range; i <= kernel_size; i++)
      {
        for (int j = -range; j <= kernel_size; j++)
        {
          int value = (int)image.at<uchar>(x+i,y+j);
          if (value > max) max = value;
        }
      }
      max_img.at<uchar>(x,y) = max;
    }
  }

  cv::namedWindow("Display Max Filter Image", cv::WINDOW_AUTOSIZE);
    cv::imshow("Display Max Filter Image", max_img);
  //cv::imwrite("max_filter_k5.jpg", max_img);
}
```

Arguments

- Images passed by string

```
#include <iostream>
using namespace std;

int max_filter(string input_image, int kernel_size);
int min_filter(string input_image, int kernel_size);
```

Making code clear

Using high-level tools

- Use std::strings instead of C-style strings (that are array of chars)

```
// Print the image provided
namedWindow("Provided image");
imshow("Provided image", src);
waitKey(0);

// Canny image
// Declare trackbar parameters
const int T_slider_max = 1000;
namedWindow("Canny image");
char Trackbar1[1000];
char Trackbar2[1000];
// create trackbars and call to callback function
snprintf(Trackbar1, sizeof(Trackbar1),"Low T", T_slider_max);
createTrackbar(Trackbar1, "Canny image", &T1,T_slider_max, on_trackbar, &src);
snprintf(Trackbar2, sizeof(Trackbar2),"High T", T_slider_max);
createTrackbar(Trackbar2, "Canny image", &T2,T_slider_max, on_trackbar, &src);
// show the canny image
imshow("Canny image", src);
waitKey(0);
```

Unused variables

- Variables defined but not used
  - street_gray2
  - sobelx2

```
#include <iostream>
#include <opencv2/highgui.hpp>
#include <opencv2/imgproc.hpp>
#include <opencv2/opencv.hpp>

int main(int argc, char** argv)
{
    //Read Image in Grayscale
    cv::Mat street_gray = cv::imread("street_scene.png",cv::IMREAD_GRAYSCALE );
    cv::Mat street_gray2 = cv::imread("street_scene.png",cv::IMREAD_GRAYSCALE );
    if(street_gray.data==NULL){
        std::cout << "Image not found. Check if the image called street_scene.png is in executable the folder\n";
        return 1;
    }

    cv::Scalar min_color = cv::Scalar(250, 250, 250);
    cv::Scalar max_color = cv::Scalar(255, 255, 255);
    cv::Mat mask;
    cv::inRange(street_gray, min_color, max_color, mask);
    cv::imshow("mask", mask);
    cv::waitKey(0);

    //1. Low-pass filter for noise removal
    // cv::GaussianBlur( street_gray, street_gray, cv::Size(11,11) ,0,0);
    cv::GaussianBlur( mask, mask, cv::Size(11,11) ,0,0);
   // cv::medianBlur(mask,mask,3);

    // 2. Gradient calculation with Sobel
    cv::Mat sobelx,sobely,sobelxy,sobelx2;
    cv::Sobel(mask, sobelx ,CV_64F, 1, 0,3); // Sobel Edge Detection on the X axis
    cv::Sobel(mask, sobely,CV_64F, 0, 1); // Sobel Edge Detection on the Y axis
   // cv::Sobel(street_gray, sobelxy,CV_64F, 1,1,5); // Combined X and Y Sobel Edge Detection

    //3. Gradient thresholding – ∇f > T
    cv::threshold(sobelxy,sobelxy,100,101,cv::THRESH_TOZERO);
    cv::threshold(sobely,sobely,160,700,cv::THRESH_TOZERO);
    cv::threshold(sobelx,sobelx,100,150,cv::THRESH_TOZERO);

    //Display Sobel Edge Detection Images
    cv::imshow("Sobel X", sobelx);
    cv::waitKey(0);
    cv::imshow("Sobel Y", sobely);
    cv::waitKey(0);
    cv::imshow("Sobel X Y", sobelxy);
    cv::waitKey(0);

}
```

Variable naming

- Some variables have a non meaningful name

```
Mat img;
Mat imgcr;
Mat imggr;
Mat edge;
Mat edge1;
Mat edge2;
```

```
int t;
```

```
int a = 100;
```

- Do this only for loop indices

Comments

- Magic numbers and no comments make the code harder to be read by others

```
for (int i = 175; i < temp.rows; i++)
{
    bool started = 0;
    int startX = 0, stopX = 0;
    for (int j = 520; j < temp.cols-550; j++)
    {
```

- Better:
  - Associate a number with a const variable describing what it is
  - Add comments

Constants

- If you define a constant:
  - #define is the old C-style method

```
#define WHITE 255
#define BLACK 0
```

```
#include <opencv2/highgui.hpp>
#include <opencv2/imgproc.hpp>
#include <stdlib.h>
#include <iostream>


#define INPUT_PATH "input/street_scene.png"
#define WINDOW_NAME "Canny edge detection"

#define MAX_THRESHOLD 500
```

```
#include <iostream>

#include <opencv2/core/cvdef.h>
#include <opencv2/highgui.hpp>
#include <opencv2/imgproc.hpp>
#include <string>
#include <vector>
#define WINDOW_NAME "test"
#define WINDOW_LINES_NAME "mask"
#define WINDOW_MASKED_NAME "masked"
#define THRESHOLD 30
#define MAX_THRESH 1000

using namespace cv;
```

Constants

- If you define a constant:
  - Constant variables are more modern and better
    - They have a type!

```
const double PI = 3.14159265358979323846 3;
```

Casts

- C++-style `static_cast<>()` vs the C-style cast `()`
- `static_cast` is
  - More evident and clear
  - Less powerful
    - Better! Why?

```
int hist_w = 512, hist_h = 400;
int bin_w = cvRound((double)hist_w / histogram_size);
```

- Shall become

```
int bin_w = cvRound(static_cast<double>(hist_w) /
histogram_size);
```

Casts

- Other casts exist – the most powerful one being `reinterpret_cast`
  - Same as C-style cast
  - Anyway more evident and with a clear name
- When dealing with void* you should use reinterpret_cast

```
void onMouseClick(int event, int col, int row, int flags, void* userdata) {

    //  Work for both left and right click
    if (event == cv::EVENT_LBUTTONDOWN || event == cv::EVENT_RBUTTONDOWN) {
        cv::Mat* image = (cv::Mat*)userdata;
```

- Shall become

```
cv::Mat* image = reinterpret_cast<cv::Mat*> (userdata);
```

Code indentation

- Code indentation enhances clarity
- Use a good-looking spacing between
  - Operators
  - Expressions

```
#include <opencv2/highgui.hpp>
#include <opencv2/imgproc.hpp>
#include <opencv2/opencv.hpp>
#include <iostream>
#include <stdexcept>
//Source File Task 2
using namespace cv;
using namespace std;
Mat MinFilter(Mat iMin, int nsize){
if(nsize %2 ==0){
        throw invalid_argument("Not processed");
}
int Min =255;
int a = (sqrt(nsize)-1)/2;
Mat oMin(iMin.rows-(a*2), iMin.cols-(a*2), CV_8UC1);
for(int x = a; x < oMin.rows; x++){
    for(int y = a; y < oMin.cols; y++){
        Min=(int)iMin.at<uchar>(x,y);
        for(int i = -a; i <=a ; i++){
            for(int j = -a; j<=a ; j++){
                if (iMin.at<uchar>(x+i,y+j) < Min){
                Min = iMin.at<uchar>(x+i,y+j);
                    }
                }
            }
            oMin.at<uchar>(x,y)=Min;
        }
    }
    imwrite("image_grayscale_min.jpg", oMin);
}
```

Global variables

Global variables

- Global variables **must be avoided**!
  - Why?

```
8   #include "mylib.h"
9
10  cv::Mat src;
11  cv::Mat dst;
12  int top, bottom, left, right;
13  int borderType = cv::BORDER_CONSTANT;
14  int kernel_size;
15
16
17  void maxFilter(cv::Mat src, cv::Mat dst, int kernel_size)
18  {
19      //Check that the dim of the kernel is an odd number, if it's even the function d
20      if(kernel_size % 2 == 0)
21          return;
22
23      int border = floor(kernel_size/2.f);
24      int value=0;
25      top=border;
26      bottom=top;
27      left=border;
28      right=border;
29
30      cv::Mat resized_img (src.rows, src.cols, CV_8U);
31      cv::copyMakeBorder(src, resized_img, top, bottom, left, right, borderType, value
32
33      int i_resized;
34      int j_resized;
35
36
37      for(int i=0; i<src.rows; i++)
38      {
39          i_resized = i+border;
40          for(int j = 0; j<src.cols; j++)
41          {
42              cv::Mat ker(kernel_size, kernel_size, CV_8U);
43              j_resized = j+border;
44              cv::Range r(i_resized-border, i_resized+border);
45              cv::Range c(j_resized-border, j_resized+border);
46              ker=resized_img(r, c);
47              dst.at<unsigned char>(i,j)=maxValue(ker);
48
49          }
50      }
51
52  }
53
```

Global variables

- Global variables **must be avoided**!
  - Why?

```
#include "filter_minmaxfilter.h"
#include "histplot.h"

using namespace cv;
using namespace std;
string savename;


//---------Parameters-------------------------------------------

string filename = "img/image.jpg";
int scaledown = 4; //scaling down the image to accelerate process
bool runTask2 = false; // change to true to run task 2.
int kersizeV = 5; int kersizeH = 3; // the kernel size for the min max filter
int histsize = 256; // histogram size
int histW = 400; int histH = 512; // the size of the histogram canvas

//-------------------------------------------------------------


int main() {
    // Task 1
    Mat img = imread(filename);
    Mat greyimg;
    cvtColor(img, greyimg, COLOR_BGR2GRAY); // Convert the image into gray scale
    //scale down iamge
    resize(greyimg, greyimg, Size(round((double)greyimg.cols / scaledown), round((dou
```

Global variables

- Global variables bypass any incapsulation mechanism
- They bypass the argument passing mechanism
- They make debug **very** complicated
- Quick and dirty solution, very bad design

Global variables as constants

- Global variables might be used when defining constants
- Check the constants that are provided by math libraries / OpenCV

```
const double PI = 3.14159265358979323846 3;
```

C++ style guide

- Google has a good C++ style guide:

https://google.github.io/styleguide/cppguide.html

- Includes naming conventions
  - Make the code uniform among multiple developers
  - Very useful for the final project

Reporting problems

Reporting problems

- When you report a problem, stating "This does not work" is not enough
- You should:
  - Determine the source of the problem (e.g., line of code)
  - Determine the phase of the problem (e.g., compile? Link? Execution?)
  - Find suitable keywords for describing the problem
    - Key element for a successful search on the internet

UNIVERSITÀ DEGLI STUDI DI PADOVA

Lab: feedback, code design, open discussion

Stefano Ghidoni

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

INTELLIGENT AUTONOMOUS SYSTEMS LAB