实验二 OpenCV 基础

2151094 宋正非

1. 实验目标

（1）了解并熟悉 OpenCV 的基础操作，比如 Mat(), imread(), imshow(), imwrite() 等函数。

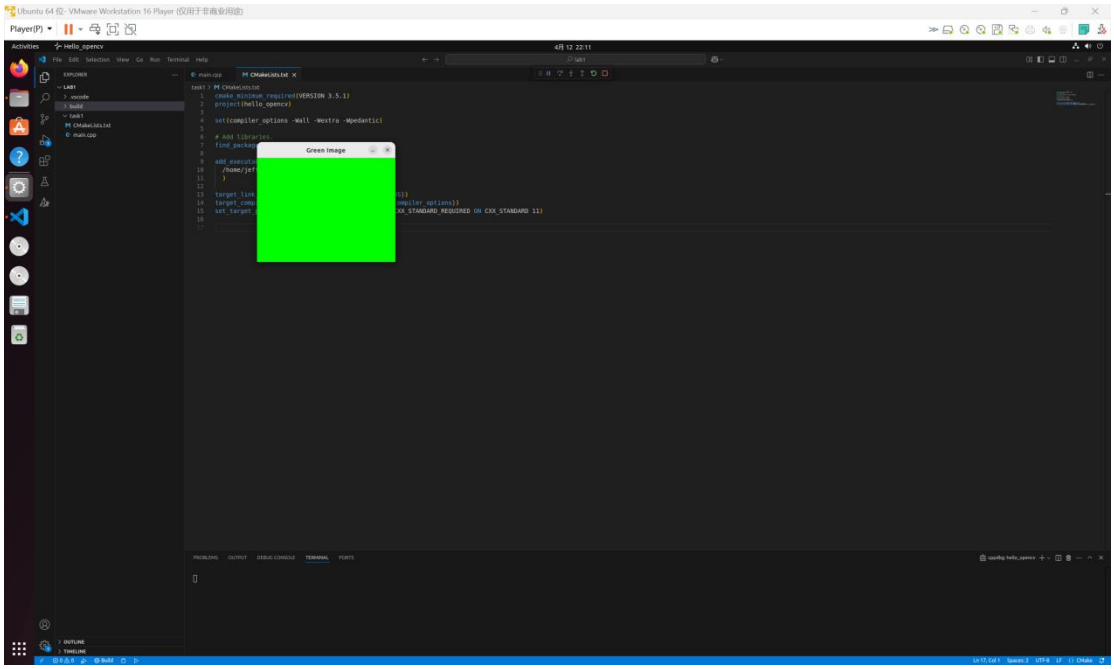
（2）获取实时的视频并处理。

2. OpenCV 基础操作

创建一个 320x240 的 OpenCV 数组，填充绿色，并用 imshow 显示它。

#main.cpp

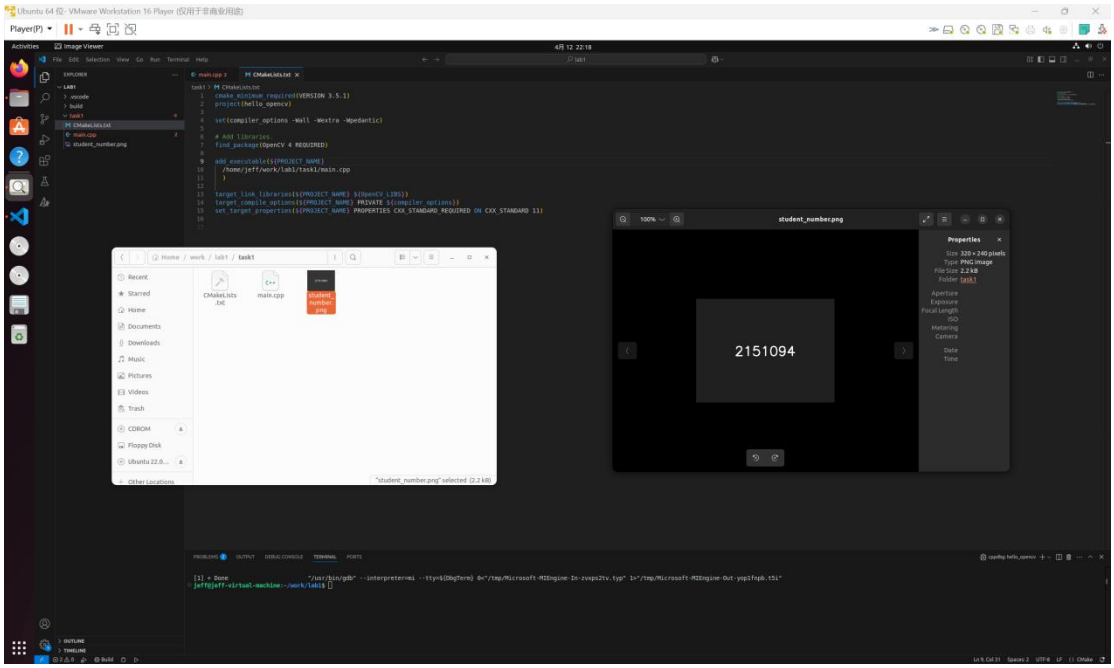
```
#include <opencv2/opencv.hpp>
#include <string>

void create_green(){
    cv::Mat image(240, 320, CV_8UC3, cv::Scalar(0, 255, 0));
    cv::imshow("Green Image", image);
    cv::waitKey(0);
}

int main() {
    create_green();
    return 0;
}
```



创建一个 320x240 的图像，填充深灰色（RGB=32,32,32）背景，并在其上写上你的学号。然后将图像保存为 PNG 文件。

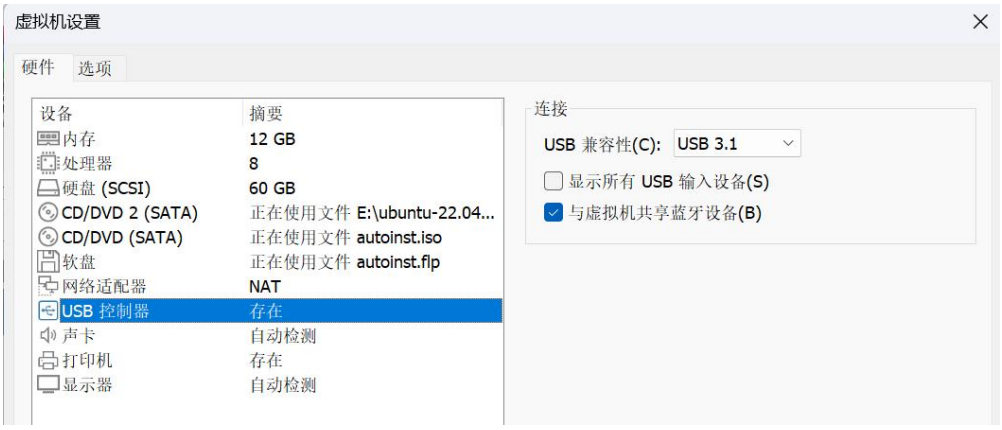
#main.cpp

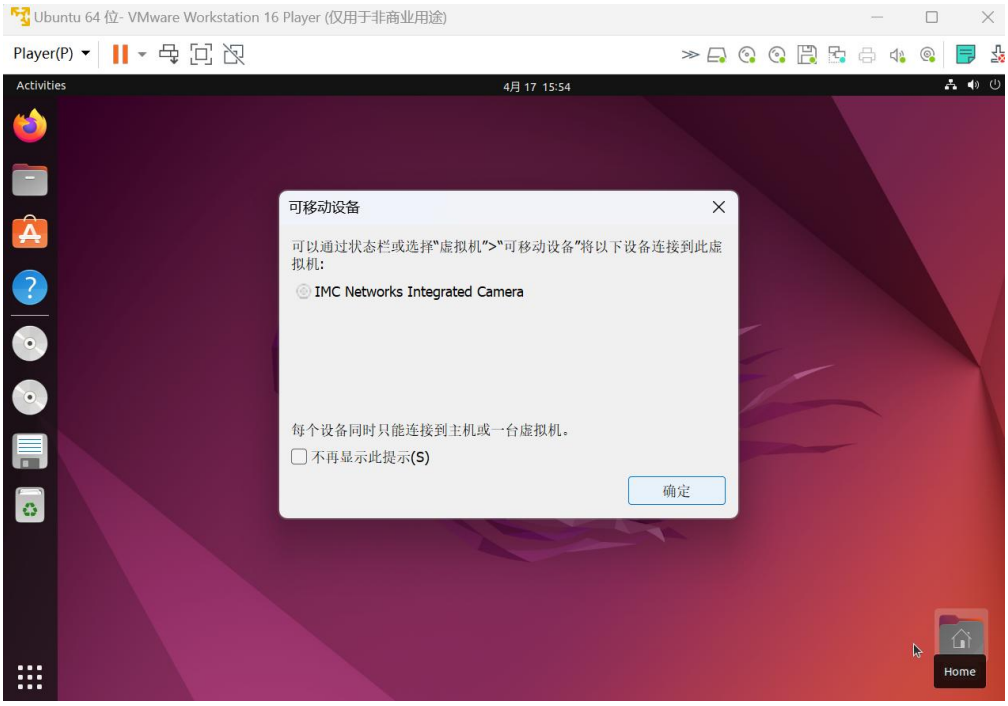
```
#include <opencv2/opencv.hpp>
#include <string>

void create_dark(){
    cv::Mat image(240, 320, CV_8UC3, cv::Scalar(32, 32, 32));

    std::string student_number = "2151094";
    int font_face = cv::FONT_HERSHEY_SIMPLEX;
    double font_scale = 1;
    int thickness = 2;
    cv::Scalar text_color(255, 255, 255);

    int baseline = 0;
    cv::Size text_size = cv::getTextSize(student_number, font_face, font_scale, thickness, &baseline);
    cv::Point text_org((image.cols - text_size.width) / 2, (image.rows + text_size.height) / 2);

    cv::putText(image, student_number, text_org, font_face, font_scale, text_color, thickness);

    cv::imwrite("/home/jeff/work/lab1/task1/student_number.png", image);
}


int main() {
   create_dark();
   return 0;
}
```

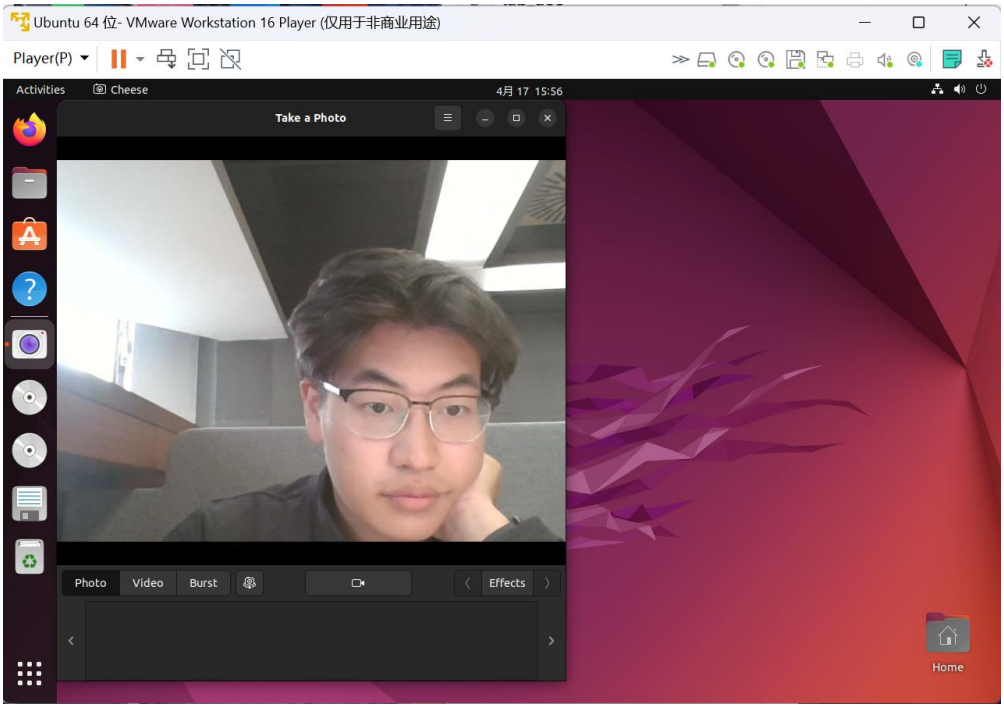


3. 获取实时视频并处理

首先，修改 VMware 关于 USB 控制器兼容性的设置，并在 Ubuntu 中找到并连接可移动设备中的相机。打开 Cheese 应用，检查并确保电脑自带的相机能够正常使用。


虚拟机设置
硬件
选项
设备
摘要
内存
12 GB
处理器
8
硬盘 (SCSI)
60 GB
CD/DVD 2 (SATA)
正在使用文件 E:\ubuntu-22.04...
CD/DVD (SATA)
正在使用文件 autoinst.iso
软盘
正在使用文件 autoinst.flp
网络适配器
NAT
USB 控制器
存在
声卡
自动检测
打印机
存在
显示器
自动检测
连接
USB 兼容性(C):
USB 3.1
显示所有 USB 输入设备(S)
与虚拟机共享蓝牙设备(B)


Ubuntu 64 位- VMware Workstation 16 Player (仅用于非商业用途)
Player(P)
Activities
4月 17 15:54
可移动设备
可以通过状态栏或选择"虚拟机">"可移动设备"将以下设备连接到此虚拟机:
IMC Networks Integrated Camera
每个设备同时只能连接到主机或一台虚拟机。
不再显示此提示(S)
确定
Home

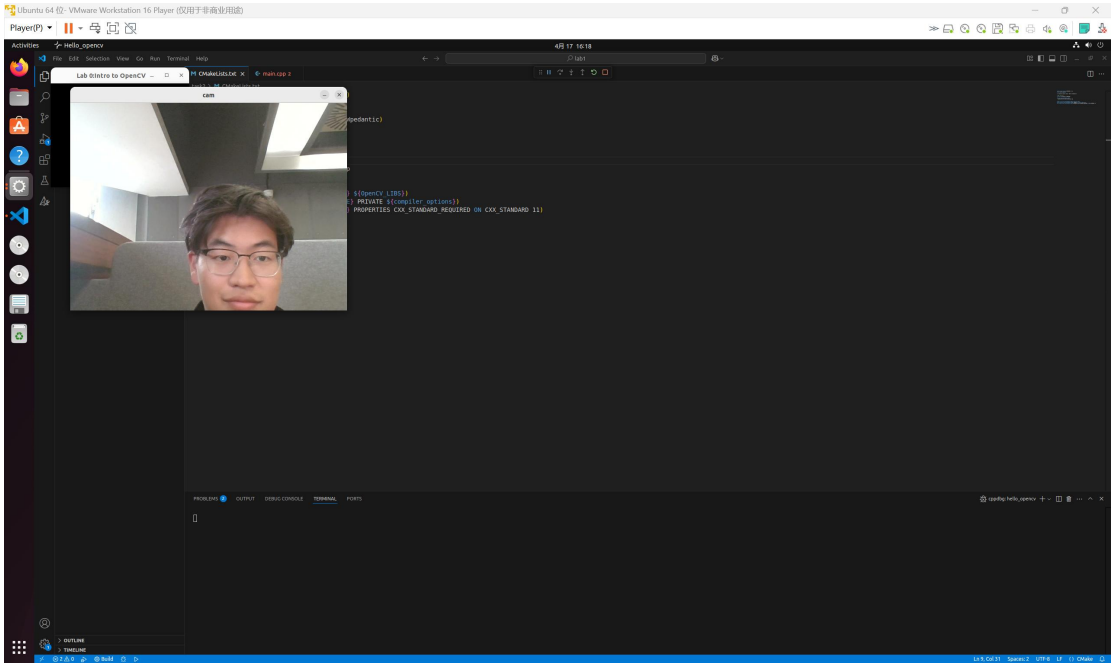

Ubuntu 64 位- VMware Workstation 16 Player (仅用于非商业用途)
Player(P)
Activities
Cheese
4月 17 15:56
Take a Photo
Photo
Video
Burst
Effects
Home

随后，使用教程提供的示例代码获取并播放实时视频。

#main.cpp

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

int main(int argc, char *argv[])
{
   cv::VideoCapture input_stream(0);
   if(!input_stream.isOpened())
   {
      std::cerr << "Could not open camera\n";
      return EXIT_FAILURE;
   }
   const std::string window_title = "Lab 0:Intro to OpenCV";
   cv::namedWindow(window_title, cv::WINDOW_NORMAL);
   cv::Mat frame;

   while(true)
   {
      input_stream >> frame;
      if(frame.empty())
      {break;}

      cv::imshow("cam", frame);
      if(cv::waitKey(15) >=0)
      {
         break;
      }
   }
}
```

其中，一开始“input”错写为“imput”导致报错。改正单词拼写错误后，使用 Cmake“Configure, build and debug”，成功获取实时视频。



随后，需要对实时获取的视频进行实时处理，包括将彩色视频转换为灰度、反转颜色、显示实时视频的直方图与使用 Canny 边缘检测。

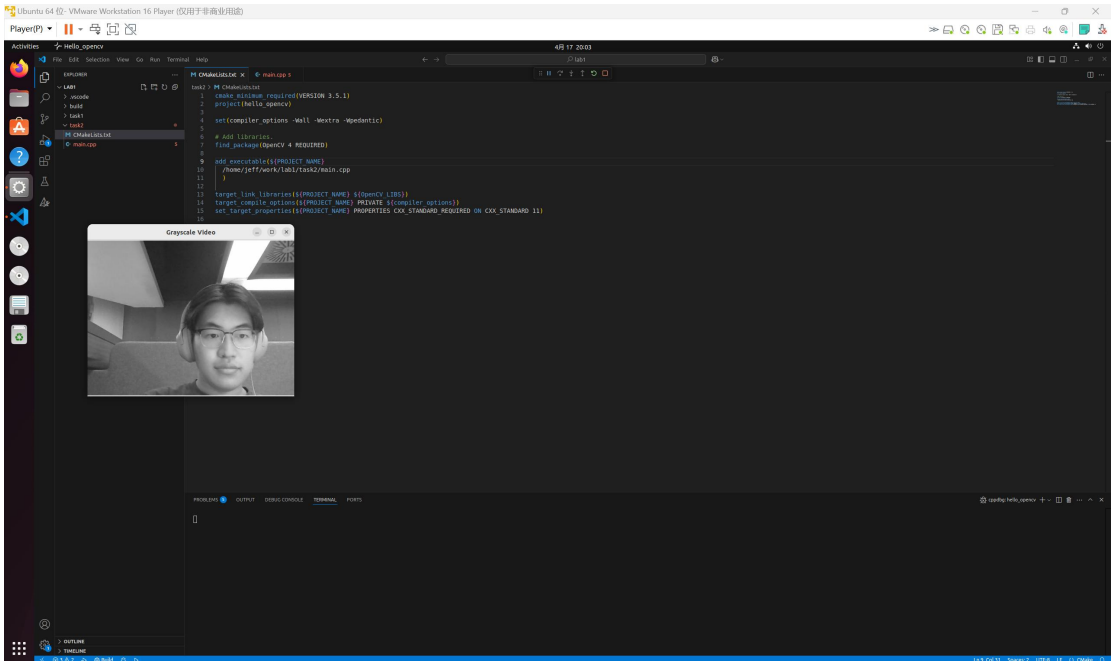
Try following processing functions with help of search engines or DeepSeek:

- Convert color video to grayscale
- Invert color
- Display Histogram of the real-time video
- Using Canny edge detection

使用如下代码**将彩色视频转换为灰度**。 使用 cv::cvtColor(frame, gray_frame, cv::COLOR_BGR2GRAY);可以将彩色帧转换为灰度帧。

#main.cpp

```
#include "opencv2/highgui.hpp"
#include "opencv2/imgproc.hpp"  // 需要添加 imgproc 头文件用于颜色转换
#include <iostream>

int main(int argc, char *argv[])
{
   cv::VideoCapture input_stream(0);
   if(!input_stream.isOpened())
   {
      std::cerr << "Could not open camera\n";
      return EXIT_FAILURE;
   }

   const std::string window_title = "Grayscale Video";
   cv::namedWindow(window_title, cv::WINDOW_NORMAL);

   cv::Mat frame;
   cv::Mat gray_frame;  // 用于存储灰度帧

   while(true)
   {
      input_stream >> frame;
      if(frame.empty())
         break;

      // 将彩色帧转换为灰度帧
      cv::cvtColor(frame, gray_frame, cv::COLOR_BGR2GRAY);

      // 显示灰度视频
      cv::imshow(window_title, gray_frame);

      if(cv::waitKey(15) >= 0)
         break;
   }
   return EXIT_SUCCESS;
}
```

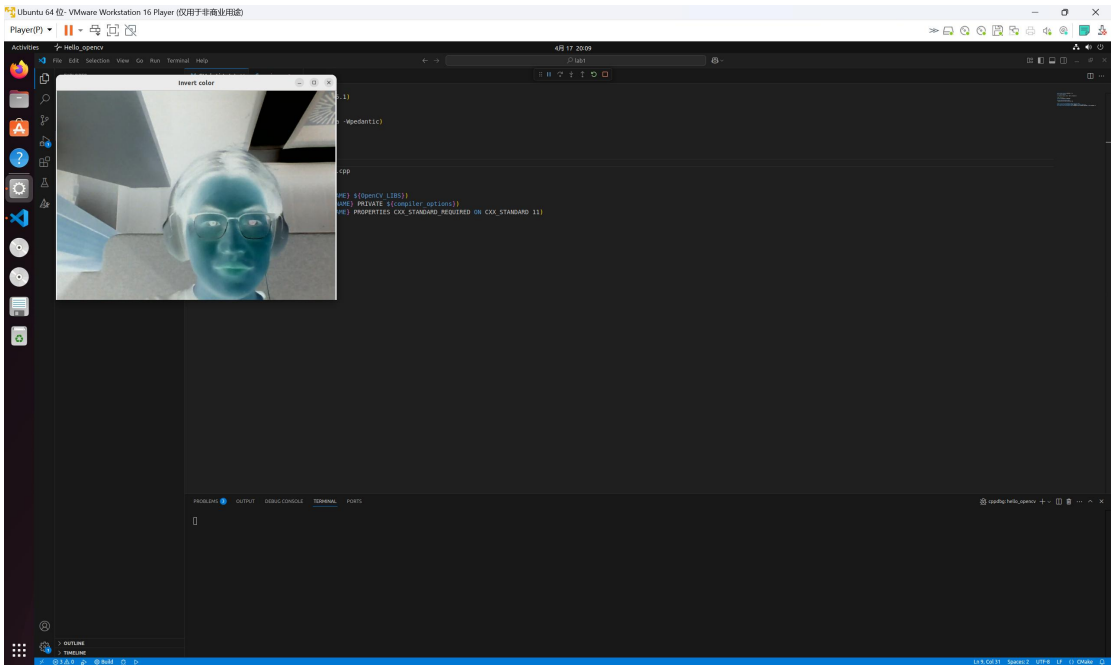


颜色反转。将每个像素的颜色值取反，比如 BGR 三个通道的值分别用 255 减去当前值。这样处理后，图像就会呈现出类似照片底片的效果。使用如下代码反转颜色。

#main.cpp

```
#include "opencv2/highgui.hpp"
#include "opencv2/core.hpp"  // 包含核心操作函数
#include <iostream>

int main(int argc, char *argv[])
{
   cv::VideoCapture input_stream(0);
   if(!input_stream.isOpened())
   {
      std::cerr << "Could not open camera\n";
      return EXIT_FAILURE;
   }
   const std::string window_title = "Invert color";
   cv::namedWindow(window_title, cv::WINDOW_NORMAL);
   cv::Mat frame;
   cv::Mat inverted_frame;  // 新增反转帧存储

   while(true)
   {
      input_stream >> frame;
      if(frame.empty())
         break;

      // 颜色反转操作（核心修改）
      cv::bitwise_not(frame, inverted_frame);

      cv::imshow(window_title, inverted_frame);  // 显示反转后的帧
      if(cv::waitKey(15) >= 0)
         break;
```

```
    }
    return EXIT_SUCCESS;
}
```



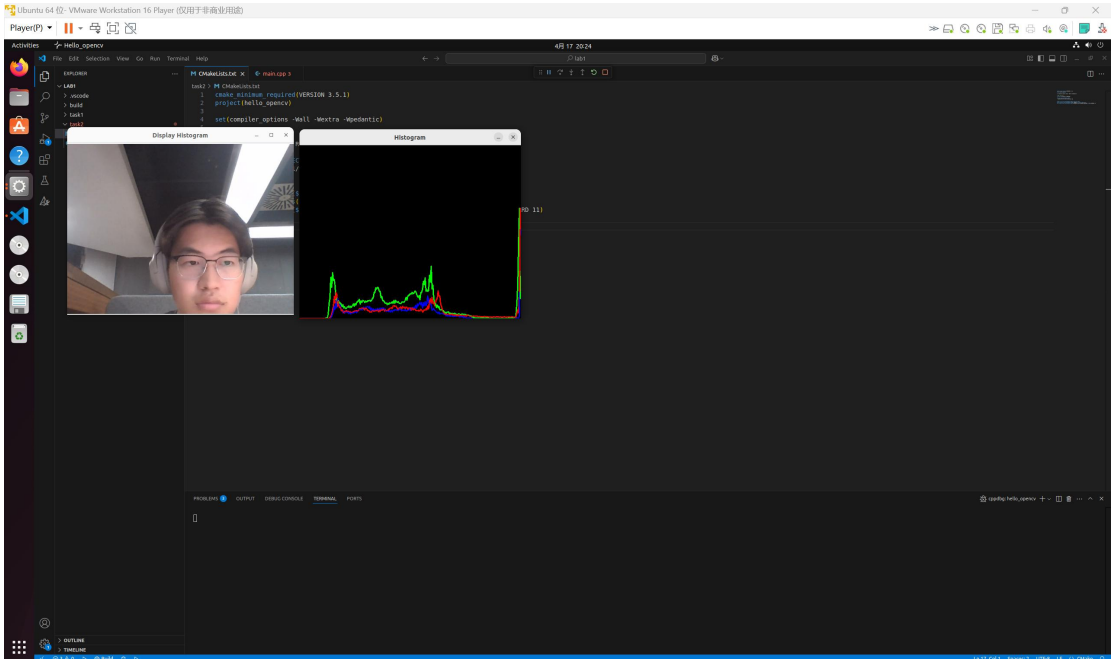
显示实时视频的直方图。直方图是用来描述图像中像素值分布的图表。图像的像素值范围从 0 到 255（对于 8 位灰度图），直方图的每个条形显示了每个像素值的出现频率。每一帧的图像有三个颜色通道：蓝色、绿色和红色，这些颜色通道存储在 bgr_planes 向量中。使用 cv::calcHist 函数来计算每个通道的直方图。每个通道的直方图是一个一维数组，表示每个像素值的频率。为了展示直方图，首先创建一个空的图像来绘制每个颜色通道的直方图。使用 cv::line 函数在这个图像上绘制每个通道的直方图。每一条线代表一个颜色通道的频率，颜色分别为蓝色、绿色和红色。bin_w 是每个直方图条形的宽度，hist_w 是直方图图像的宽度，hist_h 是图像的高度。

#main.cpp

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

int main(int argc, char *argv[])
{
    // Open the camera stream
    cv::VideoCapture input_stream(0);
    if (!input_stream.isOpened())
    {
        std::cerr << "Could not open camera\n";
        return EXIT_FAILURE;
    }
```

```
const std::string window_title = "Display Histogram";
cv::namedWindow(window_title, cv::WINDOW_NORMAL);
cv::Mat frame;

while (true)
{
    // Capture a frame from the video stream
    input_stream >> frame;
    if (frame.empty())
    {
        break;
    }

    // Convert the frame to grayscale for histogram calculation
    cv::Mat gray_frame;
    cv::cvtColor(frame, gray_frame, cv::COLOR_BGR2GRAY);

    // Calculate the histogram of the grayscale image
    std::vector<cv::Mat> bgr_planes;
    cv::split(frame, bgr_planes);

    // Define the number of bins and the range of pixel intensities
    int hist_size = 256; // For 8-bit images, there are 256 possible intensity values
    float range[] = { 0, 256 }; // Pixel values range from 0 to 255
    const float* hist_range = { range };

    // Calculate histograms for each channel (BGR)
    cv::Mat b_hist, g_hist, r_hist;
    cv::calcHist(&bgr_planes[0], 1, 0, cv::Mat(), b_hist, 1, &hist_size, &hist_range);
    cv::calcHist(&bgr_planes[1], 1, 0, cv::Mat(), g_hist, 1, &hist_size, &hist_range);
    cv::calcHist(&bgr_planes[2], 1, 0, cv::Mat(), r_hist, 1, &hist_size, &hist_range);

    // Normalize the histograms
    cv::normalize(b_hist, b_hist, 0, 255, cv::NORM_MINMAX);
    cv::normalize(g_hist, g_hist, 0, 255, cv::NORM_MINMAX);
    cv::normalize(r_hist, r_hist, 0, 255, cv::NORM_MINMAX);

    // Create an image to display the histogram
    int hist_w = 512;
    int hist_h = 400;
    int bin_w = cvRound((double)hist_w / hist_size);

    cv::Mat hist_image(hist_h, hist_w, CV_8UC3, cv::Scalar(0, 0, 0));

    // Draw the histograms for each color channel
    for (int i = 1; i < hist_size; i++)
    {
        cv::line(hist_image, cv::Point(bin_w * (i - 1), hist_h - cvRound(b_hist.at<float>(i - 1))),
                 cv::Point(bin_w * i, hist_h - cvRound(b_hist.at<float>(i))),
                 cv::Scalar(255, 0, 0), 2, 8, 0);
        cv::line(hist_image, cv::Point(bin_w * (i - 1), hist_h - cvRound(g_hist.at<float>(i - 1))),
                 cv::Point(bin_w * i, hist_h - cvRound(g_hist.at<float>(i))),
                 cv::Scalar(0, 255, 0), 2, 8, 0);
        cv::line(hist_image, cv::Point(bin_w * (i - 1), hist_h - cvRound(r_hist.at<float>(i - 1))),
                 cv::Point(bin_w * i, hist_h - cvRound(r_hist.at<float>(i))),
                 cv::Scalar(0, 0, 255), 2, 8, 0);
    }

    // Display the original frame and the histogram
    cv::imshow(window_title, frame);
```

```
        cv::imshow("Histogram", hist_image);

        // Exit the loop if any key is pressed
        if (cv::waitKey(15) >= 0)
        {
            break;
        }
    }

    return 0;
}
```

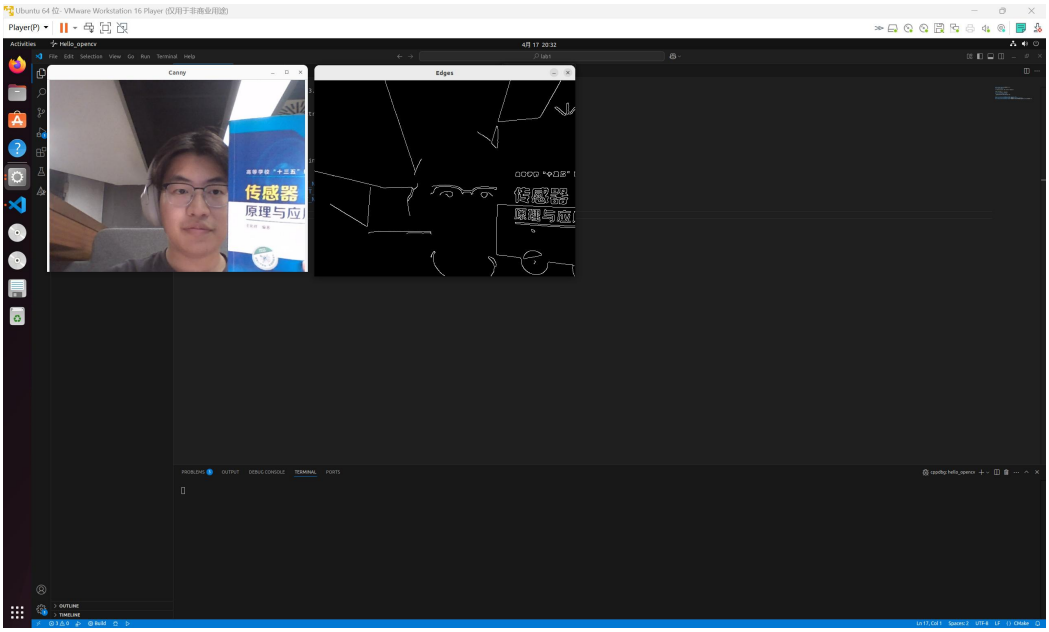


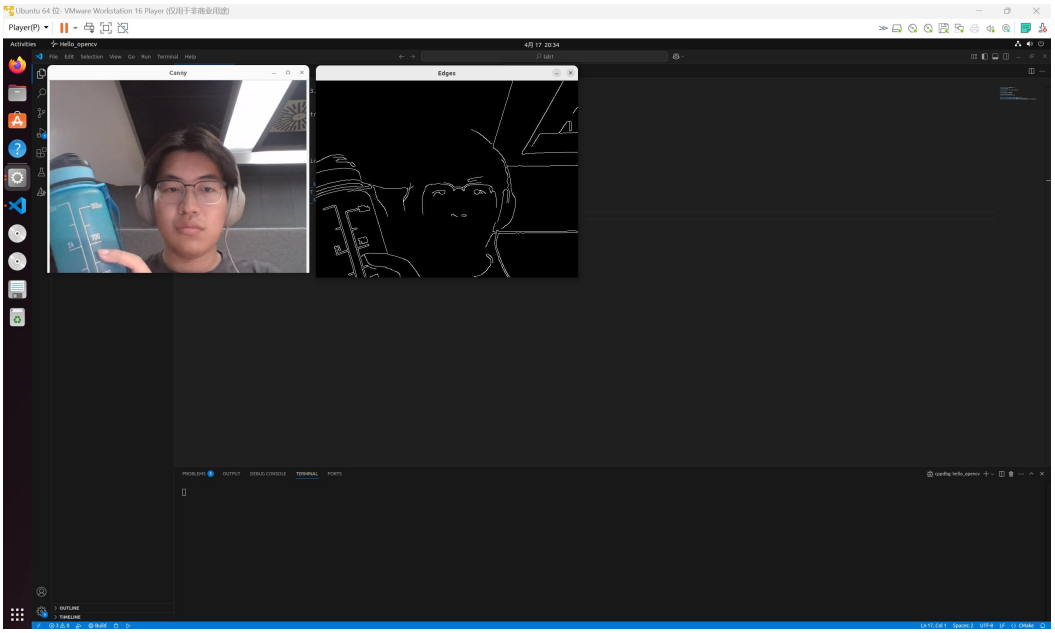
Canny 边缘检测。使用 cv::GaussianBlur 对图像进行平滑处理。这样可以减少噪声，提高边缘检测的准确性。使用 cv::Canny 函数对模糊后的图像进行边缘检测，low_threshold 和 high_threshold 是 Canny 算法中的低、高阈值，用来控制检测到的边缘的强度。

#main.cpp

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

int main(int argc, char *argv[])
{
    // 打开摄像头视频流
    cv::VideoCapture input_stream(0);
    if (!input_stream.isOpened())
    {
        std::cerr << "Could not open camera\n";
        return EXIT_FAILURE;
    }

    const std::string window_title = "Canny";
    cv::namedWindow(window_title, cv::WINDOW_NORMAL);
```

```
    cv::Mat frame;

    while (true)
    {
        // 从摄像头捕获一帧
        input_stream >> frame;
        if (frame.empty())
        {
            break;
        }

        // 将图像转换为灰度图像，因为 Canny 边缘检测通常在灰度图像上进行
        cv::Mat gray_frame;
        cv::cvtColor(frame, gray_frame, cv::COLOR_BGR2GRAY);

        // 使用高斯模糊来去噪，减少边缘检测中的噪声影响
        cv::Mat blurred_frame;
        cv::GaussianBlur(gray_frame, blurred_frame, cv::Size(5, 5), 1.5);

        // 使用 Canny 算法进行边缘检测
        cv::Mat edges;
        int low_threshold = 50; // 低阈值
        int high_threshold = 150; // 高阈值
        cv::Canny(blurred_frame, edges, low_threshold, high_threshold);

        // 显示原始视频帧和边缘检测结果
        cv::imshow(window_title, frame);
        cv::imshow("Edges", edges);

        // 如果按下任意键则退出
        if (cv::waitKey(15) >= 0)
        {
            break;
        }
    }

    return 0;
}
```




Ubuntu 64 位- VMware Workstation 16 Player (仅用于非商业用途)
Player(P)
Activities
Hello_opencv
4月 17 20:34
Canny
Edges
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
OUTLINE
TIMELINE