第二次培训：OpenCV C++ 图像处理基础

OpenCV C++ 图像处理基础

本文档总结了常见的 OpenCV 图像处理操作，涵盖：

- 图像读取与显示
- 灰度化
- 图像滤波（均值 / 高斯 / 中值）
- 二值化处理
- 边缘检测
- HSV 颜色空间处理
- 形态学操作
- 直方图均衡化

所有示例均基于 C++ + OpenCV

---

1. 基础操作：图像读取与显示

```cpp
#include <opencv2/opencv.hpp>
using namespace cv;

int main() {
    // 读取图像
    Mat img = imread("example.jpg");

    if (img.empty()) {
        std::cout << "无法读取图像!" << std::endl;
        return -1;
    }

    // 显示图像
    imshow("原图", img);
    waitKey(0);
    return 0;
}
```

📌 **说明**：

- `imread()` 读取图像，可指定灰度或彩色。
- `imshow()` 显示窗口。
- `waitKey(0)` 等待按键，否则窗口会立即关闭。

---

2. 图像灰度化

```
Mat gray;
cvtColor(img, gray, COLOR_BGR2GRAY);
imshow("灰度图", gray);
```

📌 **说明**：

- `cvtColor()` 用于颜色空间转换。
- 灰度化减少计算复杂度，常作为后续处理输入。

---

3. 图像滤波

(1) 均值滤波

```
Mat blurImg;
blur(img, blurImg, Size(5, 5));
imshow("均值滤波", blurImg);
```

(2) 高斯滤波

```
Mat gaussianImg;
GaussianBlur(img, gaussianImg, Size(5, 5), 1.5);
imshow("高斯滤波", gaussianImg);
```

(3) 中值滤波

```
Mat medianImg;
medianBlur(img, medianImg, 5);
imshow("中值滤波", medianImg);
```

📌 **区别**：

- 均值滤波：邻域取平均，模糊边缘。
- 高斯滤波：加权平均，更自然。
- 中值滤波：去除椒盐噪声效果好。

---

4. 图像二值化（阈值处理）

```
Mat binary, adaptive, gray;
cvtColor(img, gray, COLOR_BGR2GRAY);

// 全局阈值
threshold(gray, binary, 128, 255, THRESH_BINARY);
imshow("全局阈值二值化", binary);

// 自适应阈值
adaptiveThreshold(gray, adaptive, 255, ADAPTIVE_THRESH_GAUSSIAN_C,
                  THRESH_BINARY, 11, 2);
imshow("自适应阈值二值化", adaptive);
```

📌 **说明**：

- 全局阈值：整幅图像使用一个阈值。
- 自适应阈值：局部计算，更适合光照不均情况。

---

5. 边缘检测（Canny 算法）

```
Mat edges;
Canny(gray, edges, 100, 200);
imshow("Canny边缘检测", edges);
```

📌 **说明**：

- `Canny()` 通过梯度计算边缘。

- 参数：低阈值 / 高阈值。

---

6. HSV 颜色空间处理

```
Mat hsv, mask;
cvtColor(img, hsv, COLOR_BGR2HSV);

// 提取红色区域
inRange(hsv, Scalar(0, 100, 100), Scalar(10, 255, 255), mask);
imshow("红色掩膜", mask);
```

📌 **说明**：

- HSV 模型分离色彩与亮度，更适合颜色检测。
- `inRange()` 提取指定范围颜色。

---

7. 形态学操作

```
Mat dilateImg, erodeImg, morphImg;
Mat kernel = getStructuringElement(MORPH_RECT, Size(5, 5));

// 膨胀
dilate(binary, dilateImg, kernel);

// 腐蚀
erode(binary, erodeImg, kernel);

// 开运算（先腐蚀再膨胀）
morphologyEx(binary, morphImg, MORPH_OPEN, kernel);

imshow("膨胀", dilateImg);
imshow("腐蚀", erodeImg);
imshow("开运算", morphImg);
```

📌 **说明**：

- 膨胀：扩大白色区域。
- 腐蚀：缩小白色区域。

- 开运算：去除小噪点。
- 闭运算：填充小黑洞。

---

8. 直方图均衡化

```
Mat equalized;
equalizeHist(gray, equalized);
imshow("直方图均衡化", equalized);
```

📌 **说明**：

- 提高图像对比度。
- 只能作用于单通道（灰度图）。

核心部分：边缘检测与轮廓提取

一、为什么要做边缘检测

图像处理的目的通常是 **识别**（是什么）和 **分割**（在哪里）。

- **分割任务**：将目标与背景区分开。
- **边缘定义**：目标和背景的分界线就是边缘，多个连续的边缘点形成轮廓。
- **意义**：通过检测边缘，可以将目标区域提取出来，作为后续分析的基础。

二、边缘检测的方法

1. 二值化（阈值分割）

通过设定亮度阈值，将像素分为目标和背景，二者的交界即为边缘。

```cpp
Mat gray, binary;
cvtColor(img, gray, COLOR_BGR2GRAY);

// 全局阈值
threshold(gray, binary, 128, 255, THRESH_BINARY);
imshow("二值化结果", binary);
```

📌 **适用场景**：目标与背景亮度差异大。

---

2. 自适应二值化

环境光照不同会影响图像亮度，自适应方法根据局部区域动态确定阈值。

```cpp
Mat adaptive;
adaptiveThreshold(gray, adaptive, 255, ADAPTIVE_THRESH_GAUSSIAN_C,
    THRESH_BINARY, 11, 2);
imshow("自适应二值化", adaptive);
```

📌 **适用场景**：光照不均的图像。

---

3. 基于梯度的边缘检测

通过计算亮度变化率（梯度），检测亮度变化大的位置作为边缘。

(1) Sobel 算子

```cpp
Mat gradX, gradY, grad;
Sobel(gray, gradX, CV_16S, 1, 0);
Sobel(gray, gradY, CV_16S, 0, 1);
convertScaleAbs(gradX, gradX);
convertScaleAbs(gradY, gradY);
addWeighted(gradX, 0.5, gradY, 0.5, 0, grad);
imshow("Sobel 边缘", grad);
```

(2) Canny 算法

Canny 是一种改进的梯度边缘检测，效果较好。

```cpp
Mat edges;
Canny(gray, edges, 100, 200);
imshow("Canny 边缘", edges);
```

📌 **适用场景**：亮度差异较明显，但存在渐变区域。

---

4. 颜色边缘检测

当目标与背景的 **颜色差异大于亮度差异** 时，先将图像转到 HSV 空间，在 H 通道上检测边缘。

```cpp
Mat hsv, hsvChannels[3];
cvtColor(img, hsv, COLOR_BGR2HSV);
split(hsv, hsvChannels); // 分离 H, S, V 三个通道

// 在 H 通道上做 Canny
Mat colorEdges;
Canny(hsvChannels[0], colorEdges, 100, 200);
imshow("颜色边缘 (H 通道)", colorEdges);
```

三、边缘检测后的后处理

边缘检测的结果往往不完美（边缘断裂、噪声多）。
因此需要后处理，步骤如下：

1. 形态学操作（开闭运算）

- **开运算**：去除小噪声，连接断裂区域。
- **闭运算**：填补小洞，消除孤立区域。

```cpp
Mat morph;
Mat kernel = getStructuringElement(MORPH_RECT, Size(3, 3));
morphologyEx(edges, morph, MORPH_CLOSE, kernel);
imshow("形态学处理", morph);
```

---

2. 轮廓提取

OpenCV 提供 `findContours` 来提取边缘轮廓。

```cpp
vector<vector<Point>> contours;
vector<Vec4i> hierarchy;
findContours(morph, contours, hierarchy, RETR_EXTERNAL, CHAIN_APPROX_SIMPLE);

// 绘制轮廓
Mat contourImg = Mat::zeros(img.size(), CV_8UC3);
for (size_t i = 0; i < contours.size(); i++) {
    drawContours(contourImg, contours, (int)i, Scalar(0, 255, 0), 2);
}
imshow("轮廓提取", contourImg);
```

📌 **说明**：

- `RETR_EXTERNAL`：只提取最外层轮廓。
- `CHAIN_APPROX_SIMPLE`：压缩冗余点，只保留端点。

---

3. 轮廓筛选（基于几何特征）

提取到的轮廓可能包含无效边缘，需要根据几何特征筛选目标。

```cpp
for (size_t i = 0; i < contours.size(); i++) {
    double area = contourArea(contours[i]);
    if (area < 500) continue; // 面积过小的剔除

    Rect bbox = boundingRect(contours[i]);
    rectangle(img, bbox, Scalar(0, 0, 255), 2);
}
imshow("轮廓筛选", img);
```

📌 **常见筛选条件**：

- 面积大小（过滤小噪声）
- 外接矩形长宽比
- 圆形度、凸包、椭圆拟合、多边形近似

参考上海交通大学：视觉教程第七弹：边缘及轮廓检测

第二次培训任务

任务目标

- 配置 C++ 开发环境：在 Ubuntu 系统中安装必要的编译器和开发工具。
- 安装 OpenCV 库：利用 APT 或者使用编译安装的方式安装 OpenCV 的 C++ 版本。
- 组织项目结构：设计合理的项目目录结构，包含源代码、构建文件及资源文件。
- 实现基础图像处理操作：编写 C++ 程序完成图像处理的操作。
- 项目构建与运行：使用 CMake 配置项目，编译并运行程序，验证功能实现。
- 提交任务源代码以及 README：自行创建仓库并上传到 Github，将 Github 链接发送至 re5657hence@outlook.com（邮件主题为DX-RMV-姓名）

项目结构要求

```
opencv_project/
├── CMakeLists.txt
├── src/
│   └── main.cpp
├── README.md
├── resources/
│   └── test_image.png
└── build/
```

README要求

包含完成的思路，获取相关资料的方式，遇到的报错，如何解决的过程等

主要起一个记录的作用，没有格式要求

OpenCV基础使用

获取测试图片

```
mkdir resources
wget -O resources/test_image.png https://cdn.pixabay.com/photo/2022/11/11/14/27/cosmos-7585071_1280.jpg

（原图片网址：https://pixabay.com/zh/photos/cosmos-flower-pink-flower-petals-7585071/）
```

具体操作要求

在 `src/main.cpp` 中实现基础图像处理操作，需要对于任务给定的图片进行以下操作：

- 图像颜色空间转换
  - 转化为灰度图
  - 转化为 HSV 图片
- 应用各种滤波操作
  - 应用均值滤波
  - 应用高斯滤波
- 特征提取

    - 提取红色颜色区域
        - HSV 方法
    - 寻找图像中红色的外轮廓
    - 寻找图像中红色的 bounding box
    - 计算轮廓的面积
    - 提取高亮颜色区域并进行图形学处理
        - 灰度化
        - 二值化
        - 膨胀
        - 腐蚀
        - 对处理后的图像进行漫水处理
- 图像绘制
    - 绘制任意圆形方形和文字
    - 绘制红色的外轮廓
    - 绘制红色的 bounding box
- 对图像进行处理
    - 图像旋转 35 度。
    - 图像裁剪为原图的左上角 1/4。

实际应用

获取图片

```
wget -O resources/test_image_2.png https://picr2.axi404.top/test_image.jpg
```

要求

识别出装甲板区域，使用矩形框框选出来

提交内容

整体的 opencv_project 项目需提交至 Github，其中应包括，符合 `项目结构` 的内容，即至少包括以下内容：

1. **源代码**
2. **资源文件**：
    - `resources` 中的图像文件，原图与处理后的图像。
    - 程序运行时终端的输出截图。
3. README.md