Lab 1 - OpenCV 的安装和使用

此次实验中，我们将安装OpenCV并熟悉OpenCV的一些基本数据结构。

环境要求

计算摄影学课程假定同学们使用 Windows 10 操作系统并安装有（至少） Visual Studio 2015。
你也可以从 http://ms.zju.edu.cn/ 免费下载 Visual Studio 2015/2017。

OpenCV 的官方网站是 http://opencv.org/，课程推荐使用 OpenCV 3。

同学们可以从https://docs.opencv.org/查阅OpenCV的相关用法。

安装 OpenCV

课程准备了 OpenCV 3.4.5 版的安装程序，另外也准备了可以直接解压缩使用的版本。

- 安装程序：opencv-3.4.5-setup.exe

你可以运行 opencv-3.4.5-setup.exe，将其解压安装到电脑中。

如果你使用安装程序安装了 OpenCV 3.4.5 ， OpenCV 的安装目录会是如下结构：

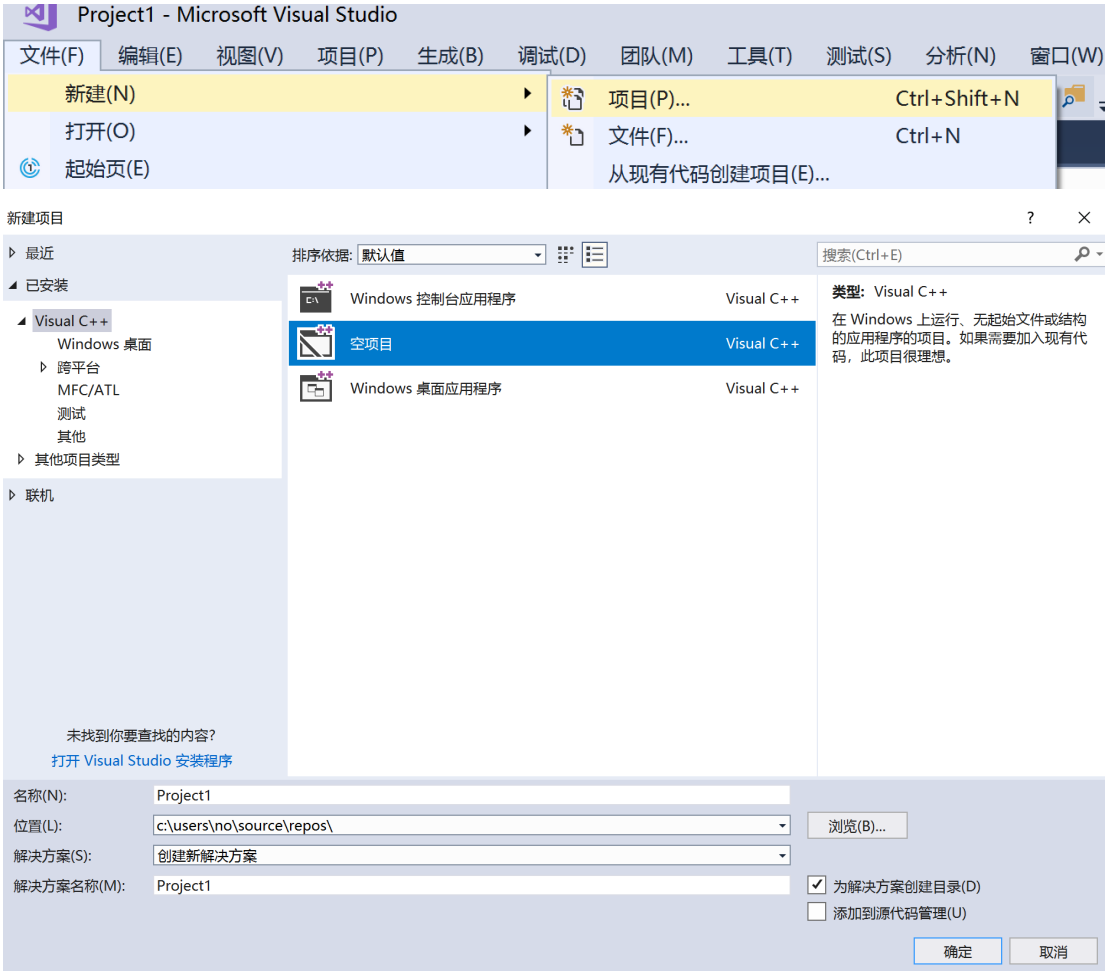
- `opencv` - OpenCV 的安装目录
  - `build`
    - `include` - OpenCV 的 C++ 头文件
    - ...
    - `x64 - 编译 64 位 OpenCV 程序所依赖的库
      - `vc14` - Visual C++ 2015 版本对应的库
        - `bin` - 运行动态链接库程序时需要的 DLL
        - `lib` - 动态链接库
      - `vc15` - Visual C++ 2017版本对应的库
  - `sources`

Hello, OpenCV

首先我们使用 OpenCV 编写一个简单的 Hello, OpenCV 程序，它将载入一个图片文件，显示在屏幕上。
实验用到的样例图片为 opencv-logo.png。

opencv-logo.png
你需要注意我们如何使用 OpenCV 的头文件，以及链接到它的库文件。

首先，在 Visual C++ 中新建一个空项目，项目名称为 `hellocv` 。



并在顶端栏目将 `x86` 改为 `x64` ， 因为我们的OpenCV是64位版本的预编译。

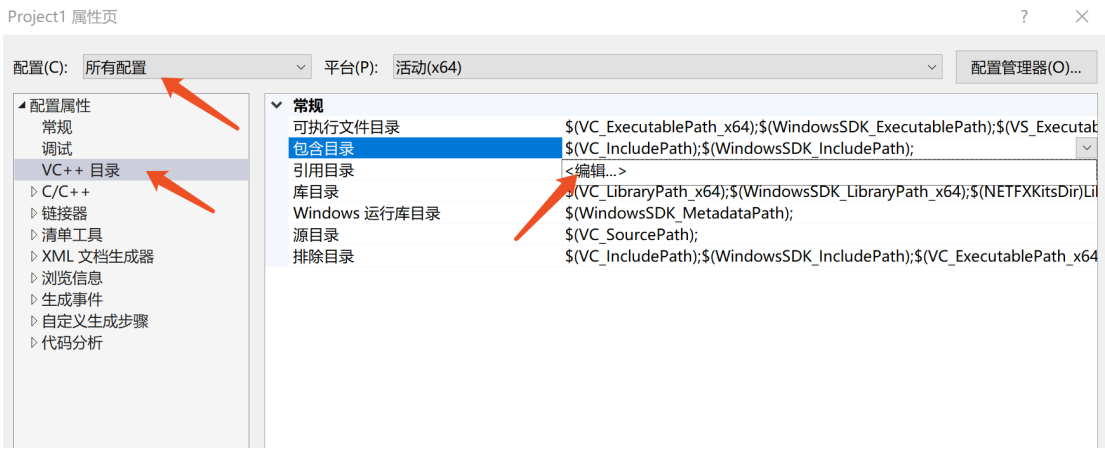
代码清单

在 hellocv 项目中添加一个源代码文件，内容如下：

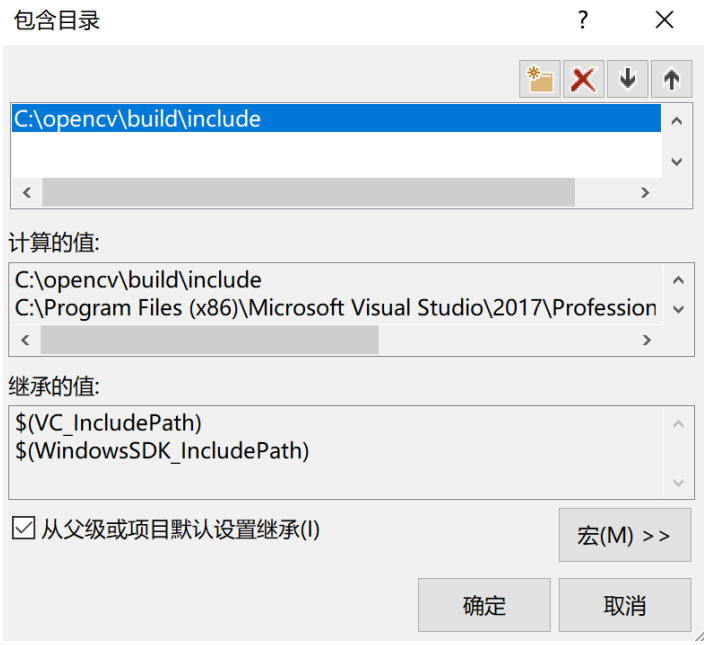
```
#include <opencv2/opencv.hpp>
using namespace cv;

int main(int argc, char* argv[]) {
    Mat image = imread("opencv-logo.png"); // 载入名为 "opencv-logo.png" 的图片
    namedWindow("hello"); // 创建一个标题为 "hello" 的窗口
    imshow("hello", image); // 在窗口 "hello" 中显示图片
    waitKey(0); // 等待用户按下键盘
    destroyWindow("hello"); // 销毁窗口 "hello"
    return 0;
}
```

编译并链接到 OpenCV

打开 hellocv 项目的属性页，首先确保 Configuration （配置）选项处选择了 All Configurations（所有配置），然后在画面左侧选择 VC++ Directories（VC++目录） 子项。在右侧的 Include Directories（包含目录） 下拉项中点选 edit， 参考下图：



在弹出的窗口中添加一个项目，指向 `opencv/build/include` 的具体路径：



类似的，找到 Library Directories（库目录），并添加对应的库文件路径：

- Visual Studio 2015： `c:/xxxxxx/opencv/build/x64/vc14/lib`
- Visual Studio 2017： `c:/xxxxxx/opencv/build/x64/vc15/lib`

注意如果使用的是 Visual Studio 2015，路径中需要选择 `vc14` 。你需要选择对应于你环境的路径。

接下来，在左侧找到 Linker（链接器） - Input（输入），右侧的 Additional Dependencies（附加依赖项）中添加需要链接到的 OpenCV 库（.lib）文件：

```
opencv_world345d.lib
```

保存改动后，编译程序即可。

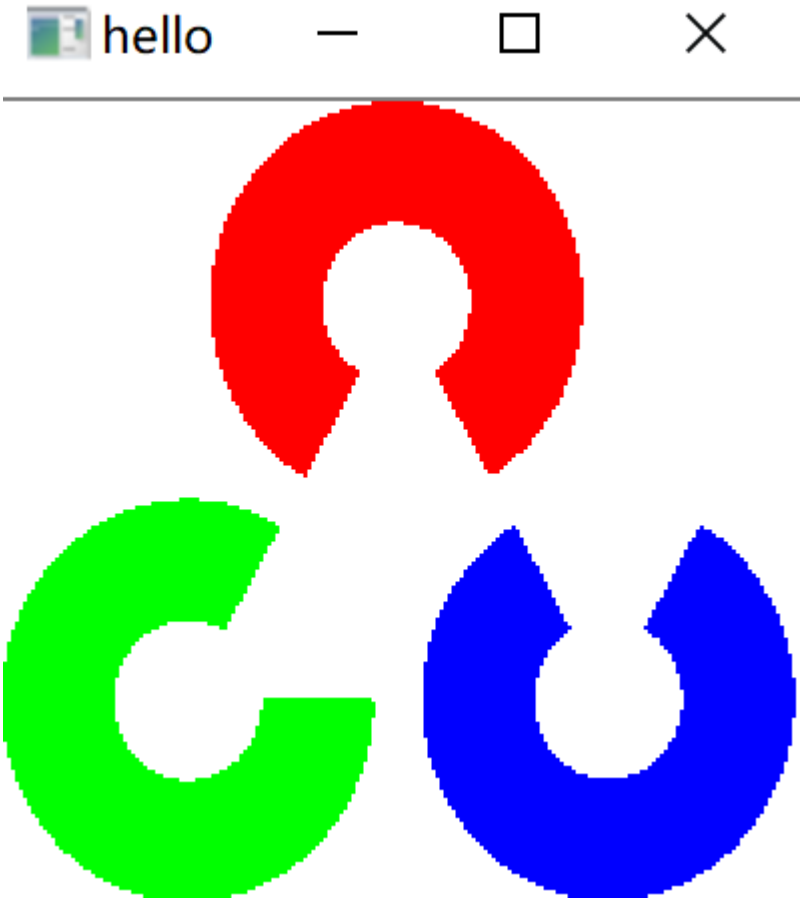
> Visual Studio在默认情况下使用Debug编译。对应的OpenCV lib库后面带字母d。如果需要更好的性能，切换到Release编译，link的lib库也需要改为 `opencv_world345.lib` 。另外，

> 属性页中也可以单独设置Debug和Release的配置（将左上角的所有配置改为Debug或者Release）。

运行带有 OpenCV 的程序

为了运行，应用程序需要能够找到依赖的动态链接库（.dll），否则会提示错误。
这些动态链接库文件位于 lib 目录旁边的 bin 目录中。

为了运行程序，你可以将需要的 `opencv/build/vc14_or_15/bin` 中的dll 文件（Debug对应 `opencv_world345d.dll`，Release对应 `opencv_world345.dll`）复制到你的程序 exe 所在的目录。
另外，别忘了把OpenCV的logo图像也放到程序的运行目录（通常是 `.vcxproj` 所在的目录，你也可以在属性页中自己修改），或者修改程序使用图像的绝对路径。
在一切都配置正常后，你应该可以看到如下的运行结果：



按下键盘上的任意按键便可退出。

操作 OpenCV 数据

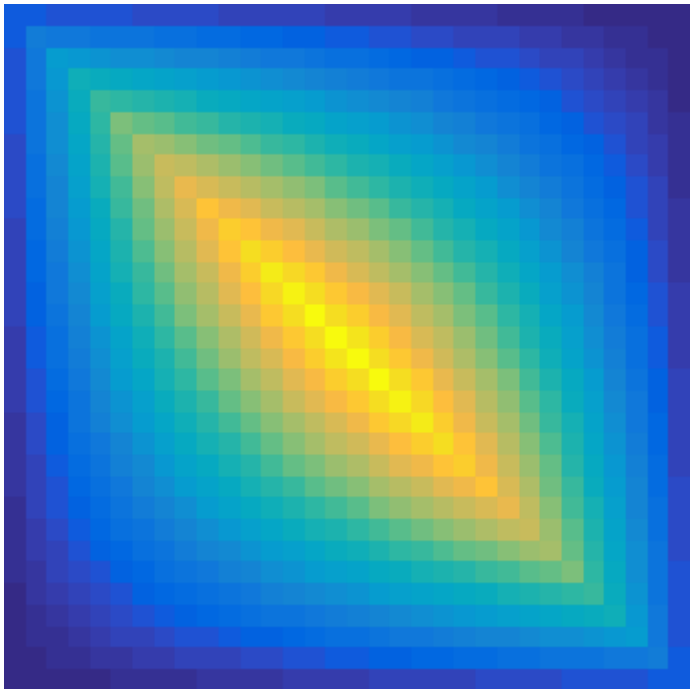
OpenCV 中最常用的数据类型是矩阵 `cv::Mat` 。在 Hello, OpenCV 的例子里，图片被载入为 `cv::Mat` 类型。

本次实验课需要你学会阅读 OpenCV 的文档。

任务清单

- 阅读 OpenCV 文档的 Introduction 和 `core` 模块中 `cv::Mat` 类型的文档。
  - Introduction: https://docs.opencv.org/3.4.5/d1/dfb/intro.html
  - `cv::Mat` : https://docs.opencv.org/3.4.5/d3/d63/classcv_1_1Mat.html
  - 参考文档的介绍，完成下面的任务：
    - 修改 Hello, OpenCV 的程序，使用 `image.at<...>(...)` 访问图像中的像素。
      - 图像的通道数是多少？每个通道是什么类型？
      - `at<...>` 的尖括号里要使用什么类型？
      - 提示： `Vec3b`
    - 遍历 `image` 的每个像素，将图像的白色部分修改为黑色。
      - 图像的长宽要怎么获得？
      - 图像的长和宽与矩阵的行数列数是什么关系？
      - 提示： `image.rows` ， `image.cols`
    - 修改程序，将 `image` 反色。
      - 尝试不遍历像素，直接用 `Mat` 的基本运算（减法）完成这个任务。
      - 提示： `Vec3b(255, 255, 255) - image`
      - 构造下面的 32×32 矩阵 M ，计算它的逆 M^{-1} :
      - $$M = \begin{pmatrix} 2 & -1 & 0 & & 0 \\ -1 & 2 & -1 & \cdots & 0 \\ 0 & -1 & 2 & & 0 \\ & \vdots & & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 2 \end{pmatrix}$$
      - 尝试将 M^{-1} 显示成一个灰度图，每个元素对应一个像素的亮度，最大值为白色，最小值为黑色。
      - 将结果给助教检查。

- 提示：下图是一个放大并且着色了的结果图



详解

`cv::Mat` 表示一个矩阵，同时也被用来表示图像。图像可以具有多个色彩通道，为了表示这种情形， `Mat` 的元素可以是特殊的向量类型。

下面的代码构造了一个 32 x 32 的矩阵，它的元素是三通道的，每个通道内容是 `float` 类型：

```
Mat m(32, 32, CV_32FC3);
```

`CV_32FC3` 代表了这种三通道 `float` 类型， `32F` 表示内容是32-bit的浮点数， `C3` 代表三通道。类似地， `CV_8UC1` 表示单通道 `uchar` 类型。
对于一个多通道矩阵，它的每个元素是一个向量，向量的维数是矩阵的通道数。OpenCV中使用特殊的 `Vec` 类型表示这类向量， `CV_32FC3` 对应的向量类型是 `Vec3f` 。*单通道时，无需使用 `Vec` 类型，例如 `CV_8UC1` 对应的类型就是 `uchar` 。*

详细的类型可以参考：
http://docs.opencv.org/modules/core/doc/basic_structures.html#datatype

在访问矩阵元素时，可以使用 `Mat::at<T>` 方法， `T` 对应于矩阵的元素类型。下面的代码获得到 `m` 矩阵第 `i` 行第 `j` 列处元素的引用：

```
Vec3f &pix = m.at<Vec3f>(i,j);
```

矩阵有若干属性，常用的有行数，列数，矩阵的行数可以用 `m.rows` 获得，矩阵的列数可以用 `m.cols` 。

当矩阵用来表示图像时，图像的宽度对应了矩阵的列数，高度则对应了行数。一幅640x480图片可以按照如下方式构造：

```
Mat img(480, 640, CV_8UC3);
```

注意长和宽在构造函数中出现的顺序。

下面的代码演示了如何遍历任务中 image 的元素，将白色区域替换为黑色：

```
for(int i=0;i<image.rows;++i) {
    for(int j=0;j<image.cols;++j) {
        Vec3b &pixel = image.at<Vec3b>(i,j);
        if(Vec3b(255,255,255)==pixel) {
            pixel = Vec3b(0,0,0);
        }
    }
}
```

矩阵支持很多基本运算，下面的代码演示了如何构造一幅白色图像，用其减去已有图像来实现图像的反色：

```
// 构造与原始图像大小相等，类型相同的白色图像
Mat white(image.size(), image.type(), Scalar(255, 255, 255));
// 利用矩阵减法实现反色
image = white - image;
```

下面的代码完成了本次实验课的最后一个任务，对于一个方阵， Mat::inv() 给出了它的逆：

```
Mat L = 2.0*Mat::eye(32, 32, CV_32FC1); // 得到主对角线为 2 的矩阵
// 将两条副对角线赋值为 -1
for (int i = 1; i < 32; ++i) {
    L.at<float>(i-1, i) = -1;
    L.at<float>(i, i-1) = -1;
}
Mat Linv = L.inv(); // 计算 L 的逆
Mat result;
normalize(Linv, result, 1.0, 0.0, CV_MINMAX); // 重新映射，使最小值为黑，最大值为白
```

这里我们使用了很多 OpenCV 提供的便利函数，例如 eye 和 normalize ，你也可以利用前面介绍的元素遍历来获得矩阵的最小值和最大值，然后自行完成重映射。

> 早期的 OpenCV 使用了 CvMat 、 IplImage 等结构体来表示矩阵和图像等。它们需要使用专门的函数进行建立或销毁，容易产生资源泄漏或悬挂引用等问题。 cv::Mat 采用C++对

> 象的生命周期来管理它的内容，可以更好的避免这类问题。我们不推荐使用早期的结构体类型。

Tips

- 使用macOS的同学， `brew install opencv@3` 以后，如果在CMake中find_package找不到OpenCV的话，可以 `find_package(OpenCV 3 HINTS /usr/local/opt/opencv@3 REQUIRED)` 。通常情况下OpenCV4也是可以的（常见操作里面主要有几个宏发生了变化）。
- 如果你所在的平台没有Visual Studio C++，你还可以选择CLion（有Student License可以免费获得），或者是开源的VSCode（配合C/CPP Tool，CMake，CMake-Tool同样可以达到优秀的C++代码体验）。
- 使用Windows的同学，如果在debug编译下出现了link error，可以将 `opencv_world345.lib` 改成 `opencv_world345d.lib` 。

参考资料

- OpenCV 官方网站（英文）
- OpenCV 在线文档 （英文）
- OpenCV 中文网站
- OpenCV 逆引きリファレンス（日文）