实验二　图像滤波与形态学

1　实验目的

① 掌握图像的空域滤波及频域滤波原理

② 设计算法实现空域滤波及频域滤波

2　实验仪器

① 机器人硬件：开源移动机器人、笔记本

② 笔记本软件：ros-kinetic-full、opencv（仅用于实现图像的读取操作）

3　实验原理

3.1 空域滤波

使用空域模板进行的图像处理，被称为空域滤波。模板本身被称为空域滤波器。在 M×N 的图像 f 上，使用 m×n 的滤波器：

$$g(x,y)=\sum_{s=-a}^{a}\sum_{t=-b}^{b}w(s,t)f(x+s,y+t) \tag{1}$$

$$m=2a+1,n=2b+1 \tag{2}$$

空间滤波的简化形式：

$$R=w_1z_1+w_2z_2+\cdots+w_{mn}z_{mn} \tag{3}$$

其中，w 是滤波器系数，z 是与该系数对应的图像灰度值，mn 为滤波器中包含的像素点总数。

在空域滤波功能都是利用模板卷积，主要步骤为：

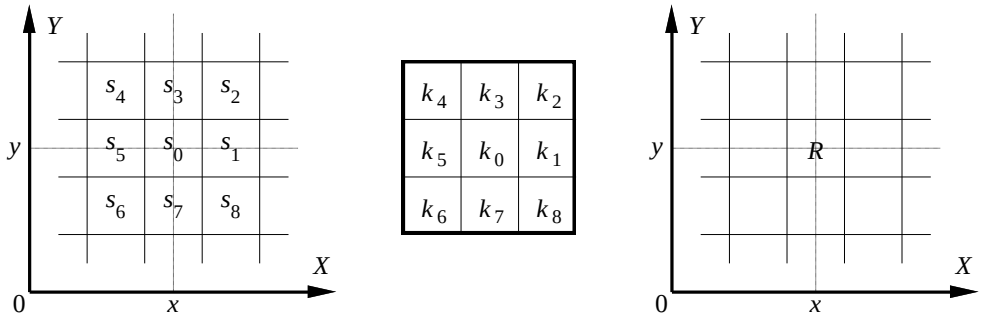
(1) 将模板在图中漫游，并将模板中心与图中某个像素位置重合；

(2) 将模板上系数与模板下对应像素相乘；

(3) 将所有乘积相加；

(4) 将和（模板的输出响应）赋给图中对应模板中心位置的像素。

例：模板滤波示意



(a) (b) (c)

模板的输出为：$R = k_0 s_0 + k_1 s_1 + \cdots + k_8 s_8$。

3.1.1 平滑空域滤波

平滑空域滤波器主要用于模糊处理和降低噪声。主要分为线性空域滤波和非线性空域滤波。

平滑线性空域滤波器的输出是包含在滤波器模板领域内的像素的平均值。这些滤波器有时也被称为均值滤波器。

$\frac{1}{9}\times$

1	1	1
1	1	1
1	1	1

$\frac{1}{16}\times$

1	2	1
2	4	2
1	2	1

左图产生标准的像素平均值；右图产生像素的加权平均，用以体现不同像素的不同重要性。

3.1.2 锐化空域滤波

锐化处理的主要目的是增强边缘及突出灰度的跳变部分。对微分的定义可以有各种表述，这里必须保证如下几点：

（1）在平坦段为 0

（2）在灰度阶梯或斜坡的起始点处为非 0

（3）沿着斜坡面微分值非 0

对于一元函数表达一阶微分：

$$\frac{\partial f}{\partial x} = f(x+1) - f(x) \qquad (4)$$

二阶微分：

$$\frac{\partial^2 f}{\partial^2 x} = f(x+1) + f(x-1) - 2f(x) \quad (5)$$

常见的是使用二阶微分-拉普拉斯算子进行图像锐化。算子定义如下：

$$\nabla^2 f(x,y) = f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1) - 4f(x,y) \quad (6)$$

使用拉普拉斯对图像增强的基本方法可表示为下式：

$$g(x,y) = f(x,y) + c[\nabla^2 f(x,y)] \quad (7)$$

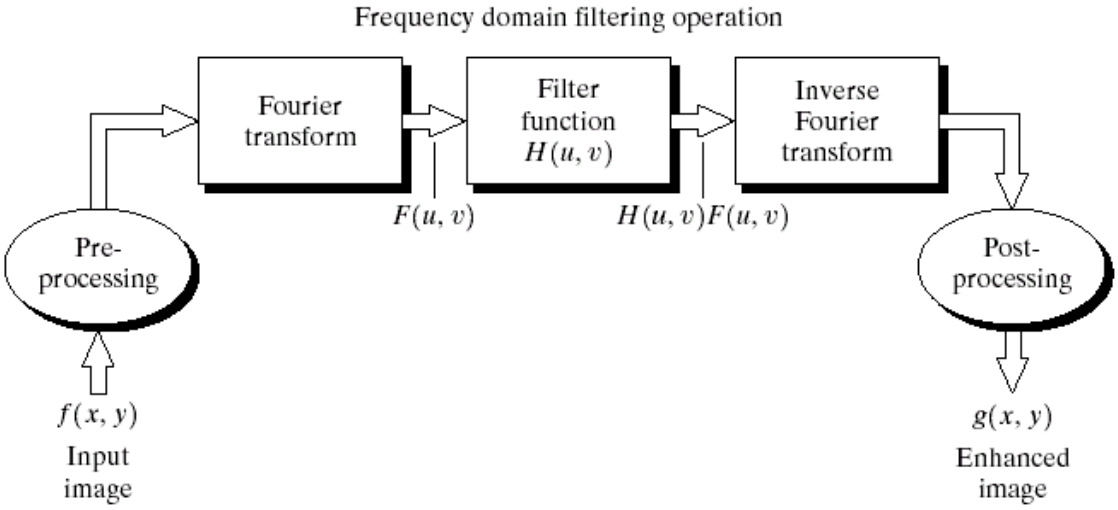
滤波器模板如下图：

0	1	0
1	-4	1
0	1	0

3.2 频域滤波

空域滤波里强调了像素之间的变化率，而频率直接关系到空间变化率，因此对图像频率域进行研究可以更好的掌握图像处理的技术原理。

具体滤波流程图如下图所示：



3.2.1 傅里叶变换

关于$f(x)$的傅里叶变换如下：

$$F[f(x)] = \int_{-\infty}^{\infty} f(x) e^{-j2\pi\omega x} dx \quad (8)$$

其中：$j = \sqrt{-1}$，并且根据欧拉公式：$e^{j\theta} = cos\theta + jsin\theta$，w为频率。

计算机处理的都是离散值，因此关于$f(x)$离散傅里叶变换（DFT）如下：

$$F[f(x)] = F(u) = \sum_{x=0}^{N-1} f(x)e^{-j2\pi ux/N} \quad (9)$$

其中 u=0,1,2,…,N-1.

具体计算过程如下：

1、令 u=0，首先计算每个 x 对应的 f(x)，并利用上式计算 F(0)。

2、令 u=u+1，再次计算每个 x 对应的 f(x)，并计算 F(u+1)

3、如果 u=N-1，则退出程序；否则，返回第 2 步，继续执行。

3.2.2 快速傅里叶变换

如果实现上述傅里叶变换会发现，对于一副像素点为 n*n 的图像，其程序时间复杂度为 $O(n^2)$，在实际中难以得到应用。因此需要快速傅里叶变换（FFT）用于实际应用。具体推导如下：

将（9）式写为：

$$F(u) = \sum_{x=0}^{M-1} f(x)W_M^{ux} \quad (10)$$

式中，u=0,1,…,M-1，其中

$$W_M = e^{-j2\pi/M} \quad (11)$$

且假设 M 具有如下形式：

$$\mathrm{M} = 2^n \quad (12)$$

其中 n 为一个正整数。因此，M 可表示为

$$\mathrm{M} = 2\mathrm{K} \quad (13)$$

K 也是一个整数。将式（13）代入式（10）得到

$$F(u) = \sum_{x=0}^{2K-1} f(x)W_{2K}^{ux} = \sum_{x=0}^{K-1} f(2x)W_{2K}^{u(2x)} + \sum_{x=0}^{K-1} f(2x+1)W_{2K}^{u(2x+1)} \quad (14)$$

通过式（11）可以证明$W_M^{2ux} = W_K^{ux}$，因此式（14）可以表示为

$$F(u) = \sum_{x=0}^{K-1} f(2x)W_K^{ux} + \sum_{x=0}^{K-1} f(2x+1)W_K^{ux}W_{2K}^{u} \quad (15)$$

定义

$$G(u) = \sum_{x=0}^{K-1} f(2x)W_K^{ux} \quad (16)$$

式中，u=0,1,2,…,K-1，和

$$H(u) = \sum_{x=0}^{K-1} f(2x+1)W_K^{ux} \quad (17)$$

式中，u=0,1,2,…,K-1。则将式（15）简化为：

$$F(u) = G(u) + H(u)W_{2K}^{u} \quad (18)$$

还可以推导，$W_M^{u+M} = W_M^u$和$W_{2M}^{u+M} = -W_{2M}^u$，式（16）到（18）可以得出：

$$F(u+K) = G(u) - H(u)W_{2K}^u \quad (19)$$

可以看到式（18）和式（19）的右半部分只相差一个符号。因此，若K=u，则可以通过$G(u)$、$H(u)$算出$F(u)$和$F(2u)$，通过$G(u+1)$、$H(u+1)$算出$F(u+1)$和$F(2u+1)$，再通过递归分治来计算 FFT。此时，算法时间复杂度为O($nlog_2n$)，相比于 DFT，其时间复杂度将以指数形式减少，更适用于实际应用场景。

3.2.3 低通滤波器

在以原点为圆心，以 D0 为半径的圆内，无衰减地通过所有频率，而在该圆外“切断”所有频率的二维低通滤波器，称为理想低通滤波器，它由下面的函数决定：

$$H(u,v) = \begin{cases} 1, & D(u,v) \leqslant D_0 \\ 0, & D(u,v) > D_0 \end{cases}$$

其中 D0 是一个正常数，D(u,v)是频率域中点(u,v)与频率矩形中心的距离，即：

$$D(u,v) = \left[(u - P/2)^2 + (v - Q/2)^2 \right]^{1/2}$$

3.2.4 高通滤波器

理想高通滤波器定义如下：

$$H(u,v) = \begin{cases} 0, & D(u,v) \leqslant D_0 \\ 1, & D(u,v) > D_0 \end{cases}$$

其中 D0 是截止频率，D(u,v)与低通滤波器中 D(u,v)定义相同。

3.3 形态学

3.3.1 膨胀

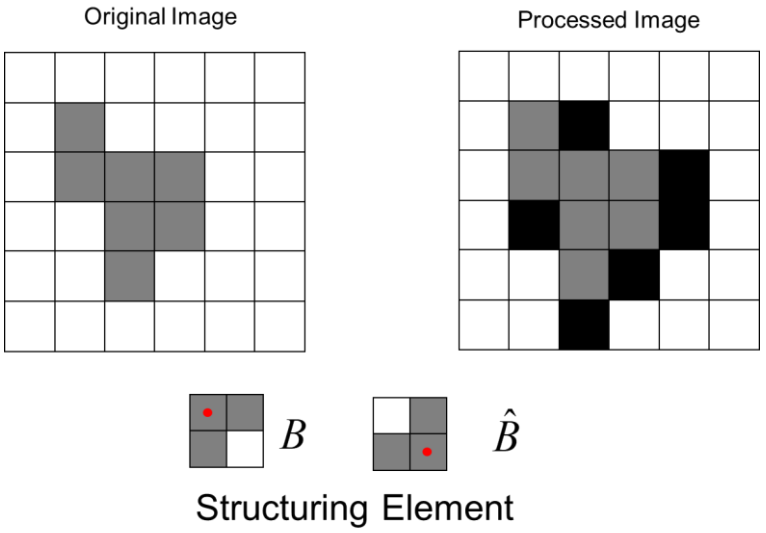
A 表示图像像素点集合，B 称为结构元，则 B 对 A 的膨胀定义为：

$$A \oplus B = \{x \mid [(\hat{B})_x \cap A \neq \varnothing\}$$

这个公式是以 B 关于它原点的映像，并且以 x 对映像进行平移为基础的。B 对 A 的膨胀是所有位移 x 的集合。因此，上式可写为：

$$A \oplus B = \{x \mid [(\hat{B})_x \cap A] \subseteq A\}$$

具体例子如下图所示：



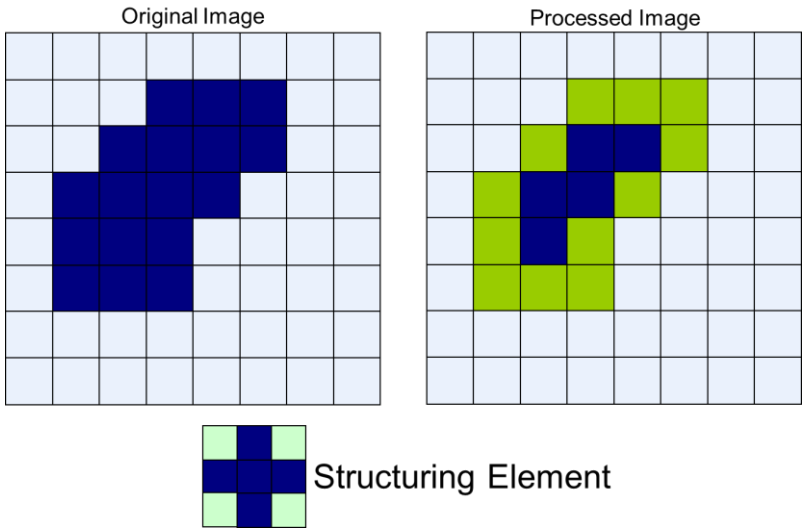
3.3.2 腐蚀

A 表示图像像素点集合，B 称为结构元，则 B 对 A 的腐蚀定义为：

$$A \ominus B = \{x \mid (B)_x \subseteq A\}$$

这个公式是一个用 x 平移的 B 包含在 A 中的所有点 x 的集合。

具体例子如下图所示：



4 实验内容

4.1 算法实现

4.1.1 空域滤波（以高斯滤波为例）

① 设计高斯滤波器模板函数

② 填充图像，将模板函数与图像进行卷积

③ 截取图像，获得滤波后的图像

4.1.2 频域滤波（以低通滤波器为例）

① 编写快速傅里叶变换函数（蝶形傅里叶变换）

② 设计低通滤波器模板函数

③ 填充图像，将模板函数与图像进行卷积

④ 截取图像，获得滤波后的图像

4.1.3 腐蚀/膨胀算法

① 使用 imread 读取图片

② 将 RGB 图像转换成灰度图

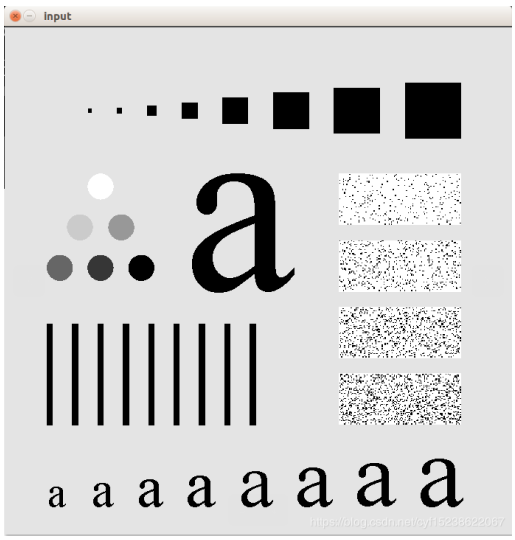
③ 进行腐蚀或膨胀

④ 将经过腐蚀或膨胀后的图片显示

4.2 实验效果

4.2.1 空域高斯滤波

左图为输入的原始图像，右侧为输出的高斯滤波后图像。通过图像可以看到最明显的效果就是模糊，设置不同的高斯滤波参数（方差、卷积核尺寸等参数），可以实现不同的模糊效果。



4.2.2 频域低通滤波

如下图，图 a 为输入的原始图像，图 b 为输出的理想低通频域滤波后图像，图 c 为图形化显示的滤波器，图 d 为图像频谱。可以看到图 b 相对于原图 a 同样是模糊效果，这是由于低通滤波器允许频率较低的信号通过，高频信号被滤除，

即图像中尖锐的细节被滤除，显示出模糊的效果。

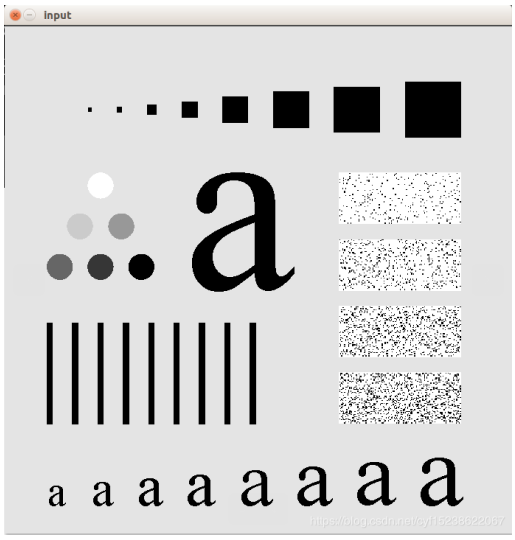


图 a

图 b

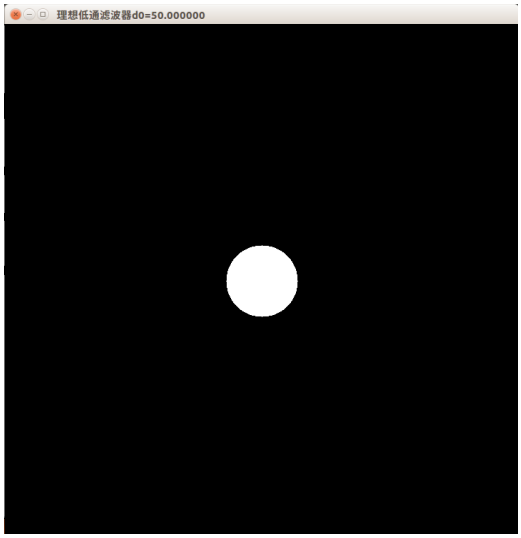


图 c

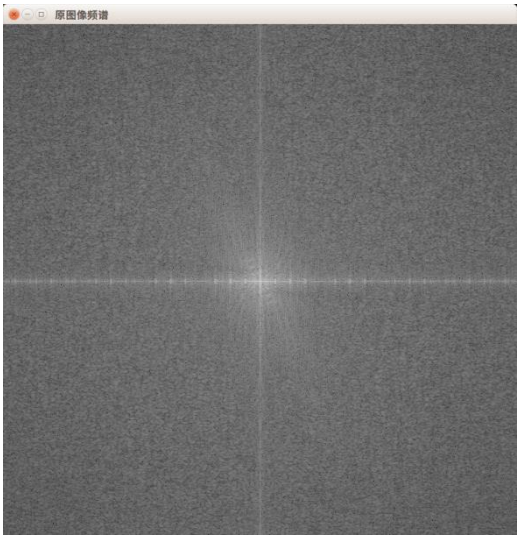
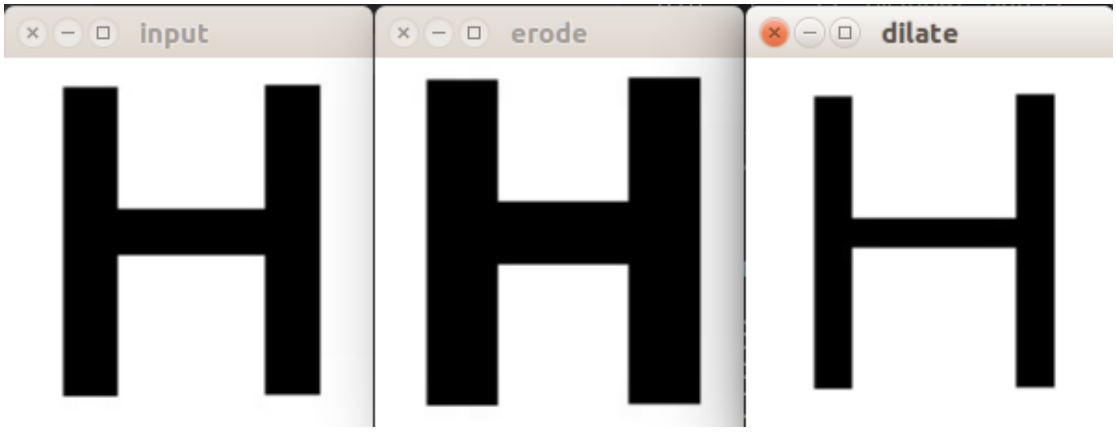


图 d

4.2.3 腐蚀/膨胀

图中左侧为输入图像，中间为腐蚀后的图像，右侧图像为膨胀后的图像。可以看到：腐蚀处理后，图像白色区域减少，黑色区域增多；膨胀处理后，图像白



色区域增多，黑色区域减少。

附录：（代码框架）

```
#include <stdlib.h>
#include <cv.h>
#include <highgui.h>
#include <opencv2/opencv.hpp>
#include <opencv2/core/core.hpp>
#include "ros/ros.h"
#include "std_msgs/String.h"
#include "std_msgs/Bool.h"
#include "std_msgs/Float32.h"

#include<geometry_msgs/Twist.h>
#include "sensor_msgs/Image.h"

#define LINEAR_X 0

using namespace cv;

//////////////////////滤波//////////////////
// 空域高斯滤波器函数
void Gaussian(Mat input, Mat output, double sigma){

}

// 快速傅里叶变换
void fastFuriorTransform(Mat image) {

}
```

```
// 理想低通滤波器函数
Mat ideal_lbrf_kernel(Mat scr,float sigma)

}

// 频率域滤波函数
// src：原图像
// blur：滤波器函数
Mat freqfilt(Mat scr,Mat blur){

}

//////////////////////形态学//////////////////
// 膨胀函数
void Dilate(Mat Src, Mat Tem, Mat Dst){

}

// 腐蚀函数
void Erode(Mat Src, Mat Tem, Mat Dst){

}


int main(int argc, char **argv)
{
    VideoCapture capture;
    capture.open(1);//打开 zed 相机
```

```
ROS_WARN("*****START");
ros::init(argc,argv,"trafficLaneTrack");//初始化 ROS 节点
    ros::NodeHandle n;

    // ros::Rate loop_rate(10);//定义速度发布频率
    ros::Publisher pub = n.advertise<geometry_msgs::Twist>("/smoother_cmd_vel", 5);//定义速度发布器


if (!capture.isOpened())
{
    printf("摄像头没有正常打开，重新插拔工控机上当摄像头\n");
    return 0;
}
waitKey(1000);
Mat frame;//当前帧图片
int nFrames = 0;//图片帧数
int frameWidth = capture.get(CV_CAP_PROP_FRAME_WIDTH);//图片宽
int frameHeight = capture.get(CV_CAP_PROP_FRAME_HEIGHT);//图片高

while (ros::ok())
{
    capture.read(frame);
    if(frame.empty())
    {
        break;
    }

    // Mat frIn = frame();//使用笔记本摄像头
```

```
            Mat frIn = frame(cv::Rect(0, 0, frame.cols / 2, frame.rows));//截取 zed 的左目图片

            // 空域滤波函数
            Gaussian();

            // 频域滤波函数
            freqfilt();

            // 膨胀函数
            Dilate();

            // 腐蚀函数
            Erode();

            imshow("1",frIn);

            ros::spinOnce();
            waitKey(5);

        }
        return 0;
}
```