云南大学

本科实验报告

课程名称：图像理解与计算机视觉

实验名称：实验六. 图像描述实验

学院（系）：

专　　业：

班　　级：

姓　　名：

学　　号：

指导教师：

成　　绩：

评　　语：

Steven

一. 实验目的

通过编程实现能够使学生掌握常用特征提取方法的基本原理，如角点检测、边缘检测、纹理检测等方法，为计算机视觉更高层次的处理奠定基础。

二. 实验内容

（1）编程实现SIFT角点特征提取算法，并完成以下两张图像的SIFT角点特征匹配；


（2）编程实现Canny算子边缘检测算法，并对下图进行边缘检测；

（3）编程实现LBP纹理检测算法，并对（2）图提取特征；

（3）编程实现HOG特征提取算法，并对（2）图提取特征。

三. 实验环境

Matlab软件是图像处理领域广泛使用的仿真软件之一。本实验基于Matlab 2022版本完成。

四. 实验代码（详细注释，Times New Roman/宋体 五号字体 单倍行距）

```
%% 使用: 在命令行内调用函数 Exp6(func)，func 是不同功能的名字。
%% 每题的 demo 调用格式如下:
```

```
% 1. SIFT 角点特征提取算法，并完成以下两张图像的 SIFT 角点特征匹配:
Exp6("Ques1");
% ---本题需要下载 siftWin32.exe，并将其放在 Matlab 安装路径的 bin 文件夹下。同时
sift 的代码较多，将 sift 的相关代码打包成压缩包，与主程序一并提供。
% 2. 使用 Canny 算子对"lena.jpg"进行边缘检测算法: Exp6("Ques2");
% 3. 使用 LBP 纹理检测算法对 2 题图提取特征: Exp6("Ques3");
% 4. 使用 HOG 特征提取算法对 2 题图提取特征: Exp6("Ques4");
function Exp6(ques)

    if ques == "Ques1"
        cd sift\
        match("E:\OneDrive\作业暂存\图像理解与计算机视觉\实验
6\Code\EiffelTower1.jpg", "E:\OneDrive\作业暂存\图像理解与计算机视觉\实验
6\Code\EiffelTower2.jpg");
        cd ..\
    elseif ques == "Ques2"
        img_gray = im2gray(imread("lena.jpg"));
        canny_handwrite = Canny_edge_detect(img_gray);
        canny_res = edge(img_gray, 'canny');
        subplot(1, 2, 1);
        imshow(canny_res);
        title("自带 Canny 算子边缘检测结果");
        subplot(1, 2, 2);
        imshow(canny_handwrite);
        title("手动 Canny 算子边缘检测结果");
    elseif ques == "Ques3"
        img_gray = im2gray(imread("lena.jpg"));
        res = LBP(img_gray);
        imshow(res);
    elseif ques == "Ques4"
        img_gray = im2gray(imread("lena.jpg"));
        vector = HOG(img_gray);
        disp(vector);
    end
end

function res = Canny_edge_detect(img_gray)
    %1-高斯滤波
    gw = fspecial('gaussian', [5, 5], 0.5); %高斯滤波设置核，5*5，标准差为 0.5
    f_filter = imfilter(img_gray, gw, 'replicate'); %高斯滤波
    f = f_filter;

    %2-利用 Sobel 算子计算像素梯度
    Sobel_X = [-1, 0, 1; -2, 0, 2; -1, 0, 1]; %X 方向 Sobel 算子(互相关算子，非卷积算子)
```

```
    Sobel_Y = [-1, -2, -1; 0, 0, 0; 1, 2, 1]; %Y 方向 Sobel 算子(互相关算子)
    [rowNum, columnNum] = size(f);
    f_extend = zeros(rowNum + 2, columnNum + 2); %图像扩充，边界补充为 0

    for i = 2:rowNum + 1
        for j = 2:columnNum + 1
            f_extend(i, j) = f(i - 1, j - 1);
        end
    end

    Gx = zeros(rowNum, columnNum);
    Gy = zeros(rowNum, columnNum);
    for i = 2:rowNum + 1 %计算 x 向和 y 向梯度
        for j = 2:columnNum + 1
            window = [f_extend(i - 1, j - 1), f_extend(i - 1, j), f_extend(i - 1, j + 1); ...
                      f_extend(i, j - 1), f_extend(i, j), f_extend(i, j + 1); ...
                      f_extend(i + 1, j - 1), f_extend(i + 1, j), f_extend(i + 1, j + 1)];
            Gx(i - 1, j - 1) = sum(sum(Sobel_X .* window)); %计算 x 向梯度
            Gy(i - 1, j - 1) = sum(sum(Sobel_Y .* window)); %计算 y 向梯度
        end
    end

    Sxy = sqrt(Gx .* Gx + Gy .* Gy); %梯度强度矩阵计算
    %3-非极大值抑制
    indexD = zeros(rowNum, columnNum);
    for i = 1:rowNum %判断梯度方向所属区间，Gx=Gy=0，则令其为 5，肯定不是边界
点
        for j = 1:columnNum
            ix = Gx(i, j);
            iy = Gy(i, j);
            if (iy <= 0 && ix >- iy) || (iy >= 0 && ix <- iy) %梯度方向属于区间 1
                indexD(i, j) = 1;
            elseif (ix > 0 && ix <= -iy) || (ix < 0 && ix >= -iy) %梯度方向属于区间 2
                indexD(i, j) = 2;
            elseif (ix <= 0 && ix > iy) || (ix >= 0 && ix < iy) %梯度方向属于区间 3
                indexD(i, j) = 3;
            elseif (iy < 0 && ix <= iy) || (iy > 0 && ix >= iy) %梯度方向属于区间 4
                indexD(i, j) = 4;
            else %Gx 和 Gy 均为 0，无梯度，肯定非边缘
                indexD(i, j) = 5;
            end
        end
    end
```

```
    Gup = zeros(rowNum, columnNum);
    Gdown = zeros(rowNum, columnNum);

    for i = 2:rowNum - 1 %计算非边界处的插值梯度强度
        for j = 2:columnNum - 1
            ix = Gx(i, j);
            iy = Gy(i, j);
            if indexD(i, j) == 1 %计算区间 1 内插值梯度，Gup 为上方区间的梯度，Gdown
为下方区间的梯度
                t = abs(iy ./ ix);
                Gup(i, j) = Sxy(i, j + 1) .* (1 - t) + Sxy(i - 1, j + 1) .* t;
                Gdown(i, j) = Sxy(i, j - 1) .* (1 - t) + Sxy(i + 1, j - 1) .* t;
            elseif indexD(i, j) == 2 %计算区间 2 内插值梯度
                t = abs(ix ./ iy);
                Gup(i, j) = Sxy(i - 1, j) .* (1 - t) + Sxy(i - 1, j + 1) .* t;
                Gdown(i, j) = Sxy(i + 1, j) .* (1 - t) + Sxy(i + 1, j - 1) .* t;
            elseif indexD(i, j) == 3 %计算区间 3 内插值梯度
                t = abs(ix ./ iy);
                Gup(i, j) = Sxy(i - 1, j) .* (1 - t) + Sxy(i - 1, j - 1) .* t;
                Gdown(i, j) = Sxy(i + 1, j) .* (1 - t) + Sxy(i + 1, j + 1) .* t;
            elseif indexD(i, j) == 4 %计算区间 4 内插值梯度
                t = abs(iy ./ ix);
                Gup(i, j) = Sxy(i, j - 1) .* (1 - t) + Sxy(i - 1, j - 1) .* t;
                Gdown(i, j) = Sxy(i, j + 1) .* (1 - t) + Sxy(i + 1, j + 1) .* t;
            end
        end
    end

    Sxy_NMX = zeros(rowNum, columnNum); %判断是否为梯度方向极大值

    for i = 1:rowNum
        for j = 1:columnNum
            if Sxy(i, j) >= Gup(i, j) && Sxy(i, j) >= Gdown(i, j) %若为梯度方向极大值，则保
留；
                Sxy_NMX(i, j) = Sxy(i, j); %否则，进行抑制（置 0）
            end
        end
    end

    %4-滞后阈值法+5-抑制孤立的弱边缘
    res = zeros(rowNum, columnNum);

    %Tl 为高阈值，Th 为低阈值，connectNum 为联通参数，一般为 1
    Tl = 60; %lena
```

```
    Th = 120;
    connectNum = 1;

    for i = 2:rowNum - 1
        for j = 2:columnNum - 1
            if Sxy_NMX(i, j) >= Th %高于高阈值的像素为强边缘
                res(i, j) = 1;
            elseif Sxy_NMX(i, j) <= Tl %低于低阈值的像素为非边缘
                res(i, j) = 0;
            else %位于高低阈值之间的像素为弱边缘，进行孤立性检测
                count = 0;
                if Sxy_NMX(i - 1, j - 1) ~= 0 %左上方像素
                    count = count + 1;
                end
                if Sxy_NMX(i - 1, j) ~= 0 %上方像素
                    count = count + 1;
                end
                if Sxy_NMX(i - 1, j + 1) ~= 0 %右上方像素
                    count = count + 1;
                end
                if Sxy_NMX(i, j - 1) ~= 0 %左方像素
                    count = count + 1;
                end
                if Sxy_NMX(i, j + 1) ~= 0 %右方像素
                    count = count + 1;
                end
                if Sxy_NMX(i + 1, j - 1) ~= 0 %左下方像素
                    count = count + 1;
                end
                if Sxy_NMX(i + 1, j) ~= 0 %下方像素
                    count = count + 1;
                end
                if Sxy_NMX(i + 1, j + 1) ~= 0 %右下方像素
                    count = count + 1;
                end
                if count >= connectNum %弱边缘非孤立，则为边缘
                    res(i, j) = 1;
                end
            end
        end
    end
end

function res = LBP(img_gray)
```

```
    res = uint8(zeros(size(img_gray)));

    [width, height] = size(img_gray);

    for i = 1:width
        for j = 1:height
            block = zeros(3, 3);
            num = 0;
            for m = i - 1:i + 1
                for n = j - 1:j + 1
                    if m < 1 || n < 1
                        block(m - i + 2, n - j + 2) = 0;
                    elseif m > width || n > height
                        block(m - i + 2, n - j + 2) = 0;
                    elseif m == i && n == j
                        block(m - i + 2, n - j + 2) = 0; %  不参与其自身 LBP 值的计算
                    else
                        block(m - i + 2, n - j + 2) = img_gray(m, n) >= img_gray(i, j);
                    end
                    % num=num+block(m-i+2,n-j+2)*2^(m-i+n-j+2); %  最初采用(m+n)越
大，在 LBP 值占比越高的形式，但效果并不好
                end
            end
            %  之后采用了从(1,1)顺时针权重依次降低的形式，效果相对好了一些
            num = block(1, 1) * 2^7 + block(1, 2) * 2^6 + block(1, 3) * 2^5 + block(2, 3) * 2^4
+ block(3, 3) * 2^3 + block(3, 2) * 2^2 + block(3, 1) * 2^1 + block(2, 1) * 2^0;
            res(i, j) = num;
        end
    end
end

function vector = HOG(img_gray)
    [width, height] = size(img_gray);
    % 1-Gamma 校正（由于传入图像就是灰度图，所以仅进行 gamma 校正即可）
    img_gray = imadjust(img_gray, [], [], 0.5);
    % 2-利用 Sobel 算子计算像素梯度
    Sobel_X = [-1, 0, 1; -2, 0, 2; -1, 0, 1]; %X 方向 Sobel 算子(互相关算子，非卷积算子)
    Sobel_Y = [-1, -2, -1; 0, 0, 0; 1, 2, 1]; %Y 方向 Sobel 算子(互相关算子)
    [rowNum, columnNum] = size(img_gray);
    img_extend = zeros(rowNum + 2, columnNum + 2); %图像扩充，边界补充为 0
    for i = 2:rowNum + 1
        for j = 2:columnNum + 1
            img_extend(i, j) = img_gray(i - 1, j - 1);
        end
```

```
    end
    Gx = zeros(rowNum, columnNum);
    Gy = zeros(rowNum, columnNum);
    for i = 2:rowNum + 1 %  计算 x 向和 y 向梯度
        for j = 2:columnNum + 1
            window = [img_extend(i - 1, j - 1), img_extend(i - 1, j), img_extend(i - 1, j + 1); ...
                    img_extend(i, j - 1), img_extend(i, j), img_extend(i, j + 1); ...
                    img_extend(i + 1, j - 1), img_extend(i + 1, j), img_extend(i + 1, j + 1)];
            Gx(i - 1, j - 1) = sum(sum(Sobel_X .* window)); %计算 x 向梯度
            Gy(i - 1, j - 1) = sum(sum(Sobel_Y .* window)); %计算 y 向梯度
        end
    end

    % G = sqrt(Gx.^2 + Gy.^2); %  梯度幅值，但是本题中没用到
    alpha = (180 / pi) .* atan(Gy / Gx); %  梯度方向(角度)
    vector = zeros(width / 16, height / 16, 36);

    for i = 1:width / 16
        for j = 1:height / 16 %  循环 block
            % cell 按照[1,2;3,4]的顺序来的
            cell1 = alpha(i * 16 - 15:i * 16 - 8, j * 16 - 15:j * 16 - 8);
            cell2 = alpha(i * 16 - 7:i * 16, j * 16 - 15:j * 16 - 8);
            cell3 = alpha(i * 16 - 15:i * 16 - 8, j * 16 - 7:j * 16);
            cell4 = alpha(i * 16 - 7:i * 16, j * 16 - 7:j * 16);
            v = [histogram(cell1, 9).Values, histogram(cell2, 9).Values, histogram(cell3,
9).Values, histogram(cell4, 9).Values]'; %  特征向量
            vector(i, j, :) = v;
        end
    end
end
```

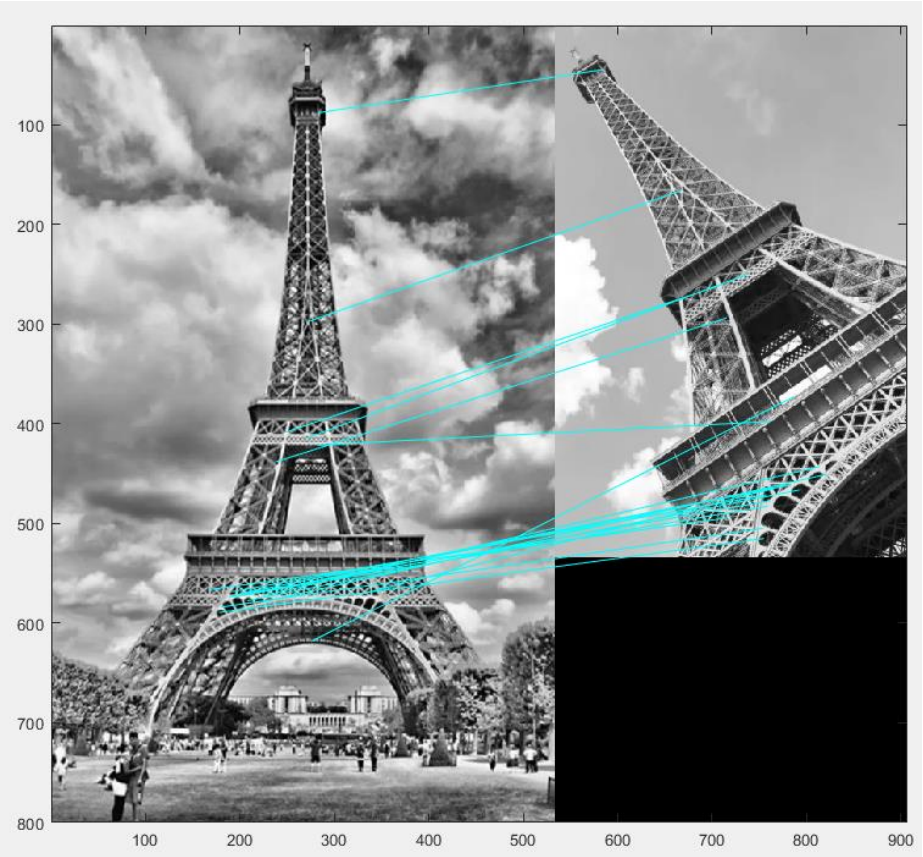

Exp6.m


sift.zip

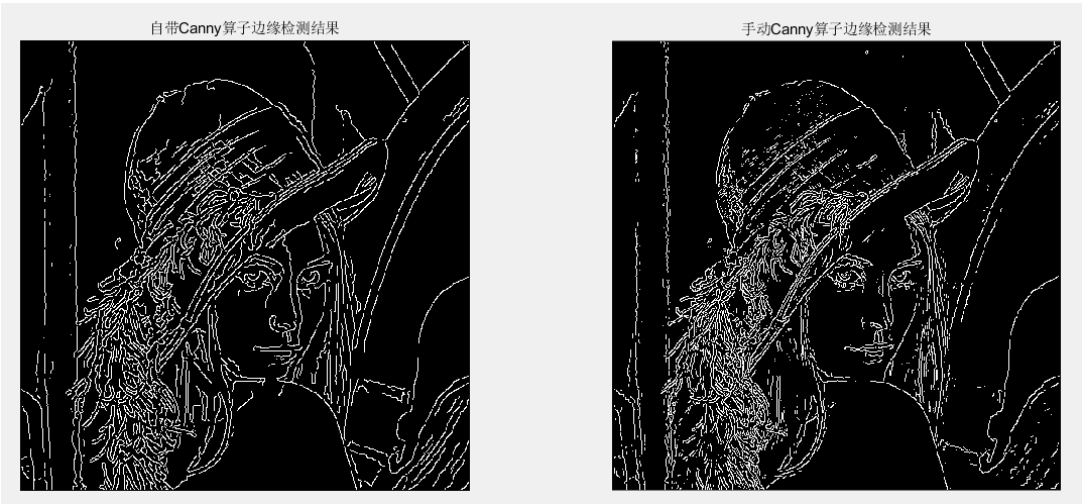
附件（.m文件）：

五. 实验结果

（1）SIFT角点检测结果（标出角点位置和方向），并展示两张图像的特征匹配结果



（2）Canny算子边缘检测结果



（3）LBP纹理检测结果

（4）HOG特征检测结果

命令行窗口

32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	33
32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32
32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32
32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	33
32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	31
32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32

列 17 至 32

33	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32
34	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32
32	33	32	32	32	32	32	32	32	32	32	32	32	32	32	32
32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32
32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32
32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	31
33	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32
32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32
31	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32
34	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32
32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32
33	32	32	32	32	32	32	32	32	32	32	32	32	32	32	31
33	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32
32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32
32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32
33	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32
31	32	32	32	32	32	32	32	32	32	32	32	32	32	32	31
31	32	31	32	32	32	32	32	32	32	32	32	33	32	32	32
32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32
32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32
32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32
34	32	32	32	32	32	32	32	32	32	32	32	32	32	32	33
32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32
32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32
31	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32

六. 结果分析及体会