深圳大学 SHENZHEN UNIVERSITY

计算机视觉

南区计软328

深圳大学
计算机与软件学院

实验内容

序号	实验主题	实验内容	实验要求	实验时数	每组人数	实验类型
1	图像处理应用实验	1．熟悉图像的表示及基本元素、通道操作； 2．掌握基本图像增强方法； 3．掌握OpenCV计算机视觉库；	必做	6	1	讲授+实验
2	图像特征提取及综合应用实践	1．熟悉图像处理基本操作； 2．掌握图像边缘检测原理； 3．掌握图像基本特征抽取以及在实际问题中的应用；	必做	6	1	讲授+实验
3	计算机视觉系统实践	1．熟悉计算机视觉分类任务； 2．掌握数据集的准备及模型训练过程； 3．培养应用计算机视觉解决问题的能力；	必做	6	1	讲授+实验

涉及学科

- 数字图像处理
- 计算机视觉
- 模式识别
- 程序设计

实验考察形式

- 三次实验报告（在Blackboard发布与提交）
- 平时编程练习及表现
- 实验汇报

实验考察内容

- 计算机视觉与模式识别基础知识掌握能力
- 算法设计能力
- 编程及系统架构能力
- 论文写作及表达能力

● 实验三

实验三：计算机视觉系统实践-自选项目

一、实验目的：

1. 了解图像处理和计算机视觉在人工智能中的地位、作用以及应用场景；
2. 掌握图像处理和计算机视觉常用方法及原理；
3. 掌握视觉系统设计思想，培养学生解决实际问题的能力

实验要求：

1. 实验提交文件为实验报告和相关程序代码，以压缩包的形式提交，实验报告命名规则为“计算机视觉-学号-姓名-实验报告3.doc”，其他文件打包成压缩文件，命名为“计算机视觉-学号-姓名-实验报告3-其他.zip”；
2. 所有素材和参考材料需列明出处，实验报告中的图片和程序代码建议标注个人水印或标识信息：姓名，班级，学号信息；

二、实验内容：设计计算机视觉目标识别系统，与实际应用有关（建议：最终展示形式为带界面可运行的系统），以下内容选择其中一个做。

● 实验三

1. 人脸识别系统设计

(1) 人脸识别系统设计（必做）：根据课堂上学习的理论知识（包括特征提取、分类器设计），设计一个人脸识别系统，该系统具有较好的识别率。可在提供的AR人脸图片数据集（120人）、Feret人脸图片数据集（175人）、人脸视频数据集（10人）、真实采集的人脸视频或其他公开数据集上展开实验。

(2) 人脸识别系统提升（至少选择其中1个问题做）：面向实际环境的人脸识别系统会考虑更多环节，包括图像预处理、特征提取、特征选择、分类器设计、训练与测试等。人脸识别算法在真实应用中会遇到以下问题，包括噪声干扰、光照变化、遮挡影响、角度变化。请针对以上至少1个问题（如噪声干扰、光照变化、遮挡影响、角度变化）展开探讨，分析是什么原因导致识别性能下降，提出增强人脸识别系统性能的方法，提高系统对异常情况处理的能力，使整个识别系统的适应性和稳定性达到更好的状态。

提示：

a. 噪声干扰方面可考虑图像增强算法，包括中值滤波、均值滤波、高斯滤波等；
b. 光照变化方面可考虑LBP算法及其扩展版本，或图像增强算法，如直方图均衡化、伽马变换等；
c. 遮挡影响方面可考虑对图像分块投票处理，或线性表示残差最小的方式辨别遮挡区域；
d. 角度变化方面可考虑增加不同角度的采样图片，或引入仿射变换的考虑；
e. 特征提取方面可采用Gabor特征，特征脸，深度特征等方法；
f. 分类器设计可采用贝叶斯分类器，神经网络等方法；

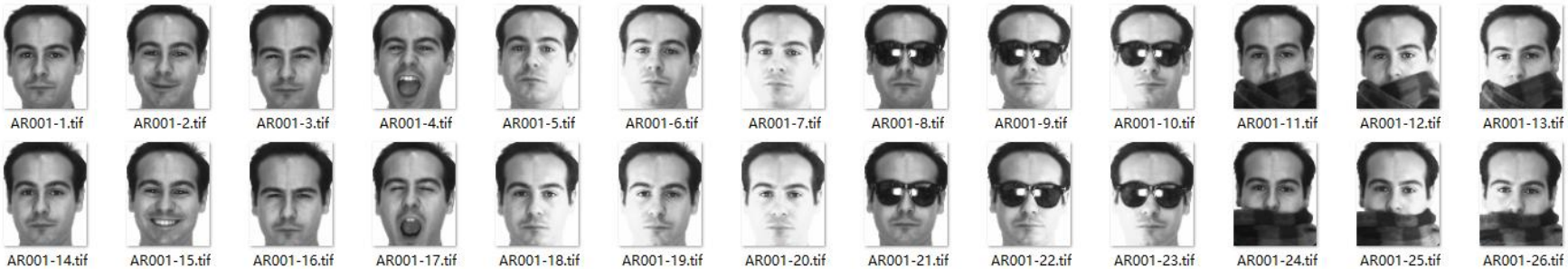
● 实验三

g. AR人脸数据集和Feret人脸数据集可以用来测试算法在噪声干扰（需要人工添加噪声）、光照变化、遮挡影响、角度变化下的性能。在此基础上可把识别系统应用在真实环境中测试。

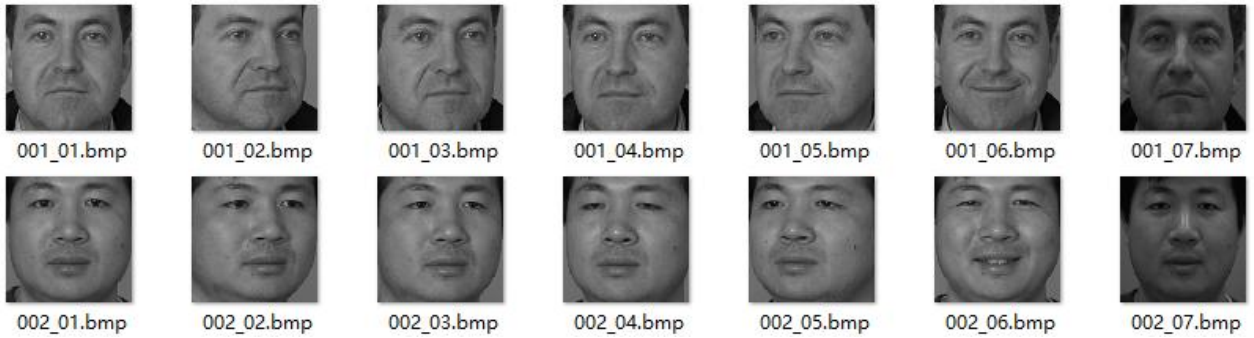
h. 可根据需要选用人脸检测器或引入仿射变换，本实验提供了Haar检测器和基于仿射变换的人脸检测器。

i. 人脸数据集说明：

- AR人脸数据集：包含120人，每人26张图片， 图像分辨率宽80，高100，可测试光照变化、遮挡情况下算法性能。



- Feret人脸数据集：包含175人，每人7张照片，图像分辨率宽80，高80，可测试不同角度、光照变化下算法性能。



实验三

- 人脸视频数据集：包含10人的视频，每个人有训练视频序列和测试视频序列，可测试不同角度、光照变化、遮挡干扰、噪声干扰下算法性能。

- 真实人脸数据采集：可根据系统实际情况拍摄。

(3)分析在实验室环境与自然环境下识别算法设计上的区别，如何提高算法创新？（可选）

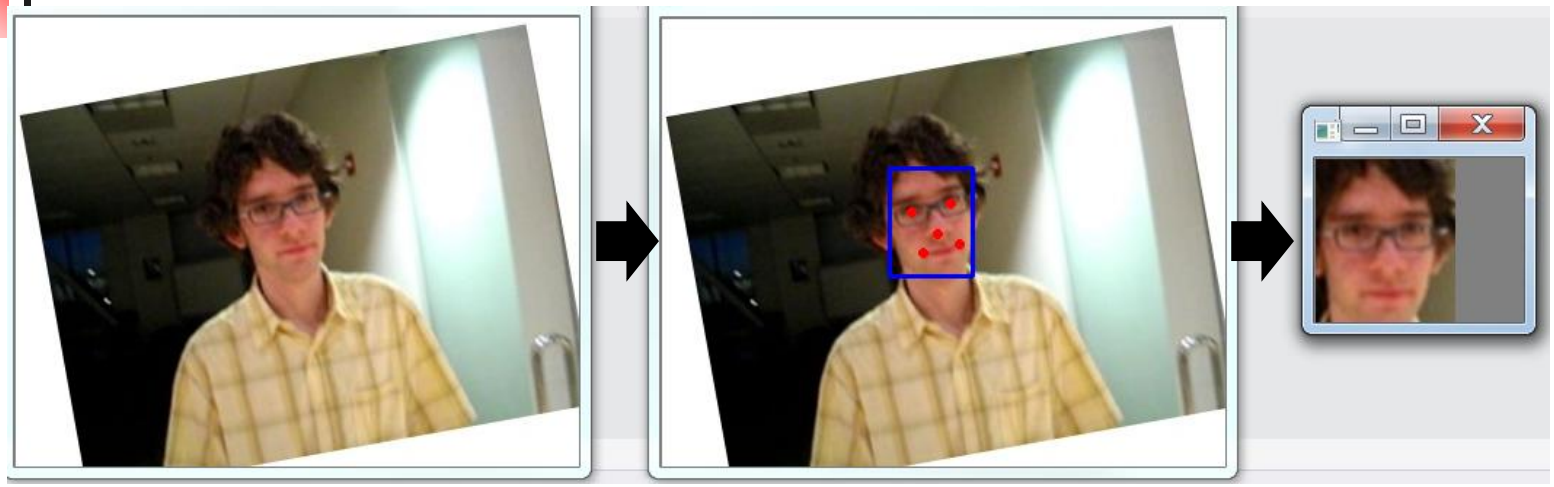
● 实验三

2. 自选目标识别内容，题目自拟

三、实验时间
实验报告统一命名为：计算机视觉-学号-姓名-实验报告3.doc
如有提交其他文件或程序，统一命名为：计算机视觉-学号-姓名-实验报告3-其他.zip

实验时间：2024年11月25日至2024年12月23日（注意按时提交，不要延期）
实验报告提交时间：2024年12月23日下午5:00

实验三-关于Fast-MTCNN的使用



https://github.com/szad670401/Fast-MTCNN

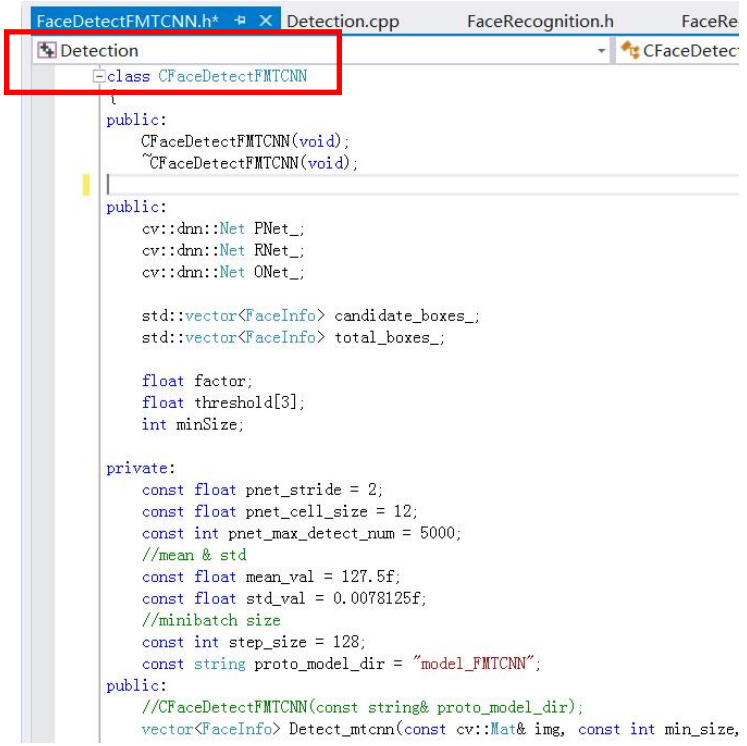
实验三-关于Fast-MTCNN的使用

```
int main(int argc, char **argv)
{
    MTCNN detector("model");
    string name_list[1] = {
        "test.jpg",
    };
//  MTCNN detector("./model");
    float factor = 0.709f;
    float threshold[3] = { 0.7f, 0.6f, 0.6f };
    int minSize = 12;
    for (int n = 0; n < 1;++n){
        cv::Mat image = cv::imread(name_list[n], 1);

        for(int i = 0 ; i < 10 ; i ++) {
            double t = (double) cv::getTickCount();
            vector<FaceInfo> faceInfo = detector.Detect_mtcnn(image, minSize, threshold, factor, 3);
            std::cout << name_list[n] << " time," << (double) (cv::getTickCount() - t) / cv::getTickFrequency() << "s"
                      << std::endl;

            for (int i = 0; i < faceInfo.size(); i++) {
                int x = (int) faceInfo[i].bbox.xmin;
                int y = (int) faceInfo[i].bbox.ymin;
                int w = (int) (faceInfo[i].bbox.xmax - faceInfo[i].bbox.xmin + 1);
                int h = (int) (faceInfo[i].bbox.ymax - faceInfo[i].bbox.ymin + 1);
                cv::rectangle(image, cv::Rect(x, y, w, h), cv::Scalar(255, 0, 0), 2);
            }
            cv::imwrite("test.png", image);
            cv::imshow("image", image);
            cv::waitKey(0);
        }
```

Package FMTCNN into a Class.



实验三-关于Fast-MTCNN的使用

faceInfo=Detect_mtcnn(image, minSize, threshold, factor, 3);

image：输入图像（cv::Mat 类型）。
minSize：最小检测人脸的尺寸。
threshold：每个阶段的置信度阈值数组（PNet, RNet, ONet）
factor：图像金字塔的缩放因子。
stage：要执行的检测阶段（1 表示 PNet，2 表示 RNet，3 表示 ONet）。

The main function returns:
(1) Face bounding box
(2) coordinates：Left eye X, Left eye Y, Right eye X, Right eye Y, Nose X, Noxe Y, Left Mouth Corner X ,Left Mouth Corner Y, Right Mouth Corner X, Right Mouth Corner Y

```
typedef struct FaceBox {
float xmin;
float ymin;
float xmax;
float ymax;
float score;
} FaceBox;

typedef struct FaceInfo {
float bbox_reg[4];
float landmark_reg[10];//currently no value
float landmark[10];// coordinates：Left eye X, Left eye Y, Right eye X, Right eye Y, Nose X, Noxe Y, Left Mouth Corner X
FaceBox bbox;                    //, Left Mouth Corner Y, Right Mouth Corner X, Right Mouth Corner Y
cv::Mat FeatureVector; //store feature vector computed from dnn model
} FaceInfo;
```

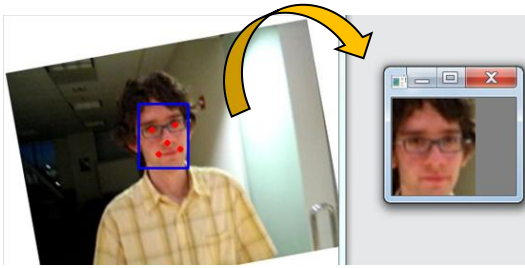
实验三-关于Fast-MTCNN的使用

```
void CFaceDetectFMTCNN::RotateFace(cv::Mat image, FaceInfo faceInfoVar, cv::Mat &dstImage)
{
    //vector<FaceDetector::BoundingBox> res = fd.Detect(src, FaceDetector::BGR, FaceDetector::ORIENT_UP, 20, 0.6, 0.7, 0.7);
    //float std_points[10] = { 89.3095, 72.9025, 169.3095, 72.9025, 127.8949, 127.0441, 96.8796, 184.8907, 159.1065, 184.7601 };
    float std_points[10] = { 30.2946, 51.6963, 65.5318, 51.5014, 48.0252, 71.7366, 33.5493, 92.3655, 62.7299, 92.2041 };

    // 假设输入图片中只有一张人脸
    //printf("\n\n");
    float facial_points[10];//坐标依次是：左眼X，左眼Y，右眼X，右眼Y，鼻尖X，鼻尖Y，左嘴角X，左嘴角Y，右嘴角X，右嘴角Y
    for (int i = 0; i < 5; i++) {
        facial_points[2*i] = faceInfoVar.landmark[2*i];
        facial_points[2*i+1] = faceInfoVar.landmark[2*i+1];
        //printf("%f, %f ", facial_points[2 * i], facial_points[2 * i + 1]);
    }
    //printf("\n");
    cv::Mat tform = getTformMatrix(std_points, facial_points);//这里得到的tform变换矩阵，是w=3, h=2, 即2*3


    warpAffine(image, dstImage, tform, dstImage.size(), 1, 0, cv::Scalar(0));
```

Affine transform

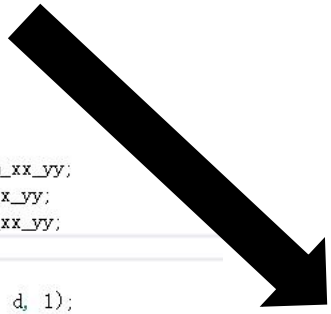


```
void CFaceDetectFMTCNN::NormalizeFace(cv::Mat dstImage, cv::Mat &normalizeImg)
{
    cv::Mat subfactor = 127.5 * cv::Mat(IMGNORHEIGHT, IMGNORWIDTH, CV_32FC3, cv::Scalar(1, 1, 1));
    dstImage.convertTo(normalizeImg, CV_32FC3);//变为浮点型
    normalizeImg = normalizeImg - subfactor;
    normalizeImg = normalizeImg / 128;//归一化，0-1
}
```

（1）调用 getTformMatrix 计算仿射变换矩阵 tform，将人脸关键点映射到标准模板点。
（2）使用 warpAffine 函数对输入图像执行仿射变换，并输出到目标图像 dstImage。
（3）对裁剪后的人脸图像进行归一化，确保像素值在 [−1, 1] 范围内。

实验三-关于Fast-MTCNN的使用

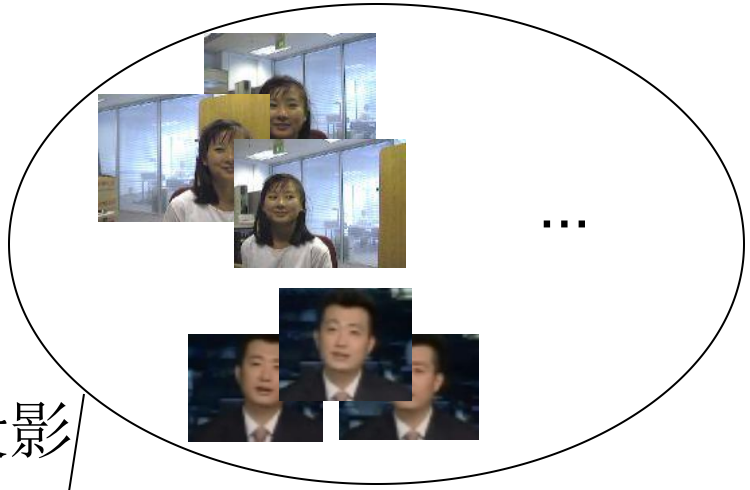
```
cv::Mat CFaceDetectFMTCNN::getTformMatrix(float* std_points, float* feat_points)
{
    int points_num_ = 5;
    double sum_x = 0, sum_y = 0;
    double sum_u = 0, sum_v = 0;
    double sum_xx_yy = 0;
    double sum_ux_vy = 0;
    double sum_vx__uy = 0;
    for (int c = 0; c < points_num_; ++c) {
        int x_off = c * 2;
        int y_off = x_off + 1;
        sum_x += std_points[c * 2];
        sum_y += std_points[c * 2 + 1];
        sum_u += feat_points[x_off];
        sum_v += feat_points[y_off];
        sum_xx_yy += std_points[c * 2] * std_points[c * 2] +
            std_points[c * 2 + 1] * std_points[c * 2 + 1];
        sum_ux_vy += std_points[c * 2] * feat_points[x_off] +
            std_points[c * 2 + 1] * feat_points[y_off];
        sum_vx__uy += feat_points[y_off] * std_points[c * 2] -
            feat_points[x_off] * std_points[c * 2 + 1];
    }
    double q = sum_u - sum_x * sum_ux_vy / sum_xx_yy
        + sum_y * sum_vx__uy / sum_xx_yy;
    double p = sum_v - sum_y * sum_ux_vy / sum_xx_yy
        - sum_x * sum_vx__uy / sum_xx_yy;
    double r = points_num_ - (sum_x * sum_x + sum_y * sum_y) / sum_xx_yy;
    double a = (sum_ux_vy - sum_x * q / r - sum_y * p / r) / sum_xx_yy;
    double b = (sum_vx__uy + sum_y * q / r - sum_x * p / r) / sum_xx_yy;
    double c = q / r;
    double d = p / r;
    cv::Mat Tinv = (cv::Mat_<float>(3, 3) << a, b, 0, -b, a, 0, c, d, 1);
    cv::Mat T = Tinv.inv();
    cv::Mat res = T.colRange(0, 2).clone();
    return res.t();
}
```



align_crop_example.cpp

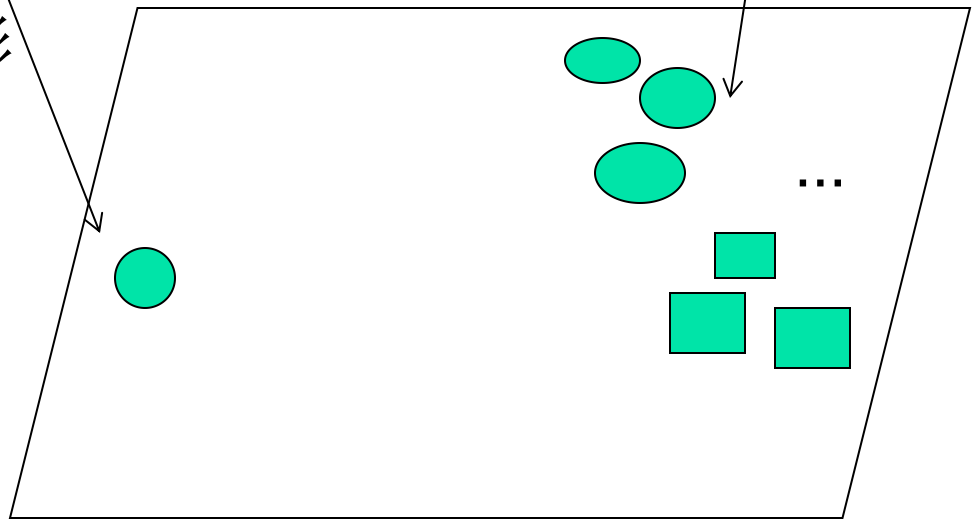
该函数计算从 feat_points 到 std_points 的仿射变换矩阵。
仿射变换通过旋转、缩放和平移操作实现。

实验三-分类框架的设计举例1-PCA方法

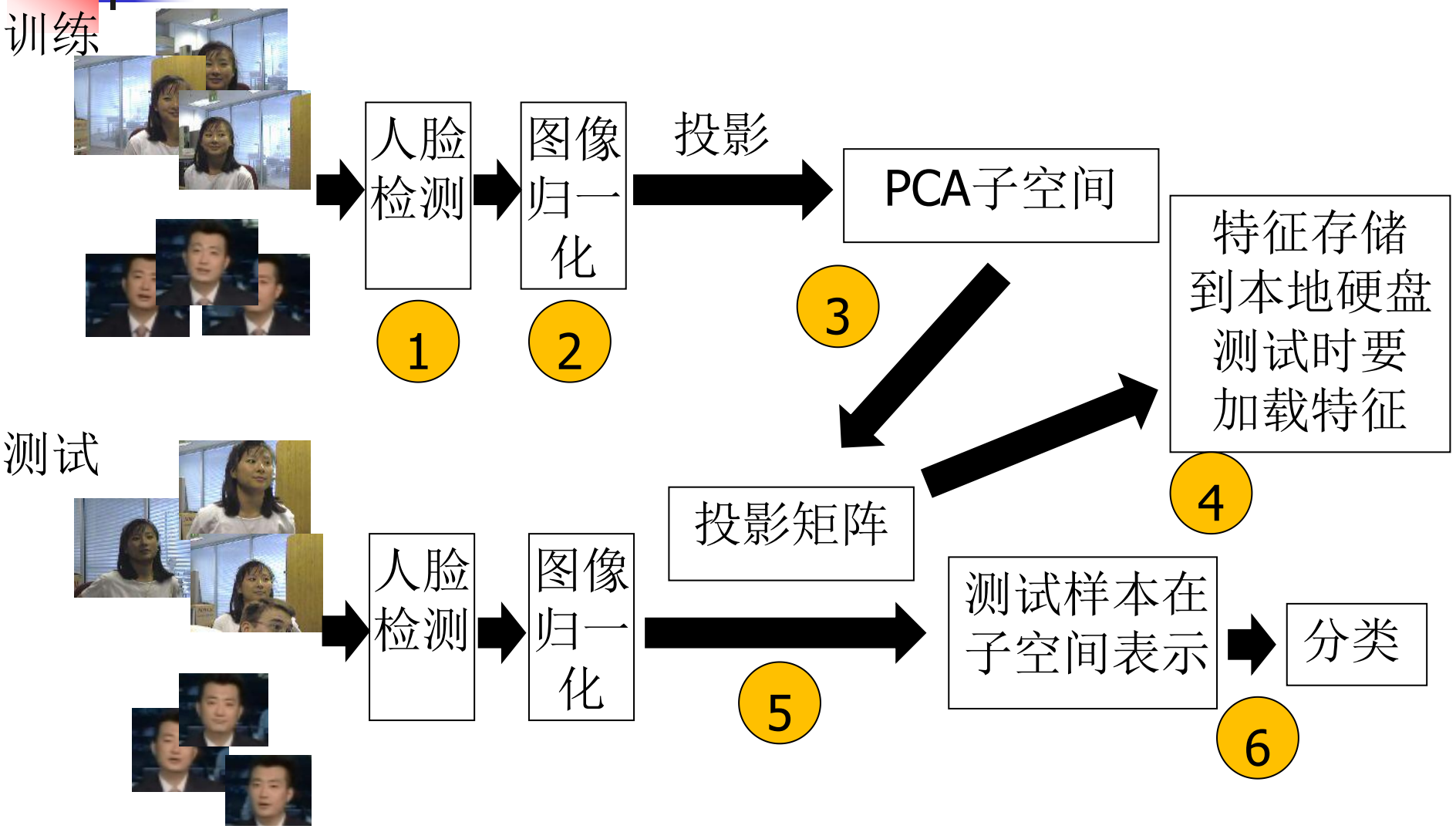


PCA投影

PCA投影



实验三-分类框架的设计举例1-PCA方法



实验三-分类框架的设计举例2-深度特征

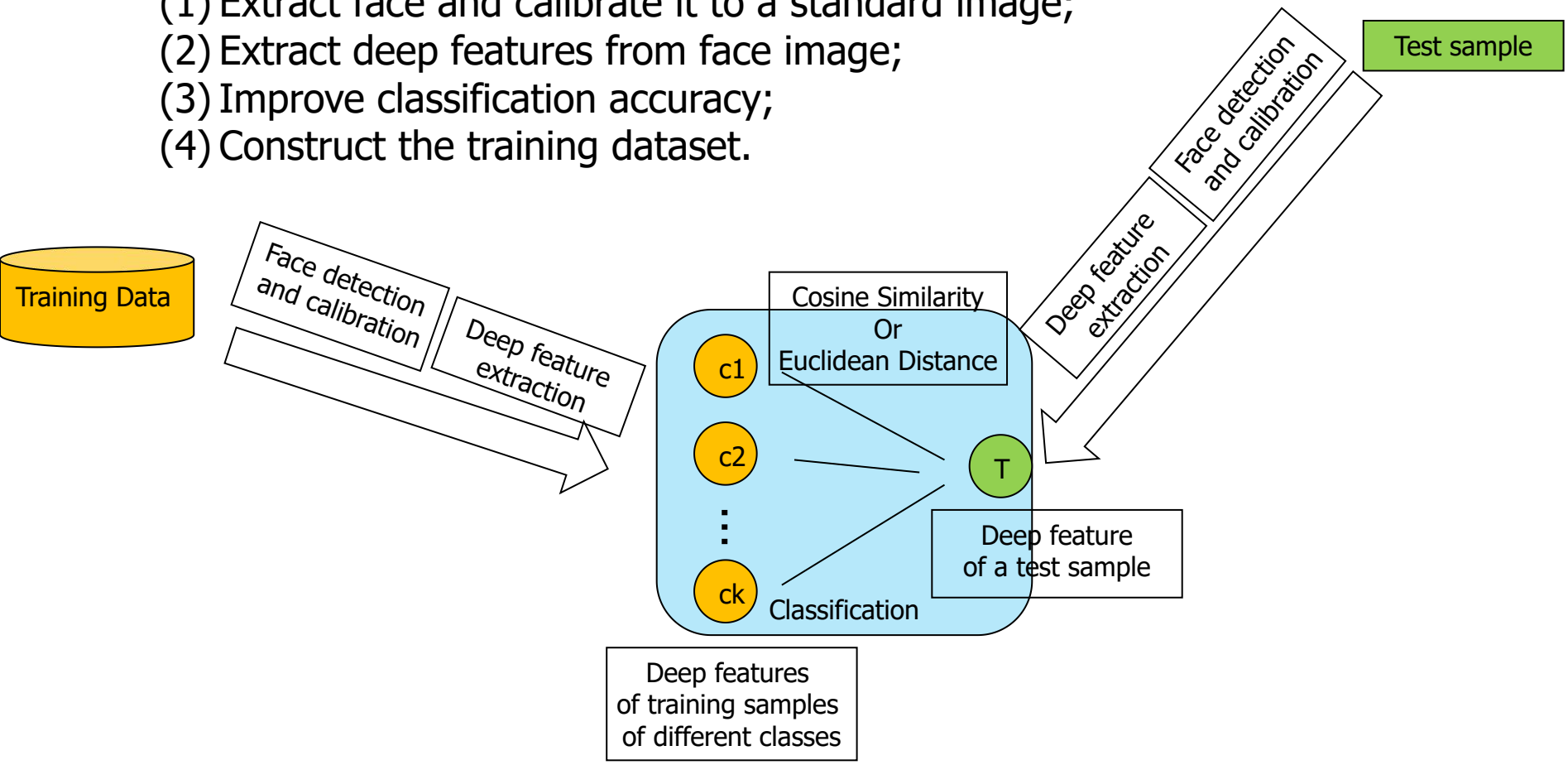
The main goals:

(1) Extract face and calibrate it to a standard image;

(2) Extract deep features from face image;

(3) Improve classification accuracy;

(4) Construct the training dataset.



实验三-分类框架的设计举例2-深度特征

sphereface_deploy.prototxt
sphereface_model.caffemodel

https://github.com/wy1iu/sphereface

```
name: "SpherefaceNet-20"
input: "data"
input_dim: 1
input_dim: 3
input_dim: 112
input_dim: 96
############### CNN Architecture ###############
layer {
  name: "conv1_1"
  type: "Convolution"
  bottom: "data"
  top: "conv1_1"
```

Network input

```
layer {
  name: "fc5"
  type: "InnerProduct"
  bottom: "res4_3"
  top: "fc5"
  param {
    lr_mult: 1
    decay_mult: 1
  }
  param {
    lr_mult: 2
    decay_mult: 0
  }
  inner_product_param {
    num_output: 512
    weight_filler {
      type: "xavier"
    }
    bias_filler {
      type: "constant"
      value: 0
    }
  }
}
```

Network Output

The extracted 512 dimensional features can be used for the recognition purpose

Deep feature extraction based on OpenCV

```
Net net;
string label_dir = TrainingSet_dir+"/Labels.txt";

//Specify Caffe Model
//modelTxt = proto_model_dir + "/face_deploy.prototxt";
//modelBin = proto_model_dir + "/face_model.caffemodel";

modelTxt = proto_model_dir + "/sphereface_deploy.prototxt";
modelBin = proto_model_dir + "/sphereface_model.caffemodel";

// Read Caffe Network
net = cv::dnn::readNetFromCaffe(modelTxt, modelBin);
if (net.empty())
{
    std::cerr << "Can't load network by using the following files: " << std::endl;
    exit(-1);
}
```

1. Read Caffe Network

```
cv::String imageFile = "mytest.jpg";
cv::Mat image = cv::imread(imageFile, 1);// Color image
if (image.empty())
{
    std::cerr << "Can't read image from the file: " << imageFile << std::endl
    exit(-1);
}
cerr << "image read sucessfully" << endl;
```

2. Load Image

```
//Detect Face
CFaceDetectFMTCNN DetectObj;
vector<FaceInfo> faceInfo;
cv::Mat copyImg;
image.copyTo(copyImg);
DetectObj.DetectFace_FMTCNN(image, faceInfo);
//显示画图
DetectObj.DispFaceBoxLandMarks(copyImg, faceInfo);
cv::imshow("image", copyImg);
```

3. Detect Face

```
cv::Mat dstImage(IMGNORHEIGHT, IMGNORWIDTH, CV_8UC3);

for (int i = 0; i < faceInfo.size(); i++)//对每一个人脸进行处理
{
    FaceInfo faceInfoVar = faceInfo[i];
    DetectObj.RotateFace(image, faceInfoVar, dstImage);

    //Normalize Face
    cv::Mat normalizeImg;
    normalizeImg.create(dstImage.size(), CV_32FC3);
    DetectObj.NormalizeFace(dstImage, normalizeImg);
    //waitKey(0);
```

4. Face Calibration

(1) Affine transform
(2) Normalization
- image value – mean value
- Image values are set to 0~1

```
    //Construct the inptBlob
    //Mat inputBlob = blobFromImage(img_new, 1.0 / 255, Size(28, 28));// 必须要归一化
    cv::Mat inputBlob = blobFromImage(normalizeImg);
    cv::Mat FeatureOutput;//Feature of the outputed inner_product layer
    net.setInput(inputBlob, "data");//set the network input, "data" is the name of the input laye
    FeatureOutput = net.forward("fc5");//compute output, "fc5" is the name of the output layer
    FeatureOutput.copyTo(faceInfo[i].FeatureVector);

    printf("FeatureOutpur Size: rows=%d, cols=%d\n", FeatureOutput.rows, FeatureOutput.cols);
    for (int i = 0; i<FeatureOutput.rows; i++)
    {
        for (int j = 0; j<FeatureOutput.cols; j++)
        {
            printf("%f ", FeatureOutput.ptr<float>(i)[j]);
        }
        printf("\n");
    }
```

5. Input image data into network

6. Network forward and feature extraction

实验三

人脸检测与人脸识别概述：
https://blog.csdn.net/daydayup858/article/details/128346738

Fast-MTCNN人脸检测参考项目：
https://github.com/szad670401/Fast-MTCNN

SphereFace人脸识别：
https://github.com/wy1iu/sphereface

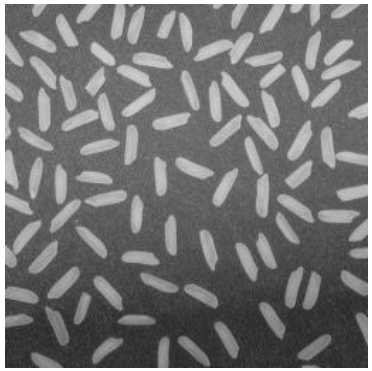
faceNet人脸识别：
https://github.com/davidsandberg/facenet
FaceNet:A Unified Embedding for Face Recognition and Clustering

。 。 。

上机练习1

实验课上机练习内容：

1. OpenCV环境配置；
2. 实现对图像的读取；
3. 把一副彩色图像的三个通道变成3个单通道图像存储到硬盘上并显示；
4. 计算一幅单通道图像的直方图；
5. 编程实现对一幅单通道图像的边缘检测。
6. 对一幅灰度图像进行直方图均衡化
7. 对图像进行二值化操作；
8. 图像形态学操作；
9. 获取图像轮廓；
10. 实现目标的计数；
11. 目标的旋转；



上机练习2

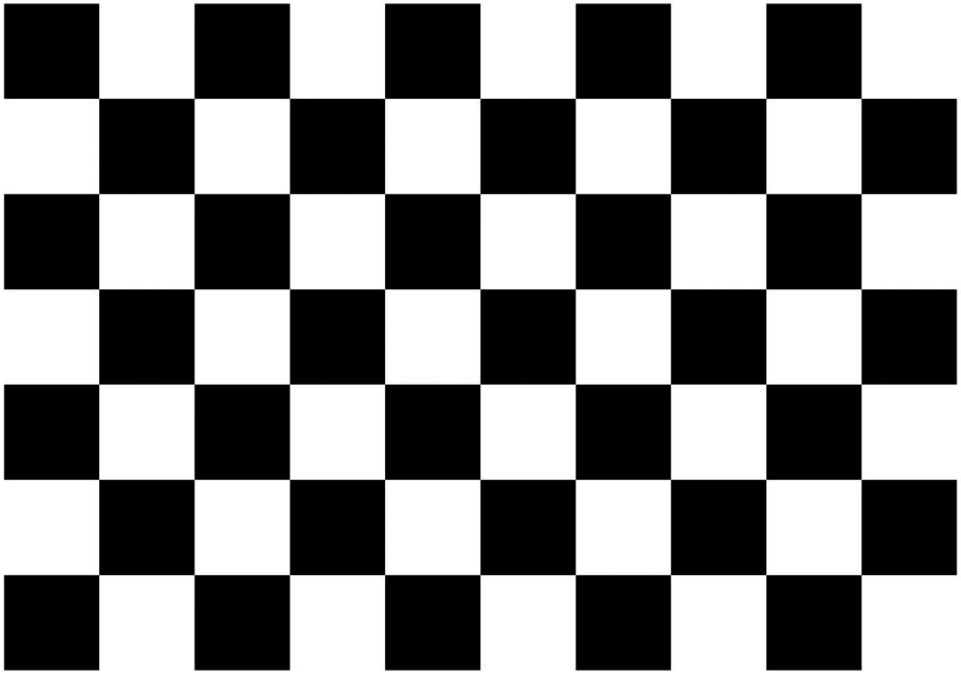
上机练习内容：

1.在图像中实现目标匹配；（先获取该目标的一个图像，然后利用这个已知的图像，在测试图像上找到该目标）

上机练习2

上机练习内容：

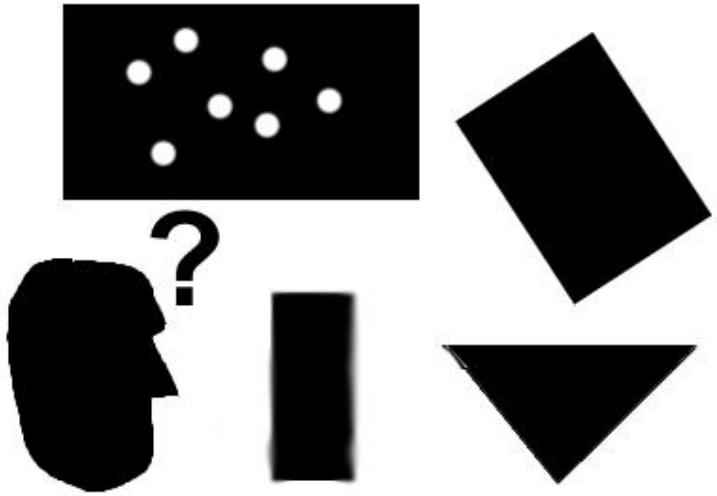
2.在图像上通过霍夫变换寻找直线；

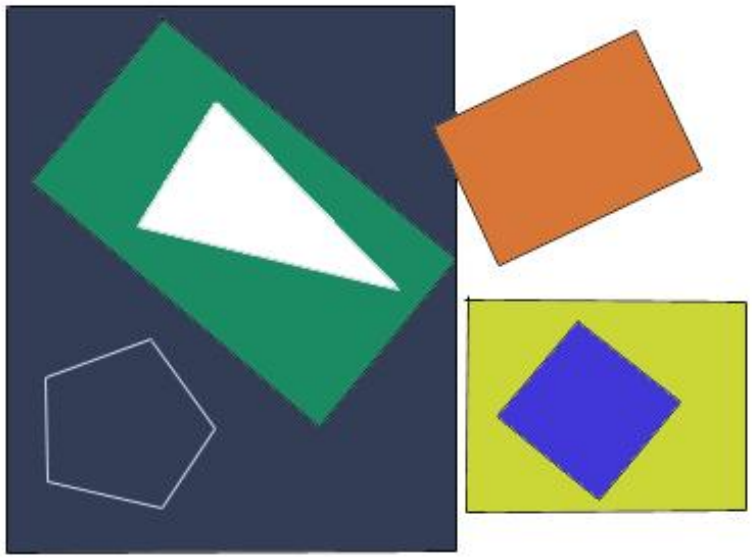


上机练习2

上机练习内容：

3.基于霍夫变换，实现三角形检测，如果是四边形呢？多边形呢？

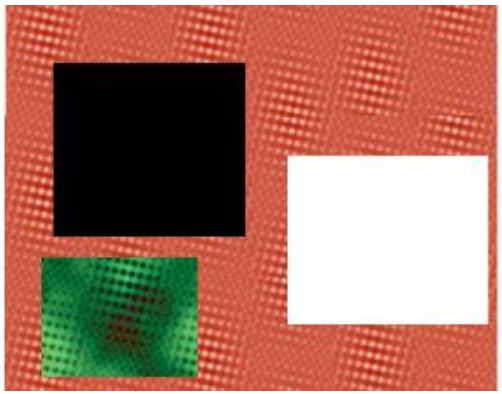


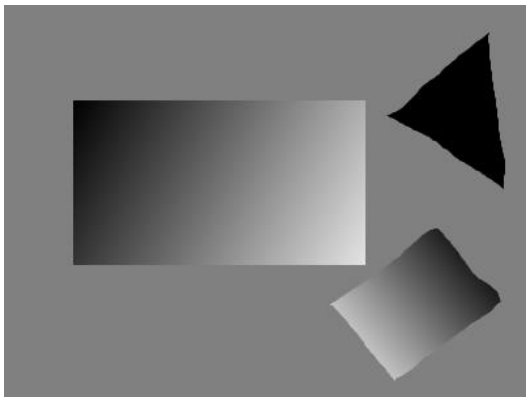


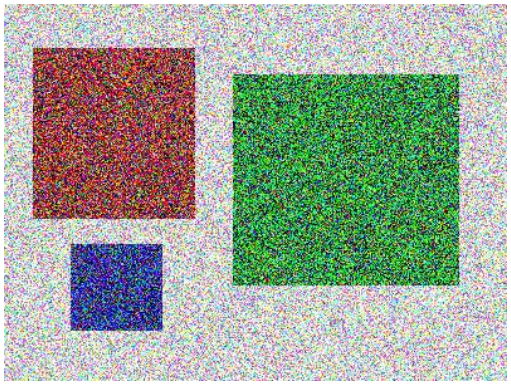
上机练习2

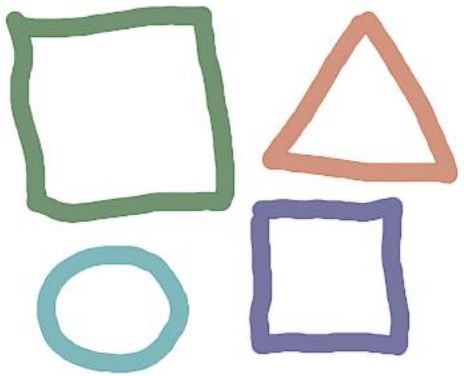
上机练习内容：

3.基于霍夫变换，实现三角形或矩形检测，如果是四边形呢？多边形呢？









上机练习2

上机练习内容：

4.练习基于OpenCV的视频读取。

实验二：图像处理综合-路沿检测

已附加文件：计算机视觉-学号-姓名-实验报告2.doc (140 KB)
实验2实验数据.zip (16.718 MB)
其他实验图像和视频.zip (17.323 MB)
实验课-第3节.pdf (1.986 MB)

实验目的：

1. 熟悉图像处理基本操作；
2. 掌握图像边缘检测原理；
3. 掌握图像基本特征抽取以及在实际问题中的应用；

实验要求：

1. 实验提交文件为实验报告和相关程序代码，以压缩包的形式提交，实验报告命名规则为“计算机视觉-学号-姓名-实验报告2.doc”，其
2. 所有素材和参考材料需列明出处，实验报告中的图片和程序代码建议标注个人水印或标识信息：姓名，班级，学号信息；

二、实验内容：

针对给定的视频，利用图像处理基本方法实现道路路沿的检测；

提示：可利用Hough变换进行线检测，融合路沿的结构信息实现路沿边界定位（图中红色的点位置）。

三、实验时间

实验报告统一命名为：计算机视觉-学号-姓名-实验报告2.doc

如有提交其他文件或程序，统一命名为：计算机视觉-学号-姓名-实验报告2-其他.zip

上机练习2

上机练习内容：

4.练习基于OpenCV的视频读取。

实验二：图像处理综合-路沿检测

已附加文件：计算机视觉-学号-姓名-实验报告2.doc (140 KB)
实验2实验数据.zip (16.718 MB)
其他实验图像和视频.zip (17.323 MB)
实验课-第3节.pdf (1.986 MB)

实验目的：

1. 熟悉图像处理基本操作；
2. 掌握图像边缘检测原理；
3. 掌握图像基本特征抽取以及在实际问题中的应用；

实验要求：

1. 实验提交文件为实验报告和相关程序代码，以压缩包的形式提交，实验报告命名规则为“计算机视觉-学号-姓名-实验报告2.doc”，其
2. 所有素材和参考材料需列明出处，实验报告中的图片和程序代码建议标注个人水印或标识信息：姓名，班级，学号信息；

二、实验内容：

针对给定的视频，利用图像处理基本方法实现道路路沿的检测；

提示：可利用Hough变换进行线检测，融合路沿的结构信息实现路沿边界定位（图中红色的点位置）。

三、实验时间

实验报告统一命名为：计算机视觉-学号-姓名-实验报告2.doc

如有提交其他文件或程序，统一命名为：计算机视觉-学号-姓名-实验报告2-其他.zip

上机练习3

- 基于最优梯度幅值的边缘检测
- 基于迭代二分法的自动阈值分割
- 基于直方图的颜色特征提取
- 彩色直方图特征提取