The purpose of this document is to demonstrate the preprocess of iris image dataset.

Due to low sharpness and high similarity of each iris scan photo, we used MATLAB to get feature of images. Then we can use the LGBP feature to train and predict which class the test image is. The code for set up classification algorithms and train the model is coded by python, the python code and introduction of this research is documented in a jupyter notebook .

The environment we run on is MATLAB\_R2018a

Here is the code:

First, below is a function to get image matirx

function M = getImageMatrix(path,fileFormat, allMatFilePath, perClassOutput, allName, className)

fileList = dir([path fileFormat]);

%filepath = strcat(path, fileList(1).name)

% fileList(1).name

% fileList(1)

%   imageName = fileList(1).name;

%      imageData = imread(filepath);

%      imshow(imageData)

if ~exist(allMatFilePath)

    mkdir(allMatFilePath);

end

M = {};

for i = 1: length(fileList)

    imageName = fileList(i).name;

    imagePath = strcat(path, imageName);

    imageData = imread(imagePath);

    if numel(size(imageData)) > 2

         imageGrayData = rgb2gray(imageData);

    else

        imageGrayData = imageData;

    end

    if strcmp(className, 'lion') % for lion, need cut

        imageGrayData = imageGrayData(19:560, 63:706);

    else

        imageGrayData = imageGrayData;

    end

    M{i,1} = imageGrayData;

    M{i,2} = imageName;

    imageMatName = strcat(allMatFilePath, imageName(1:end-4));

    imageMat = strcat(imageMatName, '.mat');

    save(imageMat, 'imageGrayData');

end

save(strcat(perClassOutput, allName), 'M')

end

Then, here is the code to get feature matrix:

clc;

clear;

LGBPPath = './LGBP/';

GLCMPath = './GLCM/';

LGBPMatrixDataPath = './matrixLGBP/';

GLCMMatrixDataPath = './matrixGLCM/';

if ~exist(LGBPMatrixDataPath)

    mkdir(LGBPMatrixDataPath)

end

if ~exist(GLCMMatrixDataPath)

     mkdir(GLCMMatrixDataPath)

end

LGBPAsianTest = strcat(LGBPPath,'LGBPAsianTest/LGBPFeature.mat');

LGBPAsianTrain = strcat(LGBPPath,'LGBPAsianTrain/LGBPFeature.mat');

LGBPHuman = strcat(LGBPPath,'LGBPHuman/LGBPFeature.mat');

LGBPHumanGlass = strcat(LGBPPath,'LGBPHumanGlass/LGBPFeature.mat');

LGBPLions = strcat(LGBPPath,'LGBPLions/LGBPFeature.mat');

LGBPWhiteTest =  strcat(LGBPPath,'LGBPWhiteTest/LGBPFeature.mat');

LGBPWhiteTrain =  strcat(LGBPPath,'LGBPWhiteTrain/LGBPFeature.mat');

%%

load(LGBPLions)

LGBPLions = [];

LGBPLionsName = {};

for i = 1:size(LGBPData,1)

    LGBPLions(i,:) = LGBPData{i,1};

    LGBPLionsName{i,1} = LGBPData{i,2};

end

LGBPMatrixDataPathLion = strcat(LGBPMatrixDataPath,'LGBPLions.mat');

LGBPMatrixDataPathLionName = strcat(LGBPMatrixDataPath,'LGBPLionsName.mat');

save(LGBPMatrixDataPathLion,'LGBPLions')

save(LGBPMatrixDataPathLionName,'LGBPLionsName')

%%

load(LGBPHuman)

LGBPHuman = [];

LGBPHumanName = {};

for i = 1:size(LGBPData,1)

    LGBPHuman(i,:) = LGBPData{i,1};

    LGBPHumanName{i,1} = LGBPData{i,2};

end

LGBPMatrixDataPathHuman = strcat(LGBPMatrixDataPath,'LGBPHuman.mat');

LGBPMatrixDataPathHumanName = strcat(LGBPMatrixDataPath,'LGBPHumanName.mat');

save(LGBPMatrixDataPathHuman,'LGBPHuman')

save(LGBPMatrixDataPathHumanName,'LGBPHumanName')

%%

load(LGBPHumanGlass)

LGBPHumanGlass = [];

LGBPHumanGlassName = {};

for i = 1:size(LGBPData,1)

    LGBPHumanGlass(i,:) = LGBPData{i,1};

   LGBPHumanGlassName{i,1} = LGBPData{i,2};

end

LGBPMatrixDataPathHumanGlass = strcat(LGBPMatrixDataPath,'LGBPHumanGlass.mat');

LGBPMatrixDataPathHumanGlassName = strcat(LGBPMatrixDataPath,'LGBPHumanGlassName.mat');

save(LGBPMatrixDataPathHumanGlass,'LGBPHumanGlass')

save(LGBPMatrixDataPathHumanGlassName,'LGBPHumanGlassName')

%%

load(LGBPAsianTrain)

LGBPAsianTrain = [];

LGBPAsianTrainName = {};

for i = 1:size(LGBPData,1)

    LGBPAsianTrain(i,:) = LGBPData{i,1};

   LGBPAsianTrainName{i,1} = LGBPData{i,2};

end

LGBPMatrixDataPathAsianTrain = strcat(LGBPMatrixDataPath,'LGBPAsianTrain.mat');

LGBPMatrixDataPathAsianTrainName = strcat(LGBPMatrixDataPath,'LGBPAsianTrainsName.mat');

save(LGBPMatrixDataPathAsianTrain,'LGBPAsianTrain')

save(LGBPMatrixDataPathAsianTrainName,'LGBPAsianTrainName')

%%

load(LGBPAsianTest)

LGBPAsianTest = [];

LGBPAsianTestName = {};

for i = 1:size(LGBPData,1)

    LGBPAsianTest(i,:) = LGBPData{i,1};

   LGBPAsianTestName{i,1} = LGBPData{i,2};

end

LGBPMatrixDataPathAsianTest = strcat(LGBPMatrixDataPath,'LGBPAsianTest.mat');

LGBPMatrixDataPathAsianTestName = strcat(LGBPMatrixDataPath,'LGBPAsianTestName.mat');

save(LGBPMatrixDataPathAsianTest,'LGBPAsianTest')

save(LGBPMatrixDataPathAsianTestName,'LGBPAsianTestName')

%%

load(LGBPWhiteTrain)

LGBPWhiteTrain = [];

LGBPWhiteTrainName = {};

for i = 1:size(LGBPData,1)

    LGBPWhiteTrain(i,:) = LGBPData{i,1};

   LGBPWhiteTrainName{i,1} = LGBPData{i,2};

end

LGBPMatrixDataPathWhiteTrain = strcat(LGBPMatrixDataPath,'LGBPWhiteTrain.mat');

LGBPMatrixDataPathWhiteTrainName = strcat(LGBPMatrixDataPath,'LGBPWhiteTrainName.mat');

save(LGBPMatrixDataPathWhiteTrain,'LGBPWhiteTrain')

save(LGBPMatrixDataPathWhiteTrainName,'LGBPWhiteTrainName')

%%

load(LGBPWhiteTest)

LGBPWhiteTest = [];

LGBPWhiteTestName = {};

for i = 1:size(LGBPData,1)

    LGBPWhiteTest(i,:) = LGBPData{i,1};

  LGBPWhiteTestName{i,1} = LGBPData{i,2};

end

LGBPMatrixDataPathLGBPWhiteTest = strcat(LGBPMatrixDataPath,'LGBPWhiteTest.mat');

LGBPMatrixDataPathLGBPWhiteTestName = strcat(LGBPMatrixDataPath,'LGBPWhiteTestName.mat');

save(LGBPMatrixDataPathLGBPWhiteTest,'LGBPWhiteTest')

save(LGBPMatrixDataPathLGBPWhiteTestName,'LGBPWhiteTestName')

At the end is the main function to set images path and call the functions we created.

clc;

clear;

class\_mat\_output = './perClassMat/';

if ~exist(class\_mat\_output)

    mkdir(class\_mat\_output);

end

human\_path = '../../experiment\_database/CASIA-Iris-Thousand/';

human\_output\_path = './thousandAllMat/';

human\_format = '\*.jpg';

human\_all\_name = 'thousandAllMat.mat';

class = 'thousand';

M\_human = getImageMatrix(human\_path, human\_format, human\_output\_path,class\_mat\_output,human\_all\_name,class);