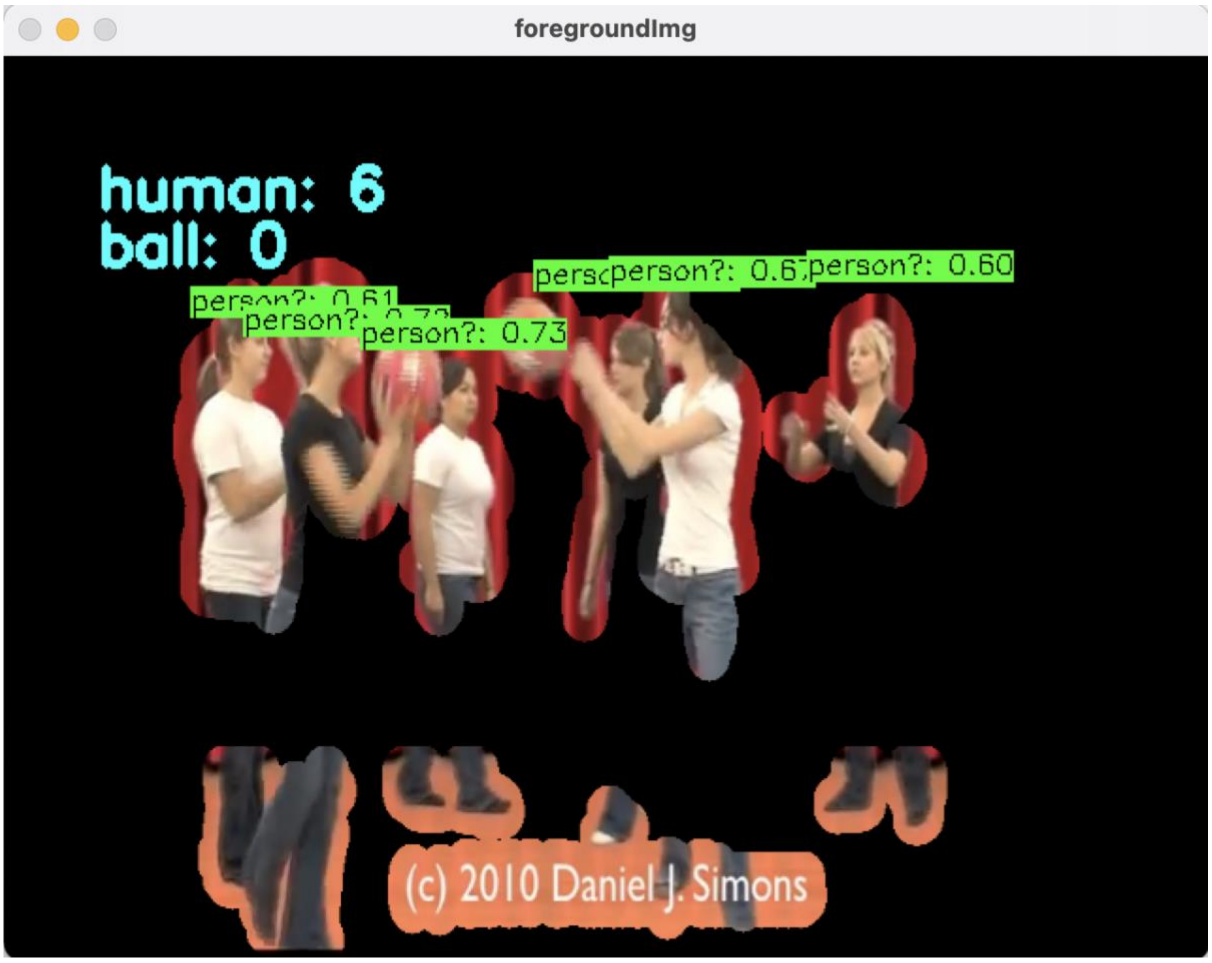
Quiz2 Report

Computer Vision
Section 02
21800181 Kim Jisu

Quiz 2

- Result



- Code

```
#include "opencv2/opencv.hpp"
#include <iostream>
#include "opencv2/objdetect.hpp"
#include "opencv2/highgui.hpp"
#include "opencv2/imgproc.hpp"
#include "opencv2/videoio.hpp"
#include "cv.hpp"
#include <opencv2/dnn.hpp>
#include <fstream>
```

```
using namespace std;
using namespace cv;
using namespace dnn;

int main() {
    Mat temp;
    Mat img_roi_1, img_roi_2,img_roi_3;
    VideoCapture cap("Quiz2 Video.mp4");
    Mat background, image, gray, result, foregroundMask, foregroundImg;

    String modelConfiguration = "deep/yolov2-tiny.cfg";
    String modelBinary = "deep/yolov2-tiny.weights";
    Net net = readNetFromDarknet(modelConfiguration, modelBinary);
    vector<String> classNamesVec;
    ifstream classNamesFile("deep/coco.names");
    if (classNamesFile.is_open()) {
        string className = "";
        while (std::getline(classNamesFile, className)) classNamesVec.push_back(className);
    }

    Mat a, sum, avg;
    cap >> temp;
    img_roi_3 = temp(Rect(0, 360, 830, 20));
    img_roi_2 = temp(Rect(0, 410, 830, 10));
    Mat img_roi_2_1 = temp(Rect(140, 410, 600, 10));

    VideoCapture capture("Quiz2 Video.mp4");
    Mat frame;
    Mat element = getStructuringElement(MORPH_ELLIPSE, Size(30, 30));

    while(true){
        cap >> image;
        if (image.empty()) {
            waitKey();
            break;
        }
        img_roi_1 = image(Rect(0, 10, 830, 10));
        Mat img_roi_1_2 = image(Rect(0, 10, 830, 5));
Mat frame = image.clone();
for(int i = 10;i<360;i+=10){
        Mat frmtoroi;
        frmtoroi = frame(Rect(0, i, 830, 10));
        img_roi_1.copyTo(frmtoroi);
}
Mat frmtoroi;

frmtoroi = frame(Rect(140, 380, 600, 10));
img_roi_2_1.copyTo(frmtoroi);
for(int i = 390;i<470;i+=10){
        Mat frmtoroi;
        frmtoroi = frame(Rect(0, i, 830, 10));
        img_roi_2.copyTo(frmtoroi);
}
background = frame.clone();
cvtColor(background, background, CV_BGR2GRAY);
cvtColor(image, gray, CV_BGR2GRAY);
absdiff(background, gray, foregroundMask);
threshold(foregroundMask, foregroundMask, 90, 255, CV_THRESH_BINARY);
dilate(foregroundMask, foregroundMask, element);


foregroundMask.copyTo(foregroundImg);
image.copyTo(foregroundImg, foregroundMask);



resize(foregroundImg, foregroundImg, Size(640, 480));
resize(image, image, Size(640, 480));
vector<Vec3f> circles;
Mat g_;
g_ = foregroundImg.clone();
cvtColor(g_,g_,CV_BGR2GRAY);
HoughCircles(g_, circles, HOUGH_GRADIENT, 1, 100, 50, 35, 20, 40);

if (image.channels() == 4) cvtColor(image, image, COLOR_BGRA2BGR);
Mat inputBlob = blobFromImage(image, 1 / 255.F, Size(416, 416), Scalar(), true, false);
net.setInput(inputBlob, "data");                    //set the network input
Mat detectionMat = net.forward("detection_out");    //compute output
```

```
        float confidenceThreshold = 0.2; //by default
                int count = 0;
        for (int i = 0; i < detectionMat.rows; i++)       {

            const int probability_index = 5;
            const int probability_size = detectionMat.cols - probability_index;
            float *prob_array_ptr = &detectionMat.at<float>(i, probability_index);
            size_t objectClass = max_element(prob_array_ptr, prob_array_ptr + probability_size) -
prob_array_ptr;
            // prediction probability of each class
            float confidence = detectionMat.at<float>(i, (int)objectClass + probability_index);

            // for drawing labels with name and confidence
            if (confidence > confidenceThreshold) {

                float x_center = detectionMat.at<float>(i, 0) * image.cols;
                float y_center = detectionMat.at<float>(i, 1) * image.rows;
                float width = detectionMat.at<float>(i, 2) * image.cols;
                float height = detectionMat.at<float>(i, 3) * image.rows;
                Point p1(cvRound(x_center - width / 2), cvRound(y_center - height / 2));
                Point p2(cvRound(x_center + width / 2), cvRound(y_center + height / 2));
                Rect object(p1, p2);
                Scalar object_roi_color(0, 255, 0);

                rectangle(image, object, object_roi_color);
                String className = objectClass < classNamesVec.size() ? classNamesVec[objectClass] :
cv::format("unknown(%d)", objectClass);
                String label = format("%s: %.2f", className.c_str(), confidence);
                char myCopied[30];
                strcpy(myCopied, className.c_str());
                if(strcmp(myCopied,"person")==13)
                    count++;
                int baseLine = 0;

                Size labelSize = getTextSize(label, FONT_HERSHEY_SIMPLEX, 0.5, 1, &baseLine);

                rectangle(foregroundImg, Rect(p1, Size(labelSize.width, labelSize.height + baseLine)),
object_roi_color, FILLED);
```

```
                                putText(foregroundImg, label, p1 + Point(0, labelSize.height), FONT_HERSHEY_SIMPLEX, 
0.5, Scalar(0, 0, 0));
                                                        }

                                                }

                putText(foregroundImg, format("human: %d",count ), Point(50, 80), FONT_HERSHEY_SIMPLEX, 1, 
Scalar(255, 255, 0), 4);
                putText(foregroundImg, format("ball: %d",circles.size() ), Point(50, 110), FONT_HERSHEY_SIMPLEX, 
1, Scalar(255, 255, 0), 4);

                if (waitKey(1) >= 0) break;
                imshow("foregroundImg", foregroundImg);
                waitKey(33);
        }

}
```