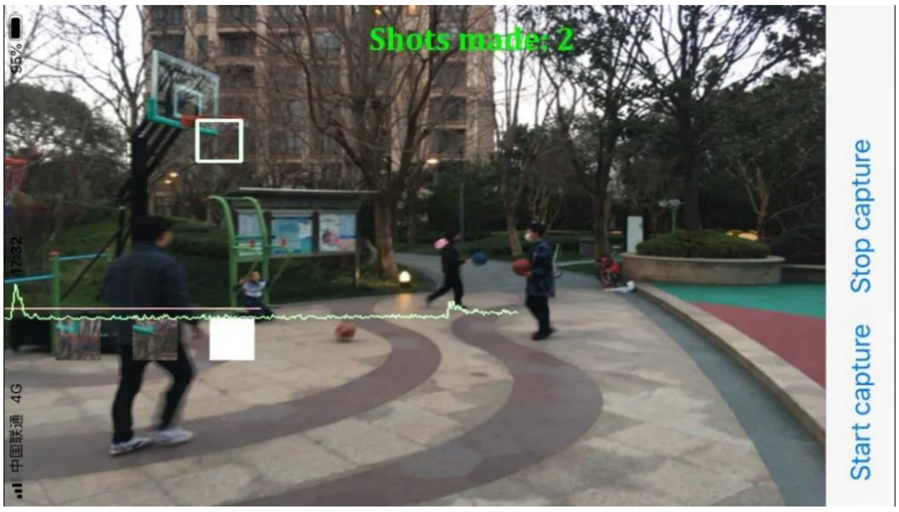
Basketball Goal Detection App

Mingjun Gao

iOS App used to identify basketball goals

GitHub Link: (https://github.com/jadongao/basketball)



1. App Overview

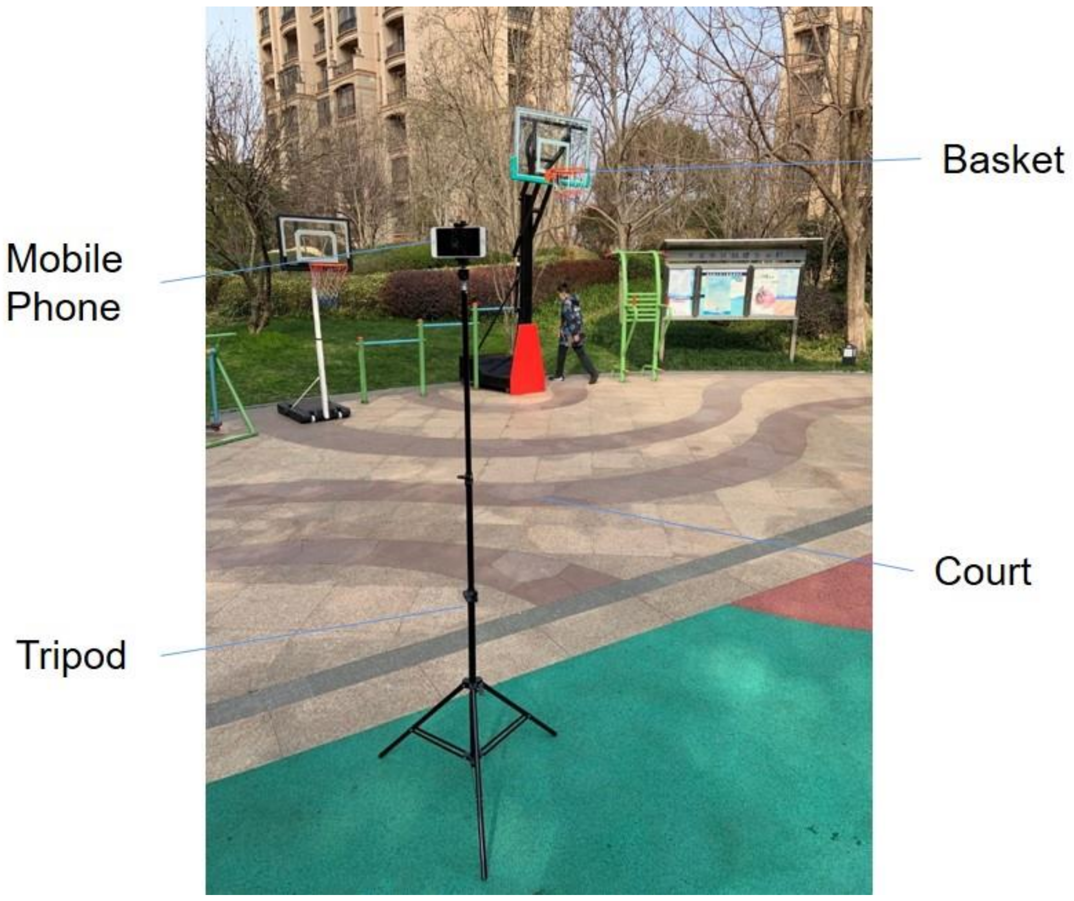
During the COVID pandemic, it became impossible for our basketball team to train together. The coach had no way of knowing each player's status, so I developed this app to detect basketball goals for my team to help keep track of their performances. All they need to do is to set up their phone to record their training.

This app's main idea is to recognize color changes in the basket. If a ball enters the recognized area, the program will display a "ball in" symbol and count the total score. This program uses an open-source computer vision library called OpenCV. The process can be broken down as follows:

1) Import the video.

2) Use OpenCV to monitor the basket area and observe the color changes to identify whether the basketball is in the basket. Something to consider here is the workload of video processing on phones. I needed to ensure it could process the video live and give timely feedback.

3) Design a user interface that is easy to set up and recognize the basket.



Each player would carry their phones and tripods with them and place them on the side of the basketball court, facing the direction of the basket.

1. A tripod is used to avoid video shaking.
2. Mobile phones (camera+screen+video recognition software) consist of video recognition and scoring functions.

2. Development Details

2.1 Learn OpenCV on a Windows computer

Idea:

1. Mark the rectangular area *rcHoop* of the basket;
2. Back up the *hoopBack* in the basket area of the first frame of the video;
3. Read the *hoopCurrent* of the basket area in the current frame;
4. Compare the backup and current region, obtain *hoopDiff*, and calculate the mean *m*;
5. Use the value of *m* to determine whether to use the basketball to enter the basket area.

```
#include <iostream>
#include <opencv2/imgproc.hpp>
#include <opencv2/highgui.hpp>

using namespace std;
using namespace cv;

int main(int argc, char** argv)
{
        cv::VideoCapture cap1("C:\\201905.MP4");
        cv::Mat frame;
        cap1.read(frame);
        Rect rcHoop = cv::selectROI("read", frame, true);
        cv::Mat hoopBack = frame(rcHoop).clone();     //backup
        cv::Mat hoopDiff;
        while (cap1.read(frame))
        {
                cv::Mat hoopCurrent = frame(rcHoop).clone();  //current
                cv::absdiff(hoopBack, hoopCurrent, hoopDiff);
//difference
                int dilation_size = 2;
                cv::Mat element = getStructuringElement(MORPH_RECT,
Size(2 * dilation_size + 1, 2 * dilation_size + 1), Point(dilation_size,
dilation_size));
                cv::erode(hoopDiff, hoopDiff, element);  //erode
algorithm
                threshold(hoopDiff, hoopDiff, 5, 255, CV_THRESH_BINARY);

                cv::Mat gray;
                cvtColor(hoopDiff, gray, CV_RGB2GRAY);
                double m = mean(gray)[0];      //m value of difference

                cout << m << endl;
                if (m > 20)
                {
                        putText(frame, "Ball in!!!", Point(40, 40),
                                CV_FONT_HERSHEY_COMPLEX, 1, CV_RGB(0,
255, 0), 2);
                }

                matDiff.copyTo(frame(rcBallIn));
                cv::rectangle(frame, rcBallIn, CV_RGB(0, 255, 0), 2);
                cv::imshow("read", frame);
                cv::waitKey(30);
                cap1.read(frame);
        }
} // main
```

Absdiff(), erode() are functions in the open-source library of OpenCV that are used to backup the differences between the image and the current region.

The actual effect is as follows: After scoring a goal, the words "ball in!!!" can be displayed.

The ball is above and below the basket, without the words "ball in!!!" appearing.

2.2 Using OpenCV on iOS

2.2.1 Button

The Start button can open the phone's camera, capture the video at 30 frames per second, and display it on the phone screen. Each Start button press can also save the basket recognition area to HoopBack.

```
- (IBAction)startCaptureButtonPressed:(id)sender {
    [videoCamera start];
    isCapturing = YES;
    imageCount = 0;
}

- (IBAction)ClearScoreButtonPressed:(id)sender {
    intScore = 0;
}
```

```
- (IBAction)stopCaptureButtonPressed:(id)sender {
    //[videoCamera stop];
    isCapturing = !isCapturing;
}
```

Press the Stop button once to freeze the screen image, then press it again to restore the video display

2.2.2 Basket area recognition

The video images in this area correspond to hoopCurrent, and m and m2 values are calculated for each frame. Then determine if the basketball is in the basket

```
- (void)processImage:(cv::Mat &)image {
    // Do some OpenCV processing with the image
    cv::Mat inputFrame = image.clone();

    cv::Rect rcBallIn(150, 110, 55, 60);
    hoopBack = inputFrame(rcBallIn).clone();   //backup
    hoopCurrent = inputFrame(rcBallIn).clone();  //current
    cv::absdiff(hoopBack, hoopCurrent, hoopDiff);   //difference

    int dilation_size = 2;
    cv::Mat element = getStructuringElement(cv::MORPH_RECT, cv::Size(2 *
dilation_size + 1, 2 * dilation_size + 1), cv::Point(dilation_size,
dilation_size));
    cv::erode(hoopDiff, hoopDiff, element);     //erode algorithm
    threshold(hoopDiff, hoopDiff, 5, 255, CV_THRESH_BINARY);

    cv::Mat gray;
    cvtColor(hoopDiff, gray, CV_RGB2GRAY);
    double m = mean(gray)[0];

    if (m > 10)
    {
           intScore ++;
           putText(inputFrame, "Ball in!!!", cv::Point(40, 40),
               CV_FONT_HERSHEY_COMPLEX, 1, CV_RGB(255, 255, 255), 2);
//Green color
    }
}
```

2.2.3 Display Score

After the basketball is recognized, the score above the phone is +1.

```
    NSString *str1 = [NSString stringWithFormat:@"Scoring  %d",intScore];
    putText(inputFrame, [str1 UTF8String], cv::Point(250, 40),
        CV_FONT_HERSHEY_COMPLEX, 1, CV_RGB(255, 255, 255), 2);   //Green
color
```

2.2.4 Display running information

For the convenience of software debugging, there are three rectangular shapes in the lower left corner of the phone: hoopBack, hoopCurrent, and hoopDiff. There is also an m-value curve that shows real-time software operation status.

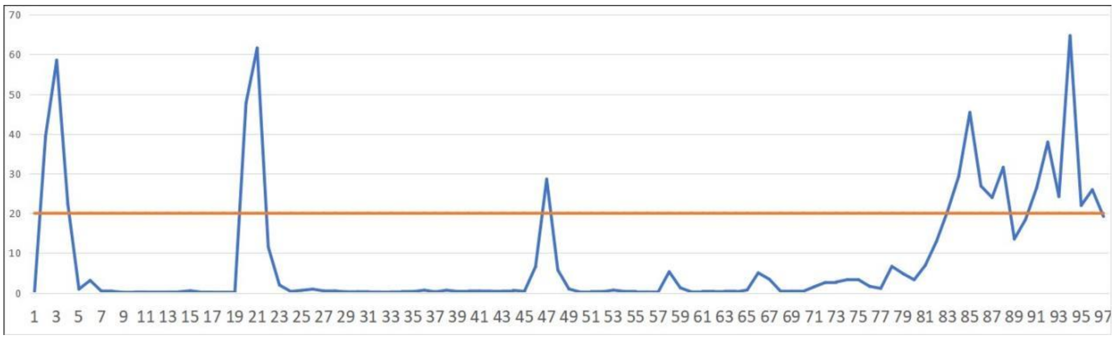
```
    hoopBack.copyTo(inputFrame(cv::Rect(40, 300, rcBallIn.width,
rcBallIn.height)));
    hoopCurrent.copyTo(inputFrame(cv::Rect(100, 300, rcBallIn.width,
rcBallIn.height)));
    hoopDiff.copyTo(inputFrame(cv::Rect(160, 300, rcBallIn.width,
rcBallIn.height)));

    int x = 0;
    int y = 300 - ((int)m_array[m_tail]) %300 ;
    cv::Point pt1 = cv::Point(x, y);    //for(int i=((m_tail+1)%200);
i==m_tail; i++) {
    for(int i=1; i<200; i++)  {
        x = i;
        y = 300 - ((int)m_array[(m_tail+i)%200]) %300 ;
        cv::Point pt2 = cv::Point(x, y);

        cv::line(inputFrame, pt1, pt2, CV_RGB(0, 255, 0), 1);
        pt1 = pt2;
    }
```

2.3 Debug and optimization

Problem: Score is duplicated of multiple frames for a single goal



Draw the m value of 100 frames as a graph line, and it can be seen that multiple frames will exceed the "judgment criteria" when scoring a goal, resulting in one goal being counted as several points.

Solution: Only score when the first m value exceeds the "judgment condition."

```
    if (m > 10)
    {
        //Determine if the m value has changed and avoid scoring multiple
times in one goal
        if (m_array[(m_tail-1 + 200)%200] <= 10) {
            intScore ++;
            putText(inputFrame, "Ball in!!!", cv::Point(40, 40),
                CV_FONT_HERSHEY_COMPLEX, 1, CV_RGB(255, 255, 255), 2);
//Green color
        }
    }
```

2.4 Deploy

Publishing an app in the iOS app store requires a paid account, and I did not apply for it. So now, the only way to publish this app is through connecting to the phone through a computer, which is relatively inefficient.

3. Reflections

After testing, this system can accurately identify the basket and whether or not to score goals. The optimization reduced the misjudgment rate from 60% to 100% to 17%

In the design, it is necessary to consider using a mobile phone as a platform for image recognition. The software is simple and can run independently. Basketball players use their mobile phones and tripods to efficiently track their shooting data, improving their basketball skills.

There are two main optimization directions for the next step:

(1) Identify the rebounds, penalty area, and three-point line in the field, which can distinguish between two- and three-point goals;

(2) Add autonomous training for the program to achieve machine learning.

References

[1]Instant OpenCV for iOS (English Edition), Alexander Shishkov、 Kirill Kornyakov

[2]iOS Application Development with OpenCV 3 (English Edition), Joseph Howse

[3]https://yq.aliyun.com/articles/64975 Alibaba Cloud AI recognizes basketball action videos

[4]https://blog.csdn.net/qq_38604769/article/details/79305879 Combining Opencv3.4 with VS2017 Environment