Basketball Shooting Recognition and Scoring Application

Gao Mingjun

Abstract:

Basketball players need daily training, but currently they can only manually record the score themselves, which is very inconvenient. This project is a shooting recognition software that can run on mobile phones. Basketball players carry their phones and tripods with them and place them on the side of the basketball court to automatically recognize and score shooting. After extensive experiments and optimization, it is possible to automatically and accurately identify whether a shot is scored or not. For basketball player who want to improve their abilities, it is very convenient to use data more efficiently to improve their basketball skills.

Keywords: Basketball Shooting Recognition, Scoring, iOS, Opencv

篮球投篮识别及计分系统

摘要：

篮球爱好者日常训练需要用到投篮命中率统计工具，但当前都只能自己手工记录，非常不方便。本项目一款可以运行在手机上的投篮识别软件，篮球爱好者随身携带手机、三脚架，放在篮球场馆侧方，自动识别投篮并计分。经过大量的实验和优化之后可以基本准确地自动识别投篮的进与不进。对于想要提高自己能力的篮球爱好者来说非常方便，更高效地使用数据来改善篮球爱好者们的篮球水平。

问题：

我们国家高级别篮球比赛会有专业分析人士分析比赛，但对于大量的篮球爱好者，现在来说却没有一款适用于日常训练用的软件系统。其中关键一项是统计球员们命中率，没有易用的工具，就无法方便地提升篮球爱好者的能力且给予球员们一个反馈。

现有的类似系统通常用到 MATLAB 等专业软件，而且还需要特定的硬件，很难普及。

思路：

识别投篮的进与不进需要有识别的对象，如篮筐和篮球。这款软件主要识别的是篮筐中的变化。如果有球进入已经识别到的区域，那么程序便会显示“ball in”的标示并统计总分。这款程序需要用到一个叫做 opencv 的开源计算机视觉库，先将视频导入，然后使用 opencv 画出需要识别的区域，观察该区域的变化，识别篮球是否进筐。

在设计中要考虑用手机作为平台进行图像识别，软件要简单，不能太耗资源。运行时手机要能独立工作。

。

一、系统设计

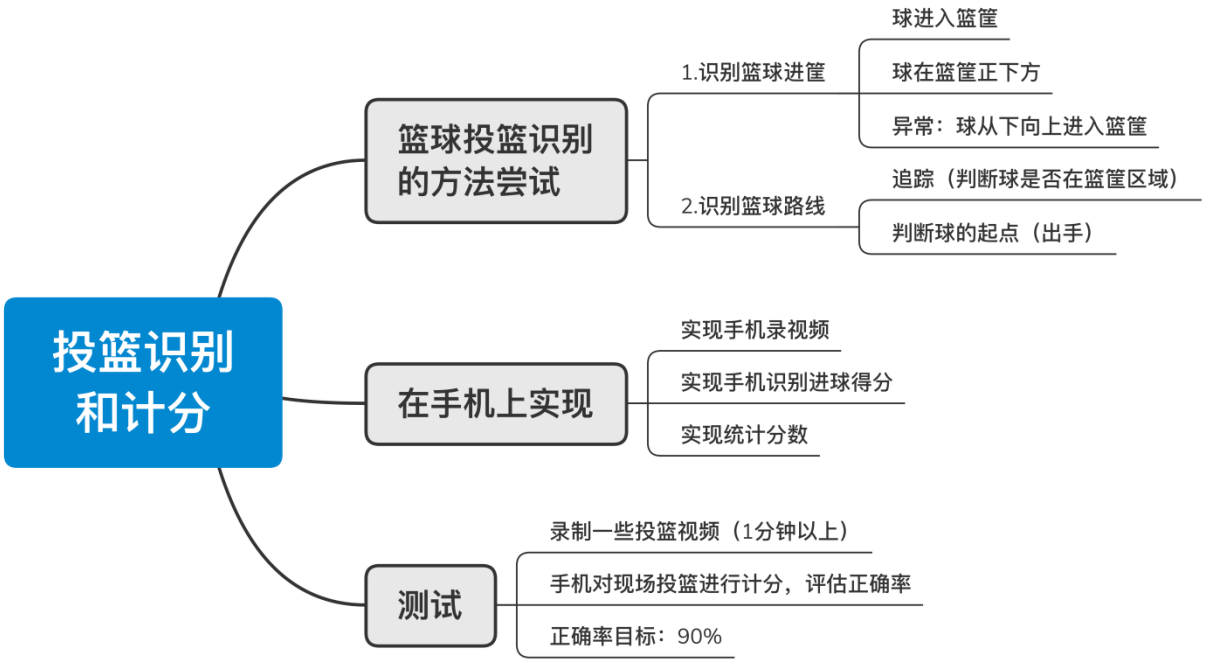

图 1：系统设计的思维导图

图 2：系统分解

篮球爱好者随身携带手机、三脚架，放在篮球场馆侧方，向着篮筐方向放置

1、三脚架用于避免视频抖动。

2、手机（摄像头+屏幕+视频识别软件）组成视频识别和计分功能

二、篮球进筐识别的程序设计

思路：1）标记篮筐的矩形区域 rcHoop；2）备份视频第一帧的篮筐区域 hoopBack；3）读取当前帧的篮筐区域 hoopCurrent；4）比较备份和当前区域，得出 hoopDiff，算出均值 m；5）用 m 值来判断是否用篮球进入篮筐区域。

表 1：识别篮球是否进入篮筐的程序

```
#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);  //腐蚀操作
        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()等都是 **opencv 开源库中的函数，用**来备份和当前区域图像的差异。

实际效果如下：

- 进球后可以显示“ball in！！！”字样。
- 球在球筐上方和下方，都没有出现“ball in!!!”字样。

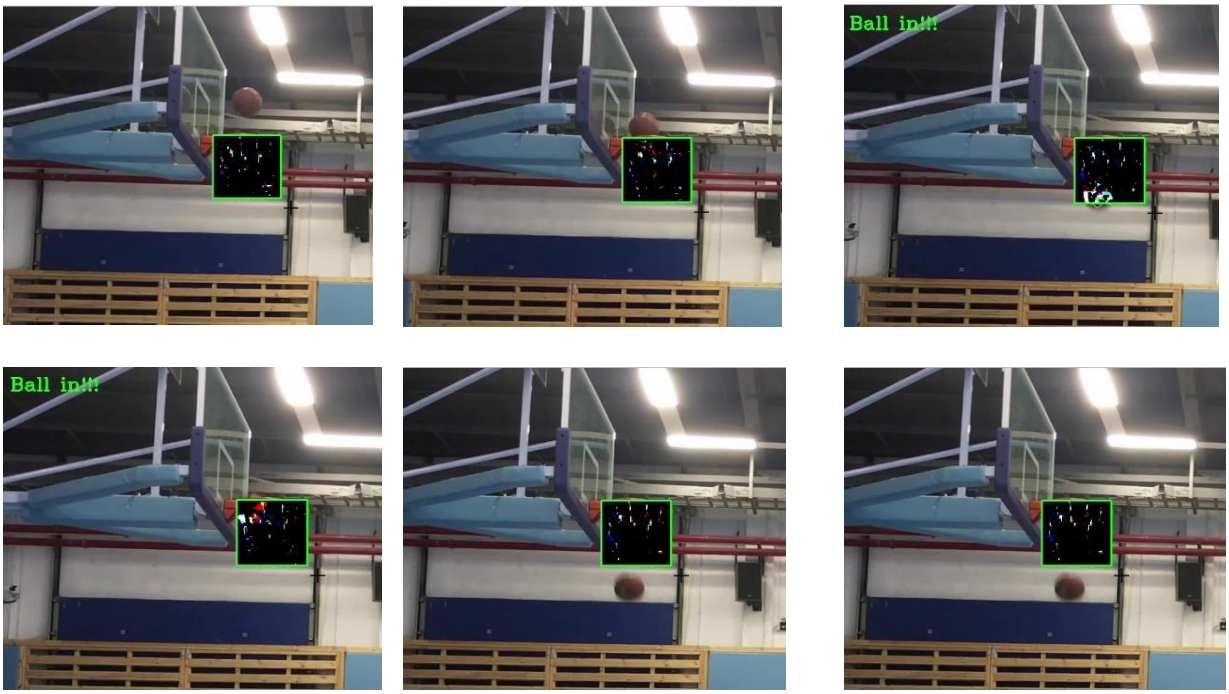

图 3：篮球进筐的分解图

测试及优化过程：

1、问题 1： 拍摄时出现抖动，会造成误判

表 2：测试视频文件“201905.mp4”的记录表

	测试 1
测试文件	201905.mp4
m 值判断条件	m>20
程序判别进球数	45
真实进球数	5
错误率	+800%

发现当视频拍摄时非常不稳，“ball in”字样显示了 45 次，而真正进球却只有 5 次，于是发现由视频抖动所产生的误判有 37 次。

优化方案：使用三脚架拍摄会提升识别的正确率。后续测试中除去抖动所产生的误判后，识别率大大提升了，误判率降低到 60%~100%。

2、问题 2 ：一次进球多帧重复计分

把 100 帧的 m 值画成图线，可以看出进球时会有多帧超出“判断条件”，就会出现一次进球计算了好几分

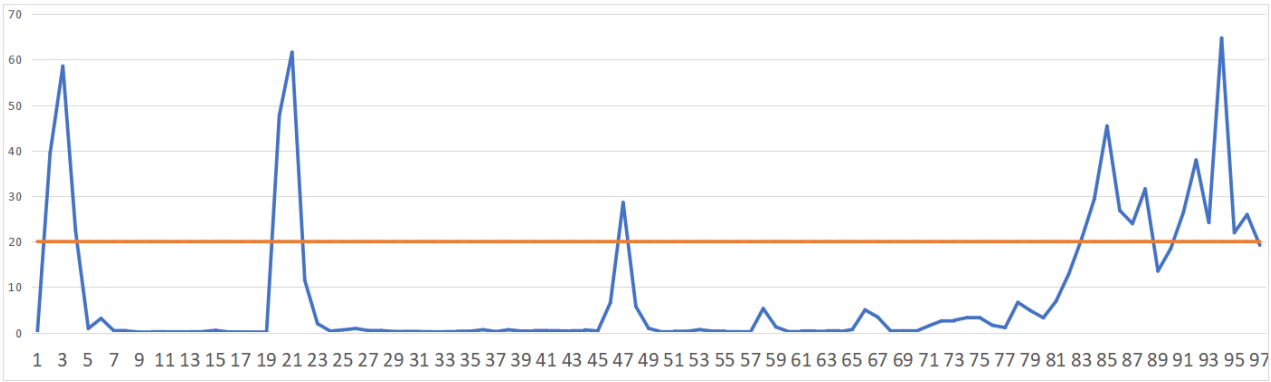

图 4：第 1 帧到第 100 帧的 m 值图线（判断条件为 20）

优化方案：只在第一次 m 值超出“判断条件”时，才计分

表 3：判断 m 值是否发生过变化的程序

```
int main(int argc, char** argv)
{
	cv::VideoCapture cap1("C:\\2020-1.mov");
	cv::Mat frame;

	const int MAXNUM_M = 1000;
	double m_array[MAXNUM_M];   for (int i = 1; i < MAXNUM_M; i++) { m_array[i] = 0; }
	int m_tail = 0;
	int ballin_count = 0;

	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);
		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 >= 30)           {
			//判断m值是否发生过变化，避免一次进球多次计分
			if (m_array[(m_tail-1 + MAXNUM_M)%MAXNUM_M] < 30) {
				ballin_count++;
			}
		}
		putText(frame, "shots made: " + std::to_string(ballin_count), Point(800, 40),
			CV_FONT_HERSHEY_COMPLEX, 1, CV_RGB(0, 255, 0), 2);

		m_array[m_tail] = m;     //保存在数组中
		m_tail = (m_tail + 1) % MAXNUM_M;
		cv::imshow("read", frame);
```

```
                cv::waitKey(30);
                cap1.read(frame);
        }
} // main
```

为了要判断 m 值的变化，需要把历史 m 值都保存到数组 m_array 中。

3、问题 3：篮球从下往上运动发生误判

表 4：测试视频文件“2020-*.mov”的记录表

	测试 2-1	测试 2-2	测试 2-3	测试 2-4	总计
测试文件	2020-1.mov	2020-2.mov	2020-3.mov	2020-4.mov	
m 值判断条件	m>30	m>30	m>30	m>30	
程序判别进球数	13	31	29	31	104
真实进球数	12	24	16	18	70
误判率	8%	+29%	+81%	+72%	+49%

总共显示了 104 次“*ball in*”，错了 34 次，错误率为大约 49%。

而在这 49%中，大多数错误原因为球从识别区域从下往上运动的时候被误判为进球。

优化方案：在篮筐上方再增加一个识别区域，也计算出一个代表备份和当前帧的差异的 m2 值。

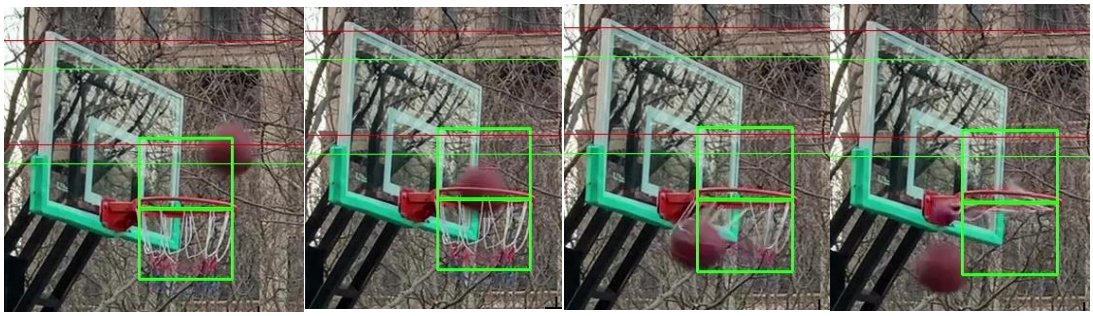

图 5：增加了篮筐上方识别区的篮球进筐分解图

正常的球应该先穿过篮筐上方的 m2 区域，然后才会到达篮筐的 m 区域。

表 5：增加 m2 值判断条件的程序

```
int main(int argc, char** argv)
{
        cv::VideoCapture cap1("C:\\2020-1.mov");
        cv::Mat frame;

        const int MAXNUM_M = 1000;
        double m_array[MAXNUM_M];   for (int i = 1; i < MAXNUM_M; i++) { m_array[i] = 0; }
        double m2_array[MAXNUM_M];   for (int i = 1; i < MAXNUM_M; i++) { m2_array[i] = 0; }
        int m_tail = 0;
        int ballin_count = 0;

        cap1.read(frame);
        Rect rcHoop = cv::selectROI("read", frame, true);
        cv::Mat hoopBack = frame(rcHoop).clone();   //backup
        cv::Mat hoopDiff;

        Rect rcM2 = cv::Rect(rcHoop.x, rcHoop.y - rcHoop.height, rcHoop.width, rcHoop.height);
        cv::Mat m2Back = frame(rcM2).clone();       //backup of M2
        cv::Mat m2Diff;
```

```
        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);
                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

                //对篮筐上方区域计算m2值
                cv::Mat m2Current = frame(rcM2).clone();   //current
                cv::absdiff(m2Back, m2Current, m2Diff);    //difference
                cv::erode(m2Diff, m2Diff, element);
                threshold(m2Diff, m2Diff, 5, 255, CV_THRESH_BINARY);
                cvtColor(m2Diff, gray, CV_RGB2GRAY);
                double m2 = mean(gray)[0];                   //m2 value of difference
                m2_array[m_tail] = m2;    //保存在数组中

                cout << m << endl;
                if (m >= 30)            {
                        //判断m值是否发生过变化，避免一次进球多次计分
                        if (m_array[(m_tail-1 + MAXNUM_M)%MAXNUM_M] < 30)
                        {
                                //判断M2是否先有变化
                                if (m2_array[(m_tail - 1 + MAXNUM_M) % MAXNUM_M] > 20)
                                {
                                        ballin_count++;
                                }
                        }
                }
                putText(frame, "shots made: " + std::to_string(ballin_count), Point(800, 40),
                        CV_FONT_HERSHEY_COMPLEX, 1, CV_RGB(0, 255, 0), 2);

                m_array[m_tail] = m;    //保存在数组中
                m_tail = (m_tail + 1) % MAXNUM_M;
                cv::imshow("read", frame);
                cv::waitKey(30);
                cap1.read(frame);
        }
} // main
```

篮筐上方的 m2 区域对应有备份 m2Back、当前帧 m2Current，比较后得出 m2Diff，计算出 m2 值。

表 6：优化后重新测试视频文件“2020-*.mov”的记录表

	测试 3-1	测试 3-2	测试 3-3	测试 3-4	总计
测试文件	2020-1.mov	2020-2.mov	2020-3.mov	2020-4.mov	
m 判断条件	m>30	m>30	m>30	m>30	
m2 判断条件	m2>20	m2>20	m2>20	m2>20	
程序判别进球数	12	26	19	25	82
真实进球数	12	24	16	18	70
误判率	**0%**	**+8%**	**+19%**	**+39%**	**+17%**

除去球从识别区域从下往上运动的情况后，识别率大大提升了。

三、跟踪篮球路线的程序设计

opencv 开源视觉库中提供了物体跟踪的样例程序，可用来对投篮进行识别。

表 7：跟踪篮球路线的程序

```
#include <opencv2/opencv.hpp>
#include <opencv2/video.hpp>
#include <opencv2/tracking.hpp>
#include <opencv2/tracking/tracker.hpp>


#include <iostream>
#include <cstring>
using namespace std;
using namespace cv;

int main(int argc, char** argv) {
    // declares all required variables
    Rect2d roi;
    Mat frame;
    // create a tracker object
    Ptr<TrackerKCF> tracker = TrackerKCF::create();
    //Ptr<Tracker> tracker = Tracker::create("KCF");
    // set input video
    //  std::string video = argv[1];
    VideoCapture cap("C:\\201905.MP4");
    // get bounding box
    cap >> frame;
    roi = selectROI("tracker", frame);
    //quit if ROI was not selected
    if (roi.width == 0 || roi.height == 0)
        return 0;
    // initialize the tracker
    tracker->init(frame, roi);
    // perform the tracking process
    printf("Start the tracking process, press ESC to quit.\n");
    for (;; ) {
        // get frame from the video
        cap >> frame;
        // stop the program if no more images
        if (frame.rows == 0 || frame.cols == 0)
            break;
        // update the tracking result
        tracker->update(frame, roi);
        // draw the tracked object
        rectangle(frame, roi, Scalar(255, 0, 0), 2, 1);
        // show image with the tracked object
        imshow("tracker", frame);
        //quit on ESC button
        if (waitKey(1) == 27)break;
    }
    return 0;
}
```

效果测评：

由图可见框刚开始还是跟踪球的，而当球和衣服颜色混在了一起，框就不再跟踪的了球了，但是在尝试跟踪球的实验中却发现无法跟踪球。衣服会让程序误认为是球。

图 6：篮球路线跟踪的分解图

所以，通用的物体跟踪方法不能用于投篮的识别。

四、在手机上实现“投篮识别”

图 7：手机应用的功能分布

1、 按钮

Start 按钮可以打开手机摄像头，按照 30 帧/秒获取视频，显示到手机屏幕上。每按一次 Start 按钮还可以将篮筐识别区域保存到 hoopBack 中。

Stop 按钮按一下可以冻结屏幕图像，再按一下恢复视频显示

2、 篮筐识别区域

该区域的视频图像对应着 hoopCurrent，每帧都会计算出 m 值和 m2 值。然后判断篮球是否进筐

3、 计分显示

篮球进筐被识别后，手机上方的计分就+1。

4、 软件运行信息

为了方便软件调试，在手机的左下方有：hoopBack、hoopCurrent、hoopDiff 三个矩形图形。还有一条 m 值曲线，可以看到实时的软件运行状况。

五、总结

这款系统在试验过后可以基本准确地识别篮筐以及进球与否。

1）第一次优化，误判率从 800%降低到 60%~100%

2）第二次优化，误判率从 60%~100%降低到 49%

3）第三次优化，误判率从 49%降低到 17%

在设计中要考虑用手机作为平台进行图像识别，软件简单，可独立运行。篮球爱好者利用随身携带的手机配合三脚架，高效地使用统计自己投篮数据，从而改善篮球爱好者们的篮球水平。

下一步的优化方向主要有两个：1）对场地中的篮板、罚球区、三分线进行识别，可以区分两分和三分的进球；2）加上给程序的自主训练，实现机器学习。

参考文献：

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

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

https://yq.aliyun.com/articles/64975 阿里云人工智能识别篮球动作视频

https://blog.csdn.net/qq_38604769/article/details/79305879 opencv3.4 与 vs2017 环境搭配