NaoRobotClub比赛准备

笔记本：	NaoRobotClub		
创建时间：	2019/11/18 16:28	更新时间：	2019/11/30 19:40
作者：	1349619363@qq.com		
URL：	about:blank		

2019年11月19日：

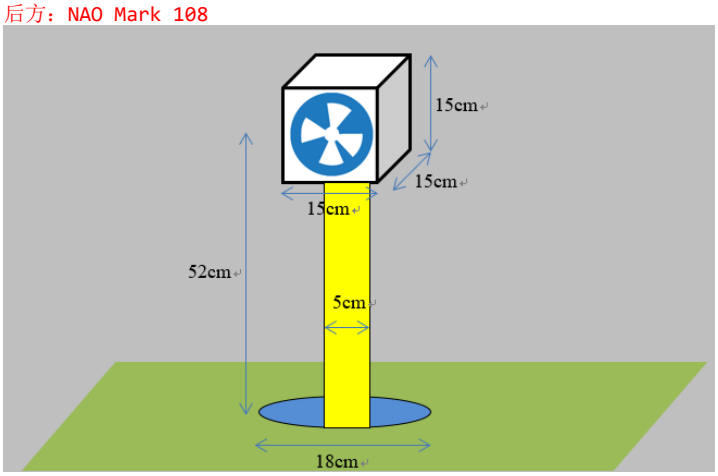
机器人Golf比赛规则：

- 球的选择：高尔夫标准球(半径为5cm)
- 球杆的选择：儿童玩具球杆,高度40-50cm
- 球洞：

球洞直径为 18cm，深 5cm。球洞内部为蓝色。球洞中央竖置一个杆，杆体为黄色（有利于远距离识别杆的位置），直径为 5cm。杆顶为一个边长为 15cm 的正方体 NAO Mark 标记，便于参赛队搜索和定位球洞。正方体是四面都贴有不同的 NAO Mark 标记。每个球洞都有一个相同的正方体 NAO Mark：

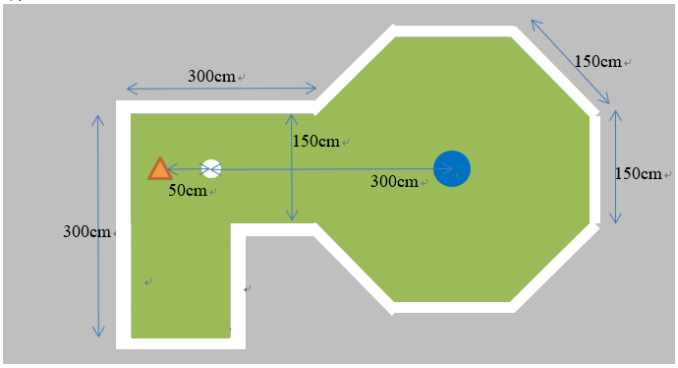
- landmark：

前方：（从起点位置看）：NAO Mark 64
右侧：（从起点位置看，右侧表面）：NAO Mark 107
左侧：（从起点位置看，左侧表面）：NAO Mark 112
后方：NAO Mark 108

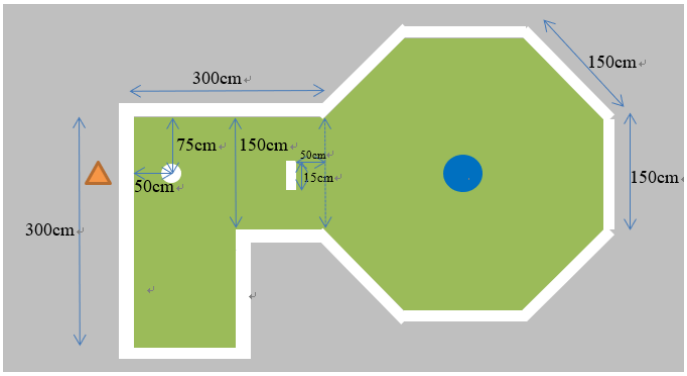


- 场地：

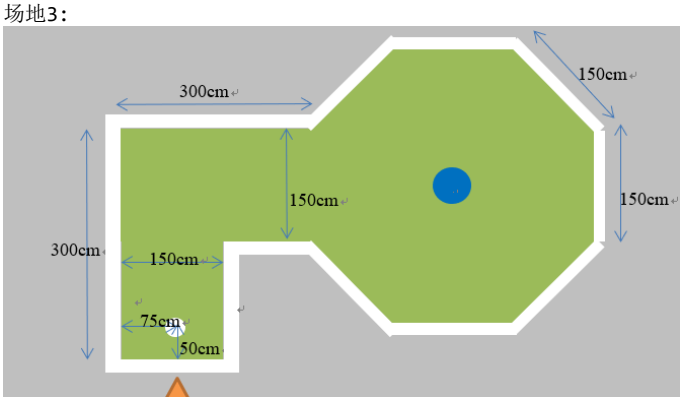
场地1：



场地2：



场地3:



- 比赛规则:

起始分:每个队进场比赛默认起始分为50分
时间:每个队比赛总用时20分钟，参赛队员和机器人进入场地比赛即开始计时。
机器人放置:开赛前，球会置于起点位置，参赛队场中队员将机器人放置于机器人起始位置进行开球。可以用语音或触摸指令来控制机器人开始击球，并完成整个进洞过程，整个过程必须是机器人自主完成的,比赛时所有程序和计算必须运行在机器人本地，不能通过无线与其他程序通信。开始的时间由裁判与场中队员确认后给出信号。
击球:机器人禁止用除球杆外的其他部位击球。如发生，裁判将给与1分罚分。
出界:击球出界时，裁判将球放置到边界上，让机器人继续击球，并给与1分罚分。 放弃某个球洞:机器人在比赛时，参赛队可以放弃当前球洞，前往下一个球洞继续完成比赛，同时由裁判和场中队员确认球与球洞距离。
杆数:如机器人无法在10杆内完成比赛，则比赛结束。

- 评分:

1号球洞，进球得15分;未进球，距洞口0-10cm得3分，距离10-20cm 得1分
2号球洞，进球得25分;未进球，距洞口 0-10cm得5分，距离10-20cm 得3 分
3 号球洞，进球得35分，未进球，距洞口0-10cm得5分，距离10-20cm 得 3 分
总分数为起始分加得分减去罚分，总分数多的队伍获胜；总得分一样进洞数多的队伍获胜；总得分、进洞数相同的的情况下，累计三个洞的高尔夫球与球洞距离更短的队伍获胜；总得分、进洞数、球洞距离均一样，用时少的队伍获胜。
机器人摔倒:如机器人在比赛中途摔倒，裁判可进场重新将球杆放置在机器人手中。(机器人若无法自行站起，裁判可以将机器人恢复站立)
机器人硬件故障:在比赛过程中若出现机器人硬件故障，经裁判认可后，可有一次更换备用机器人的机会继续比赛.

机器人代码解释:

- ConfigureNao.py:

定义的类:ConfigureNao(Object)

定义的类属性(__init__):

IP(需要配置),PORT = 9559,cameraProxy(ALVideoDevice),motionProxy(ALMotion),postureProxy(ALRobotPosture),tts(ALTextToSpeech),memoryProxy(ALMemory),landMarkProxy(ALLandMarkDetection)

模块的作用:

ALVideoDevice:file:///C:/Program%20Files/Robot/share/doc/naoqi/vision/alvideodevice.html?highlight=alvideodevice
ALMotion:file:///C:/Program%20Files/Robot/share/doc/naoqi/motion/almotion.html
ALRobotPosture:file:///C:/Program%20Files/Robot/share/doc/naoqi/motion/alrobotposture.html?highlight=alrobotposture
ALTextToSpeech:file:///C:/Program%20Files/Robot/share/doc/naoqi/audio/altexttospeech.html?highlight=altexttospeech
ALMemory: file:///C:/Program%20Files/Robot/share/doc/naoqi/core/almemory.html
ALLandMarkDetection:file:///C:/Program%20Files/Robot/share/doc/naoqi/vision/allandmarkdetection.html?highlight=allandmarkdetection

---

- NewAction.py:

  定义的类包括：NewMotionBasis(ConfigureNao)

  定义类的属性：

Name = ['RShoulderPitch', 'RShoulderRoll', 'RElbowRoll', 'RWristYaw', 'RElbowYaw'] 其中name可以包括为连接关节两个部分的舵机的名称(joints的概念) ,查阅表中可以发现(角度右为正,左为负)

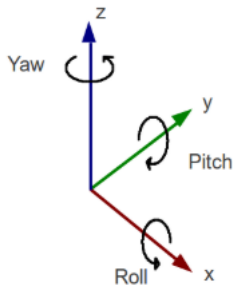
RShoulderPitch: Range (degrees): -82.7 to 127.2   Range (radian):-1.44338729 to 2.22005880(右关节前后)

RShoulderRoll(?????): ...=-120.0 to 120.0   ...: -2.09439510 to 2.09439510(右关节左右动旋转)

RElbowRoll: ...=-120.0 to 120.0   ...: -2.09439510 to 2.09439510(右肘旋转动)

RWristYaw: ...= -25.0 to 25.0   ...: -0.43633231 to 0.43633231(右手关节左右动)

RElbowYaw: ...= 0.0 to 90.0   ...: 0.00000000 to 1.57079632(右肘左右动)



理解坐标系建立

file:///C:/Program%20Files/Robot/share/doc/family/romeo/joints_romeo.html?highlight=joints

file:///C:/Program%20Files/Robot/share/doc/family/robots/joints_robot_v33.html?highlight=joints

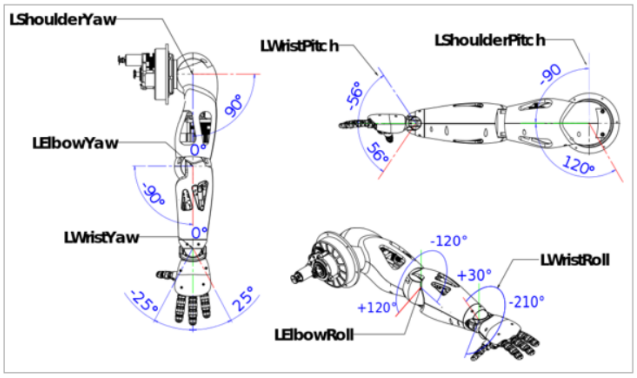
Head

Joint name	Motion	Range (degrees)	Range (radian)
NeckYaw	Neck joint (Z)	-90.0 to 90.0	-1.57079632 to 1.57079632
NeckPitch	Neck joint (Y)	-20.0 to 40.0	-0.34906585 to 0.69813170
HeadPitch	Head joint (Y)	-20.0 to 16.0	-0.34906585 to 0.27925268
HeadRoll	Head joint (X)	-20.0 to 20.0	-0.34906585 to 0.34906585

Arms

Left

Joint name	Motion	Range (degrees)	Range (radian)
LShoulderPitch	Left Shoulder joint (Y)	-82.7 to 127.2	-1.44338729 to 2.22005880
LShoulderYaw	Left Shoulder joint (Z)	-24.7 to 65.3	-0.43109632 to 1.13970000
LElbowRoll	Left Shoulder joint (X)	-120.0 to 120.0	-2.09439510 to 2.09439510
LElbowYaw	Left Elbow joint (Z)	-90.0 to 0.0	-1.57079633 to 0.00000000
LWristRoll	Left Elbow joint (X)	-210.0 to 30.0	-3.66519143 to 0.52359878
LWristYaw	Left Wrist joint (Z)	-25.0 to 25.0	-0.43633231 to 0.43633231
LWristPitch	Left Wrist joint (Y)	-56.0 to 56.0	-0.97738438 to 0.97738438



Right

Joint name	Motion	Range (degrees)	Range (radian)
RShoulderPitch	Right Shoulder joint (Y)	-82.7 to 127.2	-1.44338729 to 2.22005880
RShoulderYaw	Right Shoulder joint (Z)	-65.3 to 24.7	-1.13970000 to 0.43109632
RElbowRoll	Right Shoulder joint (X)	-120.0 to 120.0	-2.09439510 to 2.09439510
RElbowYaw	Right Elbow joint (Z)	0.0 to 90.0	0.00000000 to 1.57079632
RWristRoll	Right Elbow joint (X)	-30.0 to 210.0	-0.52359878 to 3.66519143
RWristYaw	Right Wrist joint (Z)	-25.0 to 25.0	-0.43633231 to 0.43633231
RWristPitch	Right Wrist joint (Y)	-56.0 to 56.0	-0.97738438 to 0.97738438

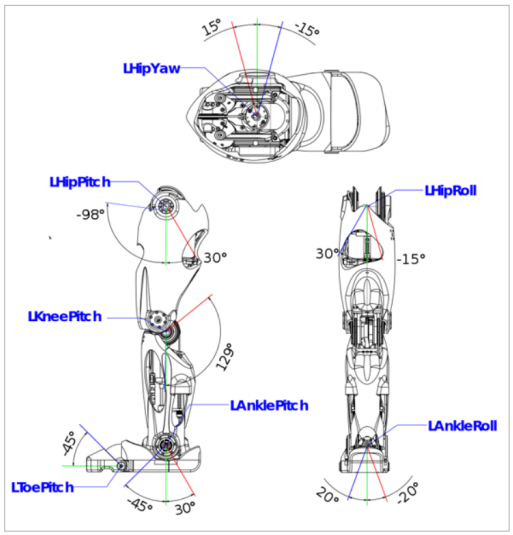
Trunk

Joint name	Motion	Range (degrees)	Range (radian)
TrunkYaw	Trunk Yaw joint (Z)	-45.0 to 45.0	-0.78539816 to 0.78539816

Legs ¶

Left

Joint name	Motion	Range (degrees)	Range (radian)
LHipYaw	Left Hip joint (Z)	-15.0 to 15.0	-0.26179939 to 0.26179939
LHipRoll	Left Hip joint (X)	-15.0 to 30.0	-0.26179939 to 0.52359878
LHipPitch	Left Hip joint (Y)	-98.0 to 23.0	-1.71042267 to 0.40142572
LKneePitch	Left Knee joint (Y)	0.0 to 115.0	0.00000000 to 2.00712863
LAnklePitch	Left Ankle joint (Y)	-30.0 to 45.0	-0.52359878 to 0.78539816
LAnkleRoll	Left Ankle joint (X)	-20.0 to 20.0	-0.34906585 to 0.34906585
LToePitch	Left Ankle joint (Y)	-45.0 to 0.0	-0.78539816 to 0.00000000



Right

Joint name	Motion	Range (degrees)	Range (radian)
RHipYaw	Right Hip joint (Z)	-15.0 to 15.0	-0.26179939 to 0.26179939
RHipRoll	Right Hip joint (X)	-30.0 to 15.0	-0.52359878 to 0.26179939
RHipPitch	Right Hip joint (Y)	-98.0 to 23.0	-1.71042267 to 0.40142572
RKneePitch	Right Knee joint (Y)	0.0 to 115.0	0.00000000 to 2.00712863
RAnklePitch	Right Ankle joint (Y)	-30.0 to 45.0	-0.52359878 to 0.78539816
RAnkleRoll	Right Ankle joint (X)	-20.0 to 20.0	-0.34906585 to 0.34906585
RToePitch	Right Ankle joint (Y)	-45.0 to 0.0	-0.78539816 to 0.00000000

Eyes

Joint name	Motion	Range (degrees)	Range (radian)
LEyeYaw	Left Eye joint (Z)	-30 to 30	-0.52359877 to 0.52359877
LEyePitch	Left Eye joint (Y)	-20 to 20	-0.34906585 to 0.34906585
REyeYaw	Right Eye joint (Z)	-30 to 30	-0.52359877 to 0.52359877
REyePitch	Right Eye joint (Y)	-20 to 20	-0.34906585 to 0.34906585

```
Alpha = -math.pi / 2    初始角度为-90度
```

nao的行走:nao的行走已经设计好,但可以修改一些参数来实现在自己的目的:

步态参数

名称		默认	最低要求	最大	可设定
MaxStepX	沿X的最大正向平移（米）	0.040	0.001	0.080 [3]	是
MinStepX	沿X的最大向后平移（米）	-0.040			没有
MaxStepY	沿Y的绝对最大横向平移（米）	0.140	0.101	0.160	是
MaxStepTheta	围绕Z的绝对最大旋转（弧度）	0.349	0.001	0.524	是
MaxStepFrequency	最大步进频率（标准化，无单位）	1.	0.	1.	是
MinStepPeriod	最小步长（秒）	0.42			没有
MaxStepPeriod	最大步长（秒）	0.6			没有
步高	沿Z的高峰脚高程（米）	0.020	0.005	0.035	是
躯干	围绕X的最大躯干旋转（弧度）	0.000	-0.122	0.122	是
躯干	躯干绕Y旋转的峰值（弧度）	0.000	-0.122	0.122	是
脚分离	沿Y改变双脚之间的距离（米）	0.1			没有
MinFoot分离	两脚沿Y的最小距离（米）	0.088			没有

[3]	对于StepX，我们建议使用0.060米，以提高稳定性。仅在平坦的硬地板上行走时，最好使用0.080米。

2019年11月20日：

代码中的设置:MaxStepX = 0.04

```
MaxStepY = 0.14

MaxStepTheta = 0.4

MaxStepFrequency = 0.6

StepHeight = 0.02

TorsowX = 0.0

TorsowY = 0.0

MoveConfig = [["MaxStepX", self.MaxStepX],
 ["MaxStepY", self.MaxStepY],
 ["MaxStepTheta", self.MaxStepTheta],
 ["MaxStepFrequency", self.MaxStepFrequency],
 ["StepHeight", self.StepHeight],
 ["TorsoWx", self.TorsowX],
 ["TorsoWy", self.TorsowY]]
```

file:///C:/Program%20Files/Robot/share/doc/naoqi/motion/control-walk.html?highlight=maxstepx#id5

naoqi的api:getdata函数获得存储在内存的有效值,函数中memoryProxy.getData("HandRightRightTouched")获取的右手边上传感器HandRightRightTouched为ALTouch的一个回调函数(在一定情况下自动触发)返回值转化成原始数据类型(???),通常1为触发,其他为失败.

file:///C:/Program%20Files/Robot/share/doc/naoqi/core/almemory-api.html?highlight=getdata#ALMemoryProxy::getData__ssCR

1. naoqi的api:almath.TO_RAD 可以把角度改成弧度制表示

2. naoqi的api:angleInterpolationWithSpeed(name,angle,speed)功能:把指定的舵机转化为指定角度,并且可以设置速度

具体的name列表:

file:///C:/Program%20Files/Robot/share/doc/family/pepper_technical/bodyparts_pep.html#effector-pep

3. naoqi的api:

angleInterpolation(name/namelist,targetlist,timelist,isabsolute)其中name可以为一个列表表示一组舵机或者关节,后面target为目标角度,timelist为定时器,isabsolte为设置绝对角度和相对角度

4. 对于打球方向设定:对比类属性的Alphe变化小于0的时候从右边往左边打球,大于0的时候从左边往右边打球,从对比两种不同角度可见,只有对于手腕的关节设置正负.

- AdjustPosition函数:

总体上看来该函数主要功能在于调整自己的位置,需要传递的参数为turndata.

1.naoqi的api: setMoveArmsEnabled(leftarm,rightarm),其中两个参数为布尔型变量,可以设置在运动过程中左右手臂是否可动

2.naoqi的api:moveTo(x,y,thera,config)其中x为前进角度,y为左右角度,thera为角度,config为设置(类变量中,例如设置步幅走路姿势之类)其中参数范围(z为幅度制):

x - Distance along the X axis in meters

y - Distance along the Y axis in meters.

theta - Rotation around the Z axis in radians [-3.1415 to 3.1415].

moveConfig - An ALValue with the custom move configuration. For further details

---

- VisualMoudle(视觉模块):

定义的类:

VisualBasis(ConfigureNao),BallDetect(VisualBasis),StickDetect(VisualBasis),LandMarkDetect(ConfigureNao)

1.VisualBasis(ConfigureNao):

所有识别的基类.

类变量:

cameraId设置使用的摄像头其中初始设置为kBottomCamera,即为1位嘴上的摄像头,0为头上的摄像头kTopCamera,2为眼睛上的3D摄像头kDepthCamera
cameraName使用的相机名称

resolution照片的分辨率

colorSpace 设置采用色彩空间,初始采用RGB模式
fps帧率
frameHeight图像的高度
frameWidth图像的宽度

frameChannels 通道的意思

frameArray初始为空

cameraPitchRange相机视野内的高度 47.64 / 180 * np.pi

cameraYawRange相机内的视野宽度 60.97 / 180 * np.pi

用到的函数:

1. naoqi的Api函数:setActiveCamera(ID)启动当前的相机,参数设置用到相机的编号

2. naoqi的Api函数:subscribe(client,ID,resolution,colorspace,fps)其中client为订阅的名称(可随意???),ID照相机ID(可忽略?),照片的分辨率,色彩空间,帧率.该函数主要功能为当client连接上照相机他会把照相机采集的数据写入内存,同时返回该数据在内存中名称,可以通过函数取出内存.
3. naoqi的Api函数:getImageRemote(sub)里面的参数为上面的subcribe返回的handle,该函数作用是返回最新的照片从图像检索源中,并可以通过网络传送,返回的是照片的规格.
其中照片的规格(frame),是一个容器(number of layers???):

Images

Images container is an array of image containers as follow:

- [0] First Image, an array containing:
  - [0]: width.
  - [1]: height.
  - [2]: number of layers.
  - [3]: ColorSpace.
  - [4]: time stamp from qi::Clock (seconds).
  - [5]: time stamp from qi::Clock (microseconds).
  - [6]: binary array of size height * width * nblayers containing image data.
  - [7]: camera ID (kTop=0, kBottom=1).
  - [8]: left angle (radian).
  - [9]: topAngle (radian).
  - [10]: rightAngle (radian).
  - [11]: bottomAngle (radian).
- [1] Second Image.
  - [0]: width.
  - [1]: height.
  - [2]: number of layers.
  - [3]: ColorSpace.
  - [4]: time stamp from qi::Clock (seconds).
  - [5]: time stamp from qi::Clock (microseconds).
  - [6]: binary array of size height * width * nblayers containing image data.
  - [7]: camera ID (kTop=0, kBottom=1).
  - [8]: left angle (radian).
  - [9]: topAngle (radian).
  - [10]: rightAngle (radian).
  - [11]: bottomAngle (radian).

3. numpy 的函数:frombuffer 用流的形式把缓存读入,变成np数组,其中需要设置参数(*buffer, dtype=float, count=-1, offset=0*)
其中起一个为缓存,第二个为数据类型,第三个为读入数据数量,offset从哪儿开始读(偏移量)

eg:

```
>>> s = 'hello world'
>>> np.frombuffer(s, dtype='S1', count=5, offset=6)
array(['w', 'o', 'r', 'l', 'd'],
      dtype='|S1')
```

4. opencv的函数: imshow(name,jpg)第一个窗口的名称,第二个为照片

2.BallDetect(VisualBasis):(球的识别)

新增的类变量:

ballData = {"centerX": 0, "centerY": 0, "radius": 0}获取球数据的字典

ballPosition = {"disX": 0, "disY": 0, "angle": 0} # *球的位置 获取球未知的字典*

ballRadius = 0.025 球的半径

speAngle = 0 !!

剩余的类变量和上面类似

```
self.alpha = -math.pi / 2
self.maxstepx = 0.04
self.maxstepy = 0.14
self.maxsteptheta = 0.4
self.maxstepfrequency = 0.6
self.maxstepfrequency2 = 0.3    # 步频
self.stepheight = 0.02
self.torsowx = 0.0
self.torsowy = 0.0
self.moveConfig = [["MaxStepX", self.maxstepx],
                   ["MaxStepY", self.maxstepy],
                   ["MaxStepTheta", self.maxsteptheta],
                   ["MaxStepFrequency", self.maxstepfrequency],
                   ["StepHeight", self.stepheight],
                   ["TorsoWx", self.torsowx],
                   ["TorsoWy", self.torsowy]]
self.moveConfig2 = [["MaxStepX", self.maxstepx],
                    ["MaxStepY", self.maxstepy],
                    ["MaxStepTheta", self.maxsteptheta],
                    ["MaxStepFrequency", self.maxstepfrequency2],
                    ["StepHeight", self.stepheight],
                    ["TorsoWx", self.torsowx],
                    ["TorsoWy", self.torsowy]]   # 主要频率不同
```

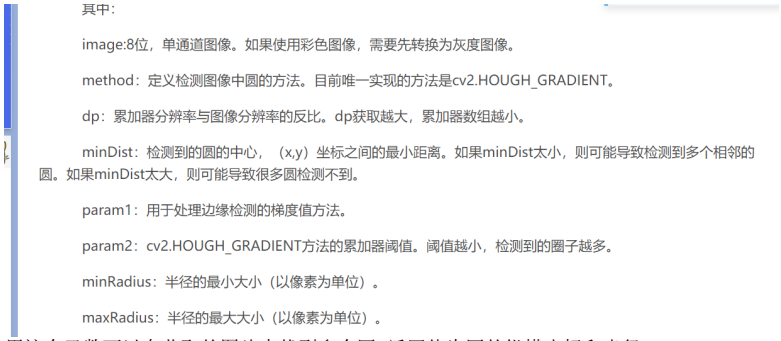
两个移动区别在于步频不一样

1.图像的模糊化函数__getChannelAndBlur:

函数中定义的类变量channel为通道的意思,具体来讲一张图片如果是rgb格式,他的特点有长和宽,rgb是红绿蓝三种颜色重叠,对应一个矩阵三个值,每一个作为一个通道.然后提取器特征值.这个函数弱化了绿色和蓝色,同时增强红色.通过高斯滤波来去除高斯噪点,设置矩阵为(9,9),方差设置为1.5,来进行模糊化处理.

理解高斯滤波 https://www.jianshu.com/p/73e6ccbd8f3f

2. __findCircles找图片中的圆形,cv2.HOUGH_GRADIENT表示使用霍夫圆环检测对于cv2.HoughCircles(image, method, dp, minDist, circles=None, param1=None, param2=None, minRadius=None, maxRadius=None)
https://blog.csdn.net/github_39611196/article/details/81128380



其中:

image:8位，单通道图像。如果使用彩色图像，需要先转换为灰度图像。

method：定义检测图像中圆的方法。目前唯一实现的方法是cv2.HOUGH_GRADIENT。

dp：累加器分辨率与图像分辨率的反比。dp获取越大，累加器数组越小。

minDist：检测到的圆的中心，（x,y）坐标之间的最小距离。如果minDist太小，则可能导致检测到多个相邻的圆。如果minDist太大，则可能导致很多圆检测不到。

param1：用于处理边缘检测的梯度值方法。

param2：cv2.HOUGH_GRADIENT方法的累加器阈值。阈值越小，检测到的圈子越多。

minRadius：半径的最小大小（以像素为单位）。

maxRadius：半径的最大大小（以像素为单位）。

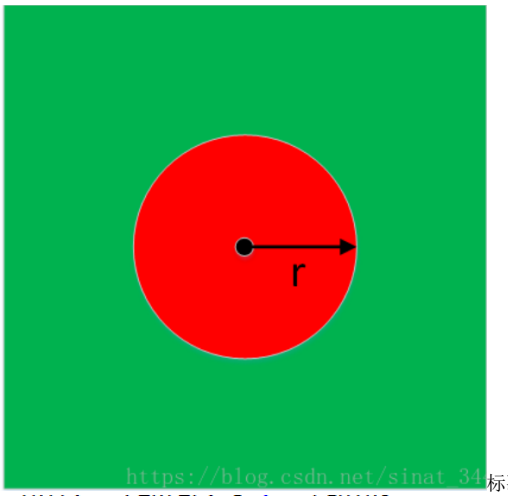
用这个函数可以在获取的图片中找到多个圆,返回值为圆的纵横坐标和半径.

3.__selectCircle(self, circles)挑选圆:

numpy中的shape属性:返回numpy数组的规格,例如array([1,2,3])返回为(3,)如果是array([[12,2],[11,3]]),返回(2,2)

```
    """
    if circles.shape[0] == 0:
        return circles
    if circles.shape[0] == 1:
        centerX = circles[0][0]
        centerY = circles[0][1]
        radius = circles[0][2]
        initX = centerX - 2 * radius
        initY = centerY - 2 * radius
        if (initX < 0 or initY < 0 or (initX + 4 * radius) > self.frameWid
                (initY + 4 * radius) > self.frameHeight or radius < 1):
            return circles
```

代码中if第一个说明没找到圆,第二个说明只找到一个圆.



标准情况.实际情况条件放宽.

```
if (initX < 0 or initY < 0 or (initX + 4 * radius) > self.frameWidth or \
        (initY + 4 * radius) > self.frameHeight or radius < 1):
    return circles
```

numpy中的flatten()函数:函数没有参数,可以把数组多维变成1维

4.__updateBallPositionFitting():

主要功能定位球的位置