任务一：基于国庆作业实现对待检测图片的传输

在本次实验中，我们首先需要通过ROS来完成对需要进行待检测图片的传输。这部分实验顺序是基于国庆作业中的任务一，可以以该任务为基础完成本次实验。

创建相关package

根据国庆作业，在根目录下建立catkin_ws文件夹

```
1, cd ~/catkin_ws/src
2, catkin_create_pkg image_tran rospy roscpp std_msgs image_transport
cv_bridge
3, cd image_tran
4, mkdir -p scripts && cd scripts
5, code .
```

相关数据准备

请在网络学堂上下载python_data.zip并解压，在之后的所有篇幅中，/path/to/your/python_data均代指python_data所对应的目录

image_pub

拷贝image_pub_node.py到scripts目录下

```
1, cd ~/catkin_ws/src/image_tran/scripts
2, cp /path/to/your/python_data/image_pub_node.py .
```

代码结构如下：

```
#!/usr/bin/env python
#coding:utf-8
import rospy
import sys
import cv2
import os
import numpy as np
from sensor_msgs.msg import Image
import random
from cv_bridge import CvBridge, CvBridgeError

root = "/path/to/your/python_data"
def pubImage():
    rospy.init_node('pubImage', anonymous = True)
    pub = rospy.Publisher('ShowImage', Image, queue_size = 10)
```

```
    rate = rospy.Rate(5)
    bridge = CvBridge()
    path = os.path.join(root, "ball_env.jpeg")
    image = cv2.imread(path)
    h, w, _ = image.shape
    while not rospy.is_shutdown():
        # 生成切分后的图片
        start_h = int(random.random() * h / 4)
        height = int(h * 3 / 4)
        start_w = int(random.random() * w / 4)
        width = int(w * 3 / 4)
        transfer_image = image[start_h:(start_h+height),start_w:
(start_w+width),:]
        # 传输切分后的图片
        pub.publish(bridge.cv2_to_imgmsg(transfer_image, "bgr8"))
        rate.sleep()
if __name__ == '__main__':
    try:
        pubImage()
    except rospy.ROSInterruptException:
        pass
```

其中需要注意的是需要将root变量改为/path/to/your/python_data

```
root = "/path/to/your/python_data"
```

在以下代码中，random.random()生成[0,1)之间内的一个随机数，通过对原始图片的切分，可以生成从一个随机位置起始，高和宽均为原始图片3/4的图片

```
# 生成切分后的图片
start_h = int(random.random() * h / 4)
height = int(h * 3 / 4)
start_w = int(random.random() * w / 4)
width = int(w * 3 / 4)
transfer_image = image[start_h:(start_h+height),start_w:(start_w+width),:]
```

image_sub

拷贝image_sub_node.py到scripts目录下

```
1, cd ~/catkin_ws/src/image_tran/scripts
2, cp /path/to/your/python_data/image_sub_node.py .
```

代码结构如下：

```
#!/usr/bin/env python
#coding:utf-8
import rospy
import cv2
from sensor_msgs.msg import Image
from cv_bridge import CvBridge,CvBridgeError
def callback(data):
    bridge = CvBridge()
    cv_image = bridge.imgmsg_to_cv2(data, "bgr8")
    cv2.imshow("view", cv_image)
def showImage():
    rospy.init_node('showImage', anonymous = True)
    cv2.namedWindow("view", cv2.WINDOW_AUTOSIZE)
    cv2.startWindowThread()
    rospy.Subscriber('ShowImage', Image, callback)
    rospy.spin()
    cv2.destroyWindow("view")
if __name__ == '__main__':
    showImage()
```

launch

为了方便同学们执行整个流程，可以将launch的策略放入默认的launch中，代码如下

```
1, cd ~/catkin_ws/src/image_tran
2, mkdir -p launch && cd launch
3, cp /path/to/your/python_data/start.launch .
```

start.launch包含如下内容

```
<launch>
    <node pkg="image_tran" type="image_sub_node.py" name="sub_node"
output="screen"/>
    <node pkg="image_tran" type="image_pub_node.py" name="pub_node"
output="screen"/>
</launch>
```

运行

完成上述代码后，可以通过以下脚本来执行

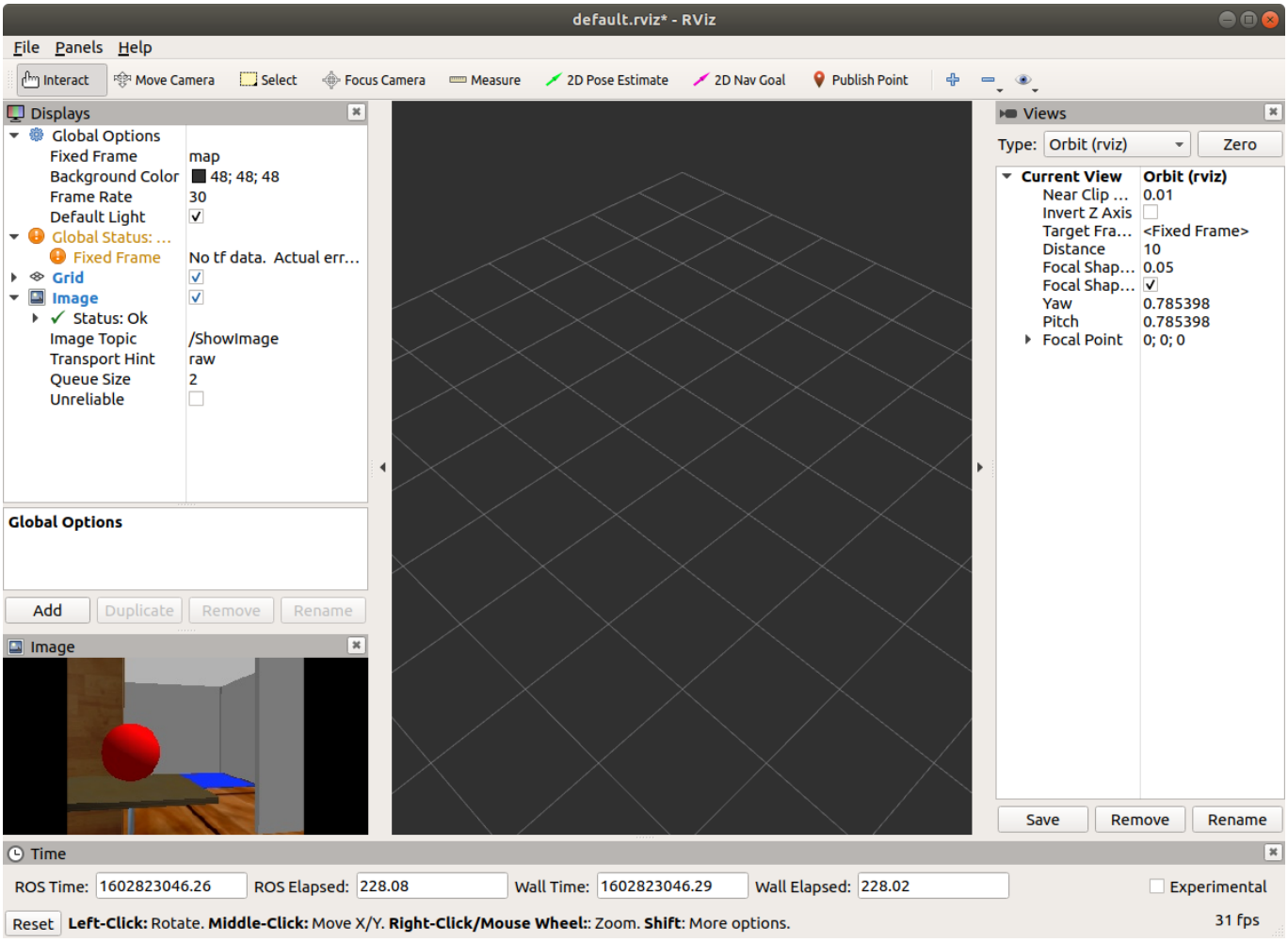
```
1, cd ~/catkin_ws
2, catkin_make
3, source devel/setup.bash
4, roslaunch image_tran start.launch
```

可以看到图片中的小球位置在抖动，这也是在现实场景中常见的情况，因此需要对目标进行定位。

同样可以使用RVIZ对传输的图片进行监控

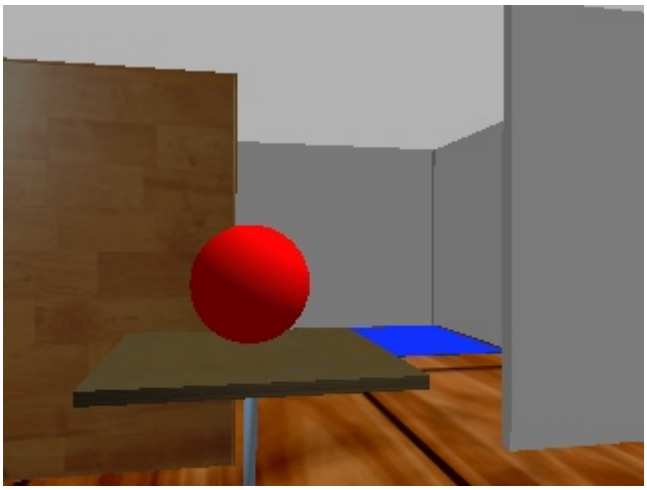
```
rosrun rviz rviz
```

点击左下角的Add，在弹出的对话框中选择By topic，再选择Image即可。界面如下：



任务二：基于OpenCV完成对图片的检

在本次实验中，我们的目标是从以下图片中找到红色的小球



拷贝cv2_image_sub_node.py到scripts目录下

```
1, cd ~/catkin_ws/src/image_tran/scripts
2, cp /path/to/your/python_data/cv2_image_sub_node.py image_sub_node.py
```

代码结构如下

```
#!/usr/bin/env python
#coding:utf-8
import os
import rospy
import cv2
from sensor_msgs.msg import Image
from cv_bridge import CvBridge,CvBridgeError
import numpy as np
root = "/path/to/your/python_data"
def callback(data):
    bridge = CvBridge()
    cv_image = bridge.imgmsg_to_cv2(data, "bgr8")
    # transfer gray and load template
    cv_image_gray = cv2.cvtColor(cv_image, cv2.COLOR_BGR2GRAY)
    template_path = os.path.join(root, "ball.jpeg")
    template_image = cv2.imread(template_path, cv2.IMREAD_GRAYSCALE)
    res = cv2.matchTemplate(cv_image_gray, template_image, cv2.TM_SQDIFF)
    # find max index, and plot
    index_position = np.where(res == np.min(res))
    index_h, index_w = index_position[0][0], index_position[1][0]
    template_h, template_w = template_image.shape
    cv2.rectangle(cv_image, (index_w, index_h), (index_w + template_w,
index_h + template_h), [0,0,255], 2)
    cv2.imshow("view", cv_image)
def showImage():
    rospy.init_node('showImage', anonymous = True)
    cv2.namedWindow("view", cv2.WINDOW_AUTOSIZE)
    cv2.startWindowThread()
    rospy.Subscriber('ShowImage', Image, callback)
    rospy.spin()
```

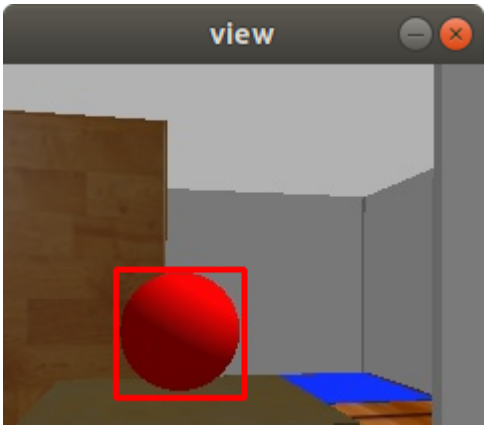
```
    cv2.destroyWindow("view")
if __name__ == '__main__':
    showImage()
```

同样需要替换/path/to/your/python_data/路径

然后再次执行以下命令即可看到

```
roslaunch image_tran start.launch
```

得到的结果如下图所示，可见检测的效果是非常准确的



相关部分代码解读： 1，transfer gray and load template，由于OpenCV模板匹配的限制，必须在灰度图上，所以首先需要将输入图片转换到灰度空间。然后将对应的路径进行修改，改为在本地电脑下的ball.jpeg，然后加载为灰度图。最后利用cv2.matchTemplate进行模板匹配，由于模板相较于原始图片会小，所以会在原始图片上进行划窗，然后计算划窗后的图片和模板图片的相似程度，TM_SQDIFF代表方差作为相似度，最终生成一个res矩阵，对应着每个坐标下划窗和模板的得分。

```
# transfer gray and load template
cv_image_gray = cv2.cvtColor(cv_image, cv2.COLOR_BGR2GRAY)
template_path = os.path.join(root, "ball.jpeg")
template_image = cv2.imread(template_path, cv2.IMREAD_GRAYSCALE)
res = cv2.matchTemplate(cv_image_gray, template_image, cv2.TM_SQDIFF)
```

2，find max index and plot，通过np.where查找最小值所在的位置（方差越小代表模板和划窗越接近），然后结合着模板的高度和宽度得到最终所要绘制框的左上角坐标和右下角坐标，最后通过cv2.rectangle将框画在原始图片上，[0,0,255]代表着框的颜色BGR，2代表着框线条宽度。

```
# find max index, and plot
index_position = np.where(res == np.min(res))
index_h, index_w = index_position[0][0], index_position[1][0]
template_h, template_w = template_image.shape
cv2.rectangle(cv_image, (index_w, index_h), (index_w + template_w, index_h
+ template_h), [0,0,255], 2)
```

任务三：基于CNN完成对图片的检测

安装相关环境

安装配置相关环境，由于需要sudo权限，可能会提示输入密码，不会显示，盲打输入即可。

步骤三需要的时间可能相对较久，请耐心等待

```
1, cd /path/to/your/python_data/
2, sudo apt install python-pip
3, pip install torch-1.3.0+cpu-cp27-cp27mu-linux_x86_64.whl
4, pip install numpy matplotlib tqdm future Pillow
```

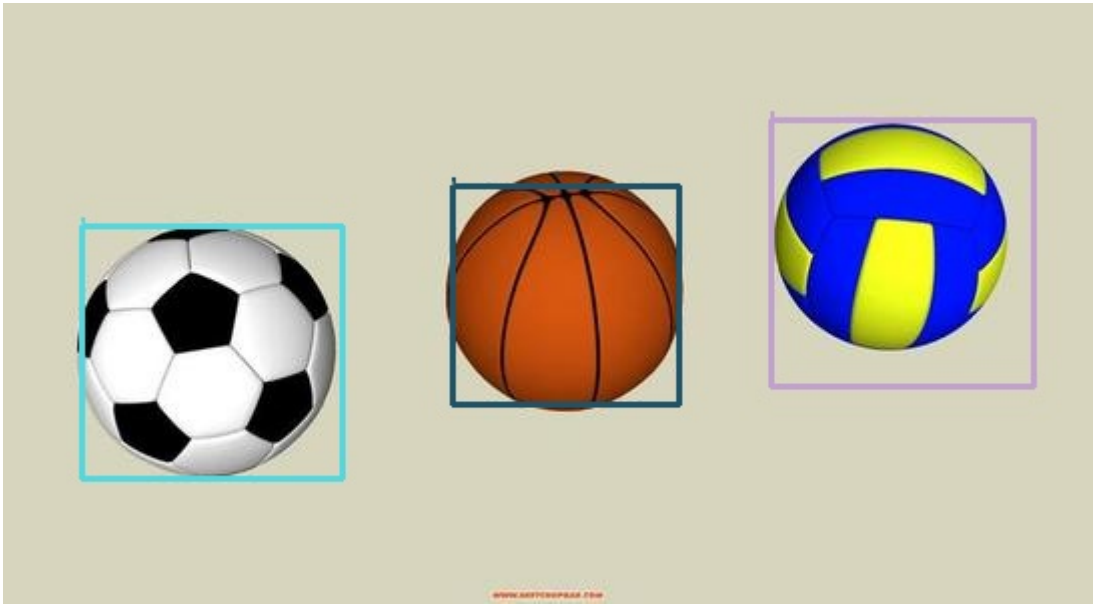
接着执行检测脚本即可测试对目标的检测效果

```
1, cd yolov3_detect
2, python detect.py
```

程序输出的结果应该为

```
result
1 basketballs
result
1 footballs
result
1 volleyballs
time: 0.2150819301615
tensor([[ 39.00000, 111.00000, 169.00000, 237.00000,   0.99957,   0.99995,   1.00000],
        [224.00000,  91.00000, 337.00000, 200.00000,   0.99901,   0.99996,   0.00000],
        [383.00000,  58.00000, 514.00000, 191.00000,   0.99340,   0.99999,   2.00000]], grad_fn=<CopySlices>)
```

同时在当前目录下会生成对图片的预测结果为



对输出结果的分析 最终的输出结果是一个数量不固定（每一个目标对应一个输出），但是每个元素长度固定为7的数组，其中前四位分别代表着位置,x,y,w,h，接着两位代表得分，最后一位代表着类别。

完成对image_pub和image_sub的改善来实现CNN目标检测

image_pub

由于检测速度较慢，所以需要将图片发送帧率改为1甚至更低，同时需要更改传输图片，使得训练的模型能够正确生效。

同样的，拷贝cnn_image_pub_node.py到scripts目录下

```
1, cd ~/catkin_ws/src/image_tran/scripts
2, cp /path/to/your/python_data/cnn_image_pub_node.py image_pub_node.py
```

代码结构如下：

```
#!/usr/bin/env python
#coding:utf-8
import rospy
import sys
import cv2
import os
import numpy as np
from sensor_msgs.msg import Image
import random
from cv_bridge import CvBridge, CvBridgeError
root = "/path/to/your/python_data"
def pubImage():
    rospy.init_node('pubImage', anonymous = True)
    pub = rospy.Publisher('ShowImage', Image, queue_size = 10)
    rate = rospy.Rate(1)
    bridge = CvBridge()
    path = os.path.join(root, "three_balls.jpeg")
    image = cv2.imread(path)
    h, w, _ = image.shape
    while not rospy.is_shutdown():
        # 生成切分后的图片
        start_h = int(random.random() * h / 4)
        height = int(h * 3 / 4)
        start_w = int(random.random() * w / 4)
        width = int(w * 3 / 4)
        transfer_image = image[start_h:(start_h+height),start_w:
(start_w+width),:]
        # 传输切分后的图片
        pub.publish(bridge.cv2_to_imgmsg(transfer_image, "bgr8"))
        rate.sleep()
if __name__ == '__main__':
    try:
        pubImage()
```

```
    except rospy.ROSInterruptException:
        pass
```

需要替换/path/to/your/python_data/路径

image_sub

在接收端，我们需要将目标检测实现 同样的，拷贝cnn_image_pub_node.py到scripts目录下

```
1, cd ~/catkin_ws/src/image_tran/scripts
2, cp /path/to/your/python_data/cnn_image_sub_node.py image_sub_node.py
```

```
#!/usr/bin/env python
#coding:utf-8
import os
import rospy
import cv2
from sensor_msgs.msg import Image
from cv_bridge import CvBridge,CvBridgeError
import numpy as np
import commands
root = "/path/to/your/python_data"
def callback(data):
    bridge = CvBridge()
    cv_image = bridge.imgmsg_to_cv2(data, "bgr8")
    # save image, detect, read result
    abs_path = os.path.join(root, "yolov3_detect")
    cv2.imwrite("%s/data/samples/three_balls.jpg" % abs_path, cv_image)
    os.system('cd %s && python detect.py' % abs_path)
    res_image = cv2.imread("%s/ball_detect.jpg" % abs_path)
    cv2.imshow("view", res_image)
def showImage():
    rospy.init_node('showImage', anonymous = True)
    cv2.namedWindow("view", cv2.WINDOW_AUTOSIZE)
    cv2.startWindowThread()
    rospy.Subscriber('ShowImage', Image, callback)
    rospy.spin()
    cv2.destroyWindow("view")
if __name__ == '__main__':
    showImage()
```

代码说明，需要替换/path/to/your/python_data/路径，然后将接收到的图片写入到一个固定的路径，然后利用os.system执行detect.py命令，再然后就可以将输出的结果取出作为最终检测的结果

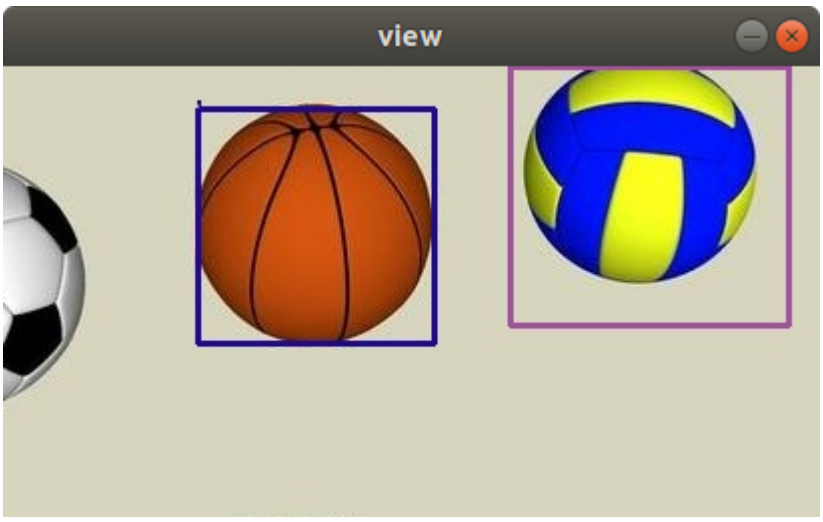
```
abs_path = os.path.join(root, "yolov3_detect")
cv2.imwrite("%s/data/samples/three_balls.jpg" % abs_path, cv_image)
```

```
os.system('cd %s && python detect.py' % abs_path)
res_image = cv2.imread("%s/ball_detect.jpg" % abs_path)
```

运行

```
roslaunch image_tran start.launch
```

得到的结果如下图所示



任务四：采用霍夫圆检测代替模板匹配

可以采用霍夫圆检测代替YOLO，相关的代码如下：

```
gray_image = cv2.cvtColor(cv_image, cv2.COLOR_BGRA2GRAY)
circles = cv2.HoughCircles(gray_image, cv2.HOUGH_GRADIENT, 1, 50,
param1=100, param2=50, minRadius=0, maxRadius=0)
circles = np.uint16(np.around(circles))
for i in circles[0, :]:  # 遍历矩阵每一行的数据
    cv2.circle(cv_image, (i[0], i[1]), i[2], (0, 255, 0), 2)
    cv2.circle(cv_image, (i[0], i[1]), 2, (0, 0, 255), 3)
```

同时我们还可以将时间纳入到检测算法的考察中去，添加如下代码到image_sub_node.py中

```
import time

def callback(data):
    bridge = CvBridge()
    cv_image = bridge.imgmsg_to_cv2(data, "bgr8")
    start_time = time.clock()
    # save image, detect, read result
    abs_path = os.path.join(root, "yolov3_detect")
```

```
    cv2.imwrite("%s/data/samples/three_balls.jpg" % abs_path, cv_image)
    os.system('cd %s && python detect.py' % abs_path)
    res_image = cv2.imread("%s/ball_detect.jpg" % abs_path)
    end_time = time.clock()
    print("time cost is %f s" % (end_time - start_time))
    cv2.imshow("view", res_img)
```