深蓝学院
shenlanxueyuan.com

古月居
GYH.AI

ROS理论与实践

—— 第11讲：机器人综合应用—— “迷宫寻宝”

主讲人 胡春旭

机器人博客“古月居”博主
《ROS机器人开发实践》作者
武汉精锋微控科技有限公司 联合创始人
华中科技大学 人工智能与自动化学院 硕士

目录

1. “迷宫寻宝”之任务发布

2. “迷宫寻宝”之任务分析

3. “迷宫寻宝”之任务实现

1. “迷宫寻宝”之任务发布

1. “迷宫寻宝” 之任务发布

传说，有一座奇幻迷宫，其中藏有一处诱人的宝藏，无数“人”进去却再未出来，只有最勇敢、聪明的“人”才能够获取宝藏。

So，**你敢接受挑战么?**

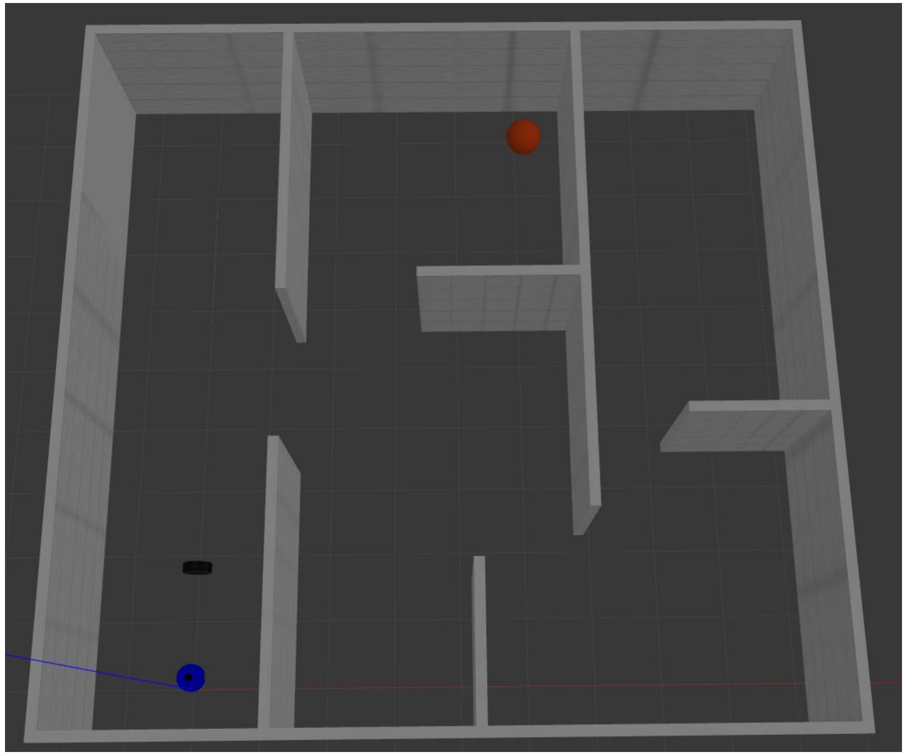
1.“迷宫寻宝”之任务发布

已知迷宫地图的覆盖范围（10*10m），其中某处藏有一处明显标记的宝藏（红色圆球），机器人在环境未知的情况下从起点出发，自主寻找宝藏，寻得宝藏之后需返回起点，任务完成。

- 地图预先使用Gazebo创建完成，并在Gazebo完成所有任务；
- 机器人在起点运动时开始计时，寻得宝藏回到起点后计时终止，须在5min内完成任务；
- 机器人搭载的传感器没有限制；
- 机器人接近宝藏1m范围内即认为获取宝藏；
- 机器人获取宝藏后，需通过语音播报状态信息；
- 宝藏的位置不固定，允许动态调整。

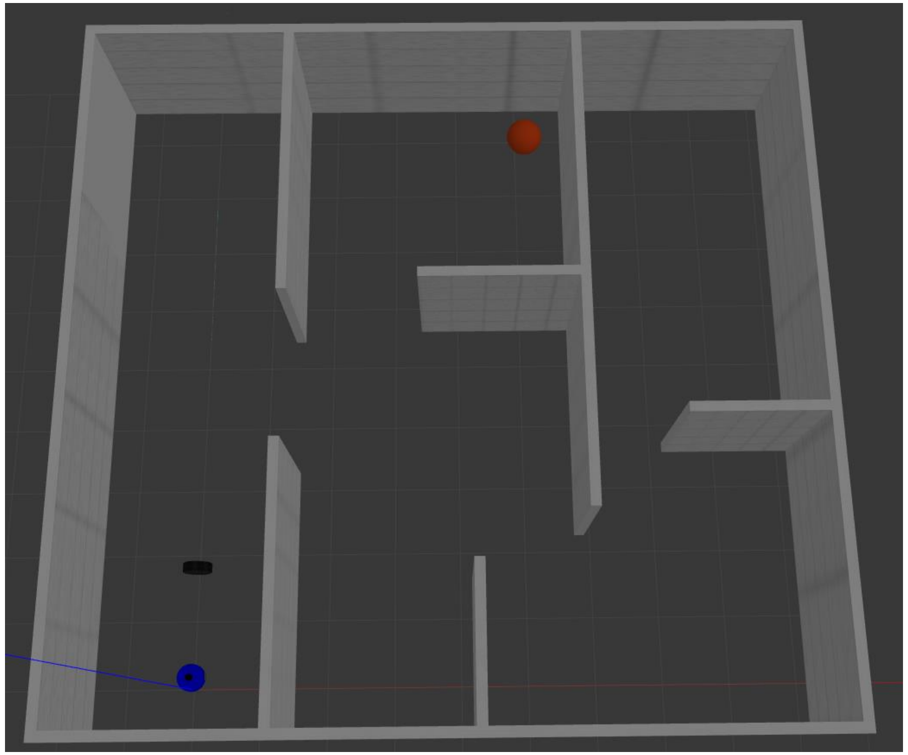
迷宫环境

```
$ roslaunch mbot_gazebo mbot_maze_gazebo.launch
```



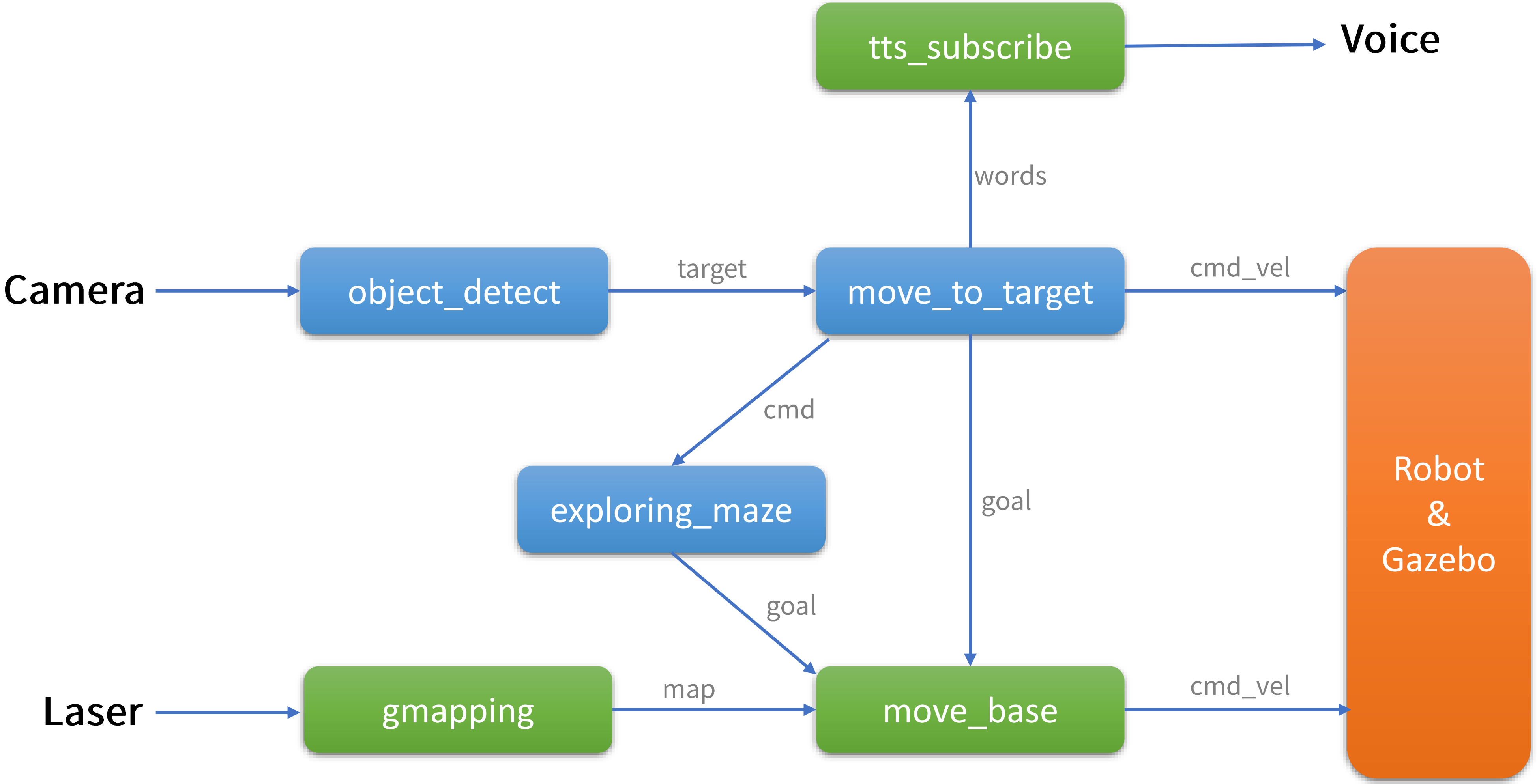
2.“迷宫寻宝”之任务分析

2. “迷宫寻宝”之任务分析



- 机器人建模仿真
- 图像识别
- 语音交互
- 自主导航
- SLAM
- ROS通信机制
- 系统设计与集成

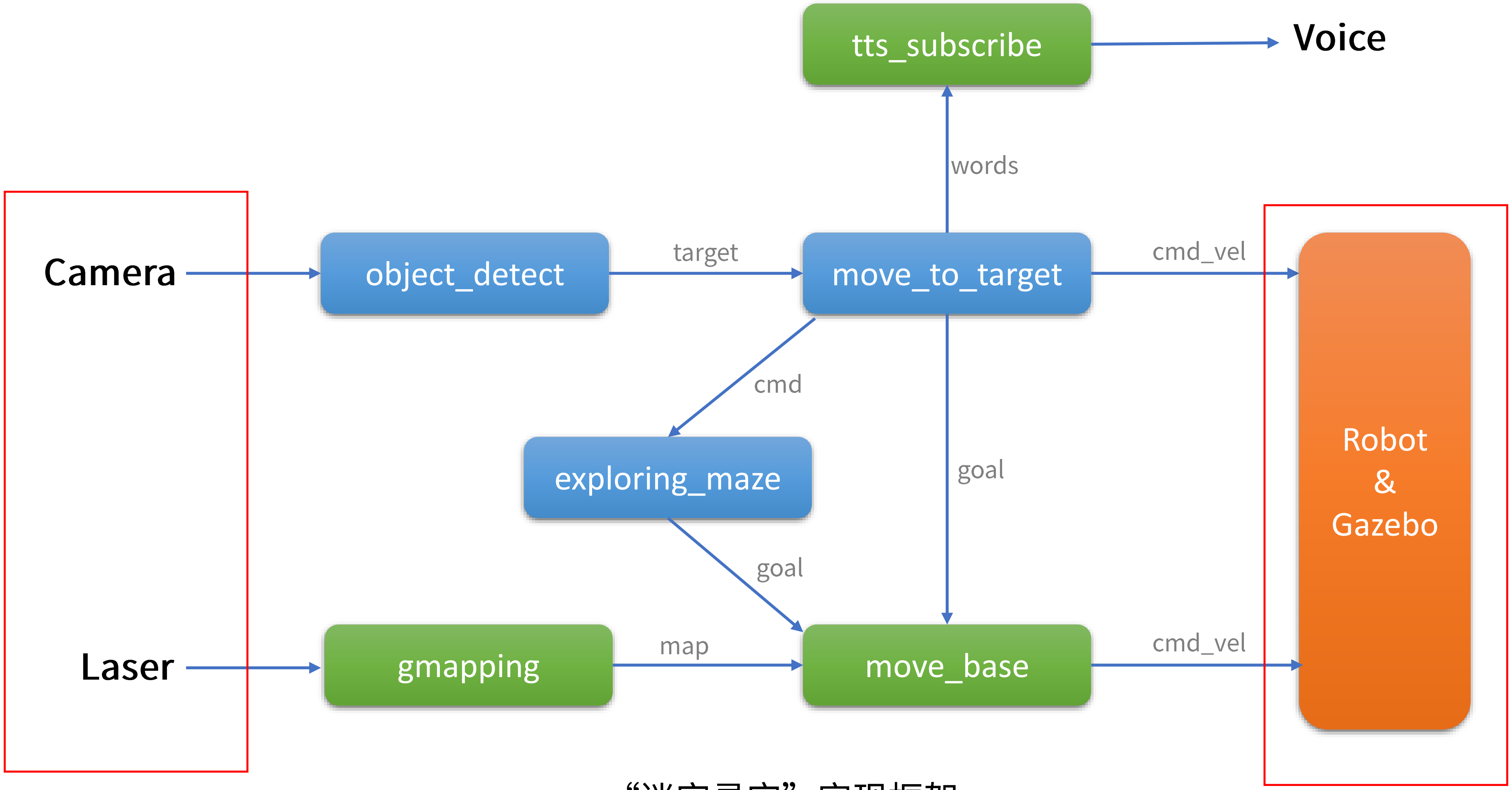
2.“迷宫寻宝”之任务分析



“迷宫寻宝”实现框架

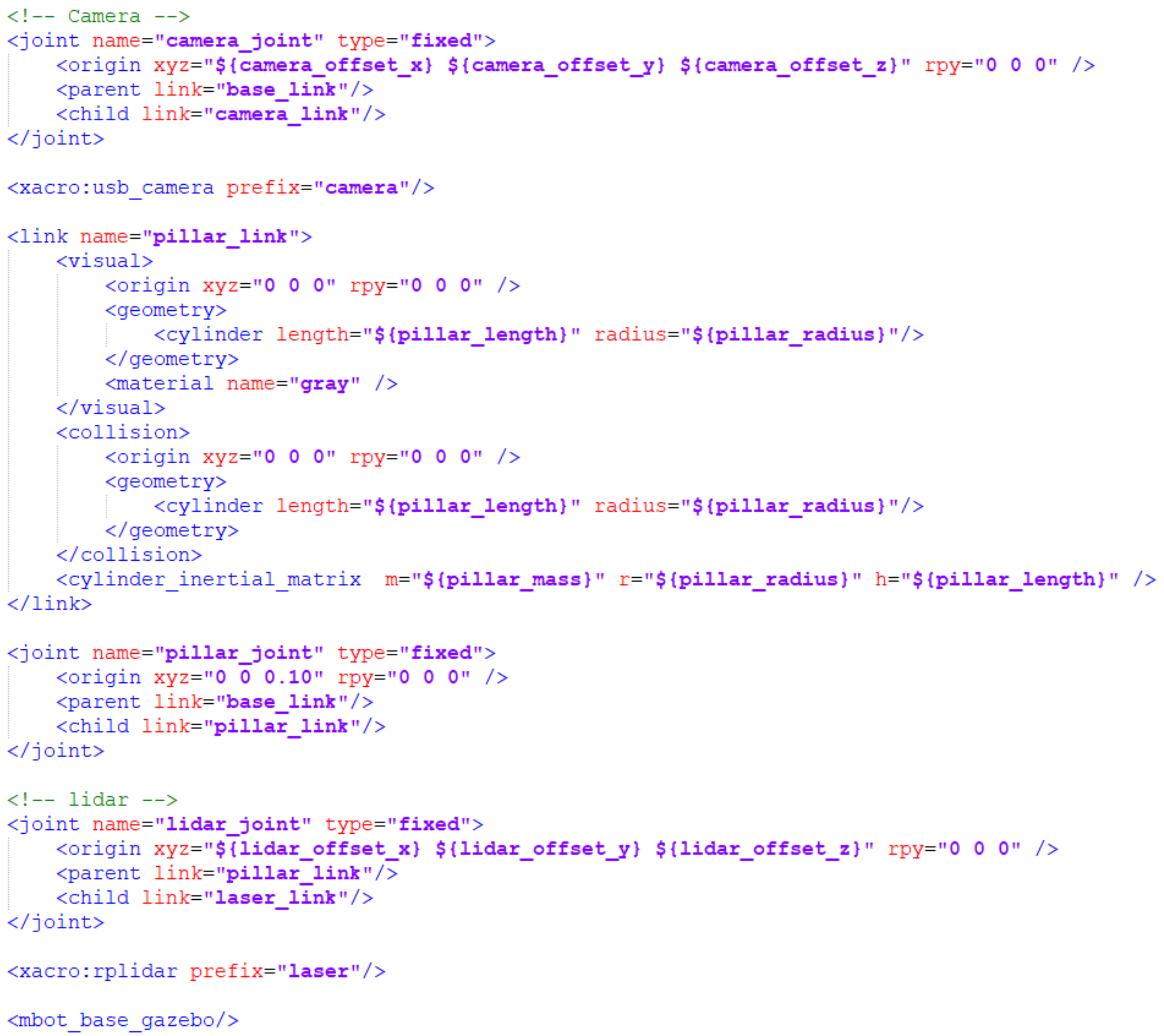
3.“迷宫寻宝”之任务实现

3.“迷宫寻宝”之任务实现 —— 机器人建模

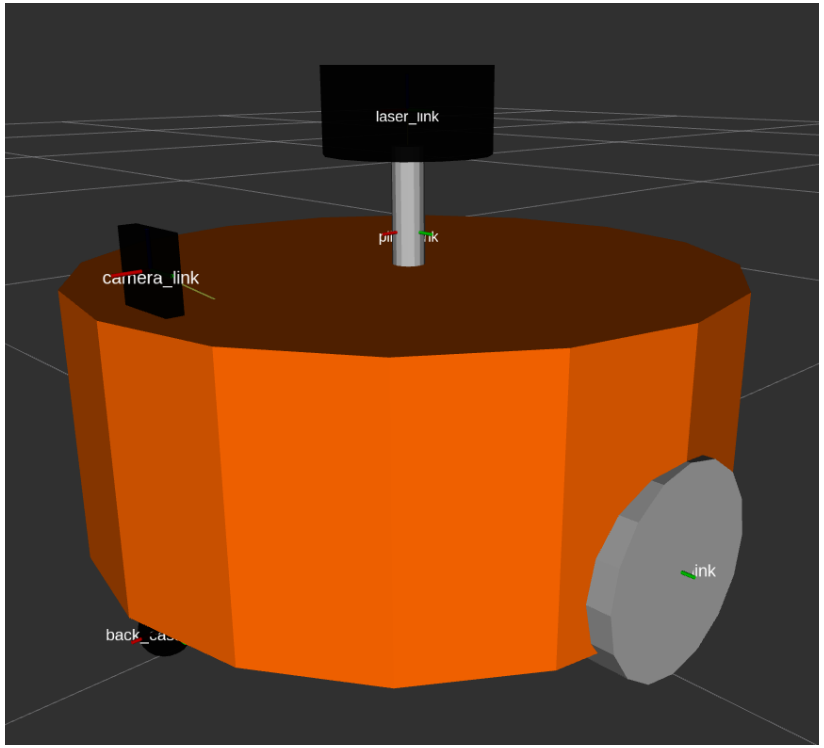


“迷宫寻宝”实现框架

3.“迷宫寻宝”之任务实现 —— 机器人建模

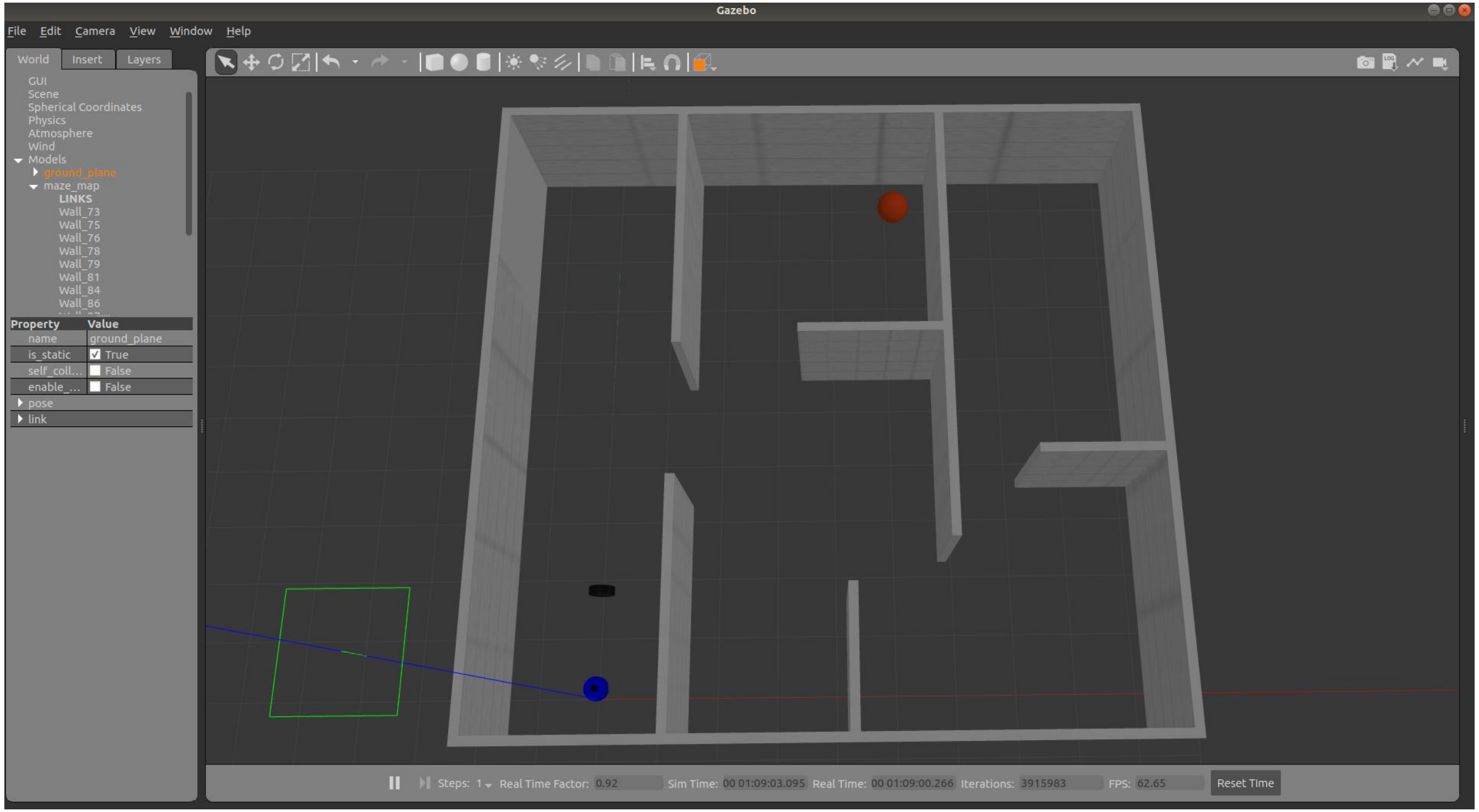


```
<!-- Camera -->
<joint name="camera_joint" type="fixed">
    <origin xyz="${camera_offset_x} ${camera_offset_y} ${camera_offset_z}" rpy="0 0 0" />
    <parent link="base_link"/>
    <child link="camera_link"/>
</joint>

<xacro:usb_camera prefix="camera"/>

<link name="pillar_link">
    <visual>
        <origin xyz="0 0 0" rpy="0 0 0" />
        <geometry>
            <cylinder length="${pillar_length}" radius="${pillar_radius}"/>
        </geometry>
        <material name="gray" />
    </visual>
    <collision>
        <origin xyz="0 0 0" rpy="0 0 0" />
        <geometry>
            <cylinder length="${pillar_length}" radius="${pillar_radius}"/>
        </geometry>
    </collision>
    <cylinder_inertial_matrix  m="${pillar_mass}" r="${pillar_radius}" h="${pillar_length}" />
</link>

<joint name="pillar_joint" type="fixed">
    <origin xyz="0 0 0.10" rpy="0 0 0" />
    <parent link="base_link"/>
    <child link="pillar_link"/>
</joint>

<!-- lidar -->
<joint name="lidar_joint" type="fixed">
    <origin xyz="${lidar_offset_x} ${lidar_offset_y} ${lidar_offset_z}" rpy="0 0 0" />
    <parent link="pillar_link"/>
    <child link="laser_link"/>
</joint>

<xacro:rplidar prefix="laser"/>

<mbot_base_gazebo/>
```



mbot_with_camera_laser_gazebo.xacro

3. “迷宫寻宝”之任务实现 —— 机器人建模



GUI
Scene
Spherical Coordinates
Physics
Atmosphere
Wind
Models
- ground_plane
- maze_map
  - LINKS
  - Wall_73
  - Wall_75
  - Wall_76
  - Wall_78
  - Wall_79
  - Wall_81
  - Wall_84
  - Wall_86

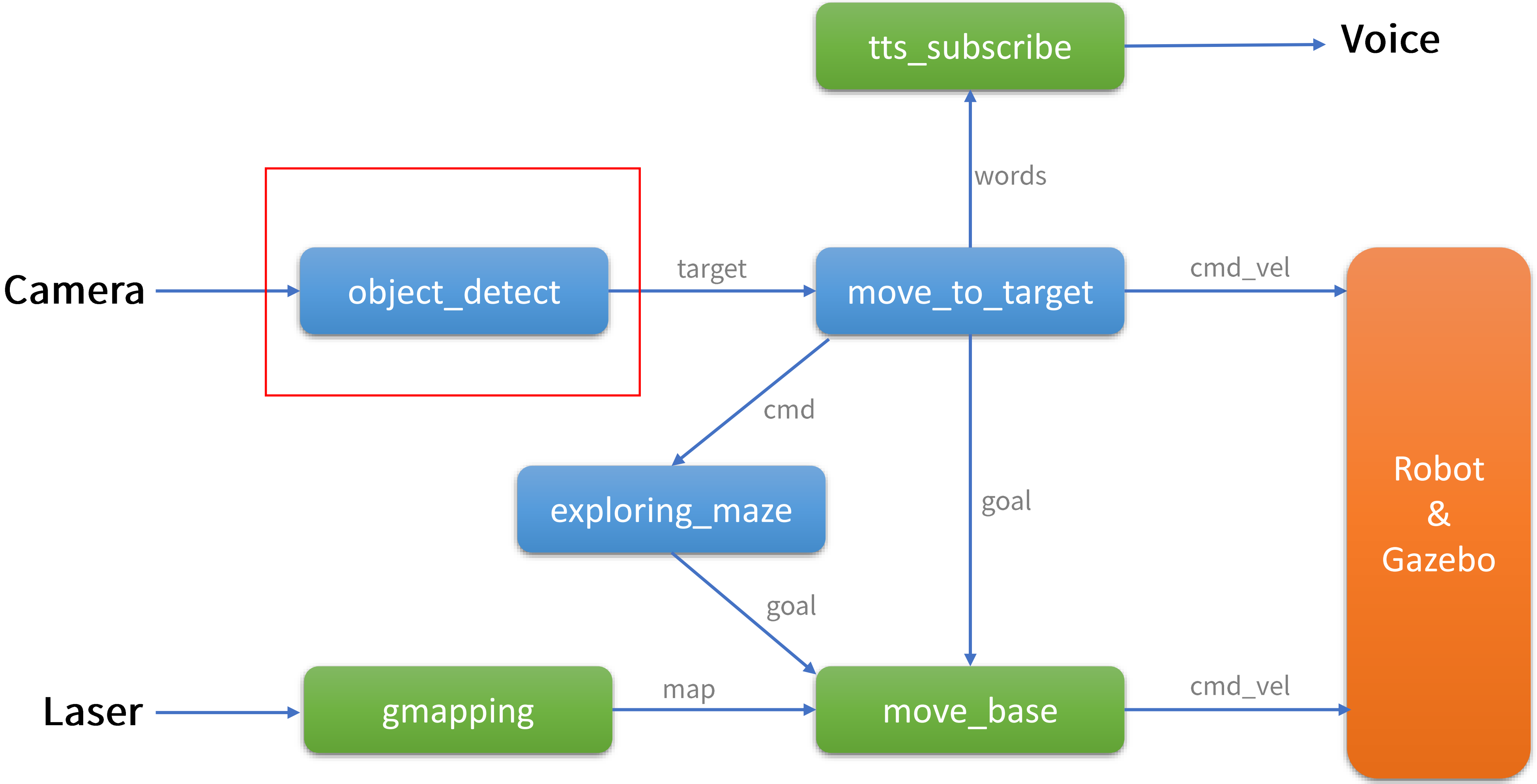
Property	Value
name	ground_plane
is_static	☑ True
self_coll...	□ False
enable_...	□ False
pose	
link	

```
Steps: 1  Real Time Factor: 0.92  Sim Time: 00 01:09:03.095  Real Time: 00 01:09:00.266  Iterations: 3915983  FPS: 62.65
```

迷宫环境

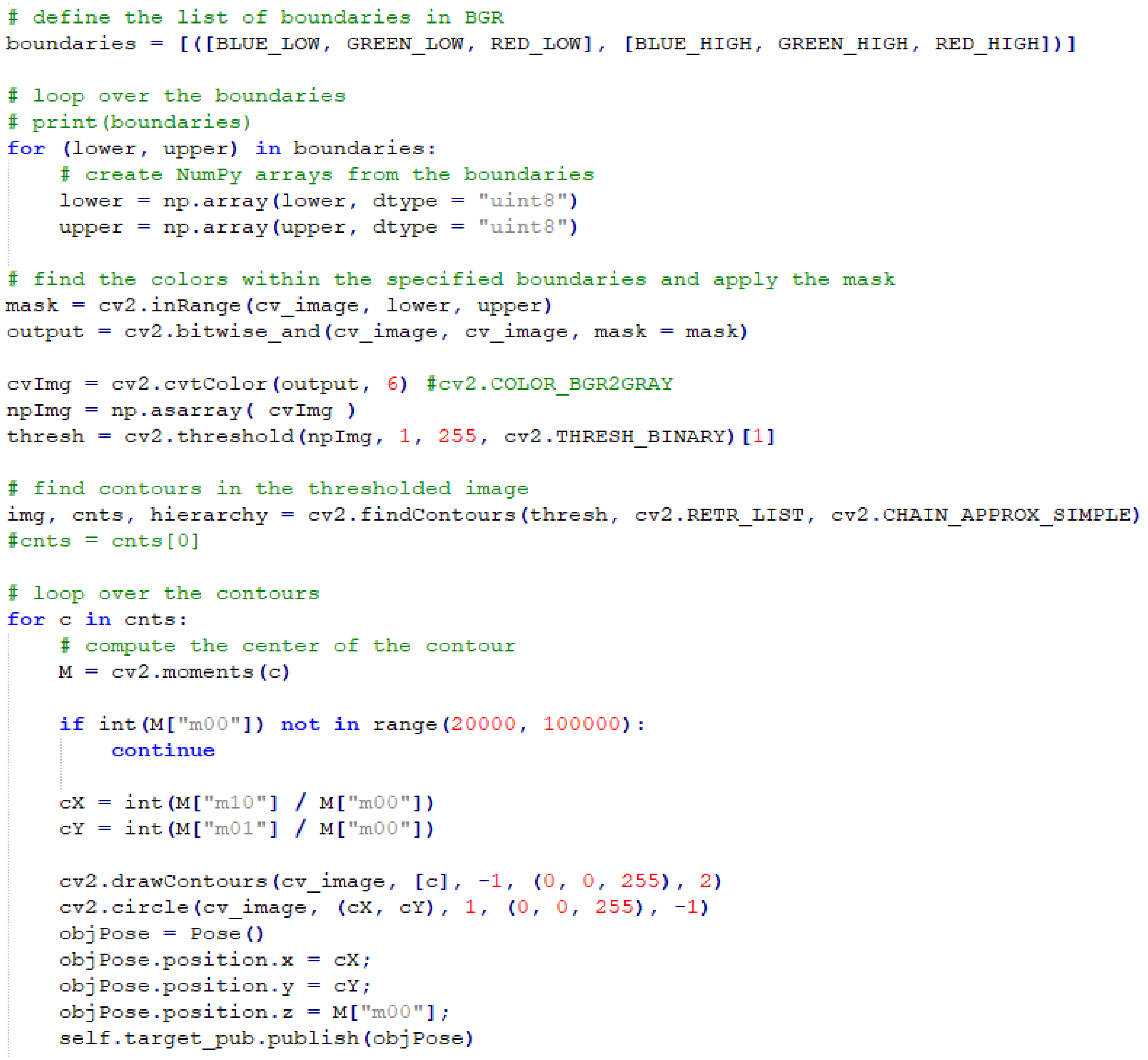
```
$ roslaunch mbot_gazebo mbot_maze_gazebo.launch
```

3.“迷宫寻宝”之任务实现 —— 宝藏识别



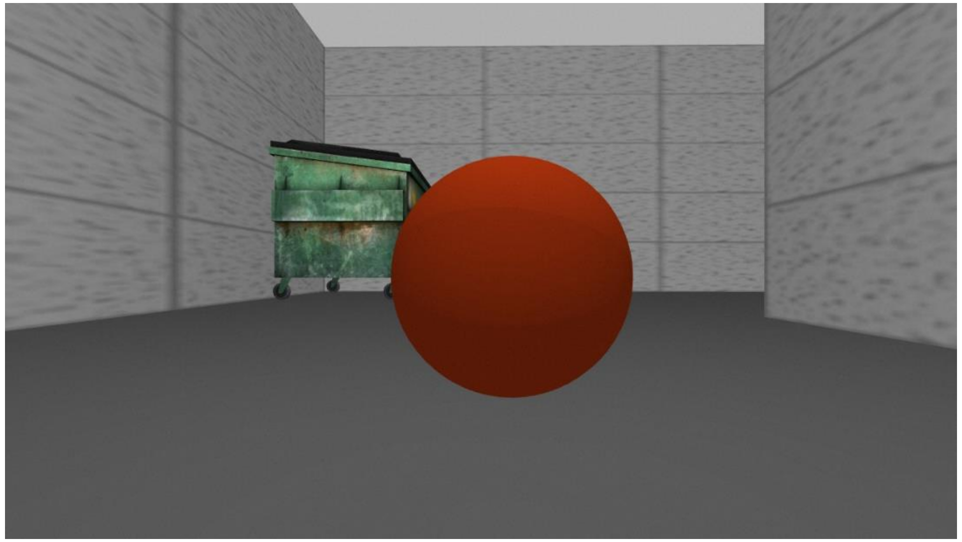
“迷宫寻宝”实现框架

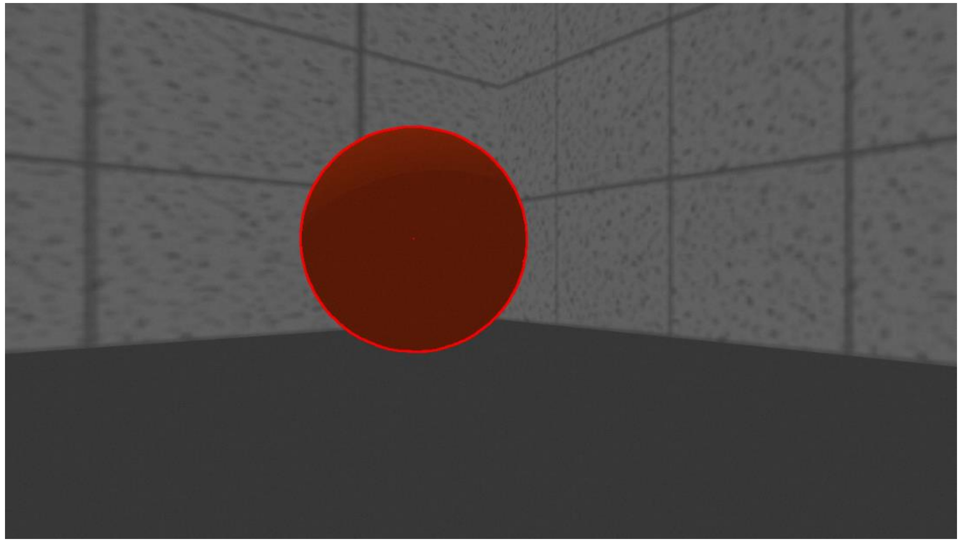
3. “迷宫寻宝”之任务实现 —— 宝藏识别



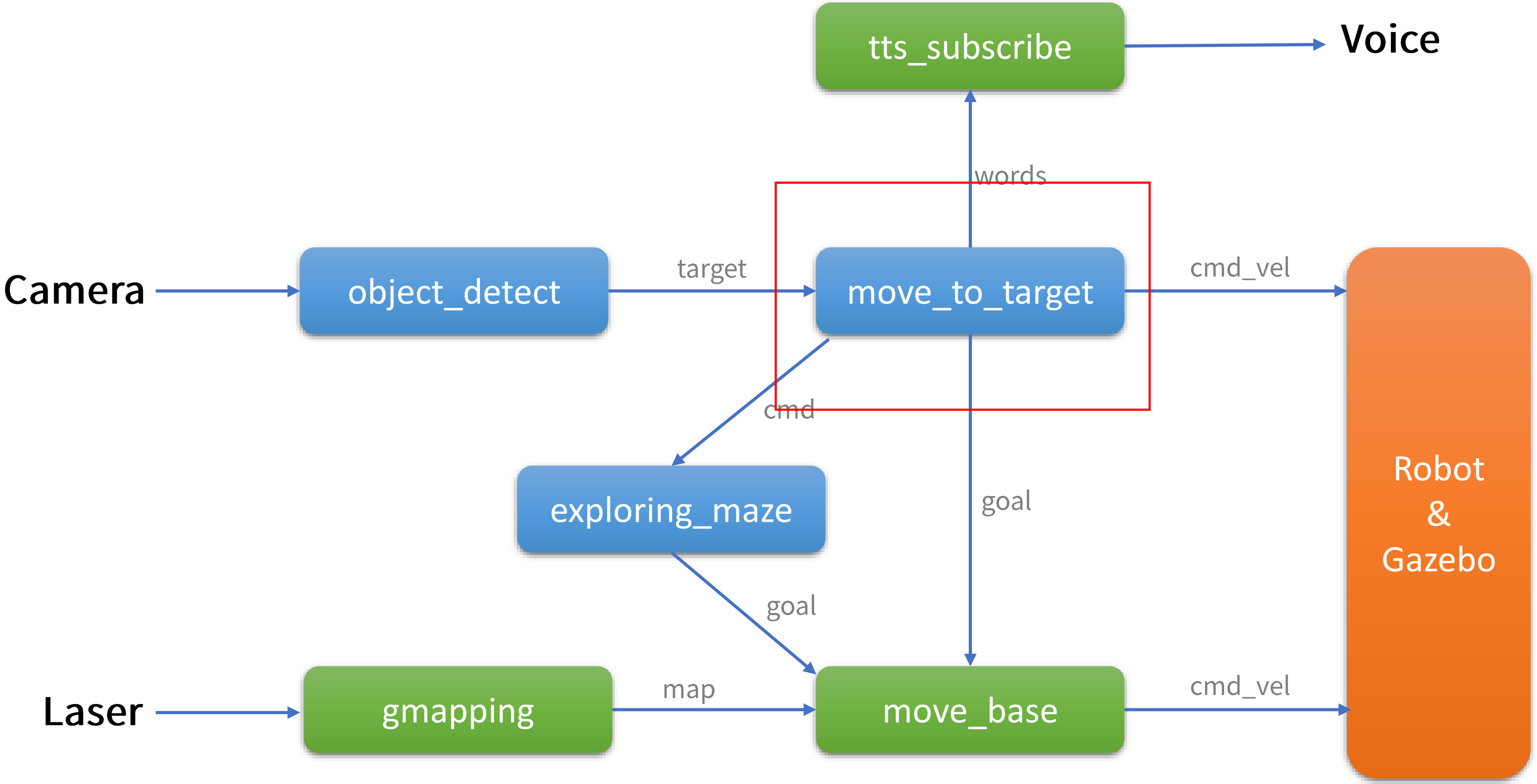
```
# define the list of boundaries in BGR
boundaries = [([BLUE_LOW, GREEN_LOW, RED_LOW], [BLUE_HIGH, GREEN_HIGH, RED_HIGH])]

# loop over the boundaries
# print(boundaries)
for (lower, upper) in boundaries:
    # create NumPy arrays from the boundaries
    lower = np.array(lower, dtype = "uint8")
    upper = np.array(upper, dtype = "uint8")

# find the colors within the specified boundaries and apply the mask
mask = cv2.inRange(cv_image, lower, upper)
output = cv2.bitwise_and(cv_image, cv_image, mask = mask)

cvImg = cv2.cvtColor(output, 6) #cv2.COLOR_BGR2GRAY
npImg = np.asarray( cvImg )
thresh = cv2.threshold(npImg, 1, 255, cv2.THRESH_BINARY)[1]

# find contours in the thresholded image
img, cnts, hierarchy = cv2.findContours(thresh, cv2.RETR_LIST, cv2.CHAIN_APPROX_SIMPLE)
#cnts = cnts[0]

# loop over the contours
for c in cnts:
    # compute the center of the contour
    M = cv2.moments(c)

    if int(M["m00"]) not in range(20000, 100000):
        continue

    cX = int(M["m10"] / M["m00"])
    cY = int(M["m01"] / M["m00"])

    cv2.drawContours(cv_image, [c], -1, (0, 0, 255), 2)
    cv2.circle(cv_image, (cX, cY), 1, (0, 0, 255), -1)
    objPose = Pose()
    objPose.position.x = cX;
    objPose.position.y = cY;
    objPose.position.z = M["m00"];
    self.target_pub.publish(objPose)
```

object_detect.py



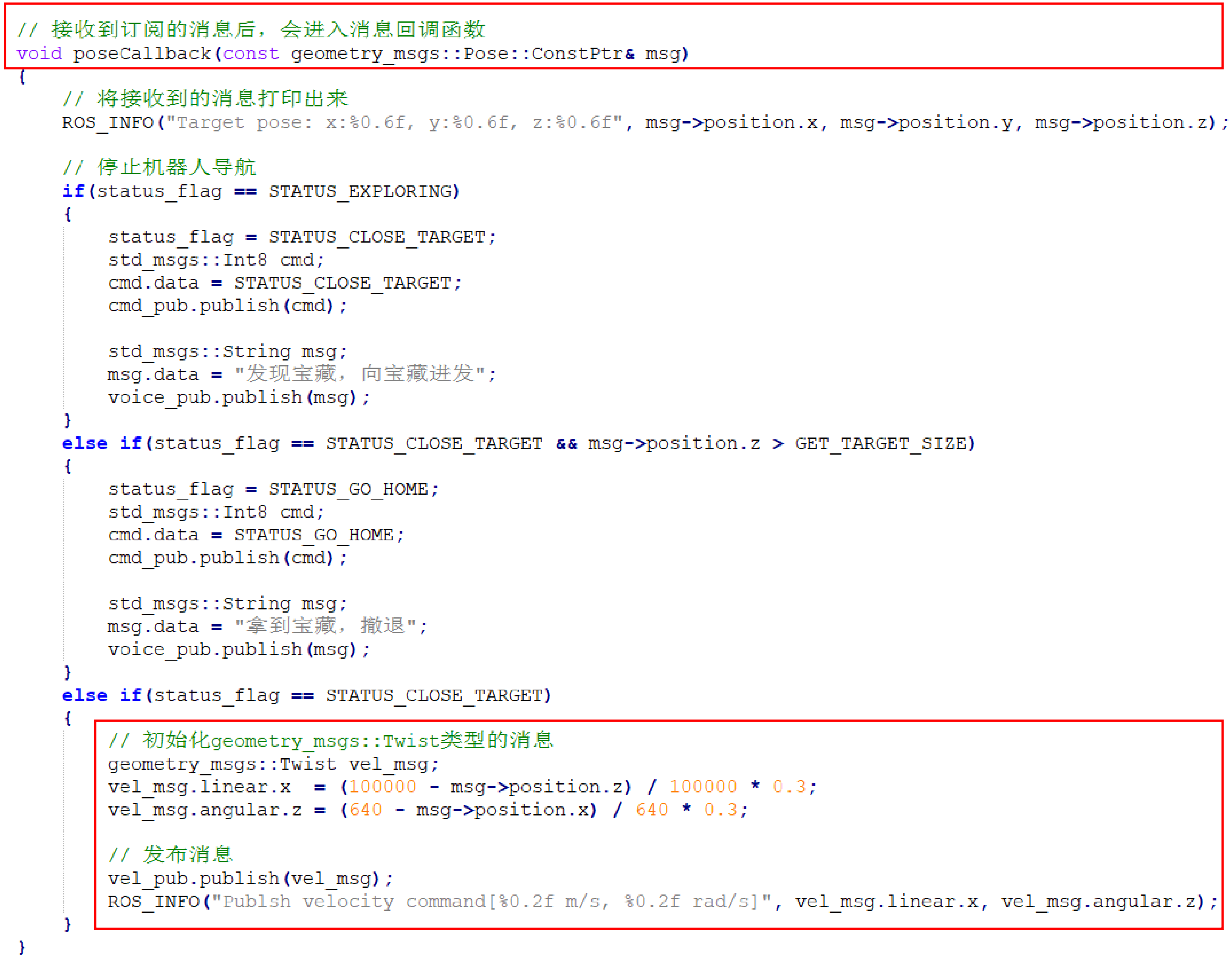


3. “迷宫寻宝”之任务实现 —— 视觉伺服



“迷宫寻宝”实现框架

3.“迷宫寻宝”之任务实现 —— 视觉伺服



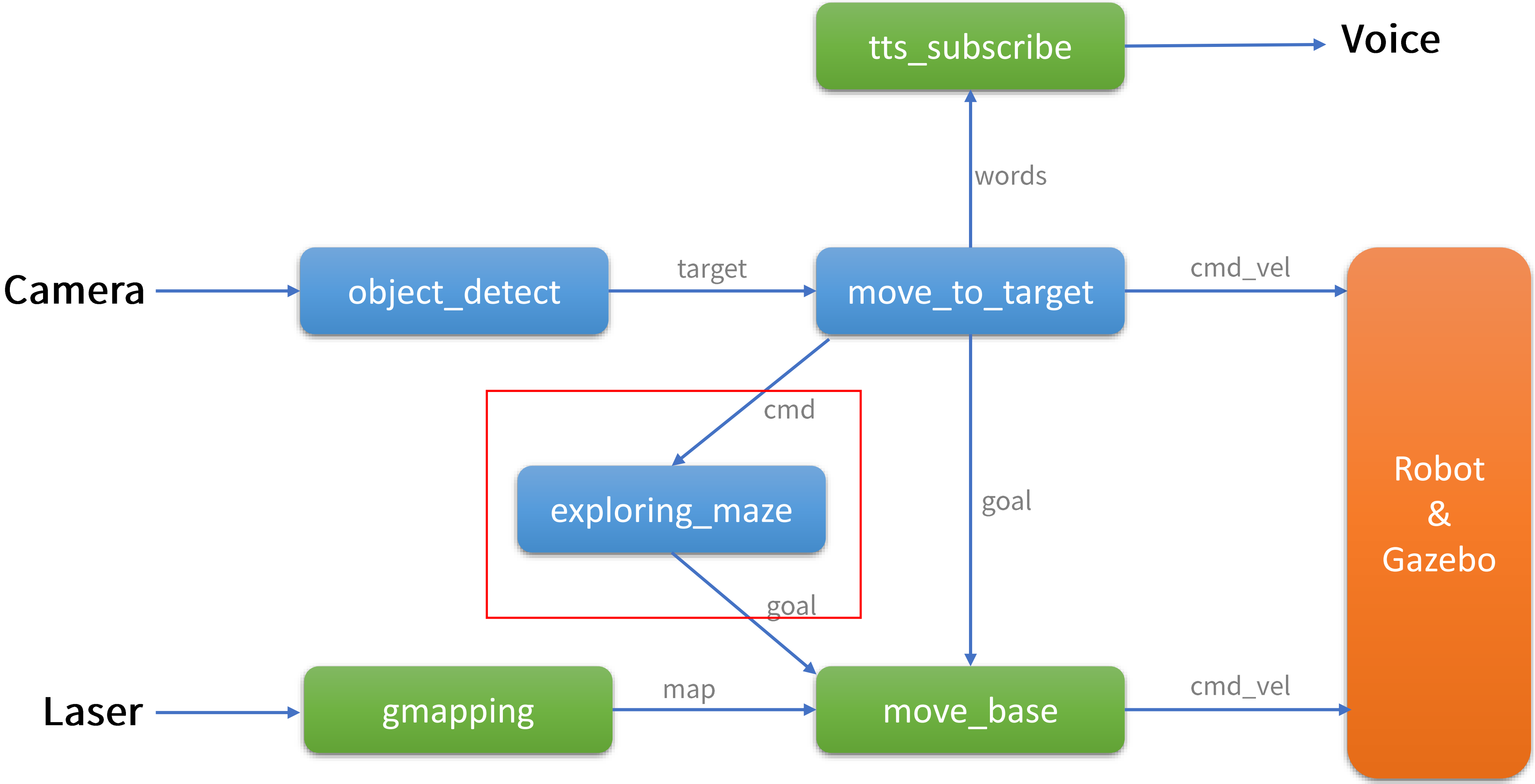
```
// 接收到订阅的消息后，会进入消息回调函数
void poseCallback(const geometry_msgs::Pose::ConstPtr& msg)
{
    // 将接收到的消息打印出来
    ROS_INFO("Target pose: x:%0.6f, y:%0.6f, z:%0.6f", msg->position.x, msg->position.y, msg->position.z);

    // 停止机器人导航
    if(status_flag == STATUS_EXPLORING)
    {
        status_flag = STATUS_CLOSE_TARGET;
        std_msgs::Int8 cmd;
        cmd.data = STATUS_CLOSE_TARGET;
        cmd_pub.publish(cmd);

        std_msgs::String msg;
        msg.data = "发现宝藏，向宝藏进发";
        voice_pub.publish(msg);
    }
    else if(status_flag == STATUS_CLOSE_TARGET && msg->position.z > GET_TARGET_SIZE)
    {
        status_flag = STATUS_GO_HOME;
        std_msgs::Int8 cmd;
        cmd.data = STATUS_GO_HOME;
        cmd_pub.publish(cmd);

        std_msgs::String msg;
        msg.data = "拿到宝藏，撤退";
        voice_pub.publish(msg);
    }
    else if(status_flag == STATUS_CLOSE_TARGET)
    {
        // 初始化geometry_msgs::Twist类型的消息
        geometry_msgs::Twist vel_msg;
        vel_msg.linear.x  = (100000 - msg->position.z) / 100000 * 0.3;
        vel_msg.angular.z = (640 - msg->position.x) / 640 * 0.3;

        // 发布消息
        vel_pub.publish(vel_msg);
        ROS_INFO("Publsh velocity command[%0.2f m/s, %0.2f rad/s]", vel_msg.linear.x, vel_msg.angular.z);
    }
}
```

订阅图像识别的结果

move_to_target.cpp

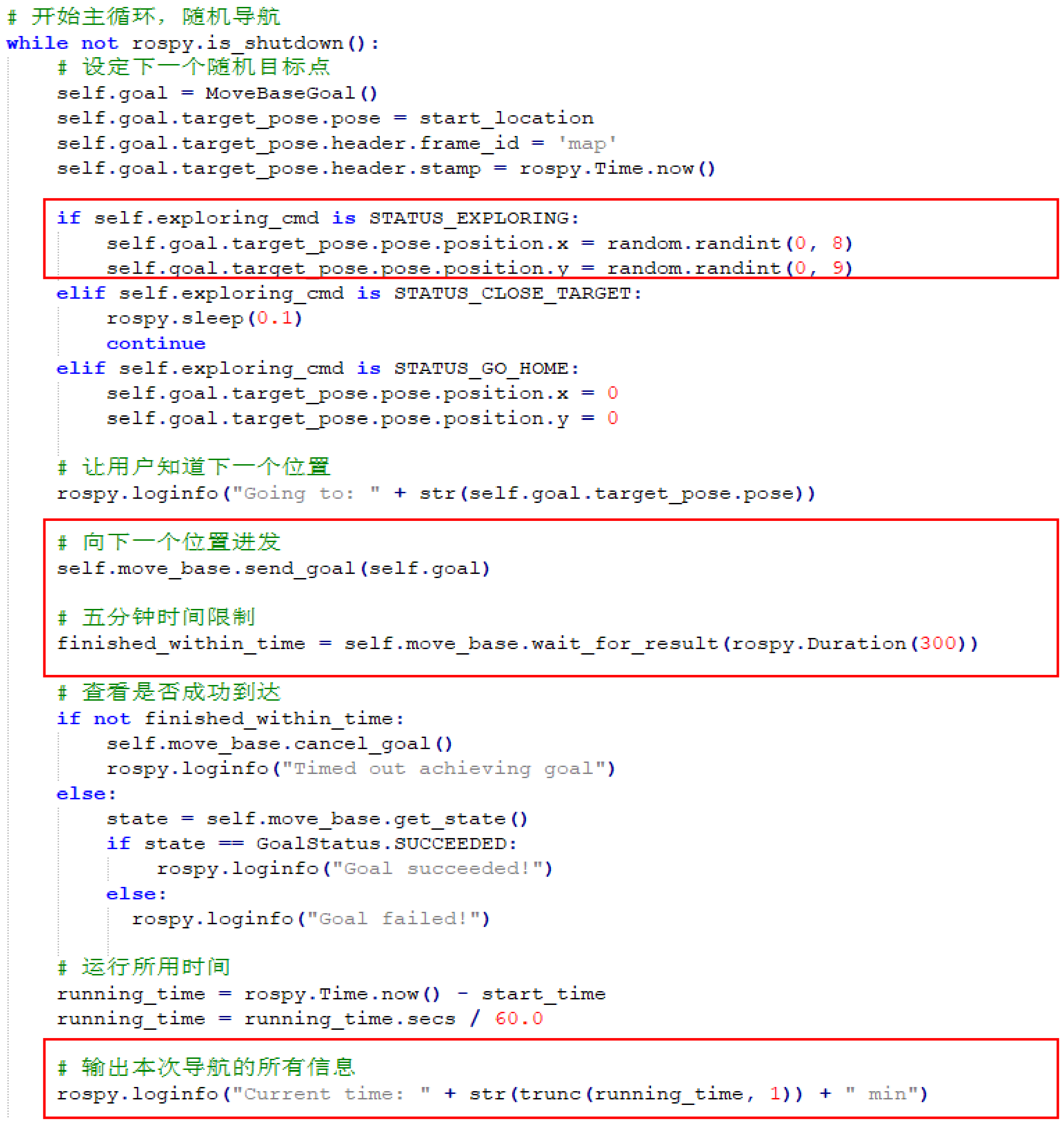
根据目标像素位置调整速度指令

3.“迷宫寻宝”之任务实现 —— 导航&SLAM



“迷宫寻宝”实现框架

3. “迷宫寻宝”之任务实现 —— 导航&SLAM



```
# 开始主循环，随机导航
while not rospy.is_shutdown():
    # 设定下一个随机目标点
    self.goal = MoveBaseGoal()
    self.goal.target_pose.pose = start_location
    self.goal.target_pose.header.frame_id = 'map'
    self.goal.target_pose.header.stamp = rospy.Time.now()

    if self.exploring_cmd is STATUS_EXPLORING:
        self.goal.target_pose.pose.position.x = random.randint(0, 8)
        self.goal.target_pose.pose.position.y = random.randint(0, 9)
    elif self.exploring_cmd is STATUS_CLOSE_TARGET:
        rospy.sleep(0.1)
        continue
    elif self.exploring_cmd is STATUS_GO_HOME:
        self.goal.target_pose.pose.position.x = 0
        self.goal.target_pose.pose.position.y = 0

    # 让用户知道下一个位置
    rospy.loginfo("Going to: " + str(self.goal.target_pose.pose))

    # 向下一个位置进发
    self.move_base.send_goal(self.goal)

    # 五分钟时间限制
    finished_within_time = self.move_base.wait_for_result(rospy.Duration(300))

    # 查看是否成功到达
    if not finished_within_time:
        self.move_base.cancel_goal()
        rospy.loginfo("Timed out achieving goal")
    else:
        state = self.move_base.get_state()
        if state == GoalStatus.SUCCEEDED:
            rospy.loginfo("Goal succeeded!")
        else:
          rospy.loginfo("Goal failed!")

    # 运行所用时间
    running_time = rospy.Time.now() - start_time
    running_time = running_time.secs / 60.0

    # 输出本次导航的所有信息
    rospy.loginfo("Current time: " + str(trunc(running_time, 1)) + " min")
```

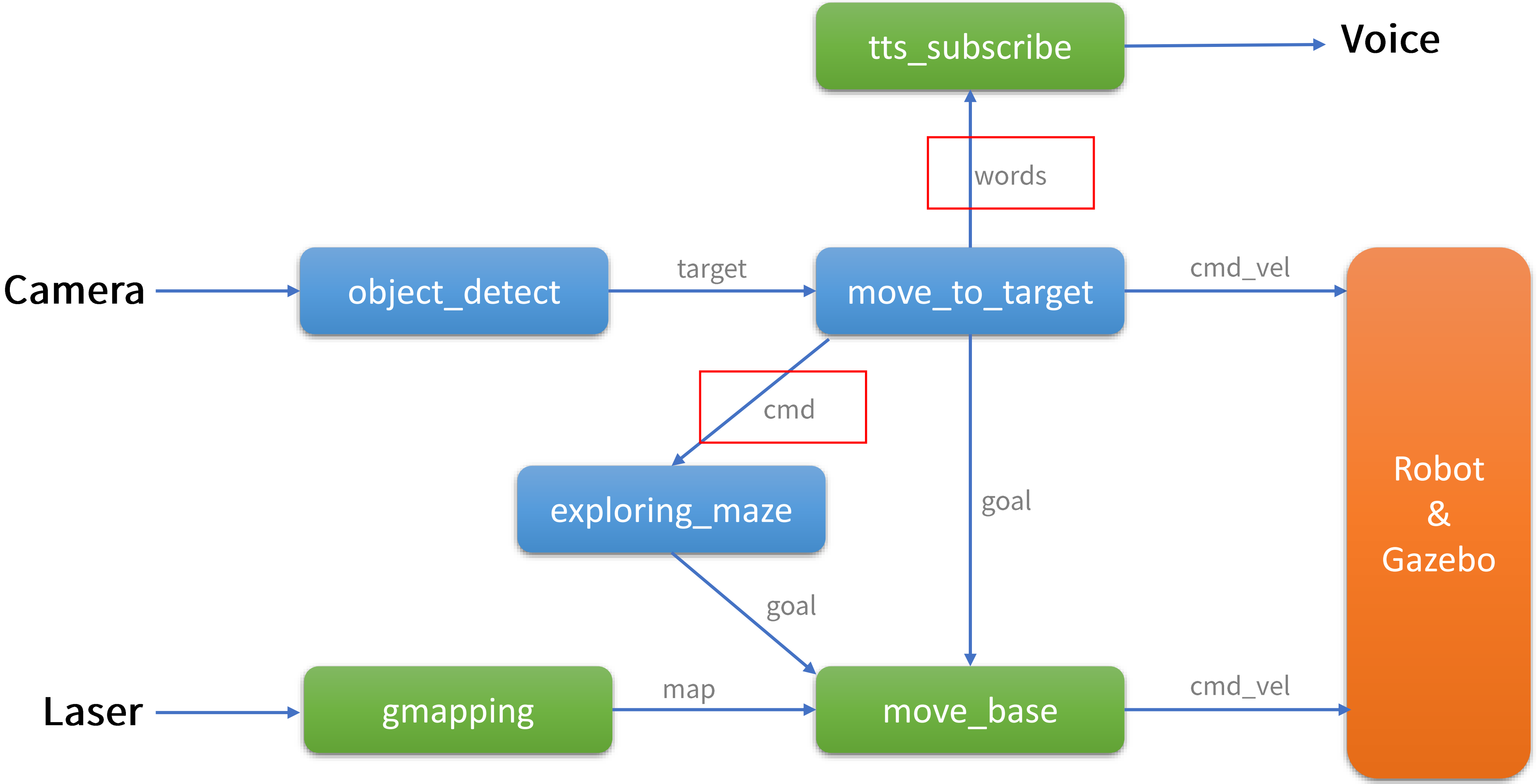
随机产生一个导航目标点进行寻宝

自主导航并视觉寻宝

exploring_maze.py

输出寻宝导航过程的时间

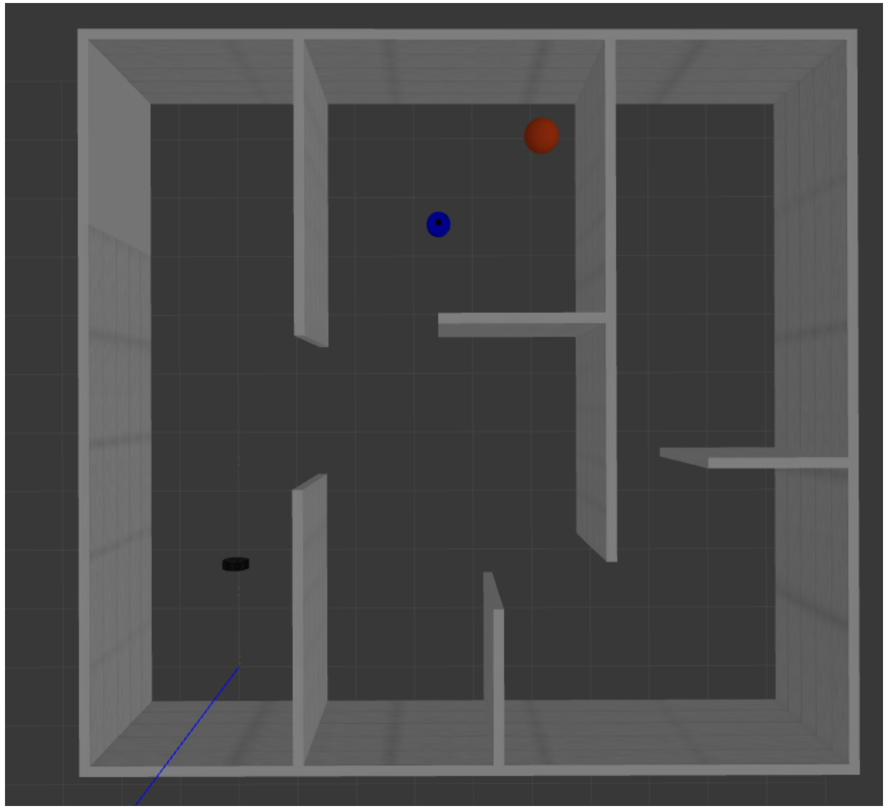
3.“迷宫寻宝”之任务实现 —— 系统集成

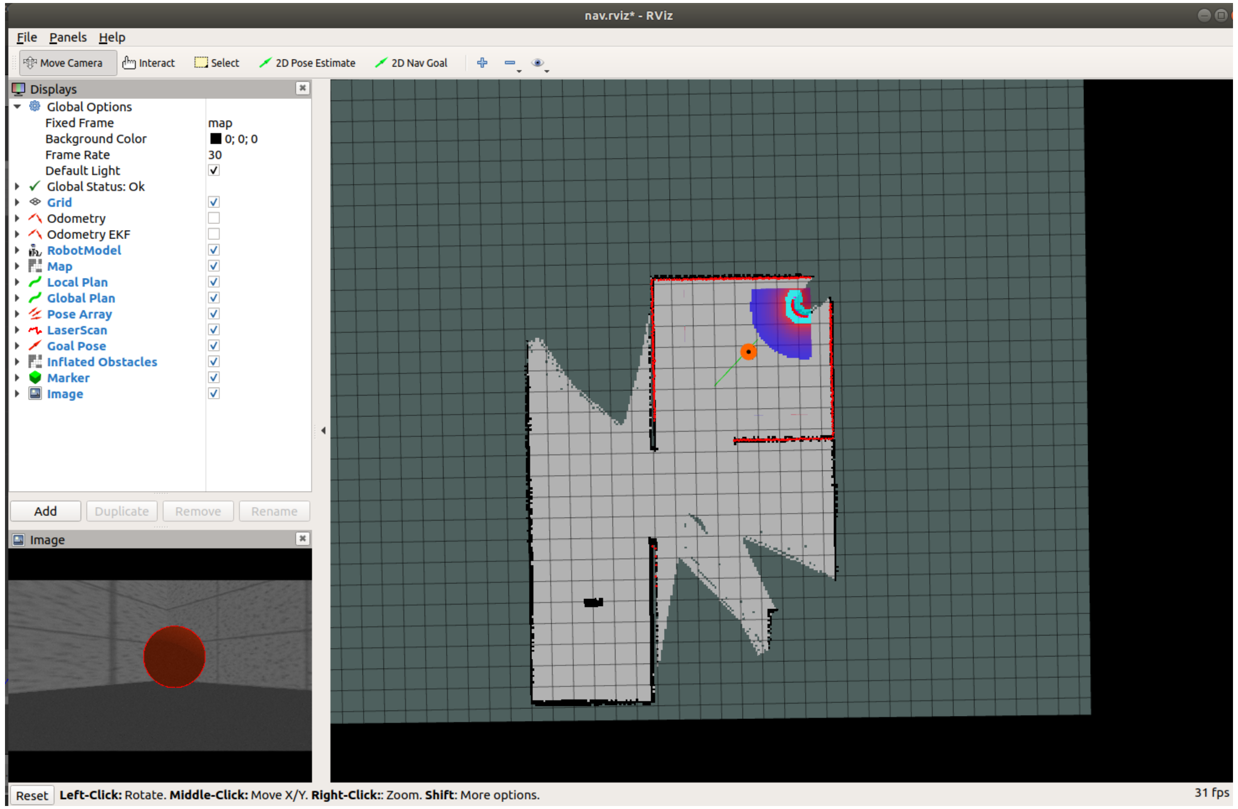


“迷宫寻宝”实现框架

3. “迷宫寻宝”之任务实现 —— 系统运行

```
$ roslaunch mbot_gazebo mbot_maze_gazebo.launch
$ roslaunch mbot_navigation exploring_slam_demo.launch
$ roslaunch mbot_vision find_target.launch
```



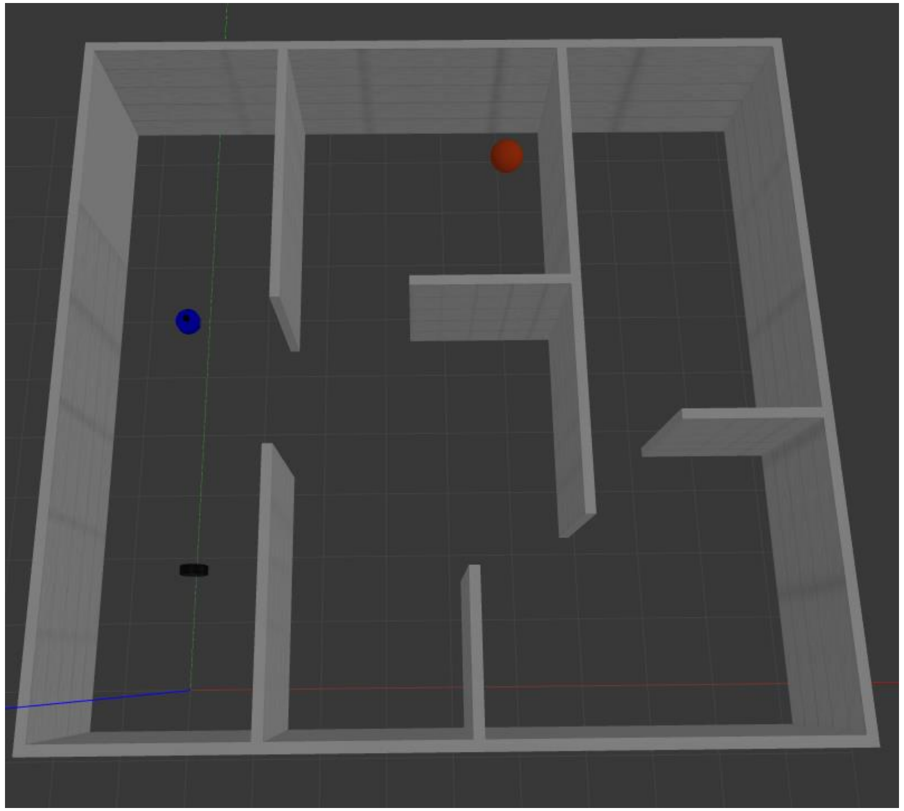


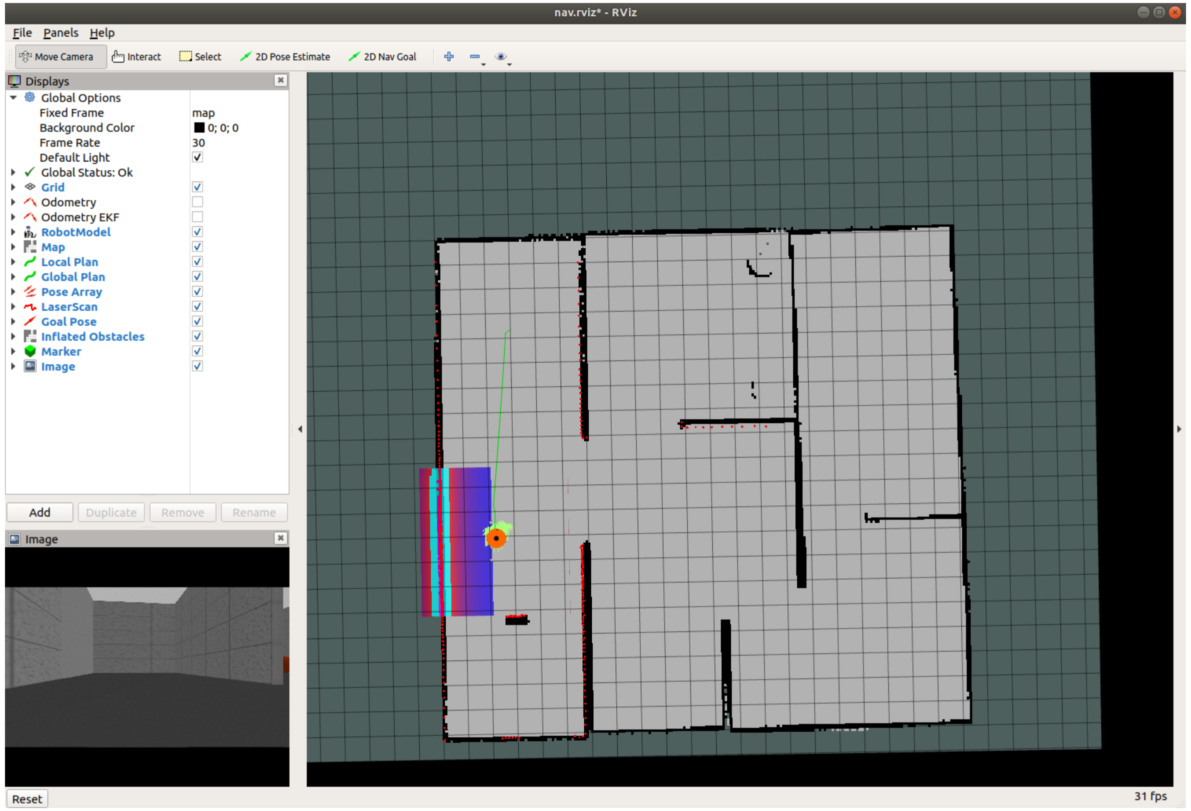
Global Options
Fixed Frame map
Background Color ■ 0; 0; 0
Frame Rate 30
Default Light ☑
✓ Global Status: Ok
Grid ☑
Odometry □
Odometry EKF □
RobotModel ☑
Map ☑
Local Plan ☑
Global Plan ☑
Pose Array ☑
LaserScan ☑
Goal Pose ☑
Inflated Obstacles ☑
Marker ☑
Image ☑

Left-Click: Rotate. Middle-Click: Move X/Y. Right-Click:: Zoom. Shift: More options.

3.“迷宫寻宝”之任务实现 —— 系统运行

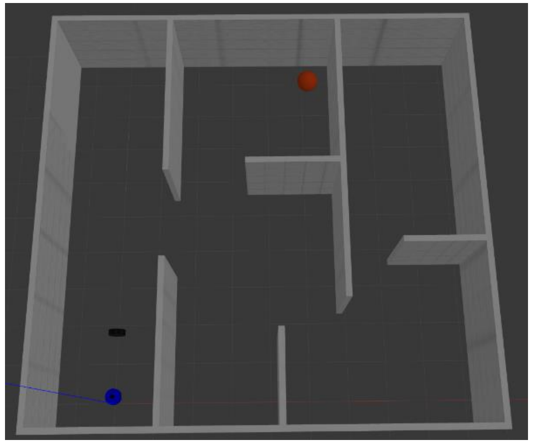
```
$ roslaunch mbot_gazebo mbot_maze_gazebo.launch
$ roslaunch mbot_navigation nav_maze_demo.launch
$ roslaunch mbot_vision find_target_pro.launch
```



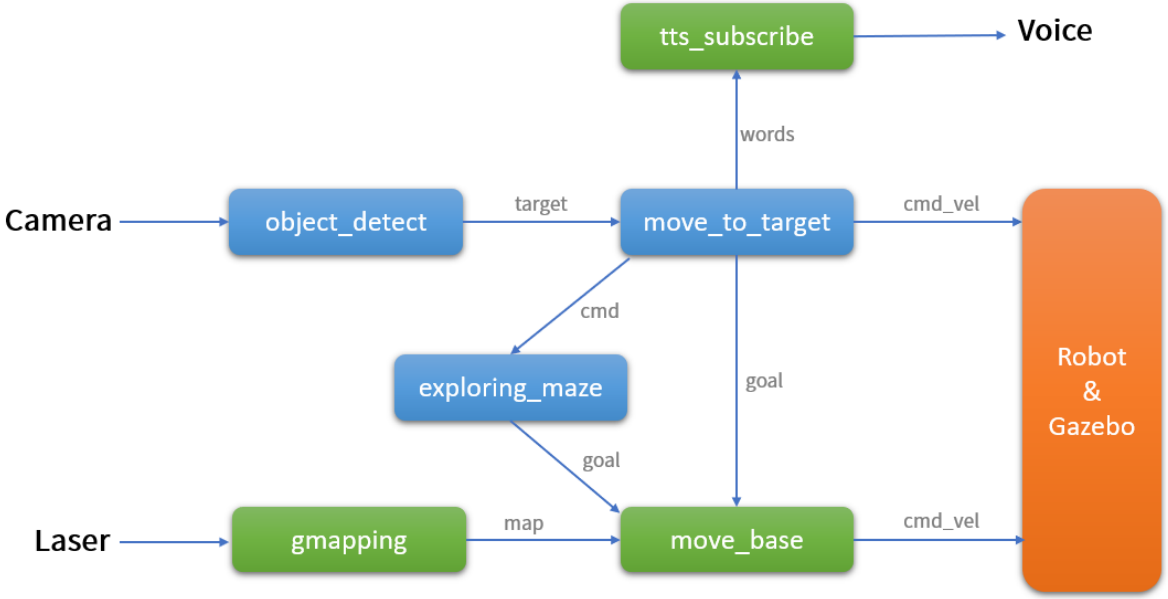


Global Options
Fixed Frame map
Background Color ■ 0; 0; 0
Frame Rate 30
Default Light ☑
Global Status: Ok
Grid ☑
Odometry ☐
Odometry EKF ☐
RobotModel ☑
Map ☑
Local Plan ☑
Global Plan ☑
Pose Array ☑
LaserScan ☑
Goal Pose ☑
Inflated Obstacles ☑
Marker ☑
Image ☑

本讲小结



- 机器人建模仿真
- 图像识别
- 语音交互
- 自主导航
- SLAM
- ROS通信机制
- 系统设计与集成



本讲作业

1、下载本讲作业代码，按照课程内容及代码提示，填充代码中缺少的部分，并复现课程中的“迷宫寻宝”效果；

（需要修改的文件：exploring_maze.py、exploring_maze_pro.py、object_detect.py、move_to_target.cpp、find_target.launch、find_target_pro.launch）

2、接受“迷宫寻宝”挑战，仿照本讲例程源码，使用自己的机器人模型及仿真环境，完成“迷宫寻宝”任务，挑战自己的时间极限。

Thank You

怕什么真理无穷，进一寸有一寸的欢喜

更多精彩，欢迎关注

古月居

古月春旭