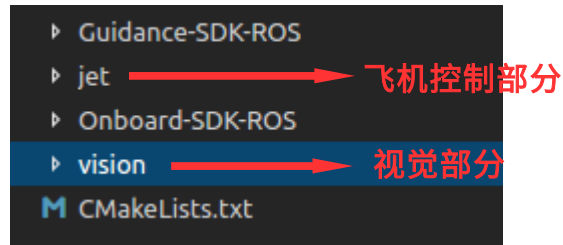


# 飞机控制逻辑说明文档

## 1.1 项目结构

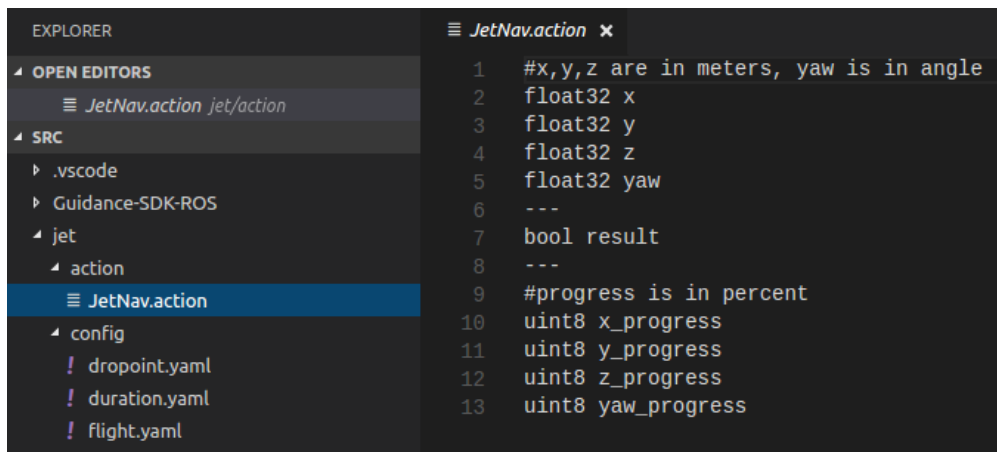
### 1.1.1 项目总体结构



### 1.1.2 飞机控制部分项目结构



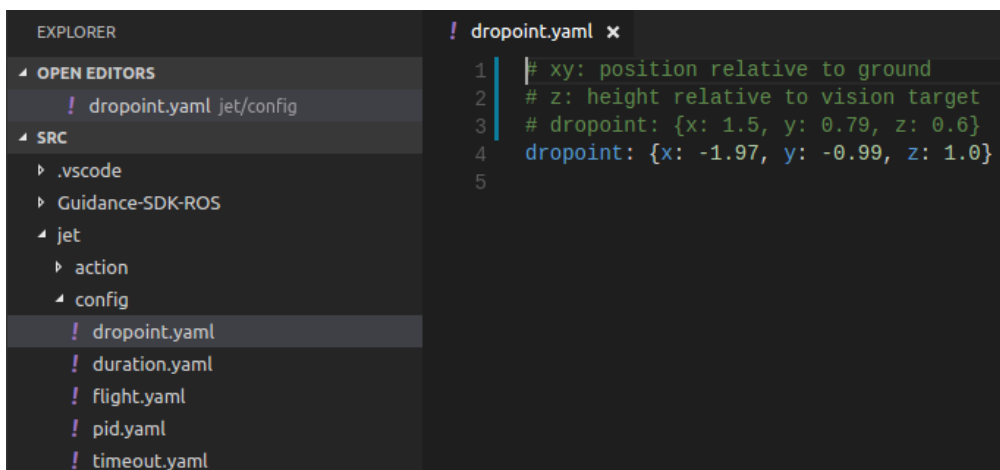
jet/action/JetNav.action: actionlib 导航



```
EXPLORER
└─ OPEN EDITORS
    └─ JetNav.action jet/action
└─ SRC
    ├── .vscode
    ├── Guidance-SDK-ROS
    └─ jet
        ├── action
        └─ JetNav.action
            ├── config
            ├── dropoint.yaml
            ├── duration.yaml
            └── flight.yaml

JetNav.action x
1 #x,y,z are in meters, yaw is in angle
2 float32 x
3 float32 y
4 float32 z
5 float32 yaw
6 ---
7 bool result
8 ---
9 #progress is in percent
10 uint8 x_progress
11 uint8 y_progress
12 uint8 z_progress
13 uint8 yaw_progress
```

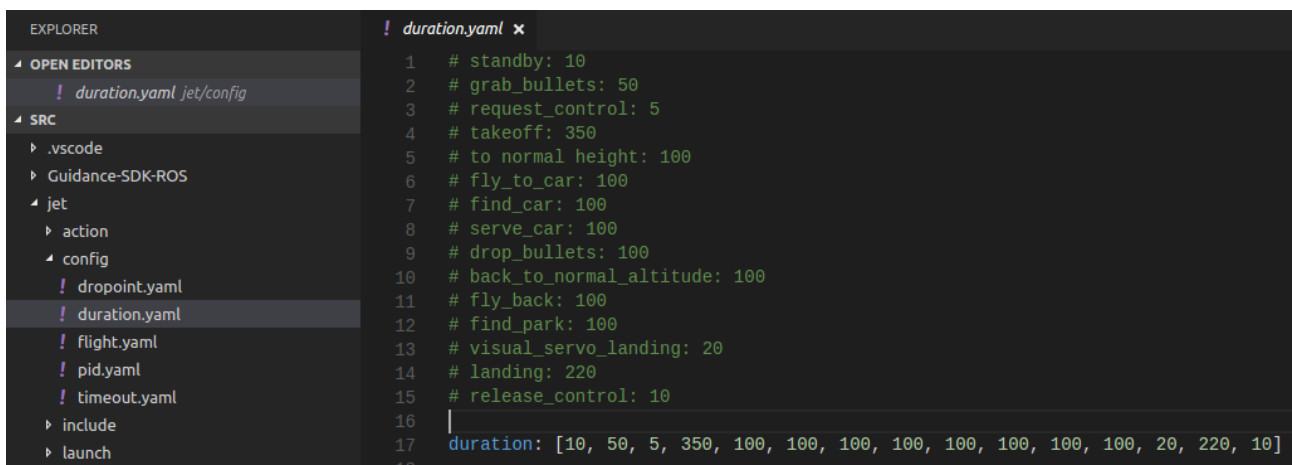
jet/config/dropoint.yaml: 空投点配置



```
EXPLORER
└─ OPEN EDITORS
    └─ dropoint.yaml jet/config
└─ SRC
    ├── .vscode
    ├── Guidance-SDK-ROS
    └─ jet
        ├── action
        ├── config
        ├── dropoint.yaml
        ├── duration.yaml
        ├── flight.yaml
        ├── pid.yaml
        └── timeout.yaml

! dropoint.yaml x
1 # xy: position relative to ground
2 # z: height relative to vision target
3 # dropoint: {x: 1.5, y: 0.79, z: 0.6}
4 dropoint: {x: -1.97, y: -0.99, z: 1.0}
5
```

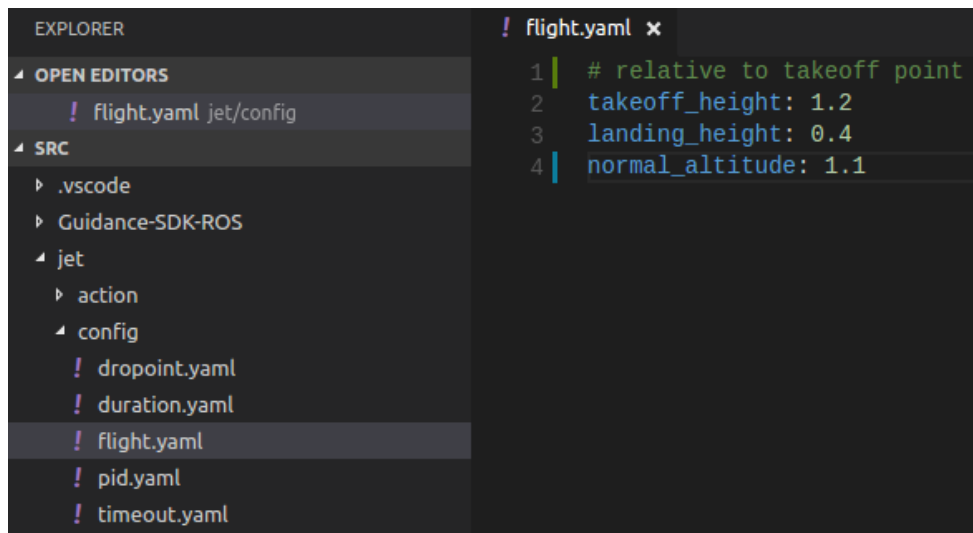
jet/config/duration.yaml: 过渡时间配置



```
EXPLORER
└─ OPEN EDITORS
    └─ duration.yaml jet/config
└─ SRC
    ├── .vscode
    ├── Guidance-SDK-ROS
    └─ jet
        ├── action
        ├── config
        ├── dropoint.yaml
        ├── duration.yaml
        ├── flight.yaml
        ├── pid.yaml
        ├── timeout.yaml
        ├── include
        └── launch

! duration.yaml x
1 # standby: 10
2 # grab_bullets: 50
3 # request_control: 5
4 # takeoff: 350
5 # to normal height: 100
6 # fly_to_car: 100
7 # find_car: 100
8 # serve_car: 100
9 # drop_bullets: 100
10 # back_to_normal_altitude: 100
11 # fly_back: 100
12 # find_park: 100
13 # visual_servo_landing: 20
14 # landing: 220
15 # release_control: 10
16 |
17 duration: [10, 50, 5, 350, 100, 100, 100, 100, 100, 100, 100, 100, 20, 220, 10]
18
```

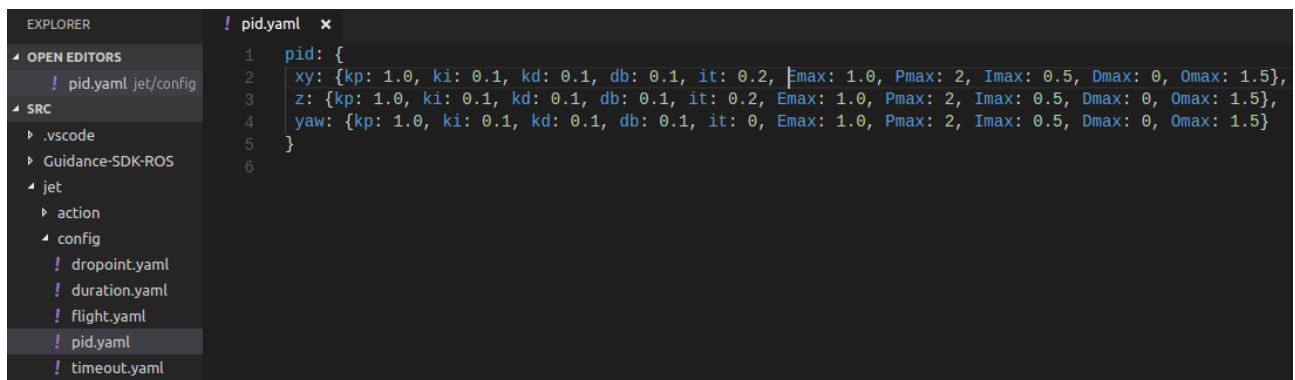
jet/config/flight.yaml: 飞行参数配置



The screenshot shows the VS Code interface. On the left, the Explorer sidebar is open, showing the project structure. The 'jet' folder is expanded, and the 'config' subfolder is selected. The 'flight.yaml' file is highlighted. On the right, the editor shows the content of 'flight.yaml'.

```
! flight.yaml x
1 | # relative to takeoff point
2 | takeoff_height: 1.2
3 | landing_height: 0.4
4 | normal_altitude: 1.1
```

jet/config/pid.yaml: PID 闭环控制参数配置



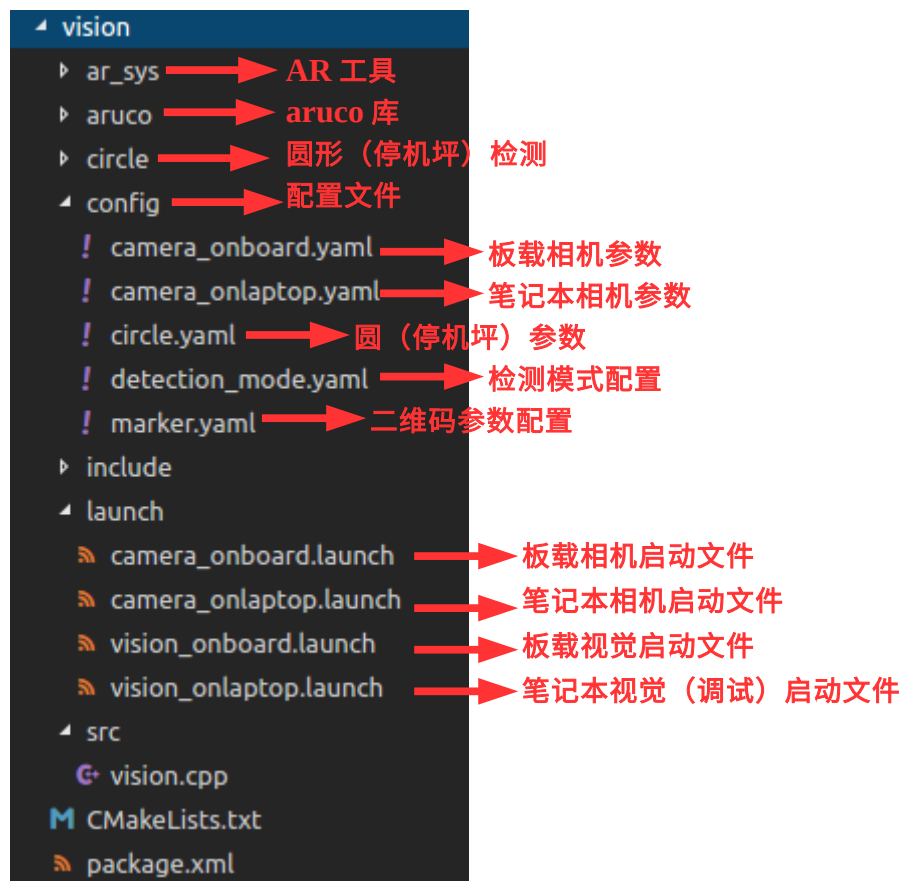
The screenshot shows the VS Code interface. On the left, the Explorer sidebar is open, showing the project structure. The 'jet' folder is expanded, and the 'config' subfolder is selected. The 'pid.yaml' file is highlighted. On the right, the editor shows the content of 'pid.yaml'.

```
! pid.yaml x
1 | pid: {
2 |   xy: {kp: 1.0, ki: 0.1, kd: 0.1, db: 0.1, it: 0.2, Emax: 1.0, Pmax: 2, Imax: 0.5, Dmax: 0, Omax: 1.5},
3 |   z: {kp: 1.0, ki: 0.1, kd: 0.1, db: 0.1, it: 0.2, Emax: 1.0, Pmax: 2, Imax: 0.5, Dmax: 0, Omax: 1.5},
4 |   yaw: {kp: 1.0, ki: 0.1, kd: 0.1, db: 0.1, it: 0, Emax: 1.0, Pmax: 2, Imax: 0.5, Dmax: 0, Omax: 1.5}
5 | }
6 |
```

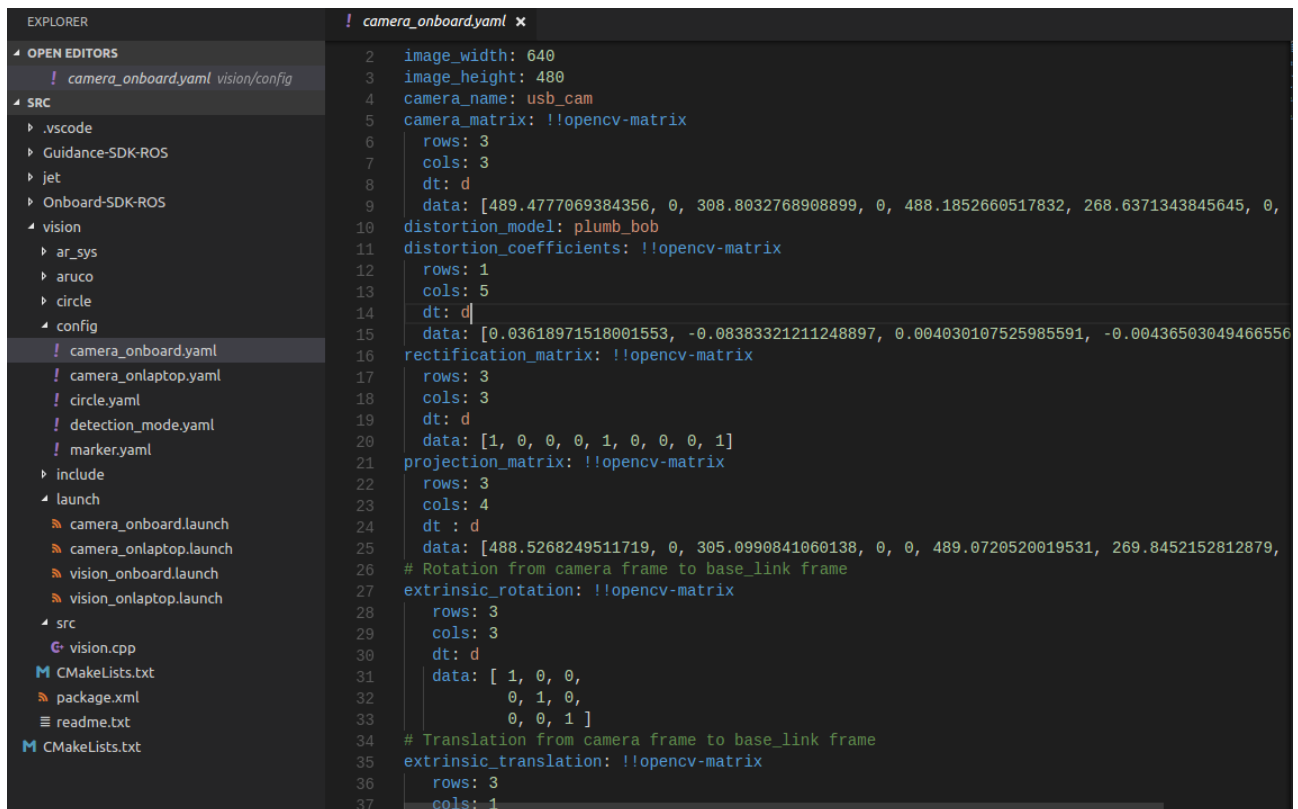
jet/config/timeout.yaml: 超时配置

```
EXPLORER
! timeout.yaml x
1 odom_callback_timeout: 60 # control loop will be cutoff when odometry latency is higher than this
2 vision_callback_timeout: 100 # control loop will be cutoff when vision latency is higher than this
```

### 1.1.3 视觉部分项目结构



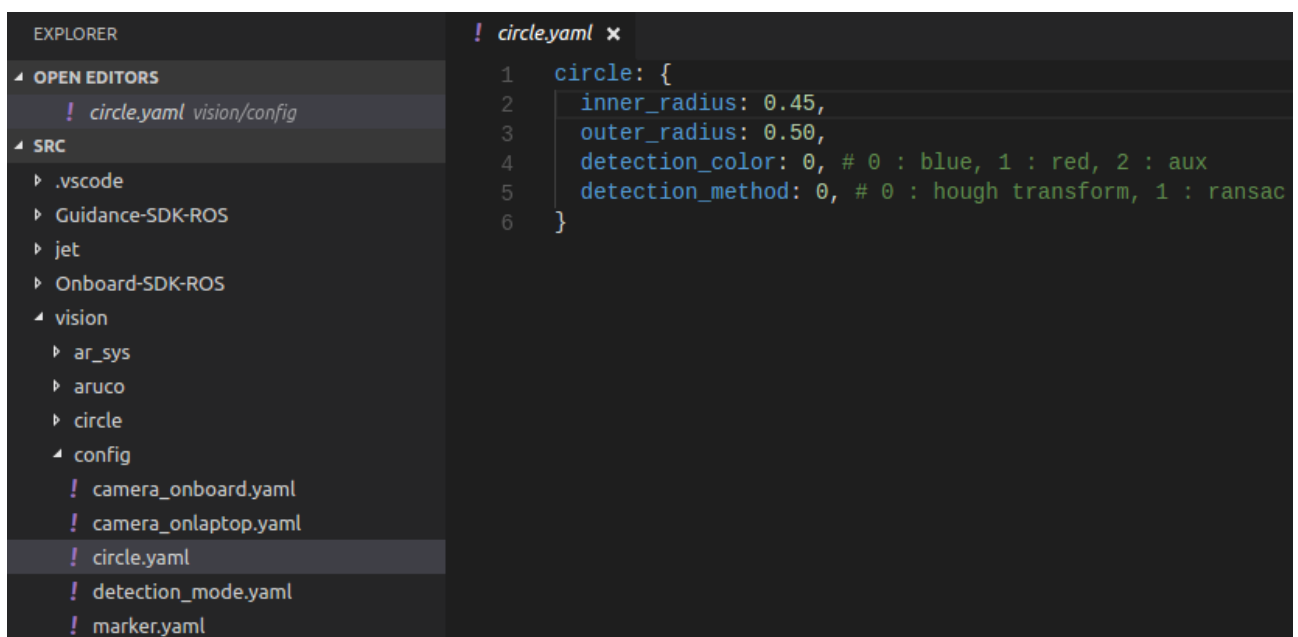
vision/config/camera\_\*.yaml: 相机参数配置



The screenshot shows the VS Code interface with the Explorer on the left and the Editor on the right. The Explorer shows the project structure with folders like .vscode, Guidance-SDK-ROS, jet, Onboard-SDK-ROS, vision, ar\_sys, aruco, circle, config, include, launch, src, and CMakeLists.txt. The Editor displays the content of camera\_onboard.yaml, which defines camera parameters for a USB camera. The configuration includes image dimensions, camera name, camera matrix, distortion model and coefficients, rectification matrix, projection matrix, extrinsic rotation, and extrinsic translation.

```
2 image_width: 640
3 image_height: 480
4 camera_name: usb_cam
5 camera_matrix: !!opencv-matrix
6   rows: 3
7   cols: 3
8   dt: d
9   data: [489.4777069384356, 0, 308.8032768908899, 0, 488.1852660517832, 268.6371343845645, 0,
10 distortion_model: plumb_bob
11 distortion_coefficients: !!opencv-matrix
12   rows: 1
13   cols: 5
14   dt: d
15   data: [0.03618971518001553, -0.08383321211248897, 0.004030107525985591, -0.00436503049466556, 0]
16 rectification_matrix: !!opencv-matrix
17   rows: 3
18   cols: 3
19   dt: d
20   data: [1, 0, 0, 0, 1, 0, 0, 0, 1]
21 projection_matrix: !!opencv-matrix
22   rows: 3
23   cols: 4
24   dt: d
25   data: [488.5268249511719, 0, 305.0990841060138, 0, 0, 489.0720520019531, 269.8452152812879,
26 # Rotation from camera frame to base_link frame
27 extrinsic_rotation: !!opencv-matrix
28   rows: 3
29   cols: 3
30   dt: d
31   data: [ 1, 0, 0,
32           0, 1, 0,
33           0, 0, 1 ]
34 # Translation from camera frame to base_link frame
35 extrinsic_translation: !!opencv-matrix
36   rows: 3
37   cols: 1
```

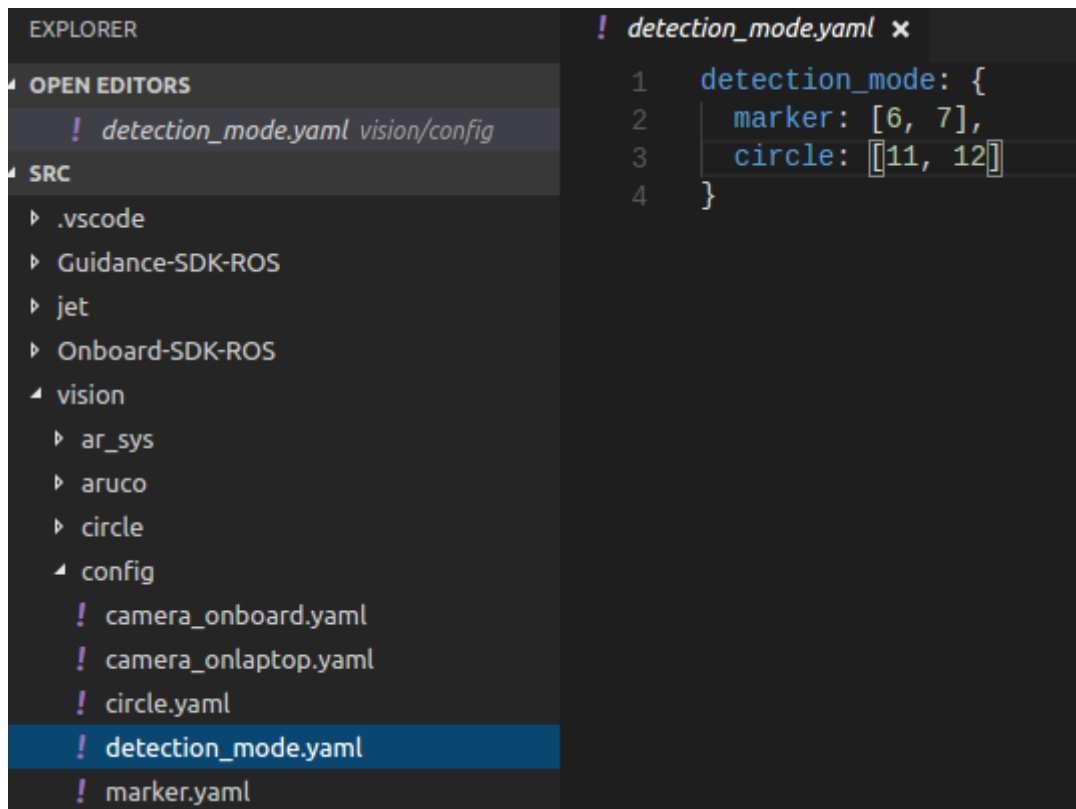
vision/config/circle.yaml: 圆（停机坪）参数配置



The screenshot shows the VS Code interface with the Explorer on the left and the Editor on the right. The Explorer shows the project structure with folders like .vscode, Guidance-SDK-ROS, jet, Onboard-SDK-ROS, vision, ar\_sys, aruco, circle, config, include, launch, src, and CMakeLists.txt. The Editor displays the content of circle.yaml, which defines parameters for a circle detection algorithm, including inner and outer radii, detection color, and detection method.

```
1 circle: {
2   inner_radius: 0.45,
3   outer_radius: 0.50,
4   detection_color: 0, # 0 : blue, 1 : red, 2 : aux
5   detection_method: 0, # 0 : hough transform, 1 : ransac
6 }
```

vision/config/detection\_mode.yaml: 检测模式配置



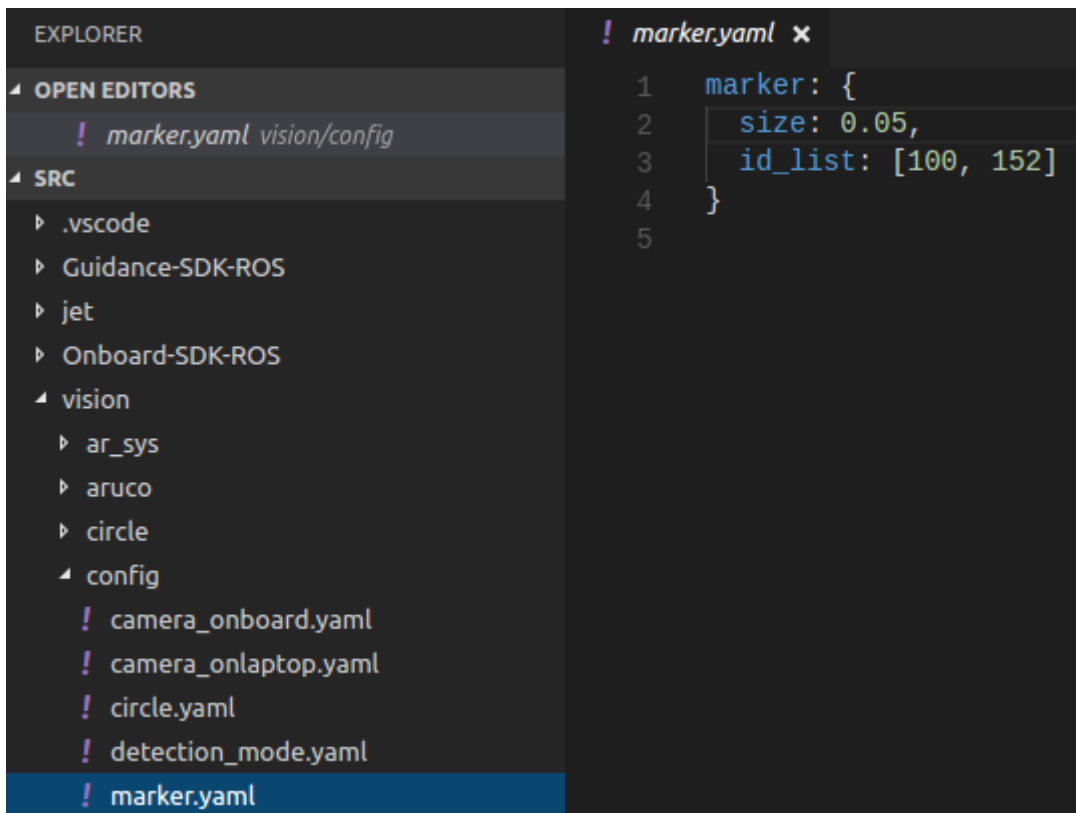
The screenshot shows the VS Code interface with the Explorer sidebar on the left and the Editor on the right. The Explorer sidebar has a tree view with the following structure:

- EXPLORER
  - OPEN EDITORS
    - ! detection\_mode.yaml vision/config
  - SRC
    - .vscode
    - Guidance-SDK-ROS
    - jet
    - Onboard-SDK-ROS
    - vision
      - ar\_sys
      - aruco
      - circle
      - config
        - ! camera\_onboard.yaml
        - ! camera\_onlaptop.yaml
        - ! circle.yaml
        - ! detection\_mode.yaml
        - ! marker.yaml

The Editor on the right shows the content of the `detection_mode.yaml` file:

```
1 detection_mode: {  
2   marker: [6, 7],  
3   circle: [[11, 12]]  
4 }
```

vision/config/marker.yaml: 二维码参数配置



The screenshot shows the VS Code interface with the Explorer sidebar on the left and the Editor on the right. The Explorer sidebar has a tree view with the following structure:

- EXPLORER
  - OPEN EDITORS
    - ! marker.yaml vision/config
  - SRC
    - .vscode
    - Guidance-SDK-ROS
    - jet
    - Onboard-SDK-ROS
    - vision
      - ar\_sys
      - aruco
      - circle
      - config
        - ! camera\_onboard.yaml
        - ! camera\_onlaptop.yaml
        - ! circle.yaml
        - ! detection\_mode.yaml
        - ! marker.yaml

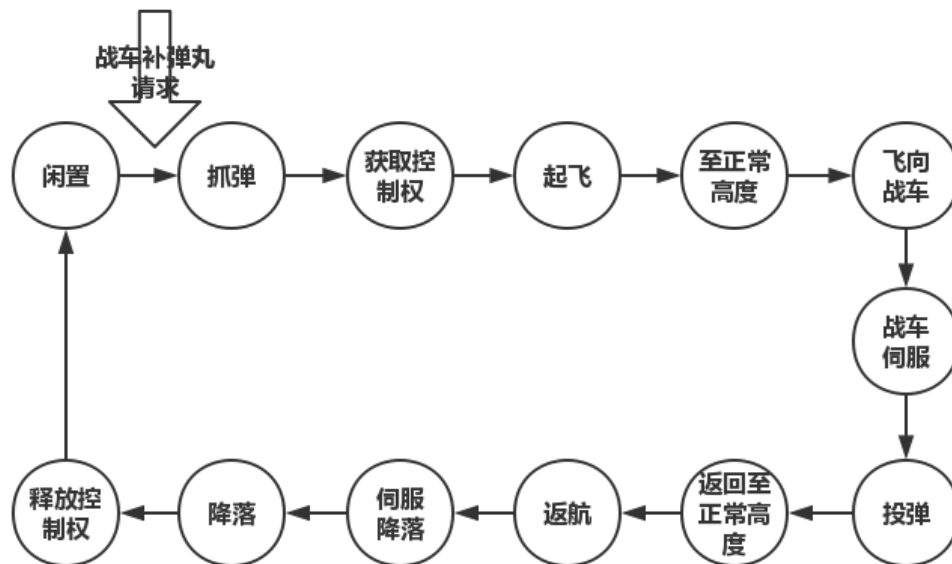
The Editor on the right shows the content of the `marker.yaml` file:

```
1 marker: {  
2   size: 0.05,  
3   id_list: [100, 152]  
4 }  
5
```

> 以上所有配置文件均可在运行时动态加载（调用相关的 service），调参十分方便。

## 1.2 状态机

### 1.2.1 主逻辑



### 1.2.2 状态/命令类型枚举

```
typedef enum
{
    STAND_BY,
    GRAB_BULLETS,
    REQUEST_CONTROL,
    TAKE_OFF,
    TO_NORMAL_ALTITUDE,
    FLY_TO_CAR,
    FIND_CAR,
    SERVE_CAR,
    DROP_BULLETS,
    BACK_TO_NORMAL_ALTITUDE,
    FLY_BACK,
    FIND_PARK,
    VISUAL_SERVO_LANDING,
    LANDING,
    RELEASE_CONTROL,
} JetCmd_e;
```

### 1.2.3 状态机执行流程

```

void Jet::stateMachine()
{
    static bool success = false;
    static uint32_t tick = 0;
    switch (jet_state)
    {
        case STAND_BY:
            if (!success)
            {
                success = doStandby();
                std::cout << "stateMachine: " << "Standby" << std::endl;
            }
            else if (tick < duration[STAND_BY])
            {
                tick++;
                std::cout << "stateMachine: " << "Standby@Tick: " << tick <<
std::endl;
            }
            else
            {
                tick = 0;
                success = false;
                jet_state = GRAB_BULLETS;
                std::cout << "stateMachine: " << "Standby->Grab Bullets" << tick <<
std::endl;
            }
            break;

        case GRAB_BULLETS:
            if (!success)
            {
                success = doGrabBullets();
                std::cout << "stateMachine: " << "Grab Bullets" << std::endl;
            }
            else if (tick < duration[GRAB_BULLETS])
            {
                tick++;
                std::cout << "stateMachine: " << "Grab Bullets@Tick: " << tick <<
std::endl;
            }
            else
            {
                tick = 0;
                success = false;
                jet_state = REQUEST_CONTROL;
                std::cout << "stateMachine: " << "Grab Bullets->Request Control" <<
tick << std::endl;
            }
            break;

        case REQUEST_CONTROL:
            if (!success)
            {
                success = doRequestControl();
                std::cout << "stateMachine: " << "Request Control" << std::endl;
            }
            else if (tick < duration[REQUEST_CONTROL])
            {
                tick++;
                std::cout << "stateMachine: " << "Request Control@Tick: " << tick
<< std::endl;
            }
            else
    
```



```

        {
            tick = 0;
            success = false;
            jet_state = TAKE_OFF;
            std::cout << "stateMachine: " << "Request Control->Takeoff" << tick
<< std::endl;
        }
        break;

        case TAKE_OFF:
            if (!success)
            {
                success = doTakeoff();
                std::cout << "stateMachine: " << "Takeoff" << std::endl;
            }
            else if (tick < duration[TAKE_OFF])
            {
                tick++;
                std::cout << "stateMachine: " << "Takeoff@Tick: " << tick <<
std::endl;
            }
            else
            {
                tick = 0;
                success = false;
                jet_state = TO_NORMAL_ALTITUDE;
                std::cout << "stateMachine: " << "Takeoff->To Normal Altitude" <<
tick << std::endl;
            }
            break;

            case TO_NORMAL_ALTITUDE:
                if (!success)
                {
                    success = doToNormalAltitude();
                    std::cout << "stateMachine: " << "To Normal Altitude" << std::endl;
                }
                else if (tick < duration[TO_NORMAL_ALTITUDE])
                {
                    tick++;
                    std::cout << "stateMachine: " << "To Normal Altitude@Tick: " <<
tick << std::endl;
                }
                else
                {
                    tick = 0;
                    success = false;
                    jet_state = FLY_TO_CAR;
                    std::cout << "stateMachine: " << "To Normal Altitude->Fly to Car"
<< tick << std::endl;
                }
                break;

                case FLY_TO_CAR:
                    if (!success)
                    {
                        success = doFlyToCar();
                        std::cout << "stateMachine: " << "Fly to Car" << std::endl;
                    }
                    else if (tick < duration[FLY_TO_CAR])
                    {
                        tick++;
                        std::cout << "stateMachine: " << "Fly to Car@Tick: " << tick <<

```

```

std::endl;
    }
    else
    {
        tick = 0;
        success = false;
        jet_state = FIND_CAR;
        std::cout << "stateMachine: " << "Fly to Car->Find Car" << tick <<
std::endl;
    }
    break;

    case FIND_CAR:
    if (!success)
    {
        success = doFindCar();
        std::cout << "stateMachine: " << "Find Car" << std::endl;
    }
    else if (tick < duration[FIND_CAR])
    {
        tick++;
        std::cout << "stateMachine: " << "Find Car@Tick: " << tick <<
std::endl;
    }
    else
    {
        tick = 0;
        success = false;
        jet_state = SERVE_CAR;
        std::cout << "stateMachine: " << "Find Car->Serve Car" << tick <<
std::endl;
    }
    break;

    case SERVE_CAR:
    if (!success)
    {
        success = doServeCar();
        std::cout << "stateMachine: " << "Serve Car" << std::endl;
    }
    else if (tick < duration[SERVE_CAR])
    {
        tick++;
        std::cout << "stateMachine: " << "Serve Car@Tick: " << tick <<
std::endl;
    }
    else
    {
        tick = 0;
        success = false;
        jet_state = DROP_BULLETS;
        std::cout << "stateMachine: " << "Serve Car->Drop Bullets" << tick
<< std::endl;
    }
    break;

    case DROP_BULLETS:
    if (!success)
    {
        success = doDropBullets();
        std::cout << "stateMachine: " << "Drop Bullets" << std::endl;
    }
    else if (tick < duration[DROP_BULLETS])

```

```

{
    tick++;
    std::cout << "stateMachine: " << "Drop Bullets@Tick: " << tick <<
std::endl;
}
else
{
    tick = 0;
    success = false;
    jet_state = BACK_TO_NORMAL_ALTITUDE;
    std::cout << "stateMachine: " << "Drop Bullets->Back to Normal
Altitude" << tick << std::endl;
}
break;

case BACK_TO_NORMAL_ALTITUDE:
if (!success)
{
    success = doBackToNormalAltitude();
    std::cout << "stateMachine: " << "Back to Normal Altitude" <<
std::endl;
}
else if (tick < duration[BACK_TO_NORMAL_ALTITUDE])
{
    tick++;
    std::cout << "stateMachine: " << "Back to Normal Altitude@Tick: "
<< tick << std::endl;
}
else
{
    tick = 0;
    success = false;
    jet_state = FLY_BACK;
    std::cout << "stateMachine: " << "Back to Normal Altitude->Fly
Back" << tick << std::endl;
}
break;

case FLY_BACK:
if (!success)
{
    success = doFlyBack();
    std::cout << "stateMachine: " << "Fly Back" << std::endl;
}
else if (tick < duration[FLY_BACK])
{
    tick++;
    std::cout << "stateMachine: " << "Fly Back@Tick: " << tick <<
std::endl;
}
else
{
    tick = 0;
    success = false;
    jet_state = FIND_PARK;
    std::cout << "stateMachine: " << "Fly Back->Find Park" << tick <<
std::endl;
}
break;

case FIND_PARK:
if (!success)
{

```

```

        success = doFindPark();
        std::cout << "stateMachine: " << "Find Park" << std::endl;
    }
    else if (tick < duration[FIND_PARK])
    {
        tick++;
        std::cout << "stateMachine: " << "Find Park@Tick: " << tick <<
std::endl;
    }
    else
    {
        tick = 0;
        success = false;
        jet_state = VISUAL_SERVO_LANDING;
        std::cout << "stateMachine: " << "Find Park->Visual Servo Landing"
<< tick << std::endl;
    }
    break;

    case VISUAL_SERVO_LANDING:
    if (!success)
    {
        success = doVisualServoLanding();
        std::cout << "stateMachine: " << "Visual Servo Landing" <<
std::endl;
    }
    else if (tick < duration[VISUAL_SERVO_LANDING])
    {
        tick++;
        std::cout << "stateMachine: " << "Visual Servo Landing@Tick: " <<
tick << std::endl;
    }
    else
    {
        tick = 0;
        success = false;
        jet_state = LANDING;
        std::cout << "stateMachine: " << "Visual Servo Landing->Landing" <<
tick << std::endl;
    }
    break;

    case LANDING:
    if (!success)
    {
        success = doLanding();
        std::cout << "stateMachine: " << "Landing" << std::endl;
    }
    else if (tick < duration[LANDING])
    {
        tick++;
        std::cout << "stateMachine: " << "Landing@Tick: " << tick <<
std::endl;
    }
    else
    {
        tick = 0;
        success = false;
        jet_state = RELEASE_CONTROL;
        std::cout << "stateMachine: " << "Landing->Standby" << tick <<
std::endl;

        calied = false; // re-calibrate odom

```

```

    }
    break;

    case RELEASE_CONTROL:
    if (!success)
    {
        success = doReleaseControl();
        std::cout << "stateMachine: " << "Release Control" << std::endl;
    }
    else if (tick < duration[RELEASE_CONTROL])
    {
        tick++;
        std::cout << "stateMachine: " << "Release Control@Tick: " << tick
<< std::endl;
    }
    else
    {
        tick = 0;
        success = false;
        jet_state = STAND_BY;
        std::cout << "stateMachine: " << "Release Control->Standby" << tick
<< std::endl;
        if (freestyle)
        {
            freestyle = false; // clear freestyle flag
            std::cout << "+-----Jetbang Free Style
Done-----+" << std::endl;
            help();
        }
    }
    break;

    default:
    jet_state = STAND_BY;
    break;
}
}
}

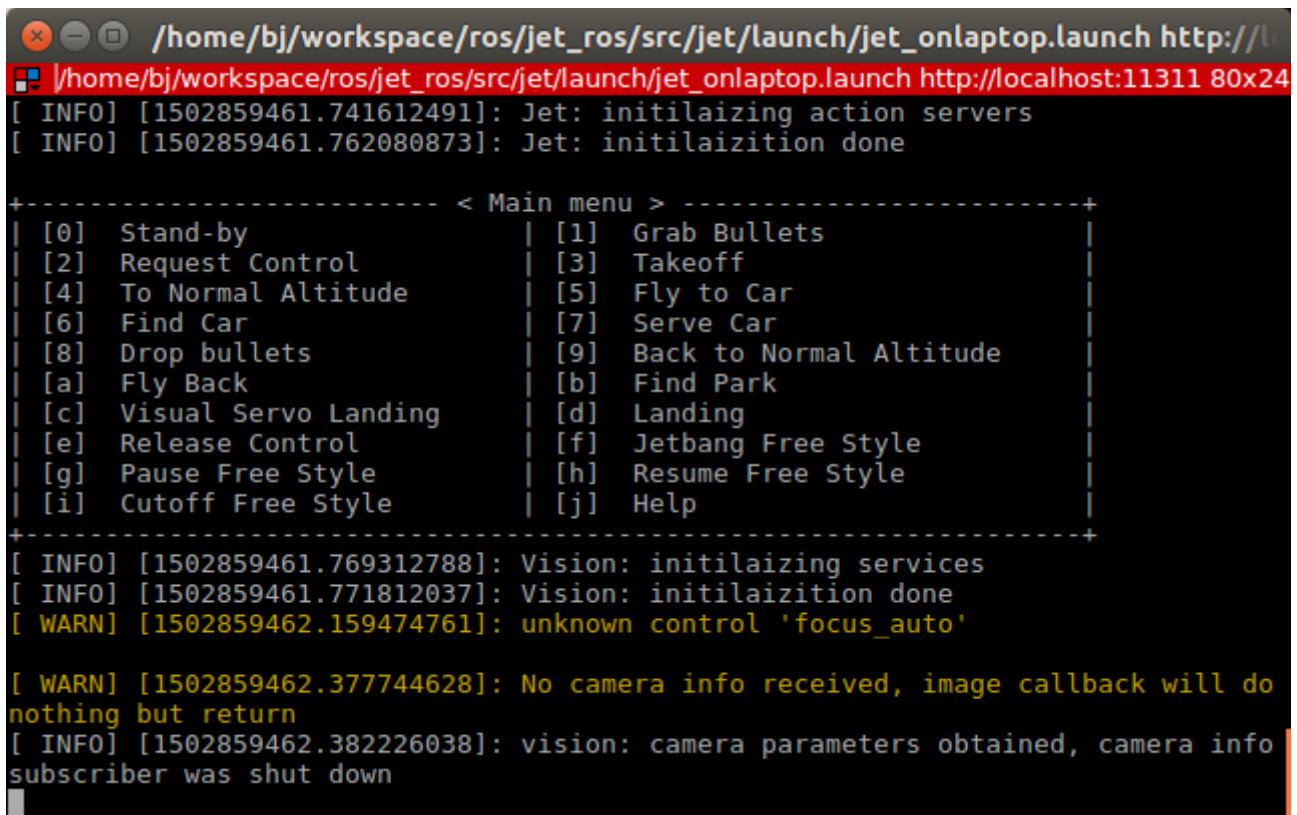
```

## 1.4 启动

### 1.4.1 仿真

- 1) 首先准备两台电脑，一台 windows，一台 ubuntu，windows 用来跑 DJI Simulator，ubuntu 跑控制程序；
- 2) 用 mini-USB 线连接 windows 电脑和飞控，用 USB 转串口线连接 ubuntu 电脑和飞机 UART-CAN2，windows 电脑上打开 DJI Assistant，启用 API 控制，打开模拟器；下面在 ubuntu 电脑上操作：
- 3) cd 到 jet\_ros 文件夹，在此文件夹下打开两个 terminal，分别 source devel/setup.bash 之后，在一个 terminal 运行 roslaunch dji\_sdk sdk\_linux.launch，在另一个 terminal 运行 roslaunch jet

jet\_onlaptop.launch; 在运行后者的窗口出现以下界面，可进行单元测试，也可运行全自动测试（Jetbang Free Style），如想从某一项目开始测试，可先进入该项目后选择[h] Resume Free Style. 自动运行过程中可中断（[g] Pause Free Style），也可停止（[i] Cutoff Free Style）。



```
/home/bj/workspace/ros/jet_ros/src/jet/launch/jet_onlaptop.launch http://localhost:11311 80x24
[ INFO] [1502859461.741612491]: Jet: initilaizing action servers
[ INFO] [1502859461.762080873]: Jet: initilaizition done

+----- < Main menu > -----+
| [0] Stand-by                  | [1] Grab Bullets              |
| [2] Request Control          | [3] Takeoff                   |
| [4] To Normal Altitude       | [5] Fly to Car                |
| [6] Find Car                 | [7] Serve Car                 |
| [8] Drop bullets             | [9] Back to Normal Altitude  |
| [a] Fly Back                 | [b] Find Park                 |
| [c] Visual Servo Landing     | [d] Landing                   |
| [e] Release Control          | [f] Jetbang Free Style        |
| [g] Pause Free Style         | [h] Resume Free Style         |
| [i] Cutoff Free Style        | [j] Help                      |
+-----+

[ INFO] [1502859461.769312788]: Vision: initilaizing services
[ INFO] [1502859461.771812037]: Vision: initilaizition done
[ WARN] [1502859462.159474761]: unknown control 'focus_auto'

[ WARN] [1502859462.377744628]: No camera info received, image callback will do
nothing but return
[ INFO] [1502859462.382226038]: vision: camera parameters obtained, camera info
subscriber was shut down
```

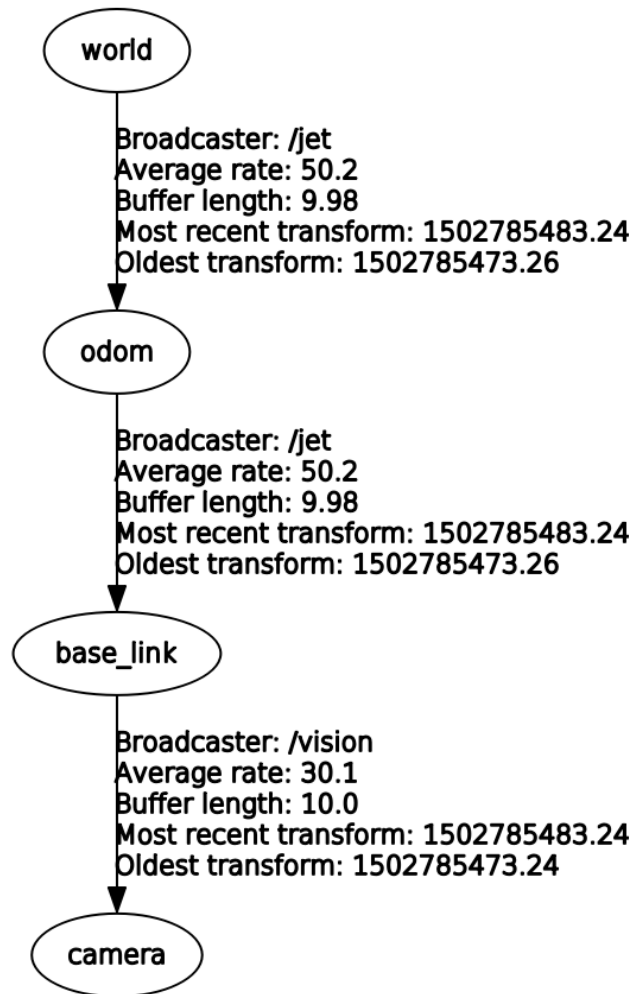
## 1.4.2 真机运行

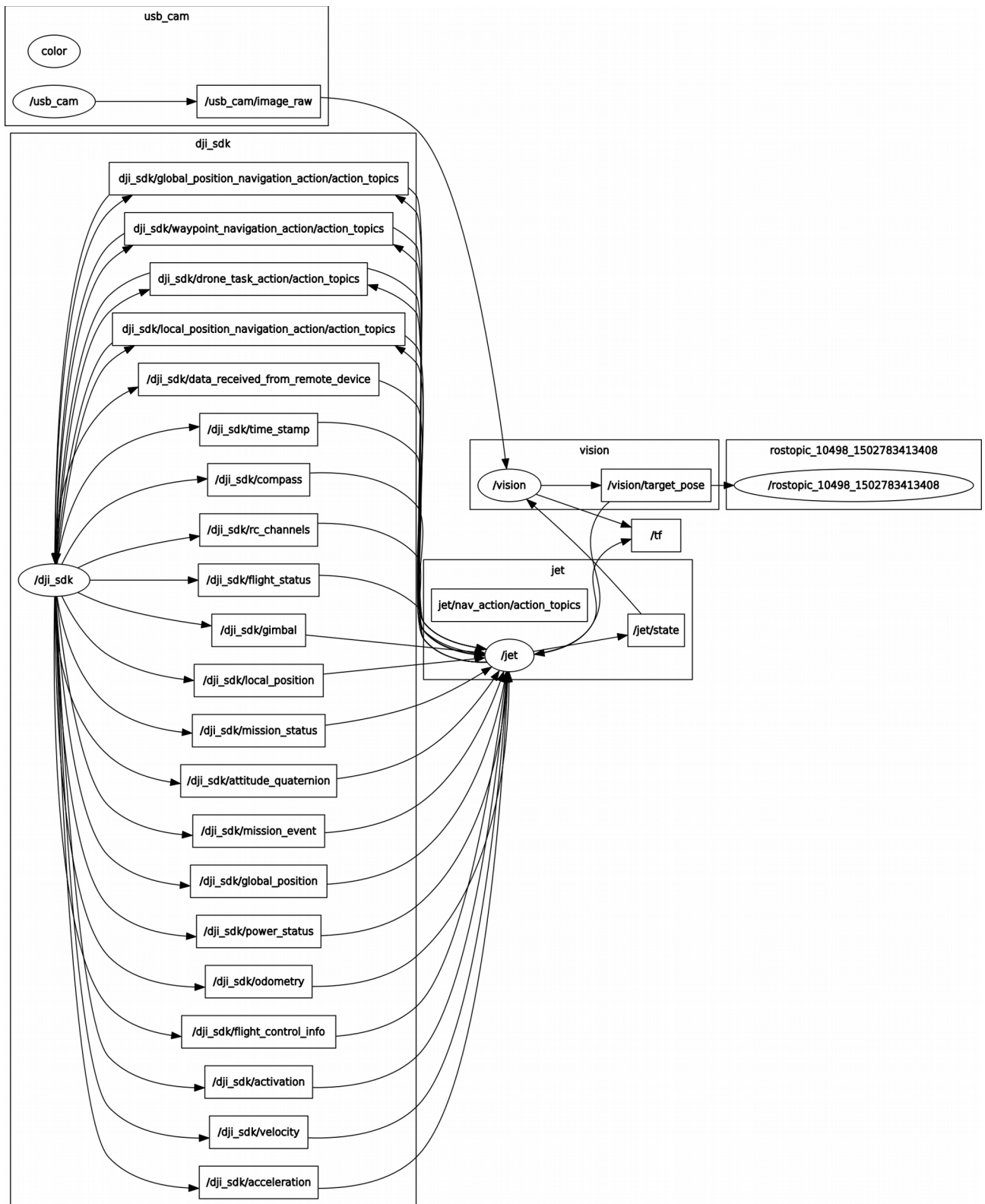
launch 相应的启动文件即可（roslaunch jet jet\_onboard.launch）

PS: 仿真环境下里程计用的是飞控的模拟数据，坐标为北东天，程序里面的飞行控制也用的是北东天，而 guidance 为北东地，实际飞行时候 jet\_onboard.launch 里面的 use\_guidance 属性一定要为 true，否则坐标系不对，会炸机！！

## 1.4.3 节点关系图与 tf 树

Recorded at time: 1502785483.25





#### 1.4.4 Topics 与 Services

rostopic list:

/jet/nav\_action/cancel



/jet/nav\_action/feedback

/jet/nav\_action/goal

/jet/nav\_action/result

/jet/nav\_action/status

/jet/pose\_calied

/jet/state

/tf

/vision/detection\_mode

/vision/result

/vision/target\_pose

rosservice list:

/jet/charge

/jet/grabber/cmd

/jet/grabber/stat

/jet/reload\_dropoint\_param

/jet/reload\_duration\_param

/jet/reload\_flight\_param

/jet/reload\_pid\_param

/usb\_cam/set\_camera\_info

/usb\_cam/start\_capture

/usb\_cam/stop\_capture

/vision/reload\_camera\_param

/vision/reload\_circle\_param

/vision/reload\_detmod\_param

/vision/reload\_marker\_param