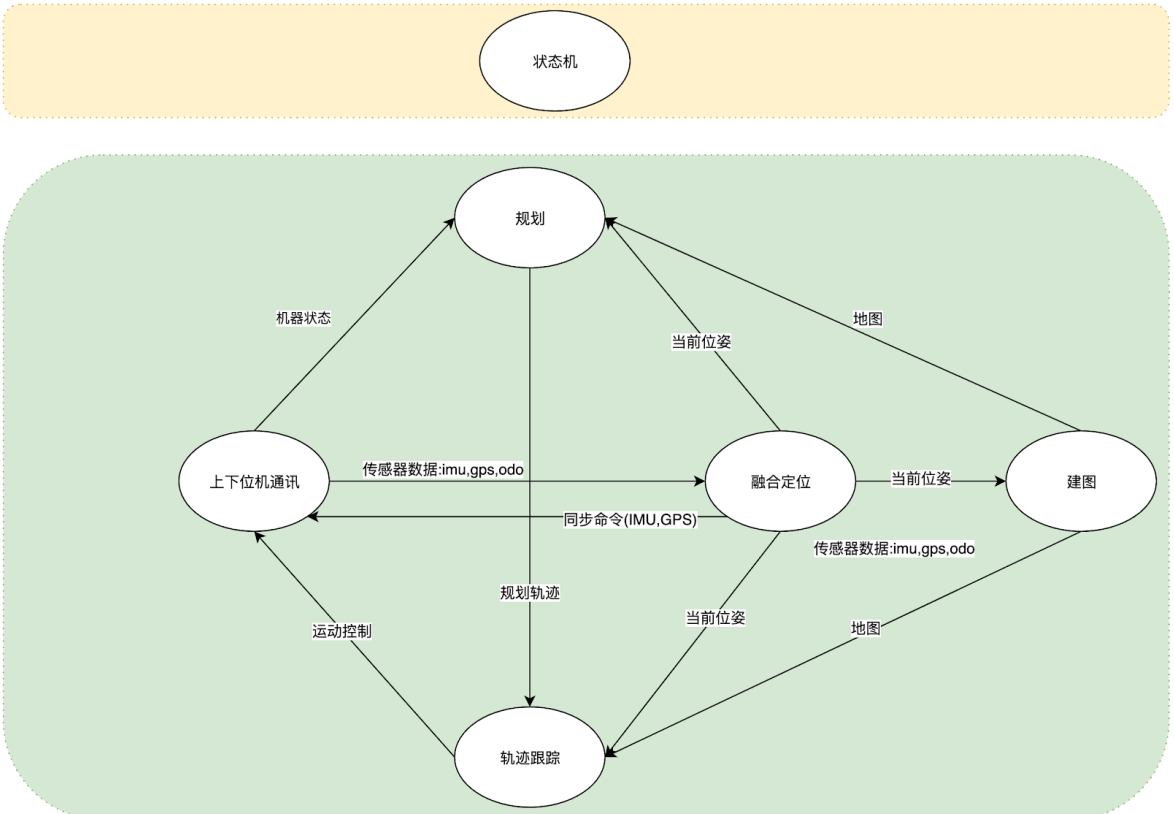
上位机ros系统框架

1.模块功能包

1. 消息,服务,活动包(msg_srv_act):系统间各个模块之间所需要的消息,服务,活动都存放于此功能包中.
2. 通讯包(protocol):上位机与下位机之间的通讯.将上位机其他模块需要发送给下位机的指令,发送给下位机.将下位机上报的数据,接收处理,发布到各个话题.
3. 定位包(location):负责机器人的定位,定位机器人再坐标系之中的位姿,将当前位姿发布到话题.
4. 地图包(map):负责机器人工作环境地图的建立与地图处理,将地图信息保存起来,在其他模块需要的时候提供地图信息.
5. 规划包(planning):负责机器人的路径规划,当其他模块有需要时,规划路径(覆盖规划,导航规划)
6. 轨迹跟踪包(tracking):根据机器人当前位姿个规划出的路径,计算出如何控制机器人跟踪规划出的路径.
7. 状态机包(FSM):建立机器人的状态机,处理机器人整套逻辑的变换.

2.系统

1.系统框架图



2.系统环境:

1. ubuntu版本:ubutn18.04
2. ros系统版本:melodic
3. python本版:3.6,2.7
4. opencv版本:cv2

ros使用的python2.7版本.程序使用的python3.7编写,因此使用虚拟环境同时使用python2.7和python3.7.

启动顺序:

1. source catkin_ws/devel/setup.bash
2. source catkin_ws/venv/ben/activate
3. 启动roslaunch 启动各个节点

3.功能包详解

1.msg_srv_act包

此包包括系统中所有的消息,服务,活动.各个模块发布的消息,接收的服务,活动,都存放与此功能包,相当于其他功能包的接口.

1) 消息:

a) M_senser_data.msg 传感器数据,包括IMU,gps,ODO:

uint32 base_time
uint32 imu_time
float64 pitch
float64 roll
float64 yaw
float64 acc_y

```
float64 gyo_x
float64 gyo_y
float64 gyo_z
uint32 gps_time
uint8 e_gps_state
float64 latitude
uint8 e_latitude
float64 longitude
uint8 e_longitude
float32 gps_speed
uint8 e_gps_mode
uint32 gps_rms
uint32 odo_time
int32 odo_speed
```

b) M_location_pose.msg ekf定位算法的位姿数据和协方差数据:

```
#pose is [x, y, yaw, pitch, v]
float64[5] location_pose
#location state    0 : rtk,imu,odo,fusion positioning    1: imu,odo,fusion #positioning
uint8 location_state
# covariance is 5*5  [[0,0,0,0,0],[0,0,0,0,0],[0,0,0,0,0],[0,0,0,0,0],[0,0,0,0,0]]
float64[25] covariance_pose
```

2) 服务:

a) S_answer_lower_machine.srv 上下位机之间的应答服务:

```
# 0 is receive the answer for lower machine
# 1 is send the answer to lower machine
uint8 service_type


---
# 1 is succeed ,0 is false
uint8 answer
```

b) S_mov_control_cmd.srv 下位机运动控制服务:

```
uint8 moving_cmd_type
int16 v
float32 w
```

```
float32 s_or_angle
---
# 1 is succeed ,0 is false
uint8 answer
```

c) S_location_server.srv 命令下位机开启响应的定位数据发送:

```
# 0 is Heading synchronization
# 1 is Coordinate synchronization
# 2 is Real-time positioning
uint8 server_type
# 1 is open ; 0 is close
uint8 server_state

---
# 1 is succeed ,0 is false
uint8 answer
```

d) S_map_server.srv 地图服务,用于其他节点获取相应的地图

```
 # the service type :0 is request the global map
uint8 server_type

---
# Pixel resolution in meters
float32 resolution
# [origin_x, origin_y, 0]
float32[] origin
#Map height in meters
float32 hight
#Map width in meters
float32 width
uint8[] map_data
```

3) 活动:

a) A_Coor_Sync.action 坐标同步活动,同步gps坐标与机器坐标:

```
# coordinate synchronization action

# Define the goal
#1 is start coordinate synchronization  0 is stop coordinate synchronization
uint8 action_type
---
# Define the result
# 1 is successed 0 is false
uint8 result
---
# Define a feedback message
# 0 is start  1 is doing  2 is finish
uint8 state
```

b) A_Heading_Sync.action 航向同步活动,同步gps航向与imu航向:

```
# heading synchronization action

# Define the goal
#1 is start heading synchronization  0 is stop heading synchronization
uint8 action_type
---
# Define the result
# 1 is successed 0 is false
uint8 result
---
# Define a feedback message
# 0 is start  1 is doing  2 is finish
uint8 state
```

c) A_build_map.action 建图活动,地图节点创建地图:

```
# build map action

# Define the goal
#1 is start build map  0 is stop build map
```

```
uint8 action_type
---
# Define the result
# 1 is successed 0 is false
uint8 result
---
# Define a feedback message
# 0 is start  1 is doing  2 is finish
uint8 state
```

2.protocol 包

上下位机之间的通讯,接收下位机上报的数据,或请求,解析后发布到相应的话题,接收其他节点发送的服务请求,将命令打包后发送给下位机.通讯协议遵循上下位机通讯协议.

1.发布:

1. `pub_senser = rospy.Publisher('lower_machine_senser_data', M_senser_data, queue_size=10)`
   发布接收到的下位机传感器数据到话题`'lower_machine_senser_data'`中,数据类型`M_senser_data`,队列大小`10`.

2.服务

1. `cmd_server = rospy.Service('send_mov_cmd', S_mov_control_cmd, Send_Moving_Control_Cmd)`
   运动控制服务,服务名称为'send_mov_cmd',数据类型为S_mov_control_cmd,将请求中运动控制的命令打包,发送给下位机.
2. `remote_cmd_server = rospy.Service('send_remote_cmd', S_mov_control_cmd, Send_Remote_Cmd)`
   遥控服务,服务名称为'send_remote_cmd',数据类型为S_mov_control_cmd,将请求中的遥控命令打包,发送给下位机
3. `answer_server = rospy.Service('answer_with_lower_machine', S_answer_lower_machine, Answer_With_Lower_Machine)`
   应答服务,服务名称为`'answer_with_lower_machine'`,数据类型为`S_answer_lower_machine`,上位机对下位机需要应答的动作做出应答.解析下位机的应答.
4. `location_server =  rospy.Service('send_location_server', S_location_server, Send_Location_Cmd)`
   定位服务,服务名称`'send_location_server'`,数据类型为`S_location_server`,向下位机发送定位数据是否发送的命令.

3.location 包

融合定位功能包,为机器人提供相对准确的定位位姿,和协方差矩阵.将定位的位姿发布到话题.

1.发布

1. `pub_location = rospy.Publisher('EKF_location_pose', M_location_pose, queue_size=10)`
   定位数据发布,话题为`'EKF_location_pose'`,数据类型为`M_location_pose`,队列为`10`.

2.订阅

1. `Sub_senser_data = rospy.Subscriber("lower_machine_senser_data", M_senser_data, Location)`
   订阅`lower_machine_senser_data`话题中的`M_senser_data.msg`数据,

3.map包

地图功能包,为机器人提供地图相关的功能:包括建立地图,地图优化,处理.在其他功能包需要的时候,提供相应的地图为其服务.

1.活动

1. `self._as = actionlib.SimpleActionServer(self.action_name, A_build_mapAction, self.execute, False)`
   活动服务器名称为`"build_map_action_server"`,数据类型为`A_build_mapAction`,当接收到的`goal`为`1`的时候开始建图,`0`的时候结束建图.

2.订阅

1. `self.subscribe = rospy.Subscriber("EKF_location_pose", M_location_pose, self.analyse,  queue_size = 10)`
   订阅定位数据,订阅话题为`"EKF_location_pose"`,数据类型为`M_location_pose`,队列长度为`10`.

3.服务

1. `map_provided_server = rospy.Service('map_provided_server', S_map_server, Map_Priovided)`
   提供地图服务,服务名`'map_provided_server'`,数据类型`S_map_server`,当地图请求到来时,提供相应的地图.

4.planning包

规划功能包,其功能包括点到点之间的路径规划,沿边路径规划,还有地图的覆盖规划.需要从map包中获取地图信息.并将规划的路径发送tracking包,让tracking包对该路径进行轨迹跟踪.

此功能包尚未完成,在此留白!!

5.tracking包

轨迹跟踪功能包,此功能为跟踪规划好的路径,包括点到点的路径跟踪,还有全局覆盖的路径轨迹跟踪,沿边轨迹跟踪.根据接收到的轨迹,当前位姿,计算处运动控制参数,通过protocol给下位机下达控制指令,控制机器延规划的轨迹运动.

接口尚未完成,在此留白!!

6.FSM包

此功能包为机器整体逻辑运作功能包,使用ros中smach状态机,调度完整的机器状态,完成整套工作逻辑.

接口尚未完成,在此留白!!

此文档尚未完成,仅为初稿!!