af_bringup

Generated by Doxygen 1.8.14

Contents

Chapter 1

Namespace Index

1.1 Packages

Here are the packages with brief descriptions (if available):

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Chapter 4

Namespace Documentation

4.1 fuse_G_A_dox Namespace Reference

Classes

- class AMCL_unit
  *AMCL*存储类
- class GPS_unit
  *GPS*数据存储类
- class Odom_unit
  里程计数据存储类
- class SELECTED
  筛选结果类

Functions

- def matrix_from_theta (theta)
  角度生成旋转矩阵*(2x2)*
- def theta_from_matrix (R)
  旋转矩阵*(2x2)*转成角度
- def sendTransform ()
  发送*topic*
- def tran_mat44 (trans)
  将*ros*格式中的*Transform*数据转为矩阵*(4X4)*
- def tran_theta_T (trans)
  将*ros*格式中的*Transform*数据转为矩阵*(2X2)*
- def normalization (a, b)
  归一化
- def make_choice ()
  筛选函数

Variables

- int MAX_STD = 999999999
  最大的标准差
- tfBuffer = tf2_ros.Buffer()
- listener = tf2_ros.TransformListener(tfBuffer)
- amcl_pose = AMCL_unit()
- gps_pose = GPS_unit()
- odom_pose = Odom_unit()
- out_pose = SELECTED()
- pub = rospy.Publisher('/fuse/tf', Fuse_tf, queue_size=100)
- t2 = threading.Thread(target=sendTransform)
- t1 = threading.Thread(target=make_choice)

4.1.1 Function Documentation

4.1.1.1 make_choice()

```
def fuse_G_A_dox.make_choice ( )
```

筛选函数

```
438 def make_choice():
439     rate = rospy.Rate(100)
440     while not rospy.is_shutdown():
441 
444         _x1 = amcl_pose.x
445 
448         _x2 = gps_pose.x
449 
452         _std_x1 = amcl_pose.x_std
453 
456         _std_x2 = gps_pose.x_std
457         if _std_x1 == MAX_STD and _std_x2 != MAX_STD:
458 
461             _w_x1 = 0
462 
465             _w_x2 = 1
466         elif _std_x1 != MAX_STD and _std_x2 == MAX_STD:
467 
470             _w_x1 = 1
471 
474             _w_x2 = 0
475         elif _std_x1 == MAX_STD and _std_x2 == MAX_STD:
476             continue
477         else:
478 
481             _w_x1 = _std_x2 / (_std_x1 + _std_x2)
482 
485             _w_x2 = _std_x1 / (_std_x1 + _std_x2)
486         if _w_x1 > _w_x2:
487             out_pose.x_choice = out_pose.choice[1]
488         else:
489             out_pose.x_choice = out_pose.choice[0]
490         _w_x1, _w_x2 = normalization(_w_x1, _w_x2)
491         out_pose.x = _w_x1 * _x1 + _w_x2 * _x2
492 
495         _y1 = amcl_pose.y
496 
499         _y2 = gps_pose.y
500 
503         _std_y1 = amcl_pose.y_std
504 
507         _std_y2 = gps_pose.y_std
508         if _std_y1 == MAX_STD and _std_y2 != MAX_STD:
509 
```

```
512             _w_y1 = 0
513
516             _w_y2 = 1
517         elif _std_y1 != MAX_STD and _std_y2 == MAX_STD:
518
521             _w_y1 = 1
522
525             _w_y2 = 0
526         elif _std_y1 == MAX_STD and _std_y2 == MAX_STD:
527             continue
528         else:
529
532             _w_y1 = _std_y2 / (_std_y1 + _std_y2)
533
536             _w_y2 = _std_y1 / (_std_y1 + _std_y2)
537         if _w_y1 > _w_y2:
538             out_pose.y_choice = out_pose.choice[1]
539         else:
540             out_pose.y_choice = out_pose.choice[0]
541         _w_y1, _w_y2 = normalization(_w_y1, _w_y2)
542         out_pose.y = _w_y1 * _y1 + _w_y2 * _y2
543
546         _theta1 = amcl_pose.theta
547
550         _theta2 = gps_pose.theta
551
554         _std_theta1 = amcl_pose.theta_std
555
558         _std_theta2 = gps_pose.theta_std
559         if _std_theta1 == MAX_STD and _std_theta2 != MAX_STD:
560
563             _w_theta1 = 0
564
567             _w_theta2 = 1
568         elif _std_theta1 != MAX_STD and _std_theta2 == MAX_STD:
569
572             _w_theta1 = 1
573
576             _w_theta2 = 0
577         elif _std_theta1 == MAX_STD and _std_theta2 == MAX_STD:
578             continue
579         else:
580
583             _w_theta1 = _std_theta2 / (_std_theta1 + _std_theta2)
584
587             _w_theta2 = _std_theta1 / (_std_theta1 + _std_theta2)
588         if _w_theta1 > _w_theta2:
589             out_pose.theta_choice = out_pose.choice[1]
590         else:
591             out_pose.theta_choice = out_pose.choice[0]
592         _w_theta1, _w_theta2 = normalization(_w_theta1, _w_theta2)
593         out_pose.theta = _w_theta1 * _theta1 + _w_theta2 * _theta2
594         out_pose.output()
595         print "************OUT*************"
596         amcl_pose.output()
597         print "############AMCL#############"
598         gps_pose.output()
599         print "############GPS#############"
600
601         out_pose.set_map(out_pose.x, out_pose.y, out_pose.theta)
602
603         rate.sleep()
604
605
```

4.1.1.2 matrix_from_theta()

```
def fuse_G_A_dox.matrix_from_theta (
                theta )
```

角度生成旋转矩阵(2x2)

Parameters

theta	输入角度

Returns

输出旋转矩阵

```
39 def matrix_from_theta(theta):
40
41     return np.array([[np.cos(theta), -np.sin(theta)], [np.sin(theta), np.cos(theta)]])
42
43
```

4.1.1.3 normalization()

```
def fuse_G_A_dox.normalization (
              a,
              b )
```

归一化

Parameters

a	输入
b	输入

Returns

a a的归一化值
b b的归一化值

```
431 def normalization(a, b):
432
433     sum = a + b
434     return a/sum, b/sum
435
436
```

4.1.1.4 sendTransform()

```
def fuse_G_A_dox.sendTransform ( )
```

发送topic

40hz发布topic

Parameters

out_pose	ros输出格式数据

```
371 def sendTransform():
```

```
372
373     rate = rospy.Rate(40)
374     br = tf2_ros.TransformBroadcaster()
375     while not rospy.is_shutdown():
376         pub.publish(out_pose.fuse_tf)
377         rate.sleep()
378
379
```

4.1.1.5 theta_from_matrix()

```
def fuse_G_A_dox.theta_from_matrix (
                R )
```

旋转矩阵(2x2)转成角度

Parameters

R	输入旋转矩阵

Returns

输出角度值

```
53 def theta_from_matrix(R):
54
55
58     _R_12 = R[0, 1]
59
62     _R_22 = R[1, 1]
63     if -0.001 < _R_22 < 0.001 and _R_12 > 0:
64         theta = math.pi / 2
65     elif -0.001 < _R_22 < 0.001 and _R_12 < 0:
66         theta = math.pi / 2
67     elif 0.001 <= _R_22 and _R_12 > 0:
68         theta = math.atan(_R_12 / _R_22)
69     elif 0.001 <= _R_22 and _R_12 < 0:
70         theta = math.atan(_R_12 / _R_22)
71     elif _R_22 <= -0.001 and _R_12 > 0:
72         theta = math.atan(_R_12 / _R_22) + math.pi
73     elif _R_22 <= -0.001 and _R_12 < 0:
74         theta = math.atan(_R_12 / _R_22) - math.pi
75     return theta
76
77
```

4.1.1.6 tran_mat44()

```
def fuse_G_A_dox.tran_mat44 (
                trans )
```

将ros格式中的Transform数据转为矩阵(4X4)

Parameters

trans	ros格式Transform数据

```
386 def tran_mat44(trans):
387 
388     trans_xyz = [trans.transform.translation.x, trans.transform.translation.y, 0]
389     trans_quat = [0, 0, trans.transform.rotation.z,
390                   trans.transform.rotation.w]
391     mat44 = np.dot(tft.translation_matrix(trans_xyz), tft.quaternion_matrix(trans_quat))
392     return mat44
393 
394 
```

4.1.1.7 tran_theta_T()

```
def fuse_G_A_dox.tran_theta_T (
               trans )
```

将ros格式中的Transform数据转为矩阵(2X2)

Parameters

trans	ros格式Transform数据

```
401 def tran_theta_T(trans):
402 
403 
406     _quat = [trans.transform.rotation.x, trans.transform.rotation.y,
407              trans.transform.rotation.z,
408              trans.transform.rotation.w]
409 
412     _theta = tft.euler_from_quaternion(_quat)[2]
413 
416     _T = np.array([[trans.transform.translation.x], [trans.transform.translation.y]])
417     return _theta, _T
418 
419 
```

4.1.2 Variable Documentation

4.1.2.1 amcl_pose

```
fuse_G_A_dox.amcl_pose = AMCL_unit()
```

4.1.2.2 gps_pose

```
fuse_G_A_dox.gps_pose = GPS_unit()
```

4.1.2.3 listener

```
fuse_G_A_dox.listener = tf2_ros.TransformListener(tfBuffer)
```

4.1.2.4 MAX_STD

```
int fuse_G_A_dox.MAX_STD = 999999999
```

最大的标准差

4.1.2.5 odom_pose

```
fuse_G_A_dox.odom_pose = Odom_unit()
```

4.1.2.6 out_pose

```
fuse_G_A_dox.out_pose = SELECTED()
```

4.1.2.7 pub

```
fuse_G_A_dox.pub = rospy.Publisher('/fuse/tf', Fuse_tf, queue_size=100)
```

4.1.2.8 t1

```
fuse_G_A_dox.t1 = threading.Thread(target=make_choice)
```

4.1.2.9 t2

```
fuse_G_A_dox.t2 = threading.Thread(target=sendTransform)
```

4.1.2.10 tfBuffer

```
fuse_G_A_dox.tfBuffer = tf2_ros.Buffer()
```

4.2 select_A_dox Namespace Reference

Classes

- class AMCL_unit
  *AMCL*存储类
- class GPS_unit
  *GPS*数据存储类
- class Odom_unit
  里程计数据存储类
- class SELECTED
  筛选结果类

Functions

- def normalize (z)
  归一化
- def angle_diff (a, b)
  角度差值
- def matrix_from_theta (theta)
  角度生成旋转矩阵*(2x2)*
- def theta_from_matrix (R)
  旋转矩阵*(2x2)*转成角度
- def sendTransform (out_pose)
  发送*TF(topic)*
- def tran_mat44 (trans)
  将*ros*格式中的*Transform*数据转为矩阵*(4X4)*
- def tran_theta_T (trans)
  将*ros*格式中的*Transform*数据转为矩阵*(2X2)*
- def make_choice ()
  筛选函数
- def amcl_initial_update (x_std=1, y_std=1, theta_std=1)
  *AMCL*初始化函数
- def amcl_initial_callback (msg)
  AMCL 初始化回调函数
- def amcl_global_update ()
  *AMCL*全局撒粒子
- def amcl_nomotion_update ()
  *AMCL*未运动状态下粒子更新
- def amcl_recovery ()
  *AMCL*恢复状态过程
- def amcl_process ()
  *AMCL*线程
- def main ()
  主函数

Variables

- int MAX_STD = 999999999
  最大的标准差
- int flag_robot_move = 0
  机器人是否运动标志
- float STATIC_LIMIT_LINEAR = 0.05
  判断机器人是否运动的速度限制
- float STATIC_LIMIT_ANGULAR = 0.05
  判断机器人是否运动的角度限制

4.2.1 Function Documentation

4.2.1.1 amcl_global_update()

```
def select_A_dox.amcl_global_update ( )
```

AMCL全局撒粒子

```
555 def amcl_global_update():
556     rospy.wait_for_service('global_localization')
557     try:
558         global_update = rospy.ServiceProxy('global_localization', Empty)
559         global_update()
560     except rospy.ServiceException, e:
561         print "Service call failed: %s" % e
562
563
```

4.2.1.2 amcl_initial_callback()

```
def select_A_dox.amcl_initial_callback (
              msg )
```

AMCL 初始化回调函数

Parameters

msg	ros回调数据

```
541 def amcl_initial_callback(msg):
542
543     # msg = PoseWithCovarianceStamped()
544     global amcl_output
545     print "   "
546     # amcl_output.x = msg.pose.pose.position.x
547     # amcl_output.y = msg.pose.pose.position.y
548     # _quat = [msg.pose.pose.orientation.x, msg.pose.pose.orientation.y,
549     #          msg.pose.pose.orientation.z,
```

```
550     #          msg.pose.pose.orientation.w]
551     # amcl_output.theta = tft.euler_from_quaternion(_quat)[2]
552
553
```

4.2.1.3 amcl_initial_update()

```
def select_A_dox.amcl_initial_update (
              x_std = 1,
              y_std = 1,
              theta_std = 1 )
```

AMCL初始化函数

Parameters

x_std	x方向方差
y_std	y方向方差
theta_std	角度房车

```
506 def amcl_initial_update(x_std=1, y_std=1, theta_std=1):
507
508     rate = rospy.Rate(10)
509     while not rospy.is_shutdown():
510         try:
511             trans = tfBuffer.lookup_transform('map', 'base_footprint', rospy.Time())
512         except (tf2_ros.LookupException, tf2_ros.ConnectivityException, tf2_ros.ExtrapolationException):
513             rate.sleep()
514             continue
515
516         global pub_amcl
517         amcl_init_pose.header.stamp = rospy.Time.now()
518         amcl_init_pose.header.frame_id = "map"
519         amcl_init_pose.pose.pose.position.x = trans.transform.translation.x
520         amcl_init_pose.pose.pose.position.y = trans.transform.translation.y
521         amcl_init_pose.pose.pose.position.z = trans.transform.translation.z
522         amcl_init_pose.pose.pose.orientation.x = trans.transform.rotation.x
523         amcl_init_pose.pose.pose.orientation.y = trans.transform.rotation.y
524         amcl_init_pose.pose.pose.orientation.z = trans.transform.rotation.z
525         amcl_init_pose.pose.pose.orientation.w = trans.transform.rotation.w
526         amcl_init_pose.pose.covariance = [0.0] * 36
527         amcl_init_pose.pose.covariance[0] = x_std ** 2
528         amcl_init_pose.pose.covariance[7] = y_std ** 2
529         amcl_init_pose.pose.covariance[35] = theta_std ** 2
530         pub_amcl.publish(amcl_init_pose)
531         break
532         # print "pub amcl ok" + str(input_yaw)
533
534
```

4.2.1.4 amcl_nomotion_update()

```
def select_A_dox.amcl_nomotion_update ( )
```

AMCL未运动状态下粒子更新

```
565 def amcl_nomotion_update():
566     rospy.wait_for_service('request_nomotion_update')
567     try:
568         nomotion_update = rospy.ServiceProxy('request_nomotion_update', Empty)
569         nomotion_update()
570     except rospy.ServiceException, e:
571         print "Service call failed: %s" % e
572 
573 
```

4.2.1.5 amcl_process()

```
def select_A_dox.amcl_process ( )
```

AMCL线程

包括AMCL位置纠正,AMCL位置正确度判断和AMCL自恢复

```
605 def amcl_process():
606 
607     global amcl_output
608     while not rospy.is_shutdown():
609         if amcl_recovery():
610             print "ok"
611             break
612     while not rospy.is_shutdown():
613         if amcl_pose.x_std > 0.5 or amcl_pose.y_std > 0.5:
614             rospy.wait_for_message('/scan', LaserScan)
615             n = 0
616             while not rospy.is_shutdown():
617                 if n <= 4:
618                     amcl_initial_update()
619                     print "amcl_initial"
620                 else:
621                     if (n - 5) % 5 == 0:
622                         amcl_global_update()
623                         print "amcl_global"
624                 n += 1
625                 if amcl_recovery():
626                     print "recovery ok"
627                     amcl_output.x = amcl_pose.x
628                     amcl_output.y = amcl_pose.y
629                     amcl_output.theta = amcl_pose.theta
630                     break
631             print '!!!!!!!!!!!!!!!!!!!!'
632         else:
633             amcl_output.x = amcl_pose.x
634             amcl_output.y = amcl_pose.y
635             amcl_output.theta = amcl_pose.theta
636             print amcl_pose.x_std, amcl_pose.y_std
637             # print "amcl_ok"
638 
639 
```

4.2.1.6 amcl_recovery()

```
def select_A_dox.amcl_recovery ( )
```

AMCL恢复状态过程

```
575 def amcl_recovery():
576     global amcl_pose
577     if flag_robot_move:
578         t = 0
579         while t < 15:
580             time.sleep(1)
581             print amcl_pose.x_std, amcl_pose.y_std
582             if amcl_pose.x_std > 0.5 or amcl_pose.y_std > 0.5:
583                 t += 1
584             else:
585                 return True
586         return False
587     else:
588         t = 0
589         while t < 15:
590             time.sleep(1)
591             amcl_nomotion_update()
592             print amcl_pose.x_std, amcl_pose.y_std
593             if amcl_pose.x_std > 0.5 or amcl_pose.y_std > 0.5:
594                 t += 1
595             else:
596                 return True
597         return False
598
599
```

4.2.1.7 angle_diff()

```
def select_A_dox.angle_diff (
                a,
                b )
```

角度差值

Parameters

a	输入角度
b	输入角度

Returns

角度差值

```
64 def angle_diff(a, b):
65
66     a = normalize(a)
67     b = normalize(b)
68     d1 = a - b
69     d2 = 2 * math.pi - math.fabs(d1)
70     if d1 > 0:
71         d2 *= -1.0
72     if math.fabs(d1) < math.fabs(d2):
73         return d1
74     else:
75         return d2
76
77
```

4.2.1.8 main()

```
def select_A_dox.main ( )
```

主函数

```
641 def main():
642     global tfBuffer, listener, amcl_output, amcl_init_pose, odom_pose, out_pose, pub, pub_amcl
643     try:
644         rospy.init_node('combined_test')
645         tfBuffer = tf2_ros.Buffer()
646         listener = tf2_ros.TransformListener(tfBuffer)
647         amcl_pose = AMCL_unit()
648         amcl_output = AMCL_unit()
649         amcl_init_pose = PoseWithCovarianceStamped()
650         gps_pose = GPS_unit()
651         odom_pose = Odom_unit()
652         out_pose = SELECTED()
653         rospy.Subscriber('/fuse/map_odom', PoseWithCovarianceStamped, amcl_pose.fix)
654         rospy.Subscriber('/Gnss_Odom_res', Gnss_Odom_lyz, gps_pose.update)
655         rospy.Subscriber('/odom', Odometry, amcl_pose.update)
656         rospy.Subscriber('/initialpose', PoseWithCovarianceStamped, amcl_initial_callback)
657         pub = rospy.Publisher('/fuse/tf', Fuse_tf, queue_size=100)
658         pub_amcl = rospy.Publisher('/initialpose', PoseWithCovarianceStamped, queue_size=1)
659 
660         t1 = threading.Thread(target=make_choice)
661         t2 = threading.Thread(target=amcl_process)
662         t1.start()
663         t2.start()
664 
665         rospy.spin()
666     except rospy.ROSInterruptException:
667         pass
668 
669 
```

4.2.1.9 make_choice()

```
def select_A_dox.make_choice ( )
```

筛选函数

```
483 def make_choice():
484     rate = rospy.Rate(10)
485     while not rospy.is_shutdown():
486         out_pose.x = amcl_output.x
487         out_pose.y = amcl_output.y
488         out_pose.theta = amcl_output.theta
489         out_pose.set_map(out_pose.x, out_pose.y, out_pose.theta)
490         # sendRVIZ()
491         sendTransform(out_pose)
492         # amcl_pose.lock()
493         # gps_pose.lock()
494 
495         rate.sleep()
496 
497 
```

4.2.1.10 matrix_from_theta()

```
def select_A_dox.matrix_from_theta (
              theta )
```

角度生成旋转矩阵(2x2)

Parameters

theta	输入角度

Returns

输出旋转矩阵

```
87 def matrix_from_theta(theta):
88
89     return np.array([[np.cos(theta), -np.sin(theta)], [np.sin(theta), np.cos(theta)]])
90
91
```

4.2.1.11 normalize()

```
def select_A_dox.normalize (
              z )
```

归一化

Parameters

z	角度输入

Returns

角度输出(0∼pi)

```
49 def normalize(z):
50
51     return math.atan2(math.sin(z), math.cos(z))
52
53
```

4.2.1.12 sendTransform()

```
def select_A_dox.sendTransform (
              out_pose )
```

发送TF(topic)

Parameters

out_pose	ros输出格式数据

```
437 def sendTransform(out_pose):
438 
439     pub.publish(out_pose.fuse_tf)
440 
441 
```

4.2.1.13 theta_from_matrix()

```
def select_A_dox.theta_from_matrix (
              R )
```

旋转矩阵(2x2)转成角度

Parameters

R	输入旋转矩阵

Returns

输出角度值

```
101 def theta_from_matrix(R):
102 
103 
106     _R_12 = R[0, 1]
107 
110     _R_22 = R[1, 1]
111     if -0.001 < _R_22 < 0.001 and _R_12 > 0:
112         theta = math.pi / 2
113     elif -0.001 < _R_22 < 0.001 and _R_12 < 0:
114         theta = math.pi / 2
115     elif 0.001 <= _R_22 and _R_12 > 0:
116         theta = math.atan(_R_12 / _R_22)
117     elif 0.001 <= _R_22 and _R_12 < 0:
118         theta = math.atan(_R_12 / _R_22)
119     elif _R_22 <= -0.001 and _R_12 > 0:
120         theta = math.atan(_R_12 / _R_22) + math.pi
121     elif _R_22 <= -0.001 and _R_12 < 0:
122         theta = math.atan(_R_12 / _R_22) - math.pi
123     return theta
124 
125 
```

4.2.1.14 tran_mat44()

```
def select_A_dox.tran_mat44 (
              trans )
```

将ros格式中的Transform数据转为矩阵(4X4)

Parameters

trans	ros格式Transform数据

```
448 def tran_mat44(trans):
449 
450     trans_xyz = [trans.transform.translation.x, trans.transform.translation.y, 0]
451     trans_quat = [0, 0, trans.transform.rotation.z,
452                   trans.transform.rotation.w]
453     mat44 = np.dot(tft.translation_matrix(trans_xyz), tft.quaternion_matrix(trans_quat))
454     return mat44
455 
456 
```

4.2.1.15 tran_theta_T()

```
def select_A_dox.tran_theta_T (
               trans )
```

将ros格式中的Transform数据转为矩阵(2X2)

Parameters

trans	ros格式Transform数据

```
463 def tran_theta_T(trans):
464 
465 
468     _quat = [trans.transform.rotation.x, trans.transform.rotation.y,
469              trans.transform.rotation.z,
470              trans.transform.rotation.w]
471 
474     _theta = tft.euler_from_quaternion(_quat)[2]
475 
478     _T = np.array([[trans.transform.translation.x], [trans.transform.translation.y]])
479     return _theta, _T
480 
481 
```

4.2.2 Variable Documentation

4.2.2.1 flag_robot_move

```
int select_A_dox.flag_robot_move = 0
```

机器人是否运动标志

4.2.2.2 MAX_STD

```
int select_A_dox.MAX_STD = 999999999
```

最大的标准差

4.2.2.3 STATIC_LIMIT_ANGULAR

```
float select_A_dox.STATIC_LIMIT_ANGULAR = 0.05
```

判断机器人是否运动的角度限制

4.2.2.4 STATIC_LIMIT_LINEAR

```
float select_A_dox.STATIC_LIMIT_LINEAR = 0.05
```

判断机器人是否运动的速度限制

4.3 serial_af_dox Namespace Reference

Classes

- class Encoder
  类用来存储里程计信息
- class Fuse_GPS_Odom
  class Fuse_GPS_Odom 用来存储*GPS*和*Encoder*的信息
- class GPS
  存储*GPS*信息的类

Functions

- def sound_range_init ()
  超声波数据初始化
- def range_pub (range_msg)
  *range_pub range*的激光输出函数
- def serial_process (port='/dev/pts/27', baudrate=460800, timeout=1)
  serial_process 串口线程
- def main ()
  *main*函数

Variables

- crc_novetal = crcmod.mkCrcFun(0x104C11DB7, 0, True, 0)
  *crc32*校验程序*(GPS)*
- mutex = threading.Lock()
  线程锁
- fuse_gps_odom = Fuse_GPS_Odom()
- sound_range = LaserScan()
  超声波模拟激光输出类
- int sound_seq = 0
  超声波输出计数

4.3.1 Function Documentation

4.3.1.1 main()

```
def serial_af_dox.main ( )
```

main函数

ros的初始化 serial线程的开始 spin线程的开始

```
478 def main():
479     try:
480         rospy.init_node('serial_fuse', anonymous=True)
481         sound_range_init()
482         global pub, pub_range_5, pub_range_6, pub_range_7, pub_range_8, size_
483 
484         m_ = np.arange(sound_range.angle_min, sound_range.angle_max, sound_range.angle_increment)
485 
486         size_ = len(m_)
487         pub = rospy.Publisher('/fuse', Fuse_G_O, queue_size=100)
488         pub_range_5 = rospy.Publisher('/range/5', LaserScan, queue_size=100)
489         pub_range_6 = rospy.Publisher('/range/6', LaserScan, queue_size=100)
490         pub_range_7 = rospy.Publisher('/range/7', LaserScan, queue_size=100)
491         pub_range_8 = rospy.Publisher('/range/8', LaserScan, queue_size=100)
492 
493         serial_process()
494         rospy.spin()
495     except rospy.ROSInterruptException:
496         pass
497 
```

4.3.1.2 range_pub()

```
def serial_af_dox.range_pub (
               range_msg )
```

range_pub range的激光输出函数

将list格式的range数据转发成ros_laser模式,并通过/range/* topic发出

Parameters

range_msg	range的输入[1,...,3](list)

Examples

```
>>> a = float("inf")
inf
>>> b = float("inf")
inf
>>> a + b
inf
>>> a * b
inf
>>> a + 12
```

```
inf
>>> a - 12
inf
```

```
301 def range_pub(range_msg):
302 
303     global sound_seq
304     sound_range.header.stamp = rospy.Time.now()
305     sound_range.header.seq = sound_seq
306     sound_range.header.frame_id = 'range5'
307     if range_msg[4] == 3500:
308         out_msg = float("inf")
309     else:
310         out_msg = range_msg[4]
311     #print range_msg,out_msg
312     sound_range.ranges = size_ * [out_msg / 1000.0]
313     pub_range_5.publish(sound_range)
314     sound_range.header.frame_id = 'range6'
315     if range_msg[5] == 3500:
316         out_msg = float("inf")
317     else:
318         out_msg = range_msg[5]
319     sound_range.ranges = size_ * [out_msg / 1000.0]
320     pub_range_6.publish(sound_range)
321     sound_range.header.frame_id = 'range7'
322     if range_msg[6] == 3500:
323         out_msg = float("inf")
324     else:
325         out_msg = range_msg[6]
326     sound_range.ranges = size_ * [out_msg / 1000.0]
327     pub_range_7.publish(sound_range)
328     sound_range.header.frame_id = 'range8'
329     if range_msg[7] == 3500:
330         out_msg = float("inf")
331     else:
332         out_msg = range_msg[7]
333     sound_range.ranges = size_ * [out_msg / 1000.0]
334     pub_range_8.publish(sound_range)
335     sound_seq += 1
336 
337 
```

4.3.1.3 serial_process()

```
def serial_af_dox.serial_process (
               port = '/dev/pts/27',
               baudrate = 460800,
               timeout = 1 )
```

serial_process 串口线程

处理串口中的三包数据,GPS,里程计和超声波,GPS和里程计分别存储在GPS类和Encoder类中,在里程计接收过程中讲两个类的数据存储在融合类Fuse_GPS_Odom中.

Parameters

port	串口
baudrate	波特率
timeout	时间容差

See also

ser_1 读取的第一个串口值
encoder_a 串口剩余部分

encoder_b 解的串口数据
crc 解的串口校验值
encoder_crc 计算的串口校验值

Notes

GPS和超声波数据类似里程计读取流程

```
358 def serial_process(port='/dev/pts/27', baudrate=460800, timeout=1):
359     serial_ser = serial.Serial(port=port, baudrate=baudrate, timeout=timeout)
360     serial_ser.flushOutput()
361     serial_ser.flushInput()
362     encoder = Encoder()
363     gps = GPS()
364     print "in serial process"
365 
366     while not rospy.is_shutdown():
367         ser_1 = serial_ser.read(1)
368         if ser_1 == '\x86':
369             if serial_ser.read(1) == '\x0a':
370                 encoder_a = serial_ser.read(9)
371                 # print '\x86\x0a' + encoder_a
372                 if len(encoder_a) == 9:
373                     encoder_b = struct.unpack('<2hIB', encoder_a)
374                     crc = encoder_b[3]
375                     crc_split = struct.unpack("<9B", '\x0a' + encoder_a[0:-1])
376                     encoder_crc = sum(crc_split) & 0xff
377                     if encoder_crc == crc:
378                         try:
379                             encoder.left_encoder_val = encoder_b[0]
380                             encoder.right_encoder_val = encoder_b[1]
381                             encoder.encoder_time = encoder_b[2]
382                             fuse_gps_odom.fuse_encoder = encoder
383                             # print "encoder_filled"
384                             fuse_gps_odom.fill()
385                             pub.publish(fuse_gps_odom.fuse_gps_odom_msg)
386                             # range_pub(range_b[0:8])
387                             # print "pub ok"
388                         except:
389                             print "encoder serial error"
390                     else:
391                         continue
392                 else:
393                     continue
394             else:
395                 continue
396         elif ser_1 == '\xaa':
397             # print 'aa'
398             if serial_ser.read(1) == '\x44':
399                 # print '44'
400                 if serial_ser.read(1) == '\x12':
401                     # print '12'
402                     print "gps_get"
403                     gps_a = serial_ser.read(86)
404                     if len(gps_a) == 86:
405                         gps_b = struct.unpack('<2I3d3f2B2f3dI4B', gps_a)
406                         crc = (gps_b[19] << 24) | (gps_b[18] << 16) | (gps_b[17] << 8) | (gps_b[16])
407                         crc_split = '\xaa' + '\x44' + '\x12' + gps_a[0:82]
408                         gps_crc = crc_novetal(crc_split)
409                         if gps_crc == crc:
410                             try:
411                                 gps.Slo_stat = gps_b[0]
412                                 gps.Pos_type = gps_b[1]
413                                 gps.lat = gps_b[2]
414                                 gps.lon = gps_b[3]
415                                 gps.hgt = gps_b[4]
416                                 gps.lat_std = gps_b[5]
417                                 gps.lon_std = gps_b[6]
418                                 gps.hgt_std = gps_b[7]
419                                 gps.SVs = gps_b[8]
420                                 gps.solnSVs = gps_b[9]
421                                 gps.heading = gps_b[10]
422                                 gps.hdg_std = gps_b[11]
423                                 gps.hor_spd = gps_b[12]
424                                 gps.Trk_gnd = gps_b[13]
425                                 gps.Vert_spd = gps_b[14]
426                                 gps.gps_time = gps_b[15]
427                                 fuse_gps_odom.fuse_gps = gps
428                                 # fuse_gps_odom.fill()
429                                 # pub.publish(fuse_gps_odom.fuse_gps_odom_msg)
430                                 # print "gps filled"
431                             except:
432                                 print "gps serial error"
```

```
433                         else:
434                             continue
435                     else:
436                         continue
437                 else:
438                     continue
439             else:
440                 continue
441         elif ser_1 == '\x88':
442             if serial_ser.read(1) == '\x13':
443                 range_a = serial_ser.read(17)
444                 if len(range_a) == 17:
445                     range_b = struct.unpack('<8HB', range_a)
446                     crc = range_b[8]
447                     crc_split = struct.unpack("<17B", '\x13' + range_a[0:-1])
448                     range_crc = sum(crc_split) & 0xff
449                     if range_crc == crc:
450                         try:
451                             # encoder.left_encoder_val = encoder_b[0]
452                             # encoder.right_encoder_val = encoder_b[1]
453                             # encoder.encoder_time = encoder_b[2]
454                             # fuse_gps_odom.fuse_encoder = encoder
455                             range_pub(range_b[0:8])
456                             # print "range_filled"
457                             # fuse_gps_odom.fill()
458                             # pub.publish(fuse_gps_odom.fuse_gps_odom_msg)
459                             # print "pub ok"
460                         except:
461                             print "encoder serial error"
462                     else:
463                         continue
464                 else:
465                     continue
466             else:
467                 continue
468         else:
469             print "!!!!!!!!!!!!!!!!!!!!"
470             continue
471
472
```

4.3.1.4 sound_range_init()

```
def serial_af_dox.sound_range_init ( )
```

超声波数据初始化

See also

range_max 超声波最远距离
range_min 超声笔最小距离(本来是0.290),range值小于等于最小值,输出无效,固最小值缩小
scan_time 超声波扫描时间
time_increment 超声波模拟激光模拟数据
angle_increment 超声波模拟激光模拟数据
angle_min 超声波的探测角(±30°)
angle_max 超声波的探测角(±30°)

```
267 def sound_range_init():
268     sound_range.range_max = 3.50
269     sound_range.range_min = 0.280
270     sound_range.scan_time = 0.4
271     sound_range.time_increment = 0.001
272     sound_range.angle_increment = 0.005
273     sound_range.angle_min = - 30 * math.pi / 180
274     sound_range.angle_max = 30 * math.pi / 180
275
276
```

4.3.2 Variable Documentation

4.3.2.1 crc_novetal

```
serial_af_dox.crc_novetal = crcmod.mkCrcFun(0x104C11DB7, 0, True, 0)
```

crc32校验程序(GPS)

4.3.2.2 fuse_gps_odom

```
serial_af_dox.fuse_gps_odom = Fuse_GPS_Odom()
```

4.3.2.3 mutex

```
serial_af_dox.mutex = threading.Lock()
```

线程锁

4.3.2.4 sound_range

```
serial_af_dox.sound_range = LaserScan()
```

超声波模拟激光输出类

4.3.2.5 sound_seq

```
int serial_af_dox.sound_seq = 0
```

超声波输出计数

Chapter 5

Class Documentation

5.1 fuse_G_A_dox.AMCL_unit Class Reference

AMCL存储类

Public Member Functions

- def __init__ (self)
  构造函数
- def update (self, odom_msg)
  里程计更新
- def fix (self, odom_map_msg)
  *AMCL*修正输入
- def output (self)
  输出

Public Attributes

- x
  x
- x_std
  *x*标准差
- y
  y
- y_std
  *y*标准差
- theta
  角度
- theta_std
  角度标准差

Private Attributes

- _theta_base
  累计角度
- _x_base
  累计*x*
- _y_base
  累计*y*

5.1.1 Detailed Description

AMCL存储类

5.1.2 Constructor & Destructor Documentation

5.1.2.1 __init__()

```
def fuse_G_A_dox.AMCL_unit.__init__ (
              self )
```

构造函数

```
114     def __init__(self):
115         self.x = 0
116         self.x_std = MAX_STD
117         self.y = 0
118         self.y_std = MAX_STD
119         self.theta = 0
120         self.theta_std = MAX_STD
121         self._theta_base = 0
122         self._x_base = 0
123         self._y_base = 0
124
```

5.1.3 Member Function Documentation

5.1.3.1 fix()

```
def fuse_G_A_dox.AMCL_unit.fix (
              self,
              odom_map_msg )
```

AMCL修正输入

将AMCL修正输入的map->odom和里程计输入融合生成实时的AMCL输出值(机器人在地图上的激光定位)

Parameters

odom_map_msg	AMCL的输入的map->odom

```
180     def fix(self, odom_map_msg):
181
182         # print "fixed"
183
186         _quat = [odom_map_msg.pose.pose.orientation.x, odom_map_msg.pose.pose.orientation.y,
      odom_map_msg.pose.pose.orientation.z,
187                  odom_map_msg.pose.pose.orientation.w]
188         self._theta_base = tft.euler_from_quaternion(_quat)[2]
189         self._x_base = odom_map_msg.pose.pose.position.x
190         self._y_base = odom_map_msg.pose.pose.position.y
191         self.x_std = math.sqrt(odom_map_msg.pose.covariance[0])
192         self.y_std = math.sqrt(odom_map_msg.pose.covariance[7])
193         self.theta_std = math.sqrt(odom_map_msg.pose.covariance[35])
194
```

5.1.3.2 output()

```
def fuse_G_A_dox.AMCL_unit.output (
              self )
```

输出

```
196     def output(self):
197         print "x: ", self.x, "| y: ", self.y, "| theta: ", self.theta
198         print "x_std: ", self.x_std, "| y_std: ", self.y_std, "| theta_std: ", self.theta_std
199
200
```

5.1.3.3 update()

```
def fuse_G_A_dox.AMCL_unit.update (
              self,
              odom_msg )
```

里程计更新

Parameters

odom_msg	ros里程计输入

See also

_pos 位置矩阵
_pose_base 位置基准矩阵
_pose_update 坐标系变换
_quat 四元数
flag_robot_move 机器人运动标志

```
138     def update(self, odom_msg):
139
140
143         _pos = np.array([[odom_msg.pose.pose.position.x], [odom_msg.pose.pose.position.y]])
144         odom_pose.x = _pos[0,0]
145         odom_pose.y = _pos[1,0]
146
149         _pos_base = np.array([[self._x_base], [self._y_base]], dtype=float)
150
153         _quat = [odom_msg.pose.pose.orientation.x, odom_msg.pose.pose.orientation.y,
    odom_msg.pose.pose.orientation.z,
154                  odom_msg.pose.pose.orientation.w]
155
158         _theta = tft.euler_from_quaternion(_quat)[2]
159
162         _pos_updated = np.dot(matrix_from_theta(self._theta_base), _pos) + _pos_base
163         odom_pose.theta = _theta
164         self.x = _pos_updated[0,0]
165         self.y = _pos_updated[1,0]
166         self.theta = _theta + self._theta_base
167         out_pose.set_odom(odom_pose.x, odom_pose.y, odom_pose.theta)
168         # print "updated"
169         # self.output()
170
```

5.1.4 Member Data Documentation

5.1.4.1 _theta_base

fuse_G_A_dox.AMCL_unit._theta_base [private]

累计角度

5.1.4.2 _x_base

fuse_G_A_dox.AMCL_unit._x_base [private]

累计x

5.1.4.3 _y_base

fuse_G_A_dox.AMCL_unit._y_base [private]

累计y

5.1.4.4 theta

fuse_G_A_dox.AMCL_unit.theta

角度

5.1.4.5 theta_std

`fuse_G_A_dox.AMCL_unit.theta_std`

角度标准差

5.1.4.6 x

`fuse_G_A_dox.AMCL_unit.x`

x

5.1.4.7 x_std

`fuse_G_A_dox.AMCL_unit.x_std`

x标准差

5.1.4.8 y

`fuse_G_A_dox.AMCL_unit.y`

y

5.1.4.9 y_std

`fuse_G_A_dox.AMCL_unit.y_std`

y标准差

The documentation for this class was generated from the following file:

- fuse_G_A_dox.py

5.2 select_A_dox.AMCL_unit Class Reference

AMCL存储类

Public Member Functions

- def __init__ (self)
  构造函数
- def update (self, odom_msg)
  里程计更新
- def fix (self, odom_map_msg)
  *AMCL*修正输入
- def lock (self)
  将*AMCL*输出结果置为不可信
- def output (self)
  输出

Public Attributes

- x
  x
- x_std
  *x*标准差
- y
  y
- y_std
  *y*标准差
- theta
  角度
- theta_std
  角度标准差

Private Attributes

- _theta_base
  累计角度
- _x_base
  累计*x*
- _y_base
  累计*y*

5.2.1 Detailed Description

AMCL存储类

5.2.2 Constructor & Destructor Documentation

5.2.2.1 __init__()

```
def select_A_dox.AMCL_unit.__init__ (
              self )
```

构造函数

```
163     def __init__(self):
164         self.x = 0
165         self.x_std = MAX_STD
166         self.y = 0
167         self.y_std = MAX_STD
168         self.theta = 0
169         self.theta_std = MAX_STD
170         self._theta_base = 0
171         self._x_base = 0
172         self._y_base = 0
173
```

5.2.3 Member Function Documentation

5.2.3.1 fix()

```
def select_A_dox.AMCL_unit.fix (
              self,
              odom_map_msg )
```

AMCL修正输入

将AMCL修正输入的map->odom和里程计输入融合生成实时的AMCL输出值(机器人在地图上的激光定位)

Parameters

odom_map_msg	AMCL的输入的map->odom

```
236     def fix(self, odom_map_msg):
237
238         # print "fixed"
239
242         _quat = [odom_map_msg.pose.pose.orientation.x, odom_map_msg.pose.pose.orientation.y,
    odom_map_msg.pose.pose.orientation.z,
243                  odom_map_msg.pose.pose.orientation.w]
244         self._theta_base = tft.euler_from_quaternion(_quat)[2]
245         self._x_base = odom_map_msg.pose.pose.position.x
246         self._y_base = odom_map_msg.pose.pose.position.y
247         self.x_std = math.sqrt(odom_map_msg.pose.covariance[0])
248         self.y_std = math.sqrt(odom_map_msg.pose.covariance[7])
249         self.theta_std = math.sqrt(odom_map_msg.pose.covariance[35])
250
```

5.2.3.2 lock()

```
def select_A_dox.AMCL_unit.lock (
              self )
```

将AMCL输出结果置为不可信

```
252     def lock(self):
253         self.x_std = MAX_STD
254         self.y_std = MAX_STD
255         self.theta_std = MAX_STD
256
```

5.2.3.3 output()

```
def select_A_dox.AMCL_unit.output (
              self )
```

输出

```
258     def output(self):
259         print "x: ", self.x, "| y: ", self.y, "| theta: ", self.theta
260         print "x_std: ", self.x_std, "| y_std: ", self.y_std, "| theta_std: ", self.theta_std
261
262
```

5.2.3.4 update()

```
def select_A_dox.AMCL_unit.update (
              self,
              odom_msg )
```

里程计更新

Parameters

odom_msg	ros里程计输入

See also

_pos 位置矩阵
_pose_base 位置基准矩阵
_pose_update 坐标系变换
_quat 四元数
flag_robot_move 机器人运动标志

```
187     def update(self, odom_msg):
188
189         global flag_robot_move
190         odom_msg = Odometry()
191
194         _pos = np.array([[odom_msg.pose.pose.position.x], [odom_msg.pose.pose.position.y]])
195         odom_pose.x = _pos[0, 0]
196         odom_pose.y = _pos[1, 0]
197
200         _pos_base = np.array([[self._x_base], [self._y_base]], dtype=float)
```

```
201
204         _quat = [odom_msg.pose.pose.orientation.x, odom_msg.pose.pose.orientation.y,
    odom_msg.pose.pose.orientation.z,
205                  odom_msg.pose.pose.orientation.w]
206
209         _theta = tft.euler_from_quaternion(_quat)[2]
210
213         _pos_updated = np.dot(matrix_from_theta(self._theta_base), _pos) + _pos_base
214         odom_pose.theta = _theta
215         self.x = _pos_updated[0, 0]
216         self.y = _pos_updated[1, 0]
217         self.theta = _theta + self._theta_base
218         out_pose.set_odom(odom_pose.x, odom_pose.y, odom_pose.theta)
219
220         if odom_msg.twist.twist.linear.x ** 2 + odom_msg.twist.twist.linear.y ** 2 <= STATIC_LIMIT_LINEAR *
    * 2 and \
221                 math.fabs(odom_msg.twist.twist.angular.z) <= STATIC_LIMIT_ANGULAR:
222             flag_robot_move = 0
223         else:
224             flag_robot_move = 1
225         # print "updated"
226         # self.output()
227
```

5.2.4 Member Data Documentation

5.2.4.1 _theta_base

select_A_dox.AMCL_unit._theta_base [private]

累计角度

5.2.4.2 _x_base

select_A_dox.AMCL_unit._x_base [private]

累计x

5.2.4.3 _y_base

select_A_dox.AMCL_unit._y_base [private]

累计y

5.2.4.4 theta

select_A_dox.AMCL_unit.theta

角度

5.2.4.5 theta_std

`select_A_dox.AMCL_unit.theta_std`

角度标准差

5.2.4.6 x

`select_A_dox.AMCL_unit.x`

x

5.2.4.7 x_std

`select_A_dox.AMCL_unit.x_std`

x标准差

5.2.4.8 y

`select_A_dox.AMCL_unit.y`

y

5.2.4.9 y_std

`select_A_dox.AMCL_unit.y_std`

y标准差

The documentation for this class was generated from the following file:

- select_A_dox.py

5.3 serial_af_dox.Encoder Class Reference

类用来存储里程计信息

Public Member Functions

- def __init__ (self)
  构造函数
- def plot (self)
  输出

Public Attributes

- left_encoder_val
  左里程计值
- right_encoder_val
  右里程计值
- encoder_time
  同步里程计时间

5.3.1 Detailed Description

类用来存储里程计信息

5.3.2 Constructor & Destructor Documentation

5.3.2.1 __init__()

```
def serial_af_dox.Encoder.__init__ (
            self )
```

构造函数

```
232     def __init__(self):
233         self.left_encoder_val = 0
234         self.right_encoder_val = 0
235         self.encoder_time = 0
236
```

5.3.3 Member Function Documentation

5.3.3.1 plot()

```
def serial_af_dox.Encoder.plot (
            self )
```

输出

```
238     def plot(self):
239         print "left_encoder_val", str(self.left_encoder_val)
240         print "right_encoder_val", str(self.right_encoder_val)
241         print "encoder_time", str(self.encoder_time)
242
243
244
```

5.3.4 Member Data Documentation

5.3.4.1 encoder_time

`serial_af_dox.Encoder.encoder_time`

同步里程计时间

5.3.4.2 left_encoder_val

`serial_af_dox.Encoder.left_encoder_val`

左里程计值

5.3.4.3 right_encoder_val

`serial_af_dox.Encoder.right_encoder_val`

右里程计值

The documentation for this class was generated from the following file:

- serial_af_dox.py

5.4 serial_af_dox.Fuse_GPS_Odom Class Reference

class Fuse_GPS_Odom 用来存储GPS和Encoder的信息

Public Member Functions

- def __init__ (self)
  构造函数
- def fill (self)
  将*GPS*和*Encoder*的信息添加到类中

Public Attributes

- fuse_gps
  *GPS*类存储值
- fuse_encoder
  *Encoder*类存储值
- fuse_gps_odom_msg
  本类融合两种存储值

5.4.1 Detailed Description

class Fuse_GPS_Odom 用来存储GPS和Encoder的信息

5.4.2 Constructor & Destructor Documentation

5.4.2.1 __init__()

```
def serial_af_dox.Fuse_GPS_Odom.__init__ (
              self )
```

构造函数

```
93     def __init__(self):
94         self.fuse_gps = GPS()
95         self.fuse_encoder = Encoder()
96         self.fuse_gps_odom_msg = Fuse_G_O()
97         self.fill()
98
```

5.4.3 Member Function Documentation

5.4.3.1 fill()

```
def serial_af_dox.Fuse_GPS_Odom.fill (
              self )
```

将GPS和Encoder的信息添加到类中

```
100     def fill(self):
101         self.fuse_gps_odom_msg.Slo_stat = self.fuse_gps.Slo_stat
102         self.fuse_gps_odom_msg.Pos_type = self.fuse_gps.Pos_type
103         self.fuse_gps_odom_msg.lat = self.fuse_gps.lat
104         self.fuse_gps_odom_msg.lat_std = self.fuse_gps.lat_std
105         self.fuse_gps_odom_msg.lon = self.fuse_gps.lon
106         self.fuse_gps_odom_msg.lon_std = self.fuse_gps.lon_std
107         self.fuse_gps_odom_msg.hgt = self.fuse_gps.hgt
108         self.fuse_gps_odom_msg.hgt_std = self.fuse_gps.hgt_std
109         self.fuse_gps_odom_msg.SVs = self.fuse_gps.SVs
110         self.fuse_gps_odom_msg.solnSVs = self.fuse_gps.solnSVs
111         self.fuse_gps_odom_msg.heading = self.fuse_gps.heading
112         self.fuse_gps_odom_msg.hdg_std = self.fuse_gps.hdg_std
113         self.fuse_gps_odom_msg.hor_spd = self.fuse_gps.hor_spd
114         self.fuse_gps_odom_msg.Trk_gnd = self.fuse_gps.Trk_gnd
115         self.fuse_gps_odom_msg.Vert_spd = self.fuse_gps.Vert_spd
116         self.fuse_gps_odom_msg.gps_time = self.fuse_gps.gps_time
117
118         self.fuse_gps_odom_msg.left_encoder_val = self.fuse_encoder.left_encoder_val
119         self.fuse_gps_odom_msg.right_encoder_val = self.fuse_encoder.right_encoder_val
120         self.fuse_gps_odom_msg.encoder_time = self.fuse_encoder.encoder_time
121
122
```

5.4.4 Member Data Documentation

5.4.4.1 fuse_encoder

`serial_af_dox.Fuse_GPS_Odom.fuse_encoder`

Encoder类存储值

5.4.4.2 fuse_gps

`serial_af_dox.Fuse_GPS_Odom.fuse_gps`

GPS类存储值

5.4.4.3 fuse_gps_odom_msg

`serial_af_dox.Fuse_GPS_Odom.fuse_gps_odom_msg`

本类融合两种存储值

The documentation for this class was generated from the following file:

- serial_af_dox.py

5.5 serial_af_dox.GPS Class Reference

存储GPS信息的类

Public Member Functions

- def __init__ (self)
  构造函数
- def plot (self)
  输出

Public Attributes

- Slo_stat
  卫星状态
- Pos_type
  定位类型
- lat
  维度
- lat_std
  纬度标准差
- lon
  经度
- lon_std
  经度标准差
- hgt
  高度
- hgt_std
  高度标准差
- SVs
  卫星数
- solnSVs
  参与解算卫星数
- heading
  航向
- hdg_std
  航向标准差
- hor_spd

- Trk_gnd

- Vert_spd

- gps_time
  *gps*校准时间

5.5.1 Detailed Description

存储GPS信息的类

5.5.2 Constructor & Destructor Documentation

5.5.2.1 __init__()

```
def serial_af_dox.GPS.__init__ (
              self )
```

构造函数

```
178     def __init__(self):
179         self.Slo_stat = 0
180         self.Pos_type = 0
181         self.lat = 0
182         self.lat_std = 0
183         self.lon = 0
184         self.lon_std = 0
185         self.hgt = 0
186         self.hgt_std = 0
187         self.SVs = 0
188         self.solnSVs = 0
189         self.heading = 0
190         self.hdg_std = 0
191         self.hor_spd = 0
192         self.Trk_gnd = 0
193         self.Vert_spd = 0
194         self.gps_time = 0
195
```

5.5.3 Member Function Documentation

5.5.3.1 plot()

```
def serial_af_dox.GPS.plot (
              self )
```

输出

```
197     def plot(self):
198         print "Slo_stat" + str(self.Slo_stat)
199         print "Pos_type" + str(self.Pos_type)
200         print "lat" + str(self.lat)
201         print "lat_std"+str(self.lat_std)
202         print "lon"+str(self.lon)
203         print "lon_std"+str(self.lon_std)
204         print "hgt"+str(self.hgt)
205         print "hgt_std"+str(self.hgt_std)
206         print "SVs"+str(self.SVs)
207         print "solnSVs"+str(self.solnSVs)
208         print "heading"+str(self.heading)
209         print "hdg_std"+str(self.hdg_std)
210         print "hor_spd"+str(self.hor_spd)
211         print "Trk_gnd"+str(self.Trk_gnd)
212         print "Vert_spd"+str(self.Vert_spd)
213         print "gps_time"+str(self.gps_time)
214
215
```

5.5.4 Member Data Documentation

5.5.4.1 gps_time

```
serial_af_dox.GPS.gps_time
```

gps校准时间

5.5.4.2 hdg_std

```
serial_af_dox.GPS.hdg_std
```

航向标准差

5.5.4.3 heading

```
serial_af_dox.GPS.heading
```

航向

5.5.4.4 hgt

```
serial_af_dox.GPS.hgt
```

高度

5.5.4.5 hgt_std

```
serial_af_dox.GPS.hgt_std
```

高度标准差

5.5.4.6 hor_spd

```
serial_af_dox.GPS.hor_spd
```

5.5.4.7 lat

serial_af_dox.GPS.lat

维度

5.5.4.8 lat_std

serial_af_dox.GPS.lat_std

纬度标准差

5.5.4.9 lon

serial_af_dox.GPS.lon

经度

5.5.4.10 lon_std

serial_af_dox.GPS.lon_std

经度标准差

5.5.4.11 Pos_type

serial_af_dox.GPS.Pos_type

定位类型

5.5.4.12 Slo_stat

serial_af_dox.GPS.Slo_stat

卫星状态

5.5.4.13 solnSVs

`serial_af_dox.GPS.solnSVs`

参与解算卫星数

5.5.4.14 SVs

`serial_af_dox.GPS.SVs`

卫星数

5.5.4.15 Trk_gnd

`serial_af_dox.GPS.Trk_gnd`

5.5.4.16 Vert_spd

`serial_af_dox.GPS.Vert_spd`

The documentation for this class was generated from the following file:

- serial_af_dox.py

5.6 select_A_dox.GPS_unit Class Reference

GPS数据存储类

Public Member Functions

- def __init__ (self)
  构造函数
- def update (self, gps_msg)
  *GPS*数据更新函数
- def lock (self)
  将*GPS*输出结果置为不可信
- def output (self)
  输出

Public Attributes

- x
  x
- x_std
  *x*标准差
- y
  y
- y_std
  *y*标准差
- theta
  角度
- theta_std
  角度标准差

5.6.1 Detailed Description

GPS数据存储类

5.6.2 Constructor & Destructor Documentation

5.6.2.1 __init__()

```
def select_A_dox.GPS_unit.__init__ (
                self )
```

构造函数

```
329     def __init__(self):
330         self.x = 0
331         self.x_std = MAX_STD
332         self.y = 0
333         self.y_std = MAX_STD
334         self.theta = 0
335         self.theta_std = MAX_STD
336
```

5.6.3 Member Function Documentation

5.6.3.1 lock()

```
def select_A_dox.GPS_unit.lock (
              self )
```

将GPS输出结果置为不可信

```
352     def lock(self):
353         self.x_std = MAX_STD
354         self.y_std = MAX_STD
355         self.theta_std = MAX_STD
356
```

5.6.3.2 output()

```
def select_A_dox.GPS_unit.output (
              self )
```

输出

```
358     def output(self):
359         print "x: ", self.x, "| y: ", self.y, "| theta: ", self.theta
360         print "x_std: ", self.x_std, "| y_std: ", self.y_std, "| theta_std: ", self.theta_std
361
362
```

5.6.3.3 update()

```
def select_A_dox.GPS_unit.update (
              self,
              gps_msg )
```

GPS数据更新函数

Parameters

gps_msg	GPS数据输入

```
343     def update(self, gps_msg):
344         self.x = gps_msg.X
345         self.x_std = gps_msg.X_std
346         self.y = gps_msg.Y
347         self.y_std = gps_msg.Y_std
```

```
348         self.theta = gps_msg.Hdg
349         self.theta_std = gps_msg.Hdg_std
350
```

5.6.4 Member Data Documentation

5.6.4.1 theta

`select_A_dox.GPS_unit.theta`

角度

5.6.4.2 theta_std

`select_A_dox.GPS_unit.theta_std`

角度标准差

5.6.4.3 x

`select_A_dox.GPS_unit.x`

x

5.6.4.4 x_std

`select_A_dox.GPS_unit.x_std`

x标准差

5.6.4.5 y

`select_A_dox.GPS_unit.y`

y

5.6.4.6 y_std

`select_A_dox.GPS_unit.y_std`

y标准差

The documentation for this class was generated from the following file:

- select_A_dox.py

5.7 fuse_G_A_dox.GPS_unit Class Reference

GPS数据存储类

Public Member Functions

- def __init__ (self)
  构造函数
- def update (self, gps_msg)
  *GPS*数据更新函数
- def output (self)
  输出

Public Attributes

- x
  x
- x_std
  *x*标准差
- y
  y
- y_std
  *y*标准差
- theta
  角度
- theta_std
  角度标准差

5.7.1 Detailed Description

GPS数据存储类

5.7.2 Constructor & Destructor Documentation

5.7.2.1 __init__()

```
def fuse_G_A_dox.GPS_unit.__init__ (
              self )
```

构造函数

```
267     def __init__(self):
268         self.x = 0
269         self.x_std = MAX_STD
270         self.y = 0
271         self.y_std = MAX_STD
272         self.theta = 0
273         self.theta_std = MAX_STD
274
```

5.7.3 Member Function Documentation

5.7.3.1 output()

```
def fuse_G_A_dox.GPS_unit.output (
              self )
```

输出

```
290     def output(self):
291         print "x: ", self.x, "| y: ", self.y, "| theta: ", self.theta
292         print "x_std: ", self.x_std, "| y_std: ", self.y_std, "| theta_std: ", self.theta_std
293
294
```

5.7.3.2 update()

```
def fuse_G_A_dox.GPS_unit.update (
              self,
              gps_msg )
```

GPS数据更新函数

Parameters

gps_msg	GPS数据输入

```
281     def update(self, gps_msg):
282         self.x = gps_msg.X
283         self.x_std = gps_msg.X_std
284         self.y = gps_msg.Y
285         self.y_std = gps_msg.Y_std
286         self.theta = gps_msg.Hdg
287         self.theta_std = gps_msg.Hdg_std
288
```

5.7.4 Member Data Documentation

5.7.4.1 theta

fuse_G_A_dox.GPS_unit.theta

角度

5.7.4.2 theta_std

fuse_G_A_dox.GPS_unit.theta_std

角度标准差

5.7.4.3 x

fuse_G_A_dox.GPS_unit.x

x

5.7.4.4 x_std

fuse_G_A_dox.GPS_unit.x_std

x标准差

5.7.4.5 y

fuse_G_A_dox.GPS_unit.y

y

5.7.4.6 y_std

`fuse_G_A_dox.GPS_unit.y_std`

y标准差

The documentation for this class was generated from the following file:

- fuse_G_A_dox.py

5.8 fuse_G_A_dox.Odom_unit Class Reference

里程计数据存储类

Public Member Functions

- def __init__ (self)
  构造函数
- def output (self)
  输出

Public Attributes

- x
  x
- x_std
  *x*标准差
- y
  y
- y_std
  *y*标准差
- theta
  角度
- theta_std
  角度标准差

5.8.1 Detailed Description

里程计数据存储类

5.8.2 Constructor & Destructor Documentation

5.8.2.1 __init__()

```
def fuse_G_A_dox.Odom_unit.__init__ (
              self )
```

构造函数

```
227     def __init__(self):
228         self.x = 0
229         self.x_std = MAX_STD
230         self.y = 0
231         self.y_std = MAX_STD
232         self.theta = 0
233         self.theta_std = MAX_STD
234
```

5.8.3 Member Function Documentation

5.8.3.1 output()

```
def fuse_G_A_dox.Odom_unit.output (
              self )
```

输出

```
236     def output(self):
237         print "x: ", self.x, "| y: ", self.y, "| theta: ", self.theta
238         print "x_std: ", self.x_std, "| y_std: ", self.y_std, "| theta_std: ", self.theta_std
239
240
```

5.8.4 Member Data Documentation

5.8.4.1 theta

```
fuse_G_A_dox.Odom_unit.theta
```

角度

5.8.4.2 theta_std

```
fuse_G_A_dox.Odom_unit.theta_std
```

角度标准差

5.8.4.3 x

fuse_G_A_dox.Odom_unit.x

x

5.8.4.4 x_std

fuse_G_A_dox.Odom_unit.x_std

x标准差

5.8.4.5 y

fuse_G_A_dox.Odom_unit.y

y

5.8.4.6 y_std

fuse_G_A_dox.Odom_unit.y_std

y标准差

The documentation for this class was generated from the following file:

- fuse_G_A_dox.py

5.9 select_A_dox.Odom_unit Class Reference

里程计数据存储类

Public Member Functions

- def __init__ (self)
  构造函数
- def output (self)
  输出

Public Attributes

- x
  x
- x_std
  *x*标准差
- y
  y
- y_std
  *y*标准差
- theta
  角度
- theta_std
  角度标准差

5.9.1 Detailed Description

里程计数据存储类

5.9.2 Constructor & Destructor Documentation

5.9.2.1 __init__()

```
def select_A_dox.Odom_unit.__init__ (
              self )
```

构造函数

```
289     def __init__(self):
290         self.x = 0
291         self.x_std = MAX_STD
292         self.y = 0
293         self.y_std = MAX_STD
294         self.theta = 0
295         self.theta_std = MAX_STD
296
```

5.9.3 Member Function Documentation

5.9.3.1 output()

```
def select_A_dox.Odom_unit.output (
              self )
```

输出

```
298     def output(self):
299         print "x: ", self.x, "| y: ", self.y, "| theta: ", self.theta
300         print "x_std: ", self.x_std, "| y_std: ", self.y_std, "| theta_std: ", self.theta_std
301
302
```

5.9.4 Member Data Documentation

5.9.4.1 theta

`select_A_dox.Odom_unit.theta`

角度

5.9.4.2 theta_std

`select_A_dox.Odom_unit.theta_std`

角度标准差

5.9.4.3 x

`select_A_dox.Odom_unit.x`

x

5.9.4.4 x_std

`select_A_dox.Odom_unit.x_std`

x标准差

5.9.4.5 y

`select_A_dox.Odom_unit.y`

y

5.9.4.6 y_std

```
select_A_dox.Odom_unit.y_std
```

y标准差

The documentation for this class was generated from the following file:

- select_A_dox.py

5.10 select_A_dox.SELECTED Class Reference

筛选结果类

Public Member Functions

- def __init__ (self)
  构造函数
- def set_map (self, x, y, theta)
  将选则的最终结果数据放在输出中
- def set_odom (self, x, y, theta)
  将里程计数据放在输出中
- def output (self)
  输出

Public Attributes

- choice
- x
  x
- x_choice
- y
  y
- y_choice
- theta
  角度
- theta_choice
- fuse_tf
  *ros*输出格式

5.10.1 Detailed Description

筛选结果类

5.10.2 Constructor & Destructor Documentation

5.10.2.1 __init__()

```
def select_A_dox.SELECTED.__init__ (
              self )
```

构造函数

```
397     def __init__(self):
398         self.choice = {0: "GPS", 1: "AMCL", 2: "NONE"}
399         self.x = 0
400         self.x_choice = self.choice[2]
401         self.y = 0
402         self.y_choice = self.choice[2]
403         self.theta = 0
404         self.theta_choice = self.choice[2]
405         self.fuse_tf = Fuse_tf()
406         self.fuse_tf.map_x = 0
407         self.fuse_tf.map_y = 0
408         self.fuse_tf.map_theta = 0
409         self.fuse_tf.odom_x = 0
410         self.fuse_tf.odom_y = 0
411         self.fuse_tf.odom_theta = 0
412
```

5.10.3 Member Function Documentation

5.10.3.1 output()

```
def select_A_dox.SELECTED.output (
              self )
```

输出

```
426     def output(self):
427         print "x: ", self.x, "| y: ", self.y, "| theta: ", self.theta
428         print "x_choice: ", self.x_choice, "| y_choice: ", self.y_choice, "| theta_choice: ", self.
      theta_choice
429
430
```

5.10.3.2 set_map()

```
def select_A_dox.SELECTED.set_map (
              self,
              x,
              y,
              theta )
```

将选则的最终结果数据放在输出中

```
414     def set_map(self, x, y, theta):
415         self.fuse_tf.map_x = x
416         self.fuse_tf.map_y = y
417         self.fuse_tf.map_theta = theta
418
```

5.10.3.3 set_odom()

```
def select_A_dox.SELECTED.set_odom (
               self,
               x,
               y,
               theta )
```

将里程计数据放在输出中

```
420     def set_odom(self,x, y, theta):
421         self.fuse_tf.odom_x = x
422         self.fuse_tf.odom_y = y
423         self.fuse_tf.odom_theta = theta
424
```

5.10.4 Member Data Documentation

5.10.4.1 choice

```
select_A_dox.SELECTED.choice
```

5.10.4.2 fuse_tf

```
select_A_dox.SELECTED.fuse_tf
```

ros输出格式

5.10.4.3 theta

```
select_A_dox.SELECTED.theta
```

角度

5.10.4.4 theta_choice

```
select_A_dox.SELECTED.theta_choice
```

5.10.4.5 x

```
select_A_dox.SELECTED.x
```

x

5.10.4.6 x_choice

```
select_A_dox.SELECTED.x_choice
```

5.10.4.7 y

```
select_A_dox.SELECTED.y
```

y

5.10.4.8 y_choice

```
select_A_dox.SELECTED.y_choice
```

The documentation for this class was generated from the following file:

- select_A_dox.py

5.11 fuse_G_A_dox.SELECTED Class Reference

筛选结果类

Public Member Functions

- def __init__ (self)
  构造函数
- def set_map (self, x, y, theta)
  将选则的最终结果数据放在输出中
- def set_odom (self, x, y, theta)
  将里程计数据放在输出中
- def output (self)
  输出

Public Attributes

- choice
- x

  x
- x_choice
- y

  y
- y_choice
- theta

  角度
- theta_choice
- fuse_tf

  *ros*输出格式

5.11.1 Detailed Description

筛选结果类

5.11.2 Constructor & Destructor Documentation

5.11.2.1 __init__()

```
def fuse_G_A_dox.SELECTED.__init__ (
            self )
```

构造函数

```
329     def __init__(self):
330         self.choice = {0: "GPS", 1: "AMCL", 2: "NONE"}
331         self.x = 0
332         self.x_choice = self.choice[2]
333         self.y = 0
334         self.y_choice = self.choice[2]
335         self.theta = 0
336         self.theta_choice = self.choice[2]
337         self.fuse_tf = Fuse_tf()
338         self.fuse_tf.map_x = 0
339         self.fuse_tf.map_y = 0
340         self.fuse_tf.map_theta = 0
341         self.fuse_tf.odom_x = 0
342         self.fuse_tf.odom_y = 0
343         self.fuse_tf.odom_theta = 0
344
```

5.11.3 Member Function Documentation

5.11.3.1 output()

```
def fuse_G_A_dox.SELECTED.output (
              self )
```

输出

```
358     def output(self):
359         print "x: ", self.x, "| y: ", self.y, "| theta: ", self.theta
360         print "x_choice: ", self.x_choice, "| y_choice: ", self.y_choice, "| theta_choice: ", self.
      theta_choice
361
362
```

5.11.3.2 set_map()

```
def fuse_G_A_dox.SELECTED.set_map (
              self,
              x,
              y,
              theta )
```

将选则的最终结果数据放在输出中

```
346     def set_map(self, x, y, theta):
347         self.fuse_tf.map_x = x
348         self.fuse_tf.map_y = y
349         self.fuse_tf.map_theta = theta
350
```

5.11.3.3 set_odom()

```
def fuse_G_A_dox.SELECTED.set_odom (
              self,
              x,
              y,
              theta )
```

将里程计数据放在输出中

```
352     def set_odom(self,x, y, theta):
353         self.fuse_tf.odom_x = x
354         self.fuse_tf.odom_y = y
355         self.fuse_tf.odom_theta = theta
356
```

5.11.4 Member Data Documentation

5.11.4.1 choice

fuse_G_A_dox.SELECTED.choice

5.11.4.2 fuse_tf

fuse_G_A_dox.SELECTED.fuse_tf

ros输出格式

5.11.4.3 theta

fuse_G_A_dox.SELECTED.theta

角度

5.11.4.4 theta_choice

fuse_G_A_dox.SELECTED.theta_choice

5.11.4.5 x

fuse_G_A_dox.SELECTED.x

x

5.11.4.6 x_choice

fuse_G_A_dox.SELECTED.x_choice

5.11.4.7 y

fuse_G_A_dox.SELECTED.y

y

5.11.4.8 y_choice

fuse_G_A_dox.SELECTED.y_choice

The documentation for this class was generated from the following file:

- fuse_G_A_dox.py

Chapter 6

File Documentation

6.1 fuse_G_A_dox.py File Reference

Classes

- class fuse_G_A_dox.AMCL_unit
  *AMCL*存储类
- class fuse_G_A_dox.Odom_unit
  里程计数据存储类
- class fuse_G_A_dox.GPS_unit
  *GPS*数据存储类
- class fuse_G_A_dox.SELECTED
  筛选结果类

Namespaces

- fuse_G_A_dox

Functions

- def fuse_G_A_dox.matrix_from_theta (theta)
  角度生成旋转矩阵*(2x2)*
- def fuse_G_A_dox.theta_from_matrix (R)
  旋转矩阵*(2x2)*转成角度
- def fuse_G_A_dox.sendTransform ()
  发送*topic*
- def fuse_G_A_dox.tran_mat44 (trans)
  将*ros*格式中的*Transform*数据转为矩阵*(4X4)*
- def fuse_G_A_dox.tran_theta_T (trans)
  将*ros*格式中的*Transform*数据转为矩阵*(2X2)*
- def fuse_G_A_dox.normalization (a, b)
  归一化
- def fuse_G_A_dox.make_choice ()
  筛选函数

Variables

- int fuse_G_A_dox.MAX_STD = 999999999
  最大的标准差
- fuse_G_A_dox.tfBuffer = tf2_ros.Buffer()
- fuse_G_A_dox.listener = tf2_ros.TransformListener(tfBuffer)
- fuse_G_A_dox.amcl_pose = AMCL_unit()
- fuse_G_A_dox.gps_pose = GPS_unit()
- fuse_G_A_dox.odom_pose = Odom_unit()
- fuse_G_A_dox.out_pose = SELECTED()
- fuse_G_A_dox.pub = rospy.Publisher('/fuse/tf', Fuse_tf, queue_size=100)
- fuse_G_A_dox.t2 = threading.Thread(target=sendTransform)
- fuse_G_A_dox.t1 = threading.Thread(target=make_choice)

6.2 select_A_dox.py File Reference

Classes

- class select_A_dox.AMCL_unit
  *AMCL*存储类
- class select_A_dox.Odom_unit
  里程计数据存储类
- class select_A_dox.GPS_unit
  *GPS*数据存储类
- class select_A_dox.SELECTED
  筛选结果类

Namespaces

- select_A_dox

Functions

- def select_A_dox.normalize (z)
  归一化
- def select_A_dox.angle_diff (a, b)
  角度差值
- def select_A_dox.matrix_from_theta (theta)
  角度生成旋转矩阵*(2x2)*
- def select_A_dox.theta_from_matrix (R)
  旋转矩阵*(2x2)*转成角度
- def select_A_dox.sendTransform (out_pose)
  发送*TF(topic)*
- def select_A_dox.tran_mat44 (trans)
  将*ros*格式中的*Transform*数据转为矩阵*(4X4)*
- def select_A_dox.tran_theta_T (trans)
  将*ros*格式中的*Transform*数据转为矩阵*(2X2)*
- def select_A_dox.make_choice ()

筛选函数

- def select_A_dox.amcl_initial_update (x_std=1, y_std=1, theta_std=1)

  *AMCL*初始化函数
- def select_A_dox.amcl_initial_callback (msg)

  AMCL 初始化回调函数
- def select_A_dox.amcl_global_update ()

  *AMCL*全局撒粒子
- def select_A_dox.amcl_nomotion_update ()

  *AMCL*未运动状态下粒子更新
- def select_A_dox.amcl_recovery ()

  *AMCL*恢复状态过程
- def select_A_dox.amcl_process ()

  *AMCL*线程
- def select_A_dox.main ()

  主函数

Variables

- int select_A_dox.MAX_STD = 999999999

  最大的标准差
- int select_A_dox.flag_robot_move = 0

  机器人是否运动标志
- float select_A_dox.STATIC_LIMIT_LINEAR = 0.05

  判断机器人是否运动的速度限制
- float select_A_dox.STATIC_LIMIT_ANGULAR = 0.05

  判断机器人是否运动的角度限制

6.3 serial_af_dox.py File Reference

Classes

- class serial_af_dox.Fuse_GPS_Odom

  class Fuse_GPS_Odom 用来存储*GPS*和*Encoder*的信息
- class serial_af_dox.GPS

  存储*GPS*信息的类
- class serial_af_dox.Encoder

  类用来存储里程计信息

Namespaces

- serial_af_dox

Functions

- def serial_af_dox.sound_range_init ()
  超声波数据初始化
- def serial_af_dox.range_pub (range_msg)
  *range_pub range*的激光输出函数
- def serial_af_dox.serial_process (port='/dev/pts/27', baudrate=460800, timeout=1)
  serial_process 串口线程
- def serial_af_dox.main ()
  *main*函数

Variables

- serial_af_dox.crc_novetal = crcmod.mkCrcFun(0x104C11DB7, 0, True, 0)
  *crc32*校验程序*(GPS)*
- serial_af_dox.mutex = threading.Lock()
  线程锁
- serial_af_dox.fuse_gps_odom = Fuse_GPS_Odom()
- serial_af_dox.sound_range = LaserScan()
  超声波模拟激光输出类
- int serial_af_dox.sound_seq = 0
  超声波输出计数

Index