robot_localization package

⇒ Collection of state estimation nodes.

↓

{ Each of which is an implementation of a nonlinear state estimation for robot moving in 3D space }

(1) ekf_localization_node

(2) ukf_localization_node

⇒ All the state estimation nodes in robot_localization share common features:

1) Fusion of arbitrary numbers of sensor.

2) Support for multiple ros message type:
   - → nav_msgs/Odometry
   - → sensor_msgs/Imu
   - → geometry_msgs/PoseWithCovarianceStamped
   - → geometry_msgs/TwistWithCovarianceStamped

3) Per-sensor input customization

4) All state estimation nodes track the 15-dimensional state of the vehicle:

$$(X, Y, Z, \text{roll}, \text{pitch}, \text{yaw}, \dot{X}, \dot{Y}, \dot{Z}, \ddot{x}, \ddot{y}, \ddot{z}, \dot{\text{roll}}, \dot{\text{Pitch}}, \dot{\text{Yaw}})$$

★ State Estimation Nodes

ekf_localization_node

→ Implementation of Extended Kalman filter.
→ It uses an omnidirectional motion model to project the state forward in time.

ukf_localization_node

→ Implementation of an unscented Kalman filter.

→ It uses a set of carefully selected sigma points to project the state through the same motion model that is used in the EKF.

→ This eliminates the use of Jacobian matrices and makes the filter more stable.

↳ However, it is also more computationally taxing than ekf_localization_node.

Parameters common to ekf_localization_node and ukf_localization_node

① ~frequency (Hz)

→ Hz at which filter produces a state estimate.

② ~sensor_timeout (seconds)

→ Period after which we consider any sensor have timed out.

→ In this event we carry out a predict cycle on the EKF without correcting it.

③ ~two_d_mode

→ If your robot is operating in a planar environment and you're comfortable with ignoring the subtle variations in the ground then set this to true.

→ It will fuse 0 values for all 3D variables.

③ ~[frame]

~map_frame
~odom_frame
~base_link_frame
~base_link_output_frame
~world_frame

{Default}

→ These parameters define the operating "mode" for robot_localization.

* Set the map_frame, odom_frame and base_link_frame Parameters to the appropriate frame names for your system.
  - → If your system does not have a map_frame, just remove it, and make sure world_frame is set to the value of odom_frame.
  - → base_link_output_frame is optional and will default to the base_link_frame.

* If you are only fusing continuous position data such as wheel encoder odometry, visual odometry or IMU data, set world_frame to your odom frame value.

* If you are fusing global absolute position data that is subjected to discrete jumps then:
  - → Set your world_frame to your map_frame value.
  - → Make sure something else is generating the odom → base_link transform.

(5) ~transform_time_offset

→ Some packages required that your transforms are future-dated by a small time offset.

→ The value of this parameter will be added to the timestamp of map → odom or odom → base_link transform being generated by state estimation nodes.

(5) ~transform_timeout

→ The robot_localization package uses tf2's lookupTransform method to request transforms.

→ This parameter specifies how long we would like to wait if a transform is not available yet.

→ Default is set to 0 if not set.
  ↳ The value 0 means we just get us the latest available transform so we are not blocking the filter.

(7) ~[Sensor]

→ For each sensor, users need to define this parameter based on the message type.

Example:

~imu0: "robot/imu/data"
~odom0: "wheel_encoder/odometry"
~odom1: "visual_odometry/odometry"

→ The index for each parameter name is 0-based and must be defined sequentially.

8 ~[Sensor]_config

⇒ Specific parameters

~odomN_config

~twistN_config

~imuN_config

~poseN_config.

⇒ For each of the sensor messages defined above, user must specify what variables of those messages should be fused into the final state estimate.

```
~[Sensor]_config: [true,  true,  false,
                   false, false, true,
                   false, false, false,
                   true,  false, false,
                   false, false, false]
```

⇒ The order of the boolean values are:

X, Y, Z, roll, pitch, yaw, $\dot{X}, \dot{Y}, \dot{Z}$, $\dot{\text{roll}}$, $\dot{\text{pitch}}$, $\dot{\text{yaw}}$, $\ddot{X}, \ddot{Y}, \ddot{Z}$

⇒ The Specification is done in the frame_id of the sensor, not the world_frame or base_link_frame

9 ~[Sensor]_queue_size

→ Users can use these parameters to adjust the callback queue size for each sensor.

→ This is usefull if your frequency parameter value is much lower than your sensors frequency as it allows the filter to incorporate all measurements that arrived in between update cycles.

(10) ~[Sensor]_differential

⇒ For each of the sensor messages, that contains pose information, user can specify whether the pose variables should be integrated differentially.

⇒ If a given value is set to true, then for a measurement at time t from the sensor in question, we first substract the measurement at time t-1, and convert the resulting value to a velocity.

⇒ This setting is especially useful if your robot has two sources of absolute pose information.

- → If the variance on the input sources are not configured correctly, these measurements may get out of sync with one another and cause oscillation in the filter.
- → but by integrating one or both of them differentially, we avoide this scenario.

(11) ~[Sensor]_relative

⇒ If this parameter is set to true, then any measurement from this sensor will be fused relative to the first measurement received from that sensor.

(12) ~imuN_remove_gravitational_acceleration

⇒ If fusing accelerometer data from IMUs, this parameter determines whether or not acceleration due to gravity is removed from the acceleration measurement before fusing it.

→ This assumes that the IMU that is providing the acceleration data is also producing an absolute orientation.
↳ The orientation data is required to correctly remove gravitational acceleration.

(13) ~gravitational_acceleration

⇒ If imuN_remove_gravitational_acceleration is set to true, then this parameter determines the acceleration in Z due to gravity that will be removed from the IMU linear acceleration data.

→ Default is 9.80665 (m/s^2)

(14) ~initial_state

⇒ The state is given as a 15D vector of doubles, in the same order as the sensor configurations.

(15) ~publish_tf (Default: true)

→ Publish tf from the frame specified by the world_frame parameter to the frame specified by base_link_frame parameter.

(16) ~publish_acceleration

(17) ~print_diagnostics

Advanced Parameters

TODO

Parameters specific to ukf_localization_node

~alpha, ~kappa, ~beta

{ follows the nomenclature of the original paper }

Published Topics

1. odometry/filtered (nav_msgs/Odometry)
2. accel/filtered (geometry_msgs/AccelWithCovarianceStamped)

{If enabled}

Published Transforms

- If world_frame == odom_frame
  then
  odom_frame ⟵ base_link_frame

  else if world_frame == map_frame
  then
  map_frame ⟵ odom_frame

Services

Set Pose (robot_localization/SetPose)
