indigo | kinetic *Show EOL distros:*

Documentation Status

**Package Links**

- **Code API (http://docs.ros.org/kinetic/api/geonav_transform/html)**
- geonav_transform website (https://github.com/bsb808/geonav_transform)
- FAQ (http://answers.ros.org/questions/scope:all/sort:activity-desc/tags:geonav_transform/page:1/)
- Change List (/geonav_transform/ChangeList)
- Reviews (/geonav_transform/Reviews)

**Dependencies (13)**
**Jenkins jobs (2)**

# Package Summary

**✔ Continuous Integration** **✔ Documented**

The geonav_transform package

- Maintainer status: developed
- Maintainer: Brian Bingham <briansbingham AT gmail DOT com>
- Author: Brian Bingham <briansbingham AT gmail DOT com>
- License: GPL
- External website: https://github.com/bsb808/geonav_transform (https://github.com/bsb808/geonav_transform)
- Source: git https://github.com/bsb808/geonav_transform.git (https://github.com/bsb808/geonav_transform) (branch: master)

**Contents**



# Overview

The geonav_transform package includes the following

- The geonav_transform node (C++) to provide integration of geographic navigation (e.g., GPS) into ROS localization and navigation workflows.
- Utilities for conversion between 2D geographic/geodetic coordinates (lat/lon) and local coordinates (x/y), including two methods
    - AlvinXY
    - Geonav

# Installation

Since this package is currently under development, the standard git, catkin_make, source setup.bash workflow should be used to access the C++ nodes and Python utility modules.

# geonav_transform_node

The goal of this node is to simplify the integration of accurate/precise geographic navigation information (typically from a sensor) into the ROS localization and navigation workflows. To those ends, the geonav_tranform_node can perform the following functions:

- Takes incoming Odometry messages, typically from a sensor, that contain a geographic position and sensor-frame orientation and velocities. (Note, it would be nice to have a new message the specifies this type of input - using Odometry is a bit of a hack.)
- Transforms these message to new Odometry message that express the information in the following frames
  - utm
  - odom
- Broadcasts the following tf2 tranforms
  - utm->odom
  - odom->base_link

The use-case that motivated this project is integrating sensors that provide a GPS-aided INS solution (e.g., microstrain, advanced navigation, Xsens, etc.). This situation is analogous to using an ekf/ukf node from robot_localization package to fuse IMU and GPS information, but in this case the processing is done by the sensor. The purpose of this package is to allow integration of this type of sensor directly into the ROS navigation stack.

# 0.1 geonav_transform_node

ROS node to provide conversions (both data and tf transforms) between geodetic, utm and odom frames.

### 0.0.1 Subscribed Topics

`nav_odom` (nav_msgs/Odometry (http://docs.ros.org/api/nav_msgs/html/msg/Odometry.html))
: Message with geographic position and velocity data. The message is organized as described in Odometry in Geographic Coordinates (/geonav_transform#geoodom)

### 0.0.2 Published Topics

`geonav_odom` (nav_msgs/Odometry (http://docs.ros.org/api/nav_msgs/html/msg/Odometry.html))
: A nav_msgs/Odometry message in the local odom frame (relative to the datum)

`geonav_utm` (nav_msgs/Odometry (http://docs.ros.org/api/nav_msgs/html/msg/Odometry.html))
: A nav_msgs/Odometry message in the UTM frame

### 0.0.3 Parameters

`datum` (`three element array of doubles [Latitude, Logitude, Altitude]`)
: The origin of the local "odom" frame. Lat/Lon are in decimal degrees; altitude is in meters.

`frequency` (`float`, default: 10 Hz)
: The frequency of broadcasting the tf2 tranforms. The Odometry messages are published at the same rate as the incoming Odometry messages.

`broadcast_utm2odom_transform` (`bool`, default: True.)
: Whether or not to broadcast the utm->odom tranform.

`broadcast_odom2base_transform` (`bool`, default: True.)
: Whether or not to broadcast the odom->base_link tranform.

`zero_altitude` (`bool`, default: False)
: Ignore the altitude in the incoming navigation odometry

`base_link_frame_id` (`string`, default: base_link)
: What frame name to use in the

`odom_frame_id` (, default: odom) `utm_frame_id` (, default: utm) `orientation_ned` (`bool`, default: true)
: If true, specifies that the incoming Odometry attitude, coming from the subscription on the odometry/nav topic, is in a NED orientation (compass bearing convention) and converts the orientation to ENU before publishing.

### 0.0.4 Provided tf Transforms

`utm` → `odom`
: Transform from utm frame to odom frame

`odom` → `base_link`
: Provided if

## 0.1 Odometry in Geographic Coordinates

The node assumes that the geographic navigation information is provided as an Odometry message as described below.

- The header.frame_id and child_frame_id values are ignored.
- pose.pose.position is
  - .y = Latitude [dec. degrees]
  - .x = Longitude [dec. degrees]
  - .z = Altitude [m]
- pose.pose.orientation of the base_link relative to a fixed ENU coordinate frame
- If the ~orientation_ned parameter is set to true, the node will convert the orientation from NED to ENU.
- For now we are assuming the orientation is true (not magnetic). Typically the magnetic declination will be set internal to the sensor providing the information.
- pose.covariance is expressed in meters for position and radians for orientation (REP-103)
- twist.twist.linear/angular is the velocity in the base_link frame
  - twist.covariance is expressed in m/s and rad/s.

## 0.1 Coordinate Frames

- utm: The global UTM coordinate frame. The origin of this frame (which UTM zone we are in) is determined by the datum parameter
- odom: The local, fixed odom frame has an orgin specified by the datum parameter. We have assumed that there is no orientation between UTM and the odom frame. While this is not as general as possible, it simplifies the implementation, usage and interpretation.
- base_link: This mobile frame typically coincides with the sensor frame.

# Local Coordinate Transforms

The package also includes utilities, currently in Python, for conversion between geographic (lat/lon) local (x/y) coordinates.

# 1. Python Modules

Two python modules are included. These modules should be accessible (in the Python path for import) if you follow the catkin_make, source devel/setup.bash workflow due to the settings in package.xml and CMakeLists.txt.

1. alvinxy: Simple, rectilinear transform between lat/lon and x/y
2. geonav_tranform: Transform between lat/lon and x/y using UTM coordinates.

The use of both modules is documented in the API documentation and there are examples in the `examples` directory.



Wiki: geonav_transform (last edited 2019-11-23 06:04:56 by BrianBingham (/BrianBingham))