[**REP Index**] [**REP Source**]

REP:	105
Title:	Coordinate Frames for Mobile Platforms
Author:	Wim Meeussen
Status:	Active
Type:	Informational
Content-Type:	text/x-rst
Created:	27-Oct-2010
Post-History:	27-Oct-2010

Contents

- Abstract
- Motivation
- Specification
  - Coordinate Frames
    - base_link
    - odom
    - map
    - earth
  - Relationship between Frames
  - Example of multi-robot tf graph using ECEF
  - Frame Authorities
  - Transitions Between Maps
  - Exceptions
- Compliance
- References
- Copyright

Abstract

This REP specifies naming conventions and semantic meaning for coordinate frames of mobile platforms used with ROS.

Motivation

Developers of drivers, models, and libraries need a share convention for coordinate frames in order to better integrate and re-use software components. Shared conventions for coordinate frames provides a specification for developers creating drivers and models for mobile bases. Similarly, developers creating libraries and applications can more easily use their software with a variety of mobile bases that are compatible with this specification. For example, this REP specifies the frames necessary for writing a new localization component. It also specifies frames that can be used to refer to the mobile base of a robot.

Specification

Coordinate Frames

base_link

The coordinate frame called `base_link` is rigidly attached to the mobile robot base. The `base_link` can be attached to the base in any arbitrary position or orientation; for every hardware platform there will be a different place on the base that provides an obvious point of reference. Note that REP 103 [1] specifies a preferred orientation for frames.

odom

The coordinate frame called `odom` is a world-fixed frame. The pose of a mobile platform in the `odom` frame can drift over time, without any bounds. This drift makes the `odom` frame useless as a long-term global reference. However, the pose of a robot in the `odom` frame is guaranteed to be continuous, meaning that the pose of a mobile platform in the `odom` frame always evolves in a smooth way, without discrete jumps.

In a typical setup the `odom` frame is computed based on an odometry source, such as wheel odometry, visual odometry or an inertial measurement unit.

The `odom` frame is useful as an accurate, short-term local reference, but drift makes it a poor frame for long-term reference.

map

The coordinate frame called `map` is a world fixed frame, with its Z-axis pointing upwards. The pose of a mobile platform, relative to the `map` frame, should not significantly drift over time. The `map` frame is not continuous, meaning the pose of a mobile platform in the `map` frame can change in discrete jumps at any time.

In a typical setup, a localization component constantly re-computes the robot pose in the `map` frame based on sensor observations, therefore eliminating drift, but causing discrete jumps when new sensor information arrives.

The `map` frame is useful as a long-term global reference, but discrete jumps in position estimators make it a poor reference frame for local sensing and acting.

Map Conventions

Map coordinate frames can either be referenced globally or to an application specific position. A example of an application specific positioning might be Mean Sea Level [3] according to EGM1996 [4] such that the z position in the map frame is equivalent to meters above sea level. Whatever the choice is the most important part is that the choice of reference position is clearly documented for users to avoid confusion.

When defining coordinate frames with respect to a global reference like the earth:

- The default should be to align the x-axis east, y-axis north, and the z-axis up at the origin of the coordinate frame.
- If there is no other reference the default position of the z-axis should be zero at the height of the WGS84 ellipsoid.

In the case that there are application specific requirements for which the above cannot be satistfied as many as possible should still be met.

An example of an application which cannot meet the above requirements is a robot starting up without an external reference device such as a GPS, compass, nor altimeter. But if the robot still has an accelerometer it can intialize the map at its current location with the z axis upward.

If the robot has a compass heading as startup it can then also initialize x east, y north.

And if the robot has an altimeter estimate at startup it can initialize the height at MSL.

The conventions above are strongly recommended for unstructured environments.

Map Conventions in Structured Environments

In structured environments aligning the map with the environment may be more useful. An example structured environment such as an office building interior, which is commonly rectilinear and have limited global localization methods, aligning the map with building is recommended especially if the building layout is known apriori. Similarly in an indoor environment it is recommended to align the map at floor level. In the case that you are operating on multiple floors it may make sense to have multiple coordinate frames, one for each floor.

If there is ambiguity fall back to the conventions for unstructured environments above. Or if there is limited prior knowledge of the environment the unstructured conventions can still be used in structured environments.

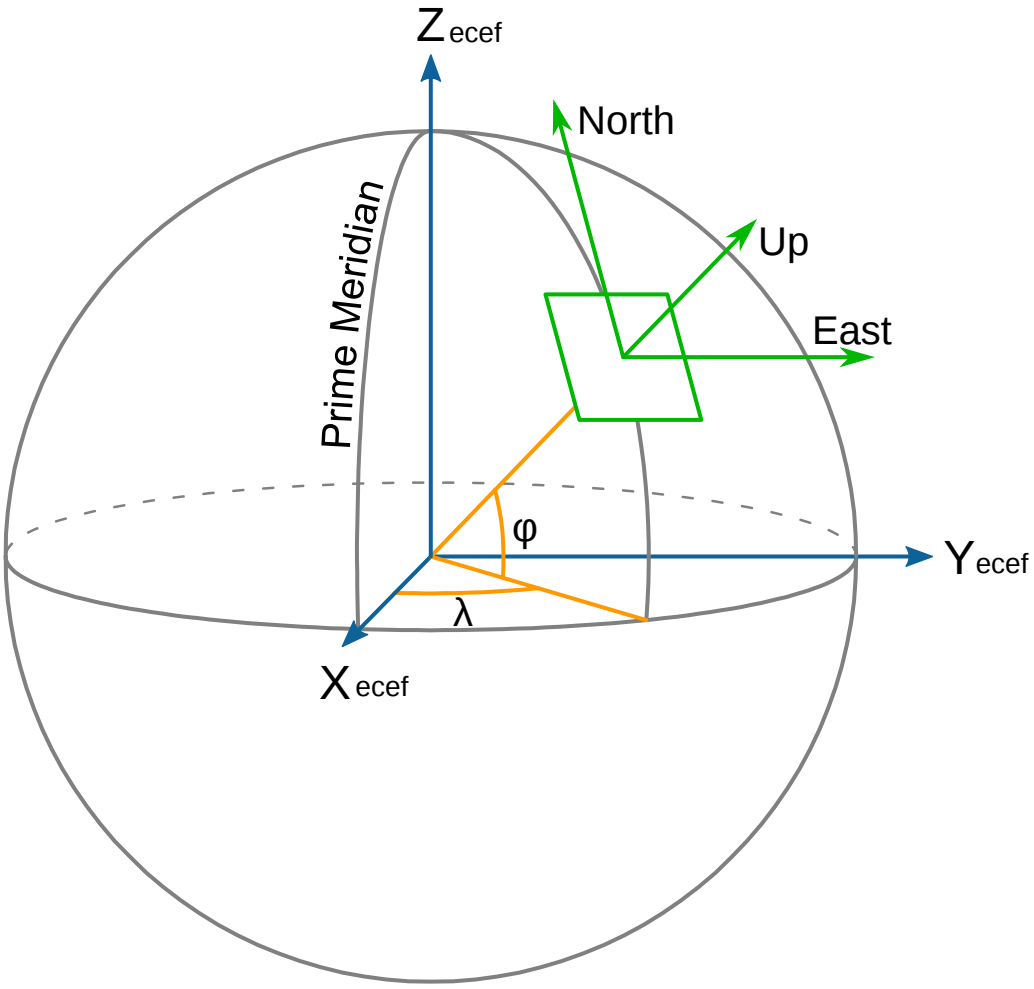
earth

The coordinate frame called `earth` is the origin of ECEF. [2]

This frame is designed to allow the interaction of multiple robots in different map frames. If the application only needs one map the `earth` coordinate frame is not expected to be present. In the case of running with multiple maps simultaneously the `map` and `odom` and `base_link` frames will need to be customized for each robot. If running multiple robots and bridging data between them, the transform frame_ids can remain standard on each robot if the other robots' frame_ids are rewritten.

If the `map` frame is globally referenced the publisher from `earth` to `map` can be a static transform publisher. Otherwise the `earth` to `map` transform will usually need to be computed by taking the estimate of the current global position and subtracting the current estimated pose in the map to get the estimated pose of the origin of the map.

In case the `map` frame's absolute positon is unknown at the time of startup, it can remain detached until such time that the global position estimation can be adaquately evaluated. This will operate in the same way that a robot can operate in the `odom` frame before localization in the `map` frame is initialized.



A visualization of Earth Centered Earth Fixed with a tangential `map` frame.

Relationship between Frames

We have chosen a tree representation to attach all coordinate frames in a robot system to each other. Therefore each coordinate frame has one parent coordinate frame, and any number of child coordinate frames. The frames described in this REP are attached as follows:

The `map` frame is the parent of `odom`, and `odom` is the parent of `base_link`. Although intuition would say that both `map` and `odom` should be attached to `base_link`, this is not allowed because each frame can only have one parent.

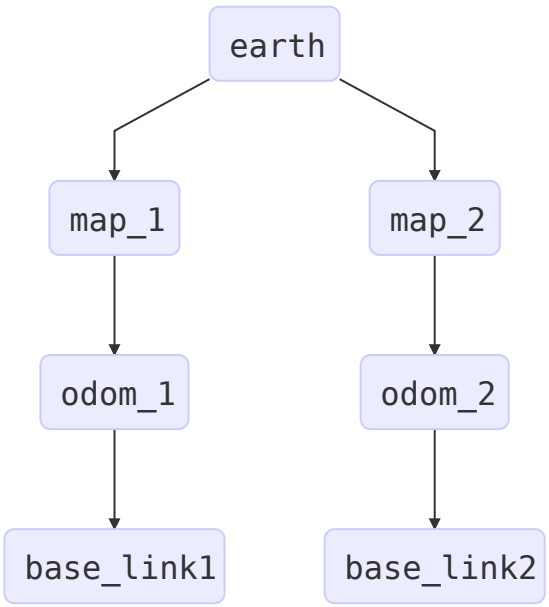
Extra Intermediate Frames

This graph shows the minimal representation of this graph. The basic topology should stay the same, however it is fine to insert additional links in the graph which may provide additional functionality.

Pressure Altitude

An example of a potential additional coordinate frame is one to represent pressure altitude for flying vehicles. Pressure altitude is an approximation of altitude based on a shared estimate of the atmospheric barometric pressure. [5] In flying applications pressure altitude can be measured precisely using just a barometric altimeter. It may drift in time like odometry but will only drift vertically. To be useful a `pressure_altitude` frame could be inserted between the inertially consistent `odom` frame and the `map` frame. There would need to be an additional estimator to estimate the offset of the `pressure_altitude` from the `map` but this extra coordinate frame can support extra functionality and does not break the abstraction outlined above.

Example of multi-robot tf graph using ECEF



This is an example of a tf tree with two robots using different maps for localization and having a common frame `earth`.

The diagram above uses different frame ids for clarity. However for maximum reusability it is recommended to use the canonical frame ids on each robot and use a script to forward information off of the robot. When the information is forwarded the frame ids should be remapped to disambiguate which robot they are coming from and referencing.

Frame Authorities

The transform from `odom` to `base_link` is computed and broadcast by one of the odometry sources.

The transform from `map` to `base_link` is computed by a localization component. However, the localization component does not broadcast the transform from `map` to `base_link`. Instead, it first receives the transform from `odom` to `base_link`, and uses this information to broadcast the transform from `map` to `odom`.

The transform from `earth` to `map` is statically published and configured by the choice of map frame. If not specifically configured a fallback position is to use the initial position of the vehicle as the origin of the map frame. If the map is not georeferenced so as to support a simple static transform the localization module can follow the same procedure as for publishing the estimated offset from the `map` to the `odom` frame to publish the transform from `earth` to `map` frame.

Transitions Between Maps

When a robot travels a long distance it is expected that it will need to transition between maps. In an outdoor context map coordinate frame is a euclidian approximation of a vicinity however the euclidian approximation breaks down at longer distances due to the curvature of the earth. In an indoor context this can be transitioning between two buildings where each has a prior map in which you are navigating or the robot is on a new floor of a building.

It is the responsibility of the localization frame authority to reparent the `odom` frame appropriately when moving between maps. The common implementation of computing the `map` to `odom` frame as the results of subtracting the `odom` to `base_link` from the localization fix `map` to `base_link` will take care of this implicitly when the choice of which `map` frame changes.

odom Frame Consistency

When transitioning between maps the odometric frame should not be affected. Data retention policies for data collected in the odom frame should be tuned such that old or distant data is discarded before the integrated position error accumulates enough to make the data invalid. Depending on the quality of the robot's odometry these policies may be vastly different. A wheeled vehicle with multiple redundant high resolution encoders will have a much lower rate of drift and will be able to keep data for a much longer time or distance than a skid steer robot which only has open loop feedback on turning.

There are other contexts which will also affect appropriate retention policy, such as the robot being moved by external motivators, or assumptions of a static environment. An example is a robot in an elevator, where the environment outside has changed between entering and exiting it. Most of these problems come from the assumption of a static environment where observations are in the same inertial frame as the robot. In these cases semantic information about the environment and its objects is required to manage persistent data correctly. Regardless, the inertial odom frame should always remain continuous.

If the vehicle travels a long enough distance that the distance from the `odom` frame's origin to the vehicle approaches the maximum floating point precision, degraded performance may be observed for float-based data persisted in the `odom` frame. This is especially true of 32-bit floating point data used in things like pointclouds. If distances on this order are encountered a systematic reset of the `odom` frame origin may be required. If centimeter level accuracy is required the maximum distance to the `odom` frame is approximately 83km. [6] There is not a standard solution to this, systems with this issue will need to work around it. Potential solutions include additional coordinate frames in which to persist obstacle data or to store obstacle data, or using higher precision.

Exceptions

The scope of potential robotics software is too broad to require all ROS software to follow the guidelines of this REP. However, choosing different conventions should be well justified and well documented.

Compliance

This REP depends on and is compliant with REP 103 [1].

References

[1] *(1, 2)* REP 103, Standard Units of Measure and Coordinate Conventions (http://www.ros.org/reps/rep-0103.html)

[2] Earth Centered, Earth Fixed Wikipedia article (https://en.wikipedia.org/wiki/ECEF)

[3] Mean Sea Level (https://en.wikipedia.org/wiki/Sea_level)

[4] EGM1996 (http://earth-info.nga.mil/GandG/wgs84/gravitymod/egm96/egm96.html)

[5] Pressure Altitude (https://en.wikipedia.org/wiki/Pressure_altitude)

[6] Floating Point Precision (https://en.wikipedia.org/wiki/Single-precision_floating-point_format)

Copyright