Motivation

For many robotics applications we need to utilize GPS fusion into our ‘map’ frame’s localization in order to have world-tied positioning and navigation. This enables achieving waypoints that are tied to lat lon coordinates for use cases where this is necessary.
Problem Statement(s)

Utilizing GPS has the following issues:

- Map frames in robotics applications tend to be defined in cartesian coordinates with a flat estimation of their portion of the Earth. GPS Coordinates of Latitude, Longitude and Altitude do not conform to this by their very nature.

- Latitude, Longitude and Altitude need an intermediary frame (such as UTM or ECEF) often defined as an Earth frame. The transform from LLA to the Earth frame and then from the Earth frame to Map is required. This transform should be consistent OR updated when the non-linear nature of the Earth’s curve cause significant inconsistency in the localization effort.

- Multiple GPS's are often used to gather vehicle orientation information. This often can be boiled down to Euler Roll Pitch and Yaw of the Vehicle base link relative to some frame.

- GPS quality can be heavily influenced by the environment and GPS drivers are not all made equal. We must be able to handle issues and properly address when to use data or update covariances based on:
  - Noisy data with inaccurate covariances
  - Data quality loss due to dead zones or multipath issues
  - Total loss of GPS signal (and a driver that keeps reporting the last known location)

- When we initialize a single GPS our orientation estimation is not valid until movement is detected. Post movement, the vehicle should align based on holonomic constraints to the global orientation.

Live User Notes:

- Firefly: UTM can be problematic for applications
- Rudis: ECEF is preferable for flying aircraft [PX4]
- Steve: what about annotation remotely without altitude data?
  - Not hearing great concerns
With tilt / pitch components, altitude error becomes problematic when annotating [Ardupilot]

- Aerial prefers ECEF
- GNSS solution on articulating vehicles to calculate angle
- Dual- (N-) GPS for corrections is important (heading information, articulation system, compute offsets due to noise)
- Supporting wgs84 vs nad83
- Hack: Just keep going off the same UTM zone, slightly inaccurate but “shrug”
- [Ryan] Injecting covariance afterward is OK, they’re overly optimistic from the manufacturers
- [Kiwibot] making sure we can align cartesian coords to 3D slam map (e.g. datem)
- Still an open question how to keep the map frame consistent across multiple tangent plane updates
  - Tangent planes should be large enough that it shouldn’t be a huge issue. +kms wide
  - [Ryan] hundreds of kms happen in our applications
- How much do you need a map frame at all — navigation. computer vision. homogeneous transformations.
- [Stephen Williams] Thought we’d be doing something fancier with GPS using GPS math, satellites, etc rather than Pose3’s. If we can boil down, that’s trivial. If we input LLA: no other plugins do that so you’d have to build everything from scratch. Fuse assumes we have multiple sensors measuring the same “stuff”. If we have LLA, that’s different, so you’d have to rebuild all the models you could possibly want to use.
- [Tom Moore] While I don’t work with GPS data these days, if I had to build navsat_transform_node again, I’d use ECEF.
- [Ryan] GPS vs GNSS not made distinct in this proposal
  - e.g. NavSatFix & NavSatFixWithOrientation
Though if we sync topics with IMU then we’re good once NavSatFix support is made

- [Ryan] Capturing modern GPS info via sensor_msgs —> start a new thread

**Part 1: Intermediary Earth Frames**

As we mention in the problem statement, the map frame is assumed to be a 2D frame. There are two main methods we have come up with in order to handle the lat-lon to earth transform that allows for converting GPS LLA into a cartesian coordinate:

- UTM
- ECEF with a Local Tangent Plane Map

**UTM (Universal Transverse Mercator)**

UTM is well known by many in robotics and non-robotics space as a consistent method to convert lat lon altitude to a pre-defined linearized tangent plane to the Earth. You can see the wiki definition [here](#) to learn more.

**Pros**

- UTM is widely accepted and successfully adopted by many entities including robot_localization with NavSatNode doing the intermediary conversion prior to EKF fusion.
- The definition is standardized for its conversion from LLA to the UTM frame
- UTM zones easily map to flat planes and can ignore altitude for the most part
- UTM is defined with meters and conversion to a meters based map frame only require an X and Y offset. This offset can also be described as a “datum” in robot_localization defining the origin in lat-lon and subsequently UTM
- The error even at boundaries is generally not huge until you go significantly into a different zone.

**Cons**

- UTM does not give unique coordinates. Metadata on Zone and Hemisphere is necessary in order to do UTM to LLA conversion. This can be assumed when well
inside a zone, but this assumption may not be valid at boundaries that overlap

- Boundary behavior is ambiguous. You can have multiple correct UTM coordinates and zones near boundaries with varying levels of error due to:
  - Overlap of Boundaries
  - Extension of Zones across boundaries
- Entities using UTM to LatLon conversion require metadata in order to guarantee synchronized usage of UTM data.
  - As an example, a node outside of navsat node in robot_localization must either:
    - Have a matching datum to the navsat node
    - Initialize at the same time as navsat node (in the case of large regions and boundaries)
    - Get metadata from navsat node
    - Rely entirely on Navsat Node services to and from LLA
- Accuracy dips the further outside of a zone you get. For large maps, you must manage multiple zones as well as manage the accuracy loss on zone boundaries
  - Work to be done would be explicitly defining boundary behaviors

**ECEF (Earth-Centered Earth-Fixed)**

ECEF is a default Cartesian frame assuming the Earth’s center as the frame’s origin. The Earth crust modeled as as a perfect ellipsoid. You can see the wiki definition [here](#) to learn more. We assume that we are creating tangent planes off of the robot’s initial position with an ECEF origin as the transform to get Map. This gives us consistent ECEF to Map transforms.

**Pros**

- ECEF is has unique positions for all points and is continuous
- Metadata is not necessary to go from ECEF to LLA, it is a constant definition
- Treating map as a local tangent plane relative to an ECEF position lets us have accurate position estimates via relatively simple static transforms from Map to ECEF to LLA
- Is by default 3D due to its nature. Altitude usage comes for free as LLA to ECEF is inherently a transform into 3D space
- Robots and Vehicles don’t need metadata synchronization among nodes when operating at larger scales or in new regions.

**Cons**
- Not extensively used in ROS2 and robotics (yet)
- Altitude can cause issues in any calculations that assume no “z” variations. This includes “flat” assumptions made by control and plan algorithms
- If a vehicle moves in altitude ECEF by default will catch it and projection will be used to go between frames. Transforms such as Roll Pitch and Yaw may lead to errors that get amplified due to Altitude shifts when planning and controlling.
- Goals are required to have accurate altitudes in order to be translated to the appropriate map frame in an appropriate way. An altitude shift is tied to an X,Y and Z transform in ECEF space.
  - Steve Note: this all means you need to specify Altitude in requested GPS targets, not simply Lat/Long. Alt noise is sensitive. Is that acceptable to folks?
- ECEF to Map also suffers from errors due to linearization at large distances
- ECEF to LLA requires a relatively lightweight but iterative approach

**Additional Thoughts:**
- 3D localization in general may have offsets when projecting between frames. ECEF naturally lends to 3D localization. Users may have to be more careful with frame definitions and orientations if using ECEF.
- Both ECEF and UTM should allow for Datums on necessity to guarantee a consistent map. however, unlike UTM, ECEF does not need to account for boundary conditions.
- If we are given GPS with Yaw, we may want to utilize GeoPoseWithCovarianceStamped (mouthful) instead of NavSatFix messages
  - Steve Note: We should not casually break sensor_msgs support. We can consider adding a new message to it that builds on NavSatFix which includes
orientation or have synchronized topics providing orientation when possible.

**Frame Proposal**

Continuing with the robot_localization tradition we propose that we have an intermediary node that transforms from LatLon to an Earth frame, whether ECEF or UTM. This intermediary node will convert us into absolute cartesian coordinates with covariances that will be used by Fuse (ideally) in order to generate a “map” frame that prefers absolute 3d coordinates.

**Polymath’s Preference: ECEF Proposal**

Polymath proposes that we rely on ECEF as our Earth frame. We feel that the advantages of not having to deal with zone metadata and boundary conditions is enough to push us towards relying on ECEF as a global and consistent transform to LLA.

Since the relationship from LLA to ECEF is always consistent we don’t need to put effort into managing this. We can instead focus on managing ECEF to Map.

To manage ECEF to Map we propose the following behavior:

- We define a parameter that is the threshold of distance we are allowed to travel after which our error is big enough to require a new transform. This may be X, Y or Z motion. This allows for configurability according to application

- On generating a new transform from ECEF to Map, the coordinates in map are maintained so the localization method does not have a significant jump. Old map points may correlate less with the original ECEF locations, however ECEF points themselves will remain reliable.
  - Steve Note: how do you propose to accomplish that?

- Allow for a 2D Mode in which only lat-lon are used to get ECEF coordinates including on initialization. Altitude is ignored.
  - This would require synchronization similar to UTM’s zone synchronization between multiple entities trying to do this transform
Steve Note: please clarify synchronization required. Do we think quashing altitude could have any notable downsides?

**Alternative: UTM Proposal**

If ECEF’s drawbacks due to its 3D natures are significant enough that the greater community feels UTM is a safer choice, we propose the following UTM behaviors:

- UTM Transform has a service call or a static publisher that defines its metadata so that all systems are synchronized. This also enables multi-gps usage
- On approaching a zone boundary, to prevent a discrete jump, a new zone is switched to and the publisher for metadata updates so all systems remain synchronized when setting a new zone. The new transform from UTM to Map maintains map coordinates so that map does not jump.
  - Steve Note: how do you plan to accomplish that?
- Fuse uses the Map coordinates for all actions

**Part 2: GPS and Accuracy and Multi GPS**

This is a shorter section speaking about GPS and Sensor accuracy when used for fusion.

**GPS Reliability and Sensor Accuracy**

GPS’s can have a number of reliability problems as discussed in the problem statement. We have seen GPS’s with

- Bad Covariances from the driver
- RTK loss
- full signal loss
- discrete and inaccurate jumps in position not captured by covariances or other forms of data

**GPS Orientation**
GPS orientation data and odometry data is currently not captured by NavSatFix messages. Separate messages are traditionally used. This is a minor annoyance.

**Multi GPS Fusion**

Currently robot_localization can't handle multi-gps fusion without multiple navsat transform nodes that all transform into map frame and publish their own UTM and Map transforms. We need to be able to look at sensor reliability and behavior and also synchronize the transform across multiple sensors so that all get ported into Map correctly.

**Proposal**

These are up for discussion and not necessarily something that needs to be solved here.

**Sensor Quality**

- Covariance Post Processing (is useful for other sensors as well) to either inject static or inject calculated covariances into messages so that they may be used more appropriately
  - Steve Note: this seems to be turning into a [ros-planning/gps_tools](https://github.com/ros-planning/gps_tools) repository with the frame conversion tool and now a covariance estimator / injector. Neat! Are there techniques in mind for computing these you propose?

- If using NavSatFix messages, leverage the Covariance Type to know when to completely ignore the message. If a message has RTK loss or signal loss, or is unreliable as defined by the driver
  - Steve Note: this points to needing either (1) a custom Fuse node that we can reject updates we don't like or (2) the injector tools just doesn't forward messages that aren't to be useful (with a timeout - we don't want to never get info if we'll never get excellent info).

- To be used GPS Drivers must tell us when the data is unusable so that we don't fuse it

- Scale Covariances as Tangent Plane Error increases
Orientation and Message Type

- NavSatFix does not capture orientation data published by the GPS. Many GPS systems will be able to give us orientation and yaw data. As such we may want to consider switching our core to GeoPoseWithCovarianceStamped. However, we lose covariance type in navsat node
  - Steve Note: I propose a NavSatFixWithOrientation message in sensor_msgs

- We may want to propose a more GPS centric message type that allows for additional definitions than “bad covariances” to instruct us on how to use and fuse their data
  - Steve Note: clarify what definitions you might have in mind

Multi GPS

- A primary GPS is defined based on configuration. This may be in the same node or other nodes. This primary GPS defines the initial transform.

- All GPS’s must use the same transform from Earth to Map in order to remain consistent
  - Steve Note: how do you propose that synchronization of distributed drivers? The LLA→Map tool have a single instance processing $N$ sensors?

- GPS’s must have accurate sensor positions and descriptions in order to fuse

- Each GPS will be published as its own pose with an accurate header to be fused

Part 3: Locus Robotics Fuse Integration

We would prefer if we could utilize Fuse to integrate GPS. This would have to be done with the following:

- Fuse GPS coordinates in Cartesian format as described in the aforementioned sections

- Fuse Orientation data (often in IMU message type) acquired from GPS’s

- Handle discrete jumps by preferring GPS positions (when accurate)

Proposal
If the above proposals are agreed to. We can leverage one of the following by default when fusing cartesian transformed GPS coordinates:

- **Pose2D**
- **Odometry2D**
- **Imu2D**

However, by their nature, these two can't handle 3D transforms which may be valid to be ignored in most cases. In cases where these are necessary we can add:

- **Pose3D** sensor model
- **Odometry3D** sensor model
- **Imu3D** sensor model

Since we have “absolute” coordinates in GPS these may work well for us. However, if we want to add better validation, we can consume the GPS model directly with LatLonAltitude and Orientation as well as GPS Quality information on top of covariances. We then use standard GeographicLib to go from LatLonAltitude to Earth, and then do a TF transform to our target frame and calculate our costs from there with standard 3D constraints. This would allow us to “validate” messages directly and utilize the loss function to ignore some level of statistical outliers in messages. This would still require some third entity to manage the Earth to Map transform, however, that entity can choose to ONLY do that transform and Fuse just uses it. This decreases the number of topics that need to bounce around for us to get good localization off of GPS.

—> Steve note: not a bad idea! If we use Fuse to provide us with fused information in the GPS frame, we’d then need to only convert the output to map. However, I think an issue you’d run into is trying to fuse other data into that same filter with the GPS data that is not in LLA coordinates, so the coordinate systems don’t align for consistent fusion

Some notes regarding fusion:

- We want a map and odom frame as defined in general by ROS with Map allowing for discrete jumps and Odom being smooth
We would like to better validate GPS messages in general. Whether this is done in Fuse or in the Earth to Map transformer node.

Do we want to build this out with 3D in mind?

Notes

GPS vs GNSS

- Full Intertial System Support for GNSS vs GPS
  - GPS NavSatFix
  - GNSS GeoPoseWithCovarianceStamped
  - Synchronized ingestion of topics with navsatfix and imu as an alternative to GeoPoseWithCovarianceStamped

Part 4: Side Topics

When operating vehicles at scale, we may need operational areas that are much larger. This can make any computations done at a large zone scale more complex. How do we want ROS2 to grow to allow for operating in large operational areas.

- How can we make map frame adjust to larger zones while also accounting for non-linearities of ECEF and UTM?
  - Density of maps and costmaps dynamically adjust to range?
  - Planning and Controlling in large areas and (or even multiple zones for UTM)