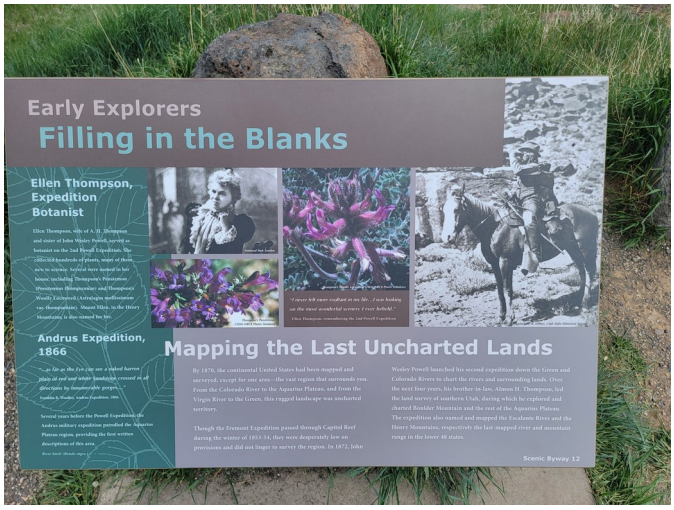
SOTM Mapping Trip

Since SOTM-US 2024 was in Utah this year, I decided to drive there from my home in Colorado. Normally it's about a 12hr drive, but I decided to spend a week or so driving each way mapping and camping. I had some updated XLSForms to test, and the best way to test them is to drop offline for a few days and use them in the field. I've done a bunch of previous mapping trips in SE Utah, but this trip was new territory, the central Utah and hitting a few national parks too on the way before they get busy or hot.



Map data in remote areas of the US is often lacking, or has changed and the maps haven’t been updated. The goal of this trip (other than software testing) was to go to an area I had never been to before and update the map data. This trip was mostly focused on good campsites, updating amenities, and find some great camping and add those too.

The software I'm testing is osm-fieldwork, which is a collection of utilities I wrote to process OpenDataKit data files to something suitable for OpenStreetMap. This software is also the core part of the backend for Field Mapping Tasking Manager (FMTM), which supplies a higher level organization for large scale field data collection and ground-truthing by groups of people. This project is focused on improving the data processing part of the FMTM backend, and to also work fully offline for days. Truly working offline is the best test. Osm-fieldwork works as both the backend for a web front-end, or standalone when offline.

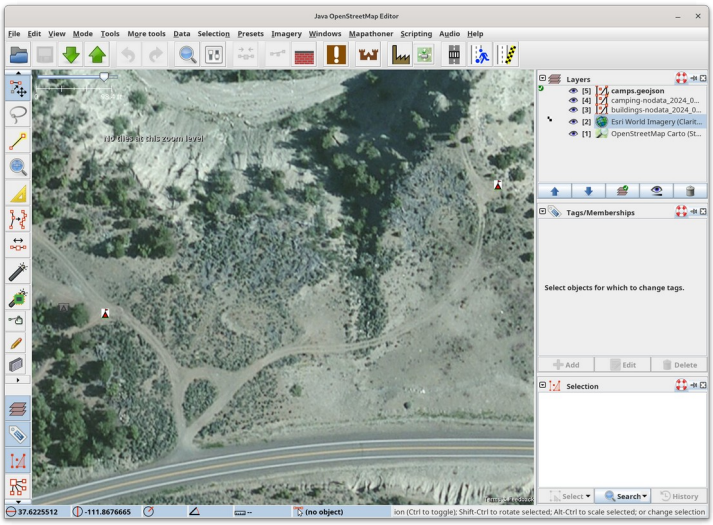
Trip Prep

I did the usual preparations for the trip, spending time looking at existing data on freecampsites.net, iOverland, OpenStreetMap, and a few other sites with user contributed free camping. I prefer free camping when possible, and don't really need much beyond a reasonably flat spot to park.

In addition I spend time looking at USGS topo maps and satellite imagery for possible camping spots. Since I spend a lot of time waiting for code to compile or data to be processed, this is what I do while waiting. And it's fun to have an idea about where I'll be going to help with spontaneous navigation decisions.

Since this is a remote area, I also did a little research on any existing restaurants and gas stations, since updating amenities was part of the purpose for this trip. In many of these tiny towns, restaurants are only open for a few hours, or only on weekends. Restaurants seem to come and go in remote towns, so it's good to ground-truth them. Although I planned to be cooking in my camp sites mostly, it's still good to know about the options. And gas stations were far and few between. When trying to identify possible campsites using satellite imagery,it's useful to have a few other files, like boundaries of public/private land. Or existing BLM or USGS data that isn't in OSM yet. In the US, often the topo basemap

has some official dispersed camping areas on the map. But on BLM or USGS land, you can camp almost anywhere. Criteria for mapping campsites is further down in this post.



When trying to identify potential campsites from satellite imagery, I also load the multi-polygon boundary of the BLM or USGS public land so when staring at imagery, I can only select potential spots that are truly on public land. Sometimes I also load the MVUM highway data for details that may not be in OSM yet. That's mostly used to just determine how far up a 4x4 road to drive. Many highways in the national forests are good to a certain point, and then it'd be better to switch to a off-road vehicle like a UTV. Those spots are easy to find in the MVUM data.

With good imagery, you can often see people camping already, which is a good sign. While some people are comfortable camping close to a major road, I look for the dirt roads that branch off of them. These usually have short spur roads that branch off those to the camp site. In public lands you must be at least 200ft off the road. For each potential camping spot, I add a node with *tourism=camp_site* so they appear in JOSM. The main reason to make a list is in remote areas, it's often impossible to tell the difference between a publicly accessible dirt road and somebody's long driveway. Since it can take a long time to download all the imagery at a sufficient resolution, this is best done when at home.

Once I have all the spots I want to checkout, I save the file in GeoJson format, and then run the osm2favorites program which produces a GPX file for the OsmAnd mobile mapping app. I use OsmAnd for all navigation because it works fully offline, and I can update data files and basemaps when in the field using adb. This utility program add the correct icon and color to each feature so they appear as something other than the default star.

Field Data Collection

For this trip I was experimenting with a few experimental XLSForms, one focused on camp sites, and the others on buildings and amenities. There are two variations of each XLSForm. One uses a data extract from OpenStreetMap, which is needed for FMTM. The other is a *nodata* version, without an extract. The goal is to make the version with a data extract work when the additional metadata you want to add is in OSM already. This is a test for editing existing OSM features, which is a large part of the FMTM data flow.

I also collect data without an extract since I want to test my conflation software. Not using a data extract gives me lots of potential duplicate features to testing conflation. To get the best coordinates, a good basemap is required, especially if offline. ODK Collect supports basemaps in the mbtiles format, which is supported by many projects. I use my basemapper program, as it also makes sqlitedb file basemaps for OsmAnd using the same tiles. Basemapper also caches all the downloaded tiles to disk. If you have enough disk space, you can uses these maps tiles in JOSM, letting you work fully offline. I often have a huge area (Colorado, Utah, Wyoming) of map tiles on my laptop of Bing or ESRI imagery, plus USGS topo maps.

The Amenities XLSForm is a data extract to cover a large area. Usually FMTM mapping campaigns cover a small area, as the mappers are walking. But sometimes you are mapping from your vehicle, with long distances between features. Think a string of small villages along a rural highway.

In this case I was going to be mapping in several rural counties in Utah. This region has several small towns, each with a few amenities. Usually a gas station, a restaurant or two, etc... Business is hard in these towns, so there are often changes. During the pandemic, many closed completely, as they were dependent on tourist traffic. For FMTM, the data extracts are for a small area, but for this trip I wanted one larger data extract to cover the entire area so I wouldn't have to keep loading different XForms all the time. That just adds clutter to ODK Collect.

Having a feature in the data extract lets me not have to wait for a GPS location as I'm just adding tags to an existing feature. Once I select the feature in Collect, it'll display the tags, so if nothing has changed, you can discard the session and continue. This is an extension of editing existing data, but testing things like practical file size, and efficient data entry.

A building extract over a large area is probably a mistake, as it’ll be huge even in this sparsely populated region. But Amenities is limited to just those features. If you come across an amenity not in the map, the building is in the basemap, so you can use that for the coordinates. Then answer everything in the XForm. This is why basemaps are critical when collecting data offline.

Mapping Campsites

In the western US, there are many, many free distributed camp sites. In any USGS or BLM land, you can often camp anywhere a few hundred feet off any road and away from water. For decades, campers, hikers, hunters, RVs, etc... have used these areas for camping. Most are not on any map, being informally created. Some are on the various free camping apps, but most are not.

There are a few classes of free campsites. Some are a bit close to a road, but are popular with the RV set. Some need a high clearance vehicle to access. Some have no shade, and most have zero amenities. Often though they can have a great view and easy access to other features. When I'm mapping rural highways, I often find great campsites way out in the middle of nowhere.

My criteria though is it must have a well-used fire ring, not just something thrown together for a night. I’m a volunteer fire-fighter, so picky about what makes a

safe fire ring.

Other signs of multiple use are good too like obvious signs of tent usage, or camper parking. And always be conscious of environmental impact, occasionally an informally established campsites is not in an appropriate place, like right next to a natural water source.

There are some real-world problems when mapping camp sites. Often you are driving down a dirt road, and there are campsites on spur roads the appropriate distance from the road. If they are occupied I can't drive or walk into their camp to get a good GPS hit. So I usually just mark where the camp site spur road branches off from the main road. This means later I have to adjust all the locations using JOSM using satellite imagery to move the node to the actual camp site location. Another advantage to having a map tile store of imagery on my laptop.

On rare occasions I stay at official BLM or USGS campgrounds. Many of the most remote ones are also free, but have some amenities like a pit toilet, picnic tables, a grill, and sometimes a metal box to protect food from bears. In OSM the campground usually exists as a POI with no data beyond the name. My XForm has support to collect that data, and often map each campsite, get the ref number, etc… This adds to the richness of the map which benefits recreational users.

Mapping Amenities

Many amenities change frequently in rural America, the pandemic killed many rural businesses that depended on tourist traffic. Often the names and cuisine has changed since it was added to OSM. Often though there is no existing OSM data at all in many of these towns. So I'll add the handful of amenities like a gas station, convenience store, or restaurant. If there is a sign with opening hours (often only the weekend, or only in the summer) then collect that too.

When using a data extract with ODK Collect, I can just touch the teardrop icon where the building is, and all the existing tags in already OSM will get displayed, and also sets the defaults for the questions in the XForm. You can also quickly see the existing tags in the OSM feature, and quit the session if you have no updates. Often it's a new business in the same location as the previous one, but since the GPS coordinates are already in OSM, you can make any changes to the tags without waiting for a good GPS location. If the amenity is not in OSM, then I just collect the data like normal.

Mapping Highways

Mapping highways with ODK Collect isn't very efficient, you're better off using StreetComplete and downloading the base dataset when online. But if you forget to do that, I still use Collect. Often the highway is in OSM already, just lacking tags beyond *highway=something*. I'll use Collect to mark where the surface changes from paved to dirt for example. Then in JOSM I'll split the highway at that point and merge in the updated tags when I can get online.

Processing The Data

I start by either downloading the JSON file from ODK Central using the odk_client.py program in osm-fieldwork. This uses the ODATA support in ODK Central to down the submissions I've collected. This assumes I have a cellular connection though, so if offline in the desert, I use adb to pull the ODK instance files off my phone and onto my laptop.

Then I use the odk2osm.py program to convert the input data to OSM XML and GeoJson. The OSM XML file is required for easier conflation, and tags are converted into the OSM tagging schema. The GeoJson file has no tag conversion and requires a more manual conflation process.

Then I load the OSM XML file into JOSM and view the results of the day's data collection. I find it's good to review fresh data as soon as possible, as there are often details I remember that weren't supported by the XLSForm. In that case I'll update the XLSForm, and use adb to copy it to my phone.

It's a touch complicated to properly create a data extract from OSM to use for machine assisted conflation, as it currently requires a local postgres database with all the data imported to make the extract. The process to do that is simple though. Download the desired region from geofabrik, and use the importer program from my osm-rawdata project. That will populate the postgres database with the data using a modified OSM schema oriented towards data analysis.

Then run the make_data_extract program in osm-fieldwork to query the database and make the data extract. This program also filters the data extract to only contain the tags in your XLSForm. If this isn't done, Collect won't launch. This data extract then needs to be uploaded to ODK Central with your XLSForm, or manually copied to your phone using adb. If your XLSForm is setup to use a data extract from OSM as documented here, you can also manually conflate the data by using the GeoJson output file instead.

The conflator program can be used to conflate the ODK data with OSM. This program is a work in progress though, so the results may be mixed.

Manual conflation

The ODK data will be a single node in JOSM, whereas the existing building is probably a polygon. If you are in the building when getting the location, it's pretty easy, the node will be in the building footprint. Since often you are recording the location in front of the building, it'll be offset. You can then manually cut & paste the tags from ODK into the building way, and delete the nodes. Also sometimes the existing data is a POI, so no building polygon, just another node with tags. When using OSM as a imagery source in JOSM, you can see the existing feature. Since often the POI is in a *building=yes* polygon, all the tags from ODK and the existing POI should be properly merged into the building polygon, and the two nodes deleted.

Then you need to validate the tags that came from ODK to make sure they fit the OSM schema for tagging. The taginfo site is very useful so is the OSM wiki for finding the best tags.