



Travel Model Two Development: Bicycle Ways

Technical Paper

Metropolitan Transportation Commission with Parsons Brinckerhoff, Inc.

September 16, 2013

m:\development\travel model two\supply\deliverables\bicycles\2013 09 16 release bicycle network.docx

1 Overview

MTC is rebuilding the representation of supply in our travel model. When complete, the new representations of space, roadways, transit service, sidewalks, and bicycle ways will become part of the *Travel Model Two* modeling system. For an overview of the model design, please see the [Travel Model Two: Strategic Design technical paper](#)¹.

This technical paper describes the development of the *Travel Model Two* bicycle network. Specifically, it discusses the steps for building the network from the *Travel Model Two* roadway network² and the MTC Bike Mapper network³. The Bike Mapper network is a framework in which local cities update a master database of bicycle infrastructure and bicycle lane attributes, from which MTC has built and now maintains a trip planner application. The steps include three implementation tools: MS-DOS batch files, Python scripts, and Cube scripts.

¹ http://analytics.mtc.ca.gov/foswiki/pub/Main/Documents/2012_08_24_RELEASE_Strategic_Design.pdf

² Please see the *Travel Model Two Development: Roadway Network* technical paper for details:
http://analytics.mtc.ca.gov/foswiki/pub/Main/Documents/2013_09_09_RELEASE_Roadway_Network.pdf.

³ <http://gis.mtc.ca.gov/btp>

2 Algorithm and Inputs

2.1 Algorithm

The procedure for building the bicycle network proceeds via the following steps:

1. Reconcile the Bike Mapper network with the TeleAtlas source for the *Travel Model Two* roadway network and create a list of link modifications.
2. Append the bicycle network data to the existing *Travel Model Two* Cube network.
3. Add the bicycle network links that do not map to any existing street/pedestrian links to the network.
4. Add the new bicycle network nodes and links to the *Travel Model Two* network.
5. Intersect the new bicycle links with the *Travel Model Two* network.
6. Add in the split bicycle/network links to the existing Cube network.
7. Query the USGS elevation data website to obtain the elevation (or z-coordinate) of each node in the network and add the elevation data to the *Travel Model Two* network.
8. Code the bridge links that are open to bike and pedestrian traffic.

2.2 Inputs

To build the bicycle network, the following input data are required:

1. `BikeNetwork.gdb`: The latest Bike Mapper geo-database, which includes two layers:
 - a. `Routes_v6` – routes which traverse roadway segments (a.k.a. the Bike Mapper on-street network)
 - b. `Trails_v6` – routes which do not traverse roadway segments (a.k.a. the Bike Mapper trail network)
2. `tana_sp_with_maz_taz_tap_centroids_connectors_routes_osm_split.net`: The *Travel Model Two* network (including pedestrian links⁴) in Cube format.
3. `ca_nw_Intersect_sp.shp`: The TeleAtlas shapefile from which the *Travel Model Two* network was originally developed.
4. `link_split_ped.csv`: The list of ordered link splits produced by the pedestrian/street link intersection procedure.
5. `link_delete_ped.csv`: The list of original network links to be deleted per the pedestrian/street link intersection procedure.

⁴ Please see the *Travel Model Two: Sidewalks* technical paper for additional details:
http://analytics.mtc.ca.gov/foswiki/pub/Main/Documents/2013_09_16_RELEASE_Pedestrian_Network.pdf.

3 Build Scripts

The bicycle network is created by the following Python and Cube scripts:

1. `netToShapefileForBike.s`
2. `bicycle_network.py`
3. `addBikeData.s`
4. `add_bike_links.py`
5. `addNewBikeTrailData.s`
6. `intersect_non_street_with_streets.py`
7. `merge_split_bike_links.s`
8. `updateElevations.py`
9. `updateElevations.s`
10. `addBikePedOk.s`

3.1 `netToShapefileForBike.py`

Purpose: Export the master network to shapefile for intersecting with ArcGIS.

Inputs: `tana_sp_with_maz_taz_tap_centroids_connectors_routes_osm_split.net`

Outputs: `tana_sp_with_maz_taz_tap_centroids_connectors_routes_osm_split.shp/`
`shx/dbf/prj`

3.2 `bicycle_network.py`

Purpose: Reconcile the Bike Mapper network with the TeleAtlas source for the *Travel Model Two* roadway network and create a list of link modifications.

The Bike Mapper network is based on a different TeleAtlas network than the *Travel Model Two* roadway network. As such, the Bike Mapper information does not perfectly correspond to the TeleAtlas network from which the *Travel Model Two* roadway network information is derived. This process matches the two networks using geographic relationships: overlaying links from the Bike Mapper with buffers to identify links from the *Travel Model Two* TeleAtlas roadway network.

The buffers are necessary because (a) the two networks are from different sources, and (b) the projection of the networks into the same coordinate systems may produce registration errors. The Cube version of

the *Travel Model Two* roadway network is not used because it does not maintain the shapes and curves described in the TeleAtlas database, which makes overlays difficult.

The script performs the following steps:

1. Create a temporary geodatabase (`temp.gdb`) in the working (output) directory.
2. Re-project the Bike Mapper network into the same geographic coordinate system as the street network.
3. Create a subset of the street network which does not include any highway links ($FRC > 2$ AND $RAMP = 0$). The reasoning behind this is that bicycles are not allowed on highways, and this will make the subsequent operations less computationally expensive (since they will work on a smaller dataset).
4. Build a single buffer (50 ft) around the entire re-projected Bike Mapper network. Overlay the street (sub-) network with this buffer and select only those links with 80 percent or more of their length inside the buffer. This set is the physical street bicycle network.
5. Build another set of buffers (50 ft) around the re-projected Bike Mapper network, only this time each buffer contains a unique set of bicycle data common to all of the links in that buffer. That is, all of the links a given buffer shape represents have identical bicycle link classifications.
6. Overlay the street bicycle network with the split buffers, and depending which buffer a majority of a given link resides, make a correspondence between the street bicycle link and the Bike Mapper data to which it corresponds.
7. Remap the links, if necessary, to account for the link splits specified by the pedestrian network build procedure. This is required for network consistency, since the splitting process may have occurred after the source network the bicycle links are being mapped to was created.
8. Write out a comma-separated value file indicating the link (identified by its A-B specification) and its bicycle data. The format of the file is shown in Table 1.
9. As an extra informative step, a buffer (50 ft) is built around the street bicycle links, and the Bike Mapper network overlaid with it (using the aforementioned 80 percent rule). The total number of Bike Mapper links *not* mapped to a street are reported.
10. If specified, the unmapped Bike Mapper links can be saved to a shapefile.

Table 1 - Bike Link Fields

| Field Name | Data Type | Description |
|-------------------|------------------|----------------------------------------------------------------------------------------------------------------------------------|
| A | Integer | From node number |
| B | Integer | To node number |
| B_CLASS | Integer | Bicycle link class identifier 1 – Class I Trail 2 – Class II Route 3 – Class III Route |
| REPRIORITIZE | Integer | Bike class priority 2 – Highly desirable 1 – Desirable 0 – No preference -1 – Undesirable -2 – Highly undesirable |
| GRADE_CAT | Integer | Bike Mapper grade category 1 – 0-5% grade 2 – 5-10% grade 3 – 10-18% grade 4 – 18% or higher grade |
| PED_FLAG | String | Bike Mapper pedestrian flag 'Y' – Pedestrian access 'N' or '' – No pedestrian access |

This process is run twice. First, it is run for the Bike Mapper links corresponding to streets (layer `Routes_v6`) against the original TeleAtlas network. Second, it is run for the Bike Mapper non-street links (layer `Trails_v6`) against the combined street/pedestrian network. The former pair is used so that street geometry can be accurately captured, the latter so that bicycle trails can be mapped to shared pedestrian links, where appropriate.

Inputs: `BikeNetwork.gdb`
`ca_nw_Intersect_sp.shp`
`link_split_ped.csv`
`link_delete_ped.csv`
`tana_sp_with_maz_taz_tap_centroids_connectors_routes_osm_split.shp`

Outputs: `bike_links.csv`

```
bike_trail_links.csv
missing_trail_links.shp
```

3.3 addBikeData.s

Purpose: Append the bicycle network data to the existing *Travel Model Two* Cube network.

This script reads in the bicycle link data output from `bicycle_network.py` and appends it to the appropriate network links in the *Travel Model Two* Cube network. For links with no bicycle data (i.e. non-bicycle links), empty fields are automatically created. A shapefile version of the network links is produced.

```
Inputs:  tana_sp_with_maz_taz_tap_centroids_connectors_routes_osm.net
           bike_links.csv
           missing_links.csv
           bike_trail_links.csv
```

```
Outputs: tana_sp_with_maz_taz_tap_centroids_connectors_routes_osm_bicycle.net
           tana_sp_with_maz_taz_tap_centroids_connectors_routes_osm_split_bikesub.shp
```

3.4 add_bike_links.py

Purpose: Add the bicycle network links that do not map to any existing street/pedestrian links to the network.

This script reads in the bicycle links that were not mapped (i.e., “missing”) to the street network, snaps their endpoints to the appropriate network nodes (if within a specified 50-foot-buffer), and writes out CSV files with the data for the new nodes and new links.

```
Inputs:  tana_sp_with_maz_taz_tap_centroids_connectors_routes_osm_split_bikesub.shp
           missing_trail_links.shp
```

```
Outputs: missing_bike_trail_links.csv
           missing_bike_trail_nodes.csv
```

3.5 addNewBikeTrailData.s

Purpose: Add the new bicycle network nodes and links to the *Travel Model Two* network.

This script reads in bicycle link and node data output by the previous procedure, and combines it with the existing Cube network. Also, it exports the resulting network links to a shapefile for later use.

```
Inputs:  tana_sp_with_maz_taz_tap_centroids_connectors_routes_osm_split_bikesub.net
```

missing_bike_trail_links.csv

missing_bike_trail_nodes.csv

Outputs: tana_sp_with_maz_taz_tap_centroids_connectors_routes_osm_split_bikesub2.net

tana_sp_with_maz_taz_tap_centroids_connectors_routes_osm_split_bikesub2.shp

3.6 intersect_non_street_with_streets.py

Purpose: Intersect the new bicycle links with the *Travel Model Two* network.

This script splits the street network into bicycle and non-bicycle (non-highway/TAZ/MAZ) links, and determines where they intersect. Wherever an intersection is found, a new network node is specified and the appropriate street/bicycle links are split at this point. This allows bicyclists on trails to merge into the street network.

Inputs: tana_sp_with_maz_taz_tap_centroids_connectors_routes_osm_split_bikesub2.shp

missing_bike_trail_links.csv

missing_bike_trail_nodes.csv

Outputs: link_split_bike.csv

link_delete_bike.csv

new_nodes_bike.csv

3.7 merge_split_bike_links.s

Purpose: Add in the split bicycle/network links to the existing Cube network.

This script reads the new link and node files, as well as the links to delete, produced in the previous step, and batches these results into a new Cube network.

Inputs: tana_sp_with_maz_taz_tap_centroids_connectors_routes_osm_split_bikesub2.net

link_split_bike.csv

link_delete_bike.csv

new_nodes_bike.csv

Outputs:

tana_sp_with_maz_taz_tap_centroids_connectors_routes_osm_bike.net

3.8 updateElevations.py

Purpose: Query the USGS elevation data website to obtain the elevation (or z-coordinate) of each node in the network.

This script reads a Cube node file saved to shapefile, looks up the USGS elevation from the USGS elevation data web service⁵, and writes out the z-coordinate by node to a text file. It requires an internet connection and can take a substantial amount of time depending on the number of lookups.

Inputs: tana_sp_with_maz_taz_tap_centroids_connectors_routes_osm_bike.shp (including the related shx, dbf, and prj files)

Outputs: nodeElevations.csv - N,Z fields

3.9 updateElevations.s

Purpose: Add the elevation data to the *Travel Model Two* network.

This script reads a text file with node number and z-coordinate and sets the node z-coordinate value in the *Travel Model Two* Cube network. The script can be run to update z-coordinates for all Cube network nodes, or for only those nodes with a z-coordinate of zero in order to save computation time.

Inputs: tana_sp_with_maz_taz_tap_centroids_connectors_routes_osm_bike.net
nodeElevations.csv

Outputs: tana_sp_with_maz_taz_tap_centroids_connectors_routes_osm_bike.net

3.10 addBikePedOk.s

Purpose: Codes the bridge links that are open to bike and pedestrian traffic.

This script reads a CSV file of link records with the following fields: A, B, BIKEPEDOK. It then sets the BIKEPEDOK link attribute in the network to 1 if the link is open to bike and pedestrian traffic.

Inputs: tana_sp_with_maz_taz_tap_centroids_connectors_routes_osm_bike.net
bikepedok.csv

Outputs: tana_sp_with_maz_taz_tap_centroids_connectors_routes_osm_bike.net

3.11 buildBikeNetwork.bat

Purpose: Execute all the bicycle network scripts.

⁵ http://gisdata.usgs.net/xmlwebservices2/elevation_service.asmx

4 Analysis

Overall, the resulting bicycle network is useful, as it contains the majority of the Bike Mapper network information (i.e. links and their attributes). The bicycle build process does not capture every possible Bike Mapper link, however, for a number of reasons, as follows:

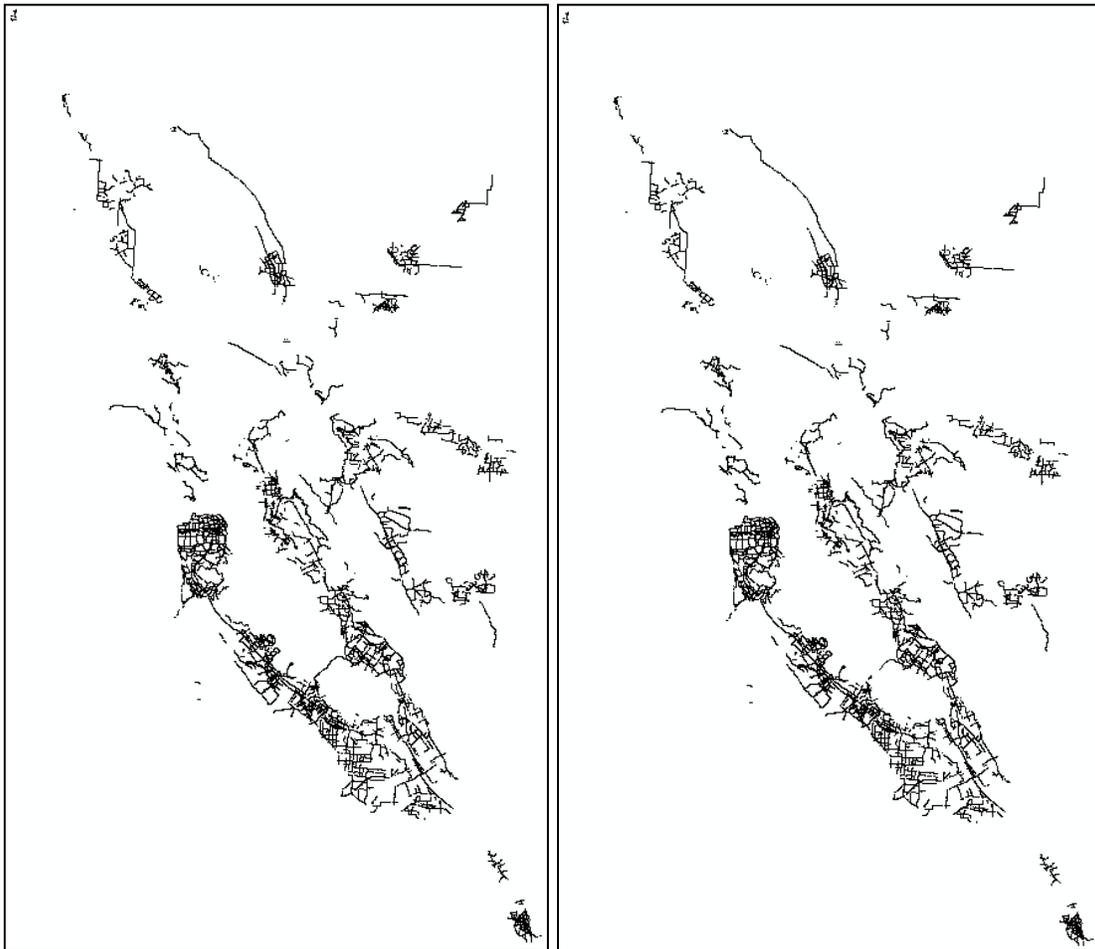
- The Bike Mapper network may have links which, while actual streets, do not exist in the version of the TeleAtlas network from which the *Travel Model Two* roadway network (“street network” is constructed).
- The Bike Mapper network may contain links which are not true streets (i.e., bicycle trails) which would not normally be included in the street network.
- Registration issues may render the 50 foot buffer for identifying the Bike Mapper-street network correspondence insufficient.
- Because of the inexact nature of the buffer analysis, it is possible that streets which are not truly bicycle links are identified as such.

The remainder of this section of the paper discusses some potential areas of improvement for the bicycle network build process and the resulting bicycle network.

4.1 Street Bicycle Links

To understand the scope of the resultant non-trail bicycle network, some statistics and maps have been created. In Figure 1, the Bike Mapper *on-street bicycle link* network and the bicycle street network are displayed side-by-side. The Bike Mapper network is on the left and the resultant bicycle street network on the right.

Figure 1 – Bike Network Comparison



At a regional scale, the two networks are nearly identical. In fact, as the table below indicates, only 70 Bike Mapper links are not included in the *Travel Model Two* network.

Table 2 - Street Bike Network Results

| Result Description | Value |
|----------------------------------------------------------|---------|
| Total Bike Mapper street links | 48,463 |
| Bike Mapper links not included in bicycle street network | 70 |
| Total bicycle street network links | 112,276 |

An analysis of the 70 Bike Mapper links that are not included in the *Travel Model Two* network shows that none of them are significant enough (i.e., a major bicycle trail or greenbelt) to require adding them to the network in a separate automated step at this point. Figure 2 is indicative of the missing links. The red links are the street network links (not just those identified as bicycle links) and the purple links are the

Bike Mapper links that were not matched to the street network. The shaded buffers identify the Bike Mapper network. As shown in Figure 2 and Figure 3 below, the missing Bike Mapper links do not even exist in the street network, but their exclusion is of little effect since plenty of alternative paths exist to traverse the area on bicycle.

Figure 2 - Bike Mapper Network Missing Links

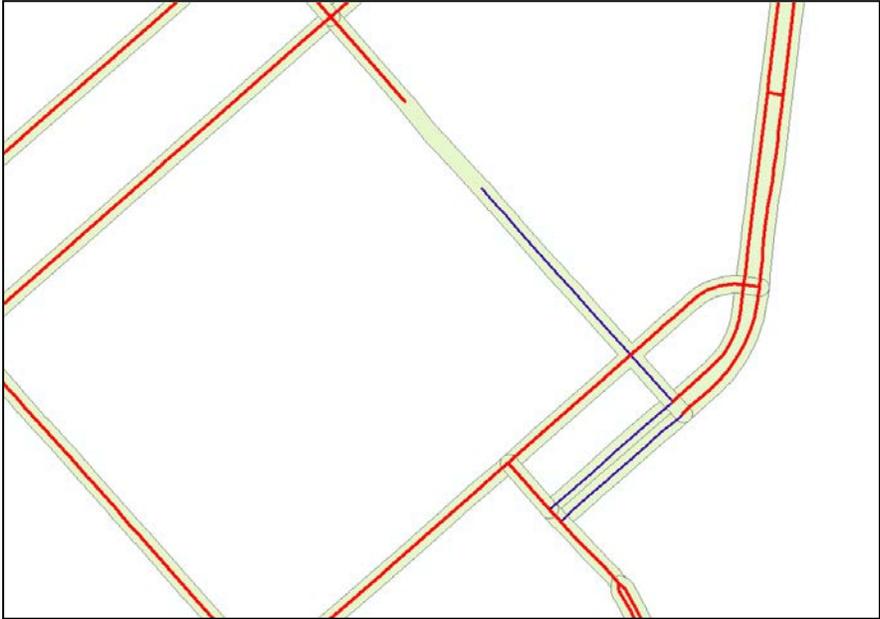
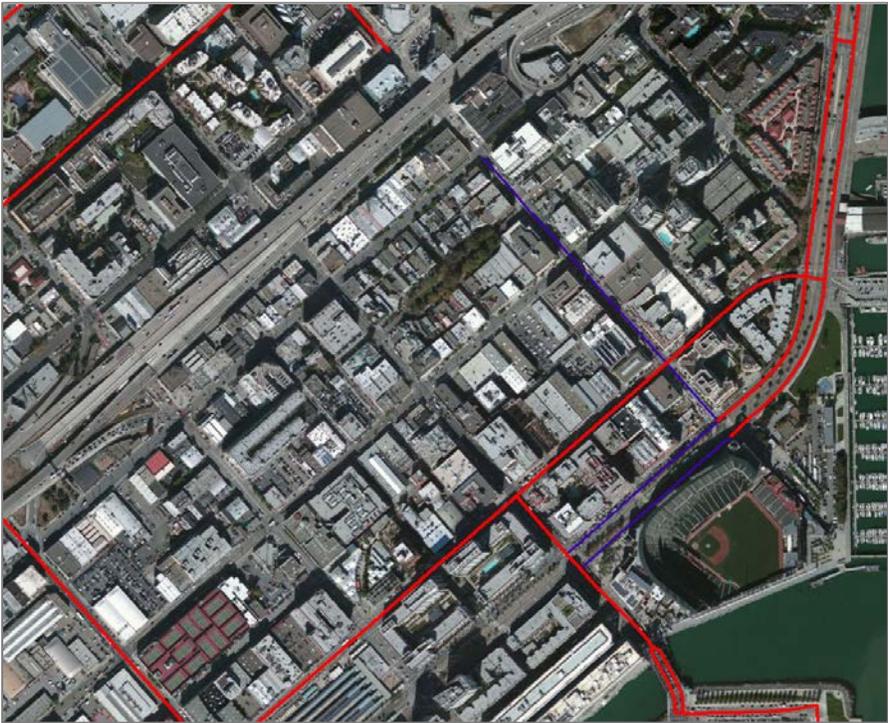


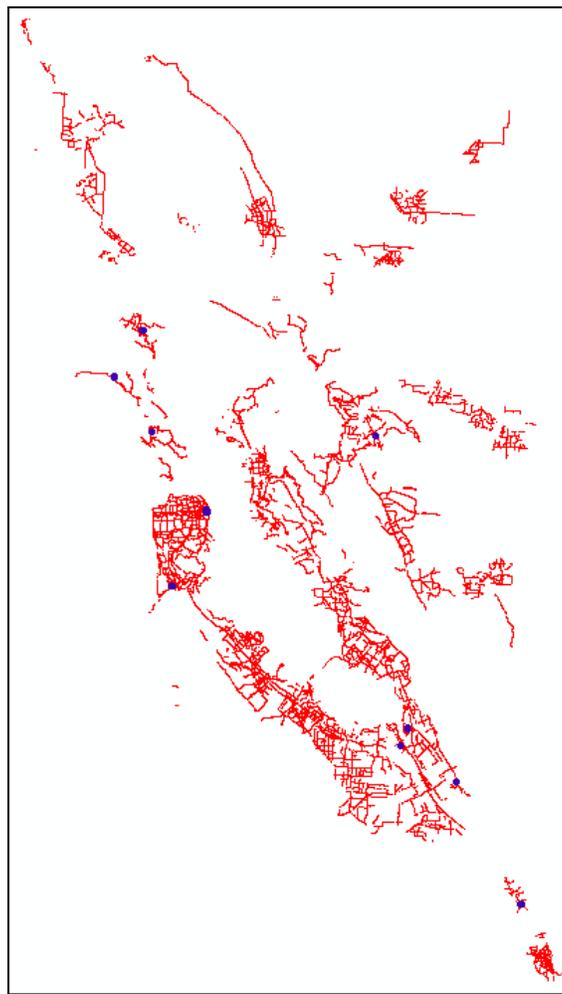
Figure 3 - Bike Mapper Network Missing Links with Aerial Photo



Another potential issue with the missing links is that they could have some sort of spatial or other type of correlation, which would indicate a systemic issue with the process. Figure 4 shows the entire street bicycle network, with the missing links highlighted in purple. There does not appear to be any obvious correlations or patterns (other than the fact that two or three missing links might be clustered together, as in Figure 2).

To overcome this issue, the handful of links which should have been included in the network, but were not, were specified in the `missing_links.csv` file, which was read into Cube and added to the network. Additionally, this issue did not apply to the non-street bicycle links (i.e. trails), as the missing links were assumed to be un-coded, and subsequently incorporated as new links in the network.

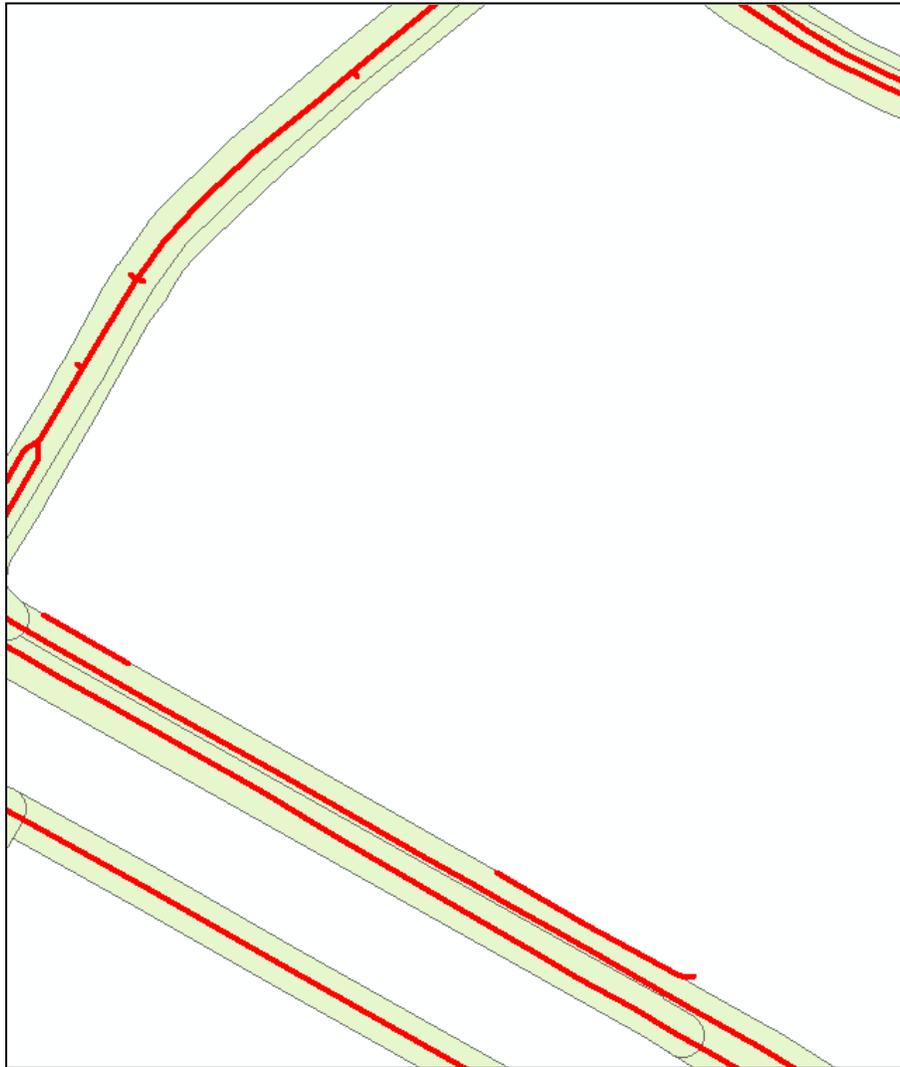
Figure 4 - Bike Mapper Network Missing Links



The final noteworthy issue is that the number of Bike Mapper links is about 40 percent of the total number of bicycle street network links. This seems to be a result of the buffers capturing non-bicycle

links, as shown in the following figure. These extraneous links are not of concern though, since they are either dead-end or orphan links, which have little impact on the assignment and skimming procedures.

Figure 5 - Bicycle Street Network Additional Links



4.2 Bicycle Trail Links

The bicycle trail links presented a somewhat more difficult problem than the street bicycle links. The primary issue with the trails is that even though the pedestrian network (which is what most trail links will map to) has better curvature/geometry embedded within it (due to a higher node count along segments), it is still difficult to capture the geographic relationships without using overly-wide buffers. Further, even though many segments of a bicycle trail may be mapped, the increased granularity of the pedestrian links means that there is a greater chance that some of them may not get captured, leading to a greater chance of network discontinuity. Figure 6 illustrates this problem. In the figure, the red links are

the bicycle network links, including the street/pedestrian network links mapped to bicycle trail links; the grey links are the rest of the street/pedestrian network; the shaded buffers identify the Bike Mapper bicycle trail network; the blue links identify the bicycle trail links not mapped to the street/pedestrian network. These missing links were added in the next step of the procedure.

Figure 6 - Bicycle Trail Network Missing (Unmapped) Links

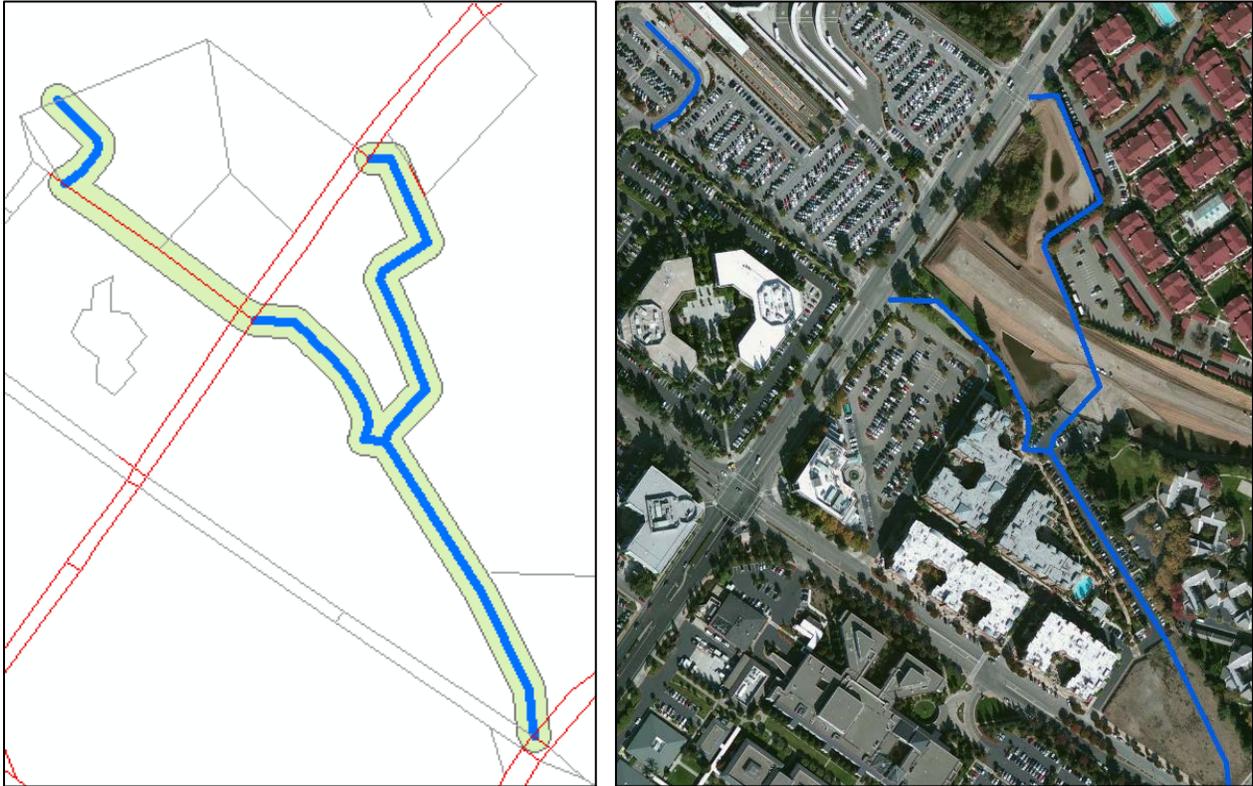
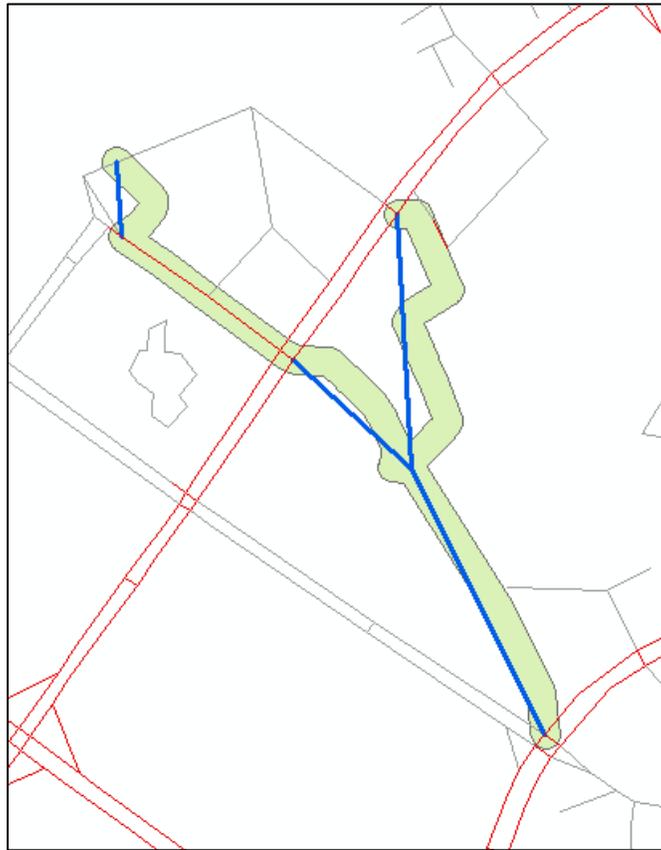


Table 3 provides statistics on the bicycle trails network. It shows that the number of missing bicycle trail links is much larger in number than for the street bicycle links; it is, therefore, not feasible to check each one by hand. However, the procedure for adding in the missing bicycle trail links, which is necessary regardless because many bicycle trails are neither roadways nor sidewalks, solves the connectivity issue. This is shown in Figure 7 below, where the red links are the bicycle network links, including the street/pedestrian network links mapped to bicycle trail links; the grey links are the rest of the street/pedestrian network; the shaded buffers identify the Bike Mapper bicycle trail network; the blue links identify the new bicycle trail links representing those not mapped to the street/pedestrian network. Compared to Figure 5, the connectivity of the bicycle trail links is retained.

Table 3 - Bike Trail Network Results

| Result Description | Value |
|-------------------------------------------------------------------------|--------------|
| Total Bike Mapper trail links | 877 |
| Bike Mapper trail links not fully captured in street/pedestrian network | 631 |
| Total street/pedestrian network links mapped as trail links | 27,214 |

Figure 7 - Bicycle Trail Network with Unmapped Links Added



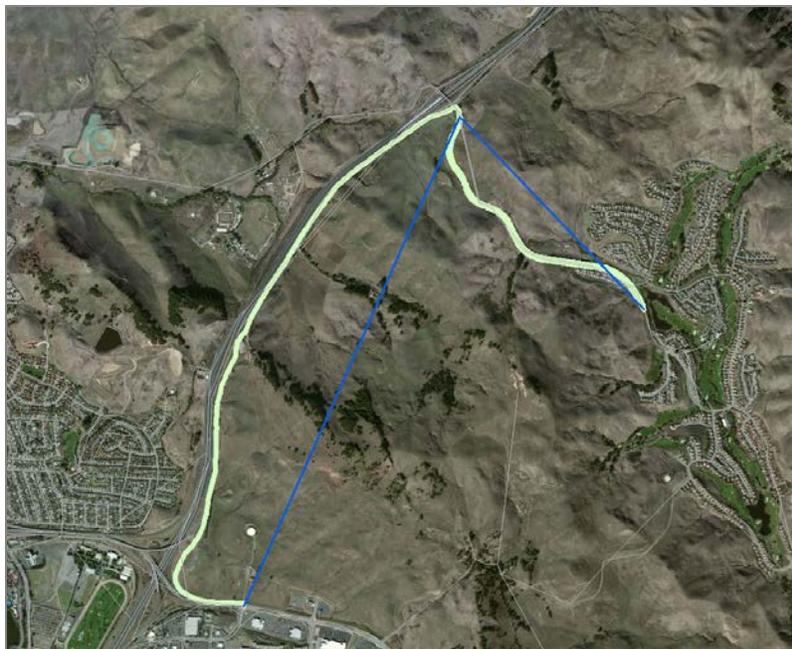
Unfortunately, many of the bicycle trail links are very long with respect to both the street/pedestrian network links and the trail links' overall curvature. This is the reason that the number of mapped street/pedestrian links listed in Table 3 is so much higher than the number of trail links. Once the trail links are imported into the Cube network, their curvature information is lost and they are represented just as straight line links between the start and end nodes. The length information is correctly retained, as it is measured before this conversion process, and the lack of shape has no impact on the other bike attributes. An example of this is shown below in Figure 8 and Figure 9. The grey links in the figure are the street/pedestrian network; the shaded buffer identifies a Bike Mapper bicycle trail link; the blue link

identifies the bicycle trail link's network representation. It is suggested that these links, which can be identified by the CNTYPE = 'BIKE' link filter, be manually edited by inserting nodes at key points where needed, as time and resources allow. In some cases, a new node may be inserted, and in other cases, an existing node may be used in order to connect the trail link into the rest of the network.

Figure 8 - Bicycle Trail Network Link Detail



Figure 9 - Bicycle Trail Network Link Detail with Aerial Photo



5 Skimming the Bicycle Network

In addition to the explicit bicycle street network links (CNTYPE=BIKE), bicyclists are allowed to use connectors (FT=0), expressways (FT=7), collectors (FT=4), arterials (FT=7), and bridges that allow bikes and pedestrians (BIKEPEDOK=1). For calculating MAZ to MAZ impedances, bicyclists are not allowed to use TAZ connectors (CNTYPE=TAZ) or TAP connectors (CNTYPE=TAP). The `shortestPathBike.s` script will generate MAZ to MAZ bicycle impedances according to these constraints.

Purpose: Generate a shortest path cost file between MAZs using the bicycle network.

Inputs: `tana_sp_with_maz_taz_tap_centroids_connectors_routes_osm_bike.net`

Outputs: `maz_bicycle_costs.csv`, a CSV file with fields I, J, NODE, COST (FEET)

6 Revisions to the Overall Network Build Process

Due to the splitting of links in order to connect new bicycle links (and pedestrian links) to the original TeleAtlas network links, the overall network build process was adjusted. The primary reason for adjusting the process was to be able to re-route the transit lines over the network links, which changed in some cases due to the new nodes. As a result, the overall network build procedures are run in the following order:

1. Build the highway network, including TAZ and MAZ centroid locations and connectors.
2. Build the transit line input file and add the TAP connectors to the highway network.
3. Build the pedestrian network links and add them to the highway network.
4. Build the bicycle network links and add them to the highway network.
5. Read the transit lines (developed in step 2) and read them into the highway network.