**Classifying Intersections and Interchanges**

****

**Spatial Solutions**

**Patrick Sullivan-Project Manager**

**Sam Johnston-GIS Analyst**

**Andrew Goin-GIS Analyst**

**Tyler Armstrong-GIS Analyst**

**October 29, 2018**

**Table of Contents**

1. **Introduction 1**
   1. Summary 1
   2. Purpose 1
   3. Scope 1
2. **Tasks 1**
   1. Eliminate Data Noise 1
   2. Run Intersect and Join Attributes to Point 1
   3. Classify Intersection by Angle and Number of Legs 1
   4. Timetable 2
3. **Problems Encountered 2**
   * 1. Using Model Builder 2
     2. Angle Errors 3
     3. Changing Milestones to Tasks 3
4. **Conclusion 3**
5. **Appendix 4**
   * 1. *Figure A*  4
     2. *Figure B*  5
     3. *Figure C* 5
     4. *Figure D* 6

**1. Introduction**

1.1 Summary

Following the acceptance of our proposal, Spatial Solutions has continued our efforts in developing a productive tool for the Texas Department of Transportation (TxDOT). This progress report is to inform our contacts at TxDOT on the improvements and the direction of our group in the completion of the project.

1.2 Purpose

The purpose of this project is to create a tool that produces attributes and locates the MIRE elements previously stated by TxDOT. With this tool, TxDOT can maintain better records of these attributes to report on safety and preventative maintenance of state-maintained roadways. TxDOT will be able to take this tool and apply it to road networks in counties across Texas.

1.3 Scope

The scope of the project still remains the same, Travis county and classifying intersections and interchanges based upon certain attributes. As mentioned before, after completing the tool, it will be applicable to other areas of Texas.

**2. Tasks**

2.1 Eliminate Data Noise

The first task we completed for the project was to eliminate all data noise in the form of false-positives on intersections and off-system roads that are not maintained by TxDOT. This was achieved by querying out all roads with specific route prefixes, with the exception of those that intersect an on-system route. Once the data noise was eliminated, we were able to run the next task without the possibility of producing inaccurate or irrelevant results.

2.2 Run Intersect and Join Attributes to Point

The second task completed was the running of the intersect tool on the cleaned dataset. This produced multiple intersection points across Travis County that needed the attributes from the intersecting lines appended to it. To achieve this a join was performed between the roads layer and the new point layer created by the intersect tool. With the needed attributes joined to the point layer, the classification of the intersections by the angles that they intersect at is possible.

2.3 Classify Intersection by Angle and Number of Legs

Classifying the angles has been the most difficult task thus far. Originally, we tried developing a python script (*Figure A*) to generate a table that contains all of the near angles for the buffer intersect points.Then the buffer intersect point angles would be joined to the intersection points for the on and off roadways. There were two main problems with this code; First the python script did not use pandas to organize and arrange the data where necessary. Second, if we changed our methodology, this program would not be flexible enough to change if new problems occurred. We would then create a new program for every new edit we would encounter. This is not an effective use of time, so we decided to create a model builder instead.

Now that we have analyzed the python script, we will describe the functionality of the model builder. Here is an explanation of how each of the model builders work:

Angle Classification:

1. Import the clipped on-system and off-system road shapefiles.
2. Use the intersect tool to generate the points where the on and off system roads meet.
3. Generate buffers with a radius of 10 feet.
4. Create intersection points where the buffer and roads meet.
5. Make a near angle table using the intersection points, buffer points, and near tool.

Intersection Classification:

1. Join the angle table to the intersection points attribute table.
2. Use a selection query for the desired angle range to classify the intersection.

2.4 Timetable

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Task | **Week Date** | | | | | | | | |
| 10/1 | 10/8 | 10/15 | 10/22 | 10/29 | 11/5 | 11/12 | 11/19 | 11/26 |
| Data Setup | ✓ |  |  |  |  |  |  |  |  |
| Intersection Angle |  | ✓ |  |  |  |  |  |  |  |
| Junction Geometry |  |  |  |  | ✓ |  |  |  |  |
| Number of Legs |  |  |  |  |  |  |  |  |  |
| Unique Identifiers |  |  |  |  |  |  |  |  |  |
| Location Identifiers |  |  |  |  |  |  |  |  |  |
| Deliverables |  |  |  |  |  |  |  |  |  |

*Figure B: Revised Timetable.*

**3. Problems Encountered**

3.1Using Model Builder

We have found that using Model Builder instead of Python has improved efficiency in our workflow (*Figure C*). This is partly because Model Builder is more approachable for all members of our team, and in turn for future users of this tool. That being said, some of our tasks are still better suited for the use of Python. We are planning on using a combination of both to create an output similar to (*Figure D*).

3.2 Angle Errors

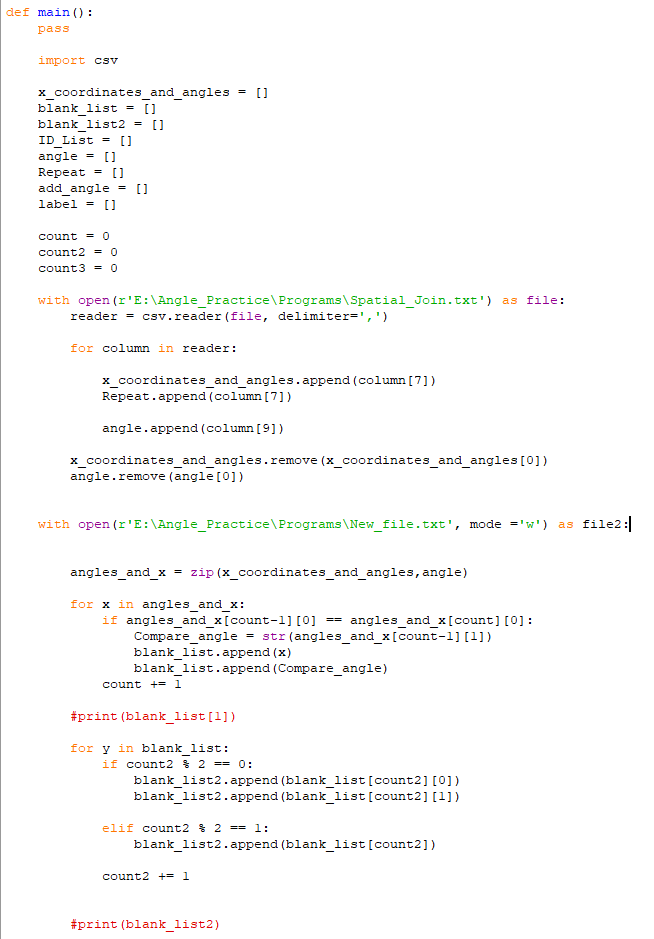
Currently, we are unsure what points the near angle function analyzes. We will do accuracy testing in the future to determine what points and angles are being calculated from the buffer points and the intersection points.

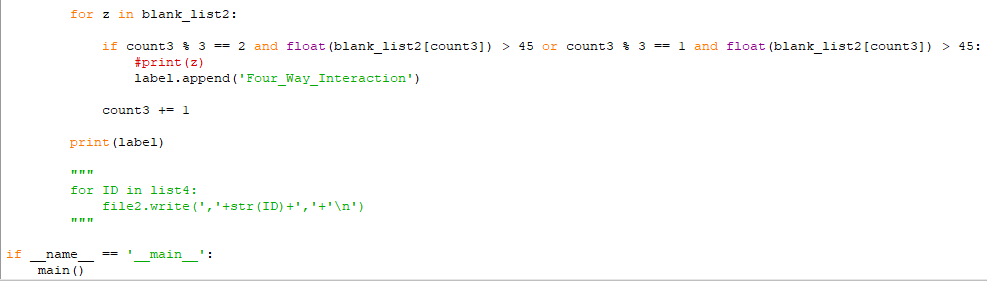
3.3 Changed Milestones to Tasks

For the purposes of organization, we decided a better approach of organization is to switch from using milestones to tasks. The tasks better correspond to what the client specifically asked for in the request for proposal. It is now easier to assign work and measure overall progress.

**4. Conclusion**

In closing, our progress towards our final output has been steady and consistent. The decision to switch our focus to building the tools in Model Builder first has not hindered our progress or pushed back our schedule. Additionally, it will also make for a more readable toolset for future use by analysts who might not be as well versed in programming. We will continue to meet our deadlines and look forward to presenting our deliverables on December 5th.

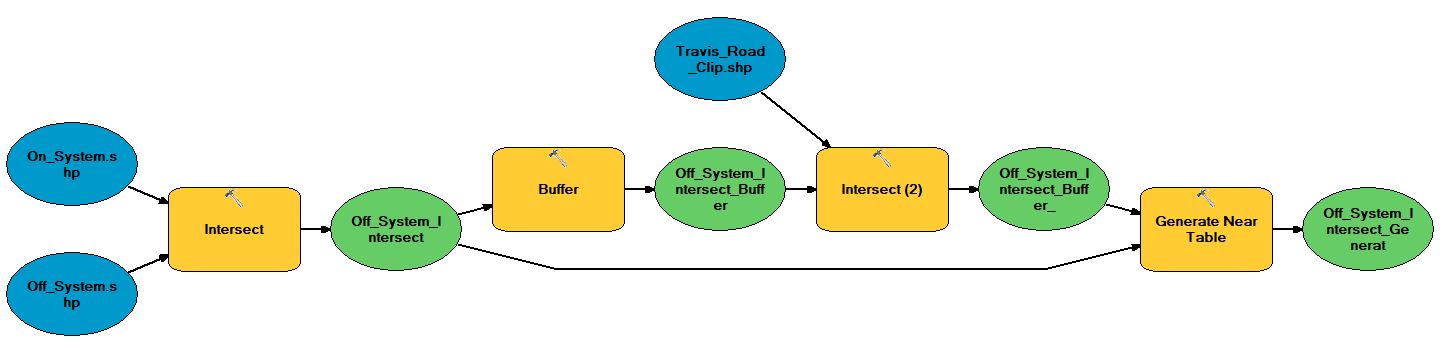
**5. Appendix**

****

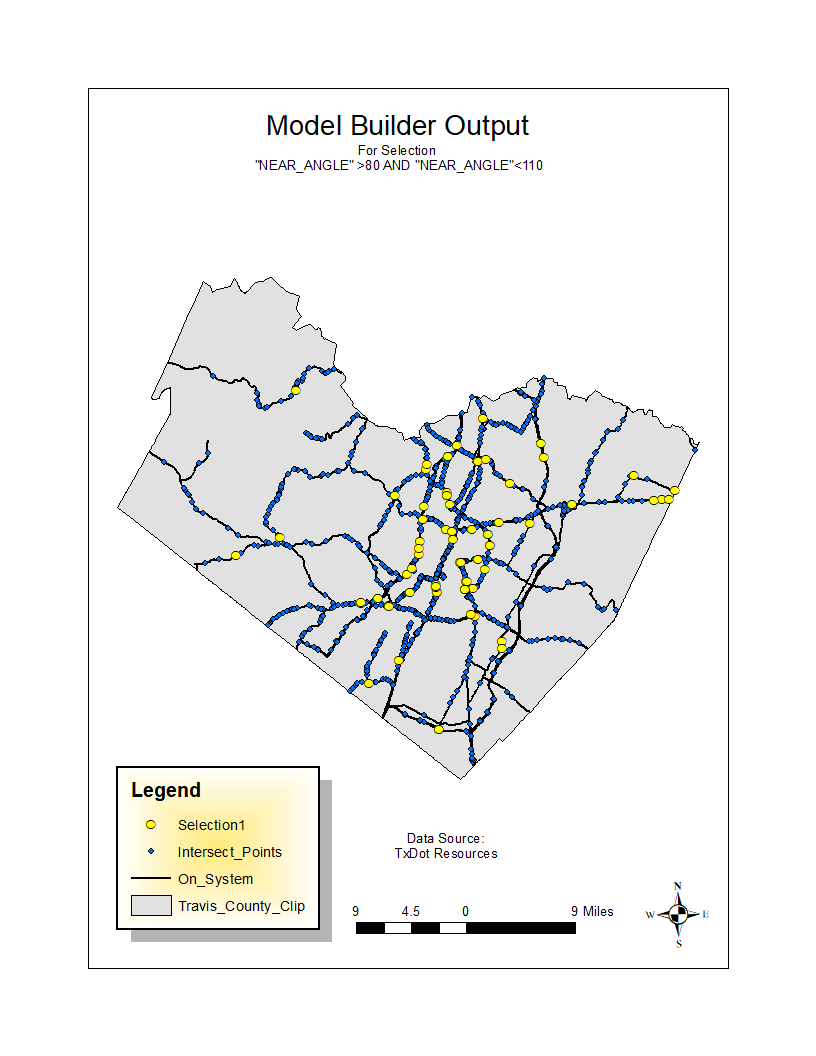
*Figure A: Python Script to generate near angles.*

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Task | **Week Date** | | | | | | | | |
| 10/1 | 10/8 | 10/15 | 10/22 | 10/29 | 11/5 | 11/12 | 11/19 | 11/26 |
| Data Setup | ✓ |  |  |  |  |  |  |  |  |
| Intersection Angle |  | ✓ |  |  |  |  |  |  |  |
| Junction Geometry |  |  |  |  | ✓ |  |  |  |  |
| Number of Legs |  |  |  |  |  |  |  |  |  |
| Unique Identifiers |  |  |  |  |  |  |  |  |  |
| Location Identifiers |  |  |  |  |  |  |  |  |  |
| Deliverables |  |  |  |  |  |  |  |  |  |

*Figure B: Revised Timetable.*

****

*Figure C: Example of Model Builder tool that intersects on and off-system roads, applies a buffer, and extracts angles from buffer points.*

****

*Figure D: Example Model Builder Output showing points of intersection.*