Appendix F: Calculated Values for Shape Point Algorithm Examples

Wrightstown Road and Cooper Road, Washington’s Crossing, PA

North Link

Original number of shape points: 12
Bin size: 7.5 %

Calculated Points:

<table>
<thead>
<tr>
<th>Longitude</th>
<th>Latitude</th>
</tr>
</thead>
<tbody>
<tr>
<td>-74.894053</td>
<td>40.2861980</td>
</tr>
<tr>
<td>-74.894418</td>
<td>40.2858722</td>
</tr>
<tr>
<td>-74.895135</td>
<td>40.2852593</td>
</tr>
</tbody>
</table>

**No data was map matched to any other bins.

East Link

Original number of shape points: 13
Bin size: 6.923 %

Calculated Points:

<table>
<thead>
<tr>
<th>Longitude</th>
<th>Latitude</th>
</tr>
</thead>
<tbody>
<tr>
<td>-74.881872</td>
<td>40.2857893</td>
</tr>
<tr>
<td>-74.882941</td>
<td>40.2859992</td>
</tr>
<tr>
<td>-74.884057</td>
<td>40.2861465</td>
</tr>
<tr>
<td>-74.88499</td>
<td>40.2860252</td>
</tr>
<tr>
<td>-74.885915</td>
<td>40.2856681</td>
</tr>
<tr>
<td>-74.886937</td>
<td>40.2853816</td>
</tr>
<tr>
<td>-74.887933</td>
<td>40.2851483</td>
</tr>
<tr>
<td>-74.888891</td>
<td>40.2848653</td>
</tr>
<tr>
<td>-74.889967</td>
<td>40.2846489</td>
</tr>
<tr>
<td>-74.891058</td>
<td>40.2844806</td>
</tr>
<tr>
<td>-74.892038</td>
<td>40.2844935</td>
</tr>
<tr>
<td>-74.893023</td>
<td>40.2845701</td>
</tr>
<tr>
<td>-74.89402</td>
<td>40.2844647</td>
</tr>
</tbody>
</table>
Wrightstown Road and Cooper Road, Washington’s Crossing, PA

Lines derived from sensed data:

<table>
<thead>
<tr>
<th></th>
<th>Average heading</th>
<th>Slope</th>
<th>B value</th>
</tr>
</thead>
<tbody>
<tr>
<td>North link</td>
<td>63.6473684</td>
<td>0.49537428</td>
<td>-77.38684</td>
</tr>
<tr>
<td>East link</td>
<td>69.1706044</td>
<td>0.38045156</td>
<td>-68.776571</td>
</tr>
<tr>
<td>West link</td>
<td>63.5518939</td>
<td>0.49745125</td>
<td>-77.541306</td>
</tr>
</tbody>
</table>

Intersections of derived lines:

<table>
<thead>
<tr>
<th>Links</th>
<th>Longitude</th>
<th>Latitude</th>
</tr>
</thead>
<tbody>
<tr>
<td>North and east</td>
<td>74.922259</td>
<td>-40.272281</td>
</tr>
<tr>
<td>North and west</td>
<td>74.3704348</td>
<td>-40.54564</td>
</tr>
<tr>
<td>West and east</td>
<td>74.912463</td>
<td>-40.276007</td>
</tr>
</tbody>
</table>

Computation of final location of intersection:

<table>
<thead>
<tr>
<th></th>
<th>Longitude</th>
<th>Latitude</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimum</td>
<td>74.3704348</td>
<td>-40.54564</td>
</tr>
<tr>
<td>Maximum</td>
<td>74.922259</td>
<td>-40.272281</td>
</tr>
<tr>
<td>Average</td>
<td>74.6463469</td>
<td>-40.40896</td>
</tr>
</tbody>
</table>

No measure of fit computed.

Final location for the intersection:

Latitude: 40.284320
Longitude: -74.895283

**Note that the algorithm left the original values for the latitude and longitude because the suggested numbers did not pass the final check because they were farther away than the distance of the longest link.**
Claremont Boulevard and Portola Drive, San Francisco, CA

Lines derived from sensed data:

<table>
<thead>
<tr>
<th></th>
<th>Average heading</th>
<th>Slope</th>
<th>B value</th>
</tr>
</thead>
<tbody>
<tr>
<td>North link</td>
<td>15.1</td>
<td>3.7061647</td>
<td>-491.61638</td>
</tr>
<tr>
<td>East link</td>
<td>62.2667</td>
<td>.5257541</td>
<td>-102.12638</td>
</tr>
<tr>
<td>West link</td>
<td>76.8</td>
<td>.23454788</td>
<td>-66.463724</td>
</tr>
</tbody>
</table>

Intersections of derived lines:

<table>
<thead>
<tr>
<th>Links</th>
<th>Longitude</th>
<th>Latitude</th>
</tr>
</thead>
<tbody>
<tr>
<td>North and east</td>
<td>122.465315</td>
<td>-37.73974</td>
</tr>
<tr>
<td>North and west</td>
<td>122.465314</td>
<td>-37.739744</td>
</tr>
<tr>
<td>West and east</td>
<td>122.465301</td>
<td>-37.739747</td>
</tr>
</tbody>
</table>

Computation of final location of intersection:

<table>
<thead>
<tr>
<th></th>
<th>Longitude</th>
<th>Latitude</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimum</td>
<td>122.465301</td>
<td>-37.739747</td>
</tr>
<tr>
<td>Maximum</td>
<td>122.465315</td>
<td>-37.73974</td>
</tr>
<tr>
<td>Average</td>
<td>122.465308</td>
<td>-37.739744</td>
</tr>
</tbody>
</table>

Measure of fit:

Before: .805965  
After: .7963636

Final location for the intersection:

Latitude: 37.739880  
Longitude: -122.464951

**Note that the algorithm left the original values for the latitude and longitude because the suggested numbers did not pass the final check because measure of fit was worse than the original.**
Monterey Boulevard and San Benito Way, San Francisco, CA
With assumption of straight roads through intersection.

**Lines derived from sensed data:**

<table>
<thead>
<tr>
<th>Links</th>
<th>Average heading</th>
<th>Slope</th>
<th>B value</th>
</tr>
</thead>
<tbody>
<tr>
<td>North/south link</td>
<td>178.835</td>
<td>-49.17415</td>
<td>5984.4768</td>
</tr>
<tr>
<td>East/west link</td>
<td>94.318333</td>
<td>-.075512</td>
<td>-28.48531</td>
</tr>
</tbody>
</table>

**Intersections of derived lines:**

<table>
<thead>
<tr>
<th>Links</th>
<th>Longitude</th>
<th>Latitude</th>
</tr>
</thead>
<tbody>
<tr>
<td>North/south and east/west</td>
<td>122.46699</td>
<td>-37.73306</td>
</tr>
</tbody>
</table>

**Measure of fit:**

Before: .7716112  
After: .8059563

**Final location for the intersection:**

Latitude: 37.73306  
Longitude: -122.46699
Appendix E: Calculated Values for Intersecting Lines
Algorithm Examples

Monterey Boulevard and San Benito Way, San Francisco, CA
Without assumption of straight roads through intersection.

Lines derived from sensed data:

<table>
<thead>
<tr>
<th></th>
<th>Average heading</th>
<th>Slope</th>
<th>B value</th>
</tr>
</thead>
<tbody>
<tr>
<td>North link</td>
<td>175.05</td>
<td>-11.54609</td>
<td>1376.2817</td>
</tr>
<tr>
<td>South link</td>
<td>182.62</td>
<td>21.85337</td>
<td>-2714.048</td>
</tr>
<tr>
<td>East link</td>
<td>95.786</td>
<td>-0.10134</td>
<td>-25.32217</td>
</tr>
<tr>
<td>West link</td>
<td>92.85</td>
<td>-0.049783</td>
<td>-31.63629</td>
</tr>
</tbody>
</table>

Intersections of derived lines:

<table>
<thead>
<tr>
<th>Links</th>
<th>Longitude</th>
<th>Latitude</th>
</tr>
</thead>
<tbody>
<tr>
<td>North and west</td>
<td>122.466947</td>
<td>-37.733052</td>
</tr>
<tr>
<td>North and south</td>
<td>122.466921</td>
<td>-37.732756</td>
</tr>
<tr>
<td>North and east</td>
<td>122.466953</td>
<td>-37.732069</td>
</tr>
<tr>
<td>West and south</td>
<td>122.466908</td>
<td>-37.733114</td>
</tr>
<tr>
<td>West and east</td>
<td>122.465672</td>
<td>-37.732989</td>
</tr>
<tr>
<td>South and east</td>
<td>122.466905</td>
<td>-37.733114</td>
</tr>
</tbody>
</table>

Computation of final location of intersection:

<table>
<thead>
<tr>
<th></th>
<th>Longitude</th>
<th>Latitude</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimum</td>
<td>122.465672</td>
<td>-37.733114</td>
</tr>
<tr>
<td>Maximum</td>
<td>122.466953</td>
<td>-37.732069</td>
</tr>
<tr>
<td>Average</td>
<td>122.466312</td>
<td>-37.732591</td>
</tr>
</tbody>
</table>

Final location for the intersection:
- Latitude: 37.732835
- Longitude: -122.466684

Measure of fit:
- Before: .7716112
- After: .72615

**Note that the algorithm left the original values for the latitude and longitude because the suggested numbers did not pass the final check because measure of fit was worse than the original.**
Wrightstown Road and Cooper Road, Washington’s Crossing, PA
(Continued)

24\textsuperscript{th} move. North.
Measure of fit: .71075

25\textsuperscript{th} move. South.
Measure of fit: .71745

Algorithm terminates with the optimal location found after 22\textsuperscript{nd} move.
Wrightstown Road and Cooper Road, Washington’s Crossing, PA
(Continued)

18th move. South.
Measure of fit: .72072

19th move. South.
Measure of fit: .72181

20th move. South.
Measure of fit: .71154

21st move. East.
Measure of fit: .71012

22nd move. West.
Measure of fit: .72475

23rd move. West.
Measure of fit: .72272
Wrightstown Road and Cooper Road, Washington’s Crossing, PA
(Continued)

12th move. East.
Measure of fit: .68008

13th move. West.
Measure of fit: .68025

14th move. West.
Measure of fit: .68927

15th move. West.
Measure of fit: .70496

16th move. West.
Measure of fit: .69375

17th move. North.
Measure of fit: .68216
Wrightstown Road and Cooper Road, Washington’s Crossing, PA
(Continued)

6th move. West.
Measure of fit: .65718

7th move. West.
Measure of fit: .67636

8th move. West.
Measure of fit: .67369

9th move. North.
Measure of fit: .66764

10th move. South.
Measure of fit: .68189

11th move. South.
Measure of fit: .67779
Wrightstown Road and Cooper Road, Washington’s Crossing, PA

Initial location.
Measure of fit: .59001

1st move. North.
Measure of fit: .60794

2nd move. North.
Measure of fit: .62163

3rd move. North.
Measure of fit: .63149

4th move. North.
Measure of fit: .62250

5th move. East.
Measure of fit: .60245
Claremont Boulevard and Portola Drive, San Francisco, CA
(Continued)

Measure of fit: .80044

7th move. South.
Measure of fit: .80570

Algorithm terminates with the optimal location found after the 4th move.
Claremont Boulevard and Portola Drive, San Francisco, CA

Initial location.
Measure of fit: .80596

1st move. North.
Measure of fit: .80180

2nd move. South.
Measure of fit: .80268

3rd move. East.
Measure of fit: .79729

4th move. West.
Measure of fit: .80789

5th move. West.
Measure of fit: .80559
Monterey Boulevard and San Benito Way, San Francisco, CA

(Continued)

12th move. South.
Measure of fit: .799992

13th move. East.
Measure of fit: .80268

14th move. West.
Measure of fit: .77787

Algorithm terminates with the optimal location found after the 11th move
Monterey Boulevard and San Benito Way, San Francisco, CA
(Continued)

6th move. West.
Measure of fit: .79303

7th move. West.
Measure of fit: .78589

8th move. West.
Measure of fit: .79904

9th move. West.
Measure of fit: .77636

10th move. North.
Measure of fit: .77294

11th move. South.
Measure of fit: .80820
Appendix D: Iterations and Measure of Fit for Iterative Algorithm Examples

Monterey Boulevard and San Benito Way, San Francisco, CA

Initial location. Measure of fit: .77161

1st move. North. Measure of fit: .77749

2nd move. North. Measure of fit: .78258

3rd move. North. Measure of fit: .78903

4th move. North. Measure of fit: .78535

5th move. East. Measure of fit: .78202
Sort ( )
{
    for each bin i,
    {
        for each data points map matched to link of interest
        {
            if(percent along link is in bin i)
            {
                if (snap_value is greater than snap_threshold),
                {
                    save data point to useful_points_i
                }
            }
        }
    }
}

Fit ( )
{
    for each bin j,
    {
        if there are any data points in bin i
        {
            new_lon_j=average longitude in useful_points_j
            new_lat_j=average latitude in useful_points_j
        }
    }
}
Appendix C: Pseudo-Code for Shape Point Algorithm

This algorithm assumes that the intersections at either end of the link have already been realigned.

**Definition of Variables:**

- `num_shape_pts`: the number of shape points that the link had prior to being straightened either in preparation for this algorithm or previously by an intersection realignment.
- `bin_size`: variable that saves the size of the bin number
- `Snap_threshold`: the value under which a data point is excluded
- `bins`: A vector that contains the bin thresholds.
- `useful_points`: A matrix of data points that have a snap value over a given for each bin
- `new_lats`: A vector of the latitudes for the new shape points that this algorithm creates.
- `new_lons`: A vector of the longitudes for the new shape points that this algorithm creates.

**Program:**

Start
  Make Bins ( )
  Sort ( )
  Fit ( )
End

Make Bins ( )
{
  \[
  bin\_size = \frac{90}{num\_shape\_points}
  \]
  for all \( bins_i \) (i from 0 to \( num\_shape\_points \))
  {
    \[
    bins_i = 5 + (bin\_size \ast i)
    \]
  }
}
Solve ( )
{
    for all \( \text{links}_j \)
    {
        for all \( \text{links}_{i,j} \)
        {
            \[ x_{\text{temp}}_{i,j} = \frac{b_i - b_j}{\text{slope}_j - \text{slope}_i} \]
            \[ y_{\text{temp}}_{i,j} = \text{slope}_j \left( \frac{b_i - b_j}{\text{slope}_j - \text{slope}_i} \right) + b_j \]
        }
    }
    \[ x_{\text{new}} = \frac{\min_{i,j}(x_{\text{temp}}) + \max_{i,j}(x_{\text{temp}})}{2} \]
    \[ y_{\text{new}} = \frac{\min_{i,j}(y_{\text{temp}}) + \max_{i,j}(y_{\text{temp}})}{2} \]
}

Check ( )
{
    if\{ \sqrt{(x - x_{\text{new}})^2 + (y - y_{\text{new}})^2} < \max(\text{links\_length}) \},
            \begin{align*}
                &\text{Export } x_{\text{new}} \text{ as longitude} \\
                &\text{Export } y_{\text{new}} \text{ as latitude}
            \end{align*}
    }
    else if\{
        \begin{align*}
            &\text{Export } x_{\text{new}} \text{ as longitude} \\
            &\text{Export } y_{\text{new}} \text{ as latitude}
        \end{align*}
    }
    else \{\text{stop program and alert the user}\}
}
Measure of Fit ( )
{
    for all data points $n$, with snap-values $s$, on all links $k$
    {
    \[ M = \frac{1}{K} \sum_{k} \left[ \frac{1}{n} \sum_{i=1}^{n} (s_i) \right] \]
    }
}

Average ( )
{
    for all links $j$
    {
    save the average heading for link $j$ as $\text{avg}_\text{heading}_j$
    }
}

Slope ( )
{
    for all links $j$
    {
\[ \text{slope}_j = -\tan(\text{avg}_\text{heading}_j + 90) \]
    }
}

Regression ( )
{
    for all links $j$ in data_points $i$
    {
    save $b_j$ as the value of $b_j$ that minimizes the following:
\[ \sum_{i} \left[ \text{latitude}_i - m \times \text{longitude}_i + b_j \right] \]
    }
}
Program:

Start

Import existing intersection longitude and save as x
Import existing intersection latitude and save as y
Call Data Manipulation ( )
Call Measure_of_fit ( )
Call Average ( )
Call Slope ( )
Call Regression ( )
Call Solve ( )
Call Check ( )

End

Data Manipulation ( )
{
    for all links_j
    {
        for all data_points_i
        {
            if { t_min ≤ t_value ≤ t_max, 
            store heading_i mod(180) in useful_heading_j
            if { heading_i ≤ deg_threshold 
            set Near_North_j equal to 1
            
            }
        }
        if { Near_North_j = 1, 
        for all useful_headings_i
        {
            if { useful_heading_i < 90, 
            add 180 to that heading
            
            }
        }
    }
}
Appendix B: Pseudo-Code for Intersecting Lines Algorithm

**Definition of Variables:**

- \( x \)= longitude of the original intersection location.
- \( y \)= latitude of the original intersection location.
- \( x_{\text{new}} \)= the updated longitude.
- \( y_{\text{new}} \)= the updated latitude.
- \( t_{\text{min}} \)= minimum allowable percent for a data point to be along its link in order to be used for the average heading calculation.
- \( t_{\text{max}} \)= maximum allowable percent for a data point to be along its link in order to be used for the average heading calculation.
- \( \text{deg}_{\text{threshold}} \)= If there are any values under this number, 180 degrees will be added to any heading under 90 degrees. A higher value should be used when data is less consistent.
- \( M \)= Measure of fit.
- \( \text{Near}_{\text{North}} \)= a vector of Boolean variables to indicate whether the heading of a given link is near North. Initialize to 0.
- \( \text{data}_{\text{points}} \)= a three-dimensional matrix that has longitude, latitude, and heading for each data point for each link.
- \( \text{links} \)= a vector of the link numbers emanating from the intersection of interest.
- \( \text{avg}_{\text{heading}} \)= a vector where the average heading for each link is stored.
- \( \text{slope} \)= a vector where the slope of each link is stored.
- \( b \)= a vector that contains the y-intercepts of the regressed lines for each link.
- \( \text{x}_{\text{temp}} \)= an \( n \) by \( n \) matrix where the x-coordinates of each line intersection can be stored and where \( n \) is the number of links emanating from the intersection of interest.
- \( \text{y}_{\text{temp}} \)= an \( n \) by \( n \) matrix where the y-coordinates of each line intersection can be stored and where \( n \) is the number of links emanating from the intersection of interest.
- \( \text{link}_{\text{lengths}} \)= a vector that contains the original lengths of each link.
Evaluate function ( )
{
    Map match data with intersection at (x_temp, y_temp)
    Call Measure of Fit and save as M_temp
    if { M_temp > M,
    
    Save M_temp as M
    Save x_temp as x
    Save y_temp as y
    Set i equal to 1
    call Move function ()

    } else {
    
    if { i=1,
     set i equal to 0
     if {d=1,
     set d equal to 2
     call Move function () } else if {d=2,
     set d equal to 1
     call Move function () } else if {d=3,
     set d equal to 2
     call Move function () } else { set d equal to 1
     call Move function () } }
    } else {
    
    if {d=1,
     set d equal to 3
     call Move function () } else if {d=2,
     set d equal to 4
     call Move function () } else { export x and y to database
     end program }

    } }
}
**Program:**

Start
- Import existing intersection longitude and save as $x$
- Import existing intersection latitude and save as $y$
- Call measure of fit and save as $M$
- Call Move function( )

Measure of Fit ( )
{ for all data points $n$, with snap-values $s$, on all links $k$

$$M = \frac{1}{k} \sum_{k} \left[ \frac{1}{n} \sum_{i=1}^{n} (s) \right]$$

}

Move function ( )
{ if{ $d=1$,  
    set $y_{temp}$ equal to $y+\delta$
    set $x_{temp}$ equal to $x$  }
else if { $d=2$,  
    set $x_{temp}$ equal to $x-\delta$
    set $y_{temp}$ equal to $y$  }
else if { $d=3$,  
    set $y_{temp}$ equal to $y-\delta$
    set $x_{temp}$ equal to $x$  }
else { set $x_{temp}$ equal to $x+\delta$
    set $y_{temp}$ equal to $y$  }

call Evaluate function()
Appendix A: Pseudo-Code for Iterative Algorithm

**Definition of Variables:**

$x$=longitude of the intersection location with the highest value for measure of fit yet

$y$=latitude of the intersection location with the highest value for measure of fit yet

$x_{\text{temp}}$=temporary storage for most recent longitude tried

$y_{\text{temp}}$= temporary storage for most recent latitude tried

$i$=variable to keep track of whether there has been an improvement in the given direction

Initialize to 0.

$d$=direction to move in (1=North, 2=East, 3=South, 4=West)

initialize to 1

$\text{delta}$=distance to move at each iteration

$M$=highest measure of fit so far

$M_{\text{temp}}$=temporary storage of most recent measure of fit


Bibliography:


Finally, if the ultimate goal is better transportation, rather than just improved maps, better position finding capabilities are needed. A reduction of error to below the one-meter level would be tremendous. Not only would it be easier to realign maps with such accuracy in the sensed data, but the necessity to do so would be lessened as the machine error would be small.
be used. Thus, I cannot say with certainty that any of the algorithms work or that they do not. I can make educated guesses based on how they behaved on the low-density data examples.

6.2. Areas for Further Study

As this is a first effort, there are more areas for further study than have been studied in this thesis. First and foremost, the algorithms in this thesis should be evaluated using larger data sets to see if the shortcomings are cured when there is more data which makes an average value less susceptible to outliers in the data. This is especially important given how critical average values are in the computations involved in these algorithms.

Additionally, there is the issue of how one deals with the fact that some streets have lots of data while their cross streets have none. The question of how one shifts one street while maintaining another one for which nothing is known must be answered. More intelligent algorithms which are more stable need to be developed as well.

The approach taken in this thesis was to work on realigning single points without worrying about how they fit into a bigger picture. The algorithms assume that an intersection found in the data corresponds to the closest intersection on the existing maps. Figure 6.1 shows a hypothetical example where this is not the case. It is perfectly clear by inspection how this setup should be realigned, but the computer is still unable to do this. This sort of problem needs to be addressed before any large-scale, automated realignment takes place.

Figure 6.1: Shows how the algorithms in this thesis would realign a map if given this sort challenge. The red represents the data, the black indicated the map as it is stored in the database both before (a) and after (b) realignment. The Green arrows indicate where the intersections will be map matched to.
6. Conclusion

The goal of this thesis was to develop and test algorithms that could be used to automate map realignment using sensed data. Specifically, I wanted to use the GPS data set that was available to me. By studying the characteristics of the data set, algorithms were created to try to realign the maps while minimizing the effects of the distortions in the data set. A standard measure of fit was created to analyze how well the data fit a given set of links or intersections. In total, three algorithms were created based on the assumption that the existing maps are more right than wrong. Two of the algorithms focus on realigning intersections. The first is an iterative algorithm which computes a measure of fit after each iteration slowly moves toward the optimal location for the intersection according to the measure of fit that I created. The second algorithm attempts to create a geometric model of the intersection and solve for the new coordinate algebraically. The third algorithm adds shape points to a link which has already had both of its end nodes realigned. It divides the link into different sections and updates each section individually. These algorithms were tested using the real world data with mixed results.

6.1. Interpretation of Results

The completion of this thesis clearly does not mark the end of inaccurate maps. Several useful contributions were made, most notably the fact that several intersections were updated automatically and that shape points were added automatically as well. The testing of the algorithms was limited by spreadsheet row limits which restricted the areas that could be tested. Any grid on the map that had more than 65,000 data points could not
5.4.3. **Shape Point Algorithm**

This algorithm proved to be a great success. It is surprisingly simple to understand and implement. I believe that its success is due to two primary reasons. First, a poor data point does not skew the computation very much. The data point must be map matched to the link and that will only happen if the data point is near to the link. In addition to the algorithm being relatively robust against bad data, the output is held to a different standard than that of the other algorithms. Shape points should guide the road, but do not necessarily need to be directly on the road. Conversely, with an intersection, it is very obvious where it belongs. These two factors, plus its simplicity, make the shape point algorithm very effective.
5.4.2. Intersecting Lines Algorithm

The performance of this algorithm was disappointing. While developing it, I had thought that it would be more robust than it turned out to be. As was stated previously, it is highly sensitive to changes in the average headings and this is what caused its poor performance here. It is also important to note that the algorithm is most likely functioning properly because the results are close to the correct solution. Even the Wrightstown and Cooper Roads intersection was only relocated fifteen miles. If the algorithm were not working properly, we might expect to see much greater variation.

Another issue with this algorithm is that because of the sensitivity it has, there has to be some sort of check at the end to ensure that it has not made a large error. However, as is seen in the preceding examples, designing a check to prevent the intersection from getting worse is a difficult task. While using the more general method, the check to ensure that the poorly realigned intersection at Monterey Boulevard and San Benito Way did not get saved prevented the intersection at Claremont Boulevard and Portal Drive from being saved despite that fact that it was a perfect update.

I include this algorithm in this thesis because, while I have found its performance to be lacking, I still believe that it might work with a sufficiently large data set. I used Microsoft’s Excel to do the computational work done in this chapter which means that I was limited to 65,000 data points for any grid that I wanted to look at. This meant that I could not do the necessary computations on the large data sets that I believe would give this algorithm the stability of heading that it needs. Furthermore, I have faith in that the ideas that this algorithm was built on and think that even if the algorithm, as written here, does not work, that it is still useful for the reader to read about and understand.
5.4. Discussion of the Performance of the Algorithms

The algorithms developed in this thesis varied drastically in their success. What follows is a brief discussion of the results of each algorithm and possible explanations of these results.

5.4.1. Iterative Algorithm

This algorithm performed as I had expected. For intersections with long links, it has been demonstrated to be very robust. Its only limitation seems to be the way in which the measure of fit is computed. I chose to use an average function, but there are other options. A measure of fit based on the sum of snap-values might work well, except that one still has the problem of what to do when data points are no longer map matched to a particular link anymore. An ideal measure of fit would always be at its highest value at an intersection in the data set regardless of link length and other factors.
5.3.2. **Intersecting Lines Algorithm**

The intersecting lines algorithm used on this intersection yielded the worst results of any update that was attempted. As the algorithm is written in the previous chapter, it would have left the intersection in its original location and alerted the user to an error. The algorithm’s suggested new location for the intersection is about two miles north of Princeton, NJ, or about fifteen miles away, which is clearly incorrect. The root of this enormous error lies in the fact that the links associated with this intersection are very long and have many turns in them. This makes the average headings less reliable than they are for straight links.

5.3.3. **Shape Point Algorithm**

The intersection of Wrightstown Road and Cooper Road is the only one whose links had shape points to begin with. Thus, it is the only intersection to have shape points added after successful intersection realignment (using the iterative algorithm). The shape point algorithm worked exceedingly well in this case. Figure 5.7 shows the intersection after the shape point algorithm has been applied to it. The road follows the data so well that in most places one cannot even see the road under the data. The measure of fit calculated after the intersection realignment was .72083 and after the shape point algorithm was run, the measure of fit jumped to .76253. This is a very high measure of fit for an intersection like this one with long, curvy links emanating from it.
5.3.1. Iterative Algorithm

The iterative algorithm worked very well in this location. The algorithm put the intersection in exactly the location that the data indicates. The shape points in the links surrounding this intersection must be removed before anything else. Figure 5.6 shows the intersection before updating with the shape points (a), the intersection before being updated but with the shape points removed (b), and finally the updated intersection. While the results are good, it required twenty-five iterations to achieve, which represents a lot of computing power. Again, all iterations and their respective measures of fit can be found in Appendix D.

Figure 5.6: The successful update of Wrightstown and Cooper Roads in Washington’s Crossing, PA. (a) Before any changes. (b) After removing the shape points, but with the intersection in its original location. (c) the intersection in its new location.
5.2.2. **Intersecting Lines Algorithm**

The intersecting lines algorithm appears to have performed well here. Figure 5.5 shows the intersection in its initial state (a) and also in its final state (b). The algorithm has placed the intersection exactly where the data indicates it belongs. However, the intersection’s measure of fit actually decreased during this move and so the algorithm did not save the location.

![Figure 5.5](image)

(a) Before. (b) After. However, due to the fact that the measure of fit is worse, it is not saved.

5.3. **Wrightstown Road and Cooper Road, Washington’s Crossing, PA**

This intersection was chosen because it has long links with many turns and because its initial state was very inaccurate.
5.2. Claremont Boulevard and Portola Drive, San Francisco, CA

This intersection was chosen because it is a three-way intersection with one very short link and two medium length links.

5.2.1. Iterative Algorithm

The iterative algorithm performed rather poorly on this intersection. Figure 5.4 shows the intersection both before (a) and after (b) the realignment. The intersection is only slightly improved. The poor performance here is due to the fact that the northern link is so short. When the algorithm tried to move the intersection west, the heading of the northern link changed drastically which caused the measure of fit to decrease due to the different heading rather than increase due to the improved positional accuracy.

![Figure 5.4](image)

(a) (b)

**Figure 5.4**: Updating the intersection of Claremont Boulevard and Portola Drive in San Francisco, CA using the iterative algorithm is unsuccessful. (a) Before. (b) After.

The performance of the algorithm would have been better had the intersection to the north been better aligned. Also, if one used a measure of fit that was based more on position than heading, the performance would have been enhanced. As before, each iteration and its measure of fit can be found in Appendix D.
problem well and has also done well with the short, straight links that are common in urban areas.

Figure 5.2: The intersecting lines algorithm using the assumption the intersection is two straight lines intersecting is successful at updating the intersection of Monterey Boulevard and San Benito Way in San Francisco, CA. (a) Before. (b) After.

Attempting to use the more general version of this algorithm to realign this intersection was a failure. As Figure 5.3 shows, the intersection was move farther away from where it belonged. The algorithm did not move the intersection in this case because the measure of fit decreased.

Figure 5.3: The more general version of the intersecting lines algorithm fails to realign the intersection. (a) Before. (b) After.

In this example, like all the others in this chapter, I used data points that were anywhere between fifteen and fifty percent along the link in the direction from the intersection of interest to calculate the average heading. These values could be changed, but, in this case, it does not seem that the results would have been different as all the links are straight and the data does not indicate any curves in the road.
5.1.2. Intersecting Lines Algorithm

Depending on what assumptions were made, the intersecting lines algorithm had both its only success and a failure at this intersection. The algorithm worked very well if it assumed that the this is a standard four-way intersection and treated it as two intersecting lines. The intersection of these two lines is exactly where the intersection on the road should be and, in fact, the final result is almost indistinguishable from that produced by the iterative algorithm. Figure 5.2 shows the intersection both before (a) and after (b) the algorithm was run. This algorithm appears to have handled the driveway
previous one and so the new location is saved as the new location of the intersection.

Also, because moving north improved the measure of fit, the algorithm moves north again. The algorithm continues to move the intersection north until the measure of fit does not improve between one iteration and the next. Here, this condition occurs after the fourth iteration. Figure 5.1 (c) shows the state of the map at the highest measure of fit attained by the algorithms first set of successive moves north.

When the measure of fit fails to improve, the algorithm reverts to the location that yielded the highest measure of fit and tries to move east. This is clearly wrong, but the algorithm does not realize this until the next iteration when the measure of fit fails to improve again. Because the measure of fit did not ever improve in the eastward direction, the algorithm attempts to move the intersection west. This is a success and the new location of the intersection is saved.

The algorithm moves westward until the measure of fit does not improve again. In this state, shown in Figure 5.1 (d), the measure of fit is .80820. The algorithm tries to move north and then south and because it does not have success in either of the directions, it terminates with the state of the map as shown in 5.1 (d). Each iteration, along with its respective measure of fit, can be found in Appendix D.

In this example, the algorithm clearly worked as it has placed the intersection in exactly the same location as the data indicates. It appears that the algorithm works well with short, straight links and is not affected by the driveway that is present on the eastern link that emanates from the intersection.
5. Implementation of Map Realignment Algorithms

This chapter concerns itself with several real world examples of the algorithms described in the previous chapter. These examples use GPS data as described in chapter three. I have selected three intersections that each have different characteristics such as number of links emanating from them, the length of those links, whether they have curves, and how good their initial measure of fit was.

5.1. Monterey Boulevard and San Benito Way, San Francisco, CA

This intersection was chosen primarily because I grew up in a house on the eastward link of it. However, this four-way intersection can also be used to evaluate how the algorithm copes with driveways and relatively short urban roads.

5.1.1. Iterative Algorithm

For this first example of the iterative algorithm, I have shown how it proceeds from start to finish in great detail. For the sake of brevity, I have done this only with the first intersection.

The initial state of the intersection is shown in Figure 5.1 (a). The data indicates that the true location of the intersection is northwest of the location where it is currently saved. The algorithm begins by map matching the data to the existing map and computing a measure of fit. In this case, this original measure of fit was .77161.

The next step in the algorithm is to move the intersection north a short distance and then map match and compute the measure of fit again. This altered map is shown in Figure 5.1 (b) and its measure of fit was .77748. This measure of fit is higher than the
points. The bins are generally small enough that an average value works well and so there is no need to complicate the algorithm with any kind of regression.

The pseudo-code for this algorithm can be found in Appendix C.
would need to be changed so that it stored the number of shape points the link originally had before they were removed.

The first step in this algorithm is to divide the link into different bins that will be used to group data points. As discussed above, the number of shape points desired is the same as the number that were there before. These shape points will be spread evenly along the middle ninety percent of the link. The first and last five percent of the link are not included in an effort to make the shape points more evenly spread out along the link. Thus, if there were nine shape points, each bin would include ten percent of the link and the first one would span from the five percent mark through the fifteen percent mark.

Before the locations of the shape points can be determined, the data must be map matched to the link and their snap-values must be calculated. The algorithm searches through the data points matched to the link and places each point in its proper bin based on how far along the link it is. Data points for the first and last five percent are discarded.

Finally, before the new shape points can be determined, data points with low snap-values are removed from the data. The reason for this is because during testing, it was found that data that was close to a given road, but not on that road or map matched to any other road, was pulling the shape points away from the actual path of the road. These stray data points present a more significant problem to the shape point algorithm than to the other algorithms because the data set is split up into bins and each point is determined using many fewer data points.

The final step of the algorithm is to take a simple average of the latitudes and longitudes within each bin. The location of the shape points is simply these average
4.6. **Shape Points**

The algorithms described thus far have both concerned themselves with updating the position of an intersection. This algorithm starts with updated intersections and completes the map realignment by adding shape points where necessary.

4.6.1. **Difficulty in Determining When Shape Points Are Necessary**

One difficult hurdle to overcome when attempting to add shape points is to know when they are needed. One solution to this is to simply add shape points where there were shape points before. This follows from the assumption of existing correctness that was discussed earlier in this chapter. Assuming that there was a reason that shape points were added originally, it makes sense that the road is still curved and that the removed shape points should be replaced. Even if this is not the case and the road is straight, adding shape points unnecessarily does not cause any harm. The only possible problem with adding too many shape points is that they take up disk space that may be limited on a personal navigation unit. However, as memory becomes ever cheaper, this should not be a concern in the near future. A related issue is how many shape points a link needs. Using the same logic as above, it follows that a good guess regarding how many shape points are needed is how many were there to begin with.

4.6.2. **The Shape Point Algorithm**

This algorithm begins with a straight link and then adds shape points back. It would most likely be run after the iteration algorithm. If this were the case, that algorithm
Again, this is done for each combination of links. The farthest east and west line intersections are found and then the midpoint of that is saved as the new longitude. Similarly, the farthest north and south are determined and the midpoint is saved as the new latitude. If only two lines were generated because two opposite links were assumed to have the same heading, then the x and y coordinates of the intersection of the lines are saved as the longitude and latitude, respectively, of the intersection of the roads.

An important thing to note is that because of the way that the regression is done, the algorithm returns the negative values for the coordinates. Thus, the coordinates that the algorithm outputs must be multiplied by negative one.

The final step in this algorithm is to perform two verifications that the move makes sense and, at least, did not make the measure of fit worse. These checks are required because of the precision required to make the algorithm work. Small errors in average headings can have large implications on the final output of the algorithm. The first check is to simply make sure that the intersection has not been moved farther than the length of the longest link emanating from the intersection of interest. The second check is to make sure that the new measure of fit is higher than the original one. If either of these conditions is not met, then the intersection should be returned to its original position and should be marked so that a person looks at it later to determine what, if any, adjustments are needed.

The Pseudo-Code for this algorithm can be found in Appendix B.
In this equation, \( m \) is the slope, which has just been calculated, and \( x \) and \( y \) are the longitude and latitude, respectively, of any of the data points. Finding \( b \) is relatively straightforward from here. The algorithm calculates the vertical distance from every data point map matched to the given link to any \( b \) by taking the absolute value of the following expression:

\[
y_i - m * x_i + b
\]

The algorithm then minimizes the sum of the vertical distances by changing \( b \). \( b \) is then set at the value that minimizes the sum of these distances. This is essentially a least absolute deviation regression where the slope has been predetermined. The regression is performed on each link.

The algorithm then takes each line and sets the \( m * x + b \) side of each equation equal to the \( m * x + b \) side of another equation in order to solve for \( x \), the longitude of the intersection of those two lines:

\[
y = m_1 * x + b_1 \quad y = m_2 * x + b_2
\]

\[
m_1 * x + b_1 = m_2 * x + b_2
\]

\[
x * (m_1 - m_2) = b_2 - b_1
\]

\[
x = \frac{b_2 - b_1}{m_1 - m_2}
\]

Solving for the \( y \)-coordinate of the intersection is done by simply placing the above value of \( x \) into one the initial equations:

\[
y = m_1 * x + b_1
\]

\[
x = \frac{b_2 - b_1}{m_1 - m_2}
\]

\[
y = m_1 * \left( \frac{b_2 - b_1}{m_1 - m_2} \right) + b_1
\]
different directions because they have all been converted into a slope that is effectively the same for both directions. Figure 4.3 shows how 315 and 135 have the same slope.

![Diagram showing headings of 315 degrees and 135 degrees have the same slope](image)

**Figure 4.3:** headings of 315 degrees and 135 degrees have the same slope

The final step before taking the average headings is to deal with the problem of taking an average of numbers based on a non-continuous scale. As was discussed in detail in a previous section, this problem is solved by adding 180 degrees to any heading below a given threshold. In the examples in this thesis, fifteen percent was used.

Once the data have been manipulated, the algorithm takes an average of the headings for each link. These average headings, which are given in degrees from North, are then converted into a slope. The conversion is performed using the tangent trigonometric identity. In order for this function to have the desired properties, the opposite value must be used and a phase shift of 90 degrees must be added to deal with the fact that a zero degree heading must correspond to a slope equal to infinity. The equation is given here:

\[ \text{slope} = - \tan(\text{average}\_\text{heading} + 90) \]

The next step in the algorithm is to take the calculated slopes and finish determining their lines. This is done by using the standard slope/intercept equation for a line:

\[ y = mx + b \]
lines should generally be good enough for the accuracy that GPS data allows. A slightly more accurate way to deal with this is to solve for the intersection of each line with all of the other lines that have been determined. The farthest East and West points of intersection are found and the longitude of the intersection of the roads is set to the midpoint. Likewise, the farthest North and South points are found and the latitude is set to the midpoint.

4.5.4. The Intersecting Lines Algorithm

This algorithm starts by map matching the data points to the existing map database. Once this is done, there is some necessary data manipulation. The first step is to remove any data points that are not within a given portion of a given link. The examples in this thesis used only data points that were map matched between fifteen percent and fifty percent of the way from the intersection of interest to the other end of the link. The points that are between zero and fifteen percent of the link are too close to the intersection and could skew the results due to cars that were making turns onto or off of the link. Any point beyond fifty percent is far enough away that perhaps a slight turn in the road could affect the average heading.

The part of the heading that is important for this algorithm is the slope of the line. Thus, to simplify the computation, the following formula is applied to each heading.

\[ \text{heading} \mod(180) \]

In practice, this has the same result as subtracting 180 degrees from any heading that is over 180 degrees. This does not affect the results because a straight line can be represented as a 180-degree angle. This step eliminates the requirement to deal with
under a given threshold, say fifteen degrees, 360 degrees is added to any heading that is
less than 180 degrees. This creates a continuous and progressive scale that causes the
averages to be computed properly. In the sample given in the preceding paragraph, this
technique yields 358.125 which is the correct heading.

4.5.3. Generality and Non-Convergence of Lines

The final problem to solve was that of non-converging lines. This algorithm
computes a model of the lines that the roads represent and then places the intersection
node where the lines meet.

There are two ways to look with this situation. The first solution is to limit the
algorithm to three-way intersections where two of the emanating links are parallel and
four-way intersections that are composed of two, straight roads crossing. These two
situations will guarantee a single point of intersection and worked quite well in my
testing.

Another solution is more general, but did not work well in my limited testing. In
order to keep the algorithm more general and allow for intersections with different angled
streets and more than four emanating links, the algorithm could treat each emanating link
as its own line. Thus, a four-way intersection would generate four lines rather than two,
as in the preceding situation. The mathematical implication of this is that it transforms a
standard four-way intersection into an intersection of four lines that are certainly not
guaranteed to meet at one point.

There is a simple solution to this problem. While the lines will practically never
intersect precisely, they should be very close and so picking the intersection of any of the
to position to calculate these tangent lines, one would introduce the positional error of
GPS, which is much greater than the error of the headings.

The intersection moving algorithm in this thesis does not take full advantage of
these implied first derivatives. While it is true that calculating a heading first and then
computing a best fit for the road given that heading yields an additional level of stability,
there are greater potential benefits to this idea. Future improvements to these algorithms
to make them more general will undoubtedly need to use headings as they strive to
handle sharp curves near or through the intersection. Thus, looking at this thesis as a first
step toward automated map realignment, it is important to use headings.

4.5.2. Computing the Average on a Non-Continuous Scale

Once the decision had been made to use headings instead of position, a serious
problem presented itself. In most cases, taking an average is simple because the scale is
continuous and progressive. Numbers continue to get higher or lower indefinitely. Taking
an average heading is similar to this in most cases. However, when the headings are close
to North, some headings will be in the high 350’s while some will be between 0 and 10
degrees. This represents a 20 degree section of the compass, but the numbers seem to
indicate something else. To illustrate the problem, take the sample headings 355, 357,
357, 358, 355, 1, 359, and 3. A simple average yields 268.125 as the average heading.
This means that, according to this method of averaging, that these cars that were all
heading almost north average to a westward heading! This is clearly incorrect.

Searching for an elegant and interesting mathematical function to solve this
problem proved fruitless and in the end, a workable solution was found. For any heading
measure of fit decreases. Points inside the red area correspond to measures of fit over
0.82. Points in the orange area have measures of fit between 0.81 and 0.82 while points
inside the yellow area are between 0.80 and 0.81. Finally, points not covered by any of
the above colors correspond to measures of fit worse than 0.80.

The pseudo-code for this algorithm is given in Appendix A.

4.5. A Geometric Approach to Realigning an Intersection

This thesis also proposes a second algorithm, developed with the idea of trying to reduce the amount of computing required. Rather than use a guess and check iterative method as in the prior algorithm, this algorithm is a more intelligent method of updating single points.

4.5.1. Use of Heading Instead of Position

When first looking at the problem of a non-iterative point update, it seems logical to use only the position data. The goal is, after all, to update the location of an intersection so it would make sense to use the position information from the data. Upon further thought, however, using the heading information from the data seems to be a better solution.

The benefit of using headings is due to the fact that a heading is essentially a first order derivative of the position of the data point. That is, it is tangent to the direction of travel. This is a huge benefit when dealing with curves because, while these tangent lines could be computed using positional data, the headings are much more reliable. By using
important to note that the algorithm could potentially move in each direction several times. For example, the optimal location might be found after moving north first, then east, and finally south.

The optimality of the solution that this algorithm yields is limited by the distance that the intersection moves at each iteration. Thus, if the distance chosen to move the intersection at each iteration is ten meters, one should expect that the solution will be within ten meters of the true solution. This algorithm is based on the assumption that, as the map’s intersection moves closer and closer toward the actual intersection, the measure of fit increases. In most cases, this assumption is true as Figure 4.2 demonstrates very clearly.

In the figure, the point (0,0) represents the location of the intersection that yields the highest measure of fit. As the intersection is moved away from the true center the
measure of fit is stored along with the current location of the intersection for later use.
The next step is to move the intersection a preset distance in one of the cardinal
directions, say north. A larger distance will have a faster convergence to a solution while
a smaller distance will allow for greater accuracy. With the intersection in its new
location, the map matching program is run again and another measure of fit is computed.

If this new measure of fit is higher than the saved value, the algorithm replaces
the saved values of fit and location with the ones from the new location. The intersection
is then moved in the same direction, north, again and the process repeats itself until the
new values are not better than the saved ones.

If the new location of the intersection does not yield a better measure of fit, the
algorithm reverts to the location of the highest measure of fit and then tries another
direction. Which direction it chooses depends on which direction it had previously been
moving and whether it had success in that direction.

If the measure of fit improves even once moving in the original direction, north,
then the algorithm tries to move in a direction that is ninety degrees with respect to the
original direction, in this case, east.

If the first attempt in a given direction is a failure, then the algorithm tries to
move the intersection in the opposite direction as the one most recently tried. That would
be south in this example.

The algorithm proceeds in this fashion until it tries all directions and does not find
an improved measure of fit. It then assumes that it has found the location that the data
most strongly supports and saves it into the database. Thus, the algorithm causes the
intersection to take a slow and meandering path toward the optimal measure of fit. It is
database is more right than wrong. By this I simply mean that the roads reflect what
exists in reality with some accuracy. If this were not true, accurate map matching would
be nearly impossible and there would be little point in “improving” intersections that are
not based on reality.

4.4. An Iterative Approach to Realigning an Intersection

The first of the algorithms developed in this thesis is an iterative algorithm. It
does not use any form of intelligence and instead relies on a systematic sequence of guess
and check movements of the intersection in question.

4.4.1. The Iterative Algorithm

The first step of this algorithm is to remove any shape points from the links
emanating from the intersection to be realigned. This is necessary in order for the
intersection to be able to be moved without causing strange kinks in the road. Figure 4.1
demonstrates this problem.

Figure 4.1: Demonstration of why shape points must be removed prior to map
realignment. (a) Before any change. (b) A move of the intersection without
removing shape points causes a strange kink. (c) Removing the shape points
in the bottom, left link takes the kink out.

With the shape points removed, the algorithm map matches the links surrounding
the intersection of interest and then computes an initial measure of fit. This initial
For my standard metric of map accuracy, I have chosen to use an average value. For any intersection, \( p \), there is a set of links, \( k \), that emanate from it. Each link has \( n \) data points map matched to it and each data point has a snap-value, \( s \). The standard metric that I have devised is the average, across all emanating links, the average snap value for each link.

\[
M = \frac{1}{k} \sum_{k \text{ in } p} \left[ \frac{1}{n} \sum_{i=1}^{n} s_i \right]
\]

The advantage of using an average is that it treats each link equally. Some links have much more data than others. If one were to use a sum, the accuracy of roads with less data would be sacrificed because there would be such emphasis on the fit of links with substantial data. An average prevents this because it takes the amount of data per link out of the equation.

On occasion, a link will not have any data map matched to it. In this case, there is no average snap-value available to use in the computation of the measure of fit described above. In this situation, a low number can be substituted into the computation in place of the non-existent average. In the examples contained in this thesis, the number 0.7 was used. It is important that the fact that no data is map matched to a link be expressed because, otherwise, the possibility exists for the value of the measure of fit for the intersection to improve while one link gets so much worse that data is no longer map matched to it.

4.3. Assumption of Existing Correctness

Before writing about the algorithms in detail, it is important to note a key assumption that I made while developing them. This assumption is that the existing map
maps were aligned perfectly and the curves were stored as curves rather than line segments, there is still error in the GPS signal.

Map matching is a complicated process that is done by using the position and heading of the data. In the demonstrations of the algorithms in this thesis, a program called SnapStats was used. This program was provided to me by ALK Technologies. In addition to determining which link a data point belongs to, the program also provides information about which direction on the link the vehicle was traveling and how far along the link the vehicle was when the data point was recorded.

4.2. **Standard Metric for Map Accuracy**

In evaluating map alignment, I have used a standard metric to measure map accuracy. A standard metric is useful for two reasons. First, it provides numerical support to what seems obvious by visual inspection. Second, one of the algorithms that I devised is iterative and requires a measure of accuracy after each iteration both to determine the next step and also to terminate the algorithm when it is done.

The map matching program that I used, SnapStats, also includes a measure of fit that is called a snap-value. For each data point, it determines a measure of fit based on the agreement of location and heading between the given data point and the link in the network. Snap-values range in value from zero to 0.93. A snap-value of zero corresponds to a data point that is not near any link in the existing network. For a local street or secondary road, the highest snap-value is 0.84 and for any other type of road, the highest value is 0.93.
4. Discussion of Automated Map Realignment and Potential Algorithms

This chapter discusses the process of map realignment using GPS data. The primary focus of this thesis is to accurately realign a map, one intersection at a time, and then add shape-points if necessary. This process could ultimately be part of a broader program designed to automatically update digital maps using GPS data. There were many challenges in determining what follows. Those problems, along with the solutions that I found, are discussed below. Demonstrations of the algorithms and discussion of their effectiveness are saved for chapter five.

4.1. Map Matching

Before any sort of map realignment can be done, it is necessary to perform a step called map matching. This is the process of assigning data points to links on a given network. This step is important because it would be almost impossible to update the map database if one did not have at least an estimate for which data points were associated with which streets.

In a perfect world, this step would be trivial because the data points would lie directly on the network links. In reality, however, it is very rare for a data point to lie directly on a link in the network. There are three reasons for this. First, the maps are not perfectly aligned. Second, even if the maps were prefect, the database consists exclusively of straight line segments. The database approximates turns using shape points, but any turn is still a straight-line approximation of a curve. Finally, even if the
DGPS is a relatively simple idea. It works by using ground stations at known positions. These ground stations constantly measure their location and then compare the location from GPS with the known location. The ground station then broadcasts signal correctors that are received by relatively nearby DGPS enabled receivers that incorporate this shift into their calculations. The closer the user is to the ground station, the better their DGPS accuracy will be. As one moves farther away from the ground station, the atmospheric and orbital corrections become less relevant. The CoPilot receivers that were used to collect the data used in this thesis are not DGPS enabled.
The potential problem with this is that the resolution of GPS under the old system was worse than the resolution of the road network. This could cause serious problems with a map update that is done using this data. The ideal thing to do would be to remove the old data from the data set. Unfortunately, this is not possible as the data, in the form that I have it in, does not include the collection date with the rest of the data point’s attributes.

Fortunately, this problem is alleviated in two ways. First, SA altered the signal in a way that was decodable, but was always different within preset tolerances. This allowed the military and other select users to be able to decode it, but not others. The significance of this is that the error that it caused averages to zero over a sufficiently long time. Thus, because the data was collected over years, it will not cause a noticeable shift in the data. Secondly, as time goes on, more new data will be added on top of the old and inaccurate data. As more data is accumulated, each individual point matters less and thus the new, more accurate points will become dominant if they have not already done so.

3.7. Differential GPS

Differential GPS (DGPS) is a technology that improves the accuracy of standard GPS significantly. Horizontal accuracy for DGPS can be as good as one to three meters. The technology takes advantage of the fact that many of the errors associated with standard GPS can be corrected for on a local basis. These include atmospheric effects as well as clock and satellite orbit errors.

objects and not just one. Thus, because we expect the positional accuracy of the system to be within thirteen meters on average, we can expect to be able to absolutely distinguish between roads that are twenty-six meters or more apart. Anytime two roads are closer than this distance it may not be possible to use current GPS technology for realignment.

3.6. Selective Availability

Another source of error in the GPS data set used in this thesis arises from Selective Availability (SA), which was the intentional degradation of the GPS signal in the interest of National Security. It was removed by a Presidential decree in May of 2000. With the removal of SA, GPS became up to ten times more accurate. The effect that this has on map realignment is that some of the data is from the period before SA was eliminated and the data from that period is much less accurate.

3.5. Error in GPS signals

In addition to the errors that can result from the collection method, there is error in the GPS data that are inherent to any GPS usage. There are several performance standards set by the Department of Defense. They site both an average positional accuracy and a “worst site” positional accuracy. The worldwide horizontal average positional accuracy is given as less than thirteen meters within a ninety-five percent confidence interval. The worst site positional accuracy, which is a worst-case scenario, is given as thirty-six meters within a ninety-five percent confidence interval.34

The most significant source of error is ionospheric error. This error is caused by the ionosphere, which is a layer of charged particles in the atmosphere that slow incoming signals as they pass through. Depending on the conditions of the ionosphere and the angle at which the signal passes through it, this can result in inconsistent timing delays that lead to inaccuracies. Figure 3.3 shows a scatter plot of the sensed location relative to the actual location, which is at (0,0). Other significant sources for error are satellite clock error, orbit error, troposphere error, multipath error and receiver errors.35

These are all capable of generating errors of a meter or more.

These errors affect the resolution of the GPS data. Resolution is an important idea to understand when using sensed data such as is the case here. The resolution of an object corresponds to how far apart two things need to be in order to distinguish them as two

These particulars of the data can cause a problem when one tries to use this data to update maps automatically. While the human eye can see the difference between these details and the roads, a machine has trouble. In areas where a lot of data has been collected, by many users, these problems are eased. By having many different drivers, it is unlikely that a single user’s driveway will sway the data enough to make a significant difference. If, however, all the data for a given area comes from only one driver, then the driveway is most likely used each time the user drives down that street and it is very likely to skew the average position and heading.

Likewise, if there is enough data from many drivers, it is unlikely that a parking lot will cause a large skew from the main road. If there were only one driver, there is the potential that that particular user parks in that lot for work and the data will be skewed.
on the plane. This makes the updating problem relatively simple, in some respects, because it is just a complicated regression.

3.4. Data Used in This Thesis

The data that I will be using has been collected over many years using ALK Technologies CoPilot navigation systems. CoPilot is a personal navigation device that uses a GPS receiver connected to a Pocket PC powered handheld computer. The system provides route guidance to the driver and can also be set to record data about the car’s movements. As the user drives, the CoPilot software saves the machine’s latitude, longitude, heading, speed, and also a time stamp and user ID. Currently, the user must manually submit the data.

This method of collection has both advantages and disadvantages. The largest advantage is that the data is inexpensive to collect. The users of the navigation systems would be driving regardless and it doesn’t cost them anything to record the data. The only inconvenience is that, as stated before, they have to submit the data manually. Regardless of this inconvenience, there is still a lot of data that has been collected and it is already of use.

One problem with this method of collection is that users do not pay attention to when the device is recording data. The device records all the time, including when the user is in a parking lot or their driveway, or some other place that is close to, but not, a road. Figure 3.2 shows a sample of these problems that have been found in the actual data.
3.3. **Usefulness of GPS Data vs. Other Data Sources**

The main advantage of GPS over other data sources is that it is sensed data. Road networks change a lot as new roads are built and old roads are rebuilt and changed. An update every five or ten years, which is what is possible with digital orthophotos, is not often enough to be truly useful in areas of rapid development. The technology exists for cars to transmit data regarding changes in the road network wirelessly so that changes might eventually be made almost instantaneously. This is simply not possible with a photographic method of network updating.

An additional benefit to GPS is that there is no need for complex image processing. If map realignment is done manually, digital orthophotos are a good option because it is easy to overlay the images and move the digital map so that it accurately depicts the photo underneath. This is much more complicated to do automatically. GPS data, on the other hand, is recorded as points on a plane and the roads are stored as lines.

---

speed of light and are received by a GPS device on the ground, at sea, in the air, or even in space. The device estimates its distance from the satellite based on the information in the signal. 31

Interestingly, the satellites are not in a geosynchronous orbit. A geosynchronous orbit means that the satellite makes one revolution around Earth every twenty-four hours which can give it the property of being over the same place on Earth at all times. The GPS satellites, however, make two revolutions per twenty-four hours. This complicates the system to some extent because the satellites must also broadcast their location as their position relative to Earth is constantly changing. 32

To get a good “fix” on position, the GPS device requires a clear view of at least four GPS satellites. Once the device calculates its distance to a satellite, it generates a sphere around that satellite with a radius equal to its distance from the given satellite. This is done to all of the satellites that are in view of the device. The intersection of these four spheres is the location of the device. Four satellites are necessary because two intersecting spheres create a circle, the third sphere intersecting with the circle yields two points in space and the final sphere intersects with one of those points and that is the location of the device. Figure 3.1 illustrates this.

31 Ibid.
32 Ibid.
3.1.2. NAVSAT Satellite Positioning System

NAVSAT is the term used to describe the United States Navy’s NAVigation SATellite System. The Navy developed this system to improve positional accuracy for its vessels. The NAVSAT system was accurate to 200 meters and consisted of six satellites. The satellites broadcast a signal that the user would receive and then measure a Doppler shift in the frequency of the transmission. This information was used to determine the vessel’s position. The major flaw of the system was that the positional information was not always available. Because there were only six satellites, information was only available every ninety minutes. Better accuracy was also desired. The system was shut down in 1996.\(^\text{28}\)

3.2. Overview of GPS and How It Works

The Global Positioning System was designed to use new technology to improve on the NAVSAT system’s performance. Specifically, GPS provides constant positional accuracy to a very high level. The first GPS satellite was launched in 1978 with a second round of satellites launched starting in 1989.\(^\text{29}\) There are currently at least twenty-four operational satellites at any given time.\(^\text{30}\)

GPS is made possible because of the super accurate timing that atomic clocks provide. These hyper-accurate clocks make it possible for each satellite to broadcast a signal at precisely the same time. These signals, which include both the time that they were sent and the location of the satellite at the time, then travel from the satellite at the

\(^{28}\) Ibid.


3. Explanation of Global Positioning System and Data Set

This chapter discusses the GPS data to which I have access, including the advantages of GPS data and inherent inaccuracies in the data. The data is a useful starting point to understanding the approach that this thesis proposes for updating map databases. I have chosen to work with GPS data because I feel that it will prove to be easier, as well as more useful, than alternative technologies such as digital orthophotos.

3.1 Before GPS

GPS is the current electronic positional finding system, but it is not the first. As early as the 1930’s, there were systems that used radio beacons instead of the stars for navigation. The benefits of electronic navigation over other forms is that bad weather does not stop radio waves and thus navigation can be done at any time.

3.1.1. LORAN

LORAN, which stands for Long Range Aid to Navigation, was a system of land-based transmitters. It is the best known of the radio navigation systems developed during World War II. A user would determine their position by timing the transmissions received from different LORAN stations. The positional accuracy of this system was several miles. While this seems a laughable amount of error by today’s standards, it was a large step forward for electronic navigation and an important steppingstone.27 In fact, a version of this system is still operational today.

thorough coverage. Digital orthophotos also promise to stay relatively up to date. The US Geological Survey has set a goal of updating all digital orthophotos at least every ten years and twice as often in areas with more rapidly changing land use.\footnote{26}

2.3. Summary

The partial history of maps that this chapter offers serves to demonstrate the continuity of improvements made to maps and mapmaking, a tradition upon which this thesis attempts to build. Understanding the predicaments that these map makers faced helps us to understand where mankind is with respect to map making and where we can go in the future.

\footnote{25 The Positional Accuracy of MAF/TIGER, US Department of Commerce, Bureau of the Census, March 2000.}
2.2.2. Uses of Digital Orthophotos

According to the US Geological Survey, digital orthophotos can be put to any number of uses. They suggest that they can be used for, “vegetation and timber management, routing and habitat analysis, environmental impact assessments, emergency planning, flood analysis, soil erosion assessment, facility management, and groundwater and watershed analysis.” The same source also suggests that they could be used for real-time data updates.

Digital orthophotos are currently being used to do projects similar to the analysis in this thesis. The US Bureau of Census is currently using digital orthophotos to update their TIGER database. They selected digital orthophotos because of their availability and

---

2.2. Previous Work In Map Realignment

The rest of this chapter concerns itself with previous and current work being done on the subject of map realignment. All of the relevant work that I found while researching this thesis used a special type of photography that is described in detail below.

2.2.1. Description of Digital Orthophotos

Digital orthophotos are photographs that have been modified to exhibit some of the qualities of a map. They look like aerial photographs but they have had all distortions removed. Thus, they are properly scaled like maps yet they have the visual quality and detail of a photograph. They are useful because distance can be measured on them just like a map.

Creating a digital orthophoto requires four primary inputs. The first input is the standard photograph which is usually taken from either an airplane or a helicopter. This can either be scanned or taken directly from a digital sensor. The second element is a digital model of the elevation of the area for the digital orthophoto. Thirdly, ground reference points are required. These are identifiable in the photograph and have known coordinates. There must be at least three of these reference points to make the digital orthophoto. The last significant input is information about the camera used to take the photos. Figure 2.6 shows two views of a digital orthophoto.\(^\text{22}\)

they have to decide how much detail to show in the map. This is a personal choice that depends on how much area the map is to cover and also how clear the map is to be. A map covering less area with great detail is useful for some things while it is useless for others.

2.1.7. Digital Maps

Current digital maps are primarily based on the United States Geological Survey’s (USGS) TIGER database. TIGER is an acronym for Topologically Integrated Geographic Encoding and Referencing system. This database was created for the 1980 census of the United States and was a joint effort by the USGS and the Bureau of the Census.19

The TIGER database used as its foundation another database known as the GBF/DIME files. GBF/DIME stands for “Geographic Base File” using “Dual Independent Map Encoding.” The US Bureau of the Census created this database in 1967 in preparation for the 1970 census. It covered only two-percent of the geographical area of the United States but these were the most densely populated urban areas. These GFB/DIME files were based on the Census Bureau’s Metropolitan Map Series.20

The rest of the TIGER database was created later using topographical maps made by the US Geological Survey. The maps used were of the scale 1:100,000. These maps were scanned and the roads, railroads, water features, and other transportation features were stored in the database. The database does not include topography.21

21 Ibid.
Chicago Times.” The map showed the route of the first organized auto race in the country. It was not long before road maps that were designed specifically for the driving public began appearing. Rand McNally & Company produced their first road map for automobiles in 1904.17

These first road maps had many problems. Most roads were not formally named and so the map consisted of many unlabeled lines. Drivers could not tell when they were supposed to turn unless they were given landmarks to look for. Thus, the first effective navigation method for automobiles were books, not flat maps. Motorists would have books that had directions from place to place using turn-by-turn directions by indicating the location of local landmarks. A modern motorist will recognize this type of navigation as the common way that individuals direct others to their home or business. A system like this works as long as the user wants to travel along a route that has been previously generated and assuming that none of the landmarks have changed. There were certainly many times when a landmark was painted, torn down, or rebuilt and the motorist simply drove past it and became lost. Clearly there was a commercial need for a more robust and reliable method of navigation to help the growing number of motorists. Various levels of government recognized that streets needed to be permanently named so that maps could be developed to assist travelers.

Once roads were given standard names and markers, there was still the problem of compiling the information. This daunting task was done, and still is done, by the United States Geological Survey.18 The database that they maintain is the starting point for almost all road maps. Once the mapmakers have the data from the US Geological Survey,

17 Douglas A. Yorke, Jr. and John Margolies, Hitting the Road: The Art of the American Road Map, pp. 15-16.
18 Ibid., pp. 21.
shows a compromise projection (specifically the Robinson projection) and its effect on a given image.

Figure 2.4: The distortions caused by a compromise projection. Note that it displays some of the qualities of both equal area and Mercator projections.\textsuperscript{15}

The compromise projection is useful precisely because it is a compromise. John Garver, of National Geographic discusses this when he speaks of his organizations choice of a compromise map (again, the Robinson projection). “The projection does not espouse any special point of view, and we believe that its compromises are the most reasonable for a general reference map of the world.”\textsuperscript{16}

2.1.6. United States Road Maps

Road maps appeared in the United States soon after the emergence of the automobile. The first road map in the United States was produced in 1895 by “The


\textsuperscript{16} As quoted in: Denis Wood, The Power of Maps, pp. 58.
ancient Greeks but only sporadically and carelessly used during the intervening centuries.”

For all its positive changes, Mercator’s projection was not an instant success. It was a complex concept and it took many years for it to be adopted by the sailing community. However, it was a brilliant breakthrough with profound implications for navigation. It is still used today for precisely the same reasons that Mercator conceived the approach over 400 years ago.

The projection has also met much criticism. Many people have asserted that the reason for the projection’s popularity is because it makes Europe larger and, thus, more powerful looking. David Turnbul, an author, is quoted as saying, “Is it a coincidence that a map which preserves compass direction (a boon for navigation) shows Britain and Europe (the major sea-going and colonizing powers of the past 400 years) as relatively large with respect to most of the colonized nations?” The answer is, of course, that it is a coincidence because the projection was made to fulfill navigational needs and the size of Europe on the map was a creation of mathematical necessity.

2.1.5. Compromise Projection

A compromise projection is any map projection that is a combination between a Mercator and an equal area projection. A compromise projection can be made almost like a Mercator or almost like an equal area projection or anywhere in between. Essentially, if one had a rubber equal area projection, they could stretch it until it became a Mercator and any projection in between would be called a compromise projection. Figure 2.4

---
Another consequence of this projection is that there is no scale for the map because it is constantly changing. The problem is negligible at the Equator and grows much larger the farther the user moved toward the Poles. This can be seen in Figure 2.3. The lack of scale can be dealt with by using tables that can estimate the change in the scale at different places on the map.

Figure 2.3: The distortions caused by a Mercator projection.¹³

A final consequence is that the North and South poles cannot be shown on this projection because the longitude lines never come back together and thus the point that is the North pole is stretched infinitely which is impossible to show. Also of note is “the reintroduction of systematic latitude and longitude lines, which were conceived by the

people an idea of what the known world looked like while Mercator was trying to find a way to help merchants navigate the known world and to assist explorers in sailing in uncharted areas to make new discoveries. According to John Wilford, Mercator appreciated, “more than his predecessors or most of his contemporaries, that maps were made not only to record discoveries but to be used in commerce with the new lands and for making other discoveries.”

Thus, in 1569, he published a map whose legend contained the inscription, “New and Improved Description of the Lands of the World, amended and intended for the Use of Navigators.”

At the time of Mercator’s work, navigating a ship was extremely difficult. Most maps still had a projection that looked something like an equal area projection. The problem with this form of projection is that a straight line drawn on such a map does not represent a straight course of travel. Mercator wanted to create a map where a straight line drawn on the map would correspond to straight travel on a constant compass bearing. While he could not prove it mathematically, he felt that his new projection would do what he hoped and he would eventually be proven correct. In order to have this property, angles on the projection must remain true. That is to say, a vertical line always goes North and South while a horizontal line always goes East and West. This causes a “spreading” of land that is sometimes referred to as the Greenland effect. This is because, in reality, the lines of longitude are far apart at the Equator and eventually merge together at the poles while in the projection the lines of longitude to not move closer together. The spread is done at the same rate in both latitudinal and longitudinal directions in order to ensure the strait line principle that was paramount to the projection.

---

11 John N. Wilford, The Mapmakers, pp. 73.
12 Ibid., pp. 73-75.
equal area property. However, this type of projection always has the side effect of distorting shapes of places. In attempting an equal area projection, Ptolemy had an impossible task in his day and age. At the time he was working, there had not been enough observation and surveying to estimate landmasses with accuracy and the relationships of landmasses in his maps were distorted both by the projection and by observational error. Figure 2.2 shows the distortion of images that an equal area projection causes.

Figure 2.2: The distortions caused by an equal area projection

2.1.4. Mercator Projection

Gerardus Mercator was a mapmaker who worked in the mid-sixteenth century. Like Ptolemy, Mercator also faced the problem of representing a sphere on a flat surface. However, Mercator’s goals were different than Ptolemy. Ptolemy was trying to give

---

extended farther East than it actually does. These two factors conspired to make Ptolemy’s map fairly inaccurate.  

Despite his inaccuracies and misconceptions, Ptolemy’s work is still evident in almost every map we see today. It seems that Ptolemy was the one who started the tradition of having North at the top of the map with East to the right. He did this because most of the known world was in the northern hemisphere and he thought it would be easier to study if the familiar European and Middle Eastern regions were on the upper portion of the map.

### 2.1.3. Equal Area Projection

In his mapmaking, Ptolemy was attempting to use something called an equal area projection. An equal area projection is one in which the relative sizes of objects represented remain equal. This avoids the stretching or enlarging of places far away from the equator. These maps are good for population density types of maps because of their

---

8 Ibid., pp. 25-26.
great significance lies in the fact that they are in part surprisingly accurate and detailed and show that the art of cartography was well advanced at this time.”

Clearly, there were many types of maps developed between these ancient examples and today. There are in fact too many to discuss them all in detail. Consequently, this thesis will focus on the most significant historical mapmakers and ideas.

2.1.2. Ptolemy

Ptolemy was a scholar in Alexandria, Egypt, who worked between 127 and 151 C.E. He was a scholar of many things, including astronomy, astrology, music, and optics. His most significant and lasting contributions were in geography and cartography. While many of his ideas were not new, he clarified the work of other cartographers and further developed their ideas.6

One of Ptolemy’s greatest and most lasting achievements was his map of the world. This map is shown in Figure 2.1. Ptolemy knew that the world was round and that he would have to deal with this issue by distorting the map in some way to display the round Earth on his flat paper. He also thought that scale was very important for a map. As a result, his map attempted to maintain spatial accuracy even though it created angles that would make the map difficult to use for navigation.7

Ptolemy’s map, for all its worth, was not without flaw. He thought that the Earth was about three-quarters of the size that we now know it to be. He also thought that Asia

---

7 John N. Wilford, The Mapmakers, pp. 28.
As this thesis deals primarily with road networks this section will also investigate how the original road maps were made and how they eventually became the electronic map database that we have today.

2.1.1. Ancient Maps

Humans have been representing physical relationships between places since the beginning of time. These early maps were used both to identify and locate property relationships, as well as to mark routes between places. Early Europeans drew rough maps on the walls of their caves. As Europe set out to colonize the world, they found that in many of the places to which they traveled, the locals were quite familiar and proficient with maps. Pacific Islanders used sticks held together with fibers to represent prevailing winds with rocks inserted to represent land. The Eskimos carved maps of the coast in ivory. Native Mexicans made maps that indicated roads as lines of footprints. Before that, the Incas, in Mexico, made complicated maps of stone and clay. Clearly, the idea of representing the notion of “here” and “there” are ancient. ³

The earliest known map originated around 2300 B.C.E. It was found at Nuzi, which is in northern Iraq. The map shows the local geography including streams and hills as well as local settlements and three of the cardinal points. Other maps have been found in Mesopotamia and Egypt from about the same time. These maps appear to be real-estate maps used for the purpose of taxation.⁴

Ancient maps have also been found in China. While they are not as old as the one found at Nuzi, they are more sophisticated. Anneliese Bulling says of these maps, “Their

---

³ John N. Wilford, The Mapmakers, pp. 8-10.
⁴ Ibid., pp. 7-8.
2. Literature Review

This chapter discusses in detail the relevant previous work regarding map realignment. Human kind has been making maps since the beginning of time to mark their relative position in their world, to aid in navigation, and to push out the boundaries of the known world. While these basic purposes for map making have not changed, the methodology has evolved to keep pace with changing technology and tools for estimation. A review of the history of how historical maps were made and their purposes is useful in assessing the current and next generation of map creation. It is also important to look at what others are currently doing in their efforts to bring about the next generation of maps.

2.1. History of Maps and Map Databases

Much of the early history of maps revolved around the struggle to improve basic accuracy. There were no tools to measure large objects like coastlines and mountain ranges, and many of these large landmasses were not well enough known by mapmakers to depict even by estimation. Over time, these inaccuracies were addressed by further exploration and better surveying.

The next phase of map making dealt with the problem of representing a round object like the Earth on a flat surface. A globe would be ideal, but a globe large enough to show any detail would be much too large to take onboard a ship. Mapmakers have never found a single projection useful for all purposes and in fact one does not exist.
Chapter five demonstrates the algorithms discussed in chapter four. Additionally, it contains a discussion regarding the successes and failures of each algorithm and hypothesizes why these results are what we might expect.

Finally, chapter six offers a conclusion to the work that I have done in this thesis and also proposes some topics for further research.
Despite these difficulties, this seems to be the method that has gained the most support. The primary reason for this is because taking digital orthophotos is relatively easy compared to collecting GPS data. All that is necessary is to fly a plane over head, mark several landmarks with GPS, and take out any distortions. Compare this to having many people driving with receivers for long enough to get a large enough sample and it is easy to understand why this method has gained initial strength. Conversely, however, to get an update on a constantly changing infrastructure requires frequent flights to gather images which is expensive.

1.4. Thesis Contents

This thesis attempts to develop a foundation for map realignment based on GPS data. Most importantly, it presents several algorithms for updating these map databases and discusses their successes and failures. The emphasis is on the development of theoretical algorithms to perform these tasks rather than creating computer code that accomplishes the task.

Chapter two is a review of the history of maps, mapmaking, and of previous work that has been done on the topic of map realignment. It ends with an in-depth look at digital orthophotos, as that seems to be where most work is currently focused.

Chapter three is a discussion of the data that I used for this thesis. It includes the collection process, explanation of certain irregularities caused by this method of collection, as well as significant potential errors in the data.

Chapter four discusses the algorithms that I developed in detail. It includes the problems faced and offers solutions to these problems.
1.3. Possible Methods

There are two primary methods that are being considered to automate map realignment. One method is by using GPS and the other is through a precise form of aerial photography known as a Digital Orthophoto. Each method has both strong and weak points.

1.3.1. GPS

This is the method that I have chosen to focus on in this thesis. Using GPS tracks gathered by cars driving allows us to see where the roads really are relative to the GPS coordinate system and within the expected tolerances of GPS. I will assume that the mean track is where the road actually is. I will then take this average track as the actual location of the road and attempt to automatically move the intersections and shape points on the digital maps to reflect the recorded GPS signals.

1.3.2. Digital Orthophotos

A Digital Orthophoto looks very similar to an aerial photograph. The difference is that an orthophoto has had any distortions that are commonly found in aerial photographs removed. Thus, a user can use the image just like a map because distances are scaled correctly. This digital orthophotos are more accurate than a traditional paper map because it has absolute locations. The problem with using these digital orthophotos for map realignment is that it must either be done manually or by using complex digital image processing if it is to be automated. Neither of these options seem as elegant as using the GPS data.
it faster and safer. Traffic congestion would be eased as well. The algorithms and methods explored in this thesis could potentially be applied to this area as well. If cars were to drive themselves using map databases, the map databases would have to be very accurate. Errors in maps would have consequences beyond the annoyances that they are now. Rather, any error could cause major problems including accidents and death. Thus, hyper-accurate maps would be necessary. These maps would have to know each lane and would have to be generated from sensed data rather than programmed data.

There are also, undoubtedly, future uses that nobody can currently foresee as the future demands of users are all but impossible to predict. The idea of “if you build it, they will come,” is very relevant. By increasing the accuracy of maps and the sensitivity of the machine’s ability to know where it is, it is probable that new technologies will be developed that nobody has yet imagined. They will develop because the technology is good enough to allow it. Erik Anderson works for Nokia and in a lecture given at Princeton University, he spoke about the success of Short Message Service (SMS) in Europe. When GSM, the European cellular standard, was developed, they added text capabilities because they had the ability to do it. They had no idea how or if it would be used and many of the developers thought it would never be used. However, in recent years, this service has become very popular and profitable in Europe and Asia. Mr. Anderson emphasized that this has been the case for many technological developments. There is no reason to assume that navigational technologies would be any different.²

---

degree of accuracy to the user. We have already seen that in fact the maps are not that accurate. This is not a significant problem with a paper map because with a medium scale map (i.e. a town fit on one map) because a paper map does not have resolution past its actual accuracy. As stated above, there are always systemic errors with regard to position, but by improving map accuracy, at least the maps would be as accurate as they imply.

1.2. Future Demands of Digital Map Users

It is impossible to tell for sure what the future demands will be. There are, however, several foreseeable possibilities. A major need for digital maps will continue to be navigation.

There are other users of Digital maps who can benefit even though they do not use GPS devices. Users of MapQuest, yahoo maps, or other web-based services also have much to gain. In an interview with the New York Times, Nicole Wolf spoke of a time when she was late arriving to a meeting because the directions and maps that had been given to her were inaccurate. She says, “The highway part is easy, but once you get into Boston things get a little hairy…I got to a point where what I was encountering on the street wasn’t at all what MapQuest made me believe was there.” Updated maps will more clearly reflect the actual road network and prevent problems of the maps not matching up with what is seen on the roads.¹

Another possibility is that future technology may allow a machine to know where it is located with much increased accuracy. Greater accuracy could allow cars to navigate themselves on major highways. This would take the human error out of driving and make

the GPS coordinate system, that is, there is no systemic error in the GPS here. How does the machine know which road it is on? If the final destination is to get to the red “X”, and the machine thinks that it is on Firestone Street, it will instruct the user to make a U-turn when if the user is actually on Williams Street the destination will be directly ahead.

![Diagram of street layout](image)

**Figure 1.2:** A common problem with inaccurate maps.

If maps were perfectly aligned with the GPS grid system, many of these errors could be avoided—but not all. There is still error in the GPS system which, given current technology, will be there regardless of how accurate the maps themselves are. The above situation, however, can occur even when the GPS device locates itself correctly. With perfectly aligned maps this problem could be minimized which would, at the very least, reduce the amount of error that users experienced.

### 1.1.2. Perceived Accuracy

Another significant problem with current maps is that of perceived accuracy. An electronic map has a perceived accuracy much higher than its actual accuracy. These maps can theoretically pinpoint things down to the meter—and frequently represent that
1.1. Limitations of Current Maps

Current maps are good at representing locations relative to the things around them. For a standard paper map, this is what is important. A typical user, for example, does not geometrically survey their surroundings to determine their precise coordinates and then locate those coordinates on their paper map to establish their position. Rather, they look for nearby landmarks to determine their current position on their map and look for other landmarks to guide them. A machine, however, must use the former method since it has no way of “seeing” landmarks and it only understands numbers. Thus, relative accuracy with its surroundings is not enough. The map must be accurate relative to a coordinate system, like GPS, that the machine understands.

1.1.1. Misidentification of Streets

With some regularity, a GPS navigation system will misidentify the street that it is on. This most often happens when two roads are relatively close together and the machine reads its position as being somewhere in between the two roads. The machine will sometimes indicate that the user is on the wrong street. This can cause great confusion to the user who is most likely in an unfamiliar area. The directions given to the user can be erroneous because the two roads possibly do not end in the same place or have all the same cross streets. As the user continues to drive in a somewhat lost state, the machine will eventually correct itself and the user will get to where they are going. This is, however, undesirable and unnecessary.

Figure 1.2 shows an example of this problem. The green arrows indicate where the machine thinks it is. Let us assume that the machine’s position is accurate relative to
1. Introduction

Map Realignment is the concept of making a map correct relative to a given coordinate system. This thesis concerns itself specifically with the Global Positioning System (GPS). The use of this satellite-based system of position finding has grown tremendously over the past decade and its future growth is predicted to be strong as well. Much of the increased use has been for in-car navigation systems. These systems have digital maps stored onboard that are used as a reference to the car’s current location and also to give directions to the user’s destination. The maps that these systems use are based on preexisting paper maps that have been digitized. They represent a sound starting point from which to improve. Figure 1.1 shows the current state of these first generation maps (a) and what they would ideally look like after map realignment (b). In this case, the realignment was done manually. This thesis is a first attempt at making this change automatically.

![Figure 1.1: Example of map realignment. The dots with lines coming off of them are data points recorded by a vehicle driving on the roads. (a) The data points displayed over the existing roads. (b) The data points displayed over manually updated roads.](image-url)
Appendix C: Pseudo-Code for Shape Point Algorithm

Appendix D: Iterations and Measure of Fit for Iterative Algorithm Examples

Appendix E: Calculated Values for Intersecting Lines Algorithm Examples

Appendix F: Calculated Values for Shape Point Algorithm Examples
4.6.1. Difficulty in Determining When Shape Points Are Necessary 46

4.6.2. The Shape Point Algorithm 46

5. **Implementation of Map Realignment Algorithms** 49

5.1. Monterey Boulevard and San Benito Way, San Francisco, CA 49
   5.1.1. Iterative Algorithm 49
   5.1.2. Intersecting Lines Algorithm 51

5.2. Claremont Boulevard and Portola Drive, San Francisco, CA 53
   5.2.1. Iterative Algorithm 53
   5.2.2. Intersecting Lines Algorithm 54

5.3. Wrightstown Road and Cooper Road, Washington’s Crossing, PA 54
   5.3.1. Iterative Algorithm 55
   5.3.2. Intersecting Lines Algorithm 56
   5.3.3. Shape Point Algorithm 56

5.4. Discussion of the Performance of the Algorithms 57
   5.4.1. Iterative Algorithm 57
   5.4.2. Intersecting Lines Algorithm 58
   5.4.3. Shape Point Algorithm 59

6. **Conclusion** 60
   6.1. Interpretation of Results 60
   6.2. Areas for Further Study 61

**Bibliography** 63

**Appendix A: Pseudo-Code for Iterative Algorithm** 66

**Appendix B: Pseudo-Code for Intersecting Lines Algorithm** 69
3. **Explanation of Global Positioning System and Data Set**

3.1. Before GPS

3.1.1. LORAN

3.1.2. NAVSAT Satellite Positioning System

3.2. Overview of GPS and How It Works

3.3. Usefulness of GPS vs. Other Data Sources

3.4. Data Used in This Thesis

3.5. Error in GPS Signals

3.6. Selective Availability

3.7. Differential GPS

4. **Discussion of Automated Map Realignment and Potential Algorithms**

4.1. Map Matching

4.2. Standard Metric for Map Accuracy

4.3. Assumption of Existing Correctness

4.4. An Iterative Approach to Realigning an Intersection

4.4.1. The Iterative Algorithm

4.5. A Geometric Approach to Realigning an Intersection

4.5.1. Use of Heading Instead of Position

4.5.2. Computing the Average on a Non-Continuous Scale

4.5.3. Generality and Non-Convergence of Lines

4.5.4. The Intersecting Lines Algorithm

4.6. Shape Points
Contents

1. Introduction 1
   1.1. Limitations of Current Maps 2
       1.1.1. Misidentification of Streets 2
       1.1.2. Perceived Accuracy 3
   1.2. Future Demands of Digital Map Users 4
   1.3. Possible Methods 6
       1.3.1. GPS 6
       1.3.2. Digital Orthophotos 6
   1.4. Thesis Contents 7

2. Literature Review 9
   2.1. History of Maps and Map Databases 9
       2.1.1. Ancient Maps 10
       2.1.2. Ptolemy 11
       2.1.3. Equal Area Projection 12
       2.1.4. Mercator Projection 13
       2.1.5. Compromise Projection 16
       2.1.6. United States Road Maps 17
       2.1.7. Digital Maps 19
   2.2. Previous Work In Map Realignment 20
       2.2.1. Description of Digital Orthophotos 20
       2.2.2. Uses of Digital Orthophotos 21
   2.3. Summary 22
Abstract

The goal of this thesis was to develop and test algorithms that could be used to automate map realignment using sensed data. Specifically, I wanted to use the GPS data set that was available to me. By studying the characteristics of the data set, algorithms were created to try to realign the maps while minimizing the effects of the distortions in the data set. A standard measure of fit was created to analyze how well the data fit a given set of links or intersections. In total, three algorithms were created based on the assumption that the existing maps are more right than wrong. Two of the algorithms focus on realigning intersections. The first is an iterative algorithm which computes a measure of fit after each iteration slowly moves toward the optimal location for the intersection according to the measure of fit that I created. The second algorithm attempts to create a geometric model of the intersection and solve for the new coordinate algebraically. The third algorithm adds shape points to a link which has already had both of its end nodes realigned. It divides the link into different sections and updates each section individually. These algorithms were tested using the real world data with mixed results.
Acknowledgements

I would like to thank Professor Kornhauser for his help and guidance throughout my writing this thesis. His knowledge and experience were invaluable to me.

Many thanks are also due to Rong He who always knew the smallest, but essential, details.

I would also like to thank TravRoute and ALK for allowing me to use their software and providing technical support. Additionally, unless otherwise sited, all images in this thesis use their maps.

Finally, I would like to thank my editors Mary Cranston, Paul Haaga, and especially Steve Vincent.
I hereby declare that I am the sole author of this thesis.

I authorize Princeton University to lend this thesis to other institutions or individuals for the purpose of scholarly research.

__________________________
John Cranston

I further authorize Princeton University to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

__________________________
John Cranston
A First Step Toward Automated Map Realignment

John Cranston
Advisor: Professor Alain L. Kornhauser
April 14, 2003

Submitted in partial fulfillment
of the requirements for the degree of
Bachelor of Science in Engineering
Department of Operations Research and Financial Engineering
Princeton University