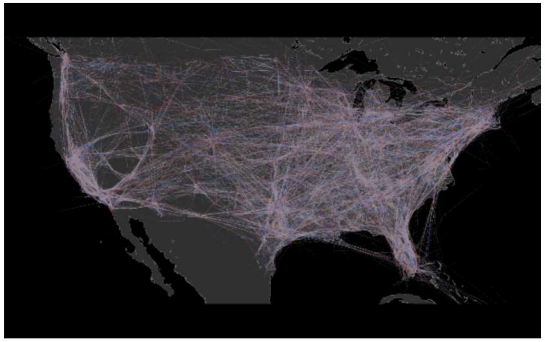
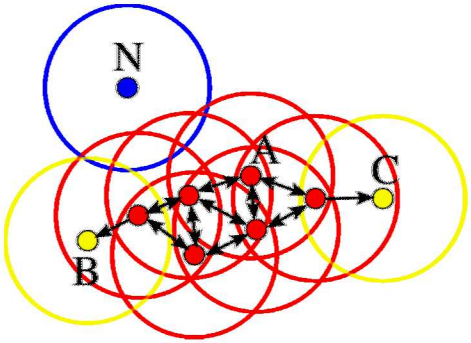
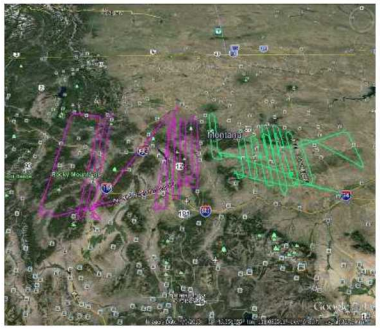
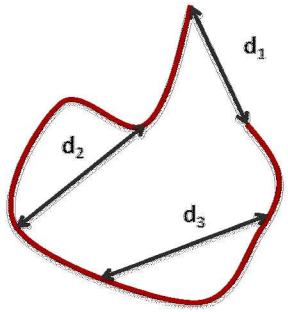
SAND2019-10012PE









Trajectory Data Analytics

August, 2019

Mark (Danny) Rintoul

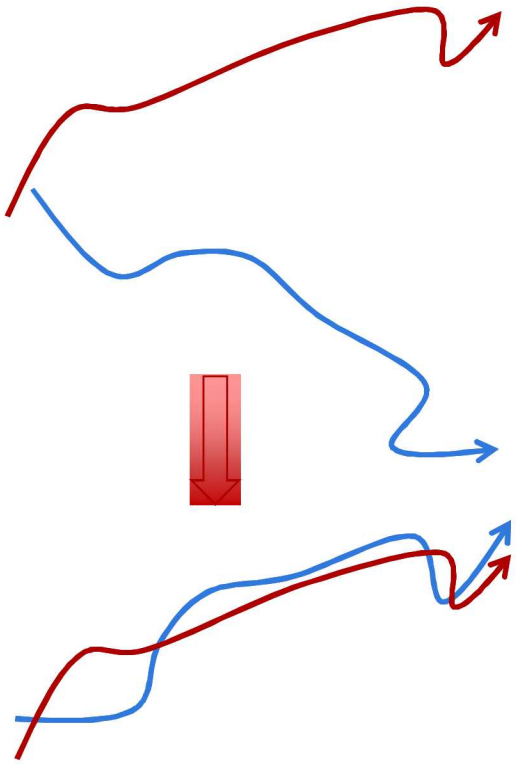
U.S. DEPARTMENT OF ENERGY | NNSA National Nuclear Security Administration

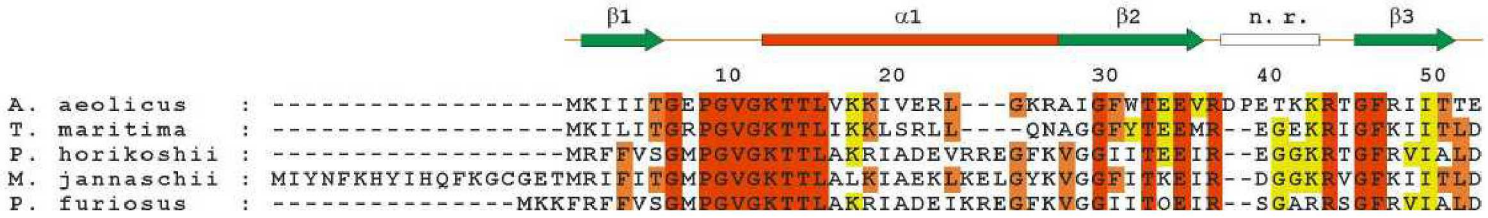
What are the mission problems?

- Enormous corpus of position/time data thanks to GPS and new sensing technologies
- But…often very little other information
- Questions then related to comparing trajectories:
  - Have we seen this pattern before?
  - Have we seen a pattern related to a known behavior of interest?
  - Can we group together similar patterns, and…
  - …can we identify patterns that are unique (outliers)?
  - Can we predict where a current trajectory will go based on historical observations?
  - Can we identify different types of collective behavior among many trajectories?

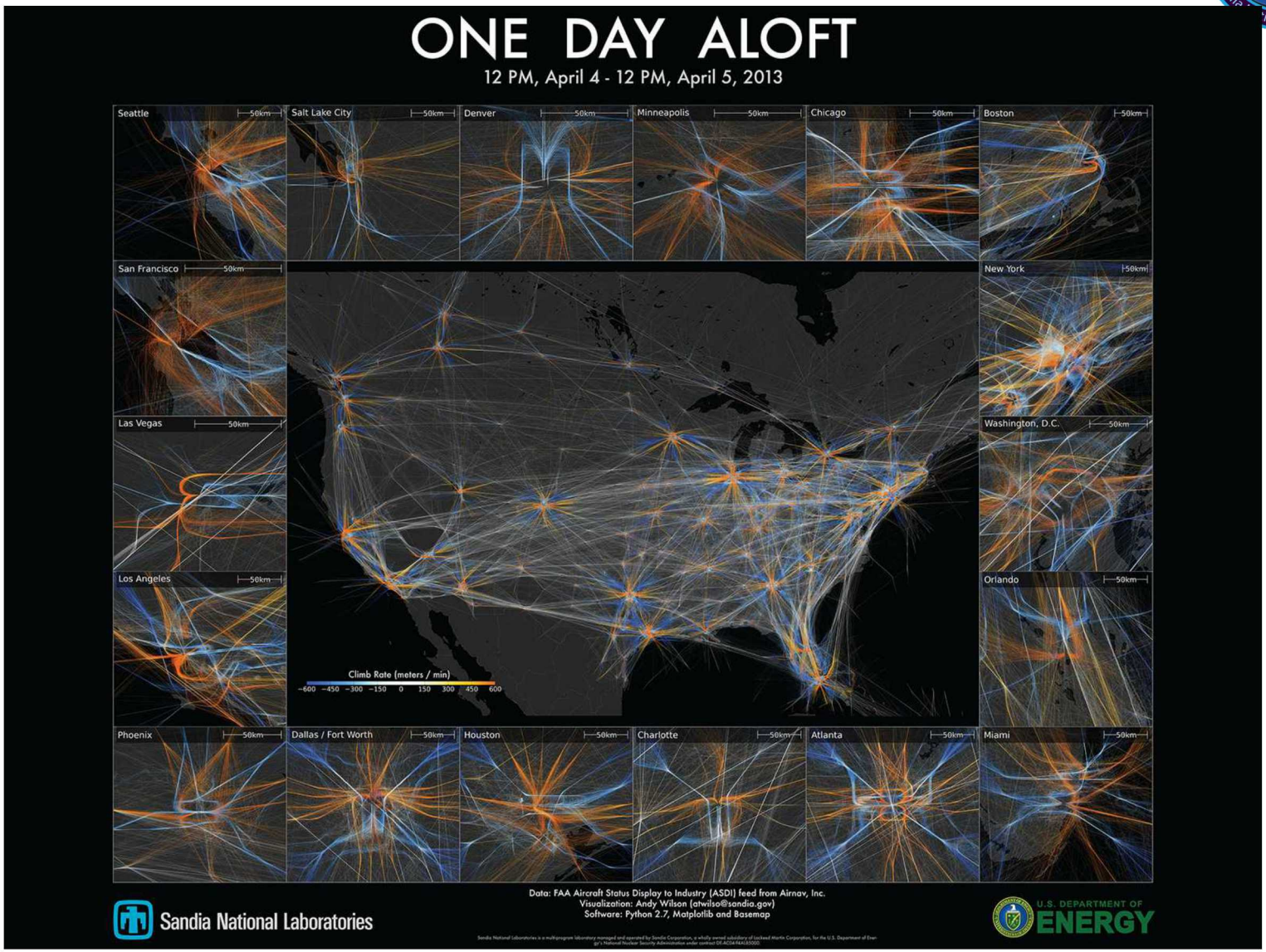
So, starting out…

- How do you compare two trajectories?
- First thought? Try line them up and measure differences
  - This is the core of how it is primarily done in the literature
  - Basically, you align their starting points and calculate how much work it is to munge the two curves together.
  - Can think of it as an edit distance sort of approach. **Very SLOW!**
  - Used in bioinformatics, and spell checking!





Scale of data



New Idea

- Why use the curve alignment process at all?!?
- If you have enough numerical descriptions of the curve, those should mostly uniquely describe it!

- End-to-end distance traveled
- Total distance traveled
- Ratio of end-to-end distance traveled to total distance traveled
- Total curvature
- Total amount of turning
- Average heading change
- Area covered by flight (convex hull of points)
- Eccentricity of the convex hull
- Perimeter of convex hull
- Centroid of points
- Centroid of convex hull
- Start/Stop point
- Nearest distance to a given point
- "Distance" from a given specified track
- "Distance" from a given specified shape
- Start/Stop time
- Time nearest to a given point
- Average speed
- Range of speeds
- Max altitude
- Fluctuations of altitude/shape of altitude/time curve
- Difference from historical data
- Most common speed/altitude (cruise)
- Place, time and heading where first seen / last seen (might not be start/stop points)
- Other…

But what are we really doing now?

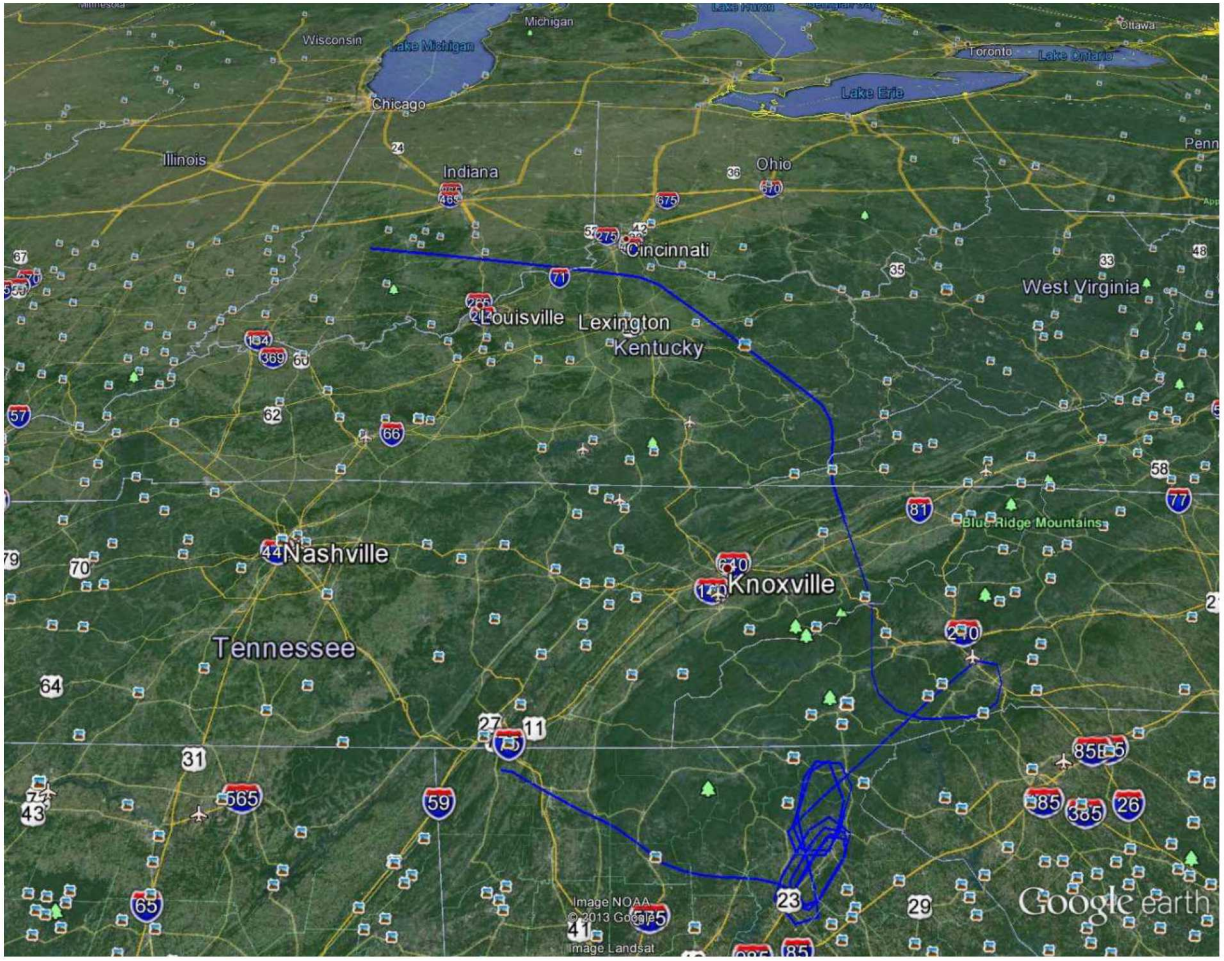
- Building up a list of descriptors about something is called a ***feature vector***

$$f_i = (x_{1i}, x_{2i}, \ldots, x_{ni})$$

- Once you have a feature vector with *n* features in it, what you have is a set of points in an *n*-dimensional space.
- Now, you can really do some analysis
- Most importantly, if your feature vector contains the features that represent similarity in the flights, you can answer key semantic questions by defining a metric on the space and looking for points close together
- Can be stored in a traditional database structure

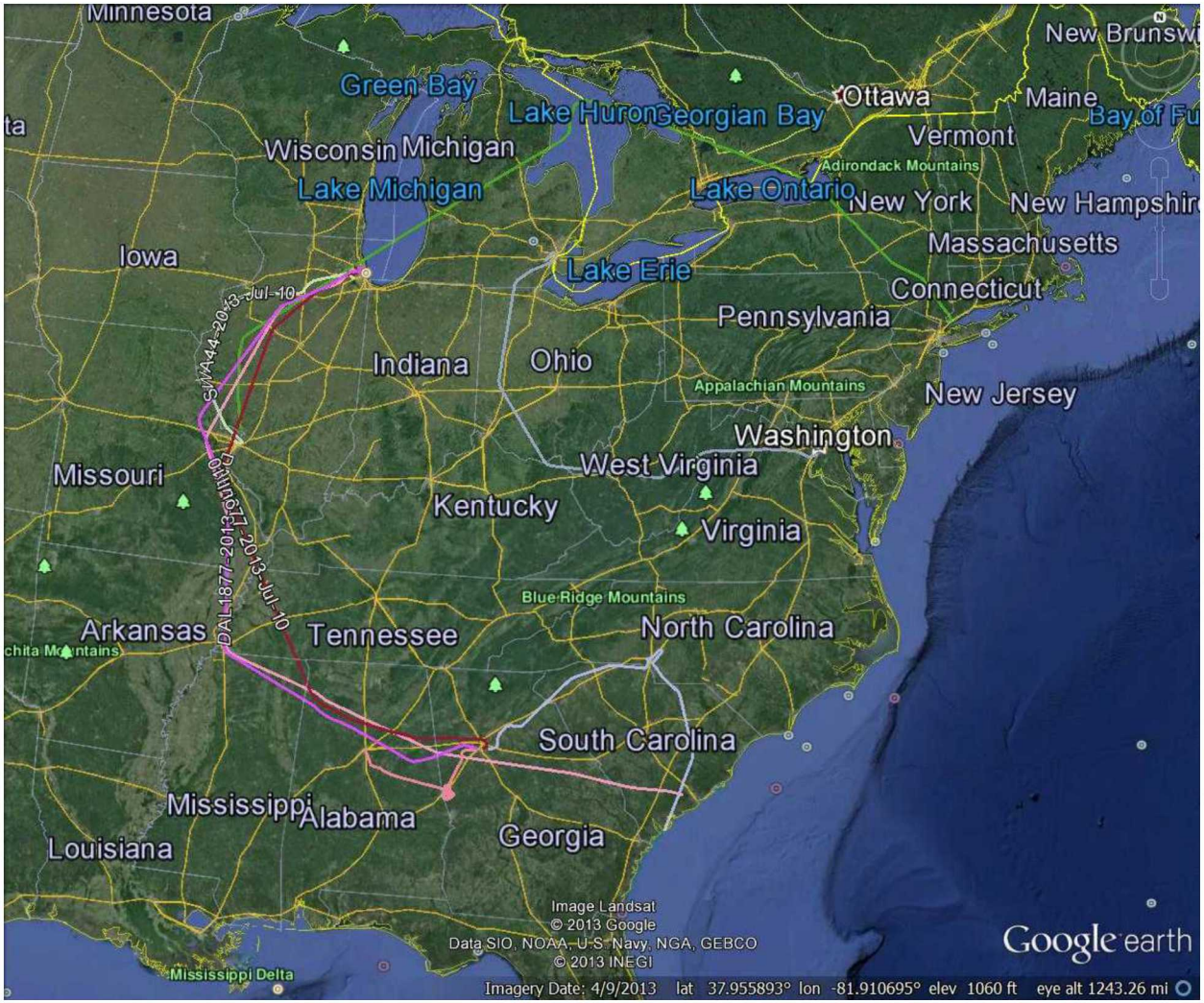
Example: Find Holding Pattern

- Looking for significant distance traveled, but lots of circular motion…



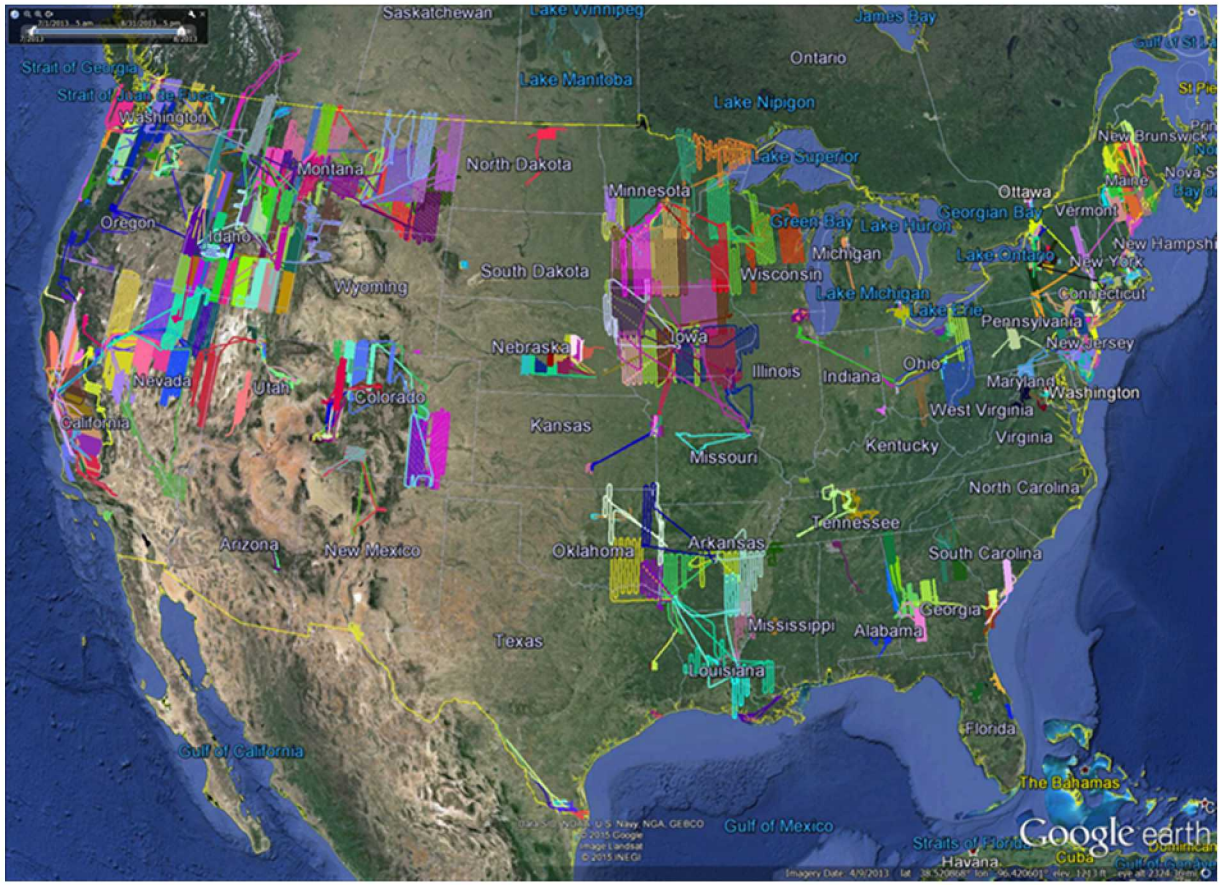
Example: Avoiding Airspace

- Flights avoiding airspace might be not quite-point-to point, but covering a wide area. Look for some clustered in area and time, and you get…



Example: Mapping flights

- Lots of ways to do it, but easiest is to just look for long, but compact flights
- Can also make “turn around” features



Spatial Indexing: Motivation

- Key idea of geometrical descriptor is that a trajectory can be characterized by a *feature vector*
  - $V = (f_1, f_2, \ldots, f_n)$
- Previous work had focused on searching based on the ranges of feature values
- However, being able to work in the abstract vector space of the features allows many more powerful approaches
  - Neighbor search
  - Clustering
  - Outlier detection
- However, you need to have a way to find nearby points in a computationally efficient manner

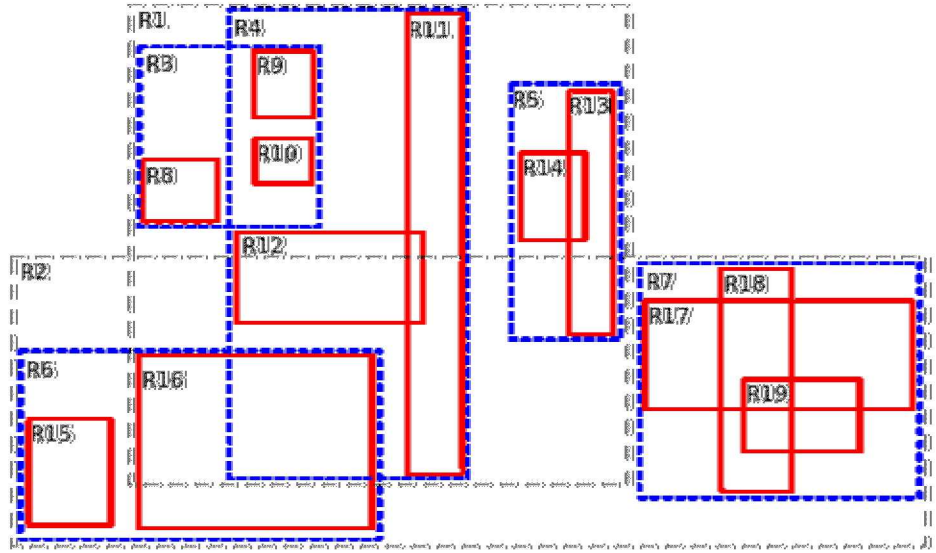
Killer app: Building a spatial index

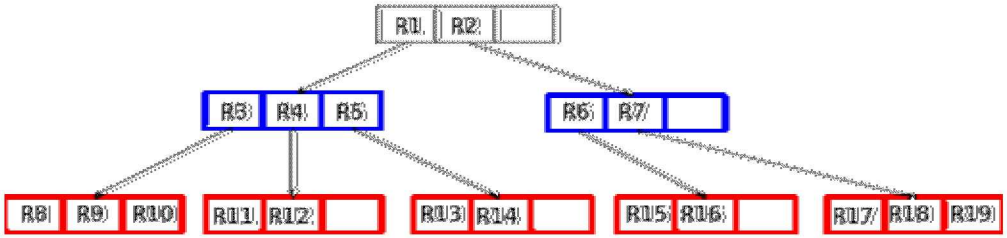
- Now, instead of just checking a list of flights for similarity to another flight, we can build a *spatial index*.
- A spatial index is a special data structure that can be used to find points quickly in n-dimensional space
- Most provide O(log n) access to the points, which makes finding neighboring points very fast, even in large ($>10^6$) databases.
- We use an R-tree, although there are others (*kd*-tree, etc.)
- Once you have a spatial index (and the associated fast neighbor look-up), you enable clustering

Spatial Indexing: R-Trees

Features:

- First proposed by Antonin Guttman in 1984
- Used for spatial indexing to enable fast search
- Essentially, just a hierarchical tree of bounding boxes, including leaves
- All leaves are the same height (balanced!)
- Search is $O(\log_M n)$, where n is the number of items and M is the max number of entries per node.
- Insertion/deletion is worst-case $O(n)$ and can be ugly

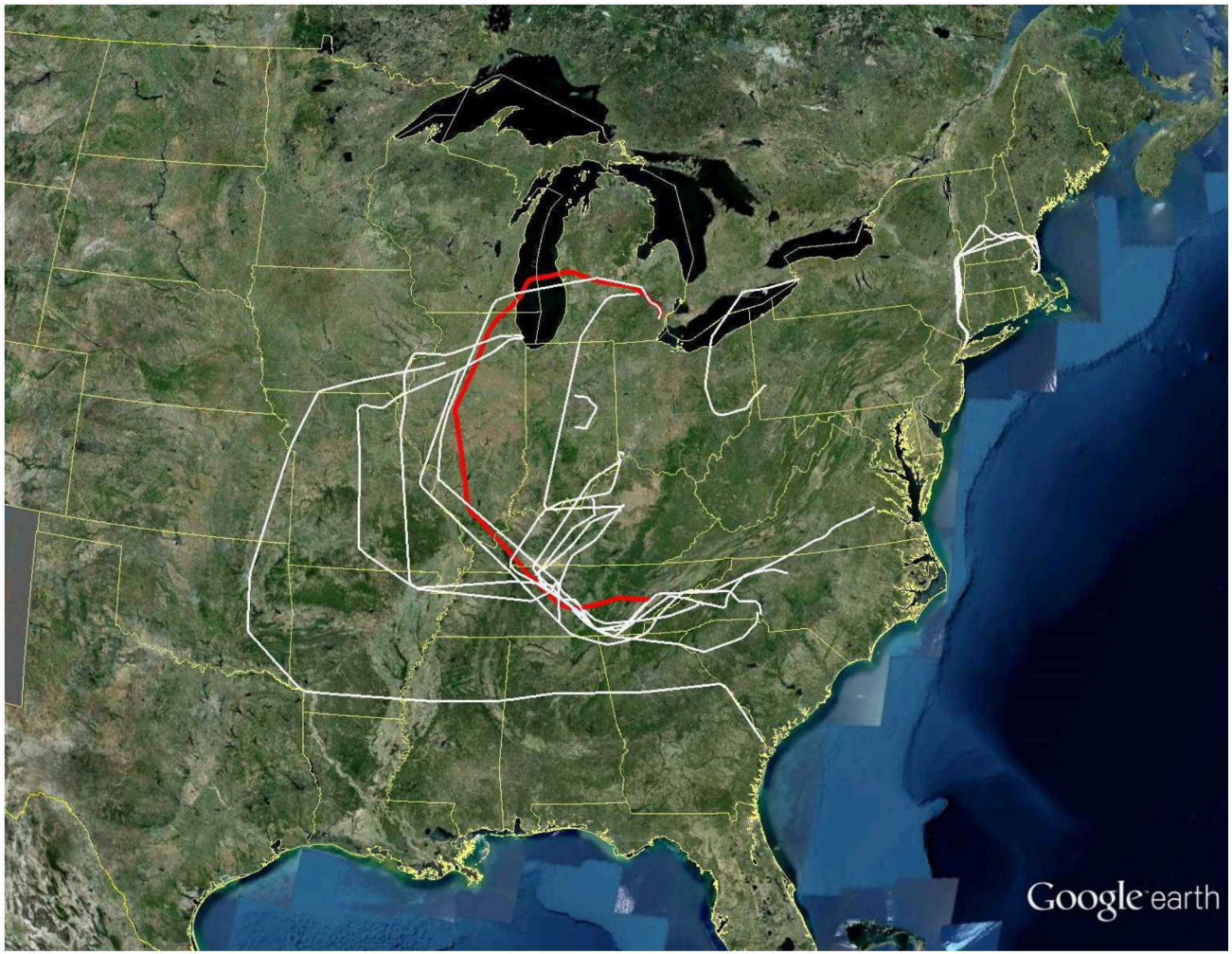




Guttman, A. (1984). "R-Trees: A Dynamic Index Structure for Spatial Searching". *Proceedings of the 1984 ACM SIGMOD International Conference on Management of Data - SIGMOD '84*. p. 47.

Example: Find Similar Shaped Flight

- Consider the red flight, and look at just two features:
  - Ratio of length to end-to-end distance
  - Aspect ratio of convex hull
- Search for flights with similar values gives the white flights
- Note that we have a superfast shape-invariant search now!

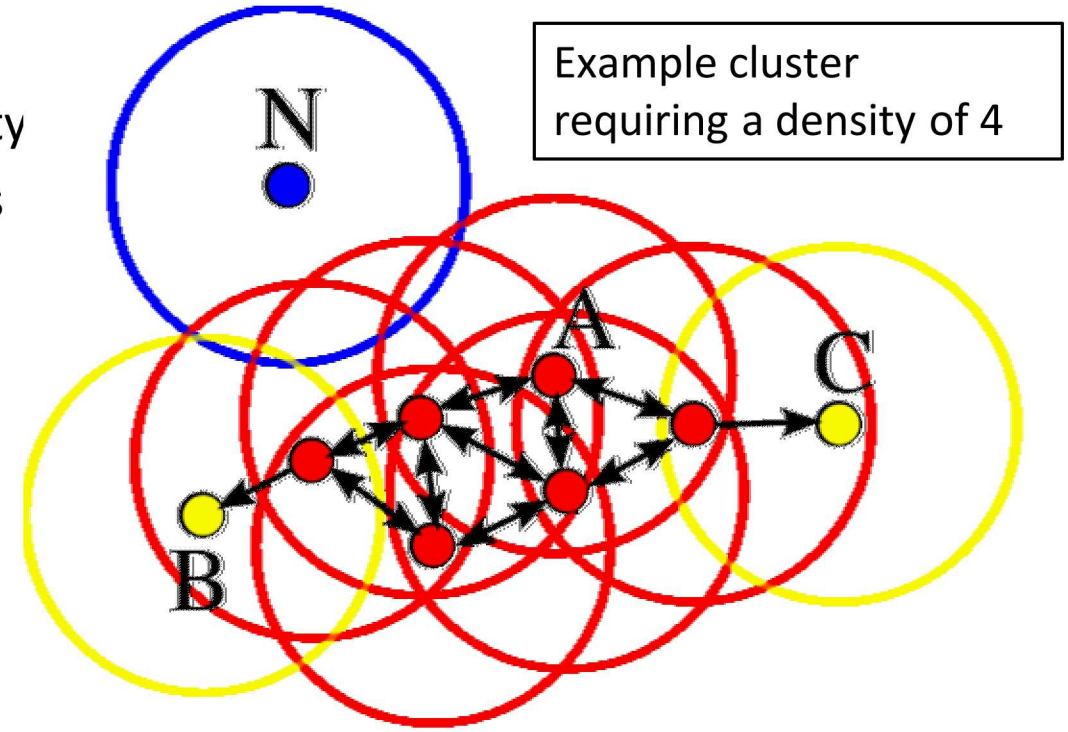


Clustering: Motivation

- The previous uses of feature vectors are very powerful, but are in the class of “supervised” approaches.
- What we ideally want, is the computer to automatically
  - Finding natural neighbors of a flight
  - Find unexpected patterns
  - Find outliers
- Once you have your flights described by feature vectors and have the capability of spatial indexing, these problems can all be solved by *clustering*.

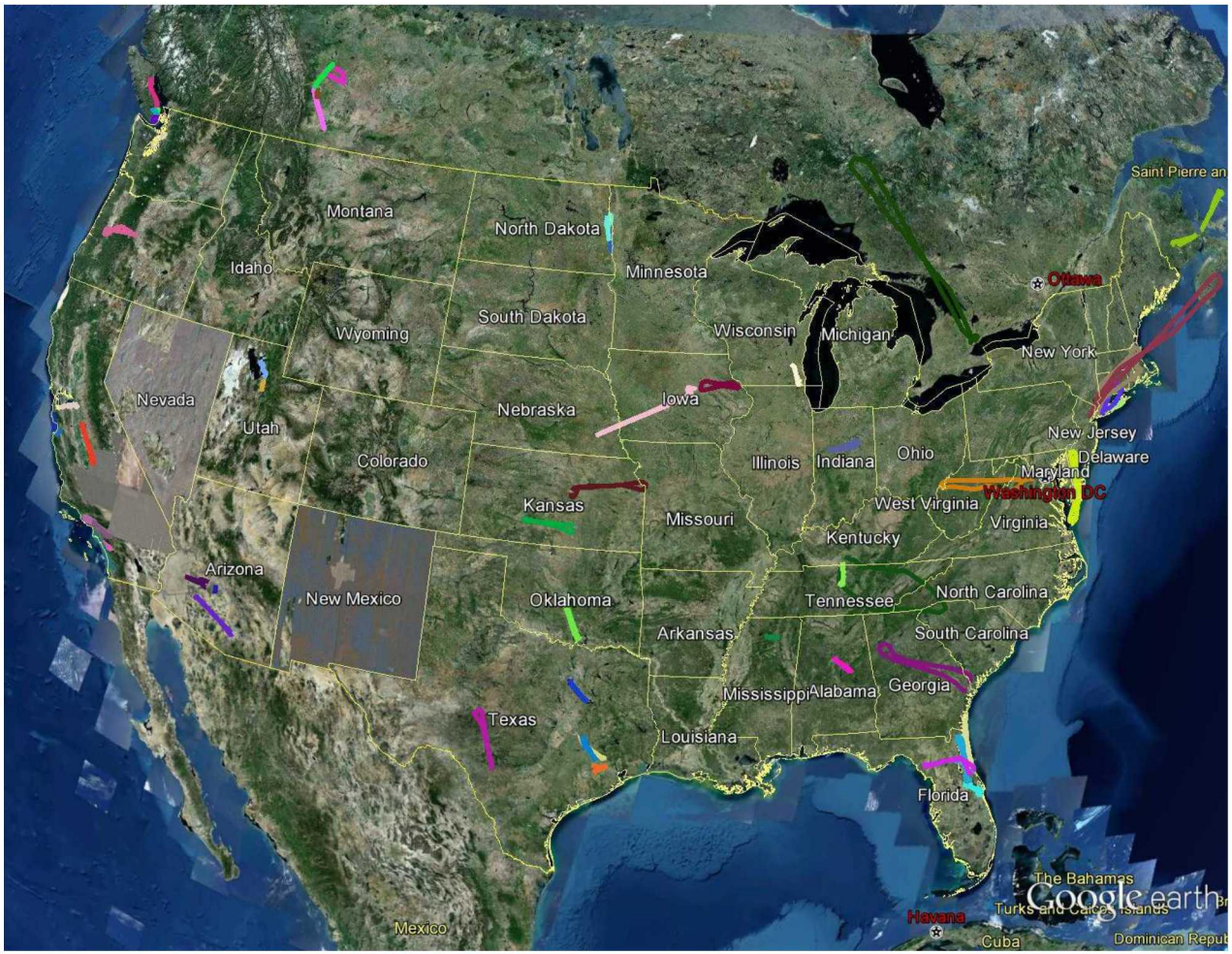
Clustering: DBSCAN

- Many different types of clustering algorithms (k-means, EM, etc.) but DBSCAN has two nice properties
  - Doesn't require a pre-defined number of clusters
  - Has the notion of noise (outliers)
- DBSCAN requires
  - A neighborhood density
  - A neighborhood radius



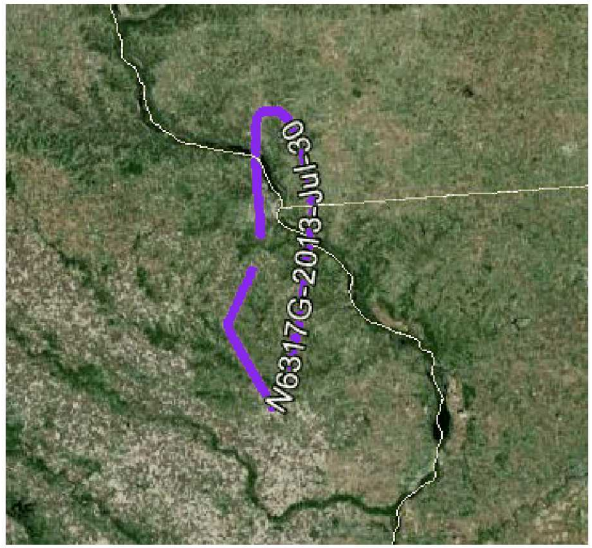
Clustering: Results

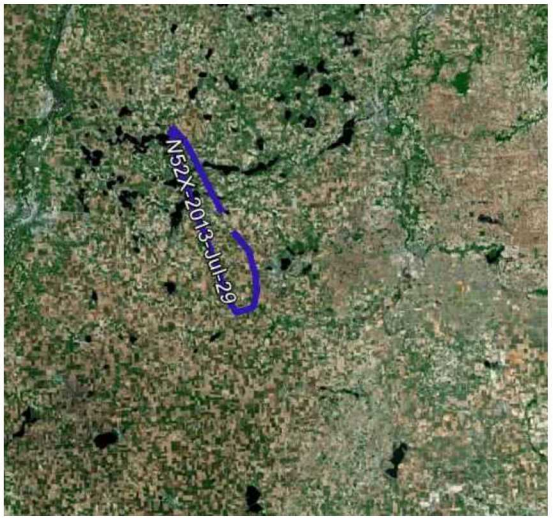
- Did not tell it to find this, only told it to “make groups of similar flights”
- This was one of many clusters that had distinctive shapes

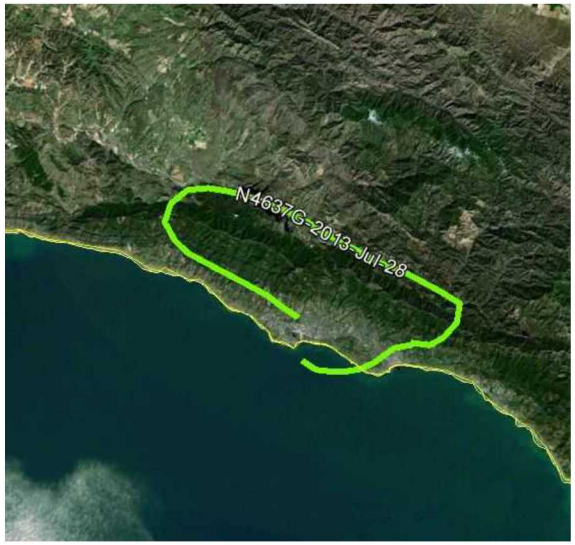


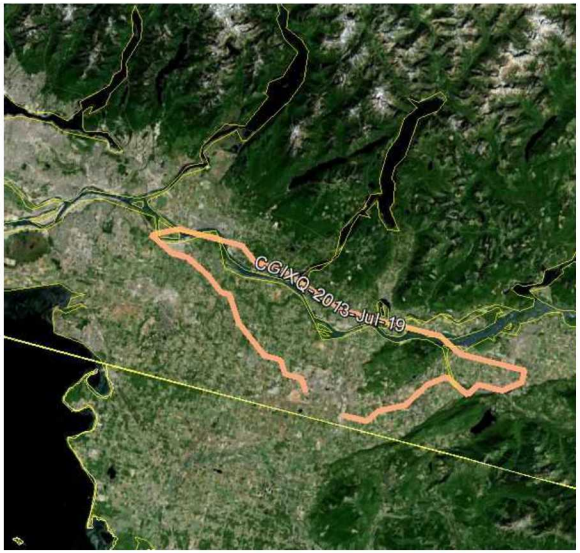
Clustering: Results

- "Paper Clip" shaped flights
- Different scales, orientations
- Seemed to be practice take-off/landing





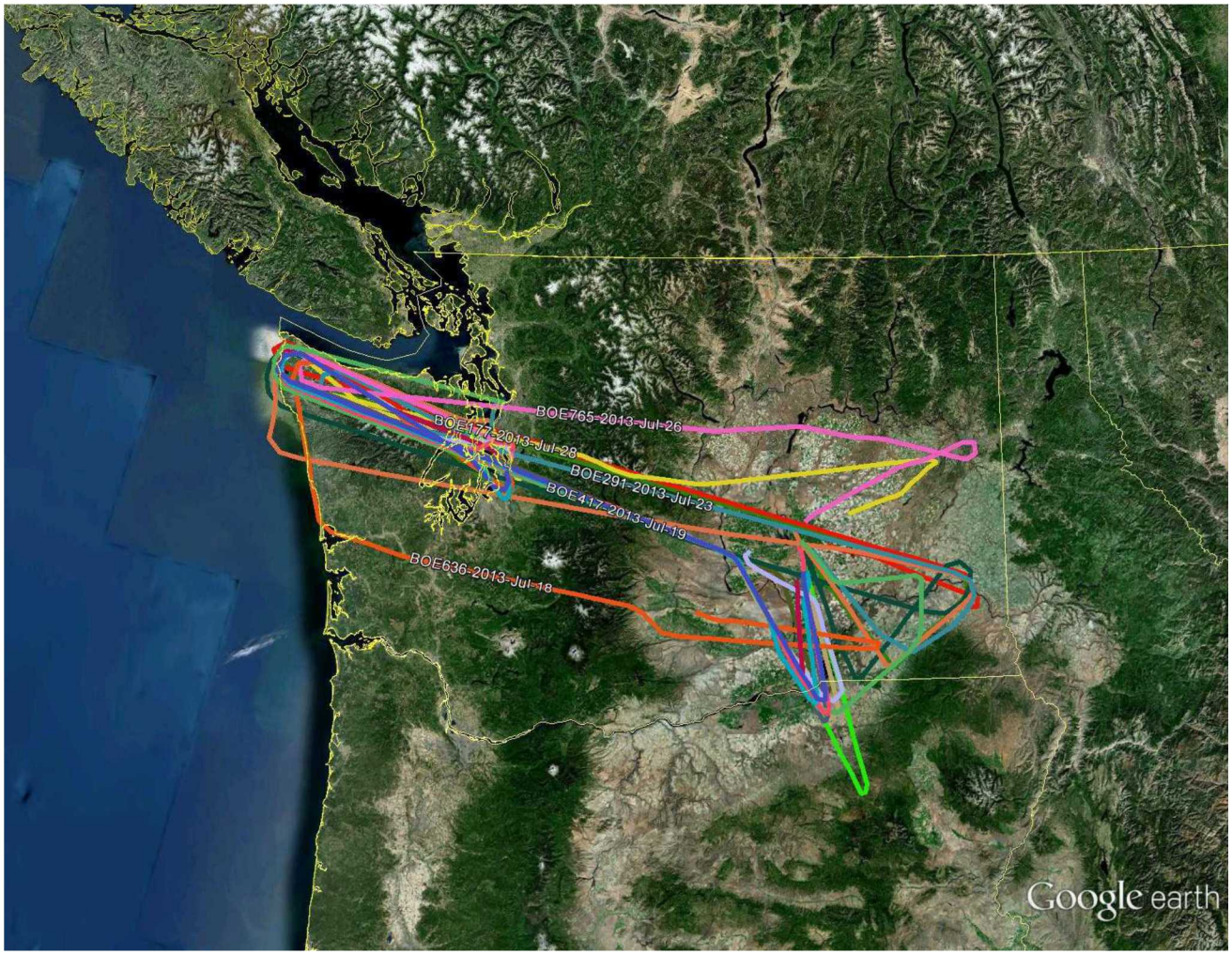




Clustering: Results

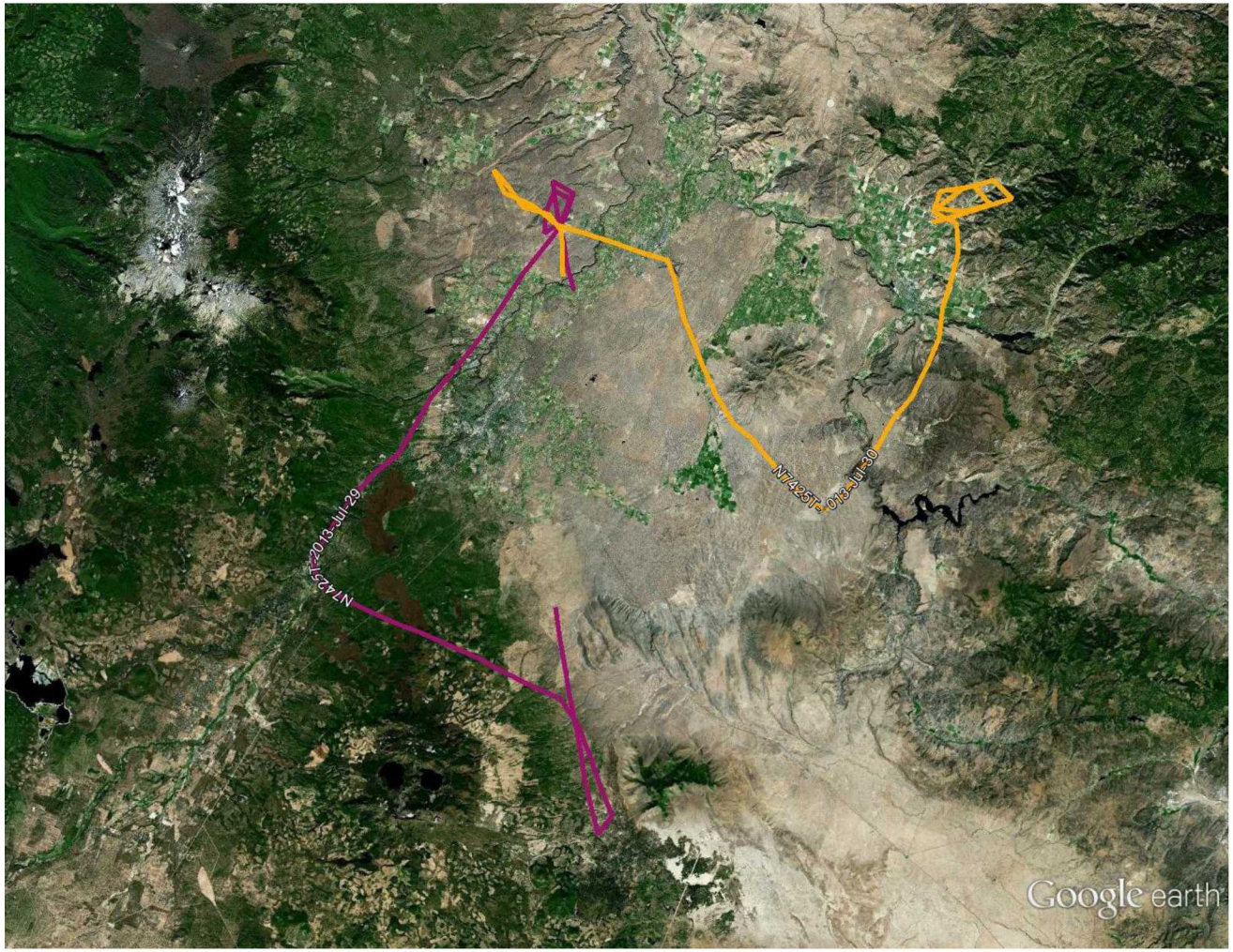
- Was not doing proximity, only shape
- But, found a group of similar shaped flights
- What is this odd behavior?

- Boeing flying from its factory to a test field in eastern Washington



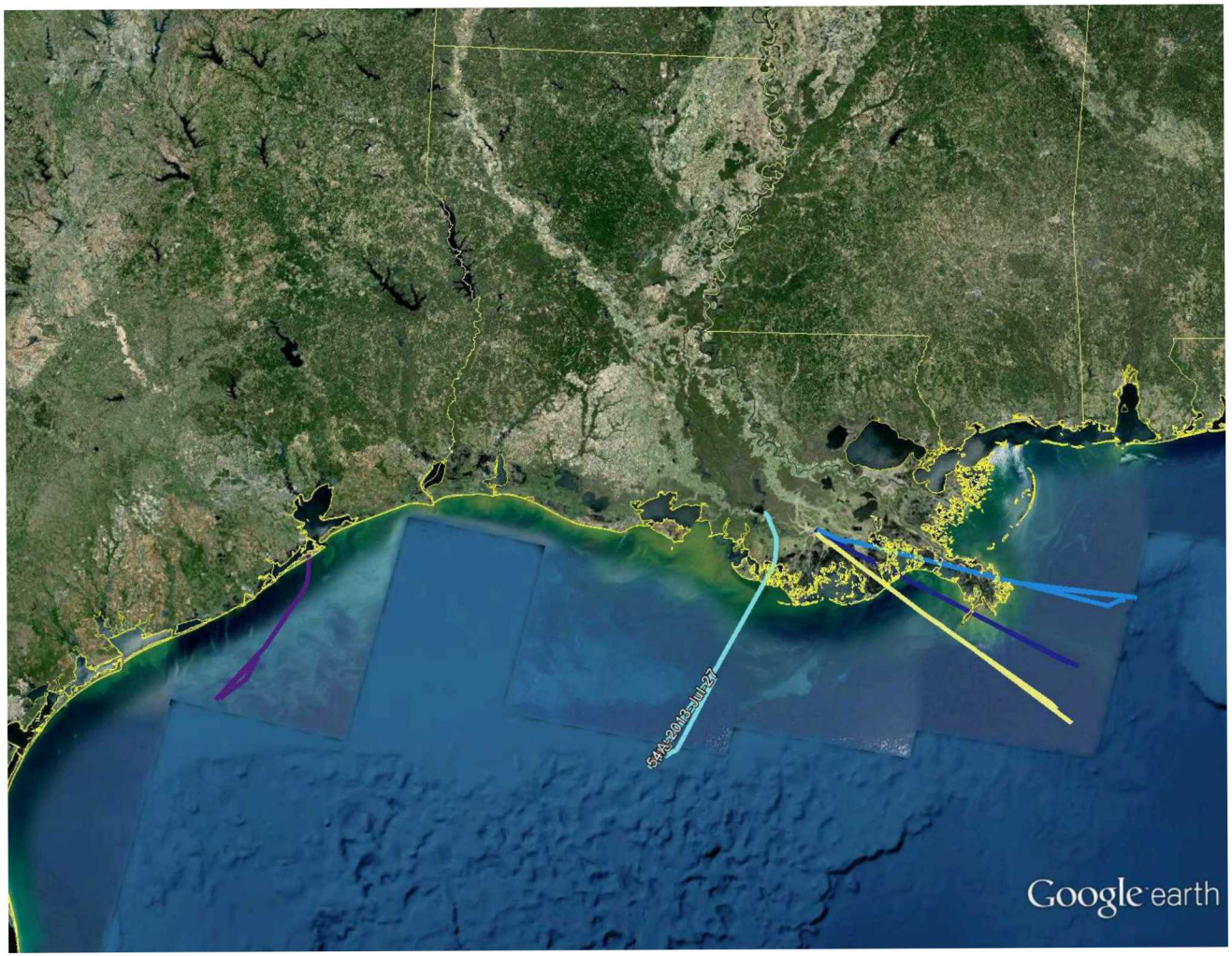
Clustering: Results

- Same aircraft, different days (still only clustering on shape)
- Flew almost identical pattern in different orientation



Clustering: Results

- Just clustering on shape, found these 5 similar flights
- All of them were helicopters belonging to the same company (Petroleum Helicopters)
- They fly people/equipment out to oil rigs



Collective Behavior

- If you look for similar geographic location **plus** similar time (time can be a feature too!), clustering will give co-travelers
  - Can find convoys
  - If your database contains two different objects, can correlate them (e.g. a phone and a vehicle)

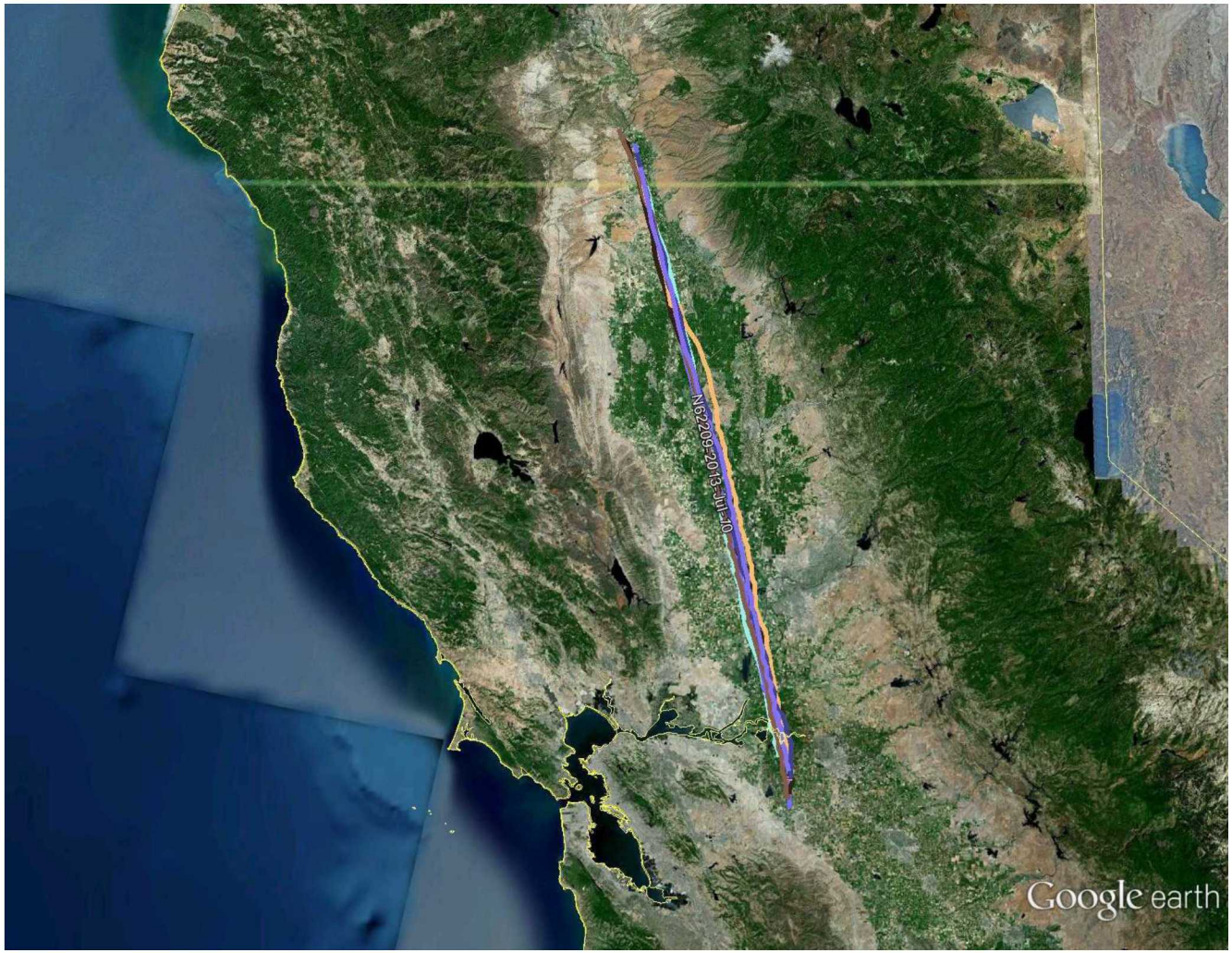
- This can also be used to find to trajectories that cross, or meet up for an extended period of time

Clustering: Results (place & time)

- Find flights that followed the same path, approximately the same time
- A group of 6 Cessna's all flew from LA to Redding in the morning on July 10th.

Clustering: Results (place & time)

- … and then from Tracy to Redding around noon on July 10th.



Clustering: Results (place & time)

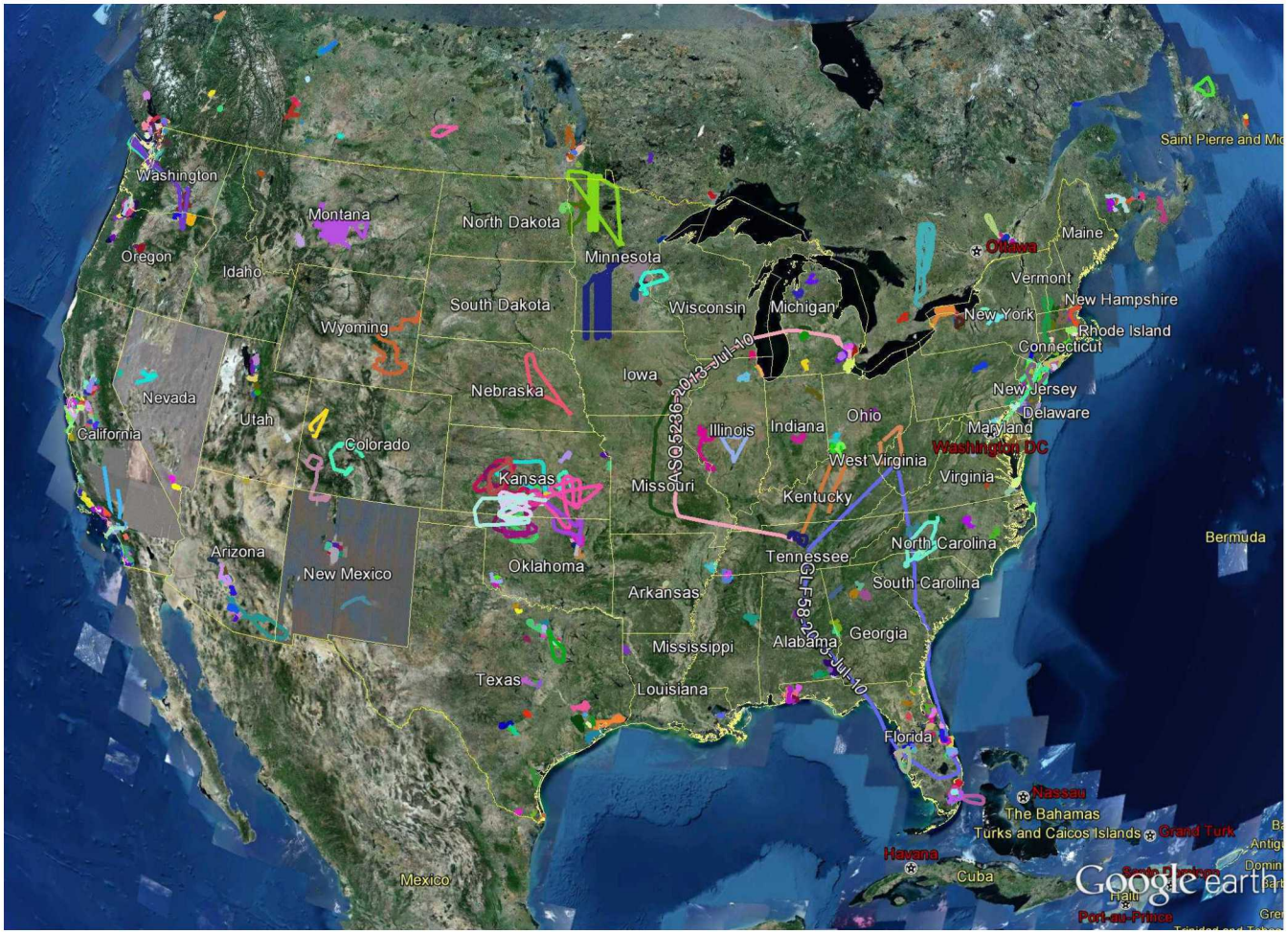
- Find flights that followed the same path, approximately the same time
- A group of 7 UAL flights took off from IAH to Central American destinations right after each other

Clustering: Outliers

- Using all of the tools described previously, we can now tackle what is one of the most important and vexing problems in trajectory analysis: discovering outliers
  - Would ideally want the computer to be able to search data and identify unusual trajectories
  - But defining “unusual” in a rigorous manner is hard
- Using DBSCAN, we can identify the “noise” points (points that don’t belong to any cluster) as outliers, giving us a fast way of finding potentially anomalous behavior.

Odd flights

- Clustering done based on geometric features
- Many clusters found, but what remains is…
- Represents approximately 700 out of a total of 50,000 flights from one day

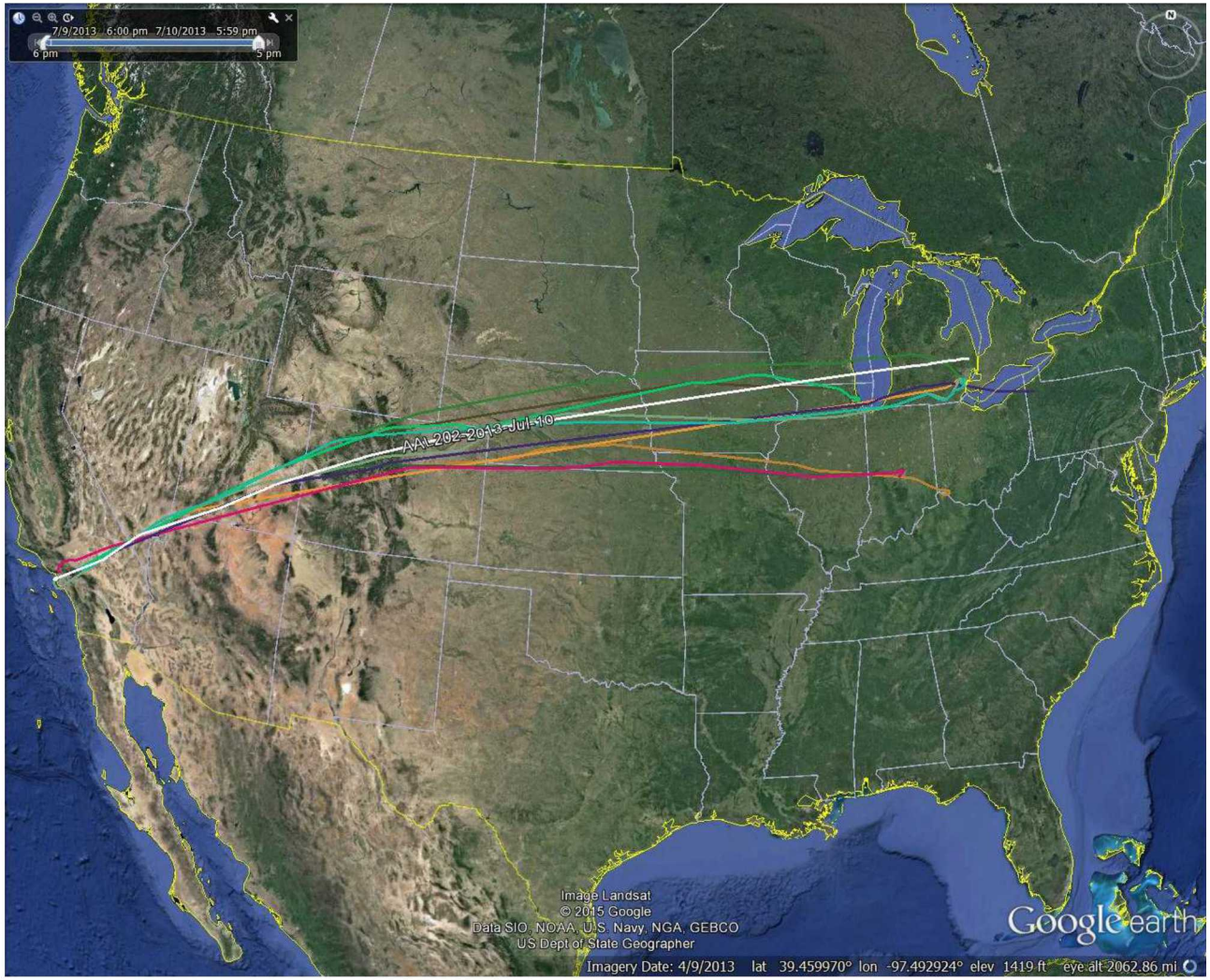


Prediction

- Idea: Put known historical trajectory fragments into a database
- Take the observations of the beginning of a new trajectory, and search for near matches
- Weight the “nearness” of the different trajectories, and sort the different possible destinations
- Work based on observing the first 20% to 80% of a flight, with that fraction unknown to prediction algorithm

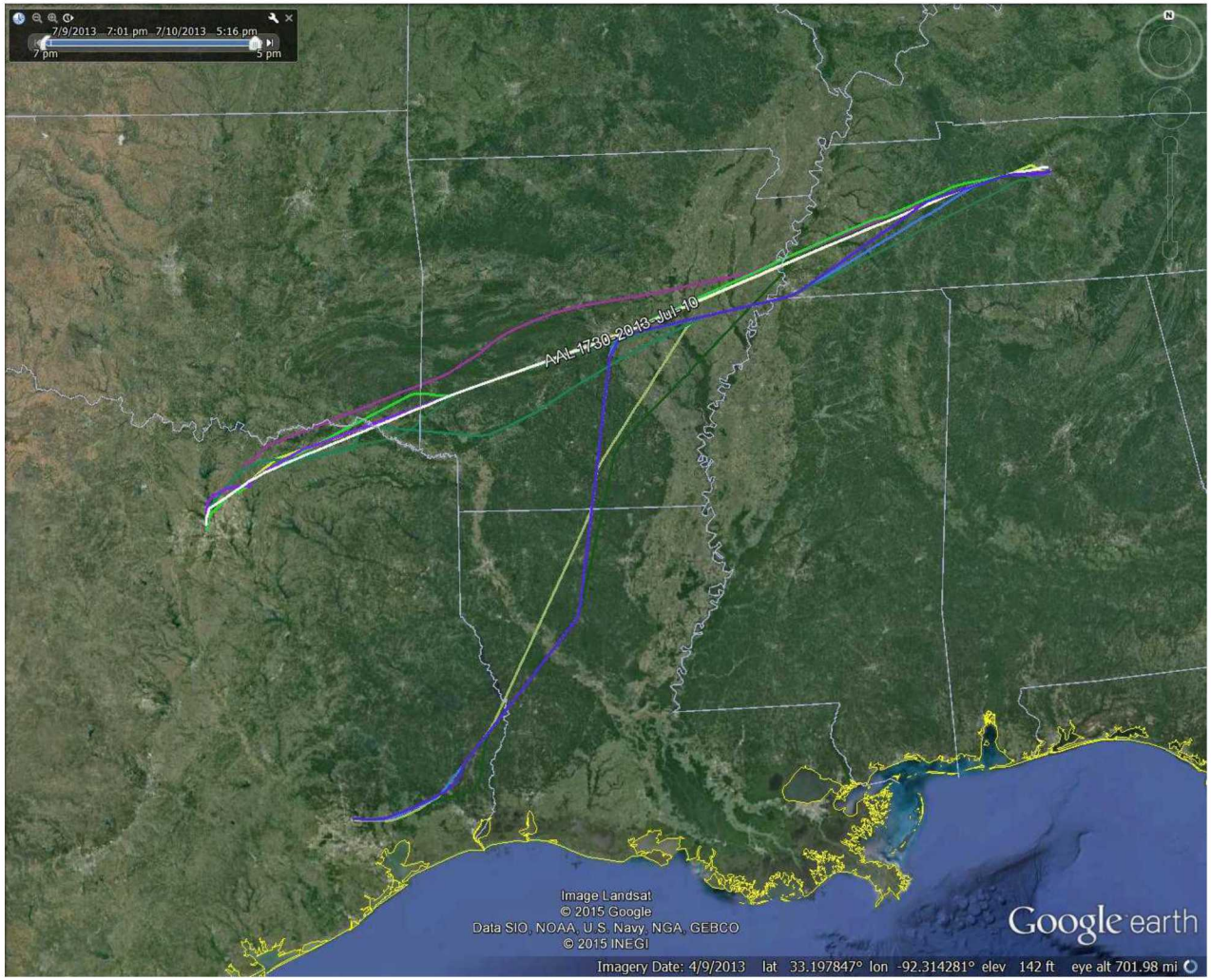
Small Data Set -> Mistake

- Target flight (white) goes from LAX to Toronto, but nearest flights are all going to more common destinations (Chicago, Detroit)



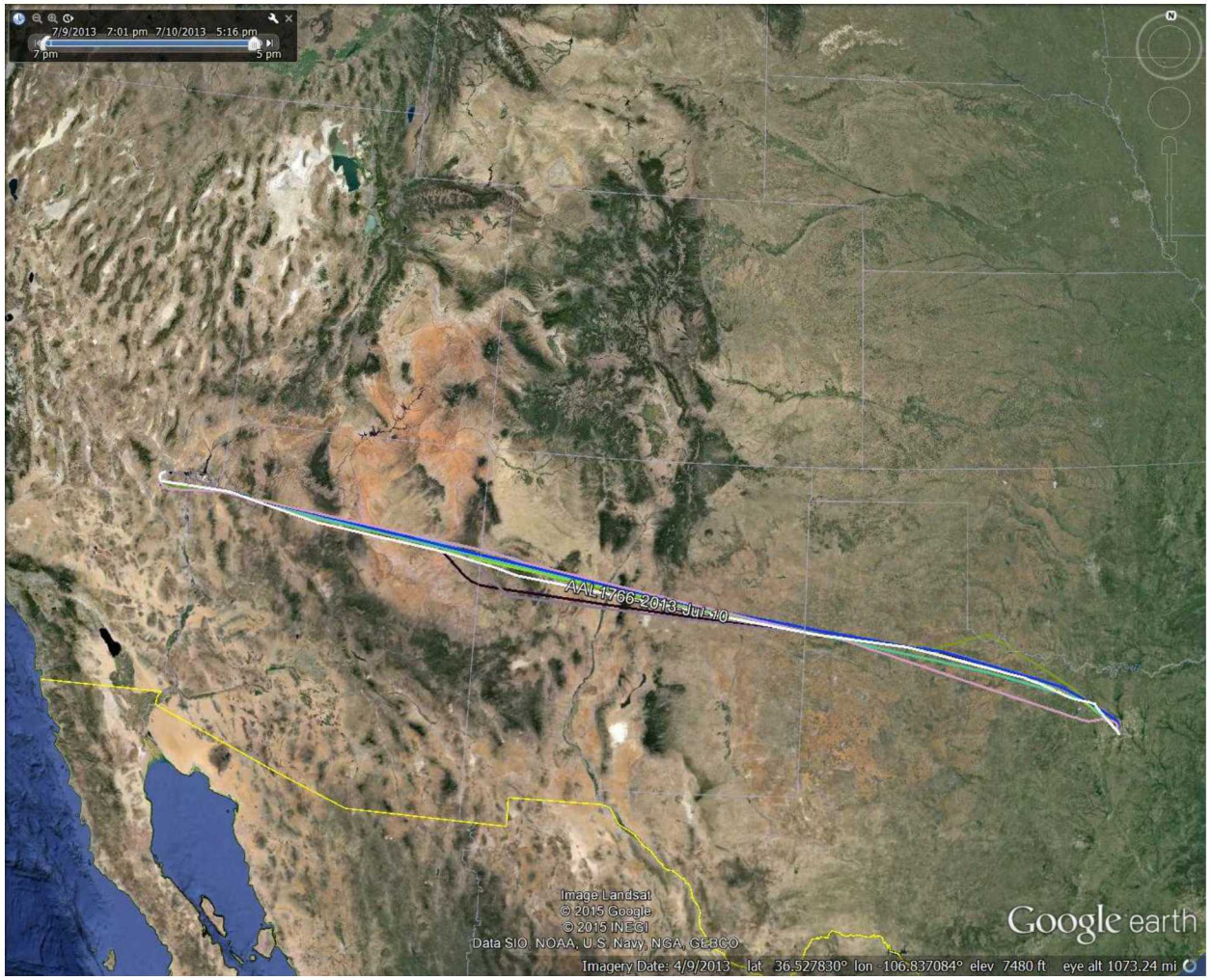
More Data, Better

- Flights from Nashville go to DFW and IAH, but proper weighting finds DFW for the target (white).

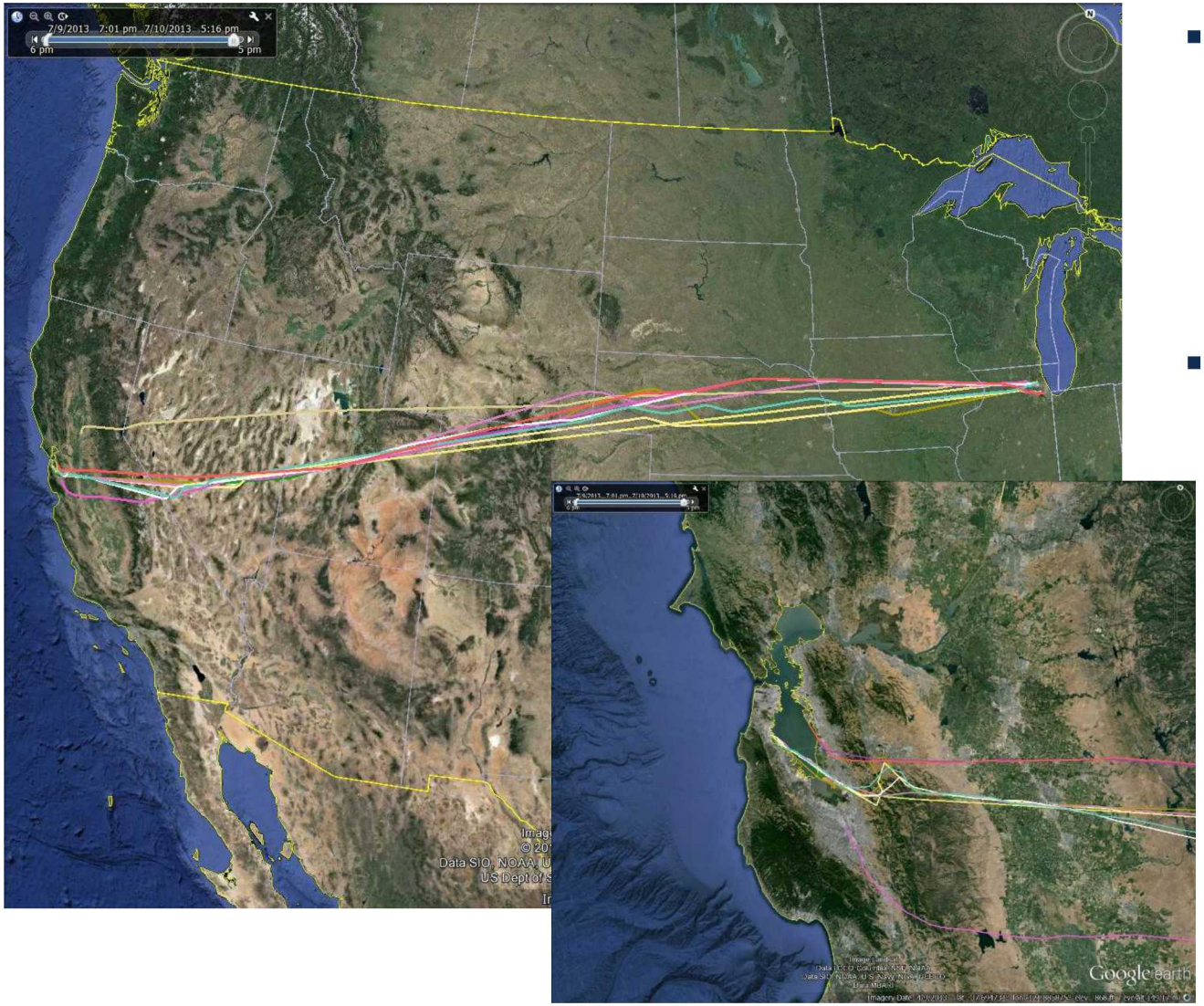


Lots of data

- Popular flights are picked perfectly (DFW to Las Vegas)

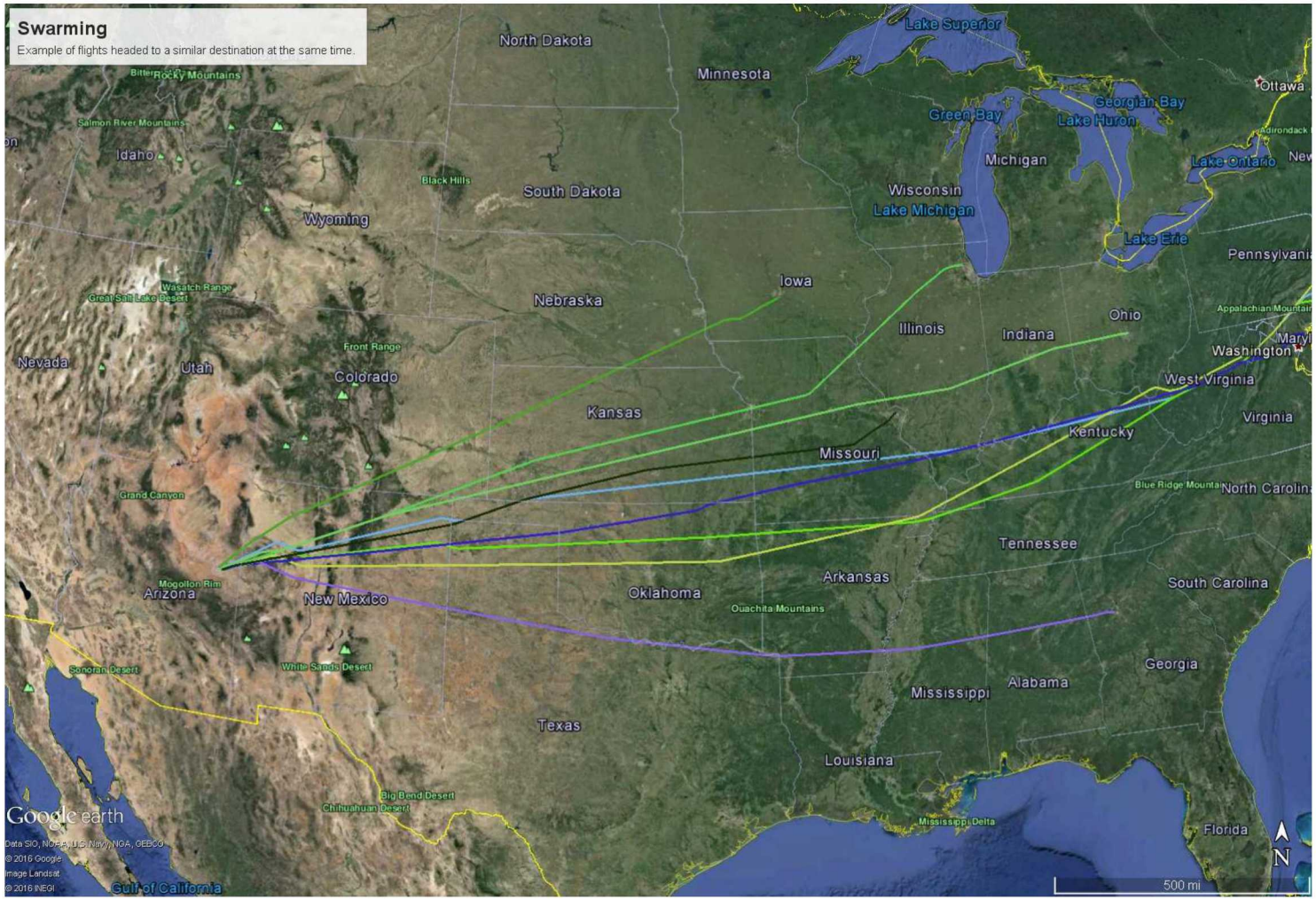


Perfection not possible

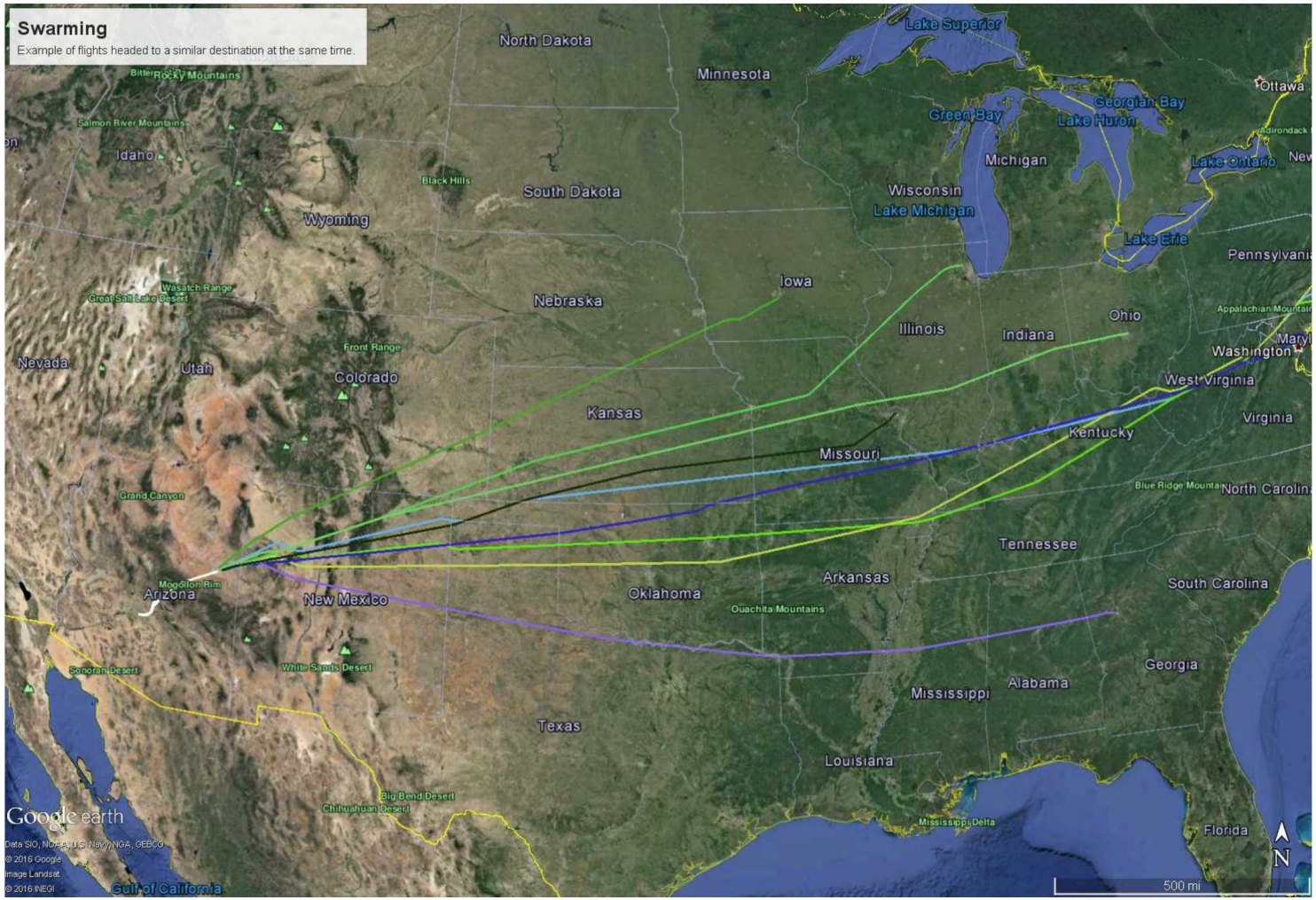


- Some flights to the same airport follow the same path for long time, like Chicago to Bay Area.
- Close-up shows flights going to SFO, OAK, SJC (and one to Sacramento!)

Swarming Example



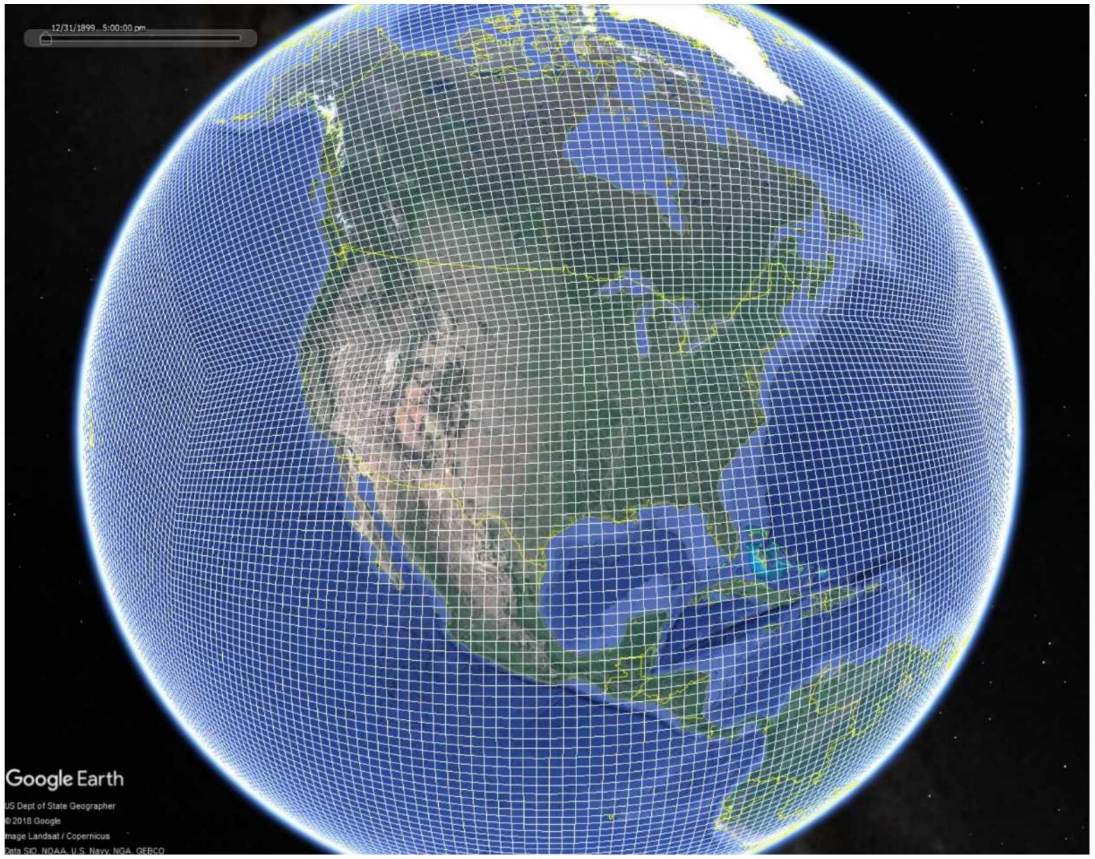
Swarming Example



Big Improvement: Efficient Indexing

Now using an cube-sphere approach

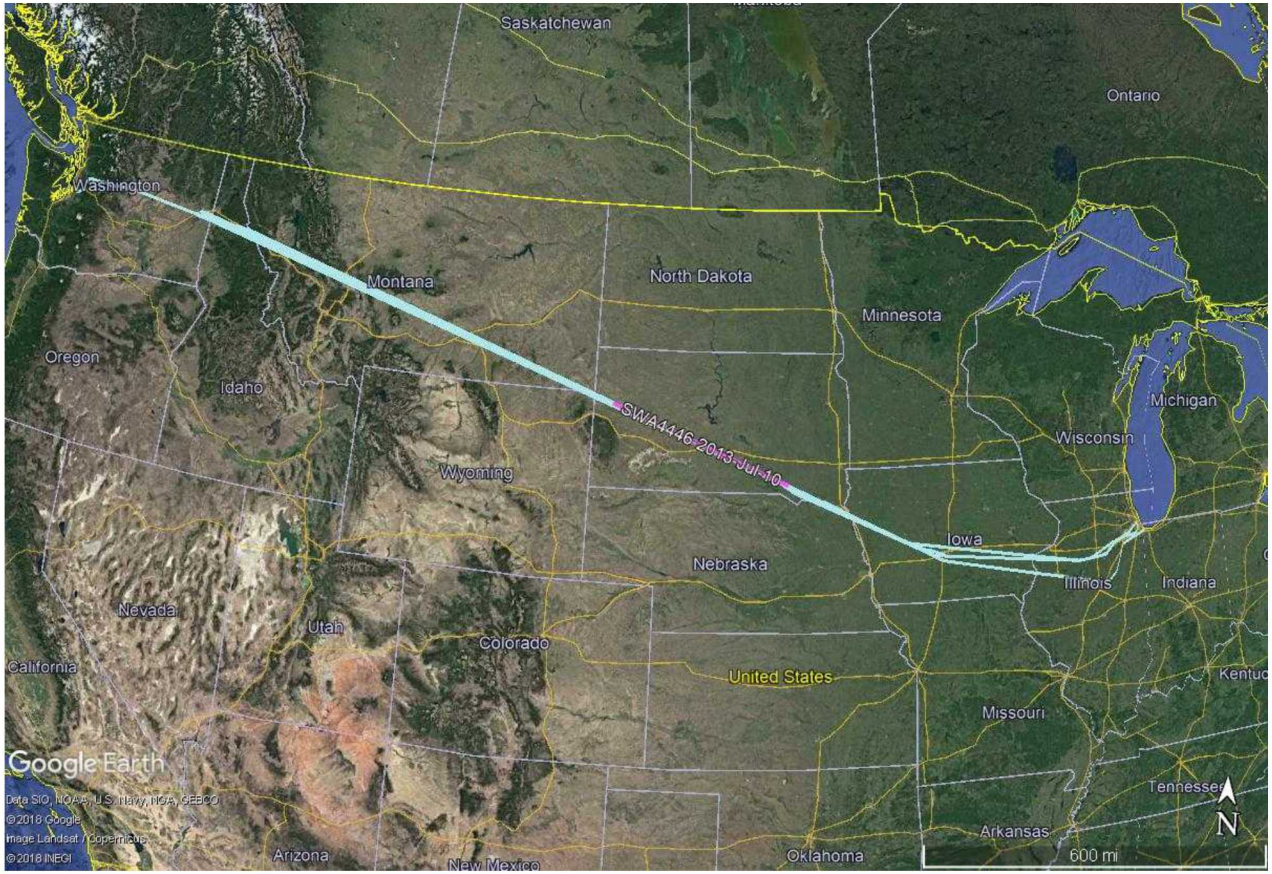
- Much more isotropic near poles
- Math not so bad for indexing
- Allows for easy alignment of fragments on a set of trajectories



Long Fragments Work Well

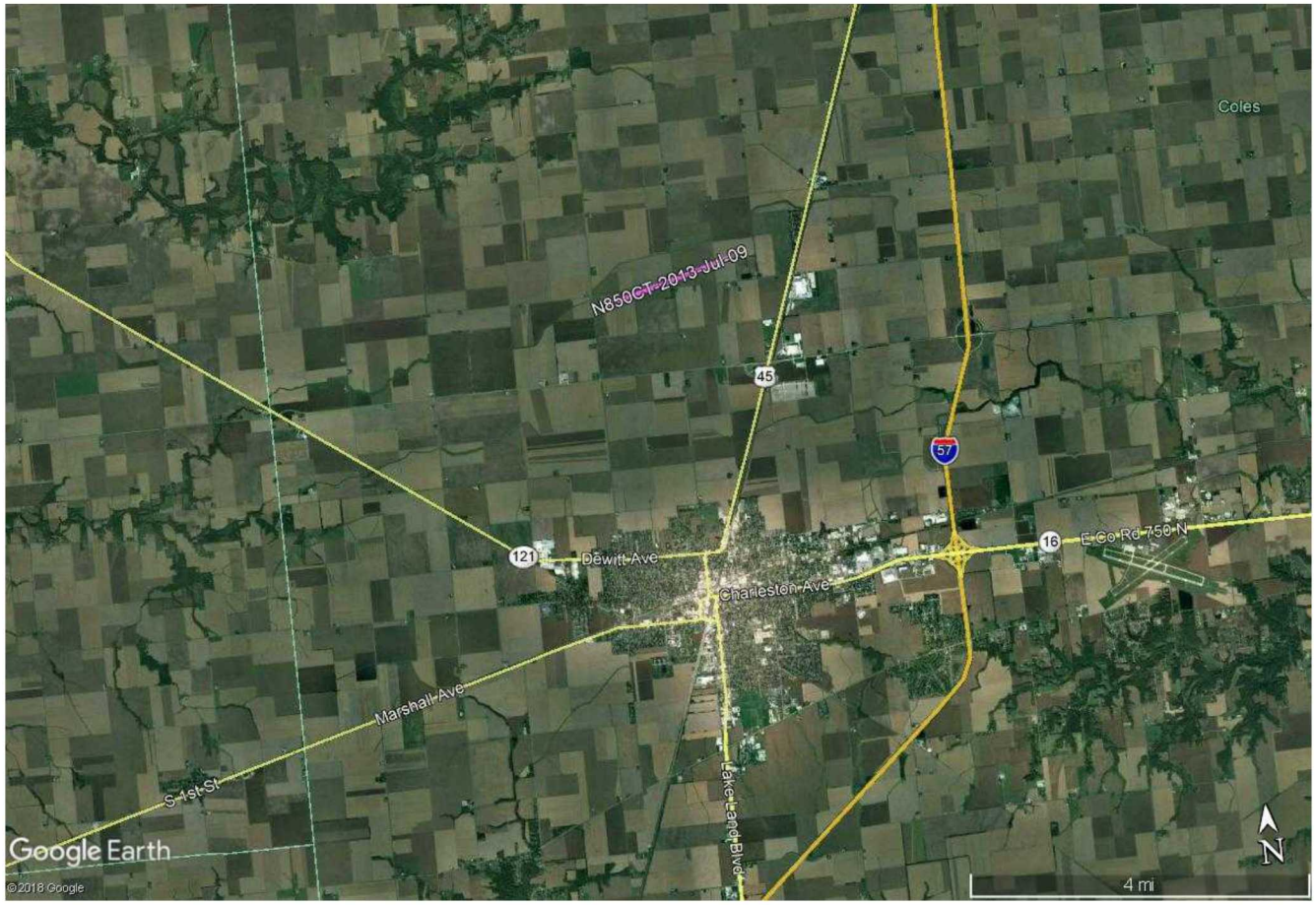
No real surprises for longer air travel fragments

Origin/Destination easy



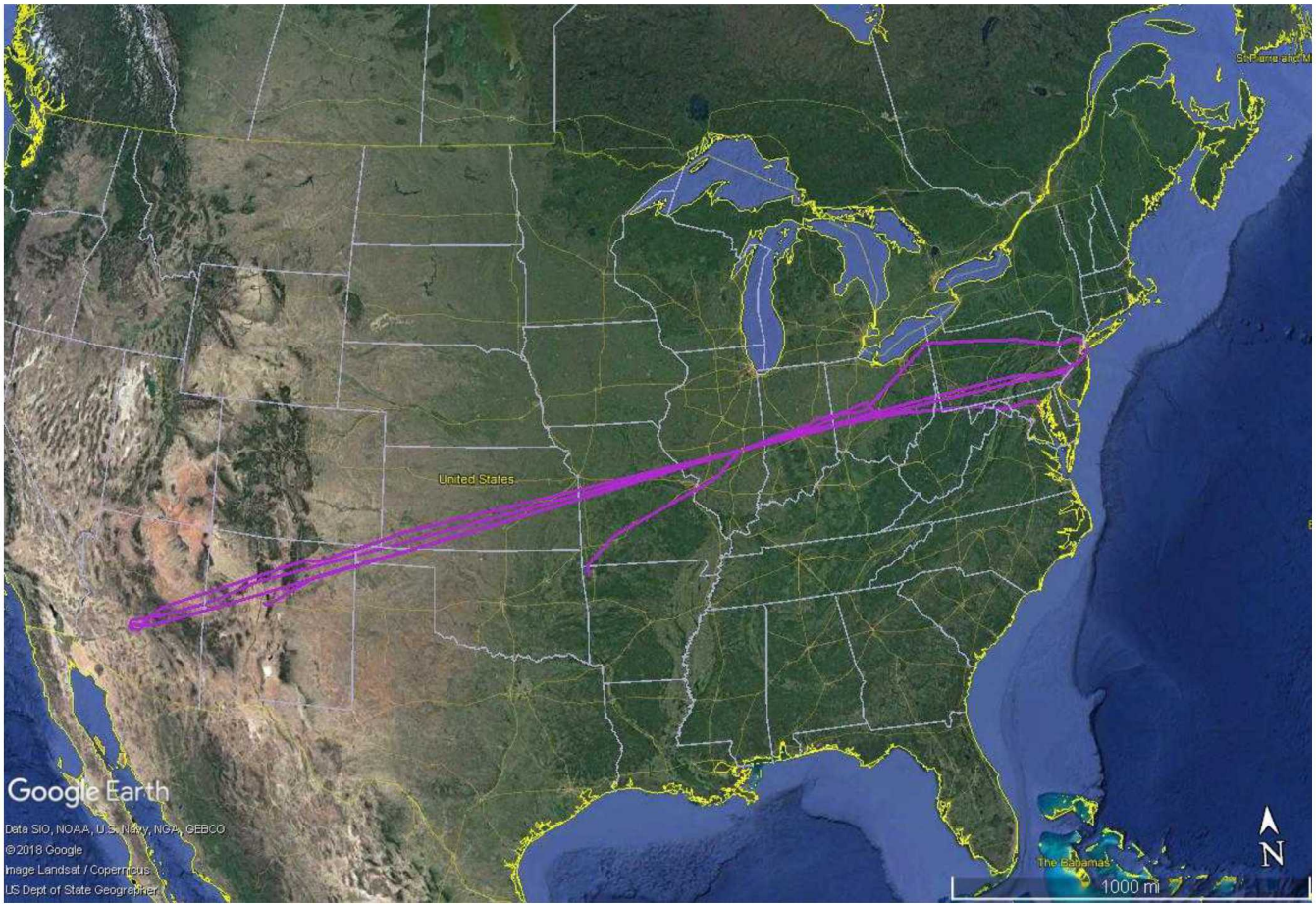
Tiny Segment near Mantoon, IL

Just 2.5 km in the middle of nowhere



Tiny Segment near Mantoon, IL

A surprising number of hits align directly with Phoenix



Summary

- Very complicated geometry problem -> informatics problem!
- Representation as a feature vector allows the rather extensive corpus of machine learning to be applied in a straightforward manner
- Have had surprising technical and practical success with this rather simple approach
- Looking forward to working with others who have other capabilities to apply

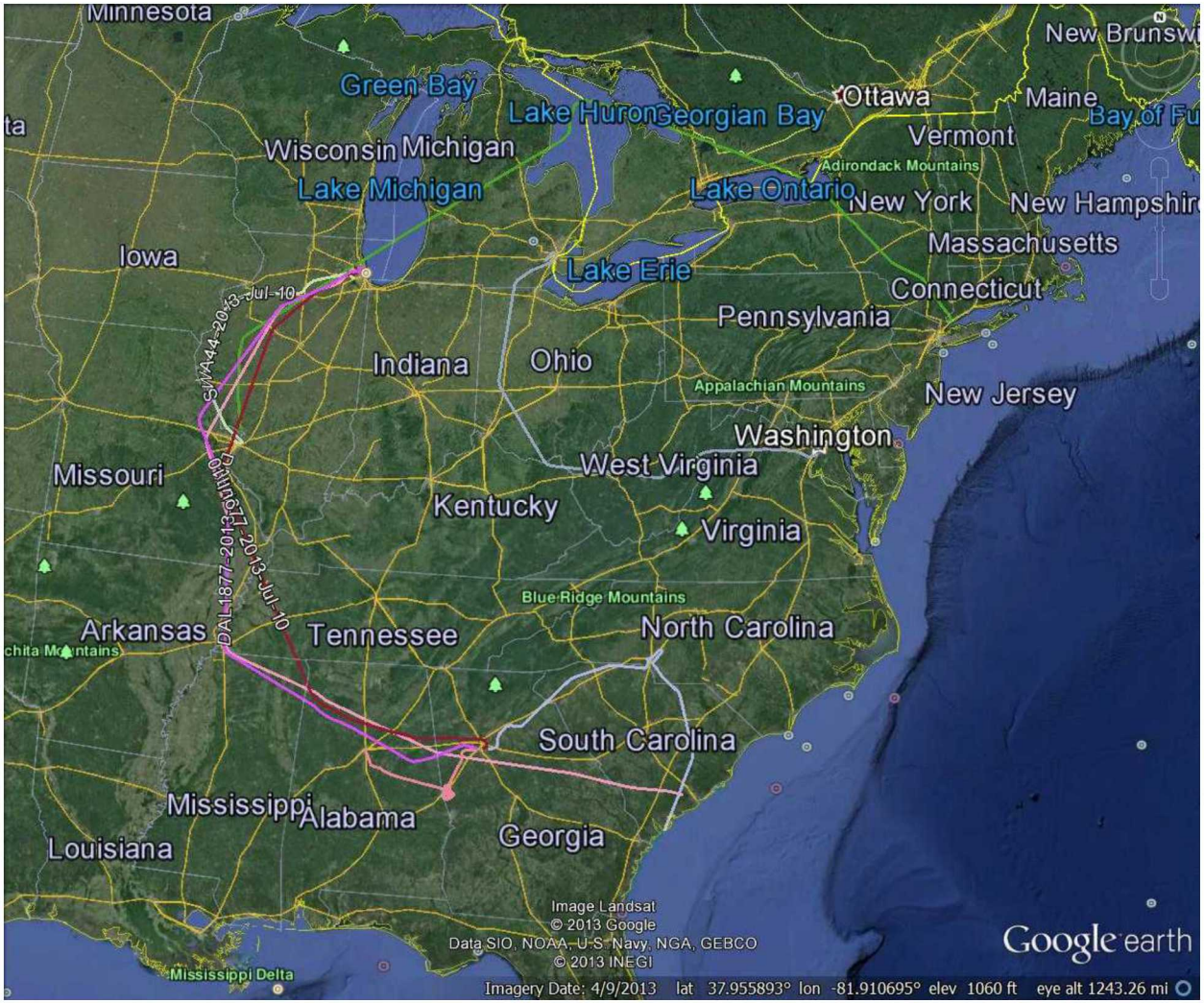
Extra Slides

Trajectory Theory Effort

- This work is connected to a similar effort in trajectory theory
  - SNL: Rintoul, Cindy Phillips
  - SUNY-SB: Prof. Joe Mitchell, Kan Huang
- Focus on:
  - Summarize information to improve understanding and guide observation
  - Develop practical algorithms/heuristics for large numbers of trajectories (>10^6)
  - Study related precise geometric problems to
    - understand limits of provable algorithms
    - guide practical algorithms
    - Impact other fields
- See poster during poster session!

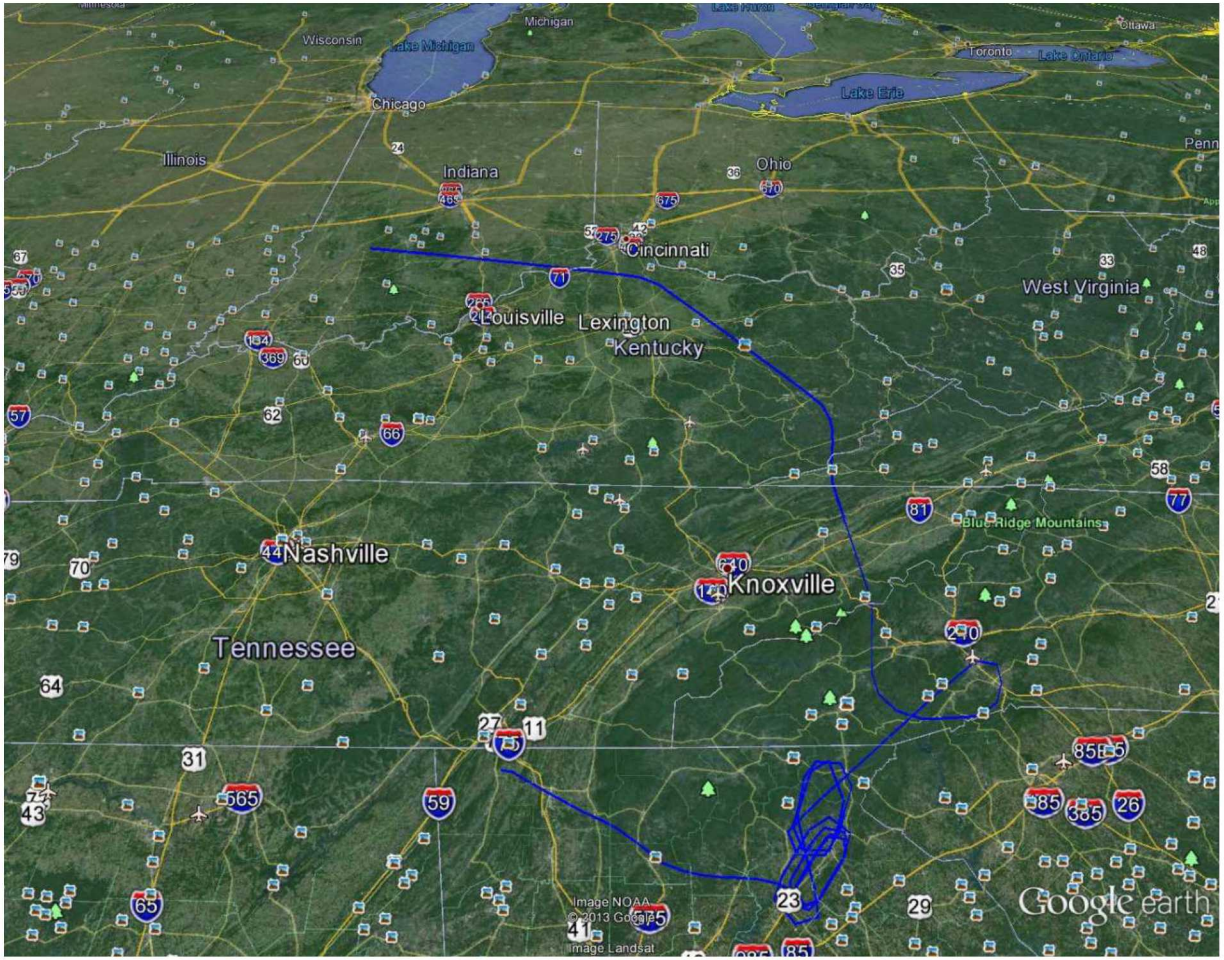
Example 1 (Avoiding Airspace)

- Flights avoiding airspace might be not quite-point-to point, but covering a wide area. Look for some that are close in location and time, and you get…



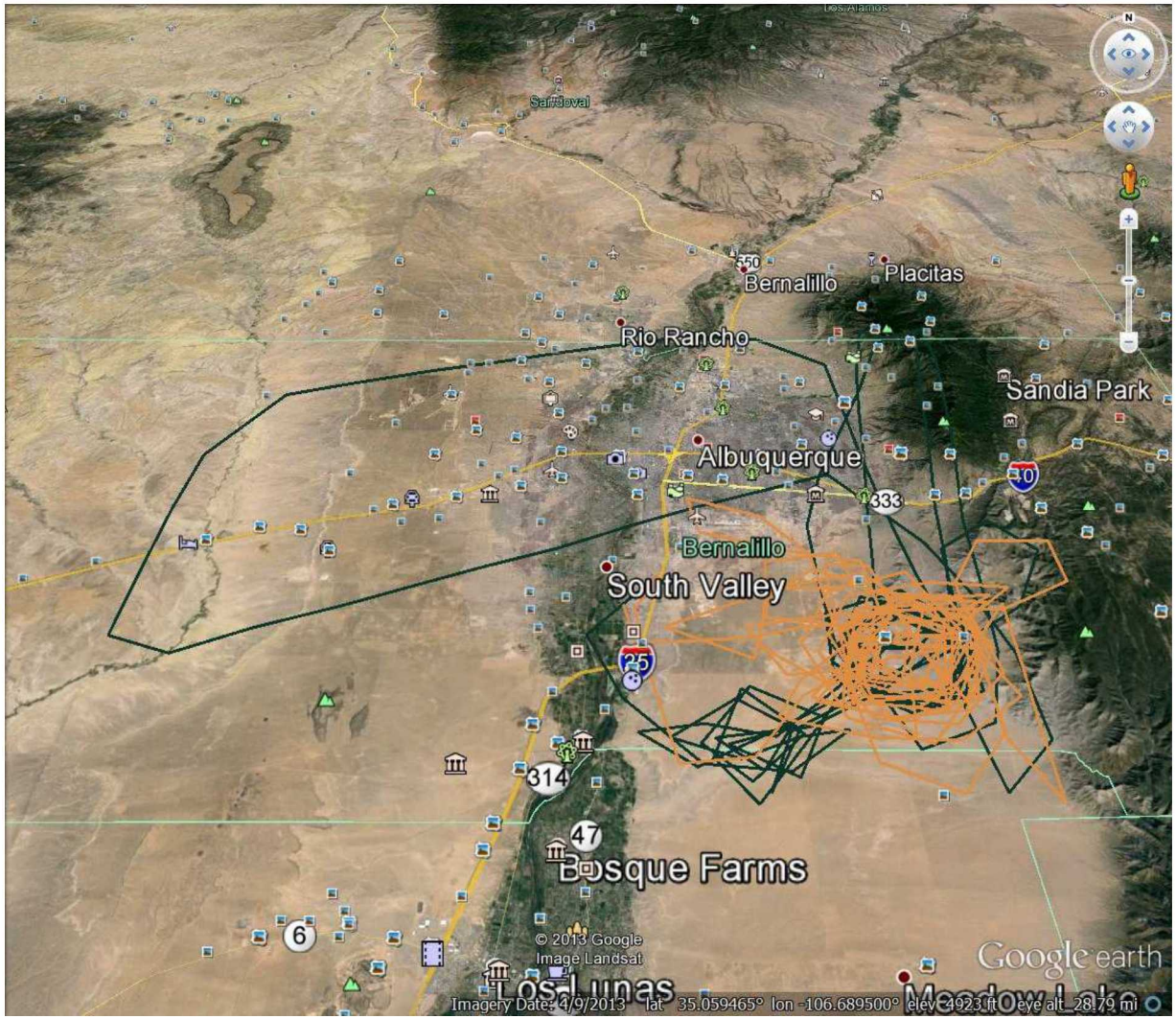
Example 2 (Holding Pattern)

- Looking for significant distance traveled, but lots of circular motion…



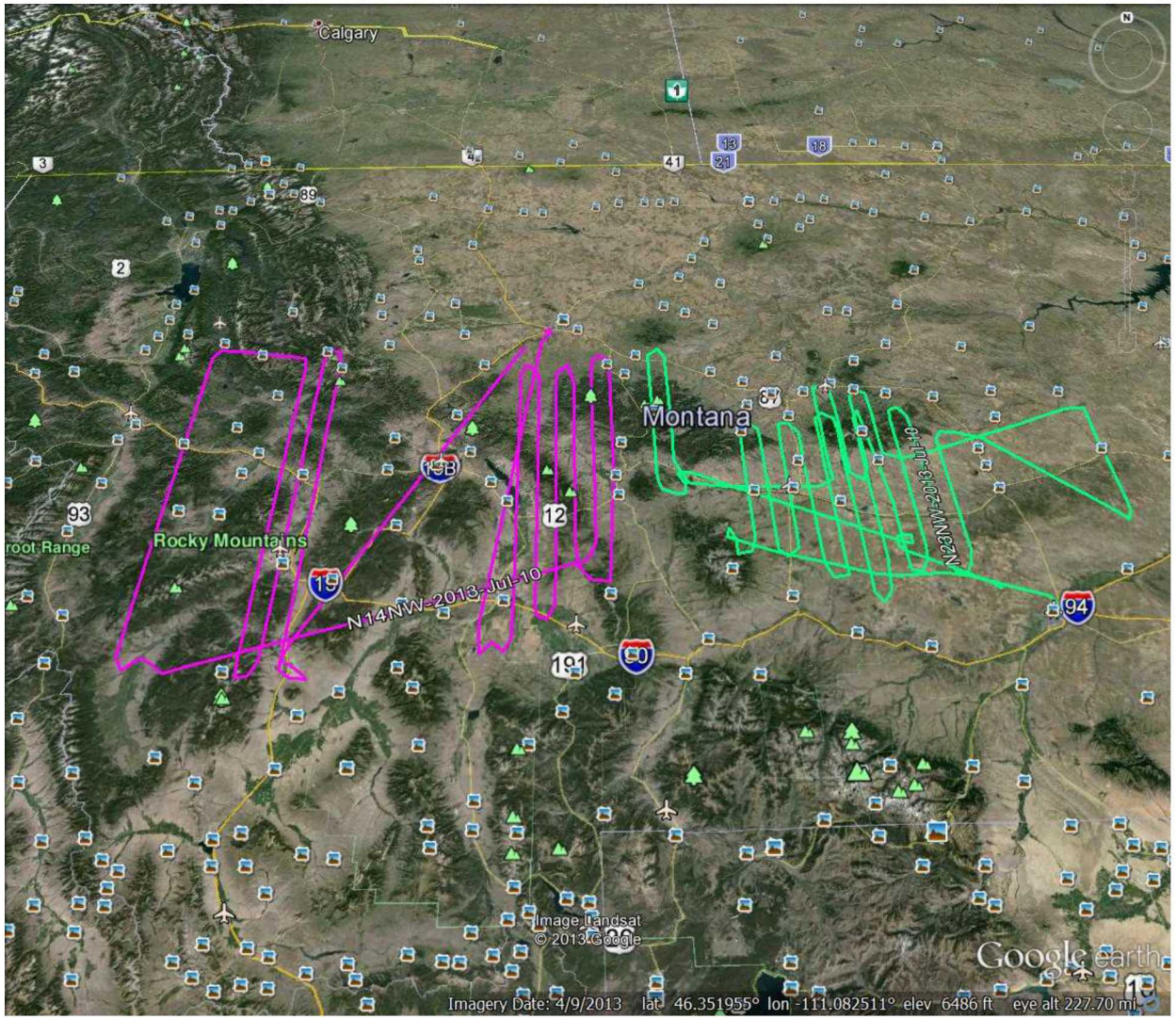
Example 3 (Scanning Area near SNL)

- Flights that aren't long point-to-point flights, but seem to be localized to an area that is centered around SNL's lat/lon…



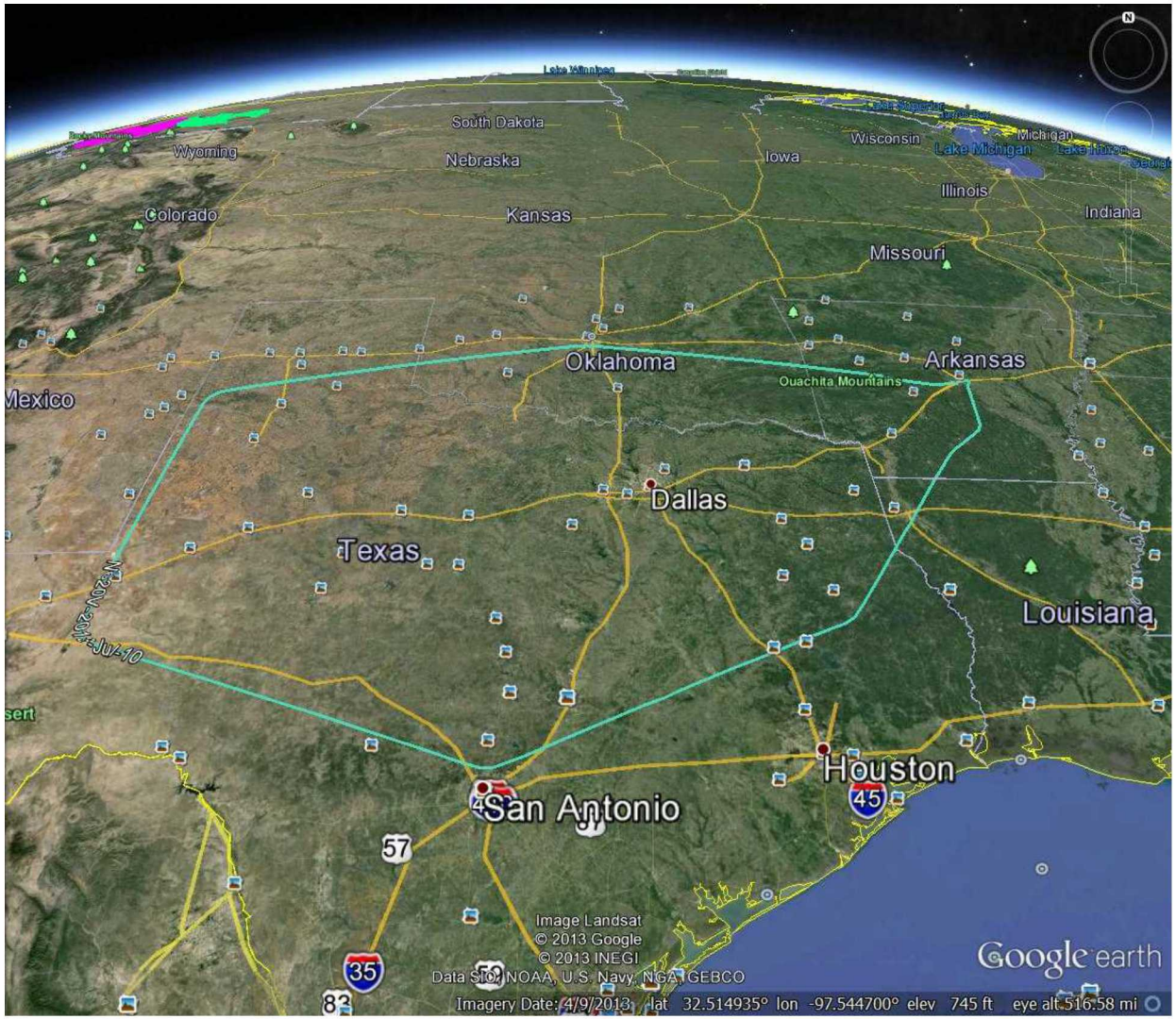
Example 4 (Search or Mapping)

- Compact, lots of turning, but not lots of winding
- The statement above is sort of a lie
- Really, they are characterized more by a small but somewhat uniaxial (roundish) convex hull for a long flight.



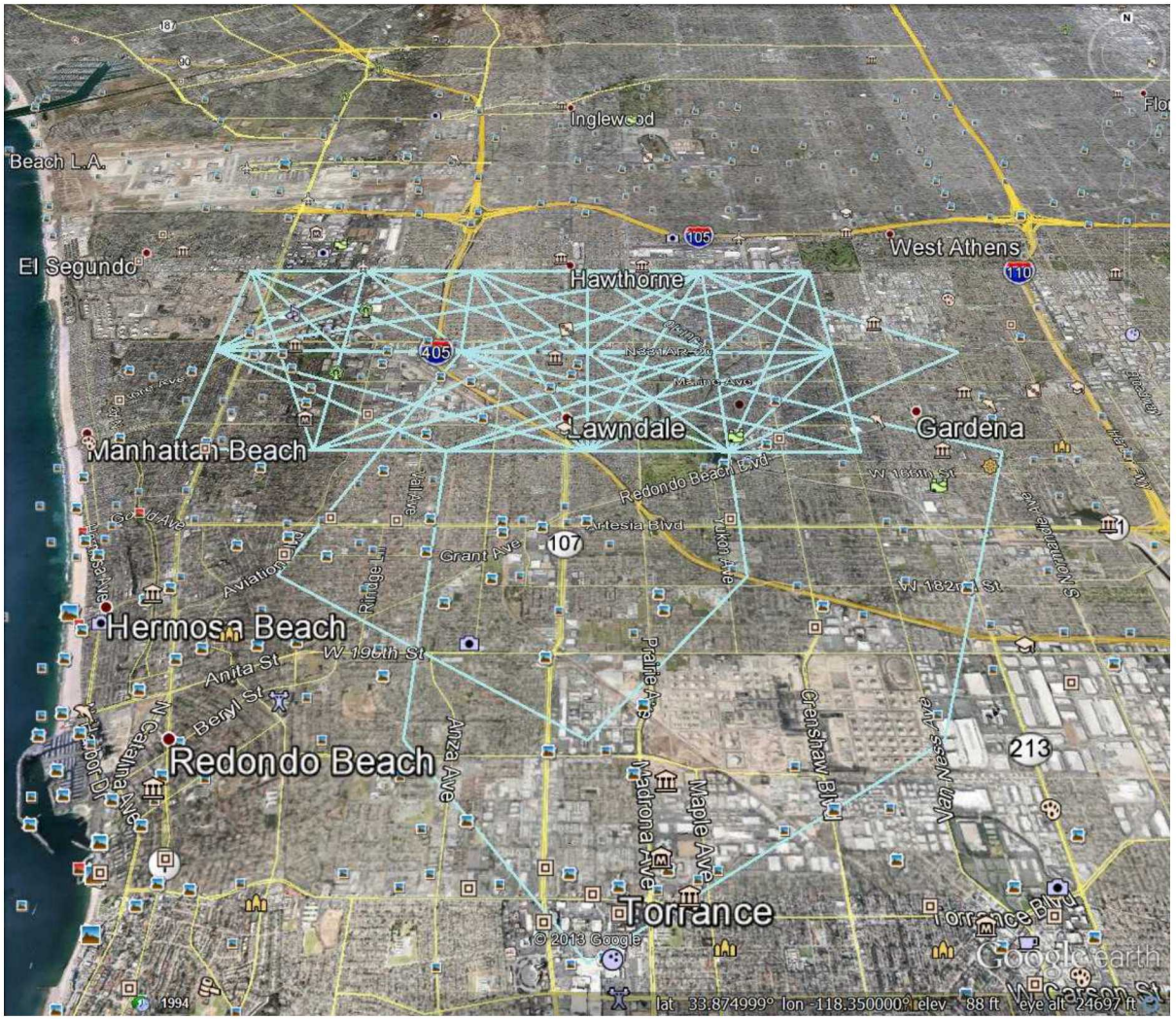
Example 5 (Big Circle)

- Long, with a relatively small eccentricity and a distance approximately equal to the perimeter



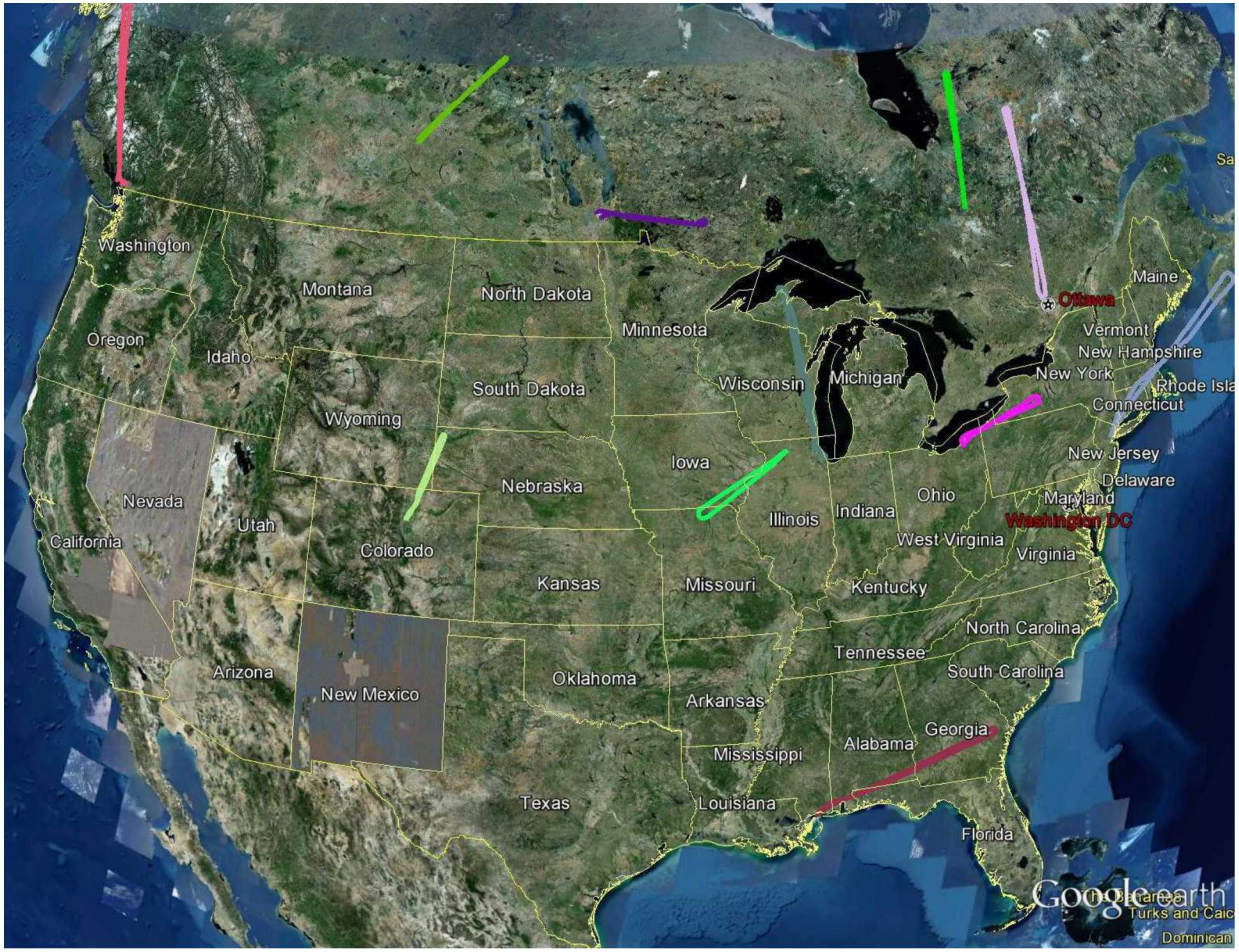
Example 6 (Compact Flight)

- Very small area to distance traveled ratio.



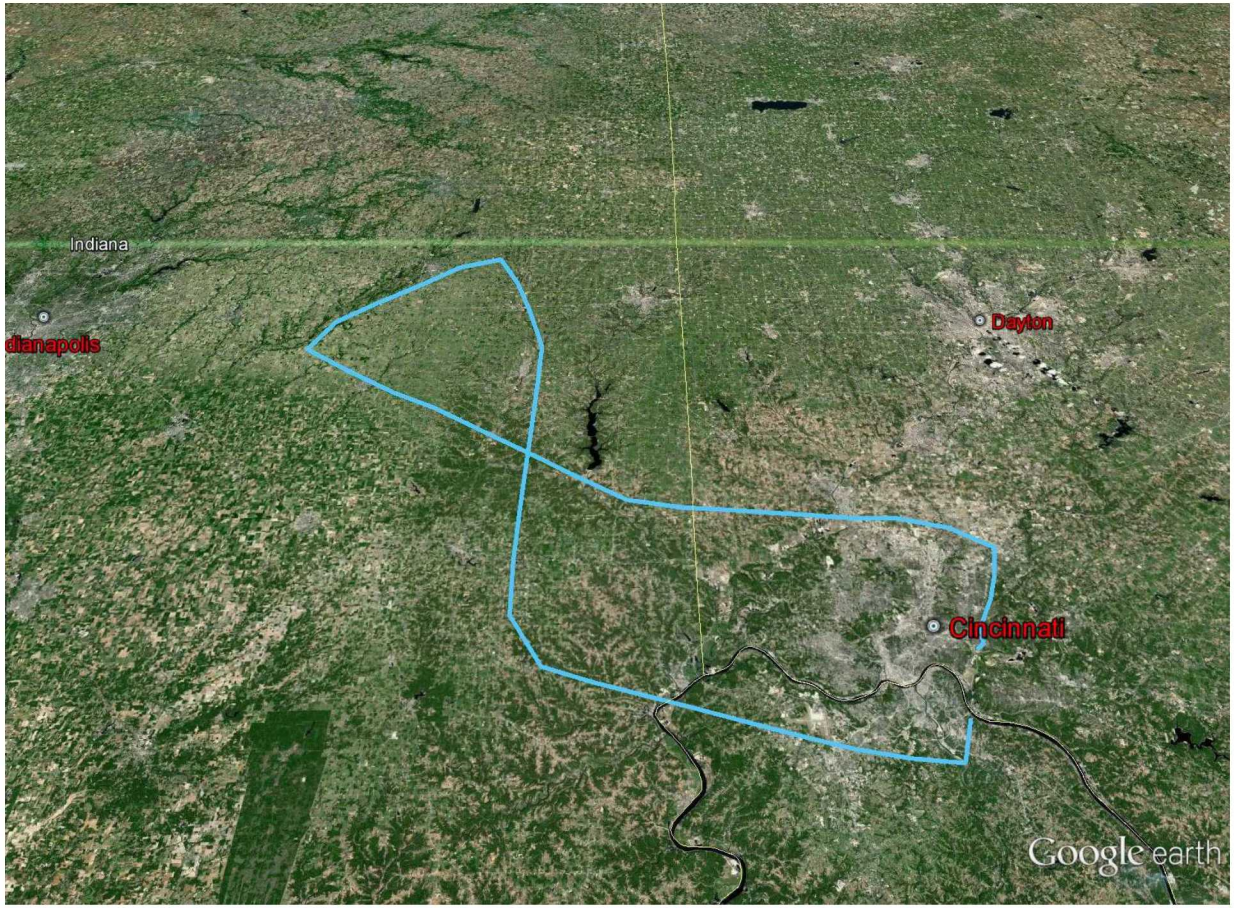
Example 7 (Forgot Something)

- Start/stop distance nearly 0
- Total flight distance long
- Very skinny shape



Example 8 (Figure 8)

- Almost no total curvature
- Significant turning
- Fairly high-aspect ratio convex hull
- Note: this was 1 of 115 flights



But what are we really doing now?

- Building up a list of descriptors about something is called a ***feature vector***
- Once you have a feature vector with *n* features in it, what you have is a set of points in an *n*-dimensional space.
- Now, you can really do some analysis
- Most importantly, if your feature vector contains the features that represent similarity in the flights, you can answer the key semantic questions:
  - Find me flights that look like this flight (nearest neighbors)
  - Find flights that are outliers (have few to no near points)
- The last one is a tough question to answer in other schemes

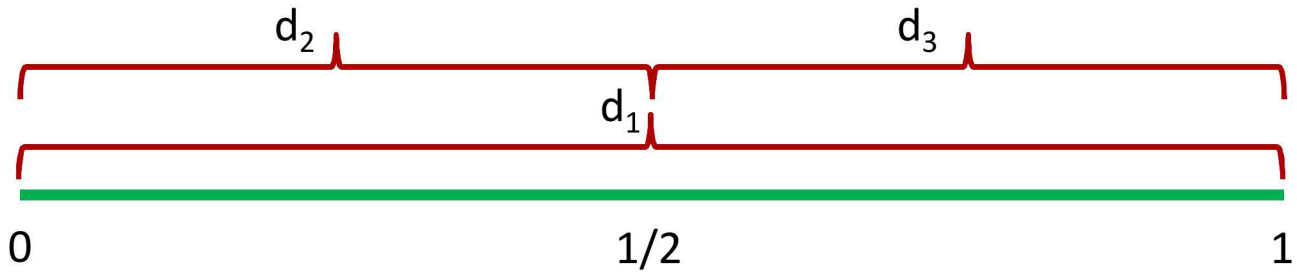
Other key advantages

- Classical clustering algorithms can be used
  - Seems obvious in the feature vector space, is extremely non-obvious without it.
  - Includes k-means, SVM's, EM, etc.
- Easy similarity (neighbor lookup) without recalculation
- Dimensionality reduction techniques can be used for visualization
- Can be implemented on a database machine
  - These machines can handle 100 TB of data (decades of flights)
  - They can also process SQL (and SQL-like) commands very quickly
- Can use techniques like PCA and similar

Note: Preliminary Implementation Completed!

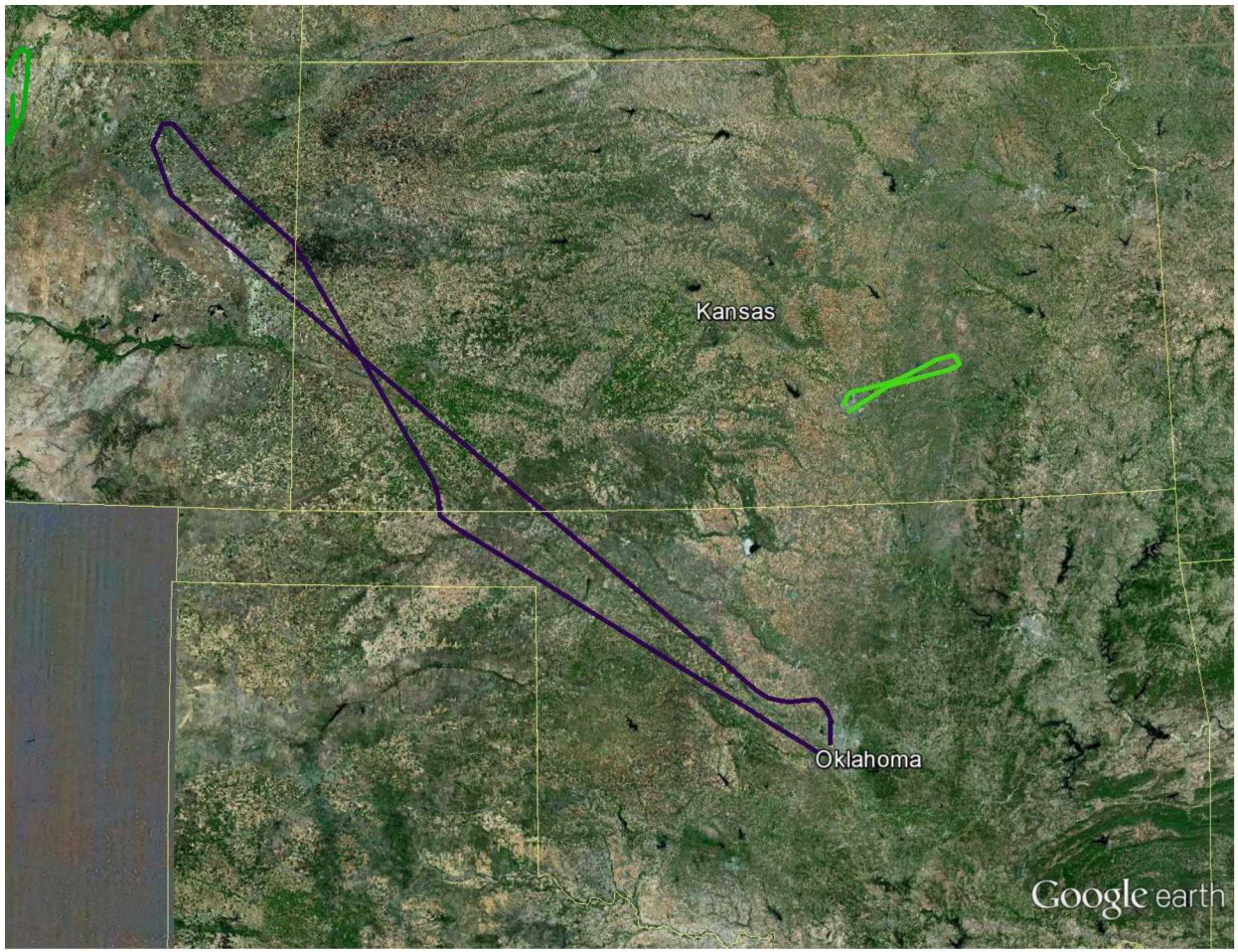
Important Discovery

- Start/Stop distance is great, so why not have other distances?
- Can use multiple intra-trajectory distances
- If you have enough, you are putting tight constraints on the shape
  - In fact, on discrete lattices you can show that you need ~N distances to exactly describe the shape (Faulon, Rintoul, Young (2002)).
  - These distances can be normalized to the largest distance in order to find shapes that are *scale invariant*!



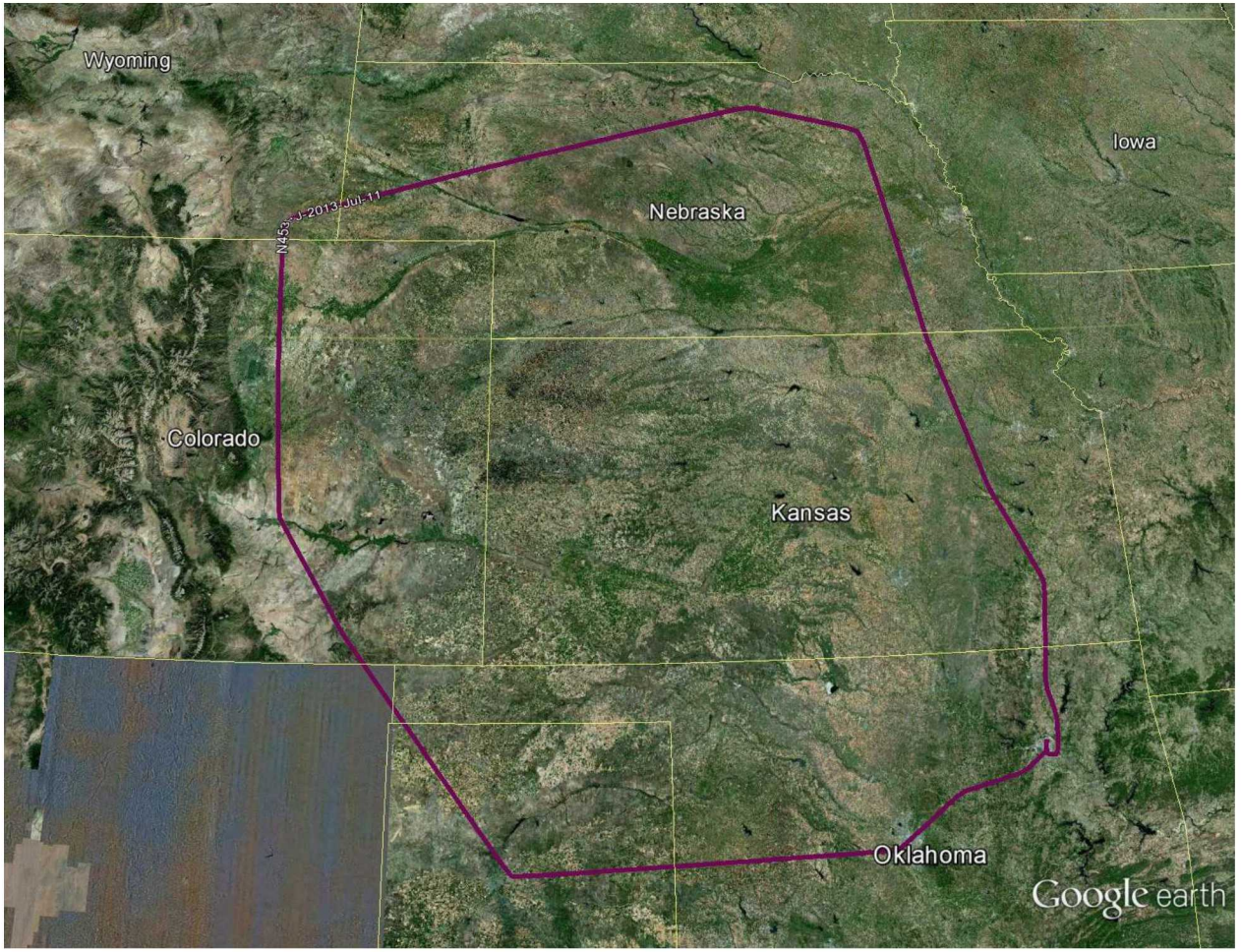
Different way to find figure-8 shape

- Let's start with the previous flight we found
- That was found from 1 day data, so expand to a week
- Use 10 control distances (full distance through all quarter distances)



Finding squares

- Can specify the distances for a square pretty easily
- A square is an interesting shape to be flying
- I did **not** know that shape was going to be there!



Big Feature Space Advantage

- Clustering, leading to *unsupervised learning* techniques
- Previous examples showed searching the space for a specific pattern or a specific volume of the feature space
- But, with clustering, the computer can group the different patterns in the feature space without knowing *a priori* what they are!
- Perhaps most importantly, many clustering algorithms specifically identify outliers in the feature space that correspond to odd behaviors

Goals for FY15 (Final Year)

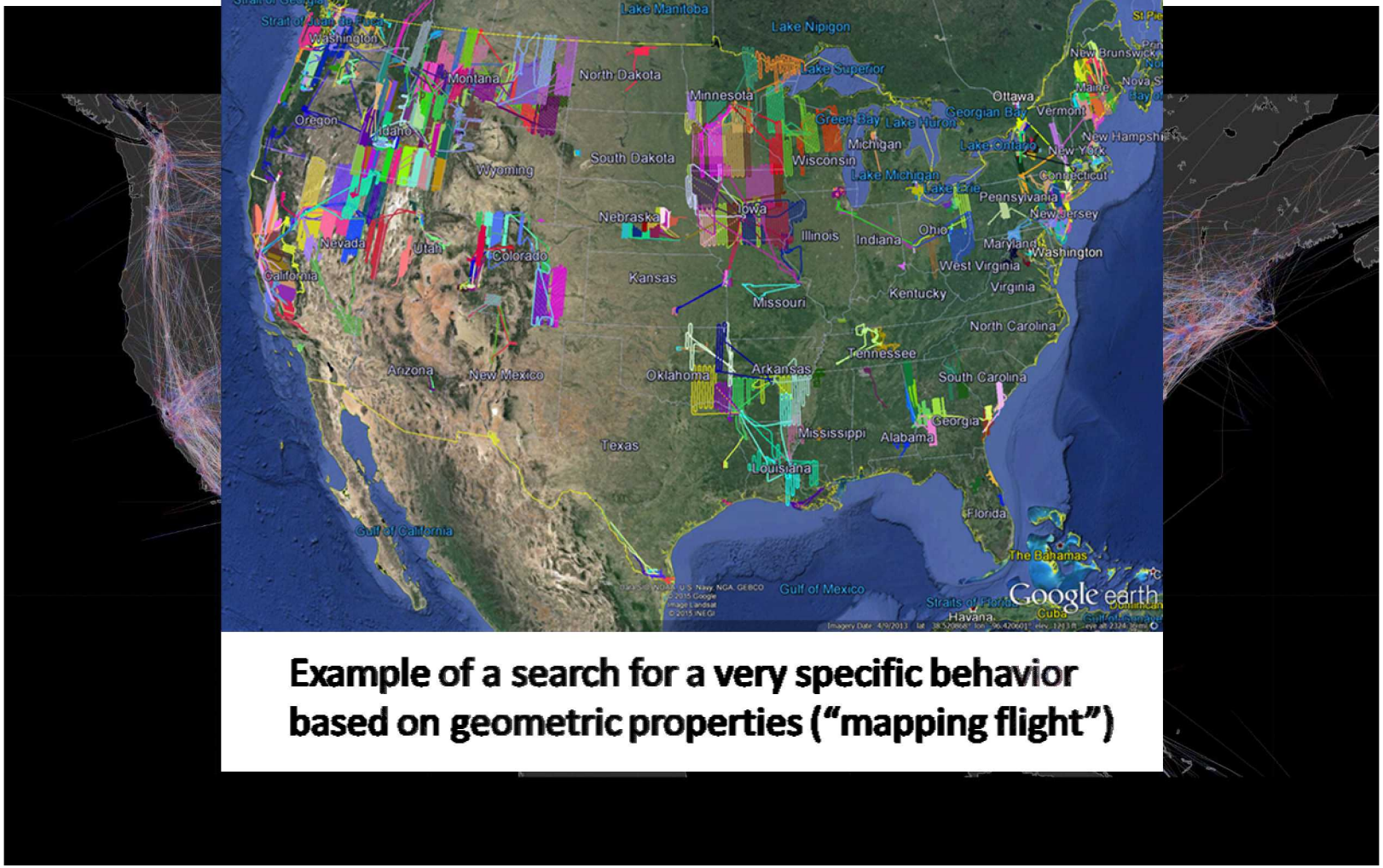
- Algorithm development
  - Incorporation of sub-trajectory analysis
  - A covariance/correlation analysis of the information in the different features
  - Develop quantitative metrics to understand the effect of sparseness and intermittency in the data
- Software Engineering:
  - Get a version 1.0 of TrackTable into the wild
  - Put a wrapper around parts of TrackTable that will enable customer usage
  - Incorporate trajectory work into GeoGraphy codebase
- Further Netezza development

Software Engineering: IBM/Netezza

- Netezza (now IBM PureData) is a data appliance designed for deep analytics on large (100's of Terabytes) data sets
- This is the machine of choice for many gov't agencies for storing the biggest data sets, including trajectories
- Andy has ported some of the trajectory assembling code to a local Netezza machine
  - Start: One **year's** worth of unassembled trajectory data (~400GB)
  - End: A list of ~15M trajectories
  - Time: 44s
  - Normally, one month takes about 3000s
- Feature vector approach potentially enables most routines to be implemented on Netezza

Software Engineering

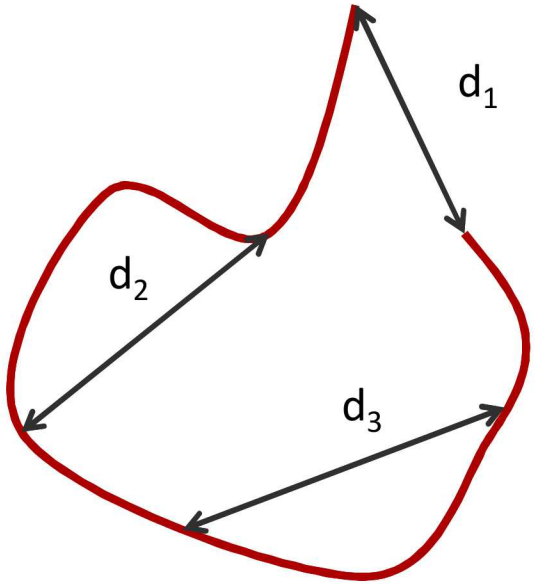
- Trajecto



Example of a search for a very specific behavior based on geometric properties ("mapping flight")

Clustering: Important Notes

- Using idea of distance geometry, we can define intra-trajectory distances as features, allowing the computer to develop its own notion of shape



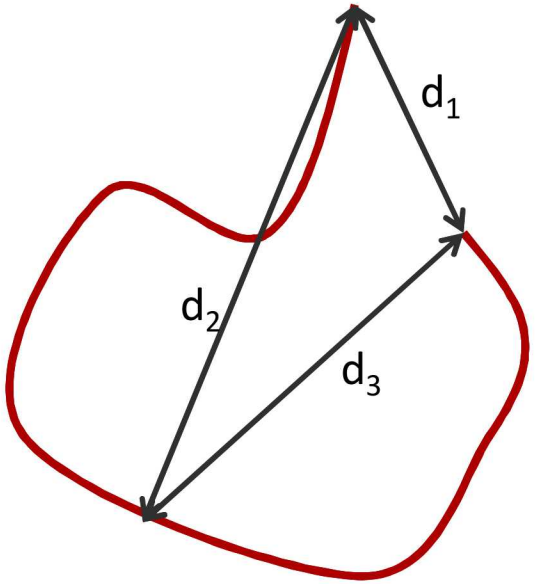
- Time (flight length, start/stop times, day of week) can also be used as a feature allowing us to find patterns in *time*, including
  - Objects travelling together
  - Trajectories that occur on a daily or weekly pattern

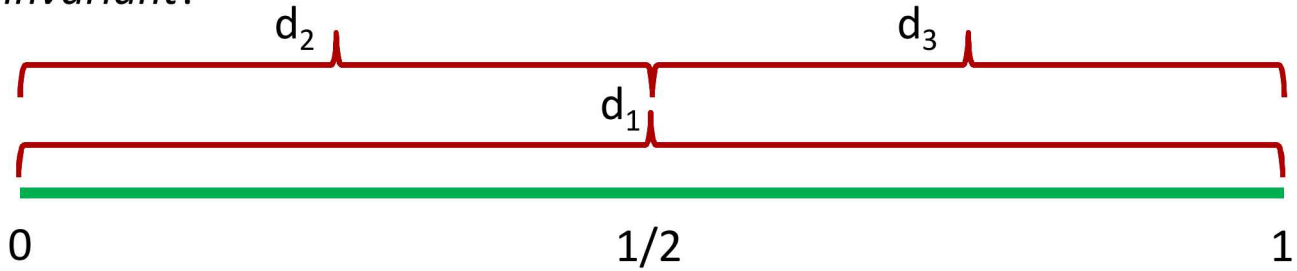
Clustering: Extended DBSCAN

- Default DBSCAN difficult to use with features that have very different scales
- Also, most spatial indexing schemes work best with the notion of a box-neighborhood (not spherical)
- We developed and proved the consistency of an extended version of DBSCAN that uses and e-box instead of an e-ball, where the box can have different sizes in each dimension.
- This allows us:
  - No need to rescale data
  - Simple control as to what constitutes "closeness" in each feature
  - A simpler and faster connection to the r-tree structure

Final Important Discovery

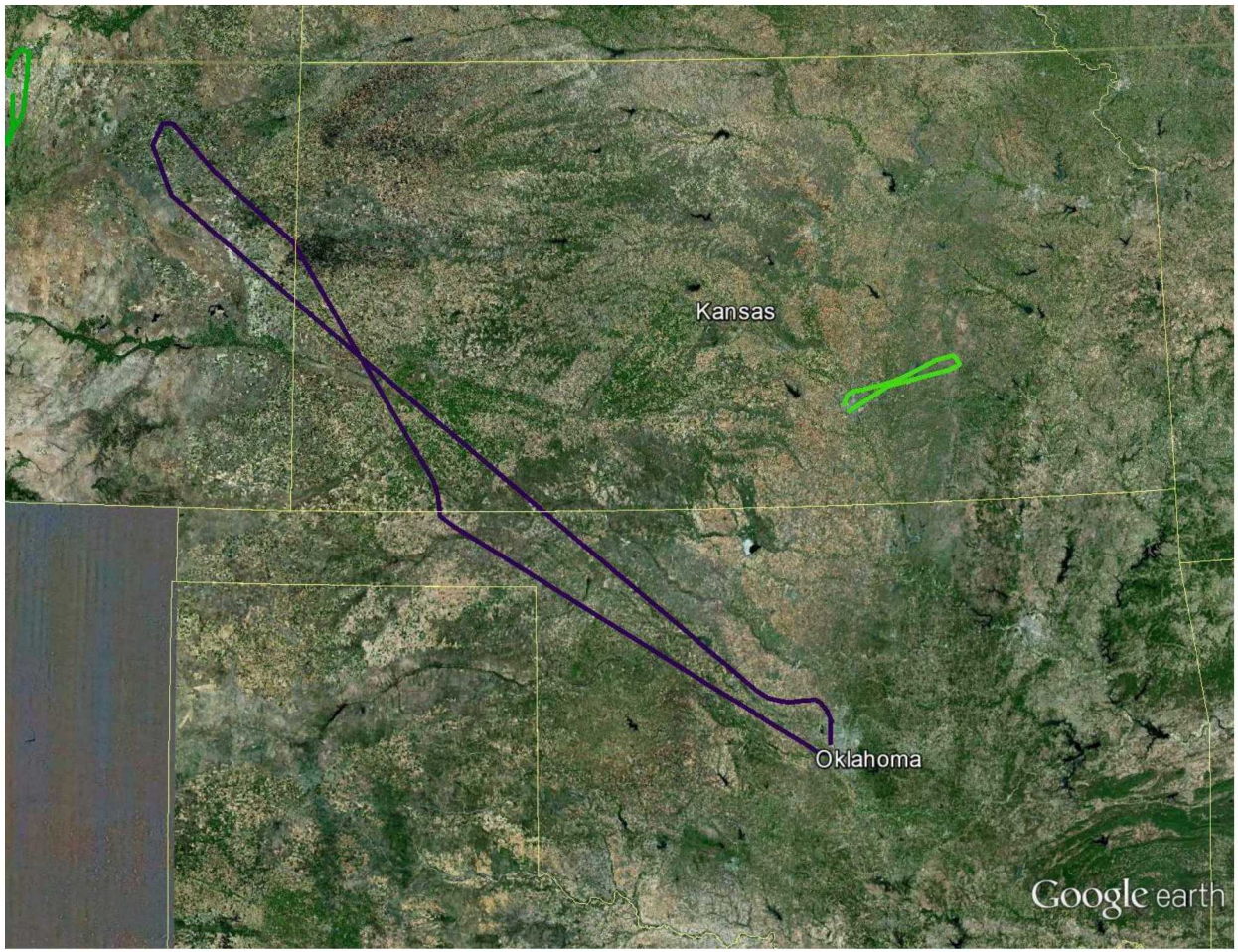
- Start/Stop distance is great, so why not have other distances?
- Can use multiple intra-trajectory distances
- If you have enough, you are putting tight constraints on the shape
  - In fact, on discrete lattices you can show that you need ~N distances to exactly describe the shape (Faulon, Rintoul, Young (2002)).
  - These distances can be normalized to the largest distance in order to find shapes that are *scale invariant*!





Different way to find figure-8 shape

- Look for shapes near the one below
- Use 10 control distances (full distance through all quarter distances)



Finding squares

- Can specify the distances for a square pretty easily
- A square is an interesting shape to be flying
- I did **not** know that shape was going to be there!