Space, Time and Groceries

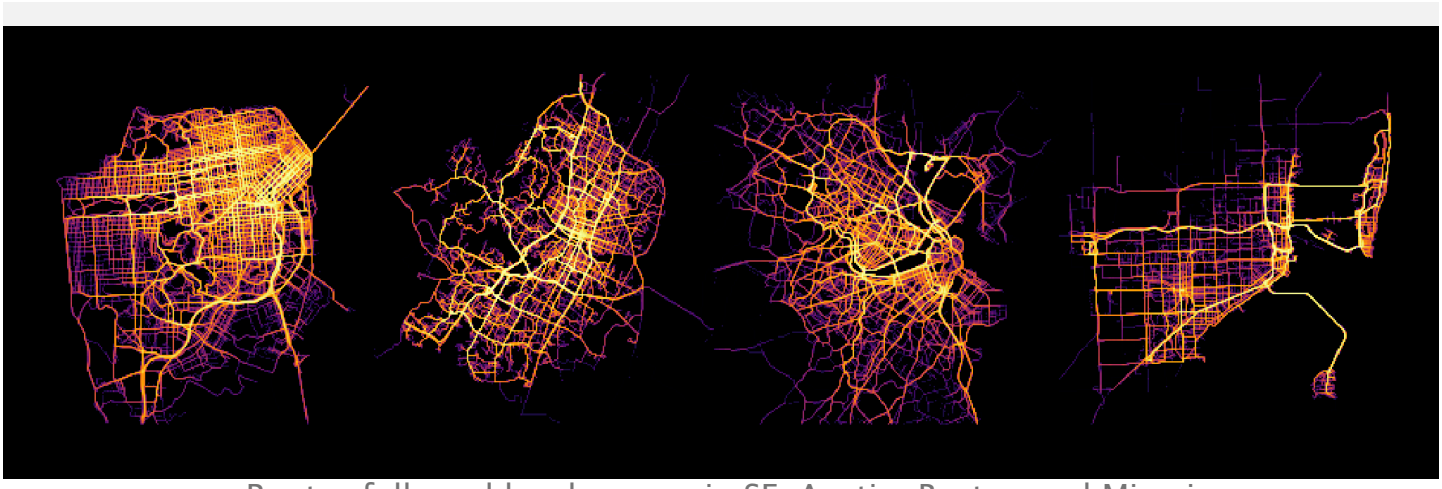
Jeremy Stanley

Jun 14, 2017 · 8 min read

Grocery delivery visualized in python with datashader.

At Instacart, we deliver a **lot** of groceries. By the end of next year, 80% of American households will be able to use Instacart. Our challenge: complete every delivery on-time, with the right groceries as fast as possible.

Over the course of a week, we traverse cities all over the United States many times over while delivering groceries:



Routes followed by shoppers in SF, Austin, Boston and Miami

How do we bring order to the chaos?

In the remainder of this post, we'll first introduce the logistics problem Instacart is solving, outline the architecture of our systems and describe the GPS data we collect. Then we will conclude by touring a series of datashader visualizations:

Example datashader visualizations at Instacart

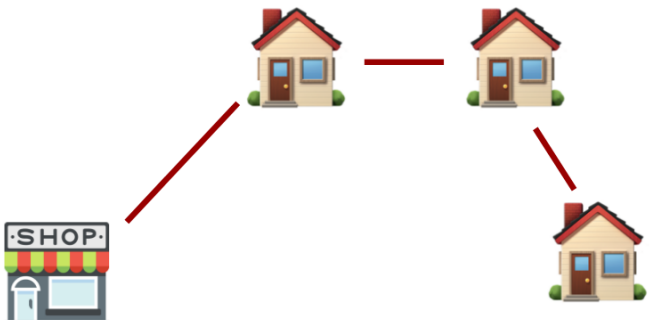
Visualizations like these help us to build intuition about our system, generate hypotheses for improvements, sanity check our changes, identify best practices and improve our operations.

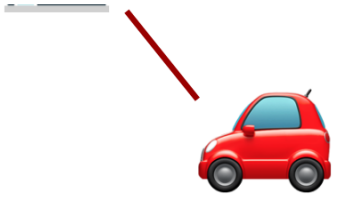
But before we get too caught up in these visualizations, let's first quickly cover the problem we are solving.

Logistics @ Instacart

When using our app to order groceries, you first choose a retailer, and then shop for groceries to be delivered. Over the course of a few hours, we have thousands of such orders to deliver. Doing this efficiently is the job of our logistics systems.

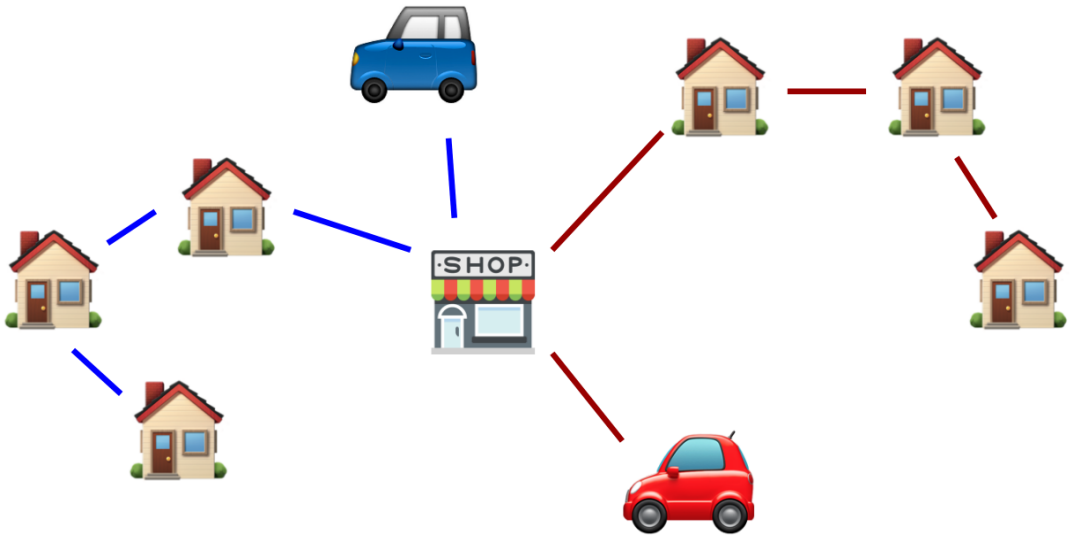
At it's simplest, our logistics problem can be viewed as solving a TSP (traveling salesman problem) where the shopper must go to the store first. For example, the shopper drives to the store, picks your groceries (along with two other orders), and then delivers them in a sequence:





There are many algorithms for solving TSPs, and perfect solutions can be found for up to tens of thousands of deliveries. Even with millions of deliveries, heuristics can come within 2–3% of the optimal solution.

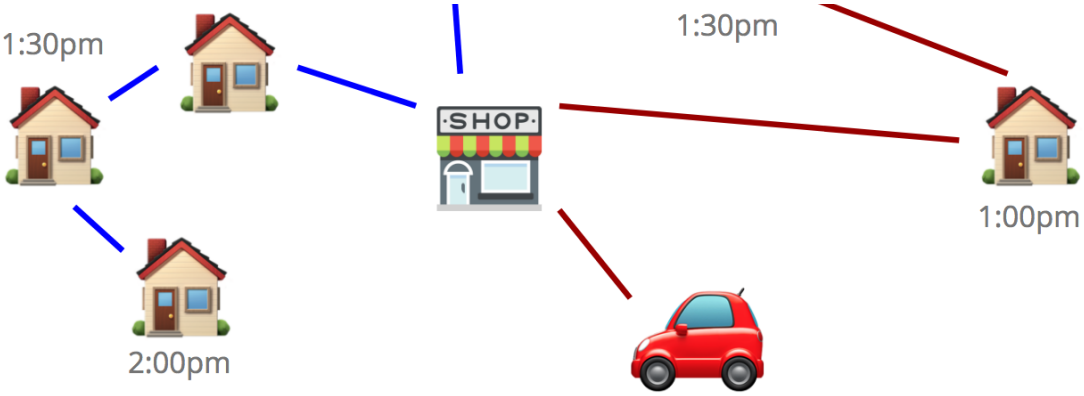
But in practice we have a fleet of shoppers to fulfill orders. Each will be given a batch of orders to shop for, and will then deliver those orders in sequence:



This problem is called a VRP (Vehicle Routing Problem), which generalizes the traveling salesman problem. (*Generalizes* is a mathy euphemism for 'even harder to solve optimally'.)

But we can't stop there. Instacart is named **Insta**cart for a reason — we commit to narrow delivery windows for our customers (usually 1 hour long). So only a subset of assigned routes will be viable, and we must jointly optimize the expected timeliness of our deliveries with the speed of our movement.

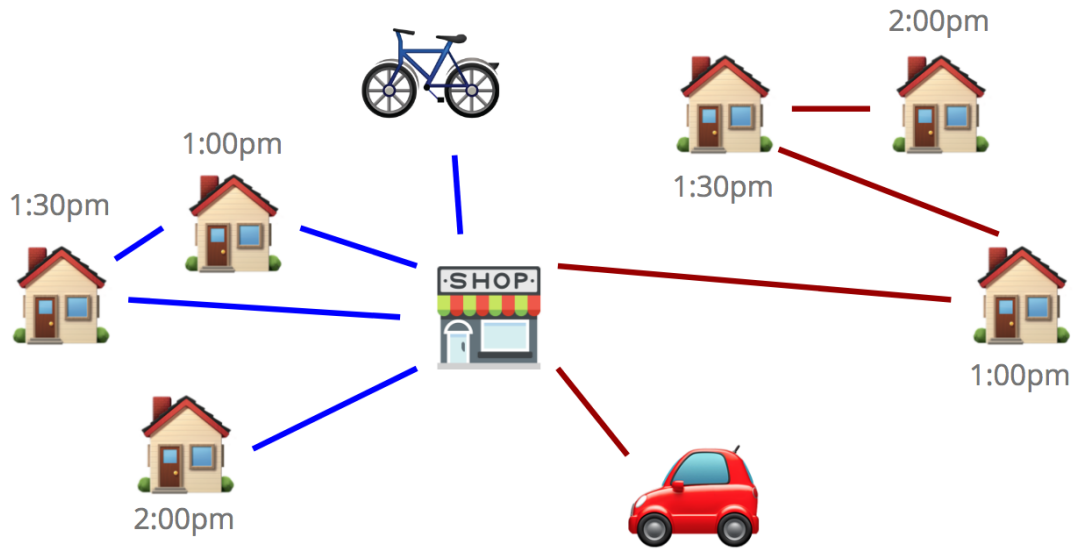
This is called a VRPTW (Vehicle Routing Problem with Time Windows):



If only life were so simple!

In reality, not all of our shoppers are equivalent. Some have large vehicles, others have small ones. Some have club-cards for retailers like Costco, while others do not. Some can fulfill alcohol orders, while others may not. This means our problem is *capacitated*, and so we can add a big C to the front of our acronym.

Furthermore, each vehicle can take more than one trip, which lets us append the letters MT (for Multiple Trips). So really, we have a CVRPTWMT:



Oh, and everything evolves continuously under many sources of uncertainty. New orders are placed. Shoppers come on and off of shift. Weather, traffic and other events wreak havoc on plans. Such problems are referred to as being *stochastic*, which gives us one more letter — an S!

So in the end, we are left to solve a SCVRPTWMT (). They say that the longer the acronym, the harder the problem is to solve.

But don’t fret, all is not lost.

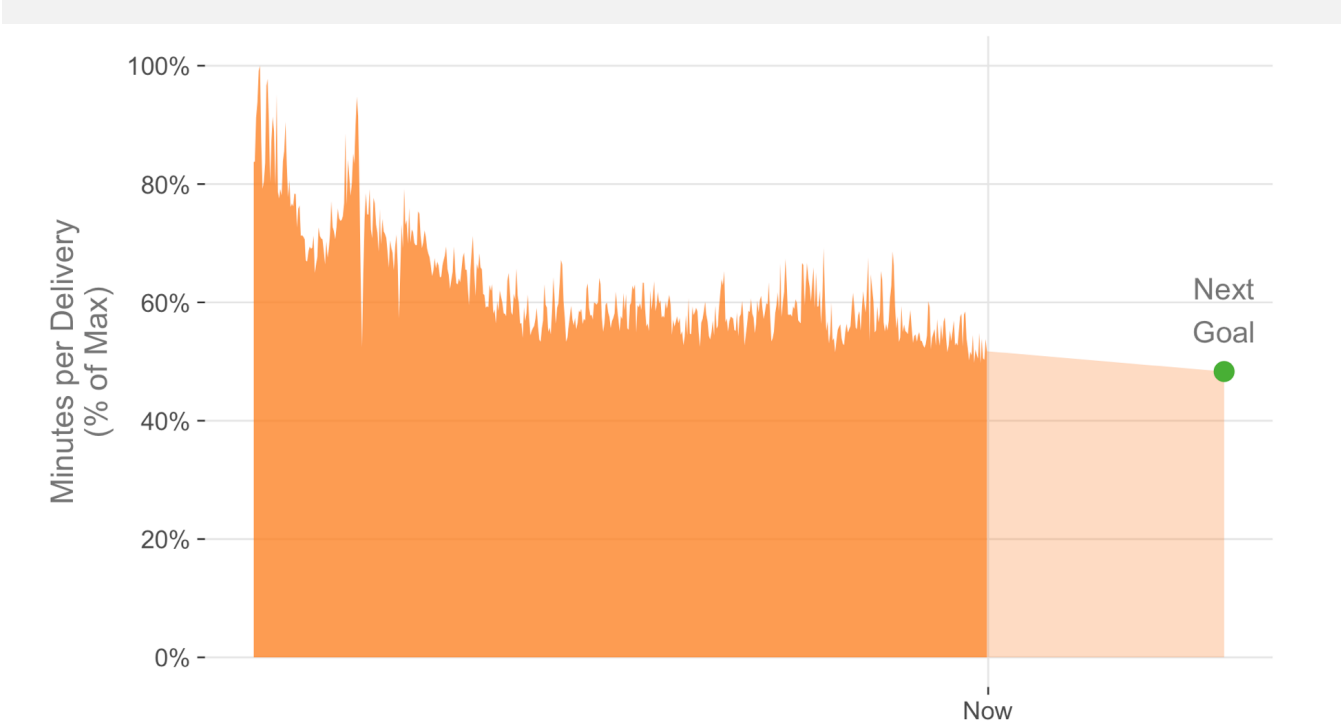
Naïve to Novel

A simple system can be implemented for routing shoppers that accomplishes some of our goals without a great deal of complexity:

1. Sort orders by when they are due
2. Find the shopper who is free that can do the first order the fastest
3. Search remaining orders for any that can be added without being late
4. Dispatch the orders found to this shopper
5. Repeat

This will optimize for fulfilling the most urgent orders in the most timely fashion, and seek efficiencies where possible as a secondary objective.

We began with a variation of this kind of simple greedy algorithm, and have since introduced novel approaches that have had a dramatic impact on our speed, without compromising on late deliveries or order quality:

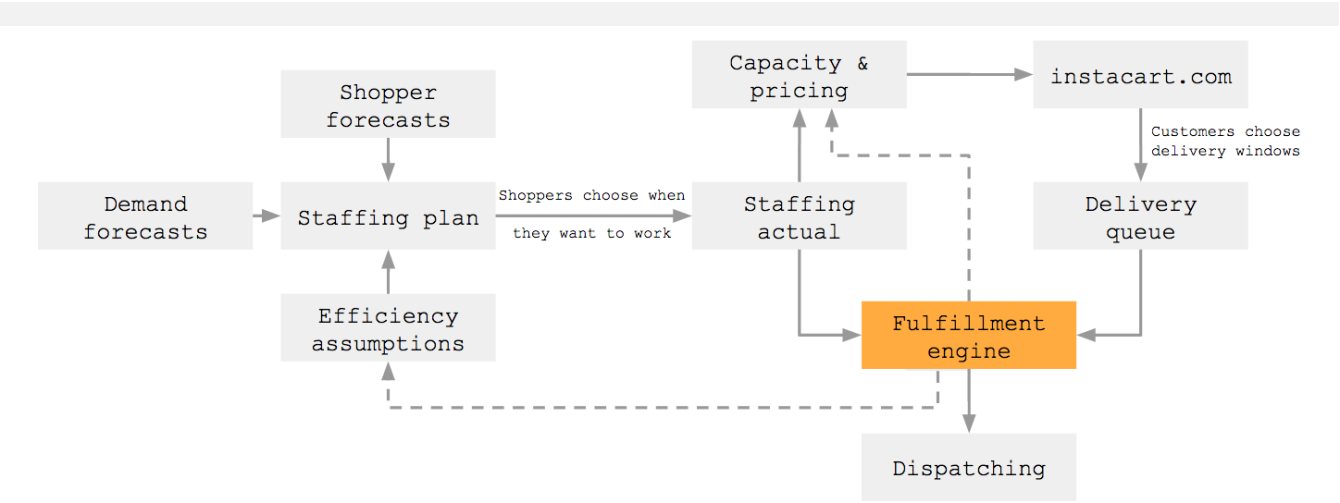


We have halved the number of minutes per delivery in San Francisco, and continue to set aggressive goals.

Some of the changes we have introduced include:

- Machine learning to predict the distribution of time expected for any given shopper and assignment
- Decomposing the CVRPTW into sub-problems (clustering deliveries, shopper assignment) and solving these sub-problems to near optimality
- Applying heuristics for limiting search spaces, dealing with anomalies, fine-tuning solutions and adapting under uncertainty
- Re-computing batch plans every minute and making dispatching decisions just in time

The application that decides what orders each shopper should fulfill is called our 'fulfillment engine', and it is just one component of our overall logistics system, which also forecasts demand and shopper behavior, manages capacity and busy pricing and plans and adapts our staffing:



These systems are highly interdependent, and we are increasingly using simulations to optimize them jointly under many sources of uncertainty.

In the months to come we will publish more detailed posts about these systems and the fun engineering, machine learning, optimization and operations challenges they present, so stay tuned!

The Data

In order to optimize the assignment and routing of our shoppers, and to communicate effectively with our consumers, we collect a stream of GPS location

data.

For example, these are what ten updates might look like for a fictional shopper:

shopper_id	measured_at	latitude	longitude	speed	direction	accuracy
124567	2017-05-01 13:35:59	37.7749	-122.4194	0.00	-1.00	10.0
124567	2017-05-01 13:36:09	37.7749	-122.4194	0.44	72.07	10.0
124567	2017-05-01 13:36:19	37.7749	-122.4193	1.09	241.17	17.8
124567	2017-05-01 13:36:29	37.7749	-122.4194	1.37	36.91	30.0
124567	2017-05-01 13:36:34	37.7750	-122.4194	0.43	18.98	10.0
124567	2017-05-01 13:36:49	37.7749	-122.4192	0.23	40.08	10.0
124567	2017-05-01 13:36:59	37.7750	-122.4192	0.22	215.86	10.0
124567	2017-05-01 13:37:09	37.7750	-122.4193	0.00	285.47	10.0
124567	2017-05-01 13:37:19	37.7750	-122.4193	0.54	108.28	10.0
124567	2017-05-01 13:37:29	37.7750	-122.4193	0.23	336.45	10.0

Every ~10 seconds, we collect the timestamp, latitude and longitude, speed, direction and accuracy reported by the device. The latitude and longitude are shown here rounded to 4 digits, but are collected to 6 digits in production. The speed is measured in miles per hour (this fictional shopper might be walking to their car). The direction is in degrees, and is -1 when a shopper is at a halt. The accuracy is in meters, and indicates the expected error of the measurement from the real position.

Over the course of a single day, we collect 10s of millions of these updates across the country.

Datashader

Datashader provides the ability to quickly and interactively visualize millions, or even billions of points.

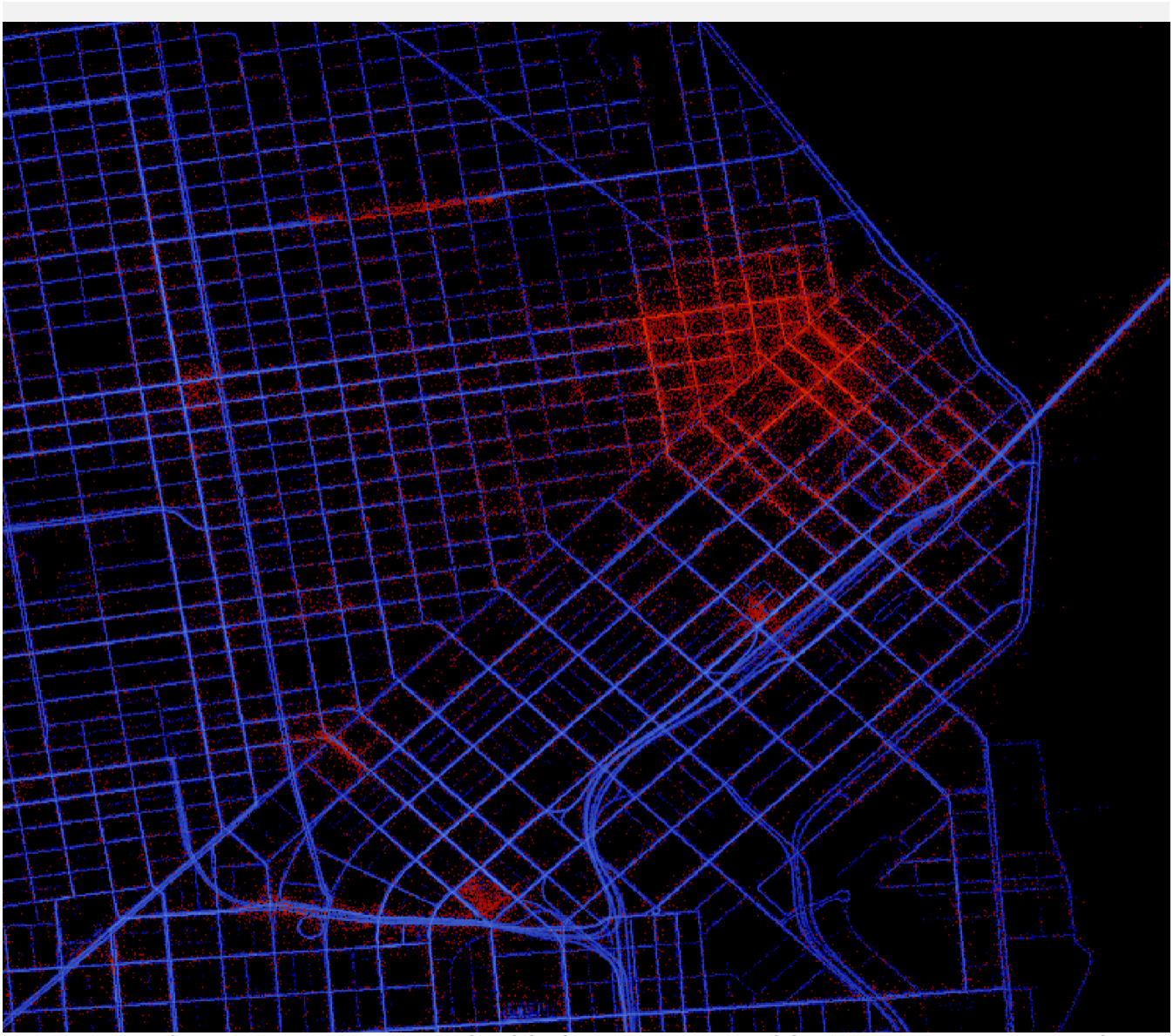
An animation of interactively zooming into a SF datashader plot

For more information on datashader, I recommend you start with their plotting pitfalls notebook. Many of the visualizations in this post are modeled after their NYC Taxi and OpenSky notebooks.

Note that for these visualizations, we show only data points where shoppers are moving quickly while driving and delivering groceries, or we are zoomed into store locations. This is to protect the privacy of our shoppers and our customers.

Accuracy

First, let's inspect the accuracy of the GPS data we collect:

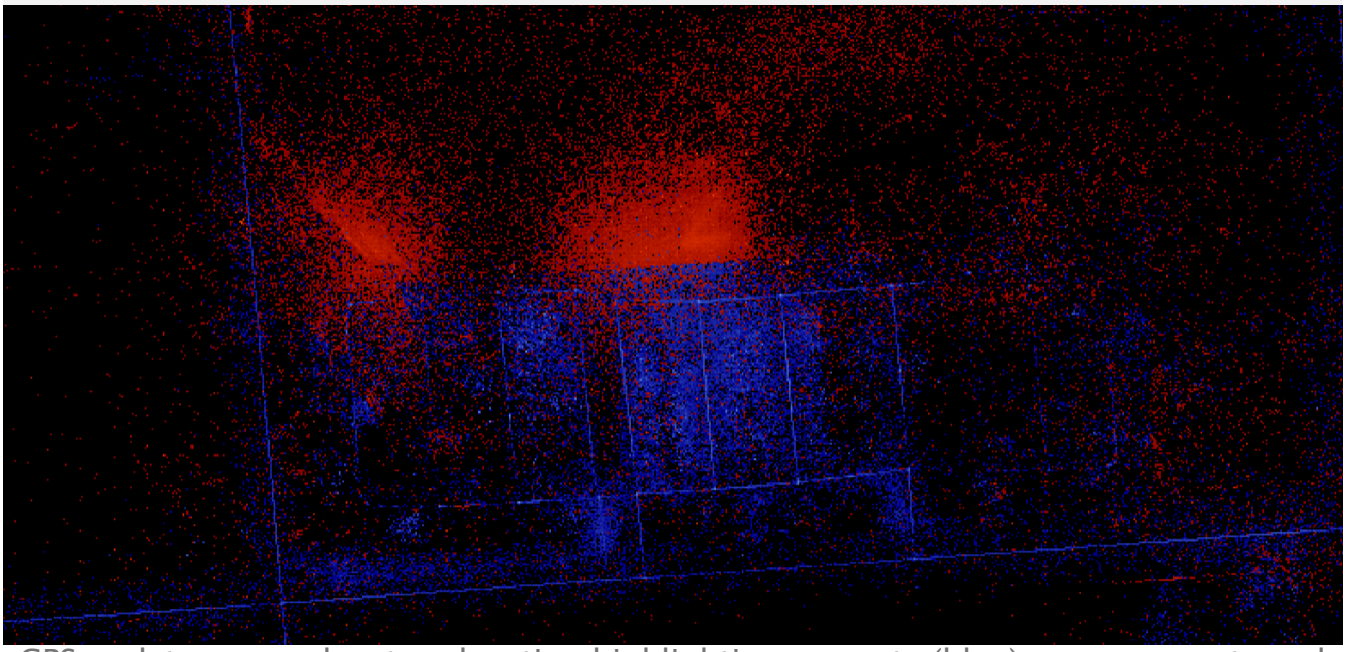


GPS updates in San Francisco highlighting accurate (blue) measurements and inaccurate (red) measurements

The red points represent inaccurate measurements (more than 10 meters), whereas those in blue are accurate measurements (10 meters or less). We can immediately

see that accuracy is poorer in the financial district (upper-right), where tall buildings obstruct the GPS. But even there the data accumulates to clearly show an outline of the streets. The measurements are also less accurate inside of any city block, where presumably the shoppers are indoors and the GPS signal is obstructed.

We can zoom into one of our store locations and see the shoppers moving through the parking lot with highly accurate GPS locations, but losing that signal within the stores themselves:

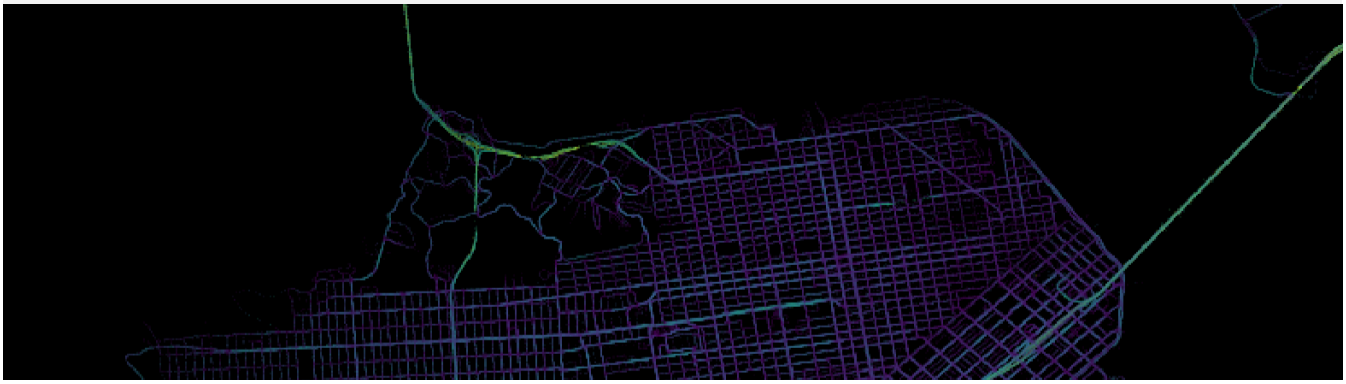


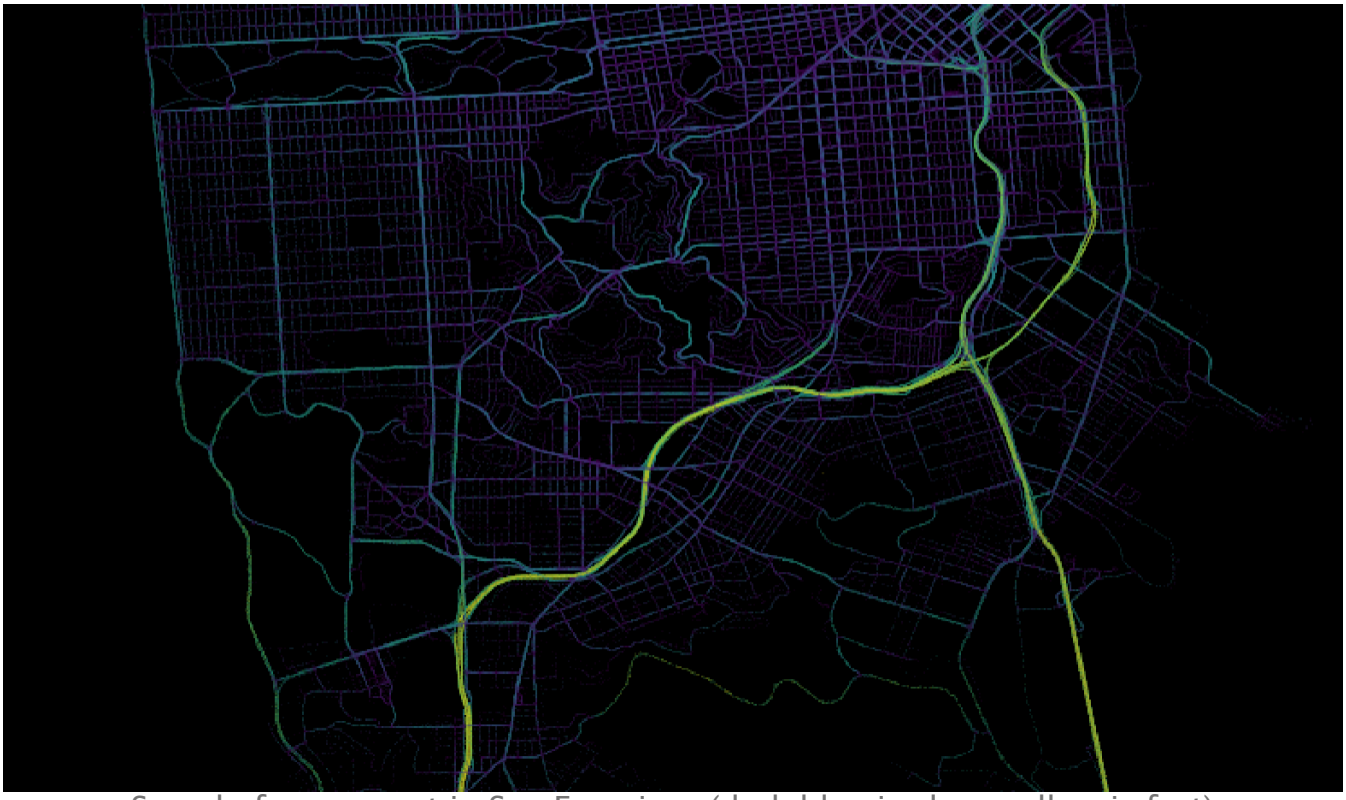
GPS updates around a store location highlighting accurate (blue) measurements and inaccurate (red) measurements

Furthermore, there appear to be buildings or other obstructions that 'shade' the GPS accuracy over certain parts of this parking lot (see the left hand side).

Speed

We can filter the the data to just the accurate observations, and then color the observations based on the speed the shopper is moving at:



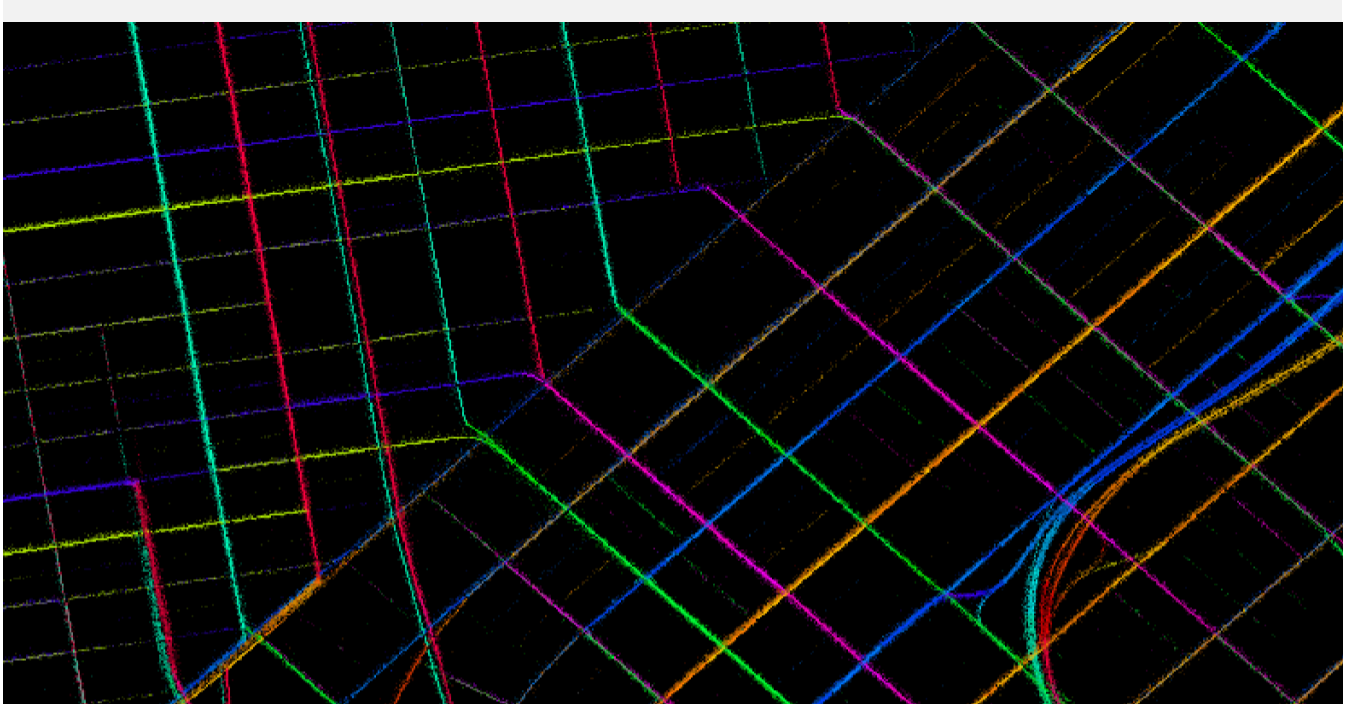


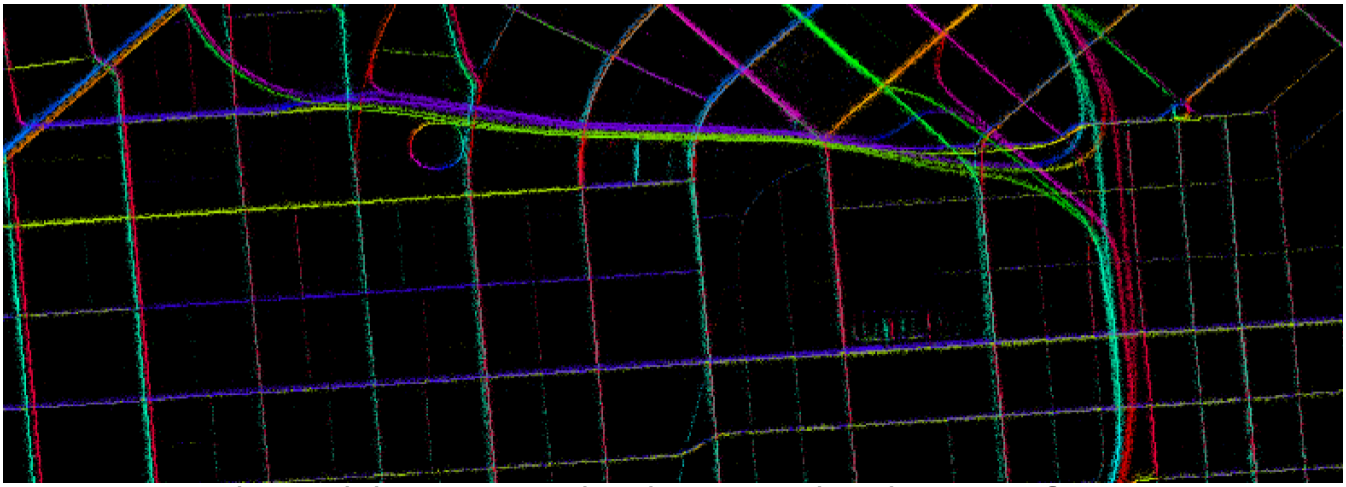
Speed of movement in San Francisco (dark blue is slow, yellow is fast)

This clearly shows our shoppers moving fastest on the highways in San Francisco, and slowest in the financial district. It also shows that moving quickly from the south to the north side of the city is difficult, as there are no fast routes making that connection.

Direction

If we instead color each point by the direction the shopper was moving in, we can clearly see the organization of the city streets:



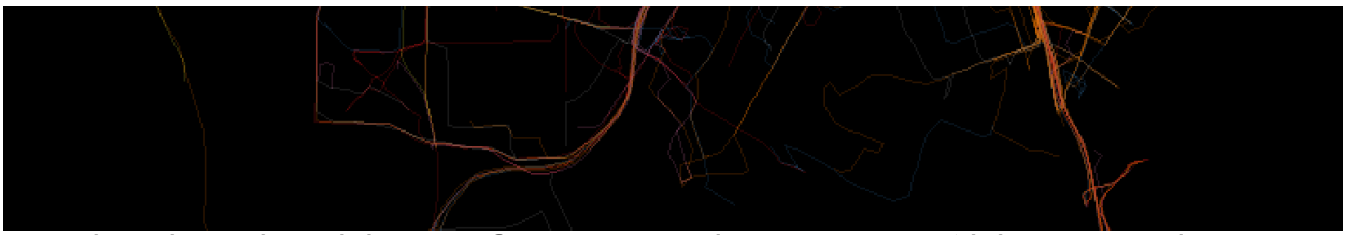


GPS updates while moving with color mapped to direction of movement

One way roads alternate from one block to the next, some roads have traffic moving both ways, and other roads switch directions at certain intersections. The roundabouts and circular exit and entrance ramps make nice color wheels.

Store Location

When shoppers are delivering groceries, we know the store location they originated from, and so can color the map to visualize what stores frequently deliver to each neighborhood:

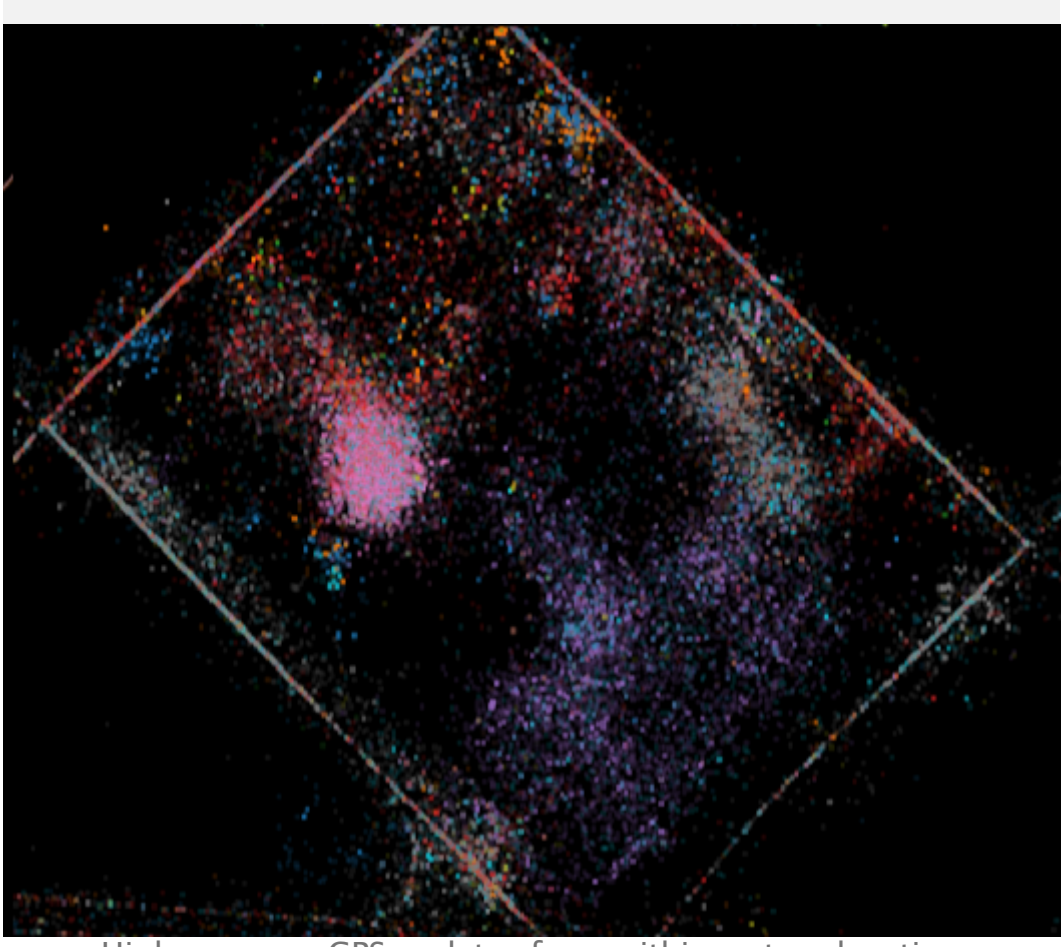


Paths taken when delivering from 10 store locations in SF (deliveries per location sampled to be constant)

Some stores dominate large areas, especially on the edge of the city. In more congested neighborhoods many streets are frequently traversed by shoppers from multiple stores, and the colors blend together into mixed hues.

Shopper State

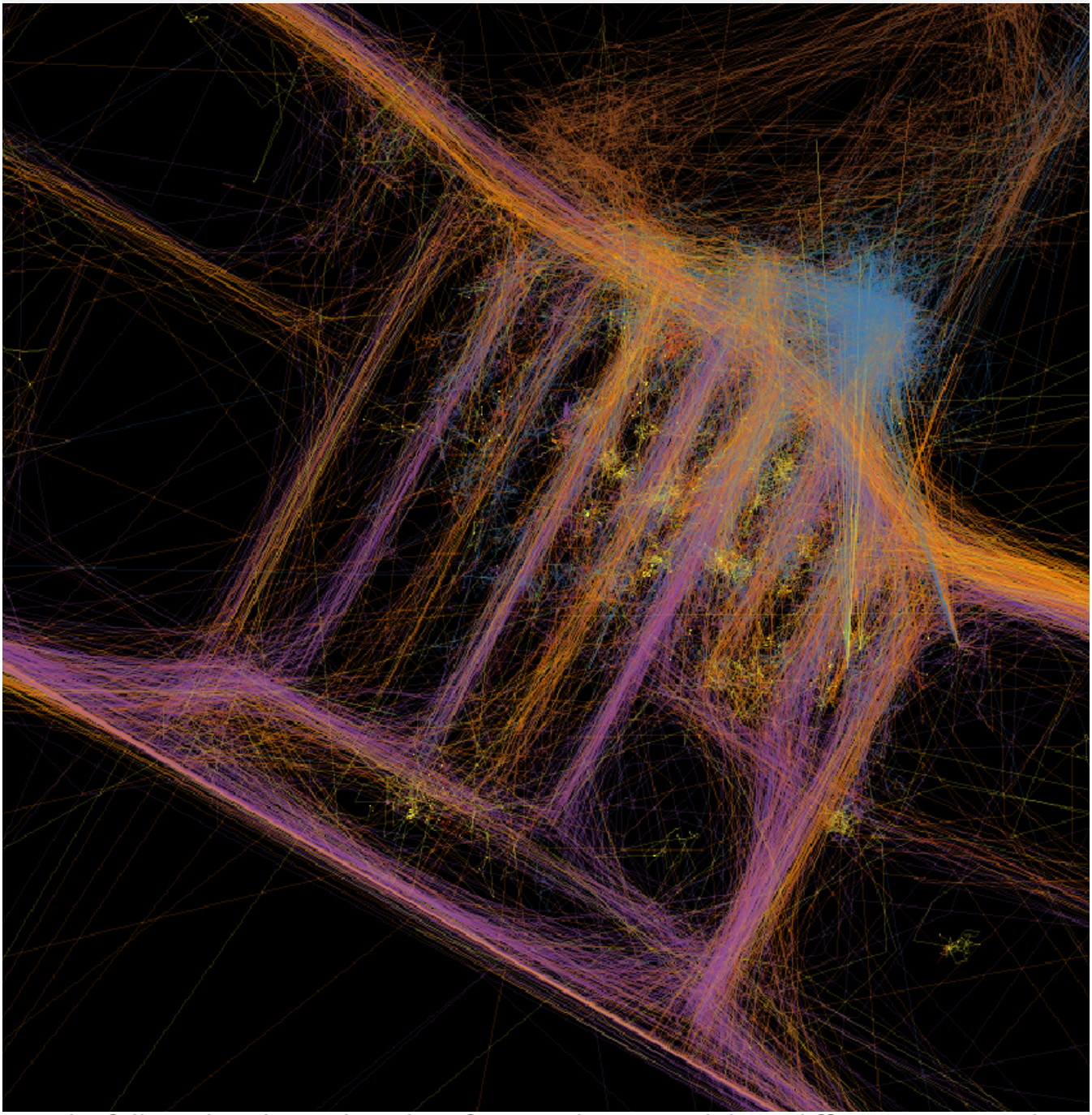
We also measure where each shopper is in their workflow at any given moment. This lets us see what shoppers are doing inside the store locations (when measurement is accurate enough):



High accuracy GPS updates from within a store location

The checkout area (brown) is near the shopping area (purple), but the staging area (pink) is on another side of the store.

Or, by visualizing paths instead of points, we can clearly see the movement of shoppers through store parking lots:



Paths followed in the parking lot of a store location while in different states in the app (color)

Each lane is (mostly) one way (pink or orange), and shoppers enter the store to pick up groceries on one side of the building (blue). You can even see where shoppers typically park while waiting for their next order (yellow).

Visualizations like these help us to:

- Build intuition for how our logistics system functions at scale

- Generate hypotheses for ways to improve our algorithms or operations
- Confirm that changes to production have the expected behavior
- Identify patterns that fast shoppers exhibit, and share those insights with other shoppers
- Make better operational decisions about parking spaces, store locations and our product offering

If you are interested in joining the team to help us engineer, optimize or analyze our logistics systems, or to work on any of the other many challenging problems we have at Instacart, check out our careers page at careers.instacart.com.