SkyQuery: An Aerial Drone Video Sensing Platform

Favyen Bastani, Songtao He, Ziwen Jiang, Osbert Bastani, Sam Madden

Applications of Aerial Drone Video Sensing

Traffic Analysis and Traffic Planning

Periodic Infrastructure Inspection

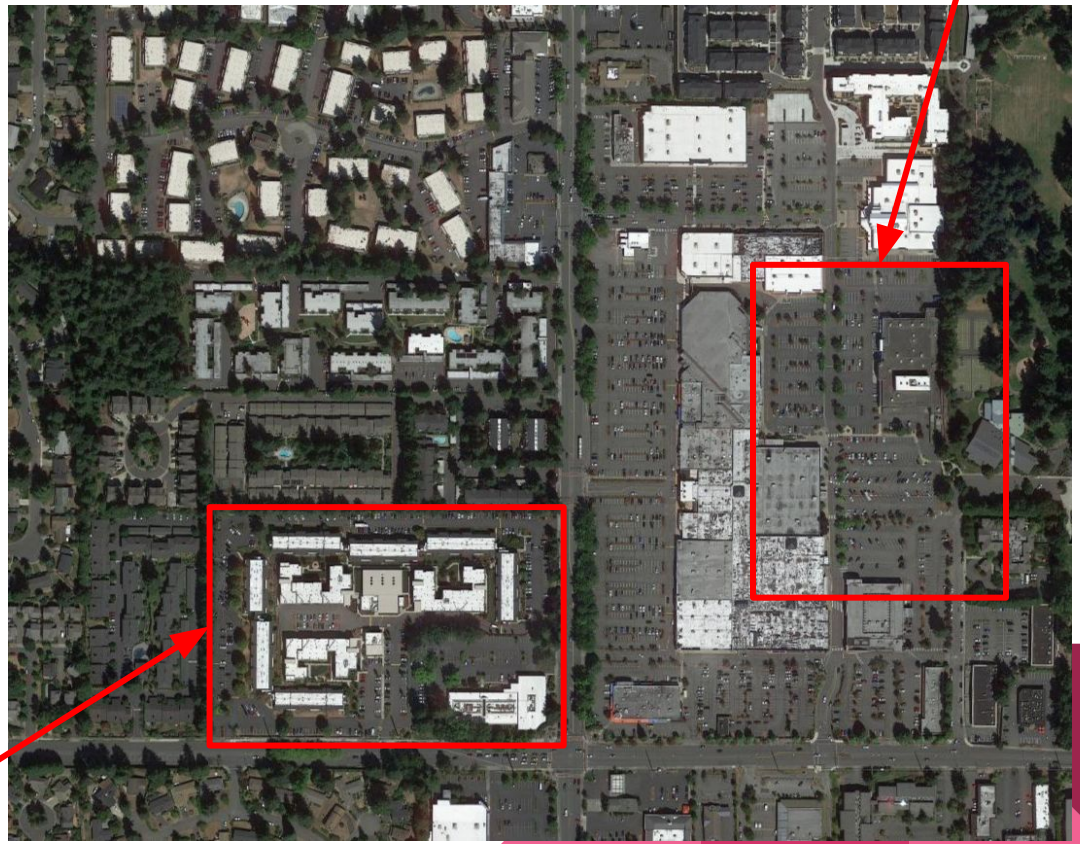
Precision Agriculture

Wildlife Population Management

Parking Monitoring

Parking Monitoring Example

- **Goal**: Monitor # parked cars in various parts of a city
- **Motivation**
  - Could make live map showing where spots are available
  - Or provide data for city planning



Parking Monitoring Example: Challenges

To monitor # parked cars across a city, need to:

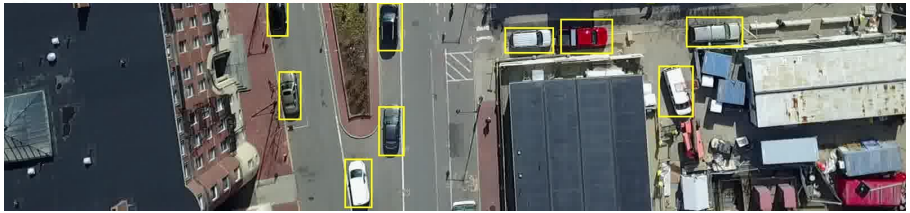
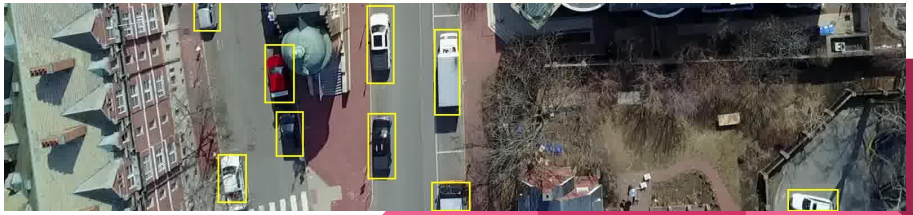
1. **Apply object detector to detect cars**

Parking Monitoring Example: Challenges

To monitor # parked cars across a city, need to:

1. **Apply object detector to detect cars**

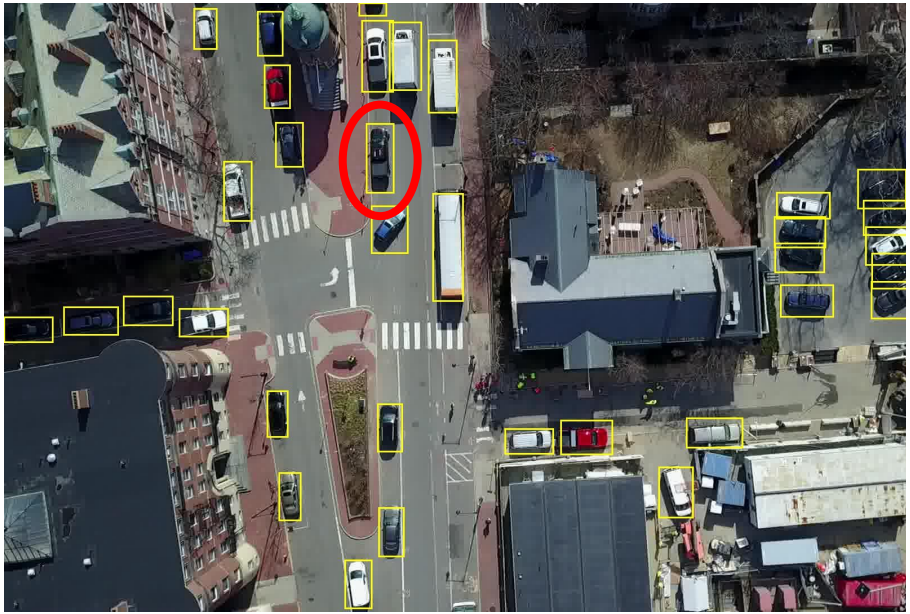


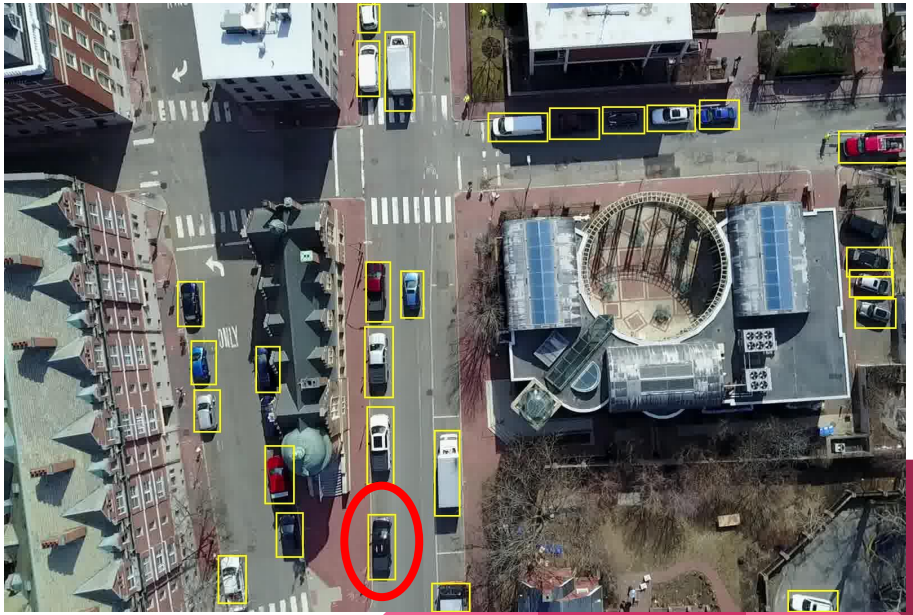




Parking Monitoring Example: Challenges

To monitor # parked cars across a city, need to:

1. **Apply object detector to detect cars**





Parking Monitoring Example: Challenges

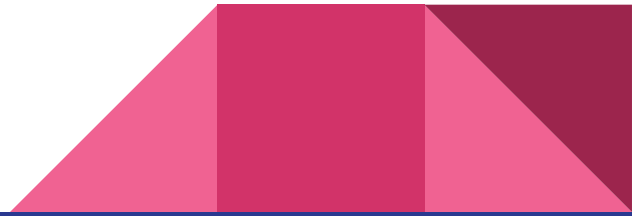
To monitor # parked cars across a city, need to:

1. Apply object detector to detect cars
2. **Align video frames to obtain absolute coordinates**

Parking Monitoring Example: Challenges

To monitor # parked cars across a city, need to:

1. Apply object detector to detect cars
2. Align video frames to obtain absolute coordinates
3. **Analyze the car detections to identify cars that are stopped in one place for longer than a couple minutes**

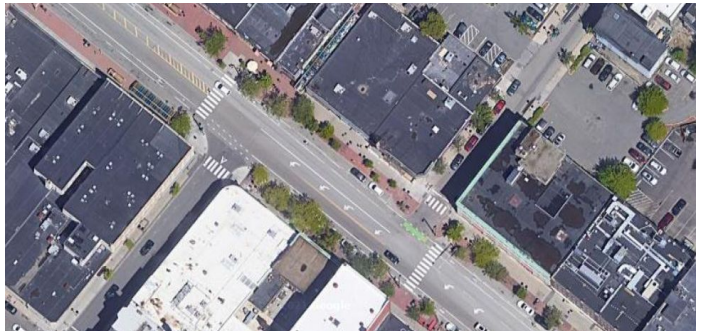


Parking Monitoring Example: Challenges

To monitor # parked cars across a city, need to:

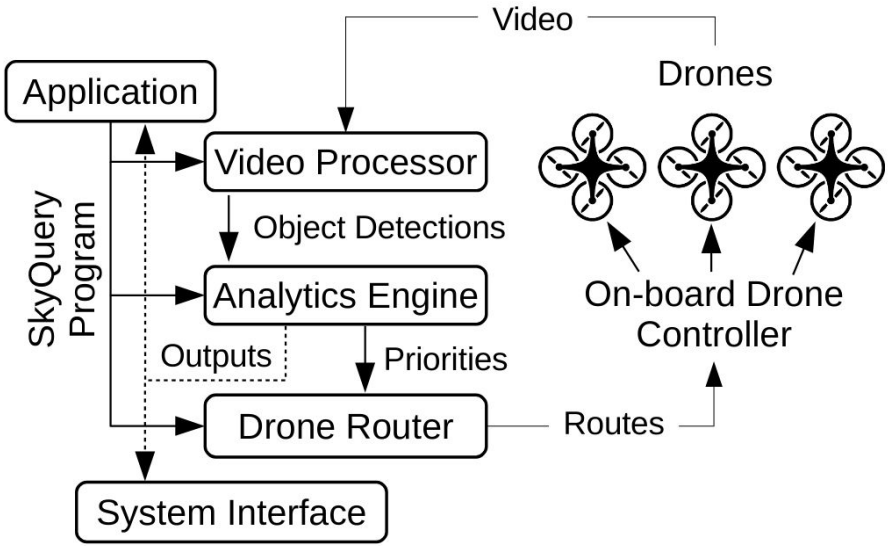
1. Apply object detector to detect cars
2. Align video frames to obtain absolute coordinates
3. Analyze the car detections to identify cars that are stopped in one place for longer than a couple minutes
4. **Update aerial drone routes to prioritize areas with more rapid change**

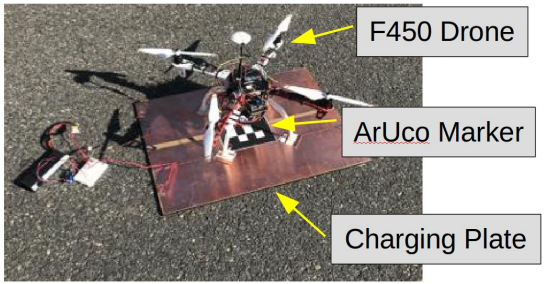
Office lot: predictable, with regular increase at 9-10am and decrease at 5pm



Retail/commercial corridor: less predictable, send drones more often

SkyQuery: An Aerial Drone Video Sensing Platform

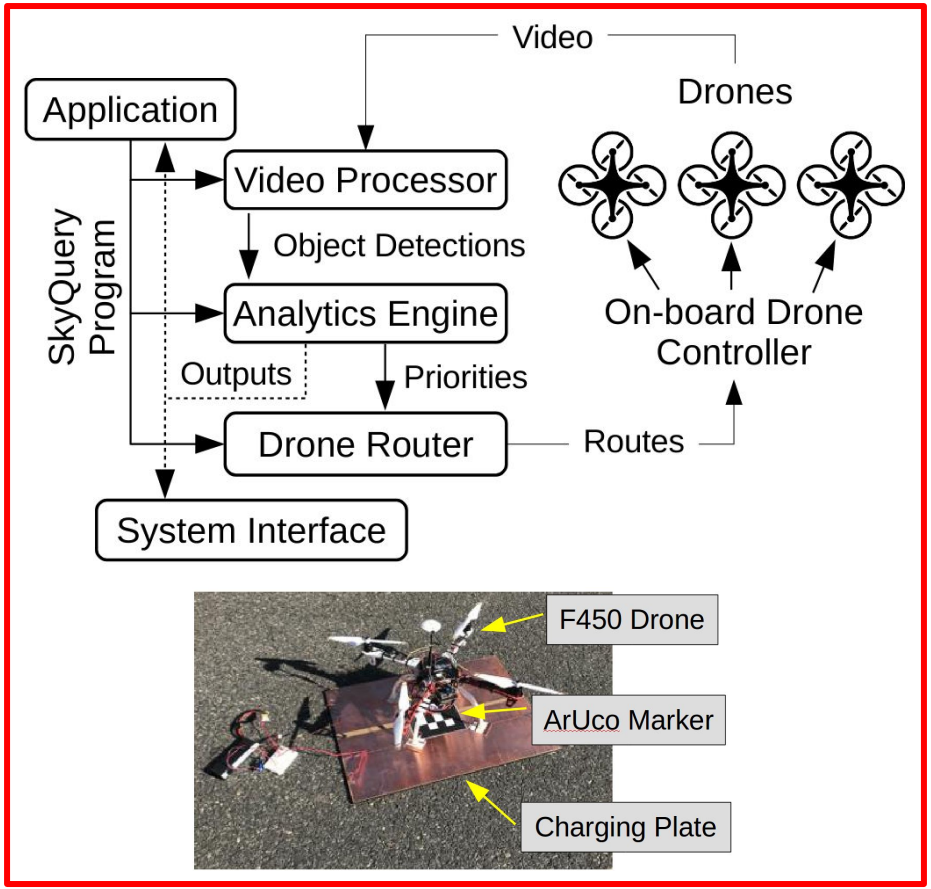




cars = ObjectDetection(**video**, 'car_model')
car_traj = ObjectTracking(**cars**)
stopped = Select(**car_traj**, displacement < 3)
merged = Merge(**stopped**)
parked = Select(**merged**, duration > 120)
raw = ToMatrix(**parked**, f=Count, cell_size=64)
spots_bin = Aggregate(**raw**, f=Sum) > 0
spots = Thin(**spots_bin**)
rates, **counts** = ForecastRates(**raw**, model=Gaussian)
priorities = Aggregate(**rates**, f=Priority)

Example Program for Parking Monitoring
and Locating Parking Spots

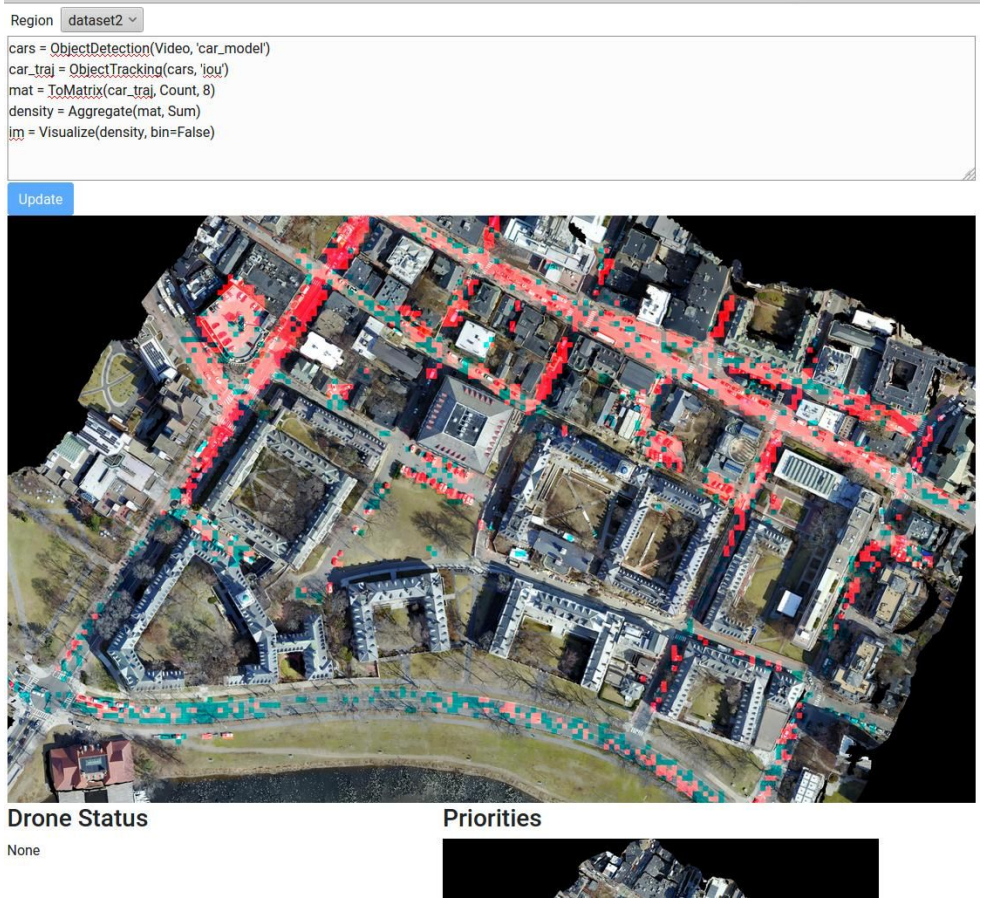
SkyQuery: An Aerial Drone Video Sensing Platform



cars = ObjectDetection(**video**, 'car_model')
car_traj = ObjectTracking(**cars**)
stopped = Select(**car_traj**, displacement < 3)
merged = Merge(**stopped**)
parked = Select(**merged**, duration > 120)
raw = ToMatrix(**parked**, f=Count, cell_size=64)
spots_bin = Aggregate(**raw**, f=Sum) > 0
spots = Thin(**spots_bin**)
rates, **counts** = ForecastRates(**raw**, model=Gaussian)
priorities = Aggregate(**rates**, f=Priority)

Example Program for Parking Monitoring
and Locating Parking Spots

SkyQuery: An Aerial Drone Video Sensing Platform



Query Language

```
cars = ObjectDetection(video, 'car_model')
car_traj = ObjectTracking(cars)
stopped = Select(car_traj, displacement < 3)
merged = Merge(stopped)
parked = Select(merged, duration > 120)
raw = ToMatrix(parked, f=Count, cell_size=64)
spots_bin = Aggregate(raw, f=Sum) > 0
spots = Thin(spots_bin)
rates, counts = ForecastRates(raw, model=Gaussian)
priorities = Aggregate(rates, f=Priority)
```

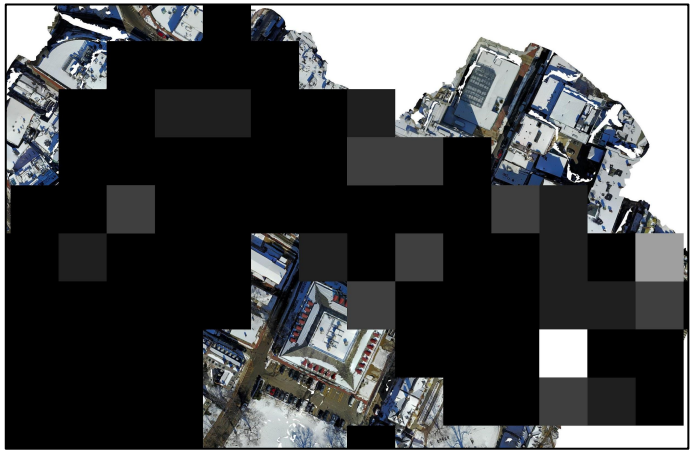
Example Program for Parking Monitoring
and Locating Parking Spots

Tables: bold names like **cars**, **car_traj**, and **stopped**.
Operations: ObjectDetection, ObjectTracking, etc.
Operations transform input tables into output tables.

Query Language: Types of Tables

Detections: each row is a bounding polygon

Sequences: each row is a sequence of detections

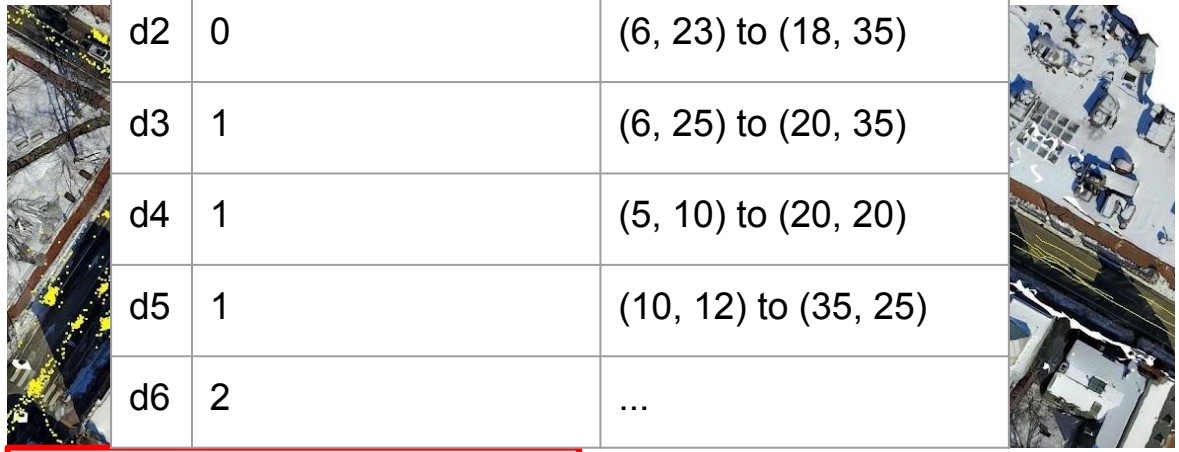


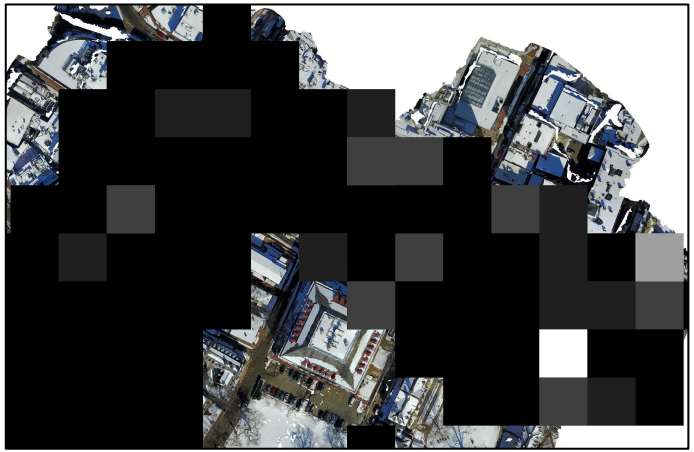
Matrices: a spatio-temporal matrix that divides the region into a 2D grid

- Associates time series to each grid cell
- e.g., # parked cars in that cell over time
- Each row is one observation of the time series at a particular cell

Q…bles

ID	Time	Bounding Polygon
d1	0	(5, 10) to (20, 20)
d2	0	(6, 23) to (18, 35)
d3	1	(6, 25) to (20, 35)
d4	1	(5, 10) to (20, 20)
d5	1	(10, 12) to (35, 25)
d6	2	...





Detections: each row is a bounding polygon

Sequences: each row is a sequence of detections

Matrices: a spatio-temporal matrix that divides the region into a 2D grid

- Associates time series to each grid cell
- e.g., # parked cars in that cell over time
- Each row is one observation of the time series at a particular cell

Q

ID	Time	Bounding Polygon
d1	0	(5, 10) to (20, 20)
d2	0	(6, 23) to (18, 35)
d3	1	(6, 25) to (20, 35)
d4	1	(5, 10) to (20, 20)
d5	1	(10, 12) to (35, 25)
d6	2	...

Detections: each row is a bounding polygon

Time	Sequence <d#(**timestamp**)>
0	d1(**0**), d4(**2**), d6(**3**), d11(**6**), d13(**7**), d18(**9**)
0	d2(**0**), d3(**1**), d5(**2**), d7(**3**), d8(**4**), d10(**5**)
4	d9(**4**), d12(**6**), d15(**8**)
7	d14(**7**), d16(**8**), d19(**10**), d21(**11**), d22(**12**)
8	d17(**8**), d20(**10**), d23(**14**), d24(**15**), d25(**16**)
17	d26(**17**), d27(**19**)

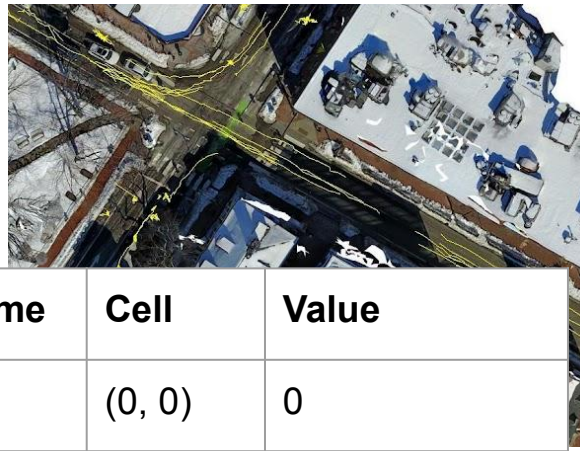
Sequences: each row is a sequence of detections

Matrices: a spatio-temporal matrix that divides the region into a 2D grid

- Associates time series to each grid cell
- e.g., # parked cars in that cell over time
- Each row is one observation of the time series at a particular cell

Query Language: Types of Tables

Detections: each row is a bounding polygon



Time	Cell	Value
0	(0, 0)	0
0	(0, 1)	0
20	(1, 0)	4
40	(2, 0)	0
40	(2, 1)	10

Matrices: a spatio-temporal matrix that divides the region into a 2D grid

- Associates time series to each grid cell
- e.g., # parked cars in that cell over time
- Each row is one observation of the time series at a particular cell


cars = ObjectDetection(**video**, 'car_model')
car_traj = ObjectTracking(**cars**)
stopped = Select(**car_traj**, displacement < 3)
merged = Merge(**stopped**)
parked = Select(**merged**, duration > 120)
raw = ToMatrix(**parked**, f=Count, cell_size=64)
spots_bin = Aggregate(**raw**, f=Sum) > 0
spots = Thin(**spots_bin**)
rates, **counts** = ForecastRates(**raw**, model=Gaussian)
priorities = Aggregate(**rates**, f=Priority)


cars = ObjectDetection(**video**, ‘car_model’)
car_traj = ObjectTracking(**cars**)
stopped = Select(**car_traj**, displacement < 3)
merged = Merge(**stopped**)
parked = Select(**merged**, duration > 120)
raw = ToMatrix(**parked**, f=Count, cell_size=64)
spots_bin = Aggregate(**raw**, f=Sum) > 0
spots = Thin(**spots_bin**)
rates, **counts** = ForecastRates(**raw**, model=Gaussian)
priorities = Aggregate(**rates**, f=Priority)


cars = ObjectDetection(**video**, ‘car_model’)
car_traj = ObjectTracking(**cars**)
stopped = Select(**car_traj**, displacement < 3)
merged = Merge(**stopped**)
parked = Select(**merged**, duration > 120)
raw = ToMatrix(**parked**, f=Count, cell_size=64)
spots_bin = Aggregate(**raw**, f=Sum) > 0
spots = Thin(**spots_bin**)
rates, **counts** = ForecastRates(**raw**, model=Gaussian)
priorities = Aggregate(**rates**, f=Priority)


ction(**video**, ‘car_model’)
racking(**cars**)
car_traj, displacement < 3)
stopped)
erged, duration > 120)
rked, f=Count, cell_size=64)
gate(**raw**, f=Sum) > 0
s_bin)
recastRates(**raw**, model=Gaussian)
gate(**rates**, f=Priority)


ction(video, ‘car_model’)
racking(cars)
car_traj, displacement < 3)
stopped)
erged, duration > 120)
rked, f=Count, cell_size=64)
gate(raw, f=Sum) > 0
s_bin)
recastRates(raw, model=Gaussian)
gate(rates, f=Priority)


cars = ObjectDetection(**video**, ‘car_model’)
car_traj = ObjectTracking(**cars**)
stopped = Select(**car_traj**, displacement < 3)
merged = Merge(**stopped**)
parked = Select(**merged**, duration > 120)
raw = ToMatrix(**parked**, f=Count, cell_size=64)
spots_bin = Aggregate(**raw**, f=Sum) > 0
spots = Thin(**spots_bin**)
rates, **counts** = ForecastRates(**raw**, model=Gaussian)
priorities = Aggregate(**rates**, f=Priority)

cars = ObjectDetection(**video**, ‘car_model’)
car_traj = ObjectTracking(**cars**)
stopped = Select(**car_traj**, displacement < 3)
merged = Merge(**stopped**)
parked = Select(**merged**, duration > 120)
raw = ToMatrix(**parked**, f=Count, cell_size=64)
spots_bin = Aggregate(**raw**, f=Sum) > 0
spots = Thin(**spots_bin**)
rates, **counts** = ForecastRates(**raw**, model=Gaussian)
priorities = Aggregate(**rates**, f=Priority)

cars = ObjectDetection(**video**, ‘car_model’)
car_traj = ObjectTracking(**cars**)
stopped = Select(**car_traj**, displacement < 3)
merged = Merge(**stopped**)
parked = Select(**merged**, duration > 120)
raw = ToMatrix(**parked**, f=Count, cell_size=64)
spots_bin = Aggregate(**raw**, f=Sum) > 0
spots = Thin(**spots_bin**)
rates, **counts** = ForecastRates(**raw**, model=Gaussian)
priorities = Aggregate(**rates**, f=Priority)

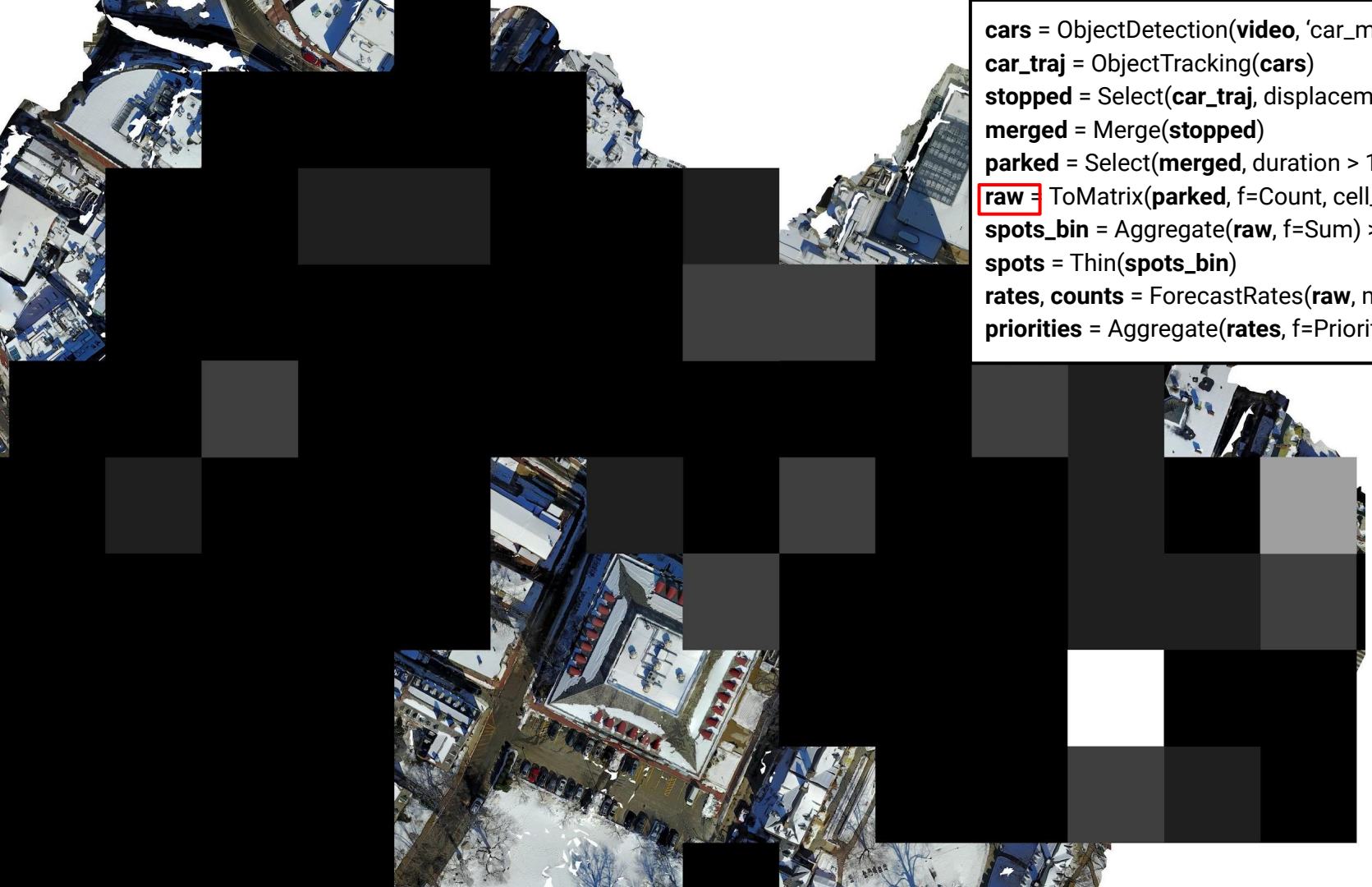
Suppose that:

- A drone observes a stopped car
- 5 minutes later, another drone observes car with similar appearance in the same spot

Merge(...) would merge these sequences into one sequence.


cars = ObjectDetection(**video**, ‘car_model’)
car_traj = ObjectTracking(**cars**)
stopped = Select(**car_traj**, displacement < 3)
merged = Merge(**stopped**)
parked = Select(**merged**, duration > 120)
raw = ToMatrix(**parked**, f=Count, cell_size=64)
spots_bin = Aggregate(**raw**, f=Sum) > 0
spots = Thin(**spots_bin**)
rates, **counts** = ForecastRates(**raw**, model=Gaussian)
priorities = Aggregate(**rates**, f=Priority)

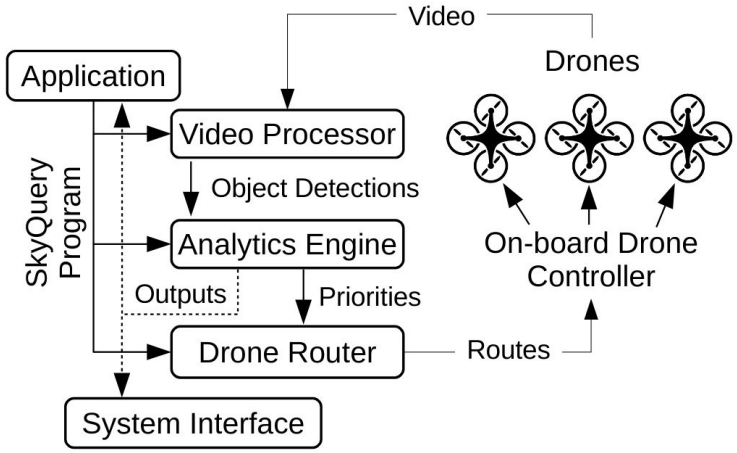
```
cars = ObjectDetection(video, 'car_model')
car_traj = ObjectTracking(cars)
stopped = Select(car_traj, displacement < 3)
merged = Merge(stopped)
parked = Select(merged, duration > 120)
raw = ToMatrix(parked, f=Count, cell_size=64)
spots_bin = Aggregate(raw, f=Sum) > 0
spots = Thin(spots_bin)
rates, counts = ForecastRates(raw, model=Gaussian)
priorities = Aggregate(rates, f=Priority)
```



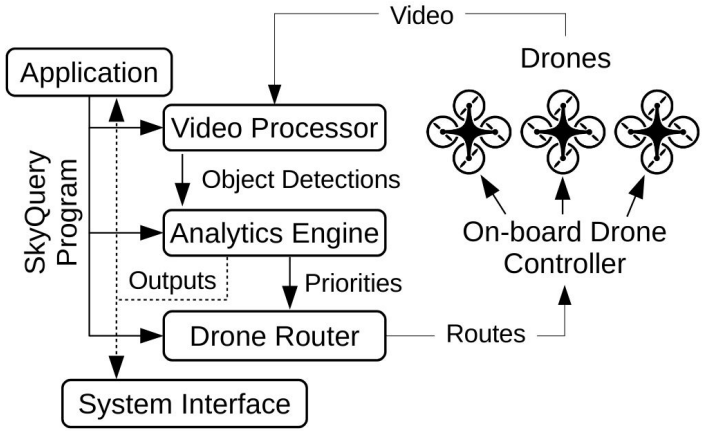
```
cars = ObjectDetection(video, 'car_model')
car_traj = ObjectTracking(cars)
stopped = Select(car_traj, displacement < 3)
merged = Merge(stopped)
parked = Select(merged, duration > 120)
raw = ToMatrix(parked, f=Count, cell_size=64)
spots_bin = Aggregate(raw, f=Sum) > 0
spots = Thin(spots_bin)
rates, counts = ForecastRates(raw, model=Gaussian)
priorities = Aggregate(rates, f=Priority)
```

cars = ObjectDetection(**video**, ‘car_model’)
car_traj = ObjectTracking(**cars**)
stopped = Select(**car_traj**, displacement < 3)
merged = Merge(**stopped**)
parked = Select(**merged**, duration > 120)
raw = ToMatrix(**parked**, f=Count, cell_size=64)
spots_bin = Aggregate(**raw**, f=Sum) > 0
spots = Thin(**spots_bin**)
rates, **counts** = ForecastRates(**raw**, model=Gaussian)
priorities = Aggregate(**rates**, f=Priority)

```
cars = ObjectDetection(video, 'car_model')
car_traj = ObjectTracking(cars)
stopped = Select(car_traj, displacement < 3)
merged = Merge(stopped)
parked = Select(merged, duration > 120)
raw = ToMatrix(parked, f=Count, cell_size=64)
spots_bin = Aggregate(raw, f=Sum) > 0
spots = Thin(spots_bin)
rates, counts = ForecastRates(raw, model=Gaussian)
priorities = Aggregate(rates, f=Priority)
```

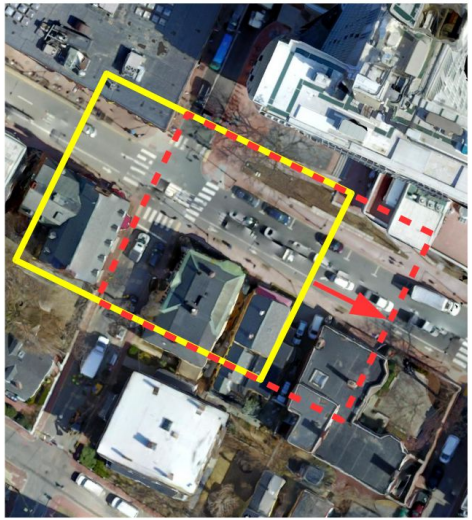


SkyQuery: Incorporates Three New Techniques

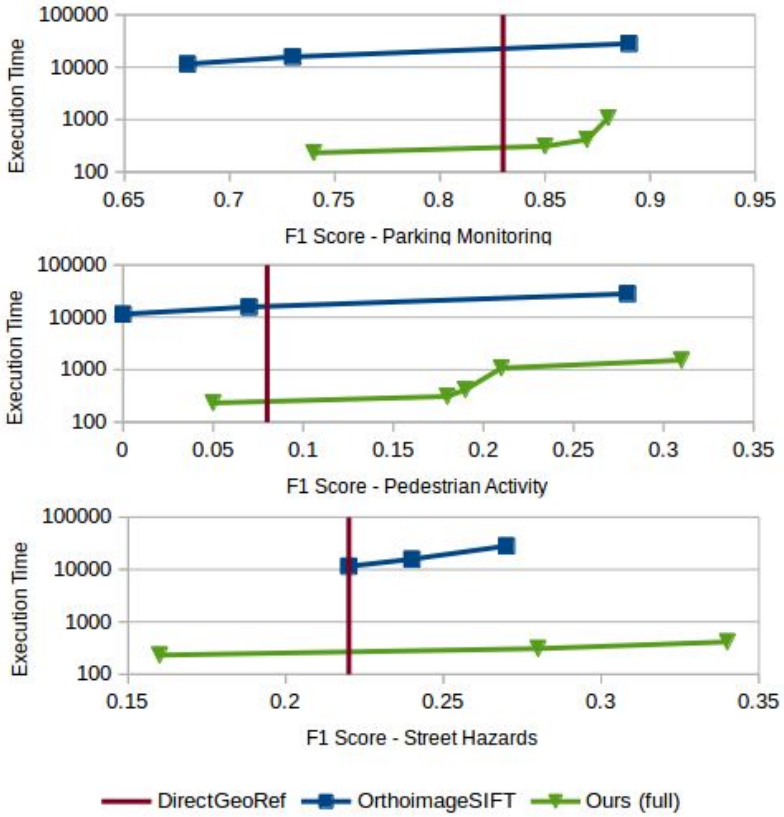


- Fast, high-accuracy frame alignment method
- Change-aware object detection for efficiently detecting small objects
- Drone router to prioritize flights over unpredictable areas

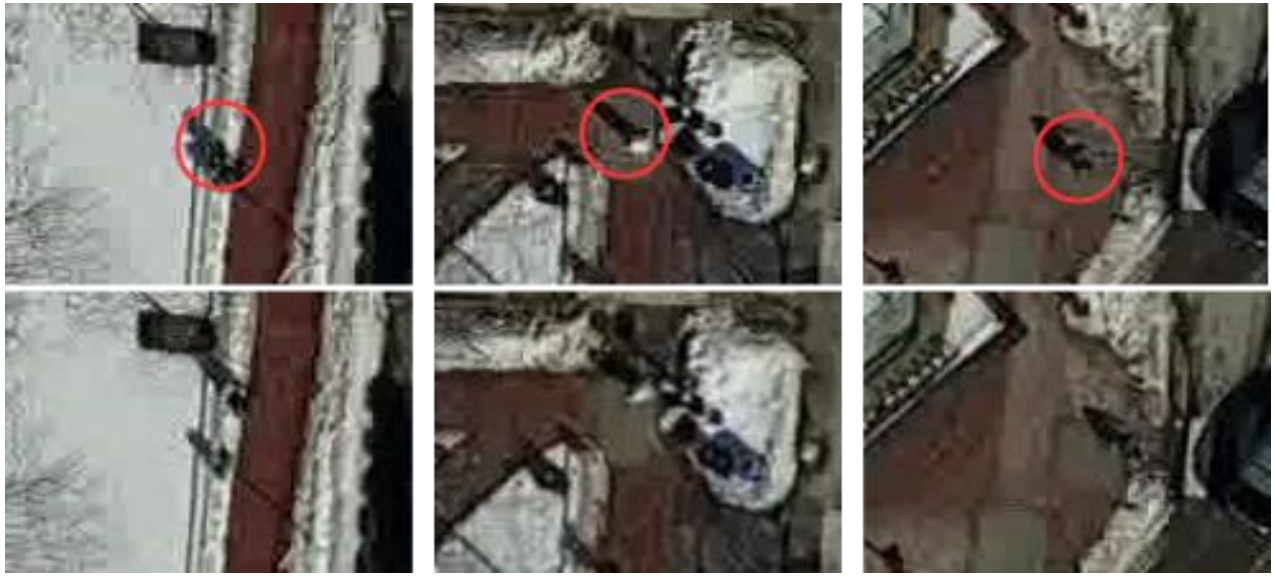
Fast, Accurate Frame Alignment

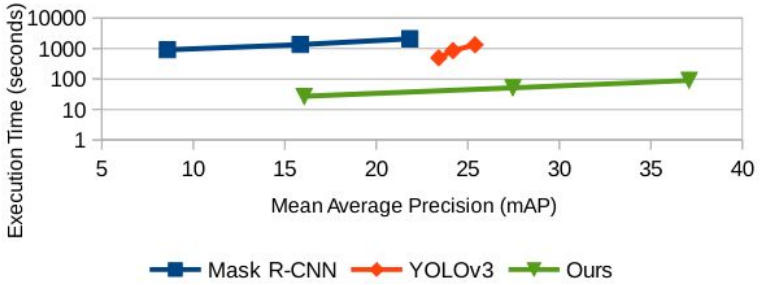


- Efficiently combine GPS and compass readings with image features
- Leverage optimizations such as employing optical flow for updating position estimate across consecutive frames



Change-Aware Object Detection



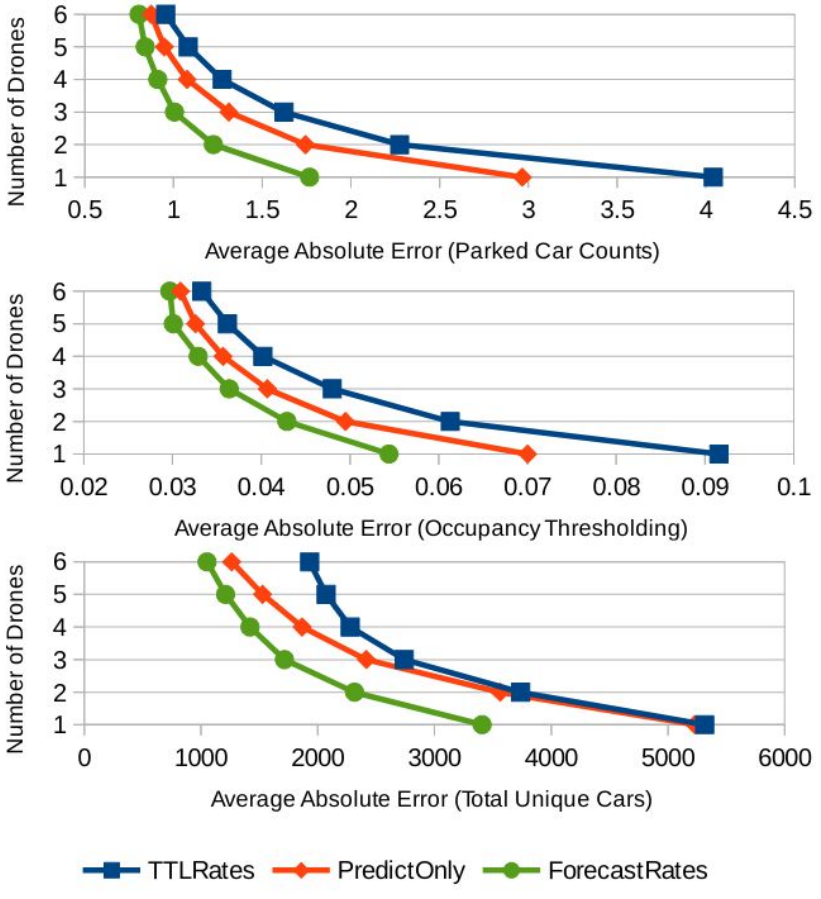


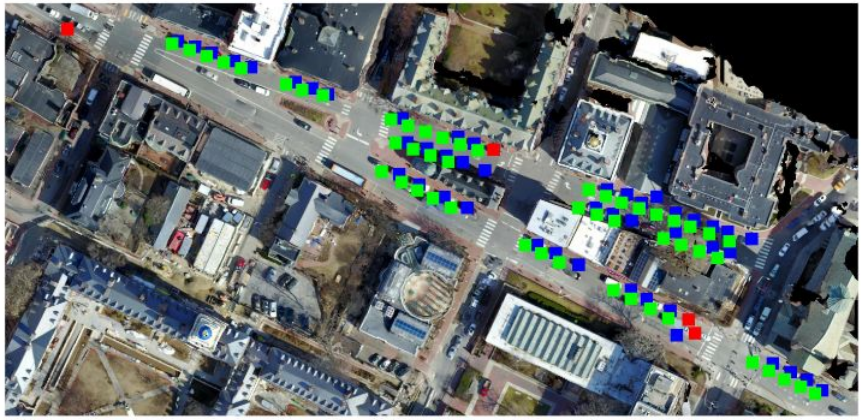
For detecting small objects:

- Increase speed by using a shallow network (don't need large field of view)
- Increase accuracy by comparing multiple frames over time

Drone Router: Forecasting Approach

- Forecast the current time series value in a matrix table at each cell
- Prioritize cells with high-variance probability distributions when computing routes





(a) Comparison of the parking spots inferred by SkyQuery (spots) with the city parking dataset. The city dataset is shown in green. Detected spots that match are colored blue, while incorrect detections are colored red.

(b) Sequences in hazards showing cars stopped in cycling lanes.

(c) At top, SkyQuery's visualization of ped_activity, where cells with detected pedestrians are red. Below, visualizations of a matrix showing locations where we detect pedestrians crossing roads outside of crosswalks (middle), and sequences in ped_road associated with these detections (bottom).

Conclusion

- Aerial drone video sensing has numerous applications
- But implementing systems is challenging
- SkyQuery: addresses these challenges and facilitates the development of applications like wildlife population management to precision agriculture

For code/data, see our project webpage:
https://favyen.com/skyquery/