Uncertainty

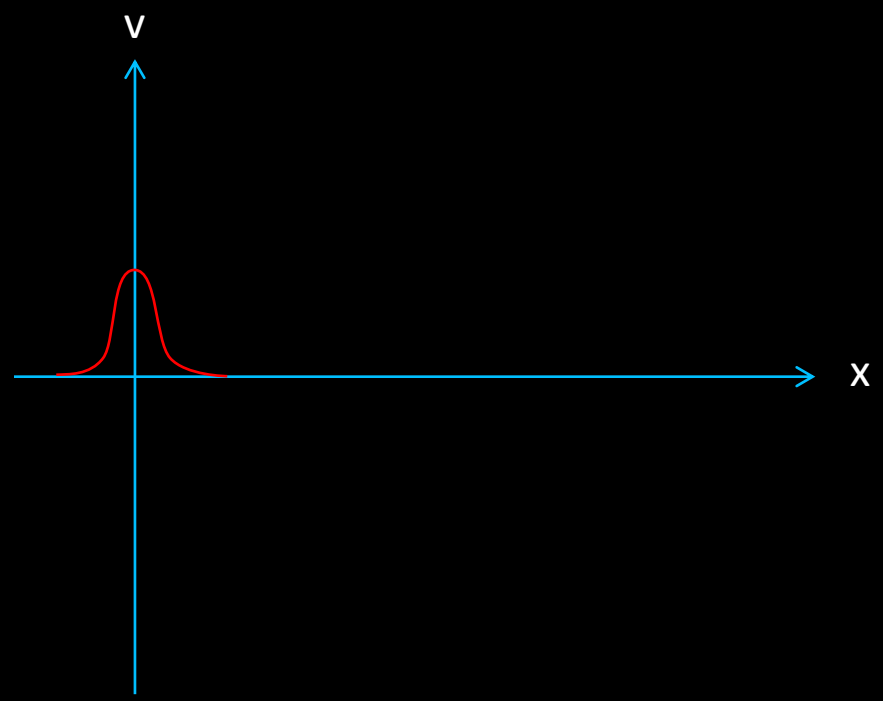
Prof Tom Drummond

Claim 1:

It’s more important to know what you don’t know than to get the right answer

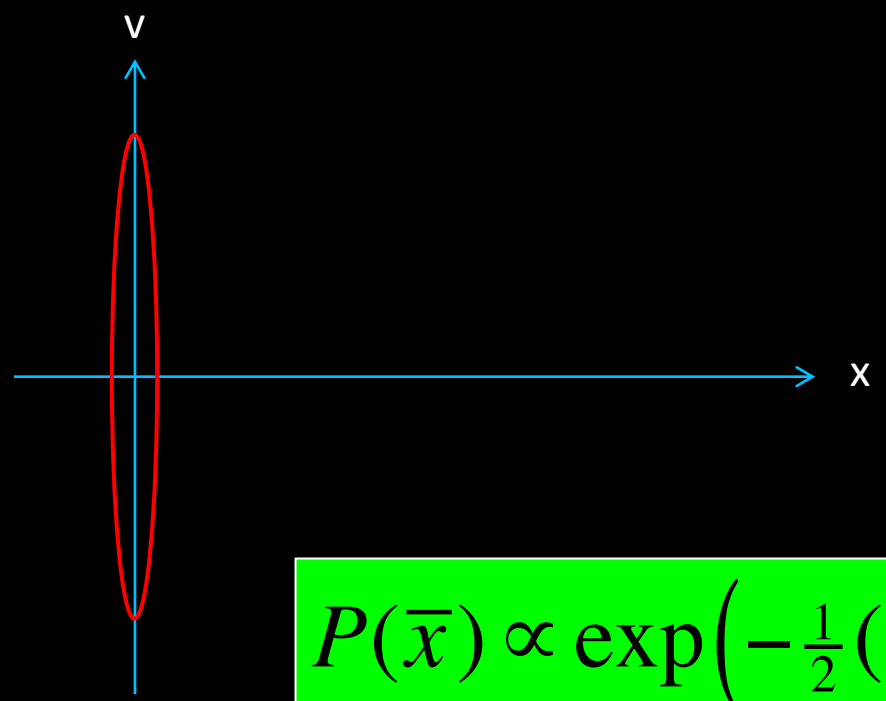
Kalman Filtering

Simple example with 1D position measurement and 1D velocity.
At time t=0, we make an uncertain measurement of x and get the value zero.



Kalman Filtering

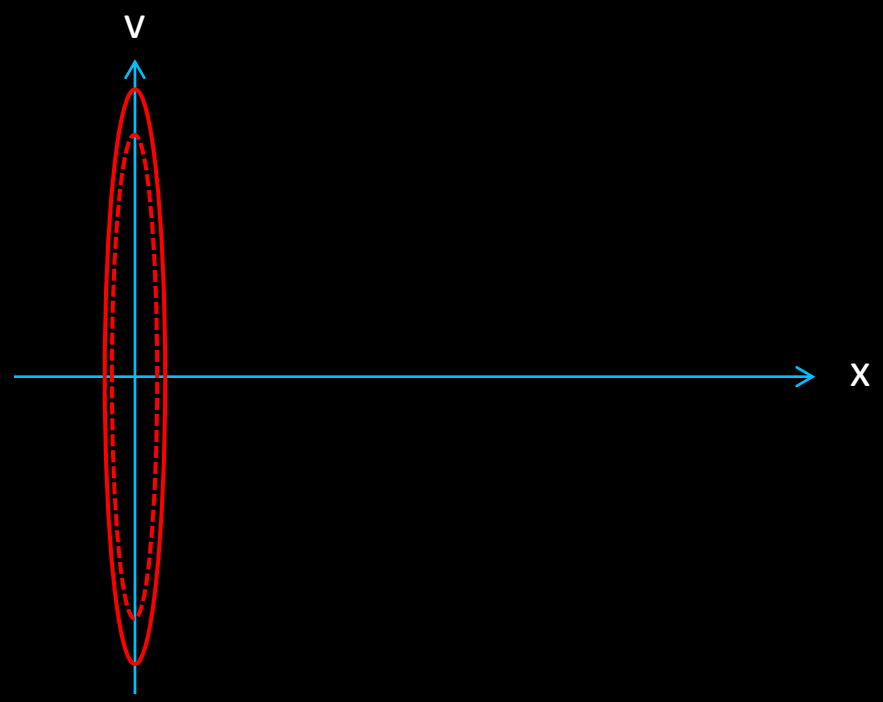
Initialize filter with position measurement and big uncertainty in velocity:



$$P(\bar{x}) \propto \exp\left(-\frac{1}{2}(\bar{x}-\bar{\mu})^T \Sigma_s^{-1}(\bar{x}-\bar{\mu})\right)$$

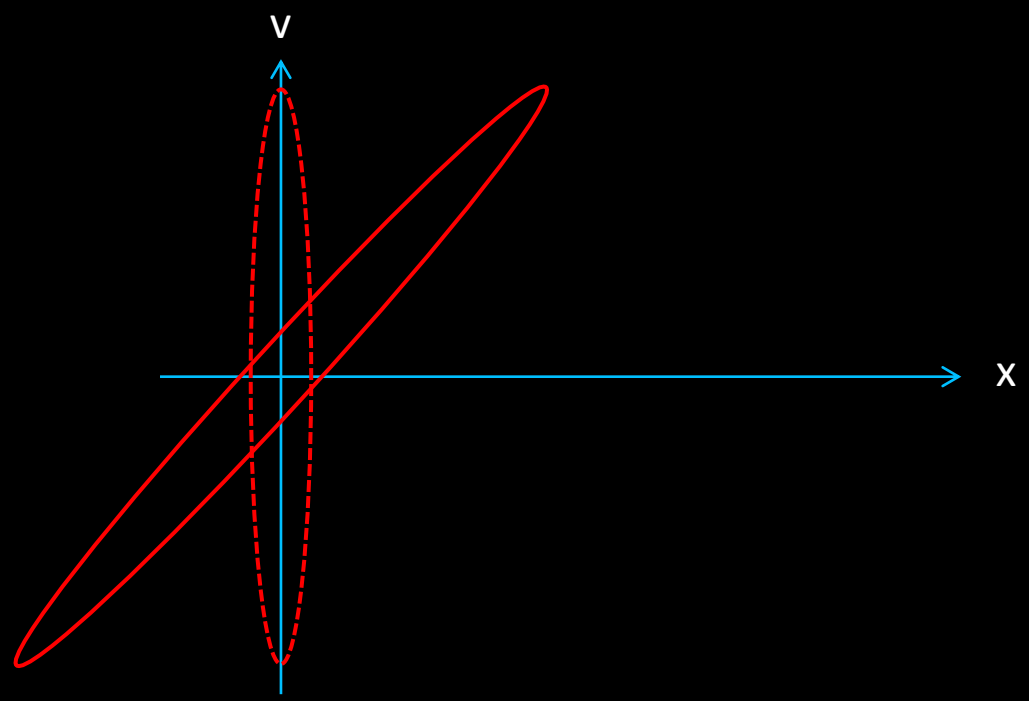
Kalman Filtering

Add process noise…



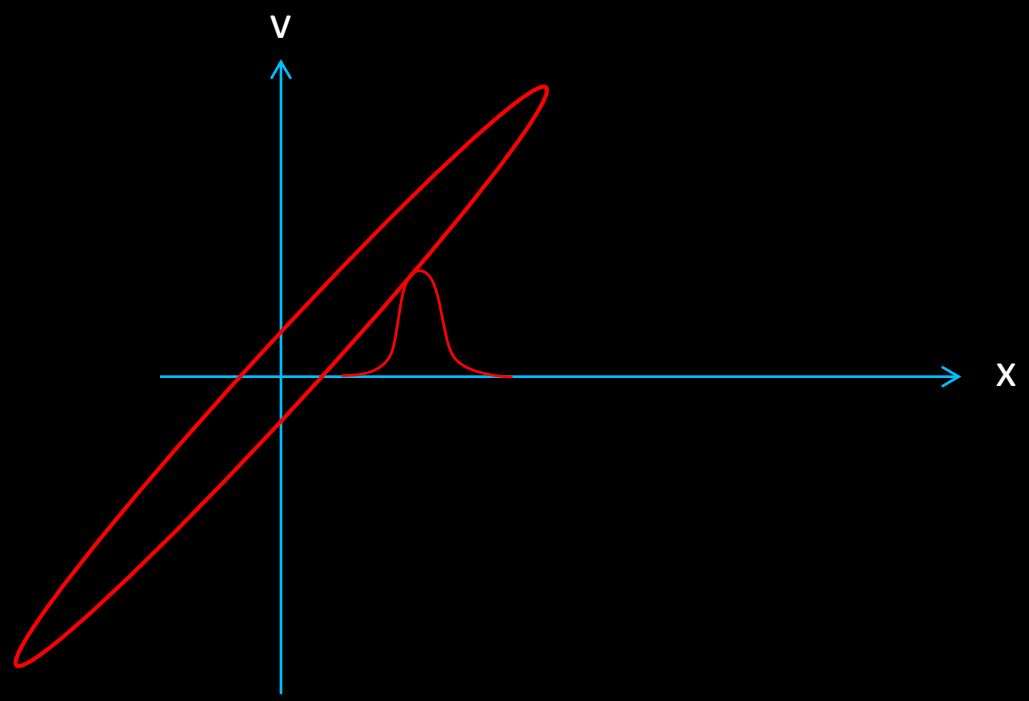
Kalman Filtering

Time passes – position and velocity uncertainties become correlated:



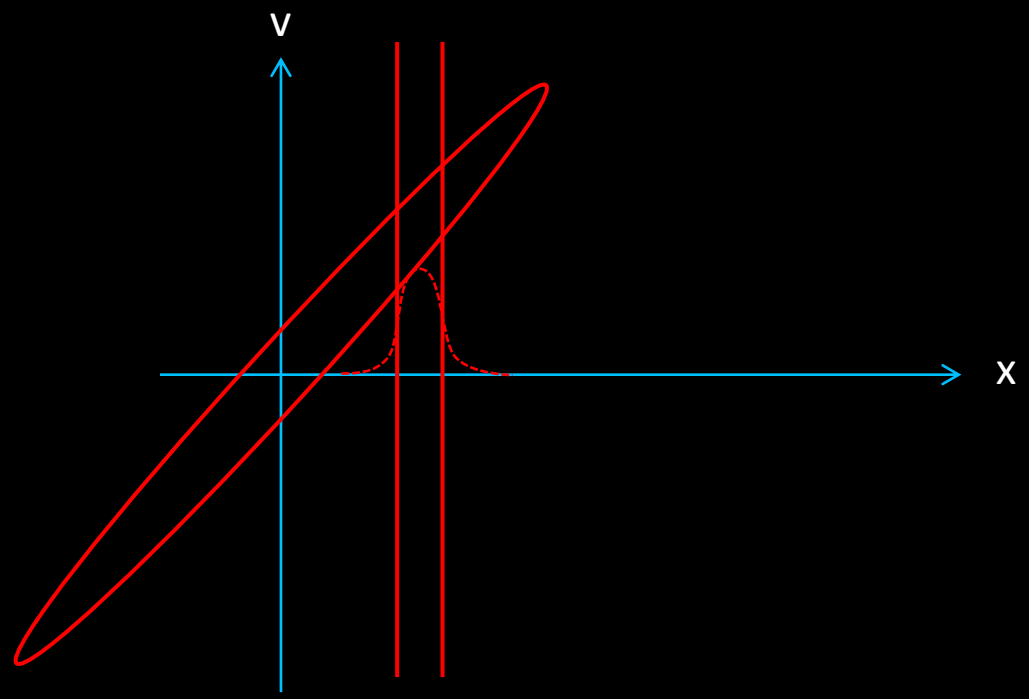
Kalman Filtering

Take a second measurement of position



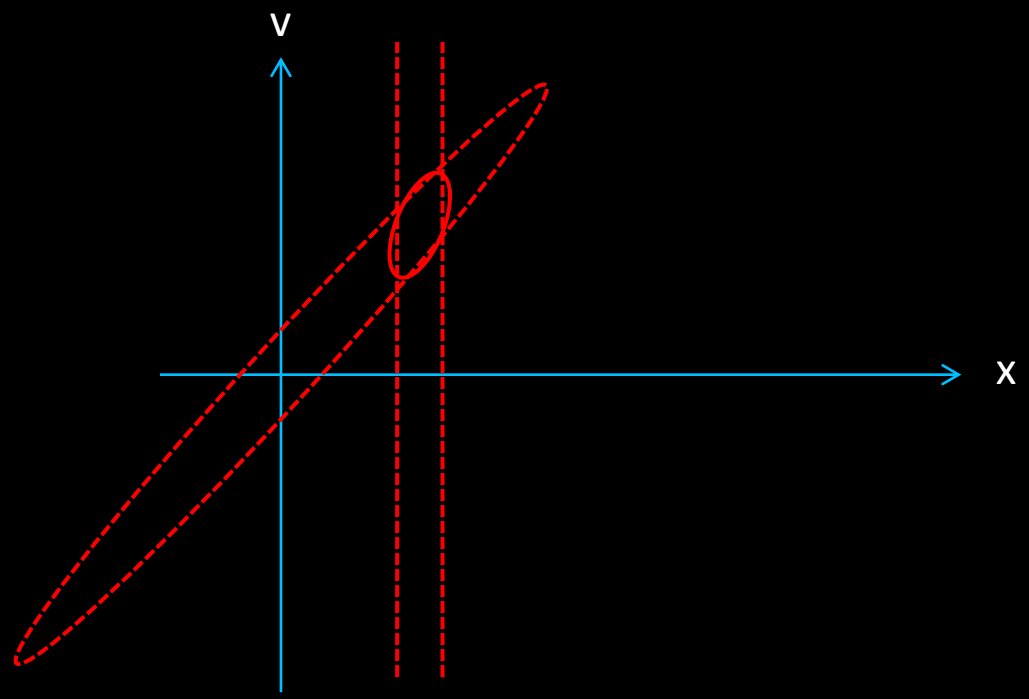
Kalman Filtering

This corresponds to a measurement with some variance in position and infinite variance in velocity



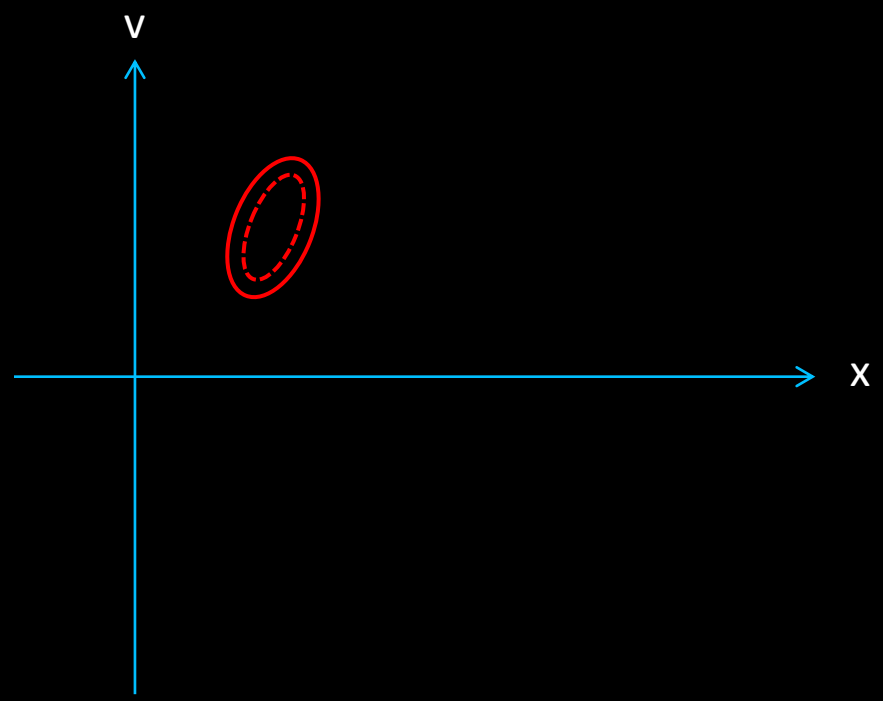
Kalman Filtering

Combine the prior with the new measurement information to obtain a posterior estimate



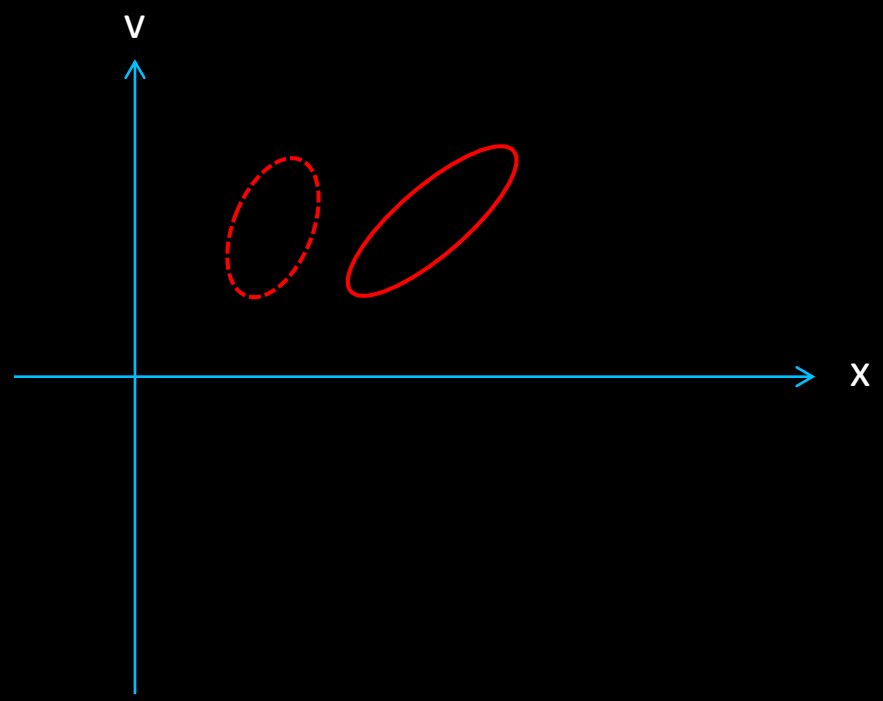
Kalman Filtering

Add process noise…



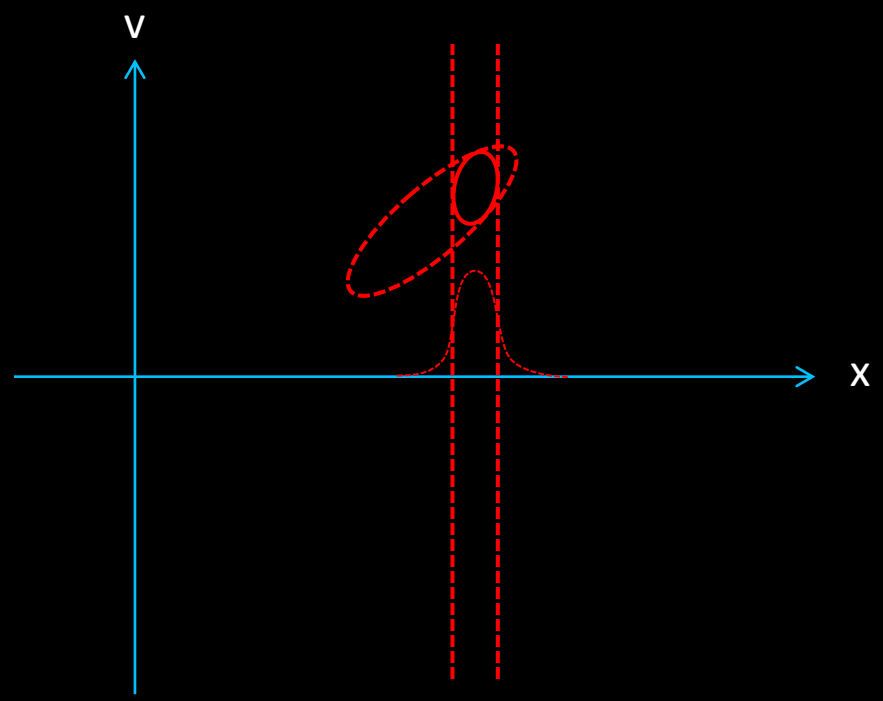
Kalman Filtering

More time passes…

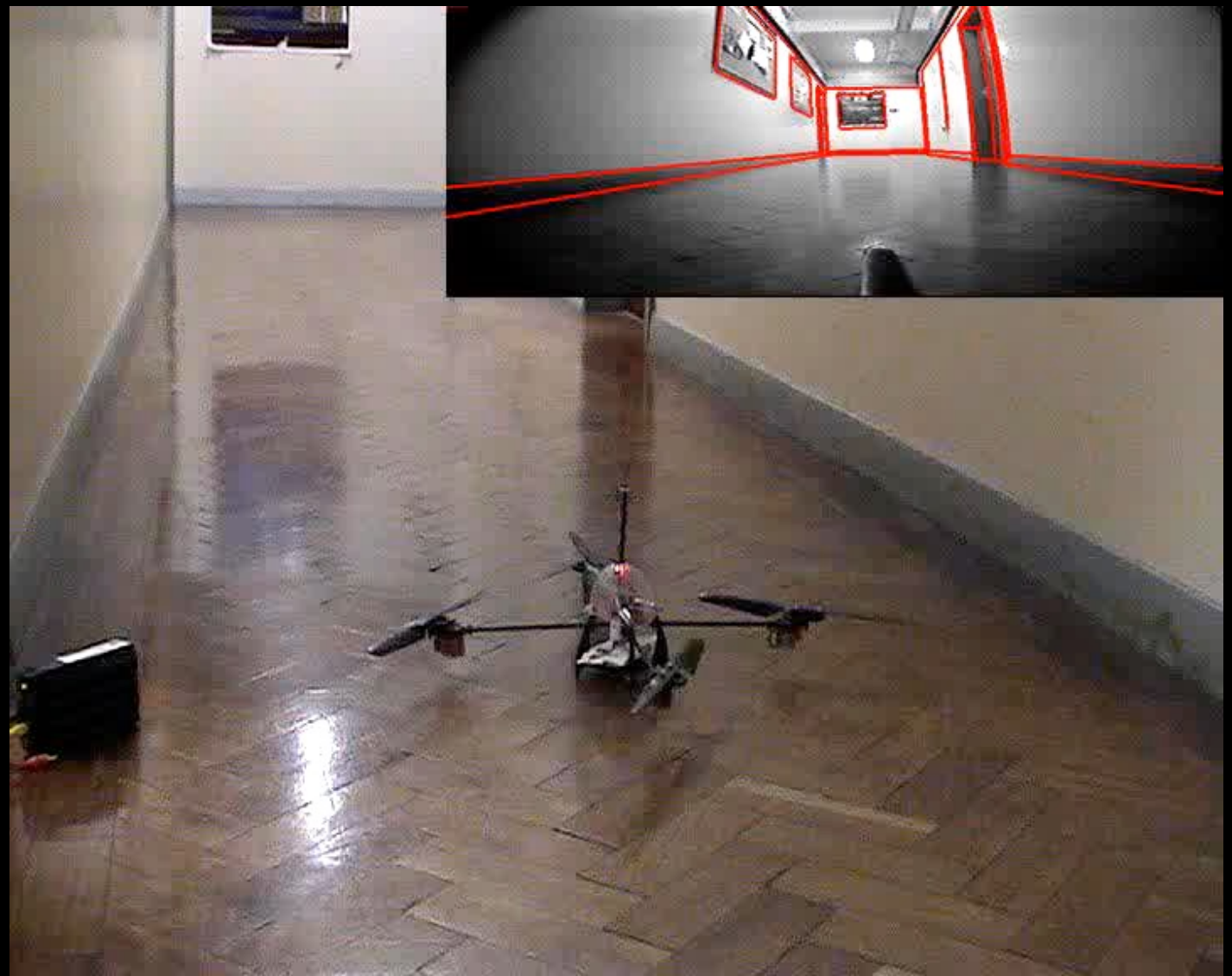


Kalman Filtering

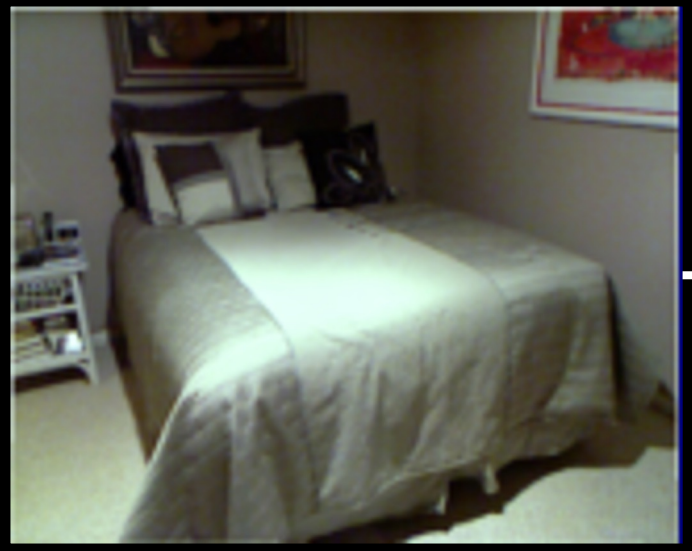
Make a third measurement:



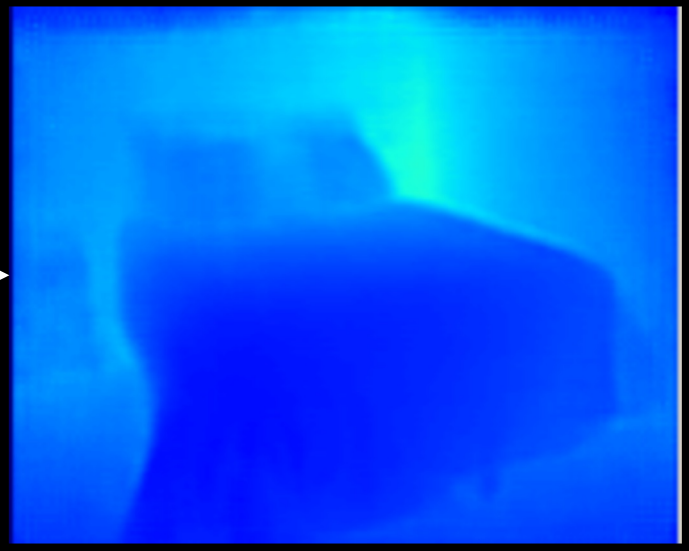
What if the errors are not Gaussian?



Can use learning to estimate depth from a single image

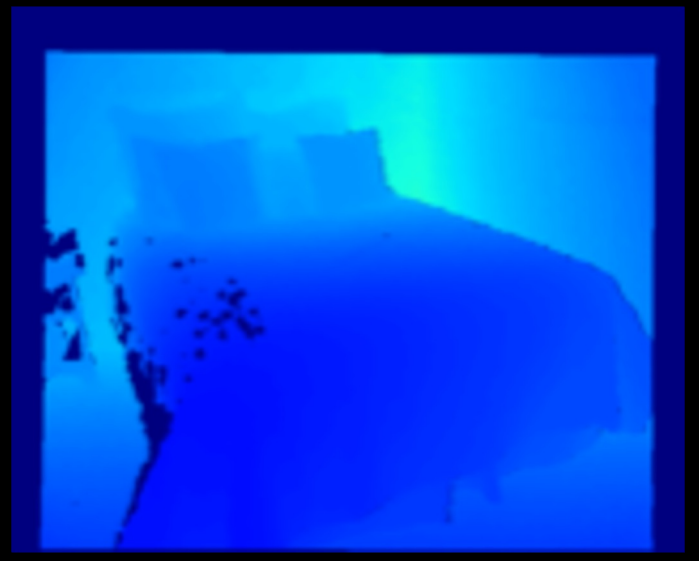


Depth estimation network



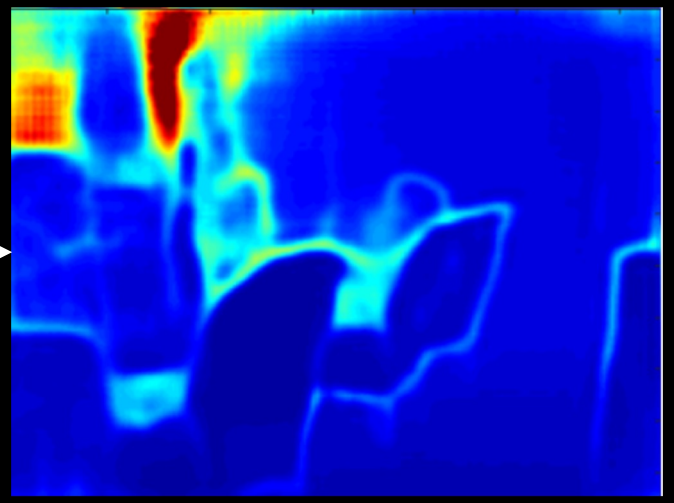
Minimise L_2 loss to ground truth.

Q: Should we estimate z, 1/z or log(z)?

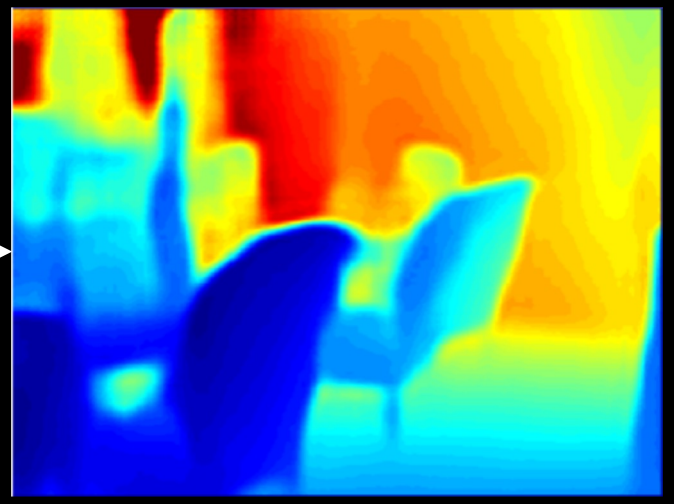


Better to learn uncertainty as well as depth estimates

Depth estimation network



Uncertainty (σ)

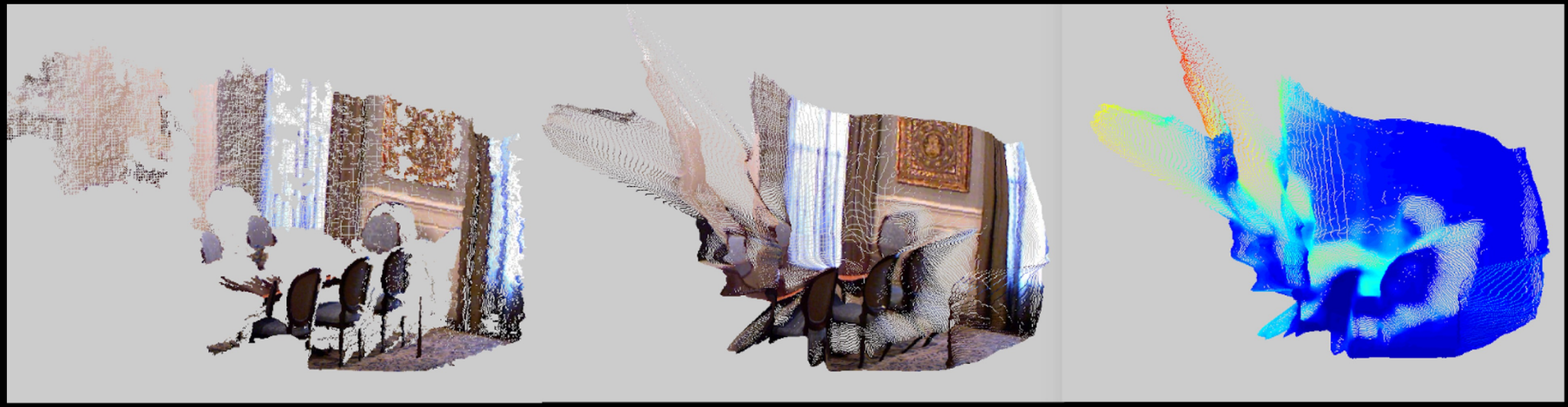


Depth (μ)

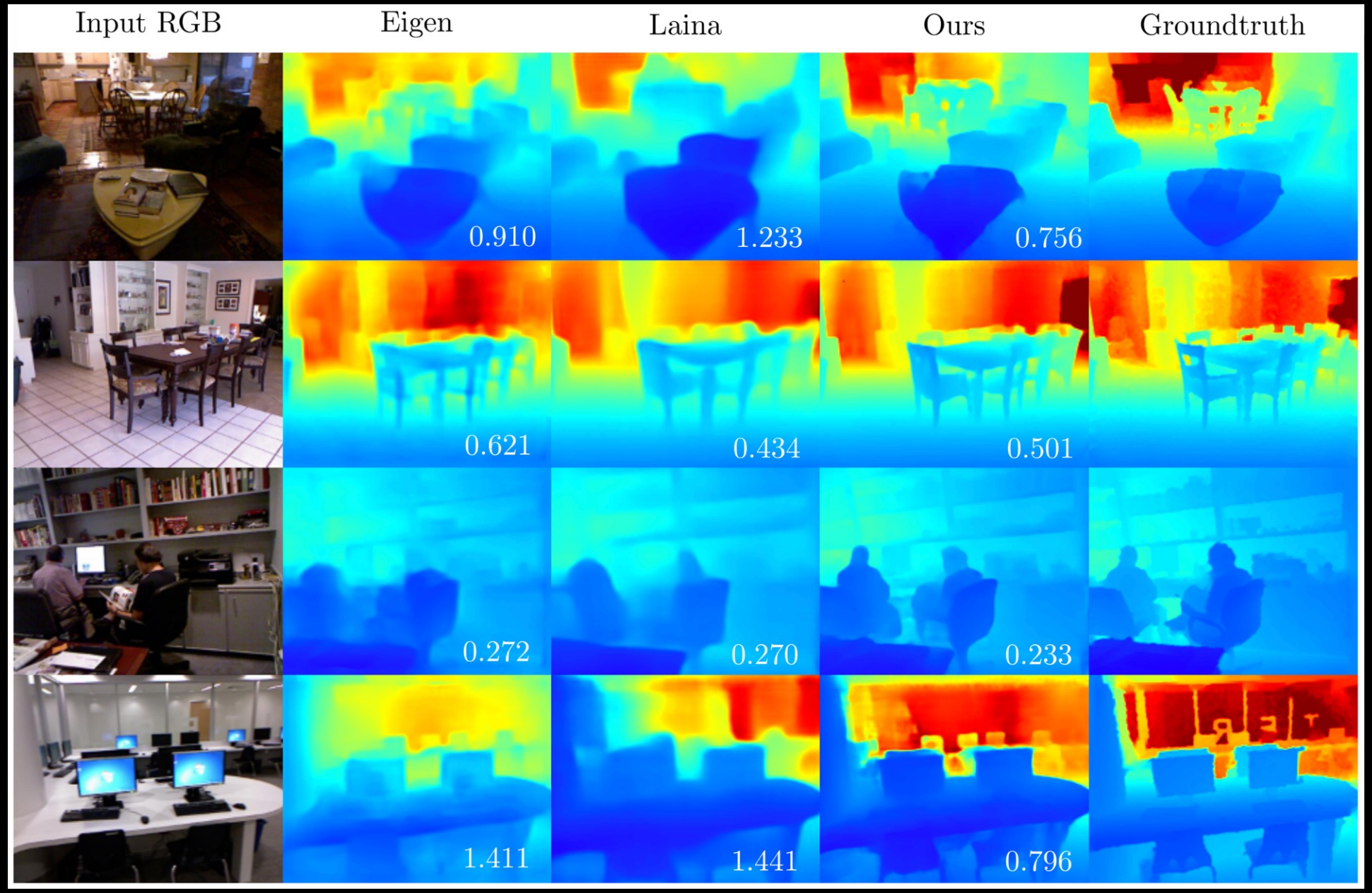
Q: How can we learn σ if we only have ground truth for μ?

Estimating uncertainty in learning

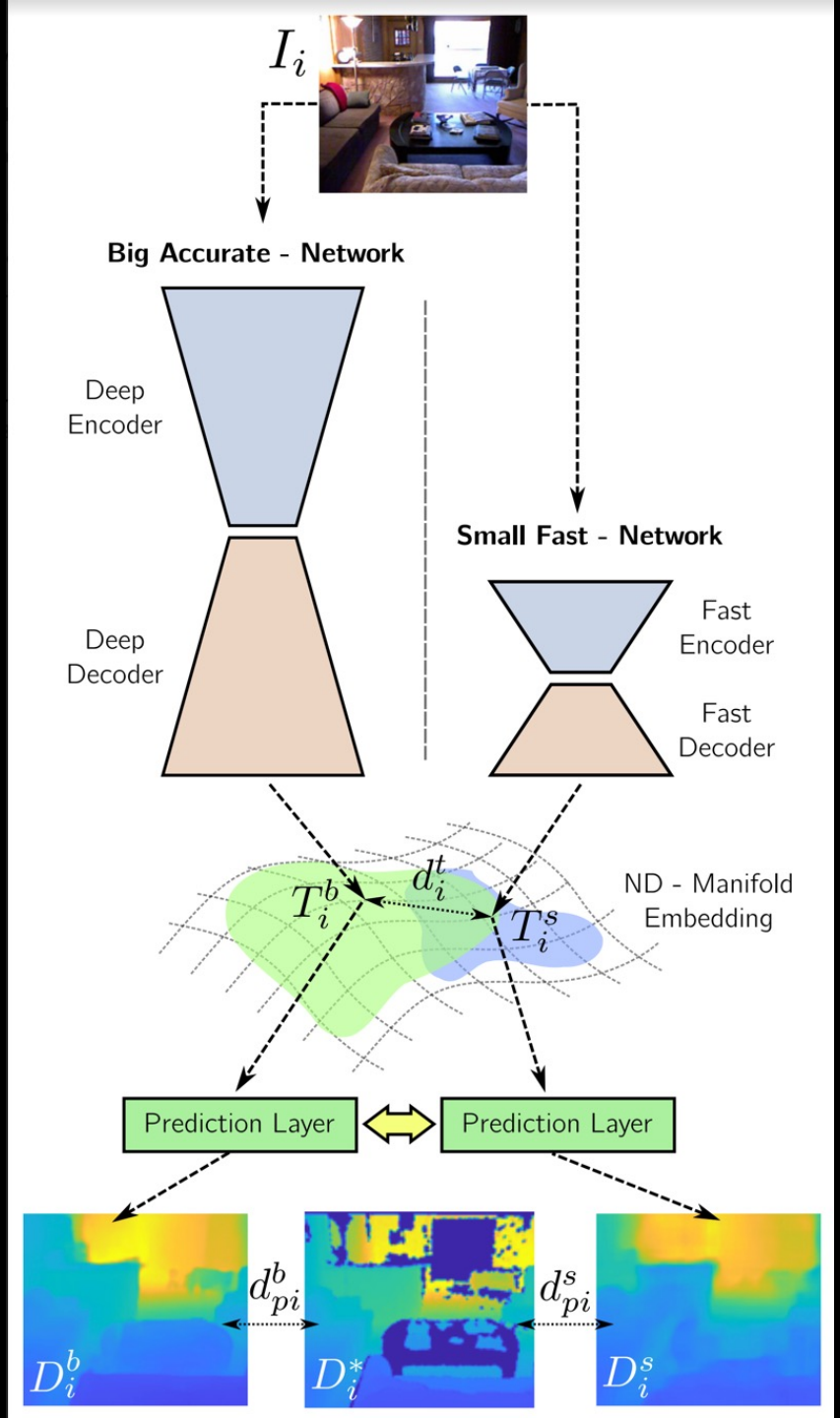
Uncertainty maps reflect the errors



Obligatory comparisons…

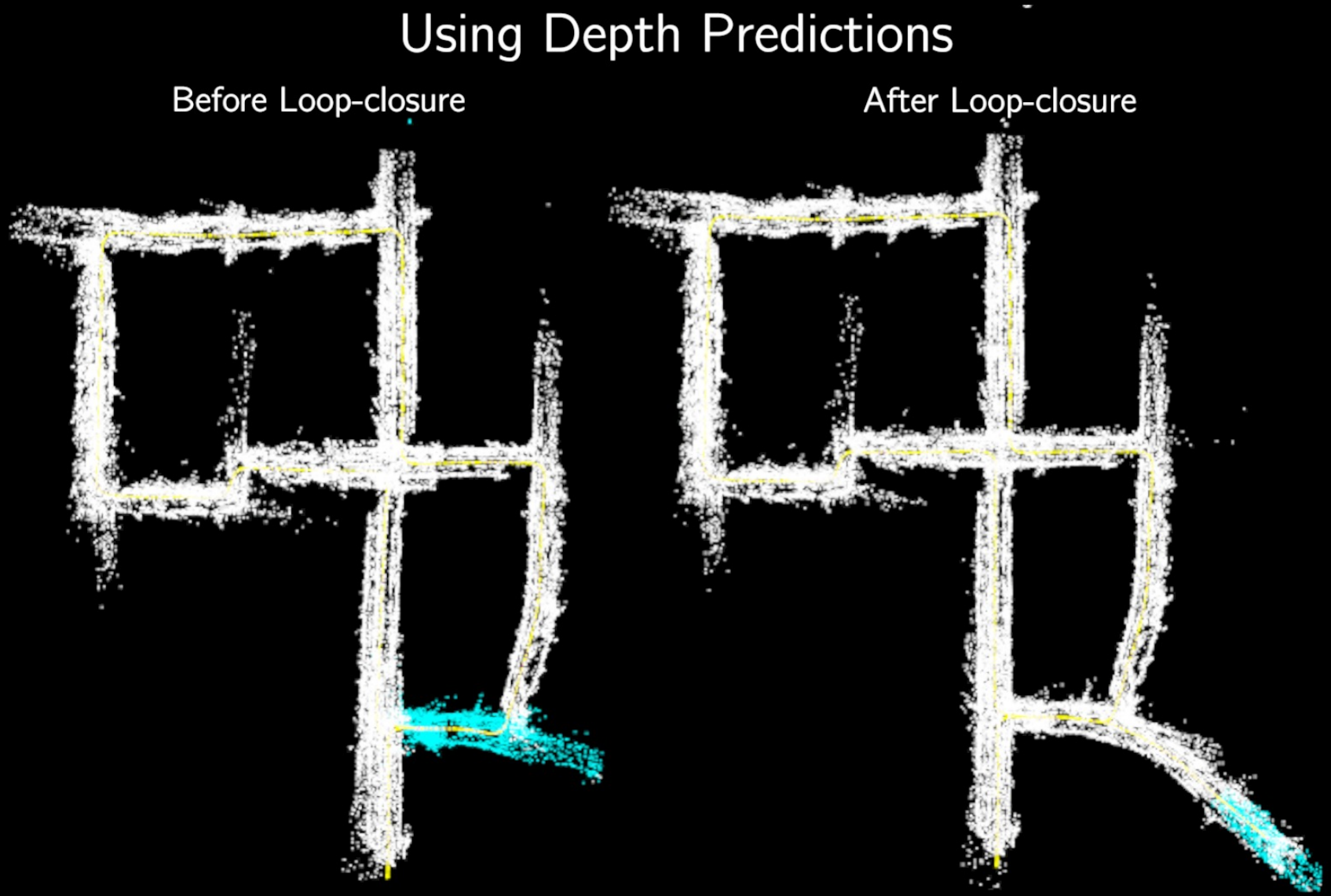


Speeding it up for robotics

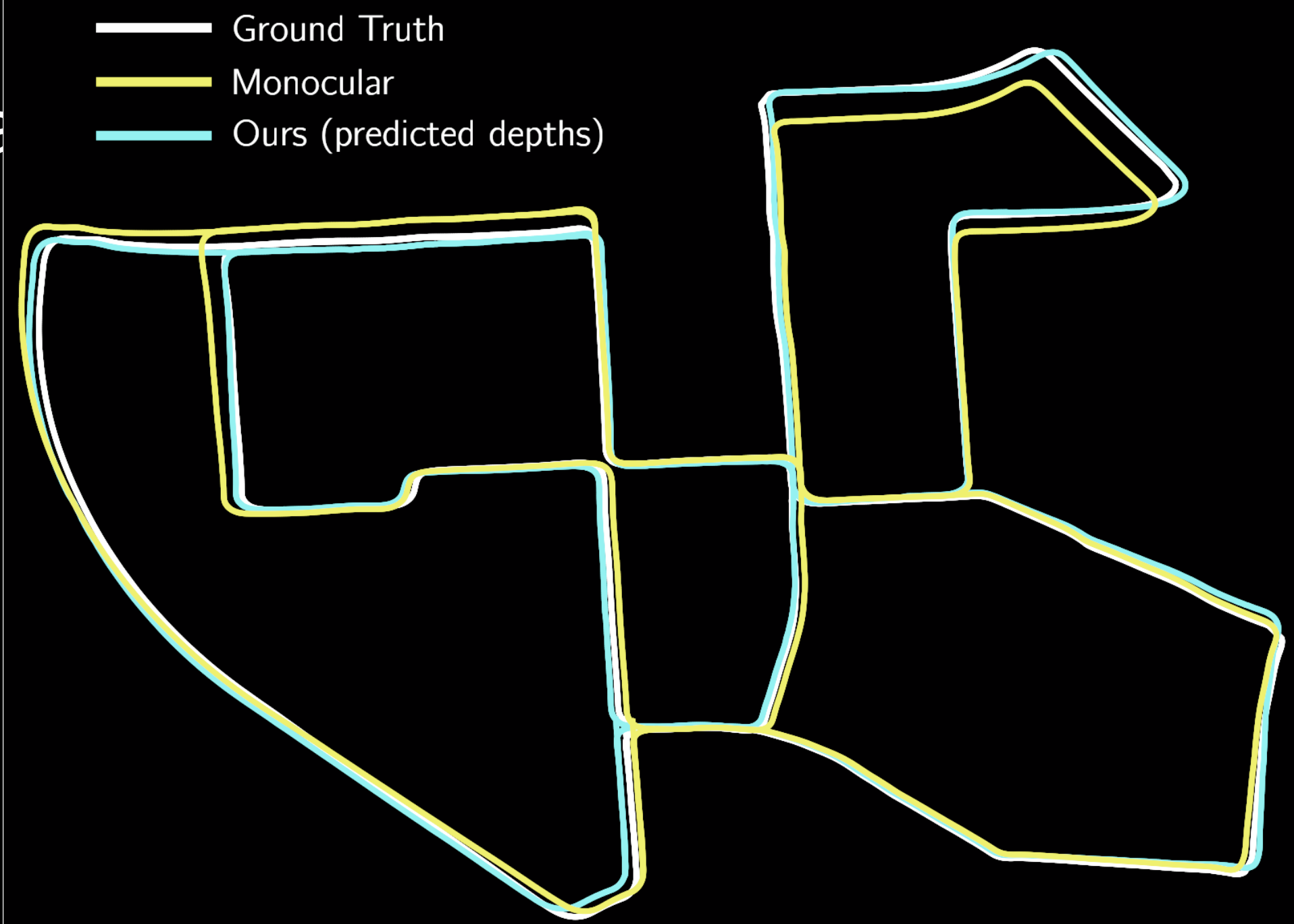


Can use depth to improve ORB SLAM

Can use depth to improve ORB SLAM

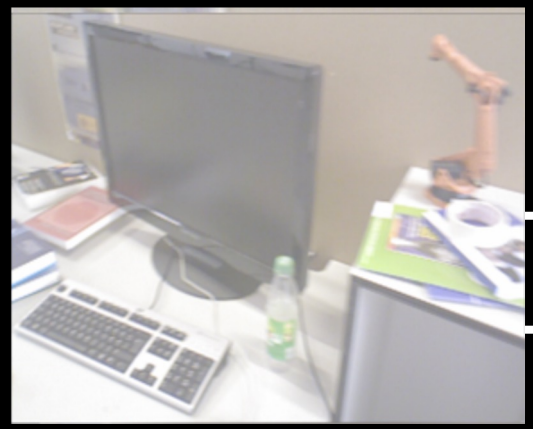


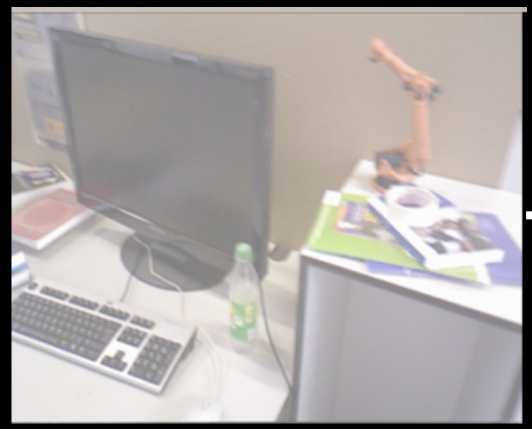
Ca



But we want more deep learning!

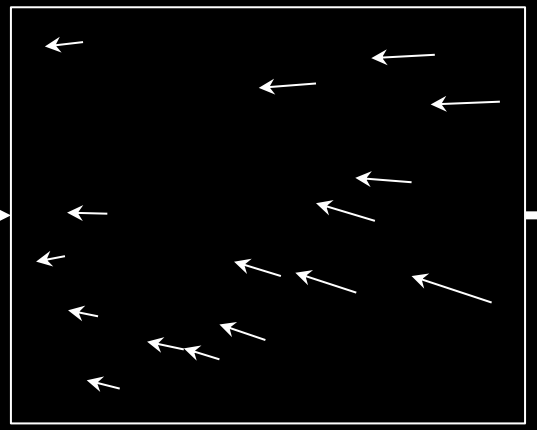
Estimate flow from two images; use this to get camera motion:





Depth estimation network

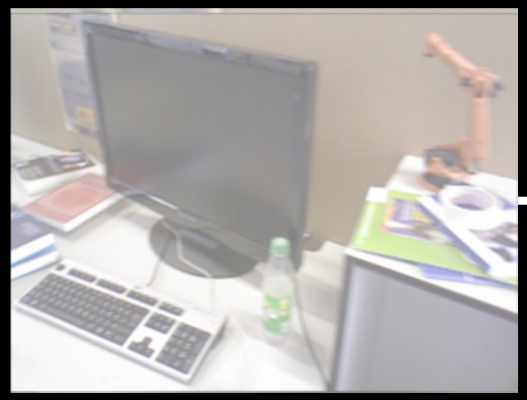
Flow estimation network

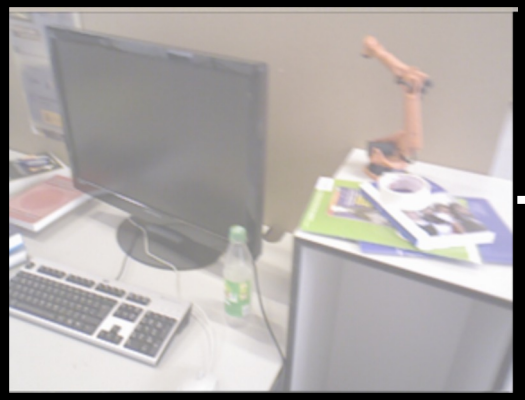


Motion estimation via geometry

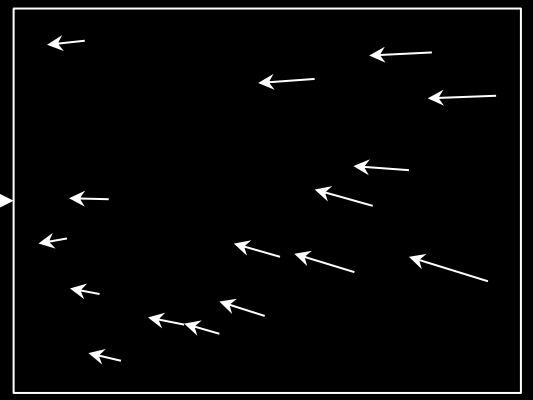
Motion and rotation

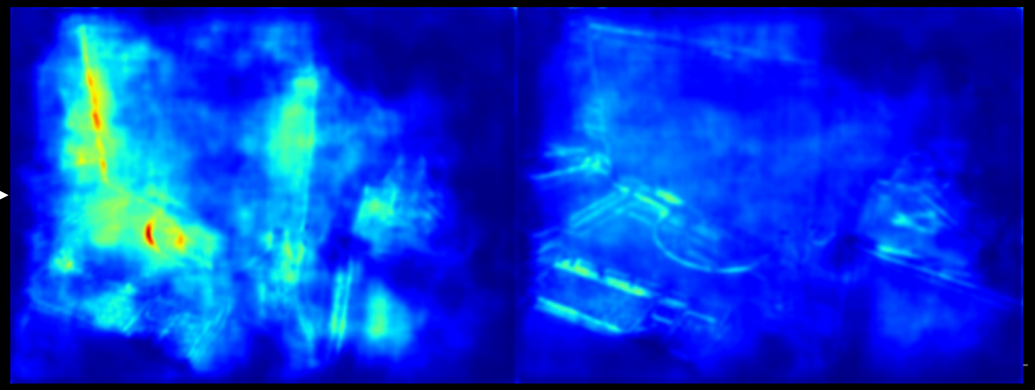
Better if uncertainty in flow is incorporated too





Flow estimation network



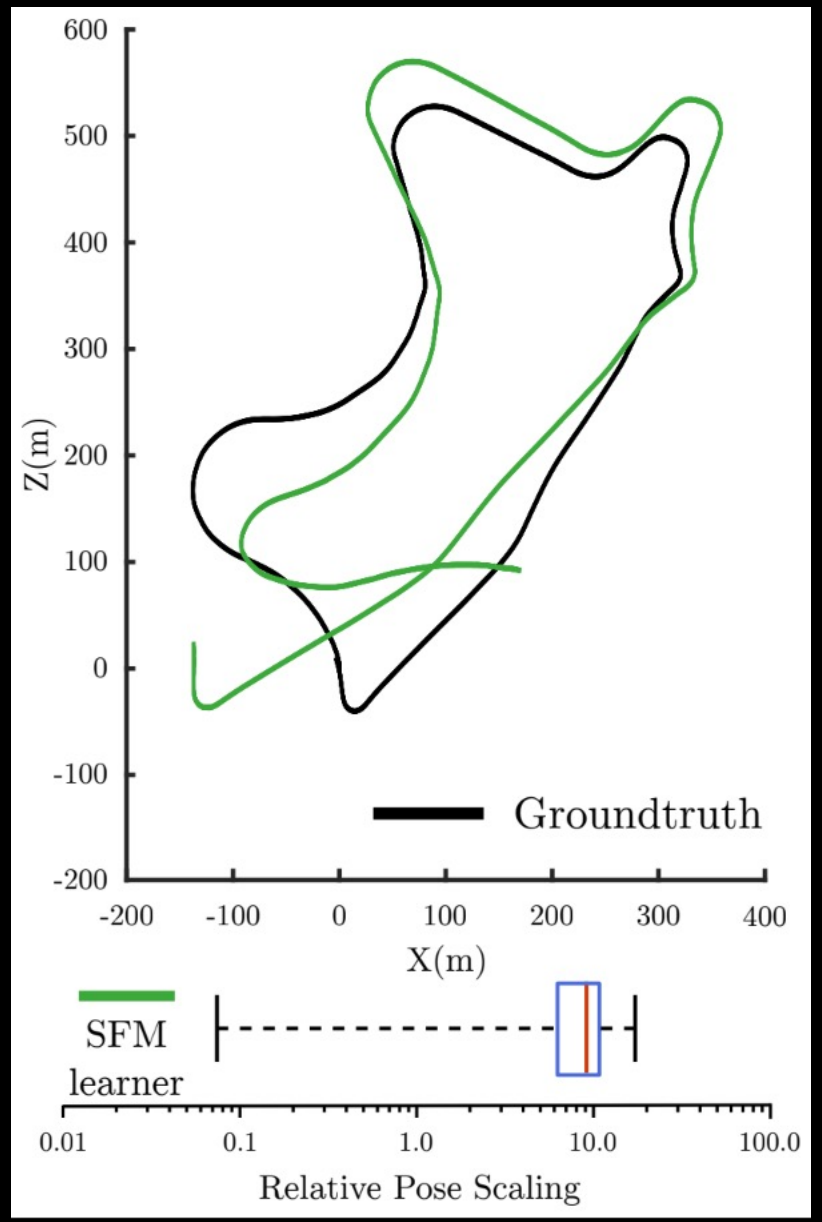


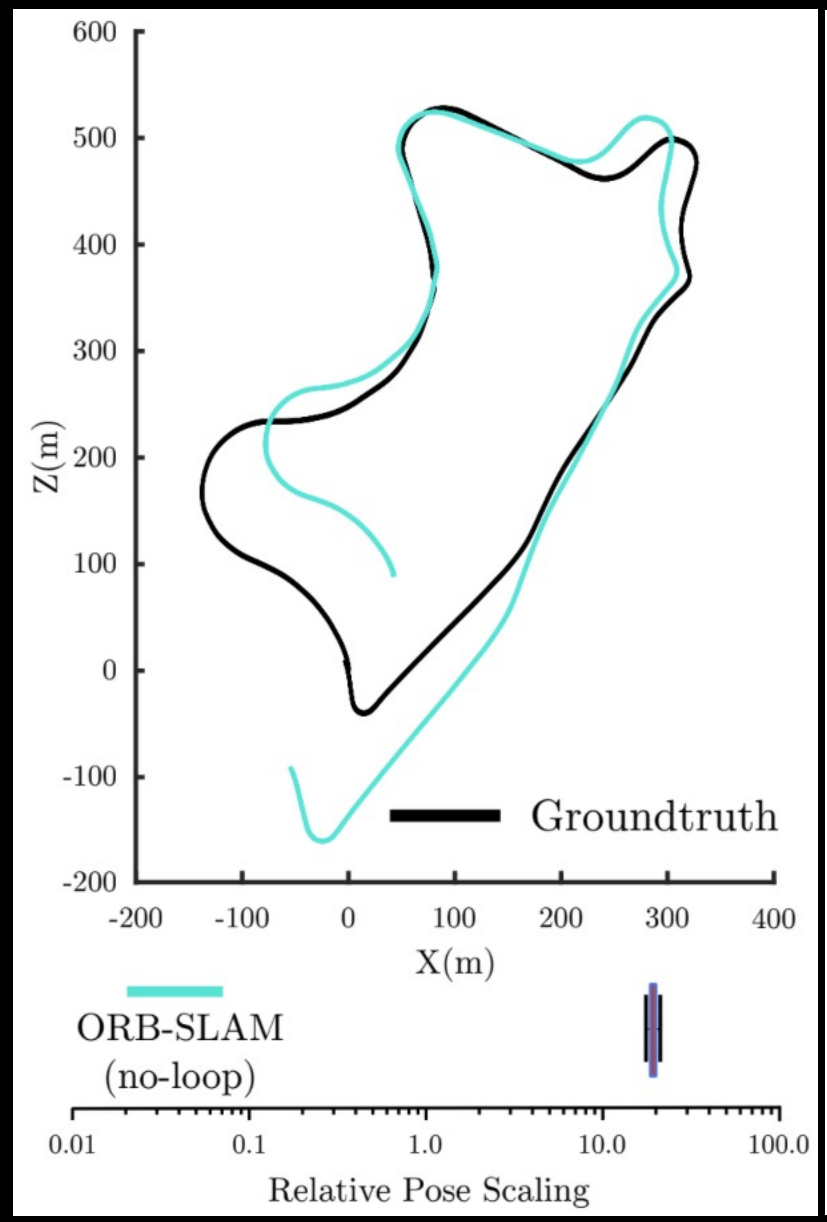
X Y

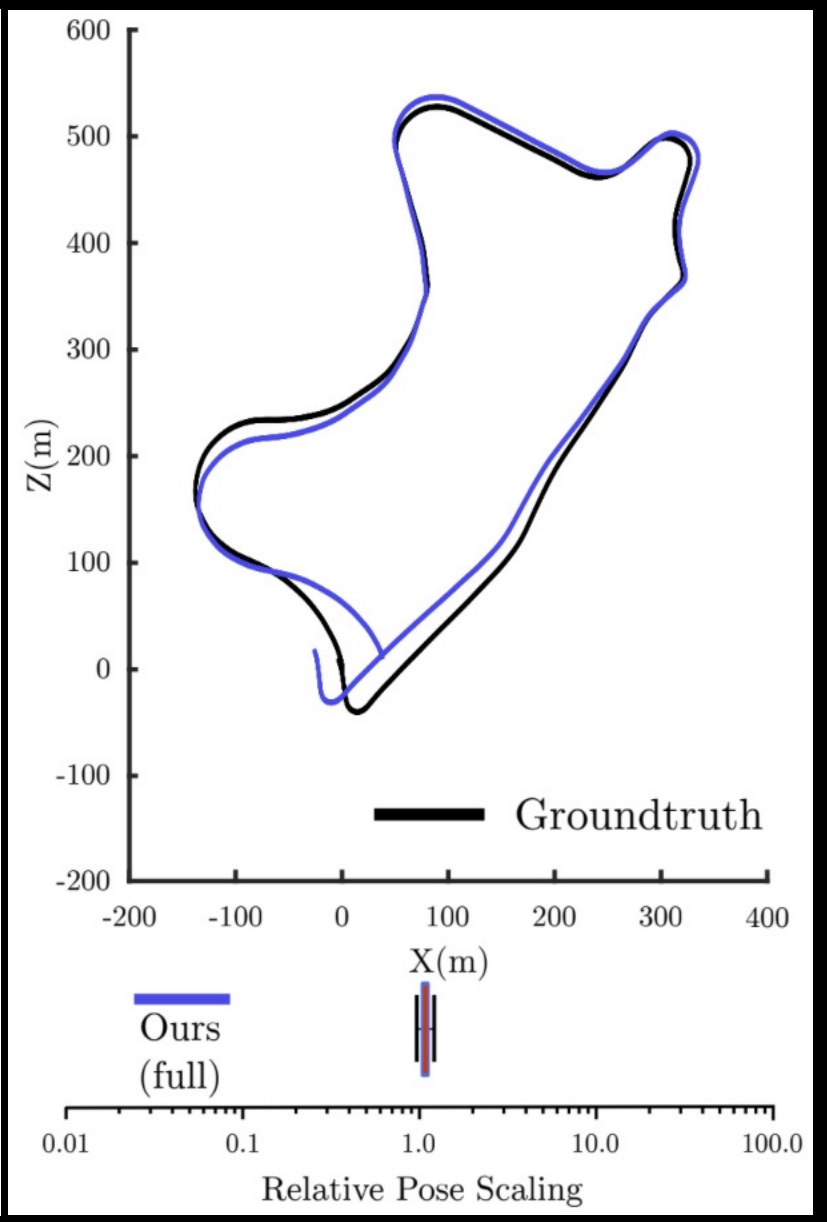
Confidence in flow estimate

Parameterising flow uncertainty

Visual Odometry (just prior to loop closure)





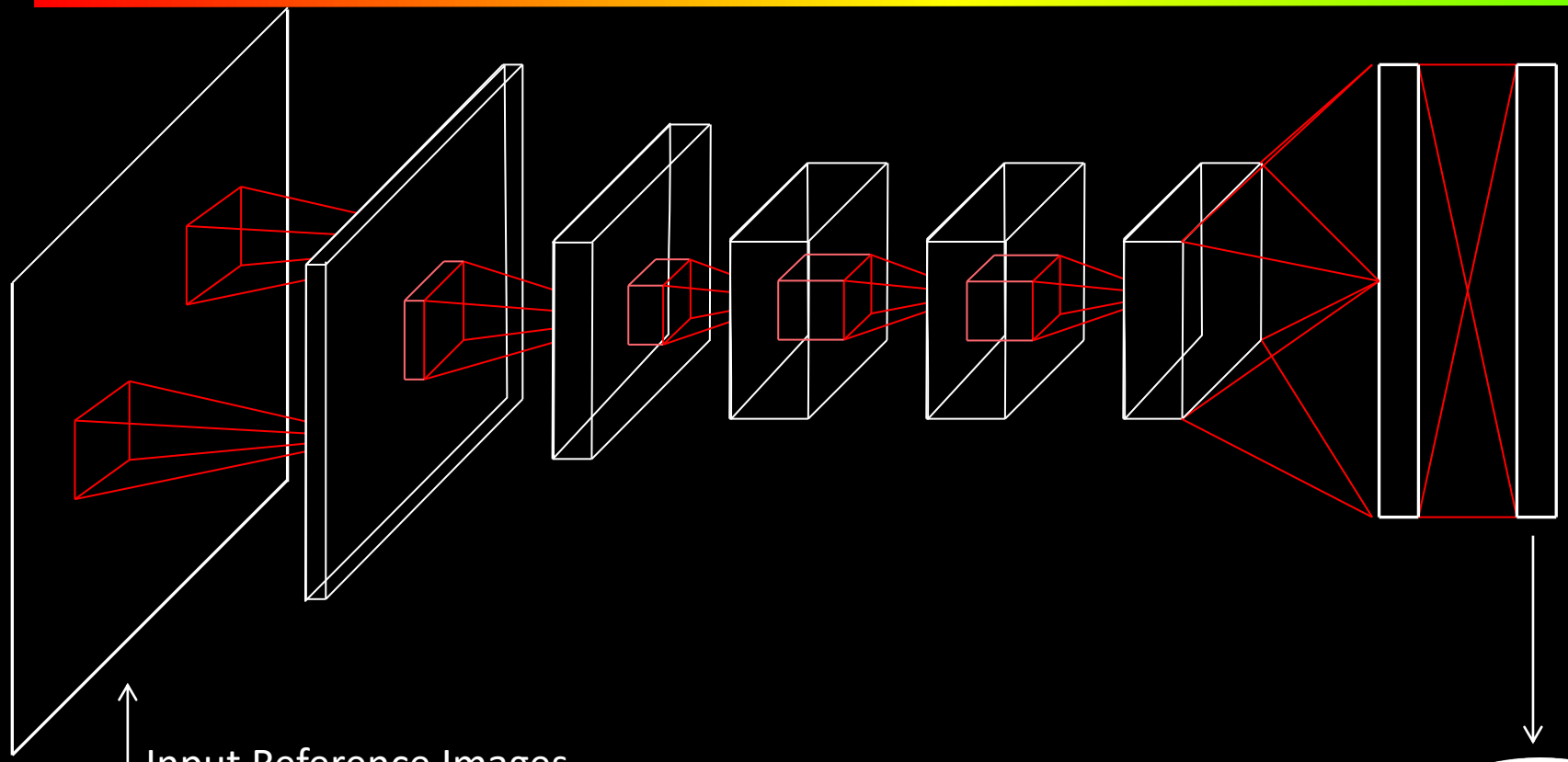


Interestingly, system learns to recognise independently moving objects and give them less confidence

Claim 3:

You should know the difference between
Aleatoric and Epistemic uncertainty

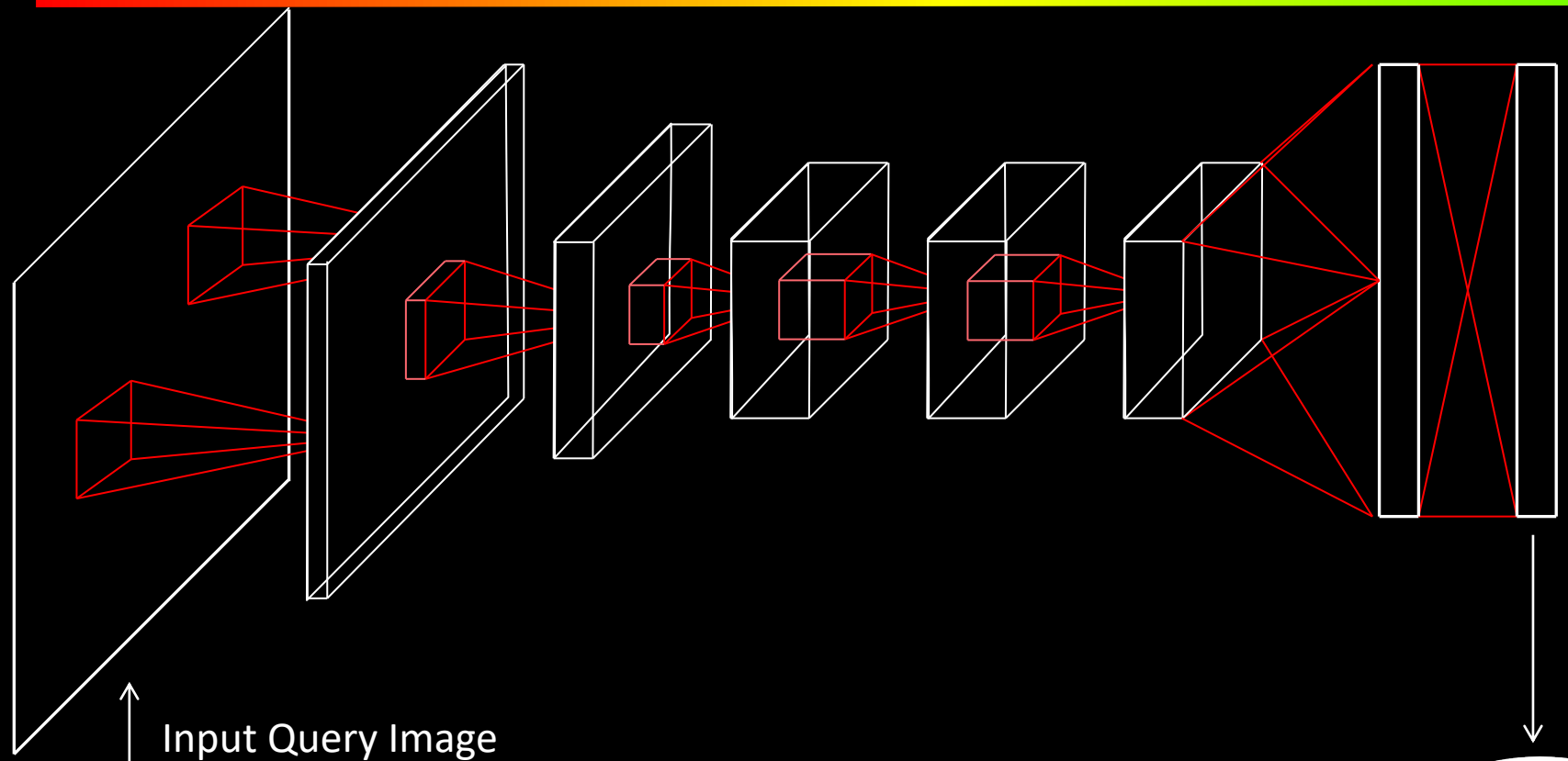
Representation learning



Input Reference Images

Database

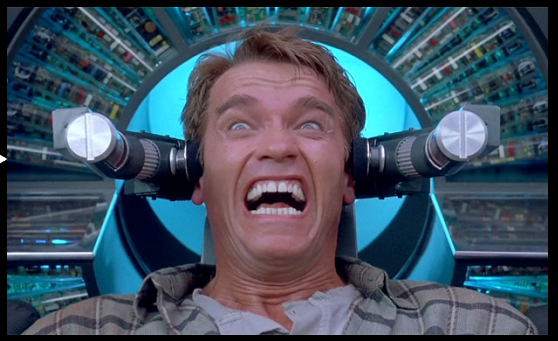
Representation learning



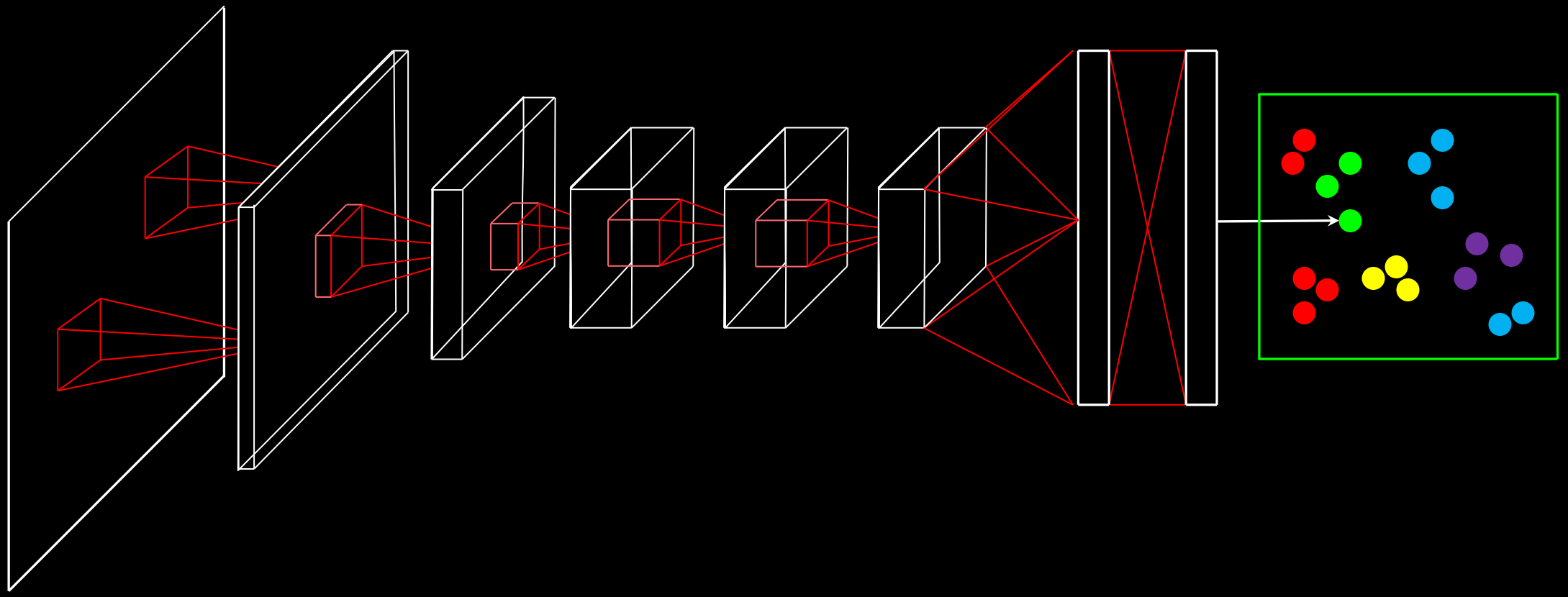
Look up nearby images
in database

Input Query Image

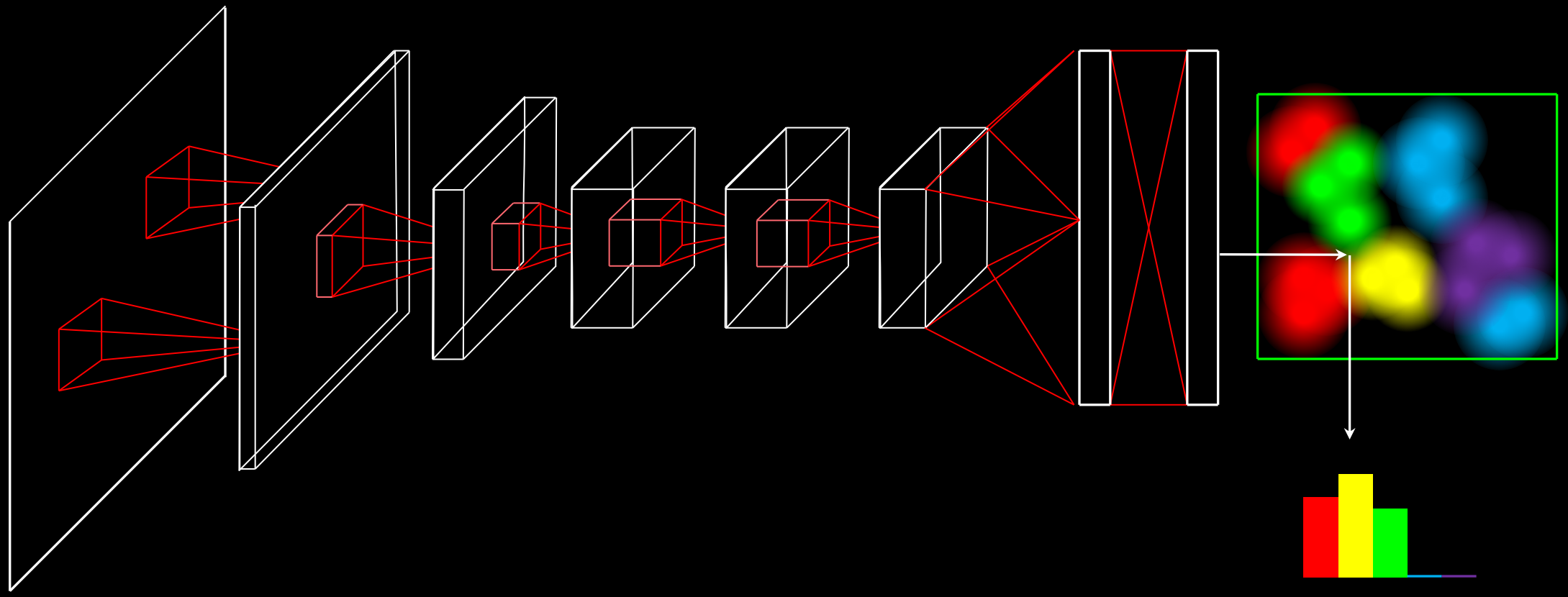
Database



Replace softmax with episodic memory using FANNG

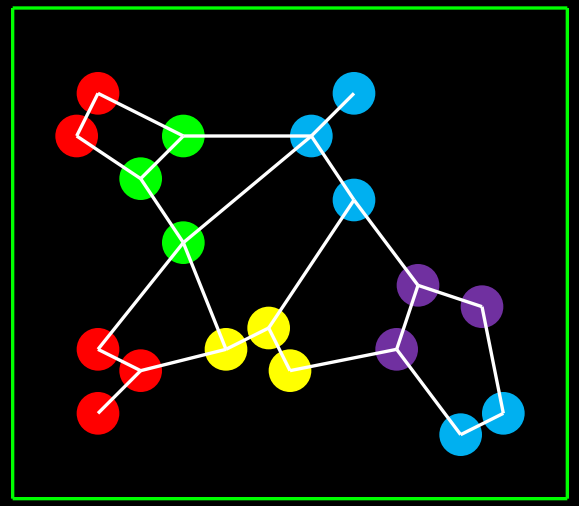


Use Marginal density of Gaussian kernels



Cross entropy loss pushes examples uphill

FANNG: Fast Approximate Nearest Neighbour Graphs



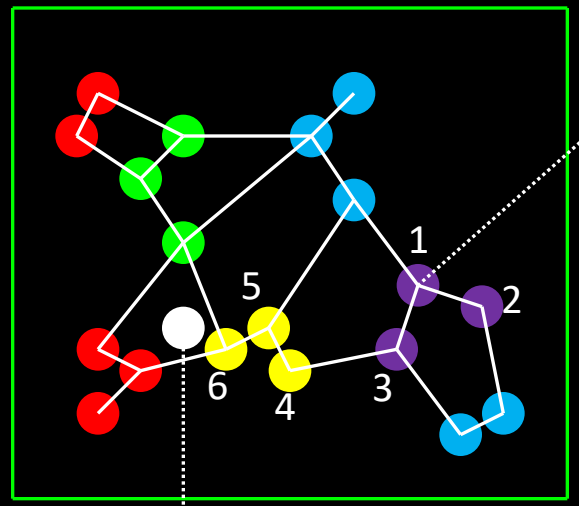
Based on two insights:

1. If a reference point is near a query point, it's worth checking its neighbors to see if they're closer
   - build a graph linking points to their neighbors

2. If there are many neighbors in the same direction it's not necessary to have edges to all of them
   - if the nearest of them isn't closer to the query then the others probably aren't either.

FANNG Algorithm

6
5
4
3
1
2



Start somewhere in the graph

While (not bored) {

Measure distance to query

Insert this point into a priority queue* of vertices visited

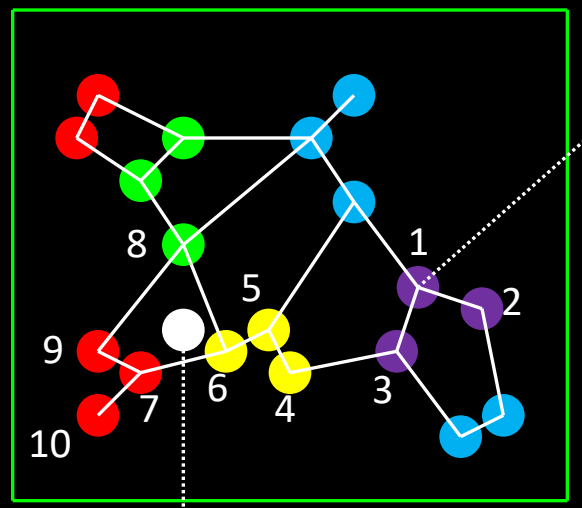
Find the nearest unvisited neighbor of the highest priority vertex with unvisited neighbors

}

*in order of distance from query

FANNG Algorithm

6 7 5 8 9 4 10 3 1 2



Query point

Start somewhere in the graph

While (not bored) {

Measure distance to target (query point)

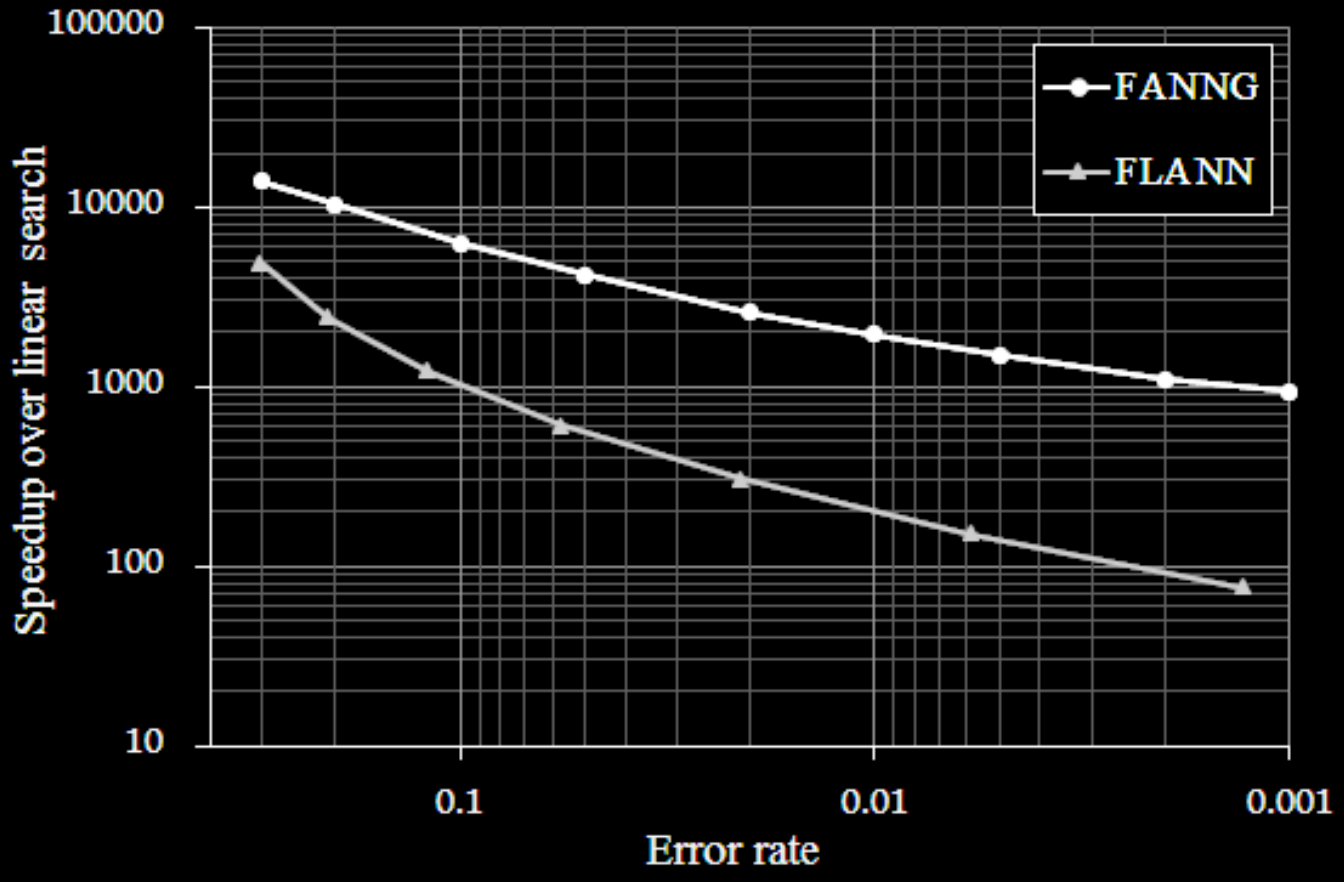
Insert this point into a priority queue* of vertices visited

Find the nearest unvisited neighbor of the highest priority vertex with unvisited neighbors

}

*in order of distance from target

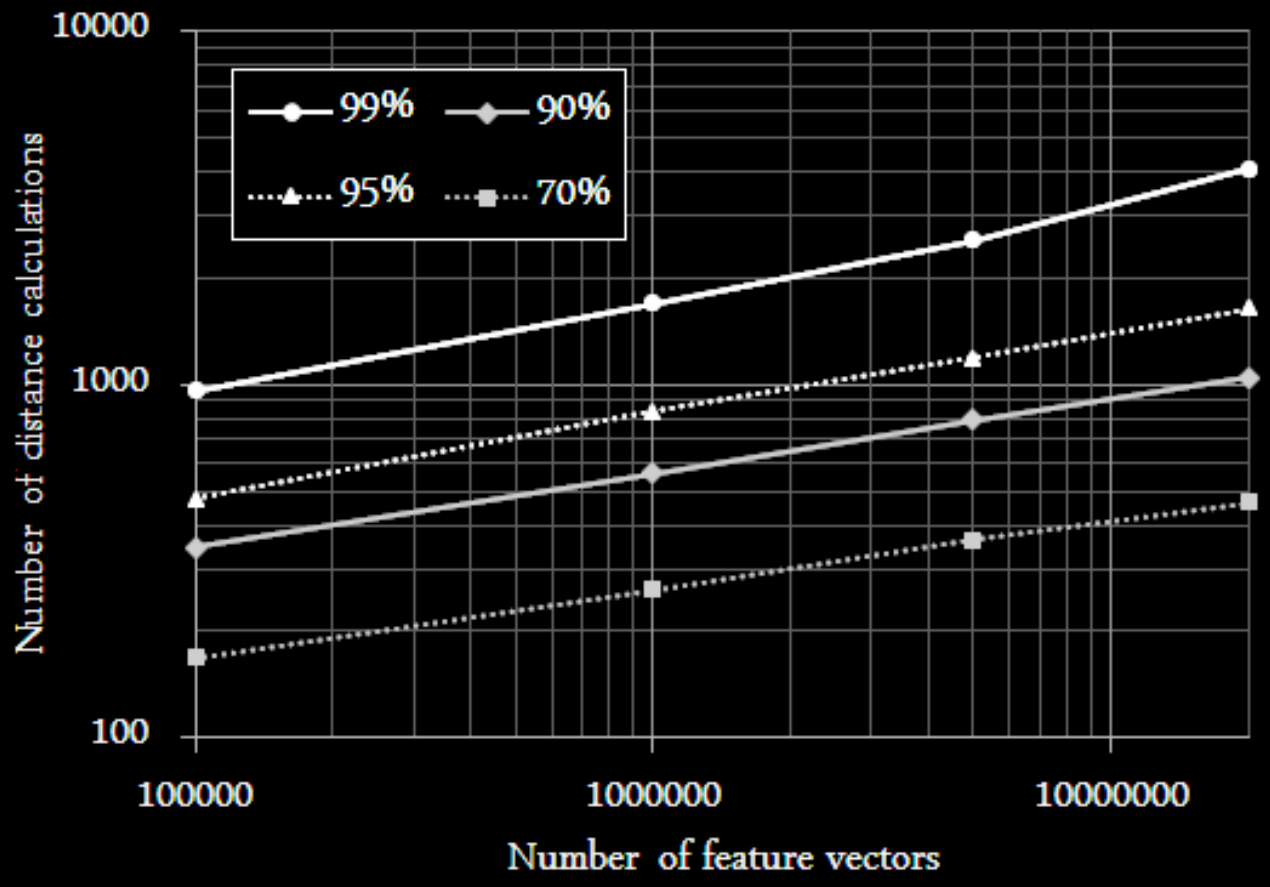
FANNG Performance



FANNG can find the nearest neighbor with 90% accuracy in 1M examples of SIFT descriptors at rate of 10M queries/s – or at 99.7% accuracy at 1M queries/s

It can also return K nearest neighbors (the highest K entries in the Priority Queue when searching is finished)

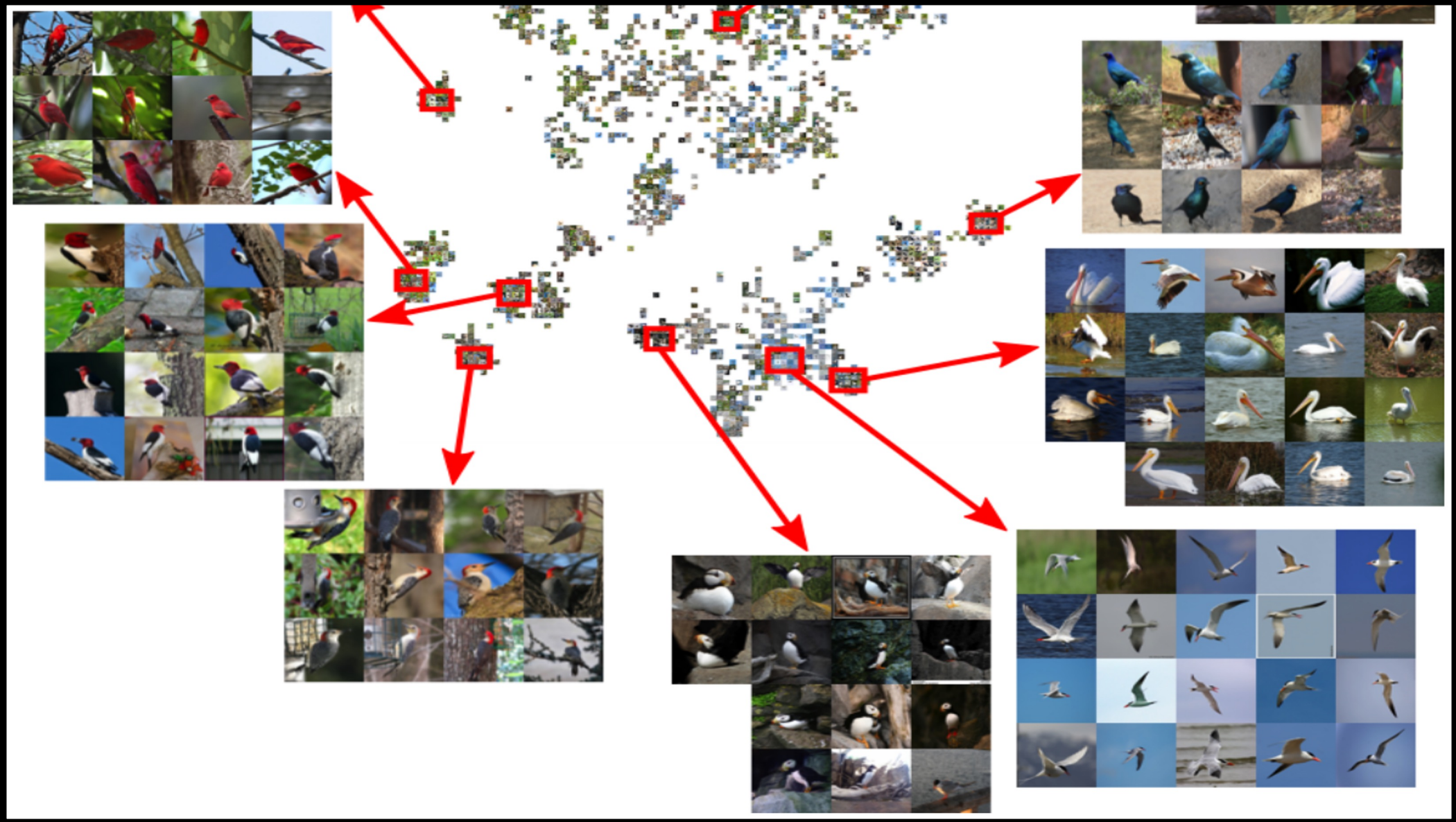
FANNG Performance



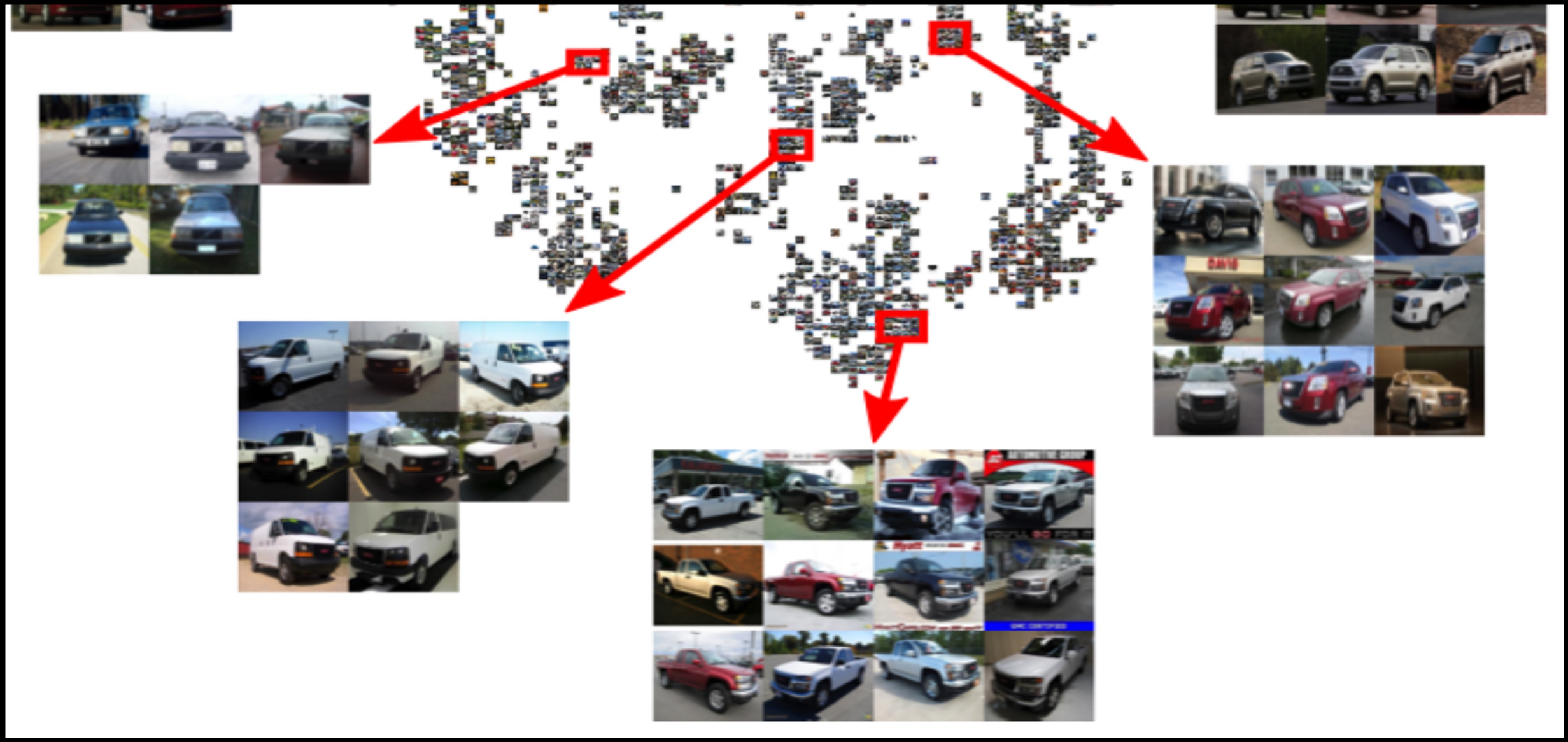
For 128D SIFT data
search time scales as 0.2 power of data set size:

$$t \propto N^{0.2}$$

Learned embedding transfers well



Learned embedding transfers well



Novel class detection / active learning