TESLA AUTOPILOT

BREAKING DOWN TESLA'S AI DAY PRESENTATIONS (PART 1)

Gene Olafsen

OVERVIEW

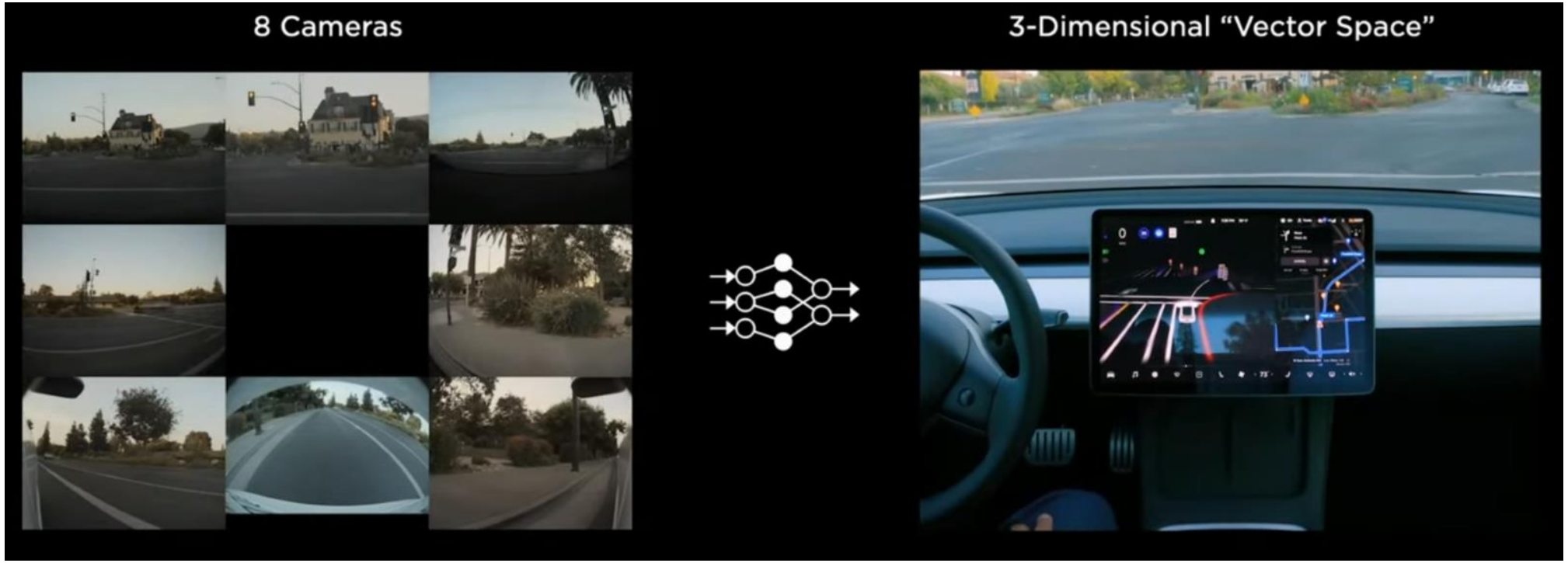
- Part I
  - Tesla Vision
  - Planning and Control
- Part II
  - Manual Labeling
  - Auto Labeling
  - Simulation
- Part II
  - HW Integration
  - Dojo

ACKNOWLEDGEMENT AND REFERENCE

- The screenshots and content is largely taken from the AI Day presentations.
- Tesla AI Day - YouTube
- 1612.03144.pdf (arxiv.org)

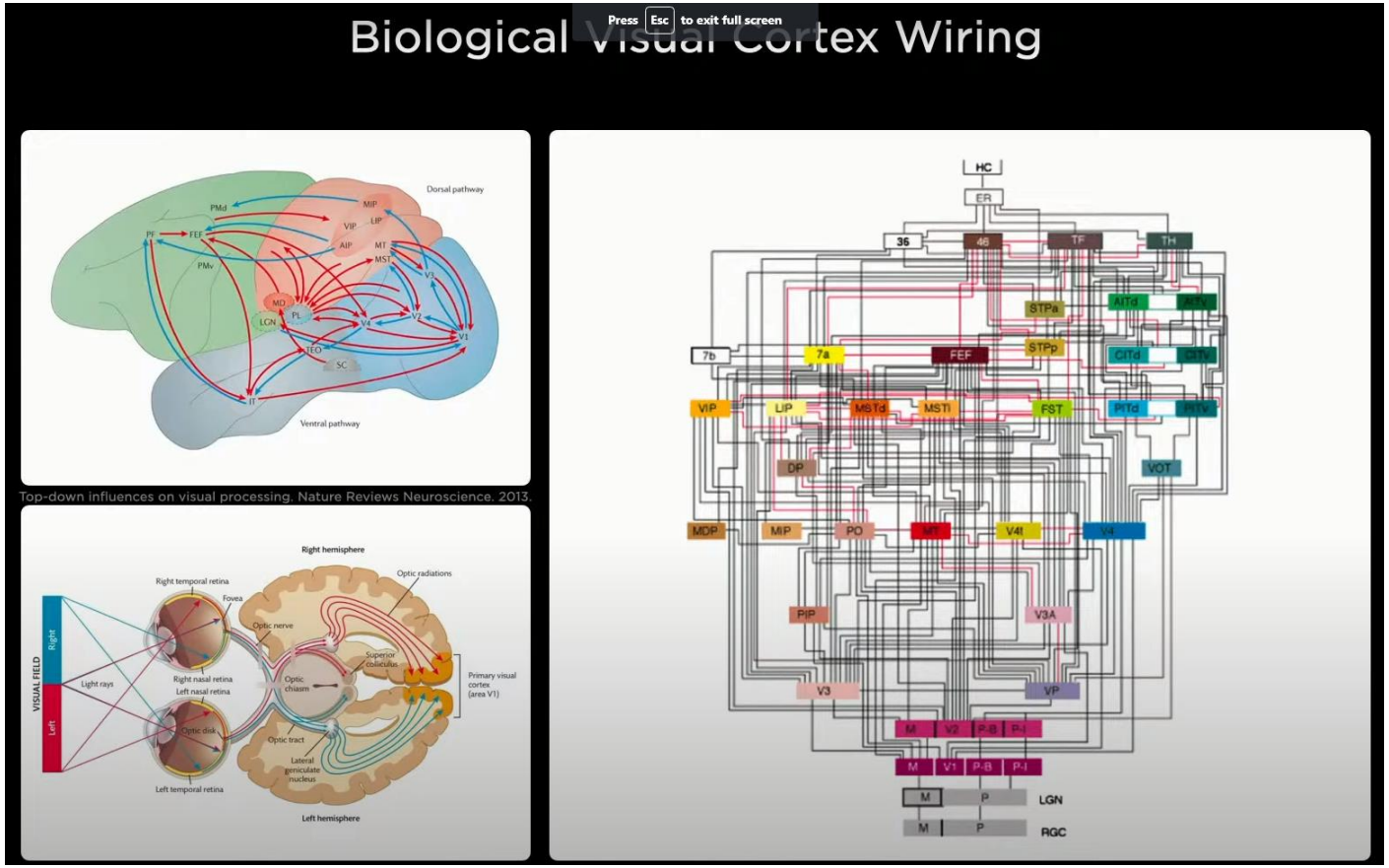
DEEP DIVE INTO THE AUTOPILOT STACK

- 8 cameras around the vehicle
- Process in real time into what Tesla calls the "Vector Space" - a 3D representation of everything that is 'seen'. Object position, orientation, depth, velocity, etc.



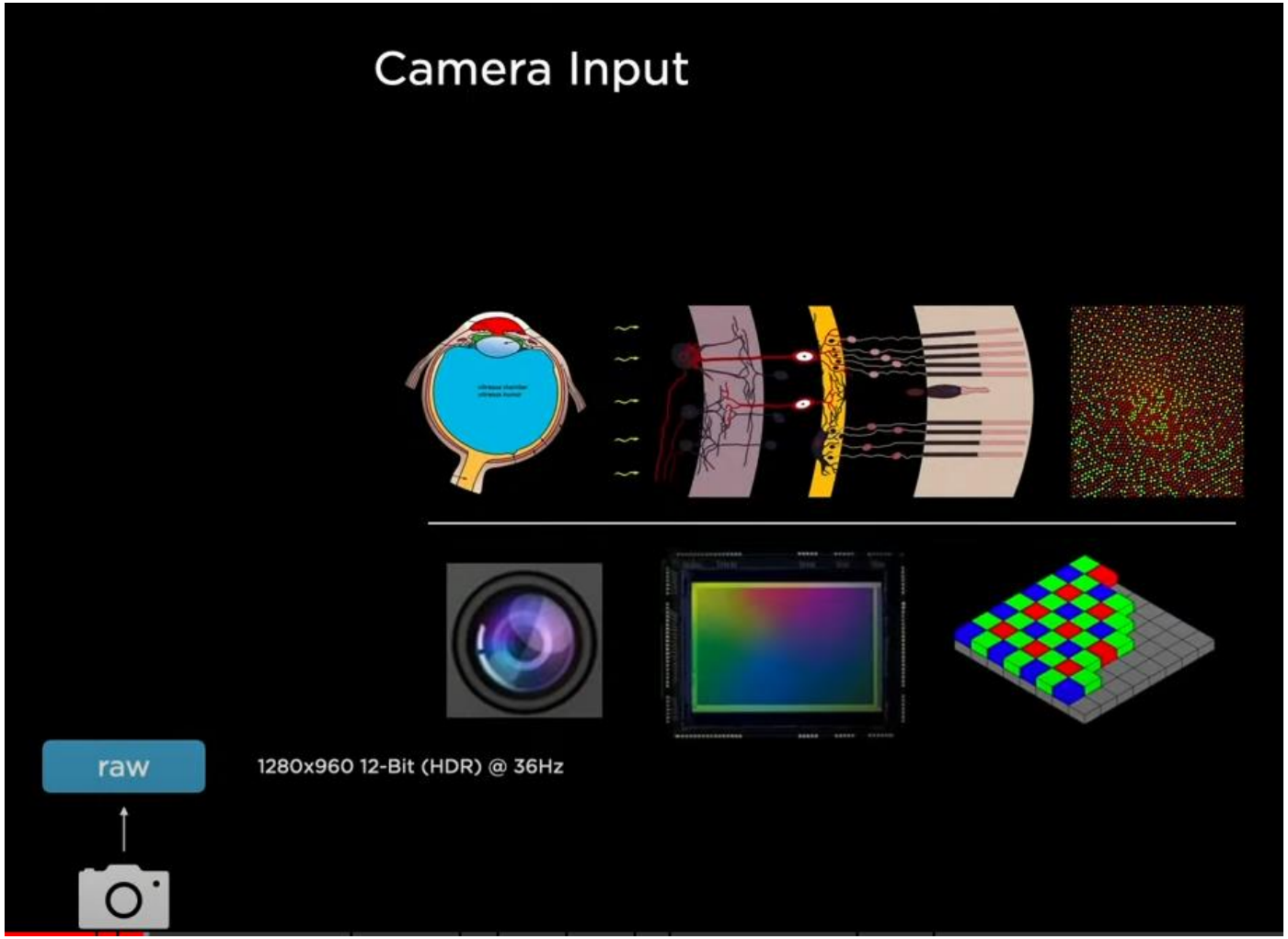
SYNTHETIC ANIMAL

- Building a synthetic animal from the ground up. Moves around, senses the environment, moves autonomously, acts intelligently.
- Building everything in house.
- Synthetic visual cortex.



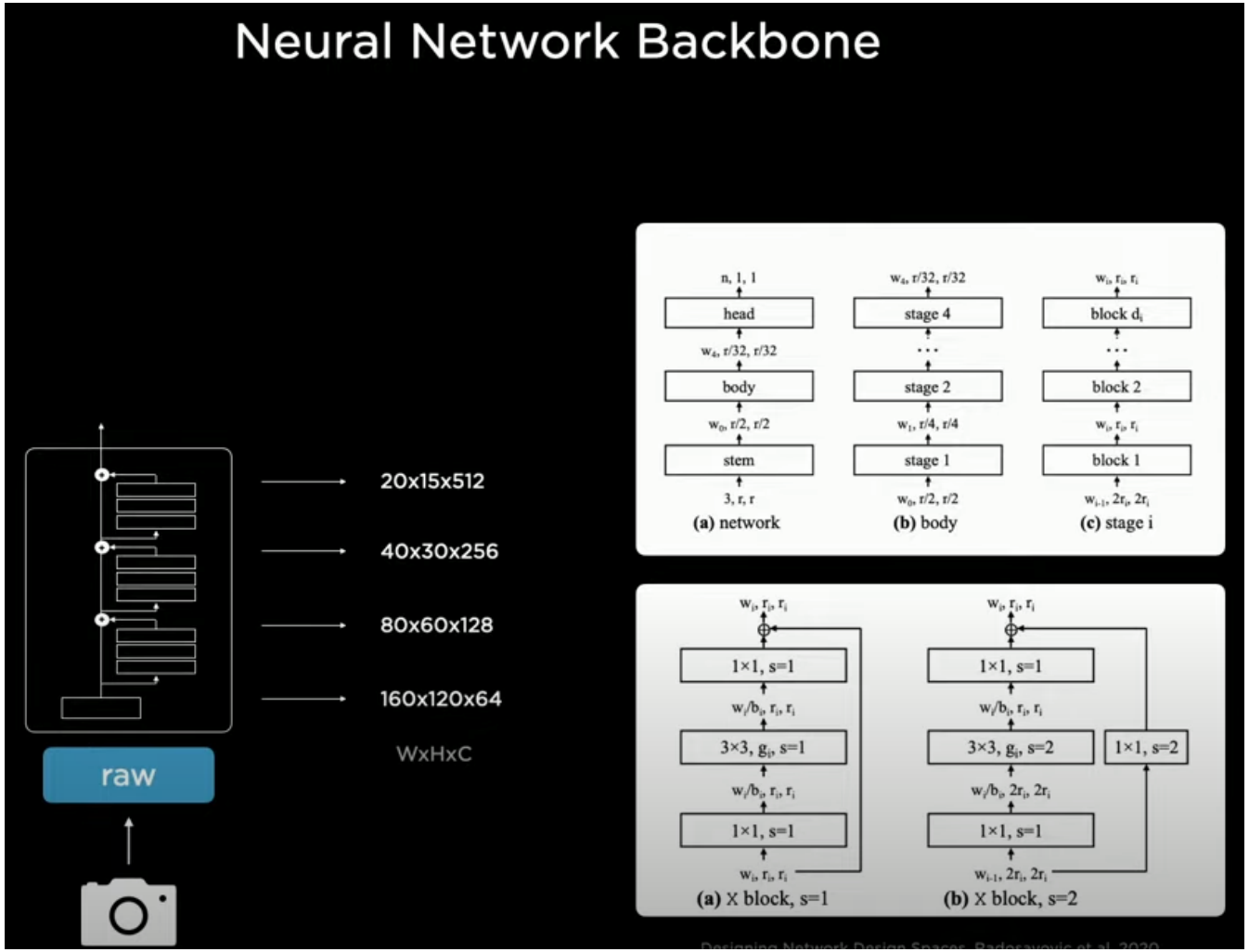
CHRONOLOGY OF AUTOPILOT DEVELOPMENT

- Four years ago, the car was driving a single lane and keeping a following distance, all processing done on a single image



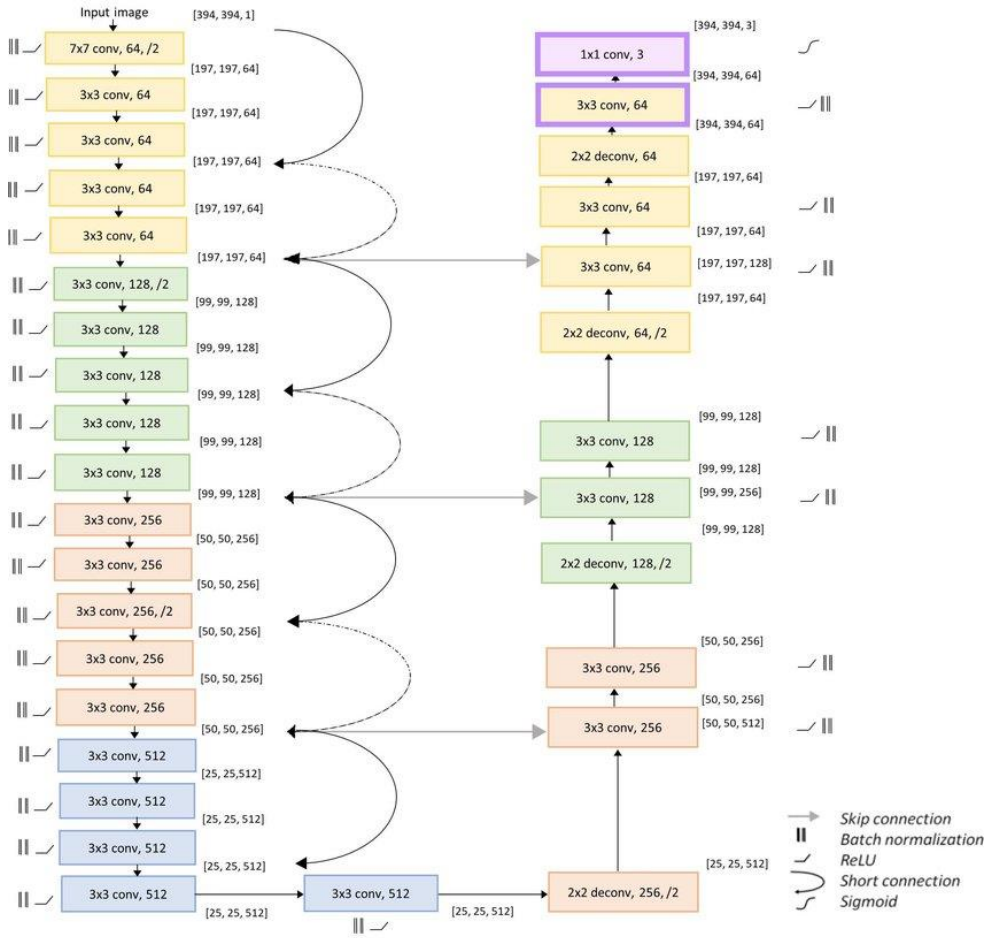
NN BACKBONE

- Leverages RESNets – allow you to trade off latency and accuracy, they provide features at different scales
- Starting at the bottom with high resolution and low channel counts – can see most of the image and have a lot of scene context
- Top – low resolution high channel counts – a lot of neurons scrutinize the image details

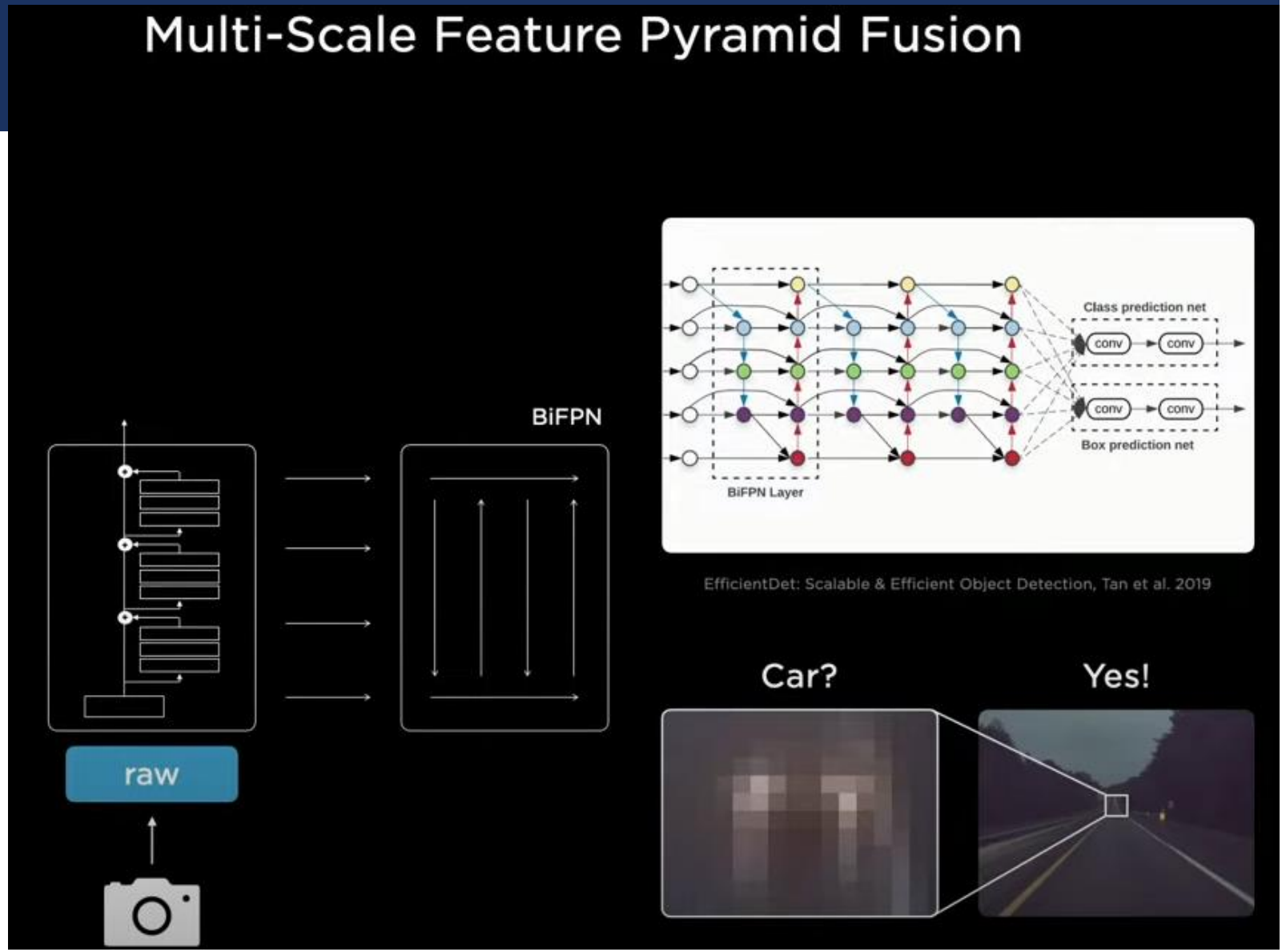


RESNETS AND SKIP CONNECTIONS

- ResNet, short for Residual Networks is a classic neural network used as a backbone for many computer vision tasks. This model was the winner of ImageNet challenge in 2015.
  - The fundamental breakthrough with ResNet was it allowed the successful training of extremely deep neural networks with 150+layers.
- Instead of treating number of layers an important hyperparameter to tune, by adding skip connections to the network, this allowed the network to skip training for the layers that are not useful and do not add value in overall accuracy.

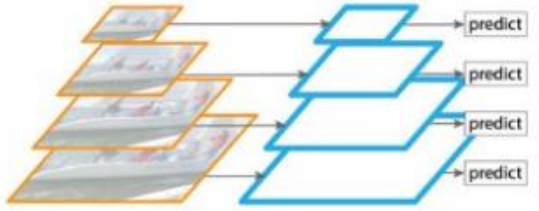


- Then processed with ‘Feature Pyramid Networks’ BiFPN
- Gets the multiple scales to share information
- If you are at the bottom of the network and you see a small patch of pixels and not sure if it is a car, the top of the network may be able to inform that the pixels are at the vanishing point of the highway. This is probably a car.

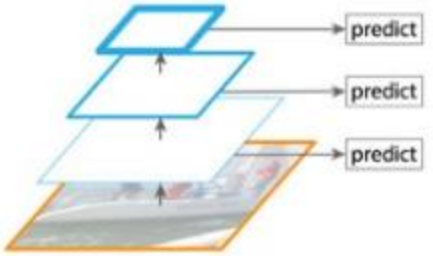


FIRST THINGS FIRST – FEATURE PYRAMID NETWORKS (FPN)

- FPN problem statement: Detecting objects in different scales is challenging, particularly for small objects.
- A 'brute force' approach, would be to take the same image at different scales and train for image detection.
  - Compute, memory and time to train all the nextworks end-to-end simultaneously.
- Alternately, we could create a pyramid of features and use them for object detection.
  - However, feature maps closer to the image layer composed of low-level structures that are not effective for accurate object detection.



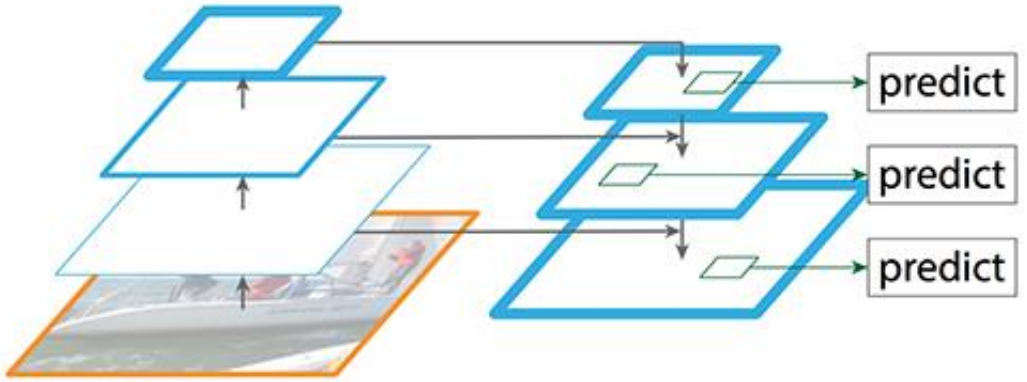
Pyramid of images



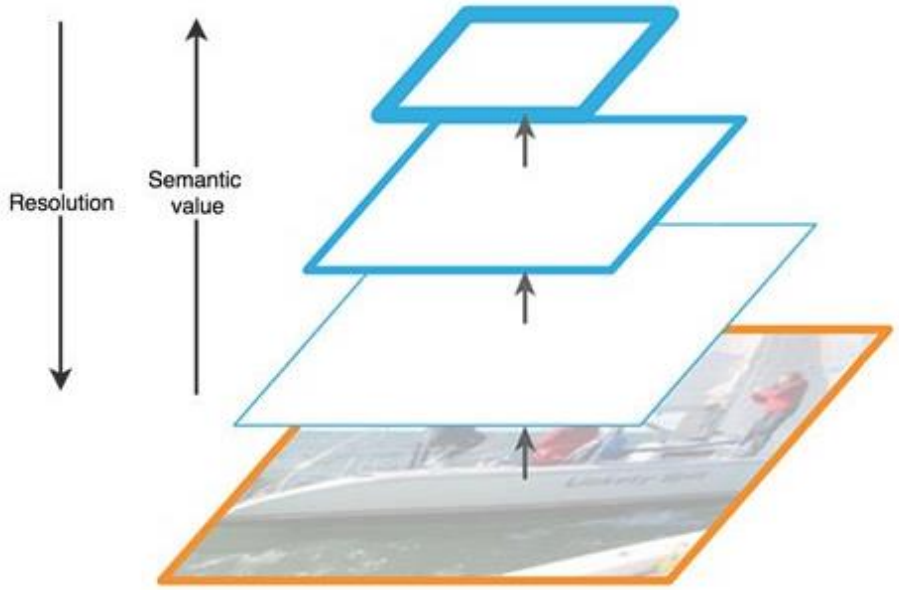
Pyramid of feature maps

MULTI-SCALE FEATURE MAPS

- Feature Pyramid Network (FPN) is a feature extractor designed for the multi-scale concept with accuracy and speed in mind.
  - (FPN is not an object detector by itself. It is a feature extractor that works with object detectors.)
- FPN replaces the feature extractor of detectors like Faster R-CNN and generates multiple feature map layers (multi-scale feature maps) with better quality information than the regular feature pyramid for object detection.

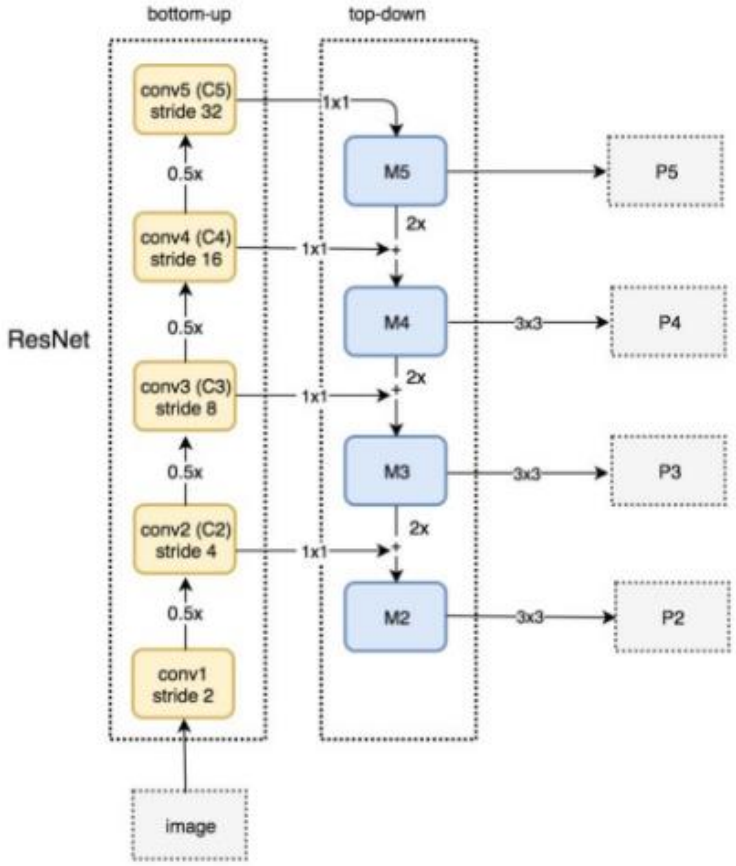


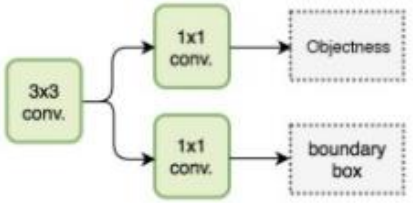
SEMANTIC VALUE VS RESOLUTION



- FPN composes of a bottom-up and a top-down pathway.
- The bottom-up pathway is the usual convolutional network for feature extraction. As we go up, the spatial resolution decreases. With more high-level structures detected, the semantic value for each layer increases.

FPN SAMPLE ARCHITECTURE





predictor head

BIFPN

- A BiFPN, or Weighted Bi-directional Feature Pyramid Network, is a type of feature pyramid network which allows easy and fast multi-scale feature fusion.
- Enables information to flow in both the top-down and bottom-up directions, while using regular and efficient connections.
- It also utilizes a fast normalized fusion technique. Traditional approaches usually treat all features input to the FPN equally, even those with different resolutions. However, input features at different resolutions often have unequal contributions to the output features. Thus, the BiFPN adds an additional weight for each input feature allowing the network to learn the importance of each.
- All regular convolutions are also replaced with less expensive depthwise separable convolutions.
- 1911.09070v7.pdf (arxiv.org)

BIFPN DIAGRAM

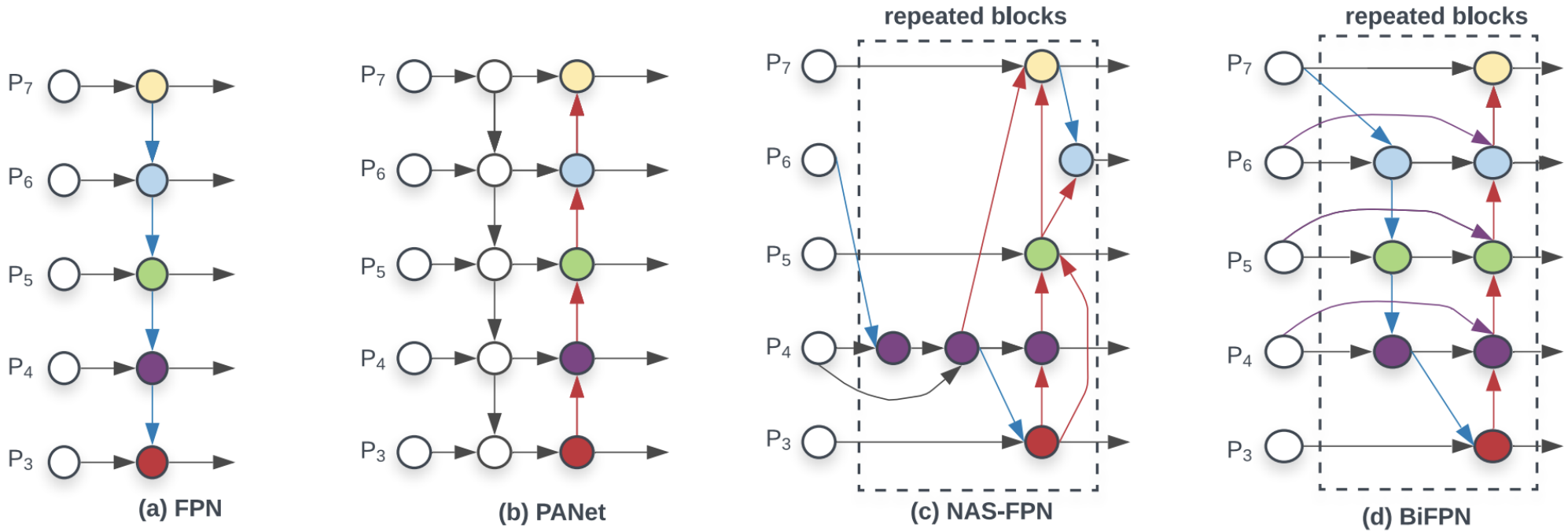
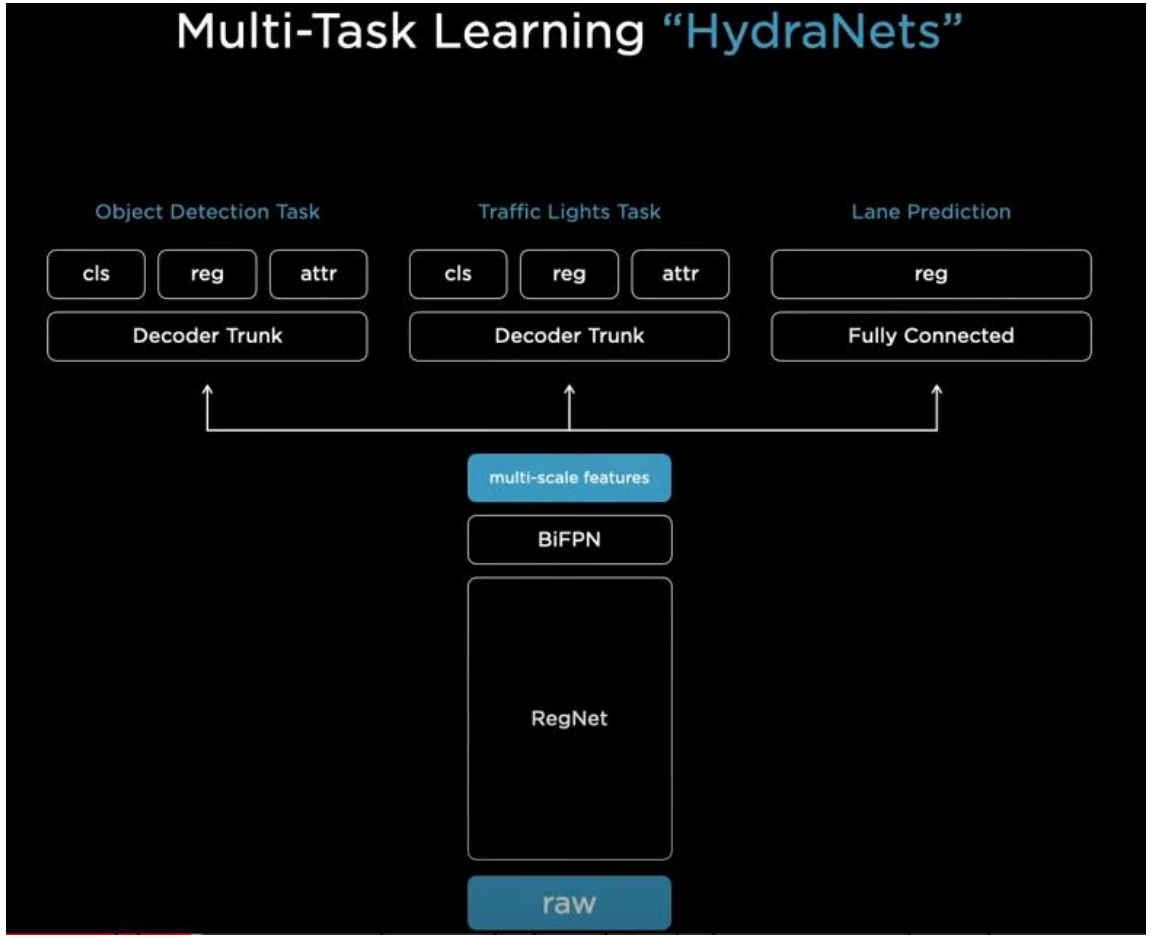


Figure 2: **Feature network design –** (a) FPN [20] introduces a top-down pathway to fuse multi-scale features from level 3 to 7 (P_3 - P_7); (b) PANet [23] adds an additional bottom-up pathway on top of FPN; (c) NAS-FPN [8] use neural architecture search to find an irregular feature network topology and then repeatedly apply the same block; (d) is our BiFPN with better accuracy and efficiency trade-offs.

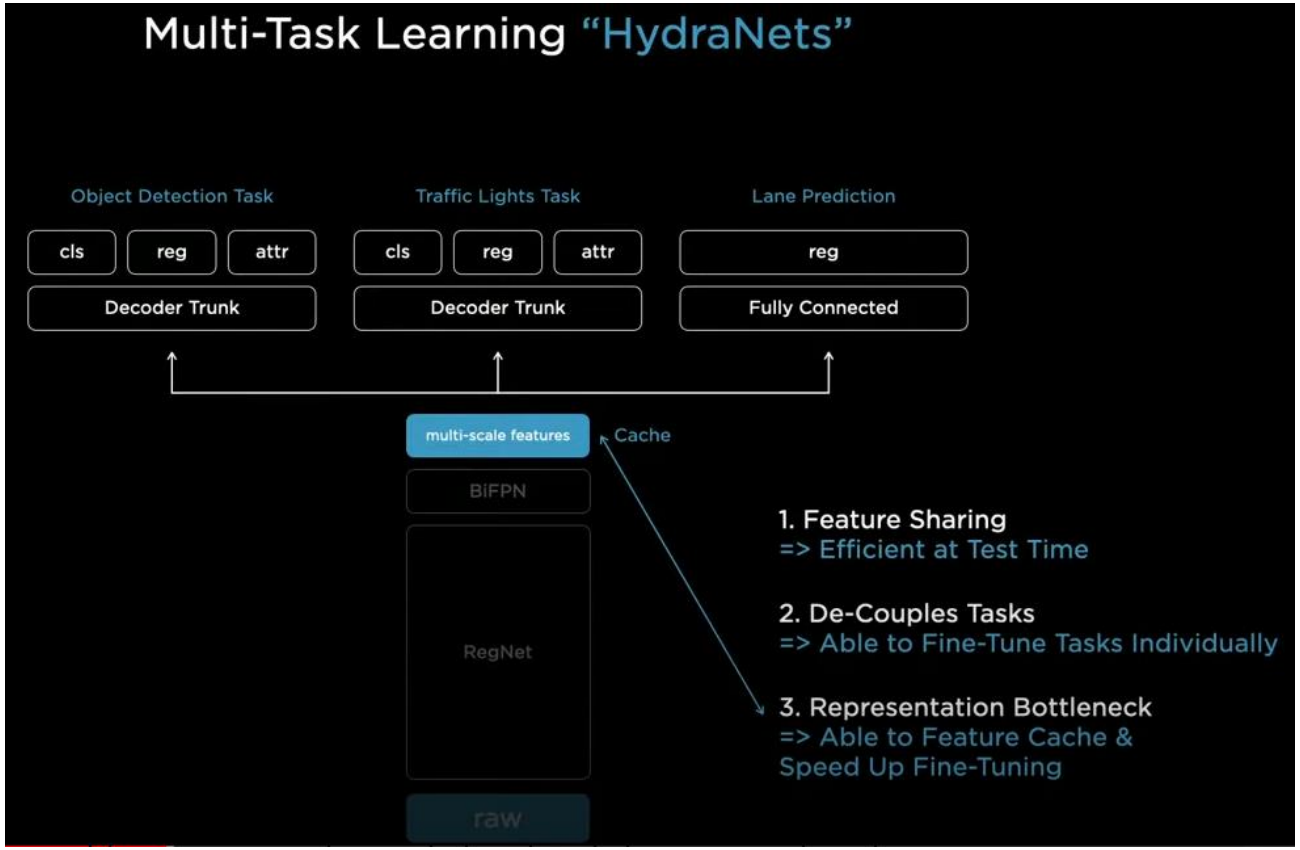
PERFORMING MULTIPLE TASKS

- Don’t want to just detect cars- want to do a large number of tasks.
- Traffic lights, lane prediction...
- Common shared backbone that branches to a number of ‘heads’



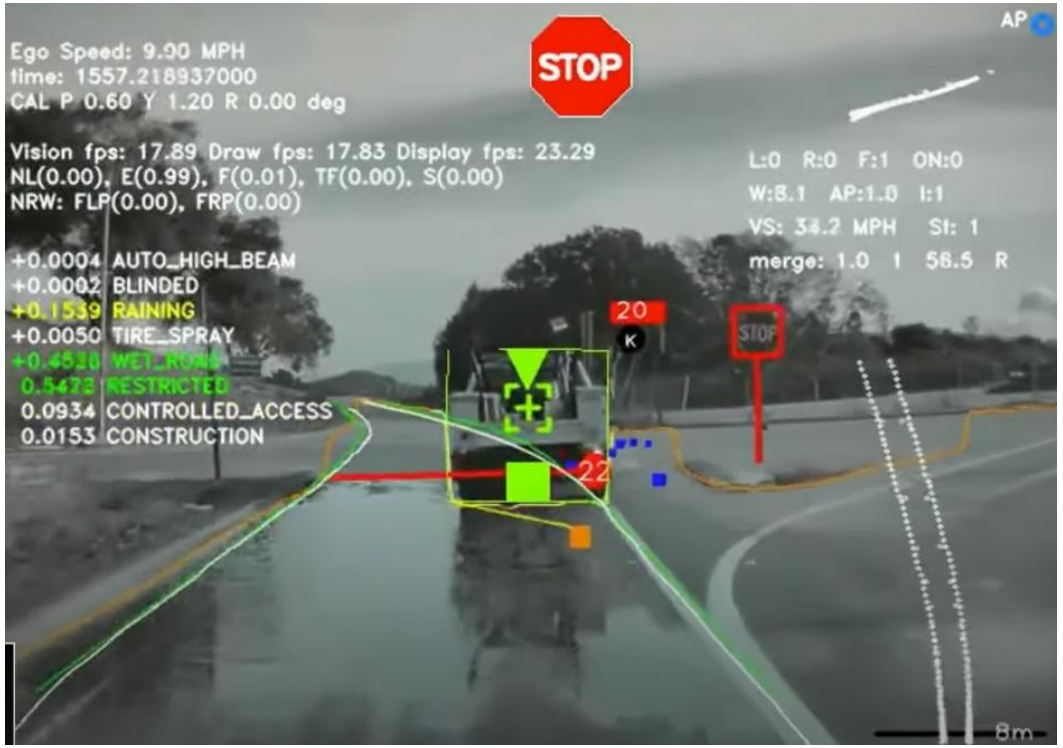
HYDRANETS

- Architecture advantages
- The main imaging trunk is only run once and shared among the ‘heads’
- The detector heads can be developed in isolation- don't have to invalidate or re-evaluate the other tasks.
- The bottleneck of features, is often cached to disk, they just refine from the cached features up to the heads.
- Once in a while they do an end-to-end training run.



EARLY PREDICTIONS

- This is some of the predictions they were getting several years ago, from one of the hydra heads.
- Processing individual images, finding: stop signs, stop lines, road boundaries, cars, traffic lights, curbs



SMART SUMMON

- But this was not enough- where it started to break when they introduced ‘smart summon’.
- These are the camera images and predictions for only the curb-detection task.
- Navigating the parking lot to find the person who summoned the car.

IMAGE SPACE

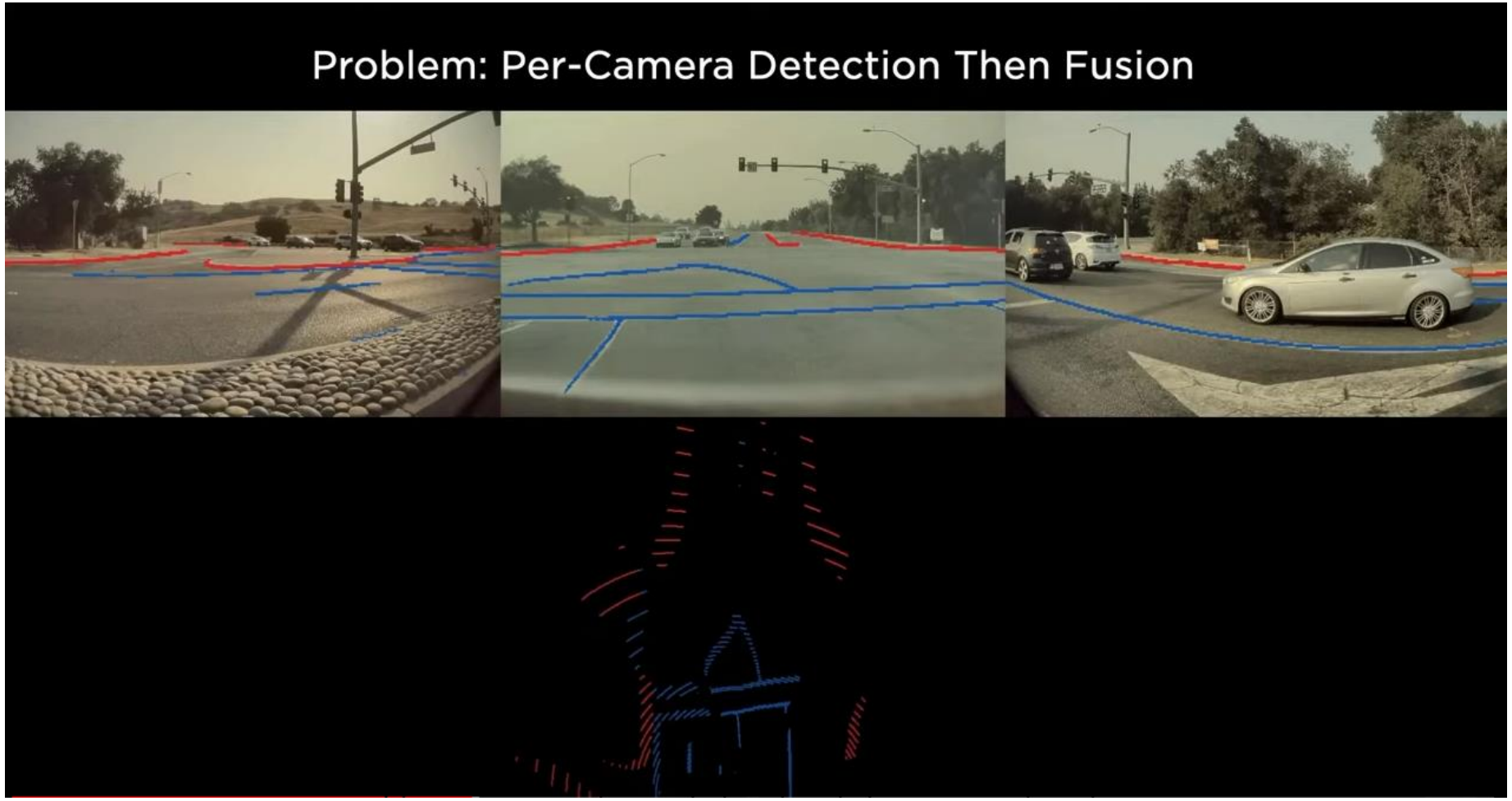
- You can't drive on image space predictions- you need to cast them into a vector space.

OCCUPANCY TRACKER

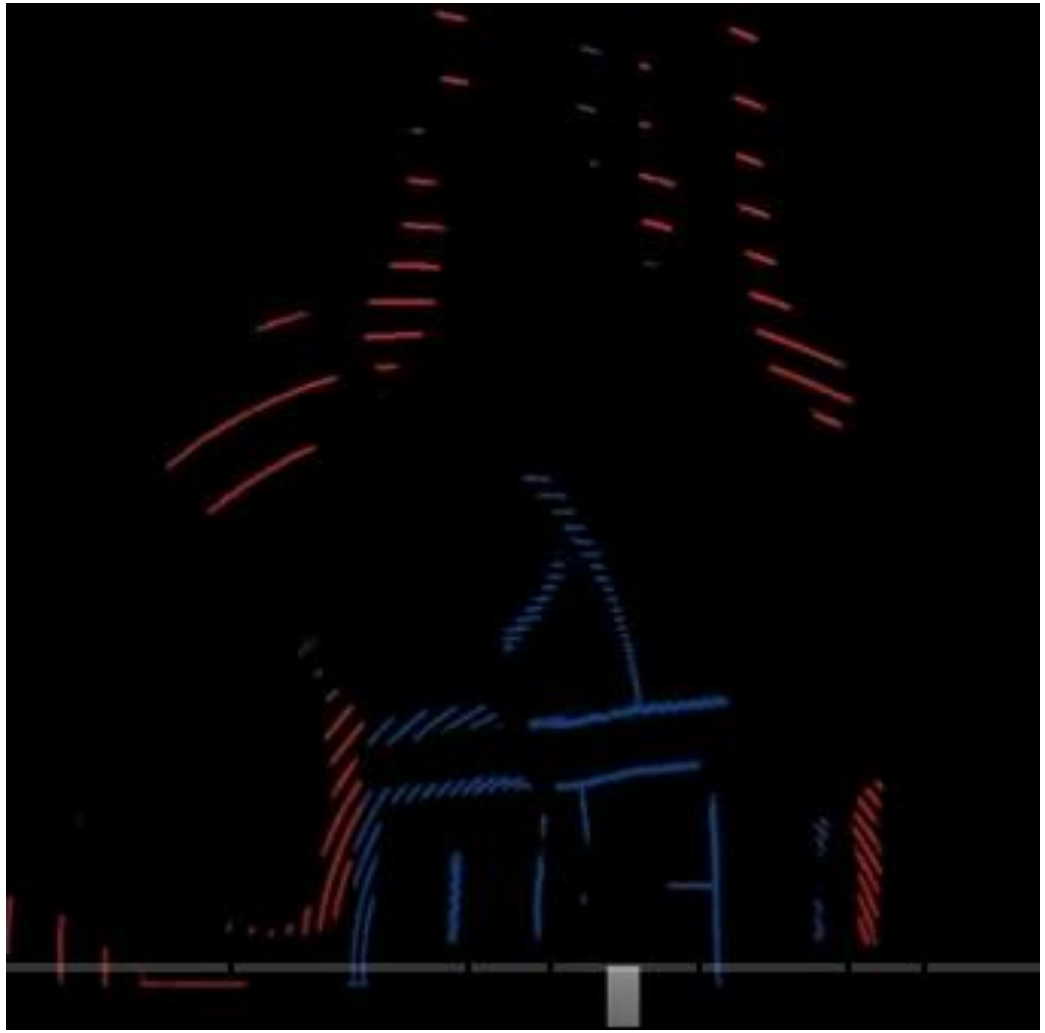
- Using C++ Tesla created the 'Occupancy Tracker' which attempted to create this vector space.
- The curb detection is being 'stitched up' across camera scenes.

OT SHORTCOMINGS

- There were too many parameters to tune in the C++ program.
- You don't want to make predictions in image space, you want to make predictions in vector space.



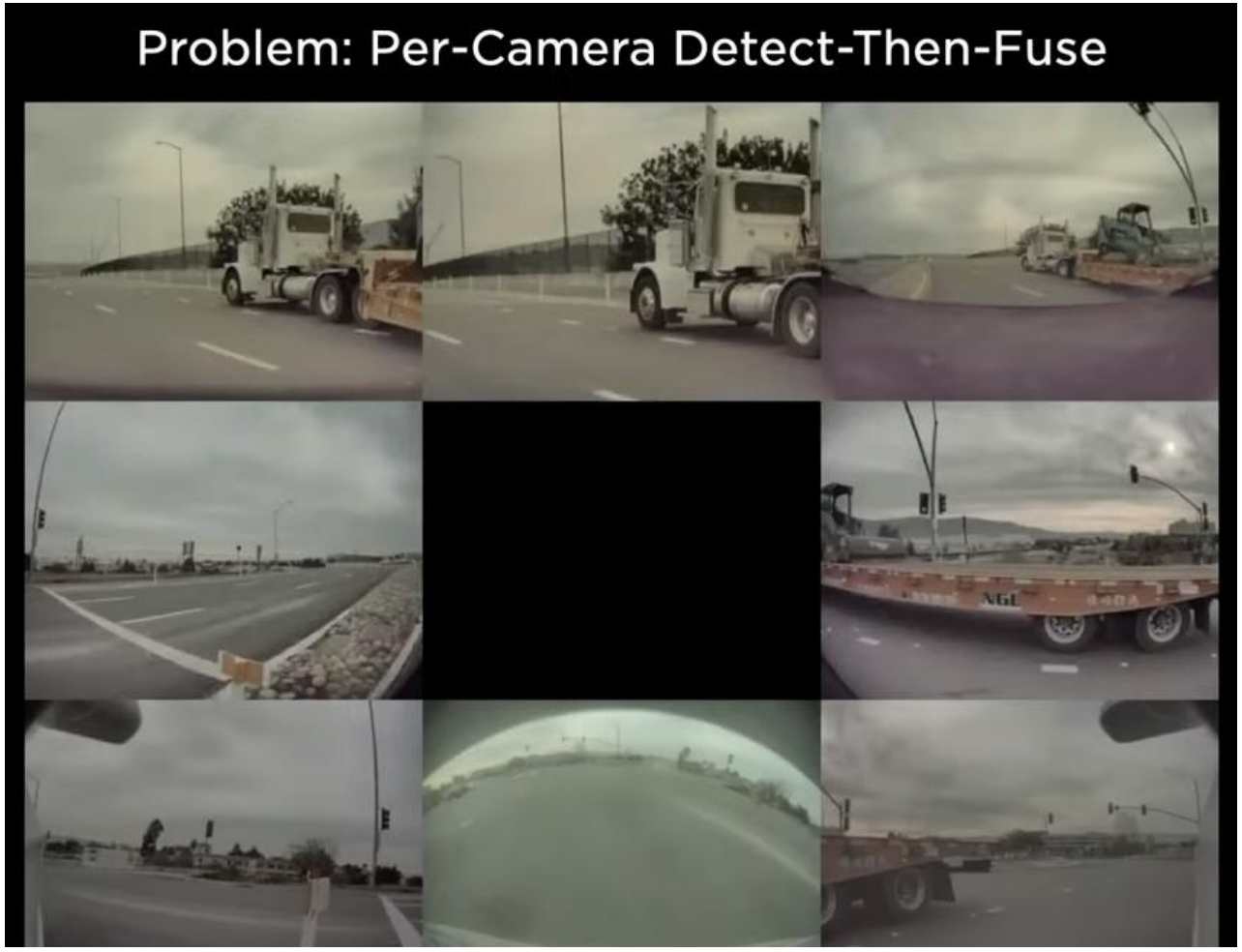
PROJECTING INTO VECTOR SPACE



- Curbs and lane markers look great in the image, but project poorly into vectorspace.
- You can't drive using this projection. The reason that this vector-space representation is so bad is because you need an extremely accurate depth per pixel. And also need to make predictions for any occluded area because its not part of the image.

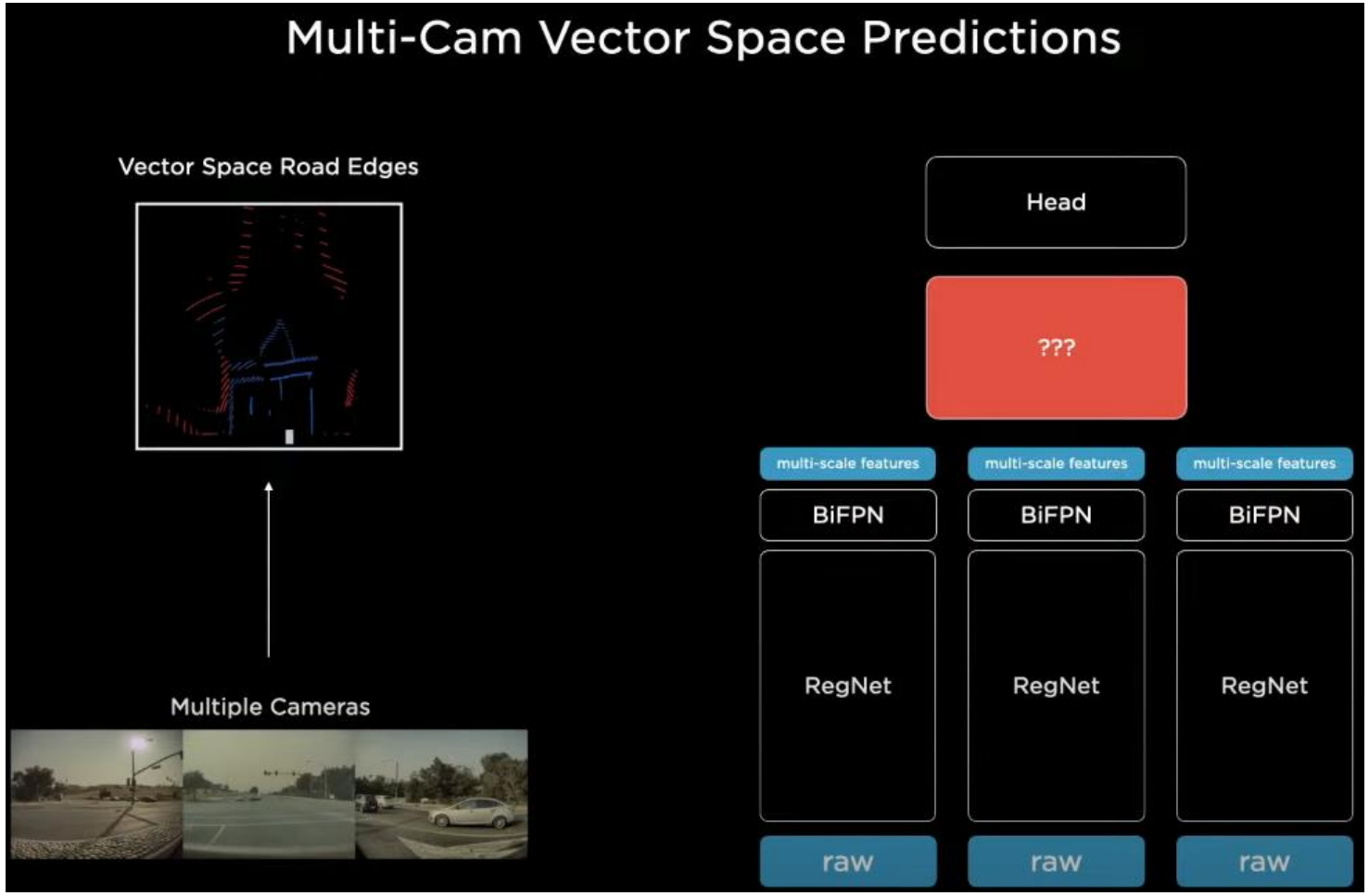
OT AND OBJECT DETECTION

- Object detection also is an issue with the occupancy tracker approach.
- If you are only making predictions per camera, then sometimes you will have a situation where a vehicle spans five of the eight cameras. No single camera sees all of the vehicle.



MULTI-CAM

- The next approach- was to take all the images and feed them into a single NN and output in vector space. Process every single image with a backbone and then fuse them and re-represent them into a vector space and then decode into the head.

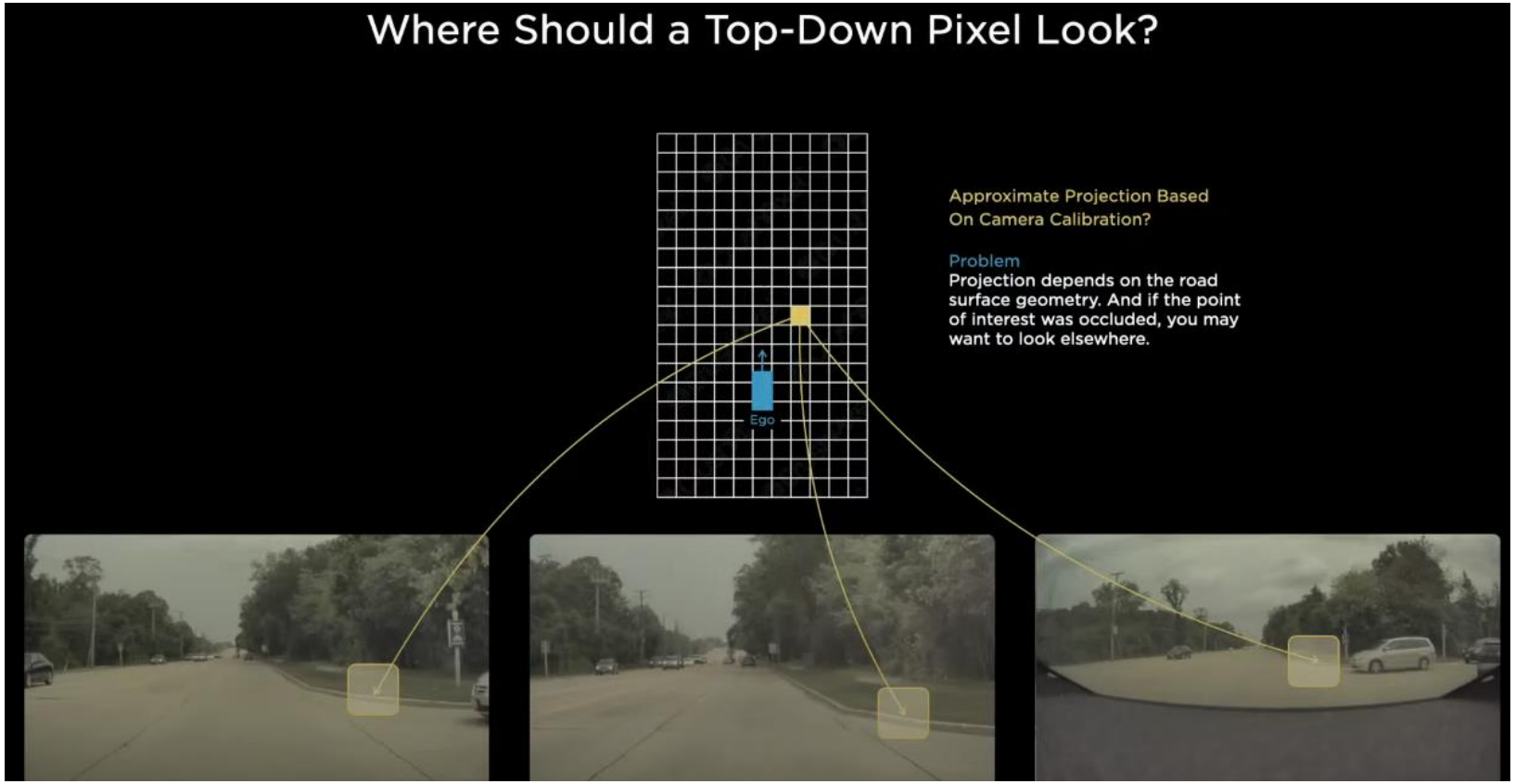


VECTOR SPACE DATASETS

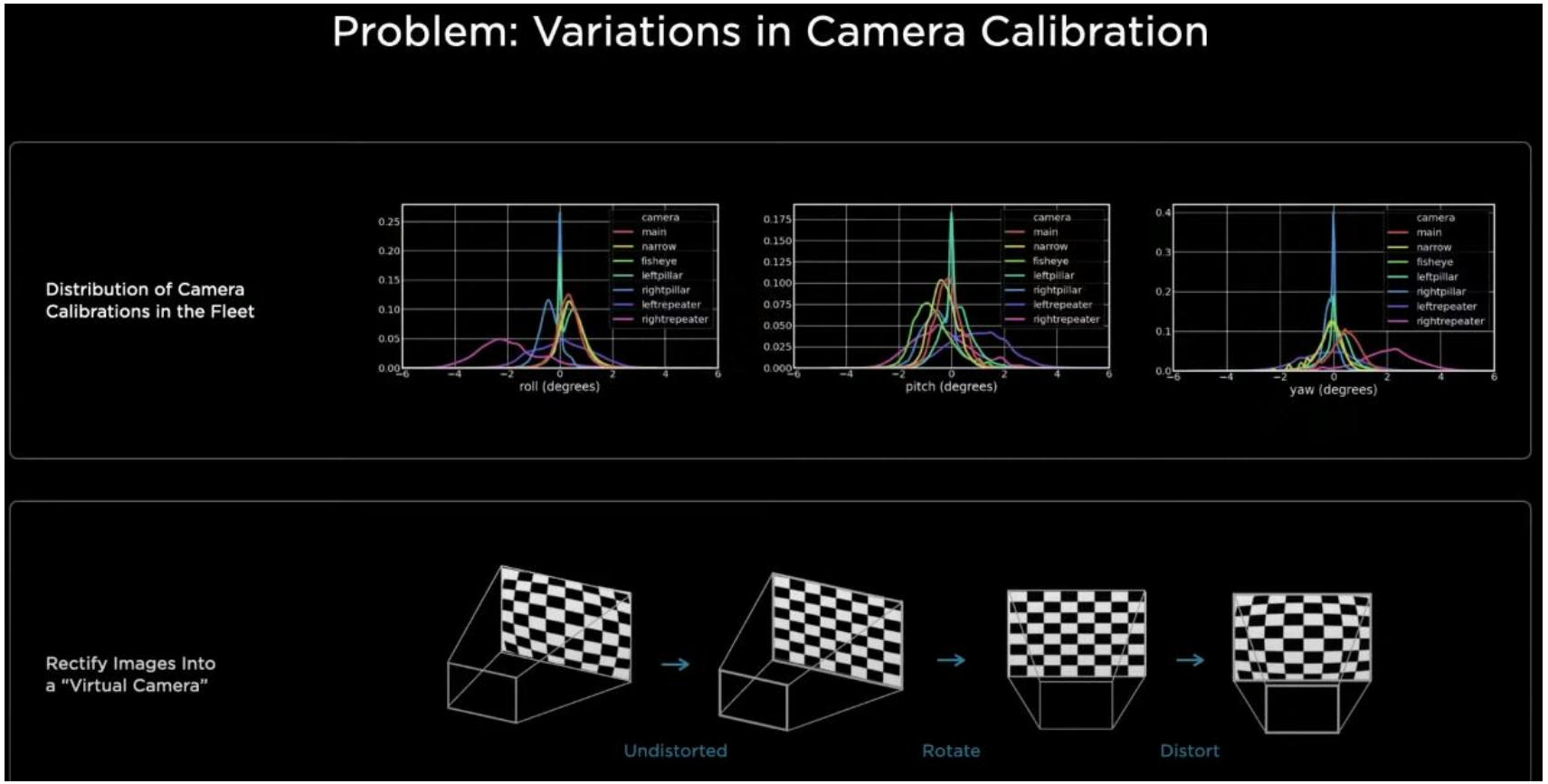
- How do you create the NN components that perform this transformation?
- If you want vector space predictions, you need vector space datasets. Just labeling images does not get you there - you need vector space labels.

CURB PIXEL?

- This single pixel is trying to determine if it is part of a curb or not.
- Where should the support for this prediction come from in the image space? We roughly know where the cameras are pointed and we can roughly project this point from the camera images.
- This projection is hard to get correct, it is a function of the road surface, which can be sloping up or down. It could be occluded by a car. Because it is data driven it is hard to have a fixed transformation component.



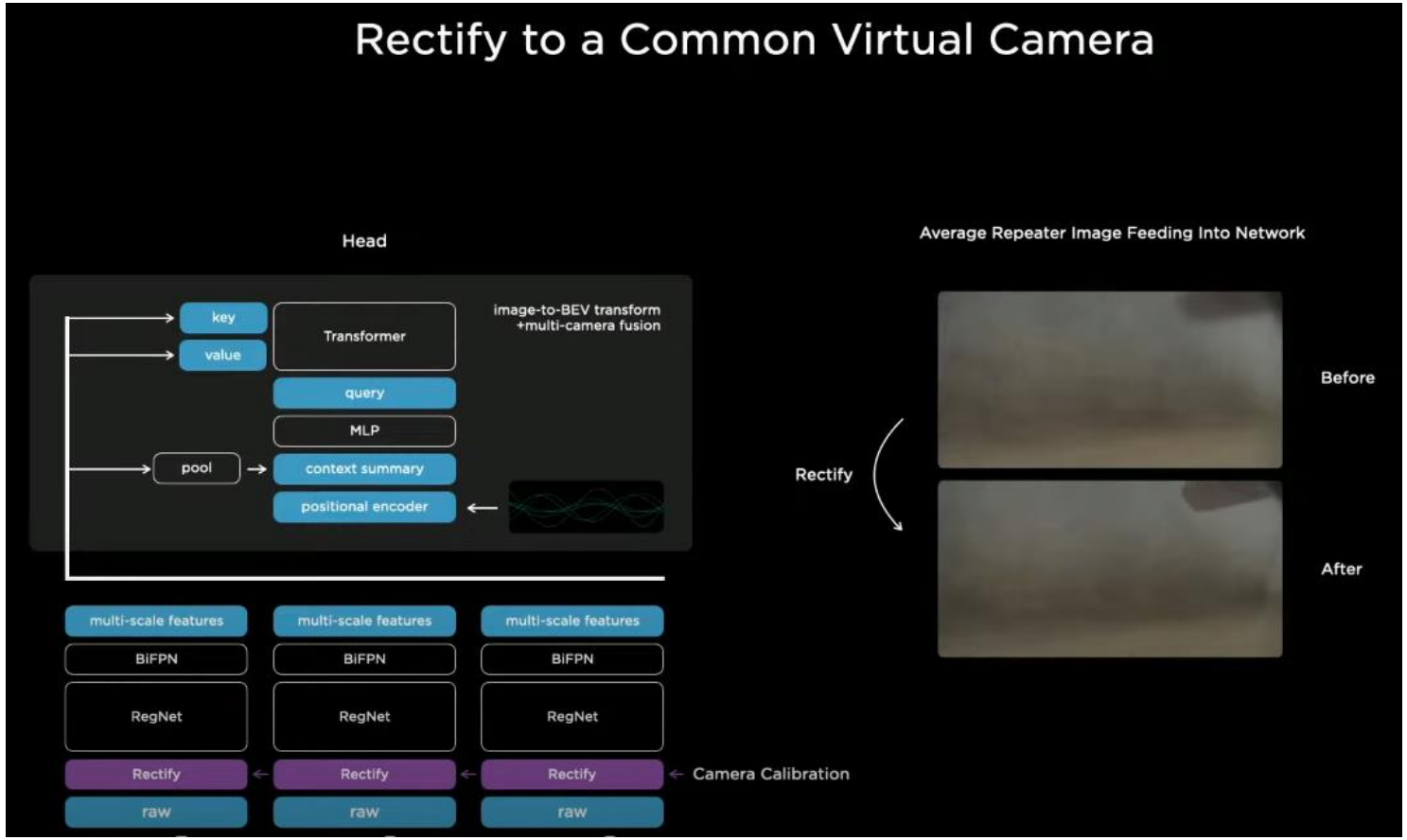
CALIBRATION



- If you are doing a transformation from the image to the output space you need to know your camera calibration.
- Each car's cameras are position slightly differently.

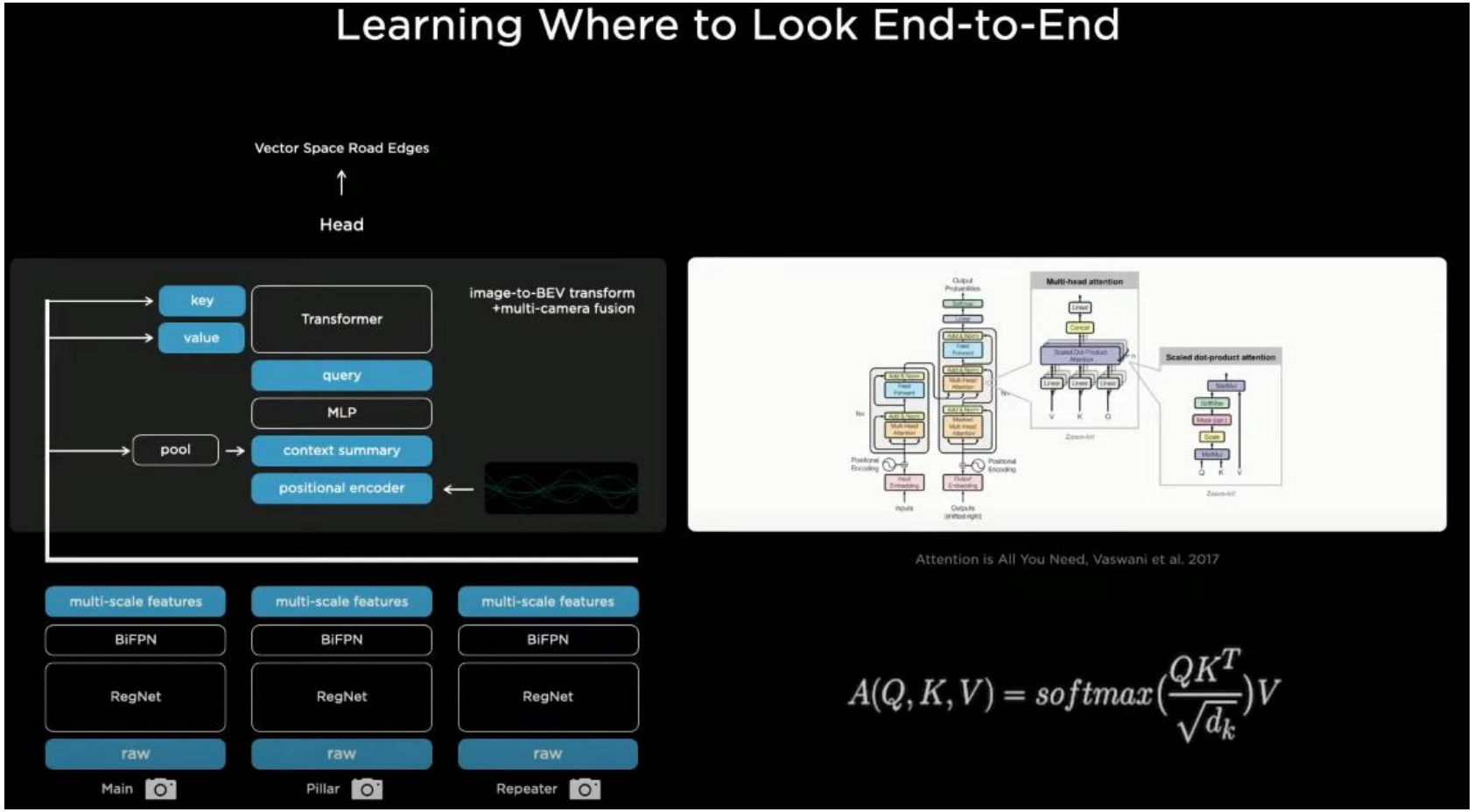
COMMON CAMERA

- Tesla transforms all of the images into a single synthetic virtual camera. They insert a new layer "rectify" just above the camera.
- Translates all of the images into a virtual common camera.



TRANSFORMER

- Tesla uses a transformer to represent this space using multi-headed self attention.

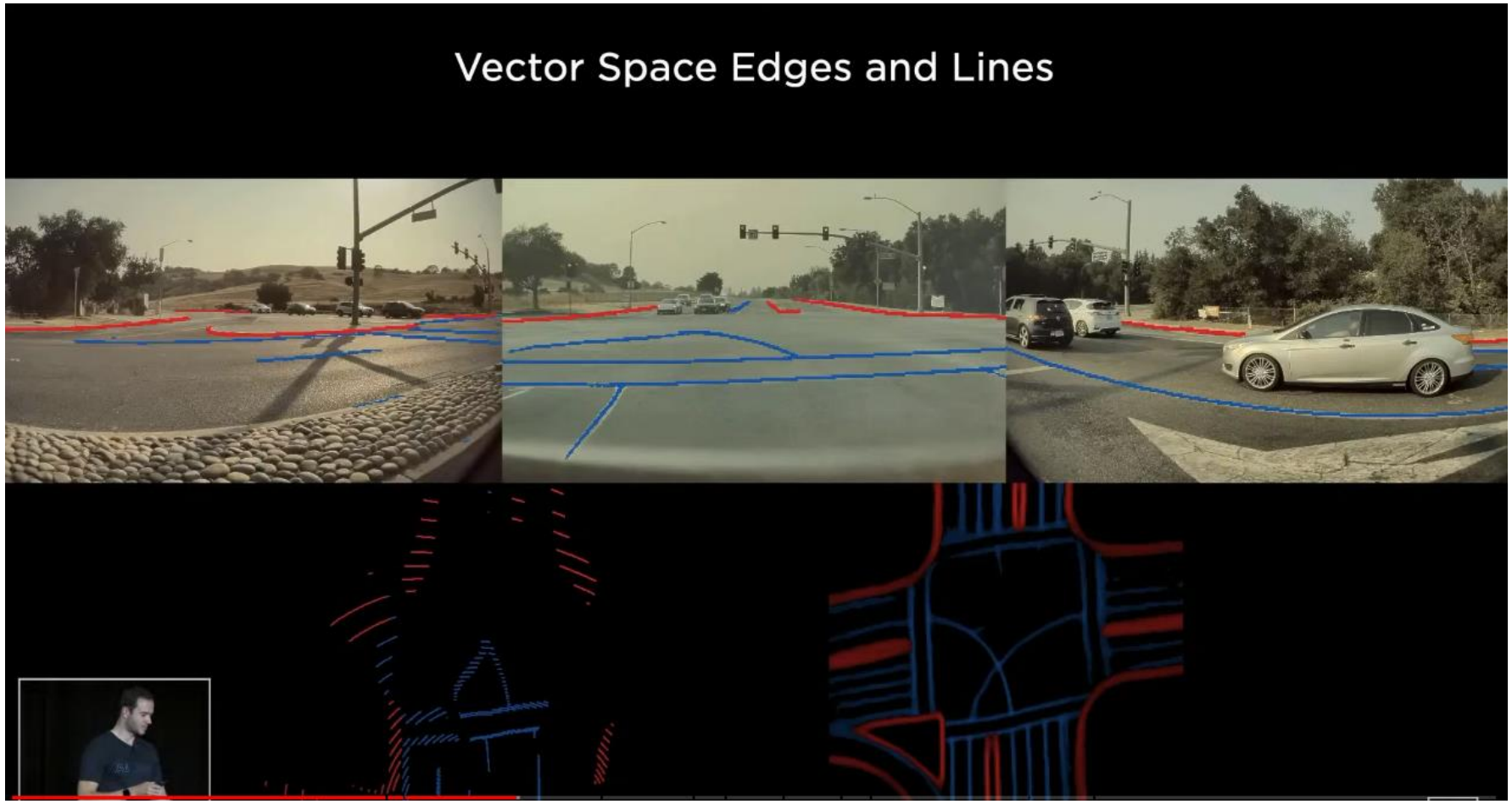


TRANSFORMER DETAILS (OVERVIEW)

- You initialize a raster the size of the output space that you want and tile it with positional encodings.
- Each of the images emit their own keys and values which feed into the multi-head self-attention.
- Essentially every single image piece is broadcasting in its key what it is a part of (e.g. I'm a pillar and I'm roughly at this location).
- Every query is similar to "i'm a pixel in the output space at this position and I'm looking for features of this type".
- The keys and values interact multiplicatively and the values get pooled accordingly.
- *(all of this is covered in detail in a couple slides)*

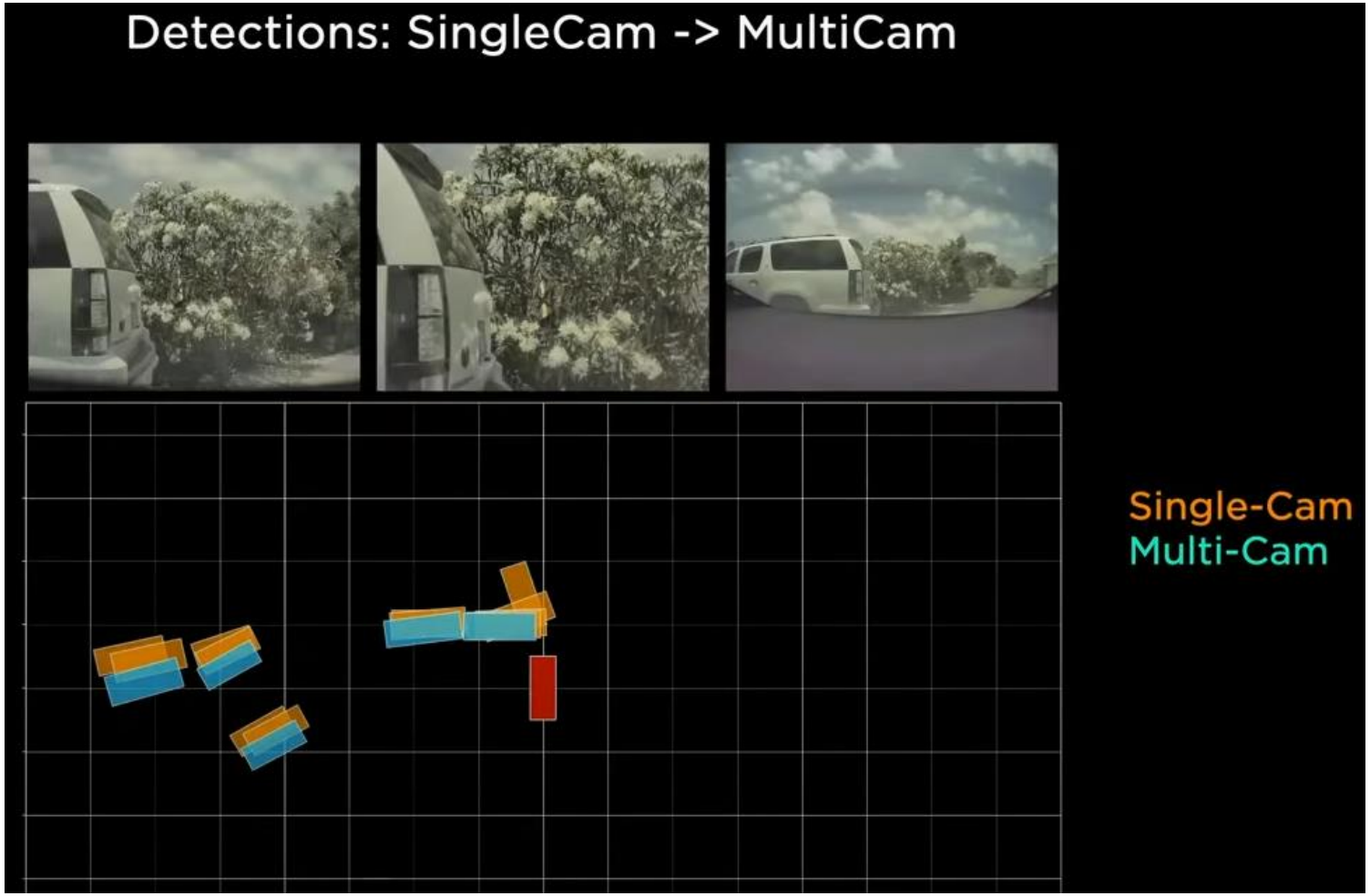
VECTOR SPACE DONE RIGHT

- Left shows the occupancy tracker code and the right is the vector space which is ‘something you can drive on’. The processing is deployed and efficient in the car.
- This also improved object detection.



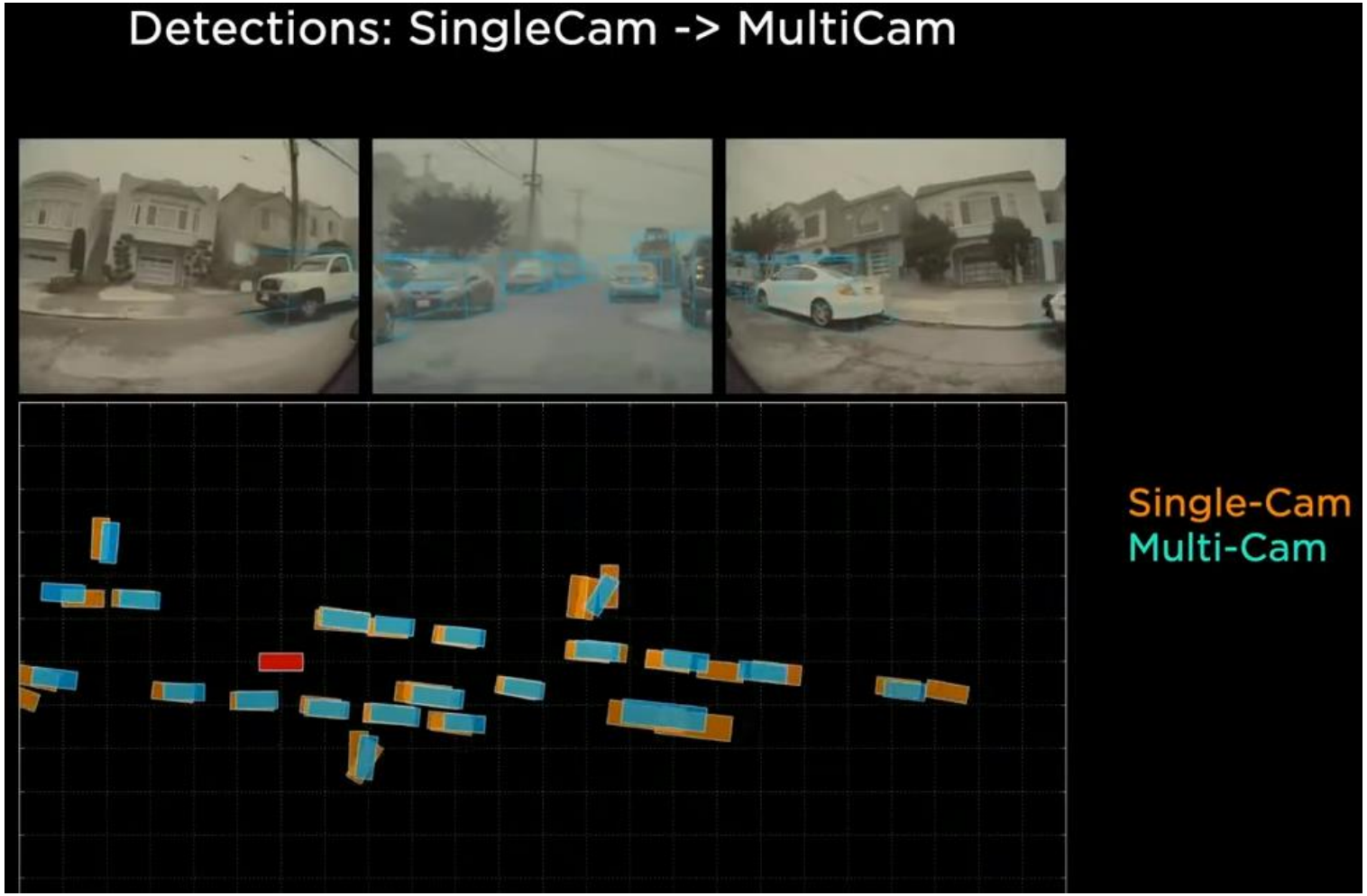
CROSS-CAMERA BOUNDARIES

- As the cars in this tight space, cross camera boundaries, there is a lot of error that enters the predictions.



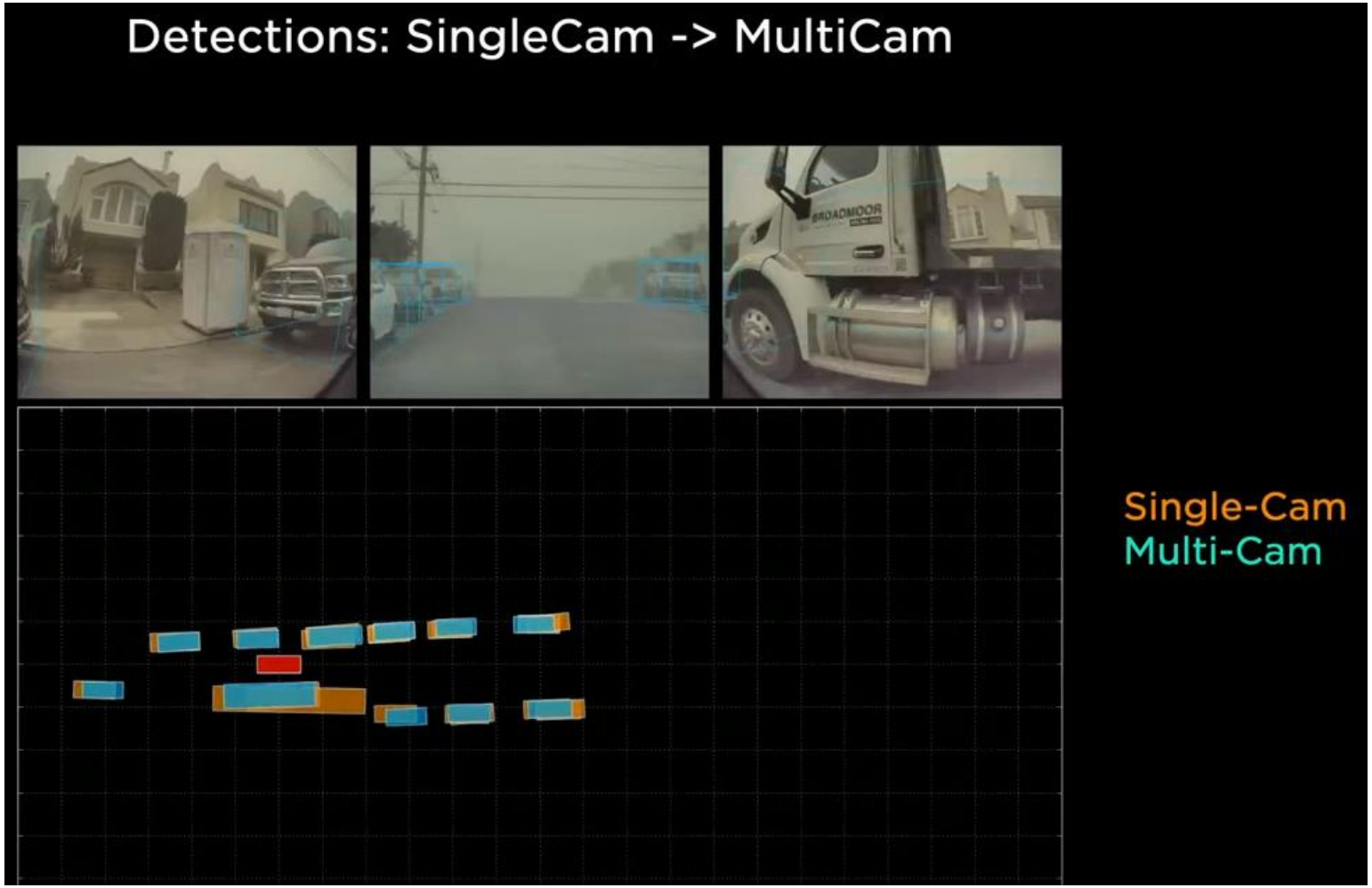
TIGHT SPACES AND CAMERA BOUNDARIES

- As the cars in this tight space, cross camera boundaries, there is a lot of error that enters the predictions.



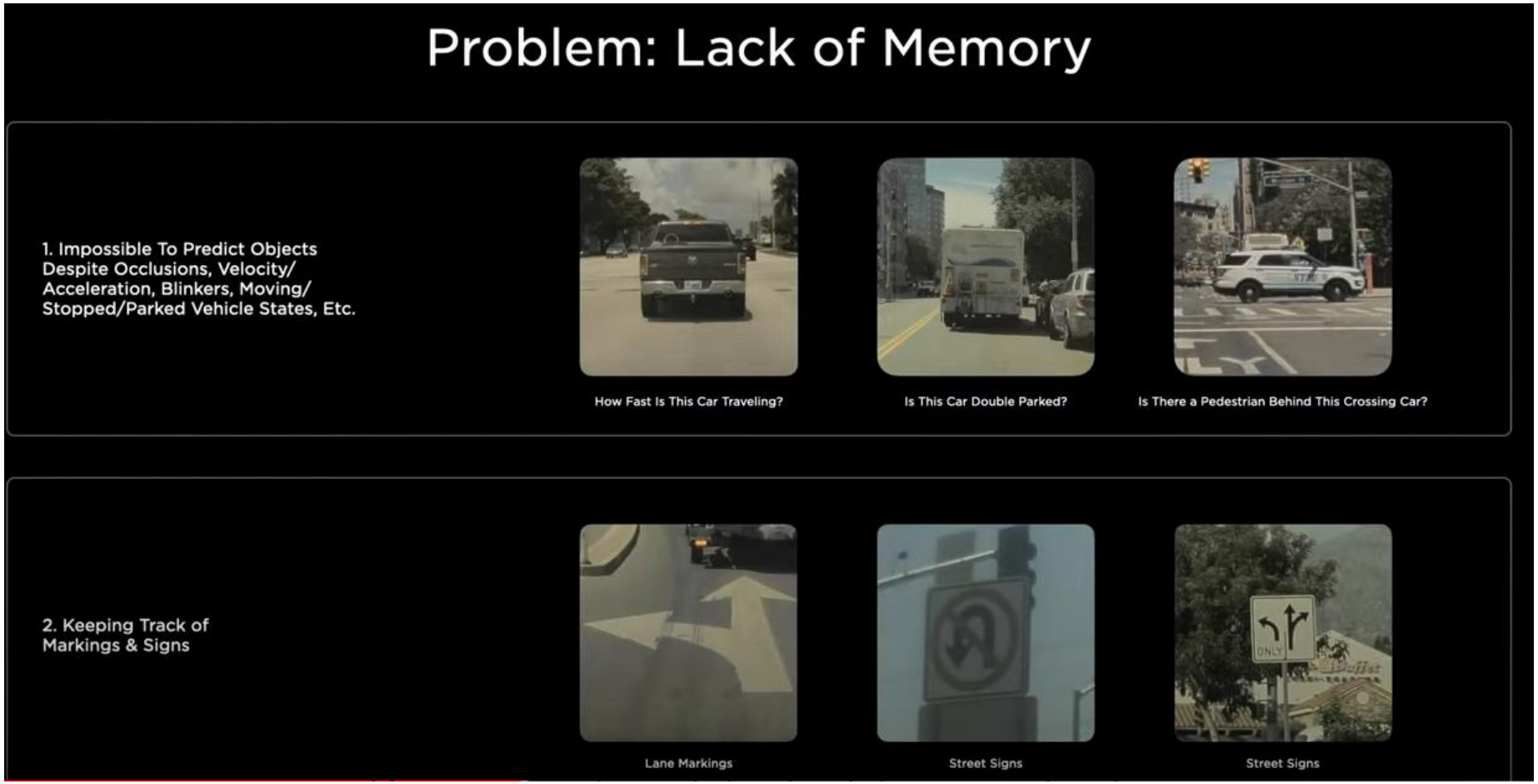
TIGHT SPACES AND CAMERA BOUNDARIES (CONT.)

- Especially for large vehicles.

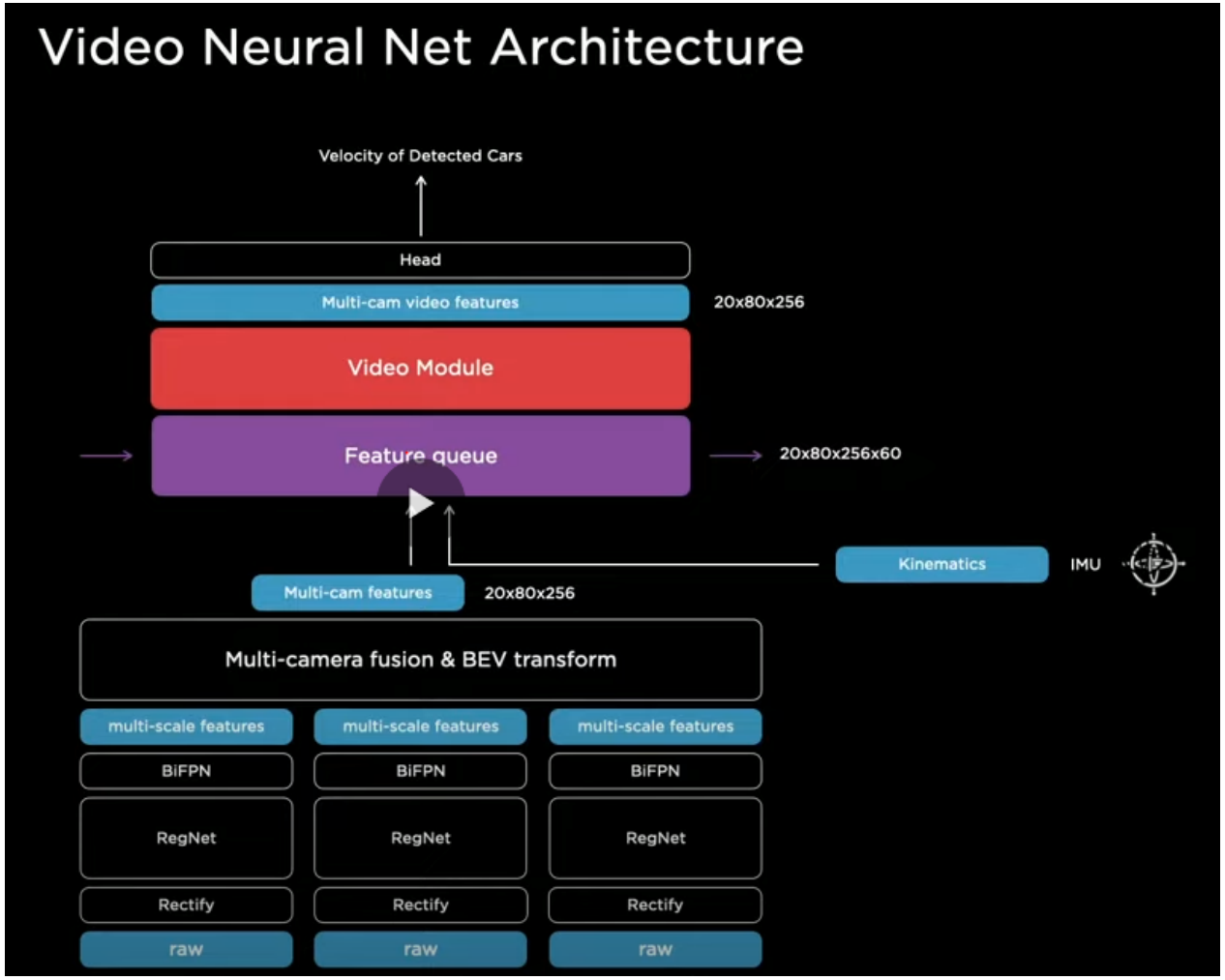


MEMORY AND CONTEXT

- The system is still operating at a every instance of time completely independently.
- A large number of predictions require the video context.



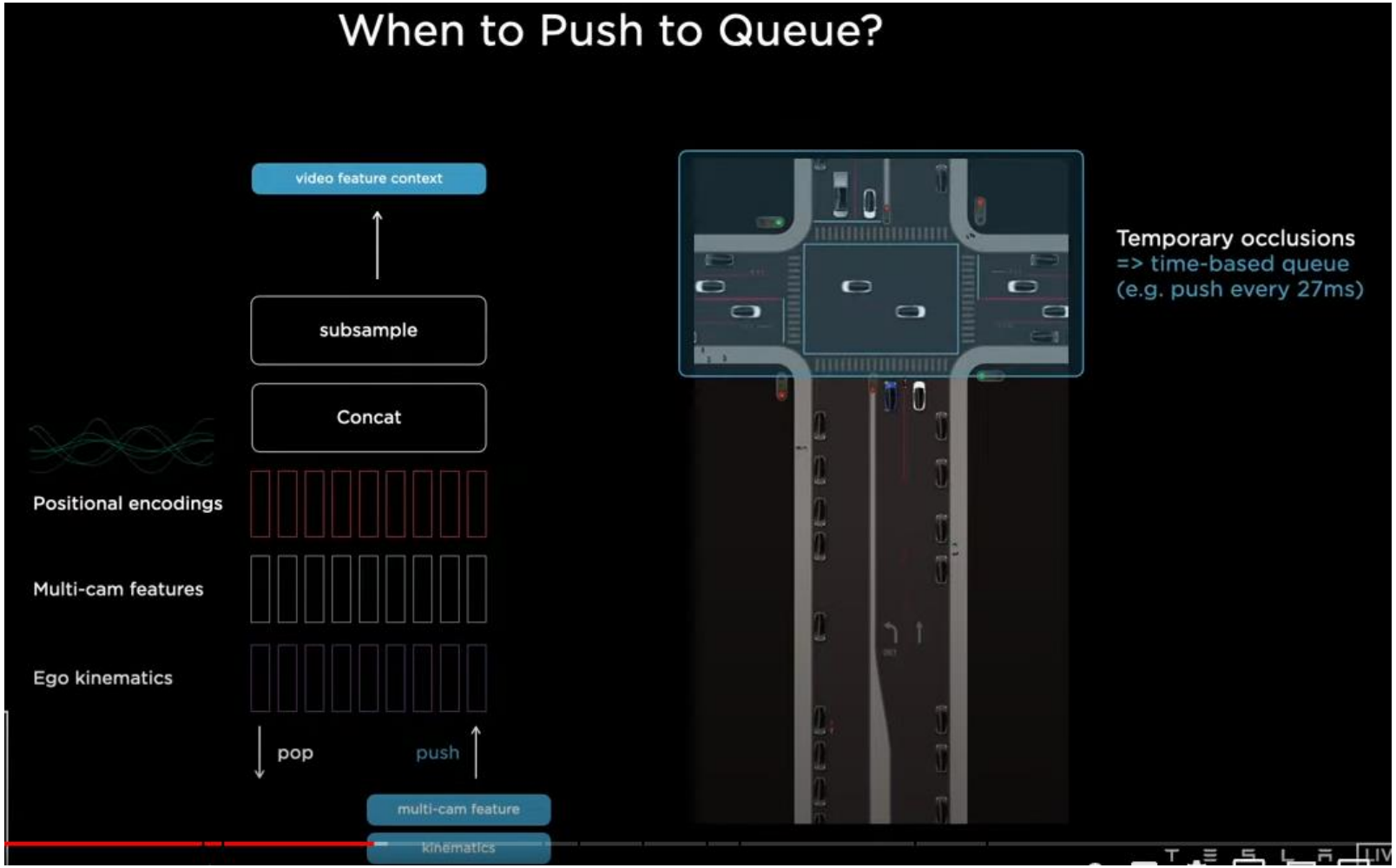
TIME AND KINEMATICS



- The feature queue module will cache some features over time. Insert a video module that will fuse the cache information temporally.
  - Kinematics are also fed in- which tell how the car is moving.
- Feature queue is concatenating features over time, how the car has moved, positional encodings and camera features.

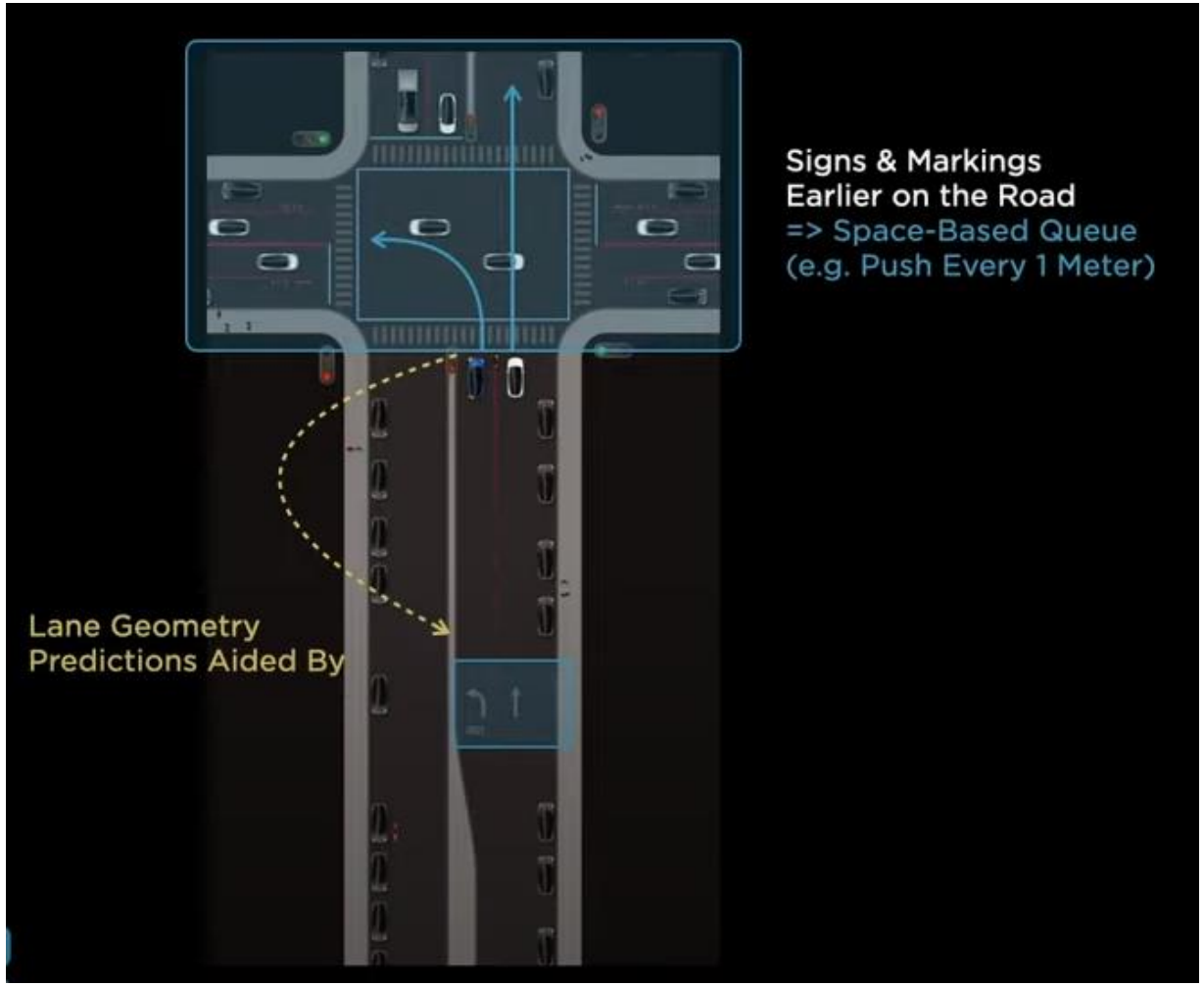
QUEUE

- Push and pop mechanism- when do you push?
- The car has travelled up from the bottom and is sitting in the left turn lane waiting at a light.

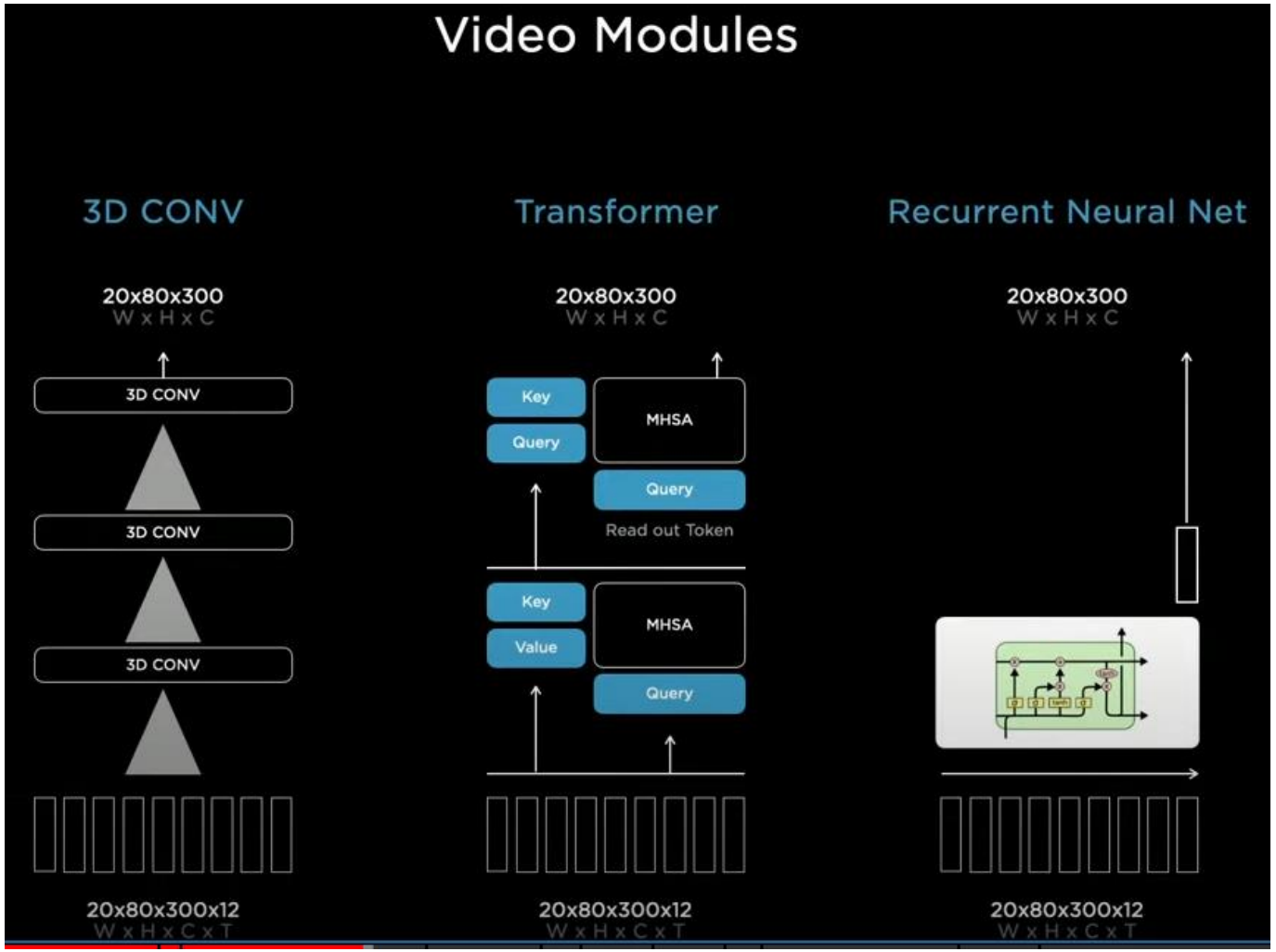


PAST MEMORIES

- Features are entered into the queue every 27ms. Even if a car is occluded, the network has a memory that a vehicle was present.
- A space-based queue feeds information as the vehicle moves. In this case, the network has a memory that it is in a turn lane, and there is a lane to the right that will continue going straight.



VIDEO NEURAL NETWORK ARCHITECTURE

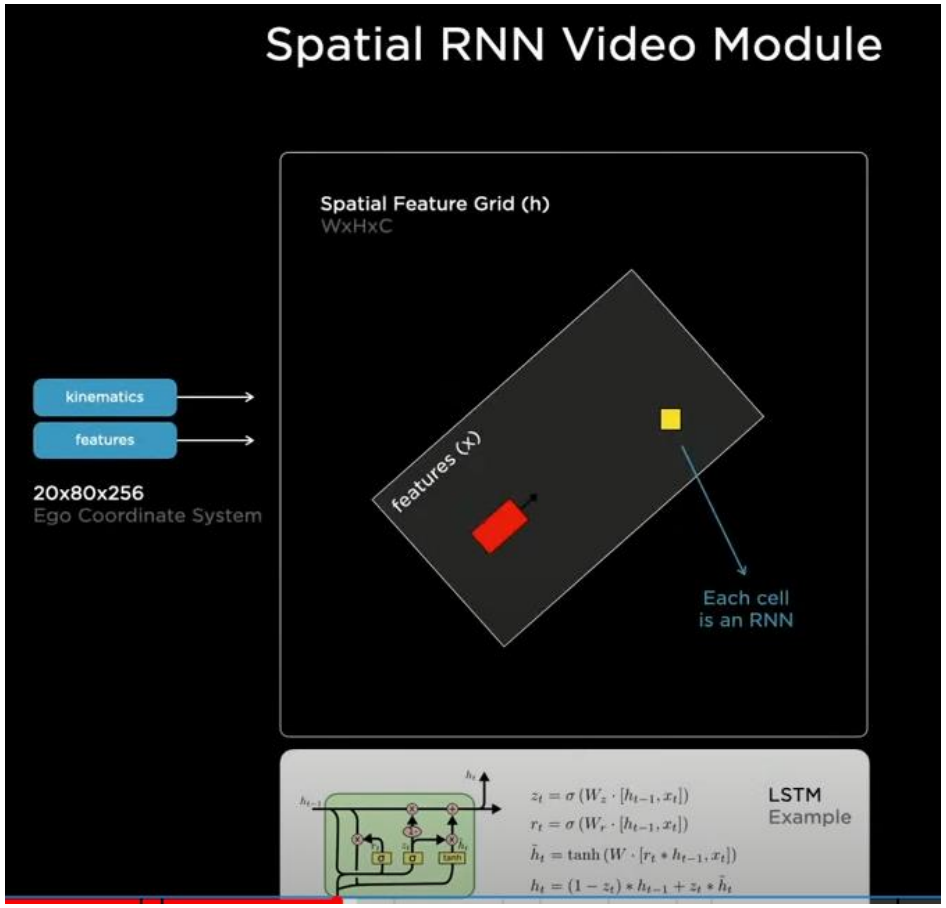


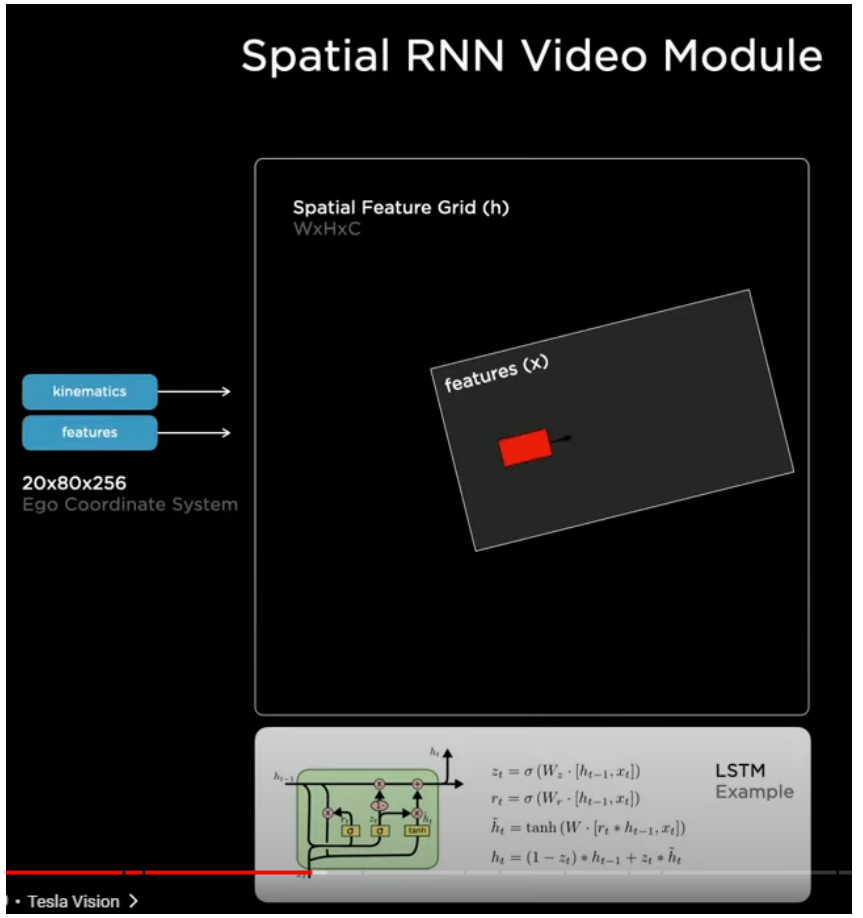
- 3D Convolutions, Transformers (Axial Transformers) and Recurrent Neural Networks.

SPACIAL RNN

- Tesla has settled on a Spacial RNN.
- Since the car is driving on a two-dimensional surface, we can organize the hidden state into a two-dimensional lattice.
- This allows us to only update the parts that are near the car, or where the car has visibility.
- The kinematics to update the position of the car in the hidden features grid.

VEHICLE MOVEMENT AND LATICE UPDATES

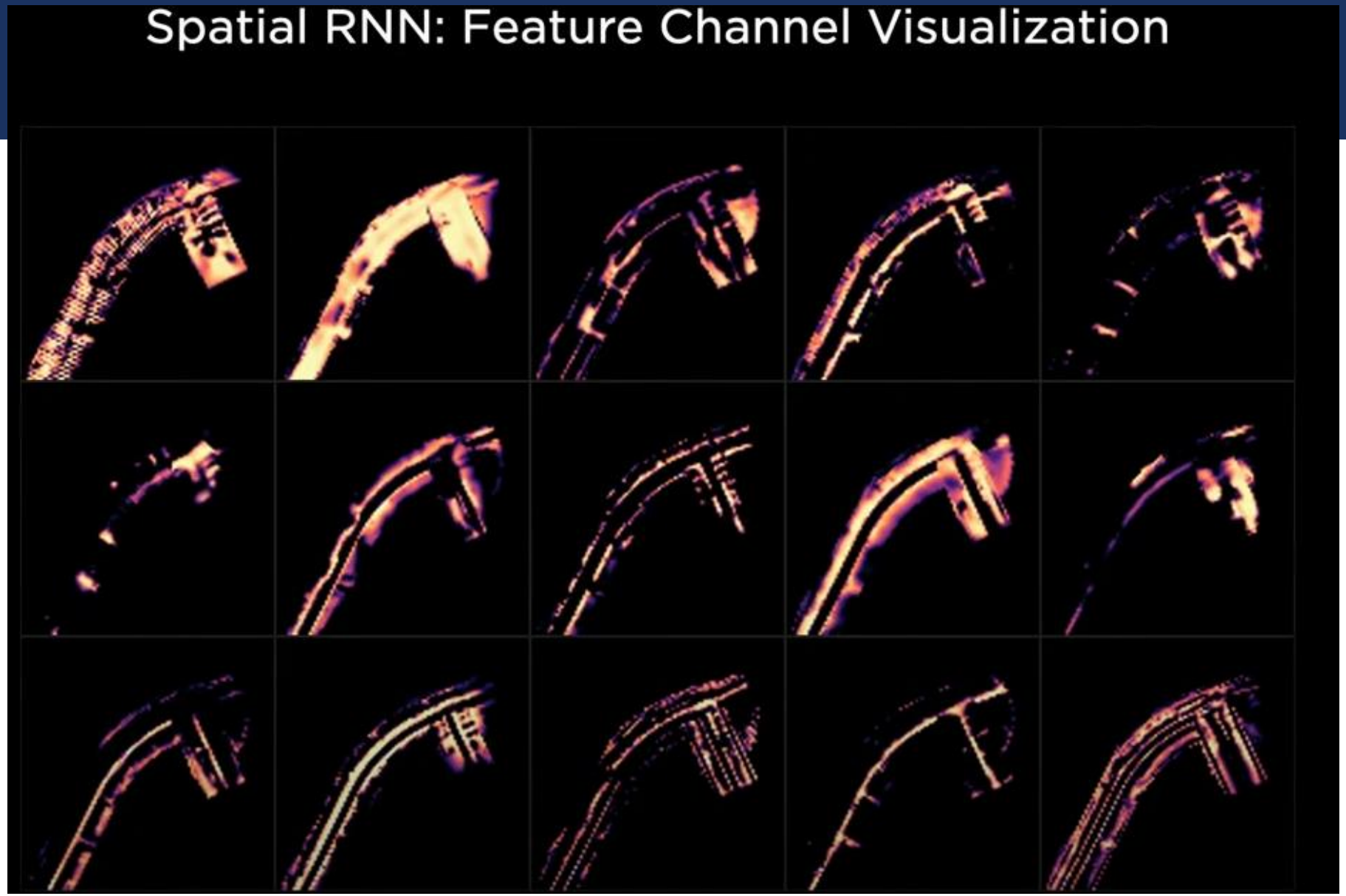




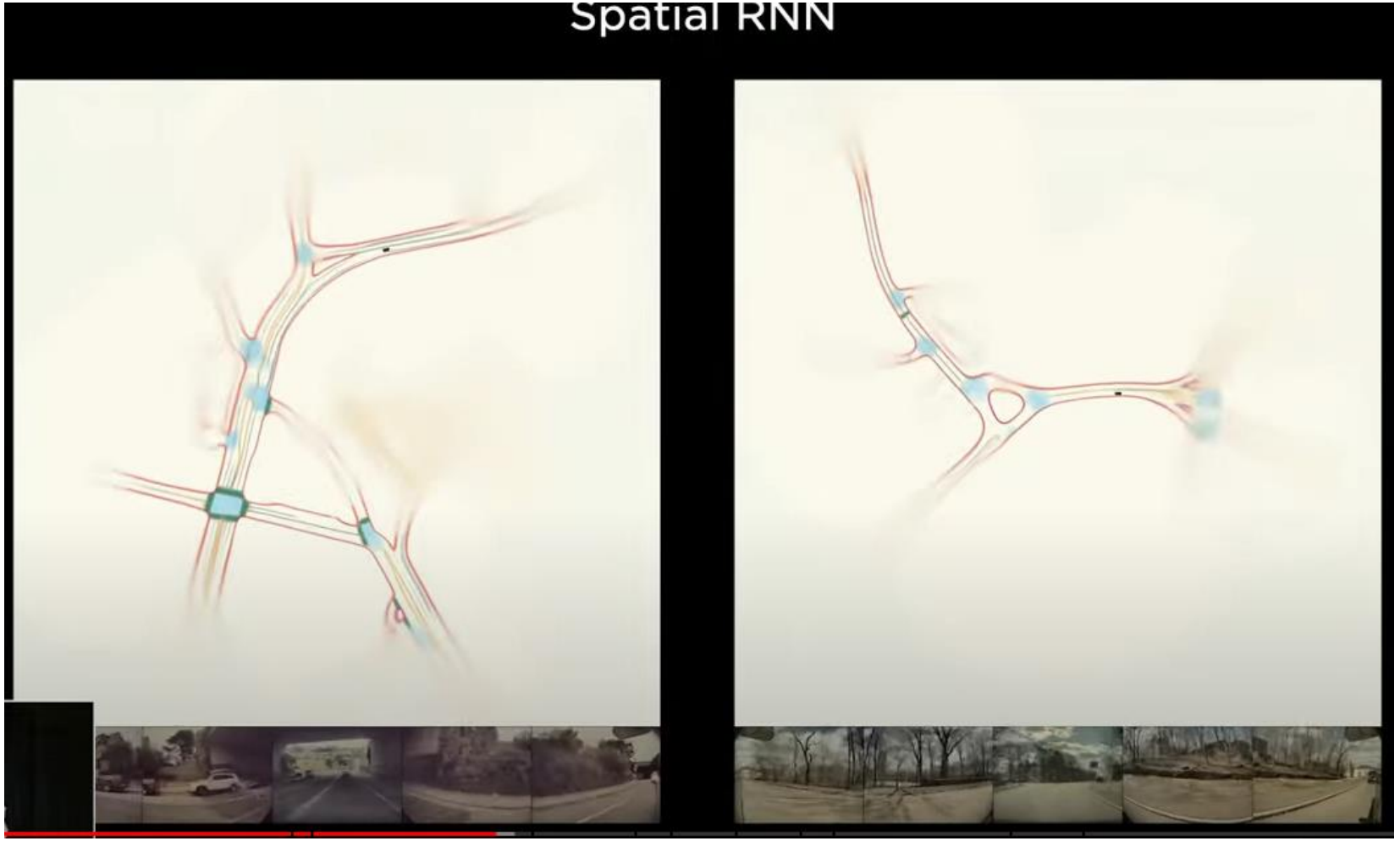
LATTICE UPDATES

- Only updating those parts of the RNN that are located 'close' to the car.
- The next slide shows a visualization of channels of the hidden state of this RNN.
- Some channels are tracking certain aspects of the road- center, edges, lines. The RNN is keeping track over time.
- The NN has the power to selectively read and write to this memory. So if there is a car occluding the view, the system can wait until there is a clear view, to update that part of memory.

Spatial RNN: Feature Channel Visualization



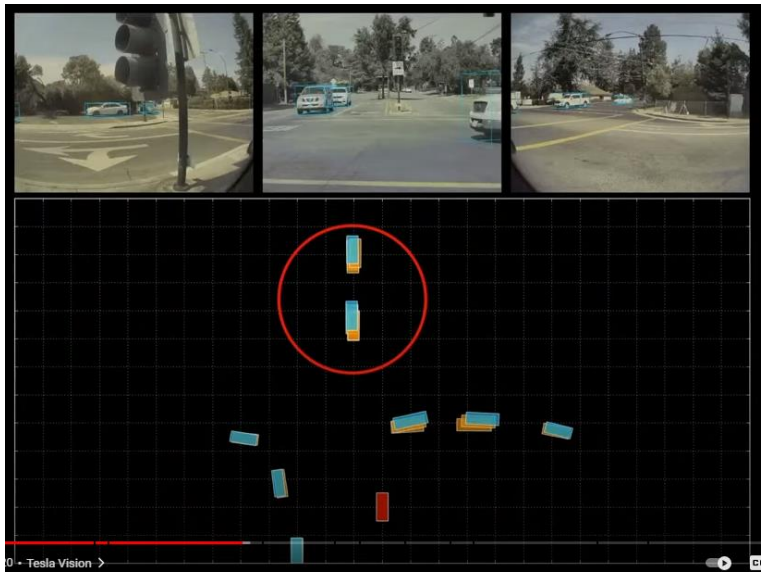
VIDEO AND RNN



- This is in the space of features of a RNN.
- It is possible to make out details of road boundaries and lanes.

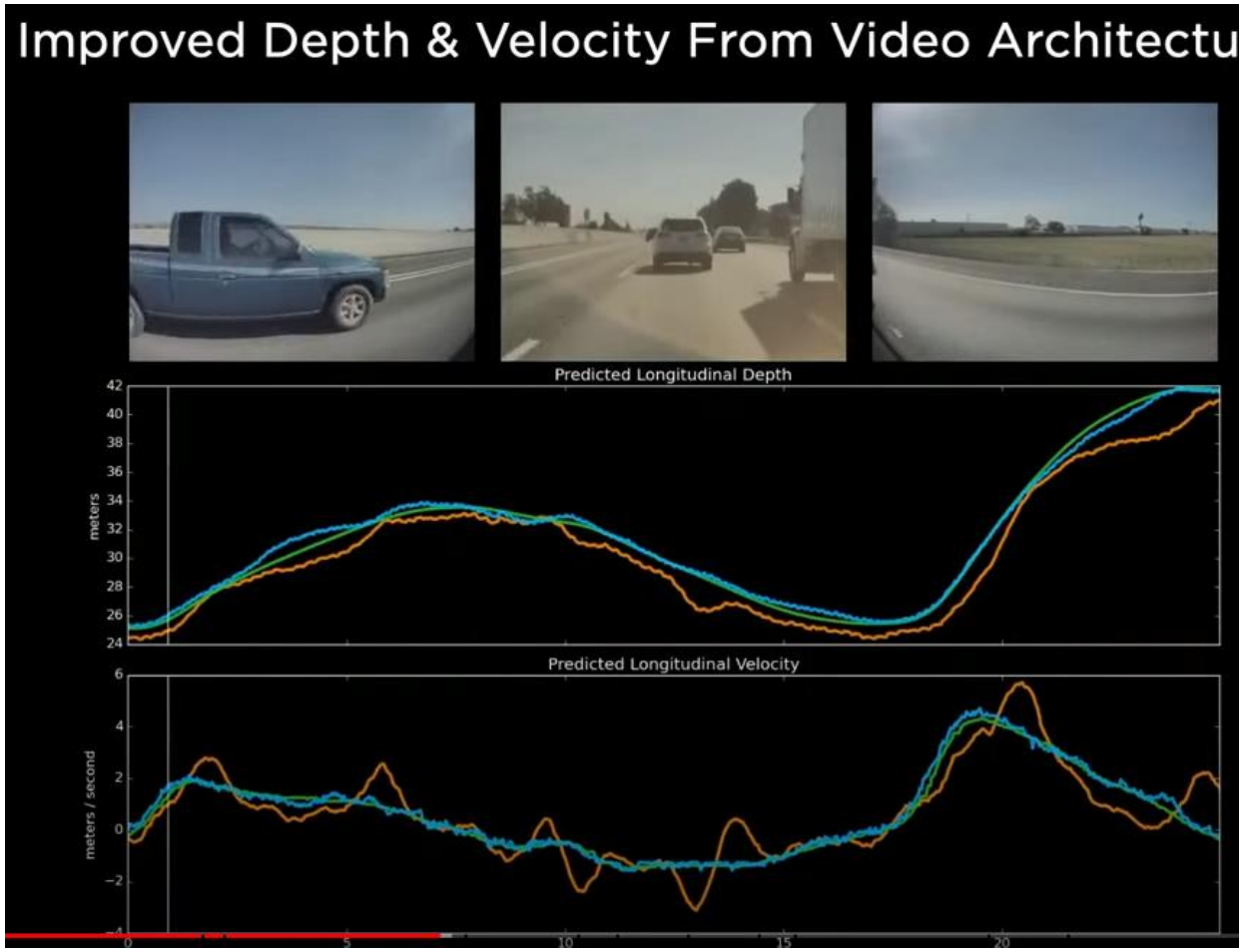
TEMPORARY OCCULSION

- Improved robustness to temporary occlusion
- The single frame (orange) system drops the cars ahead at the light when a truck passes in front of the vehicle and then struggles to re-identify and position the cars as the truck pulls around.



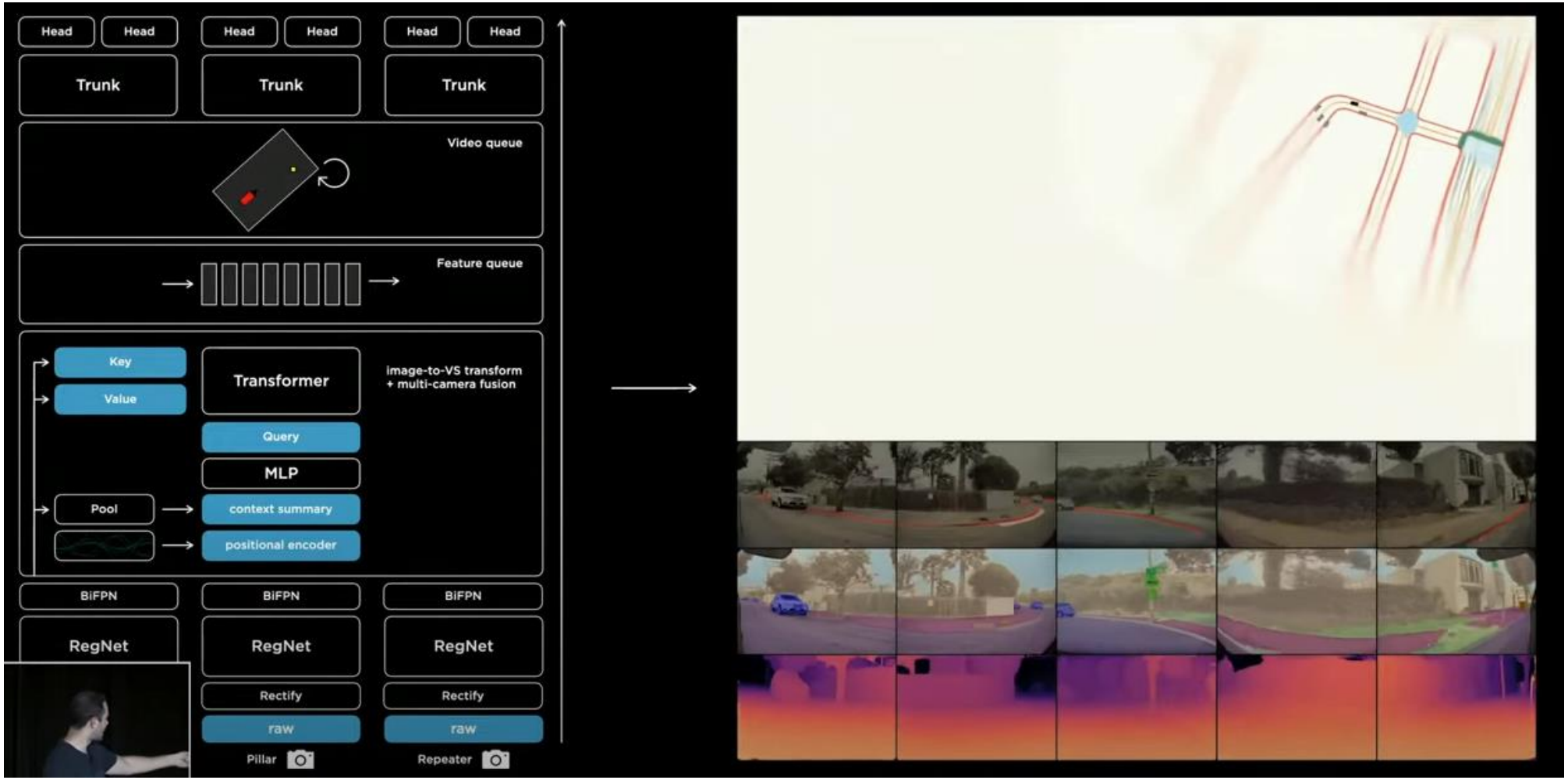
PREDICTING DEPTH AND VELOCITY

- Green indicates radar distance and velocity, orange is single frame prediction and the video module in blue.

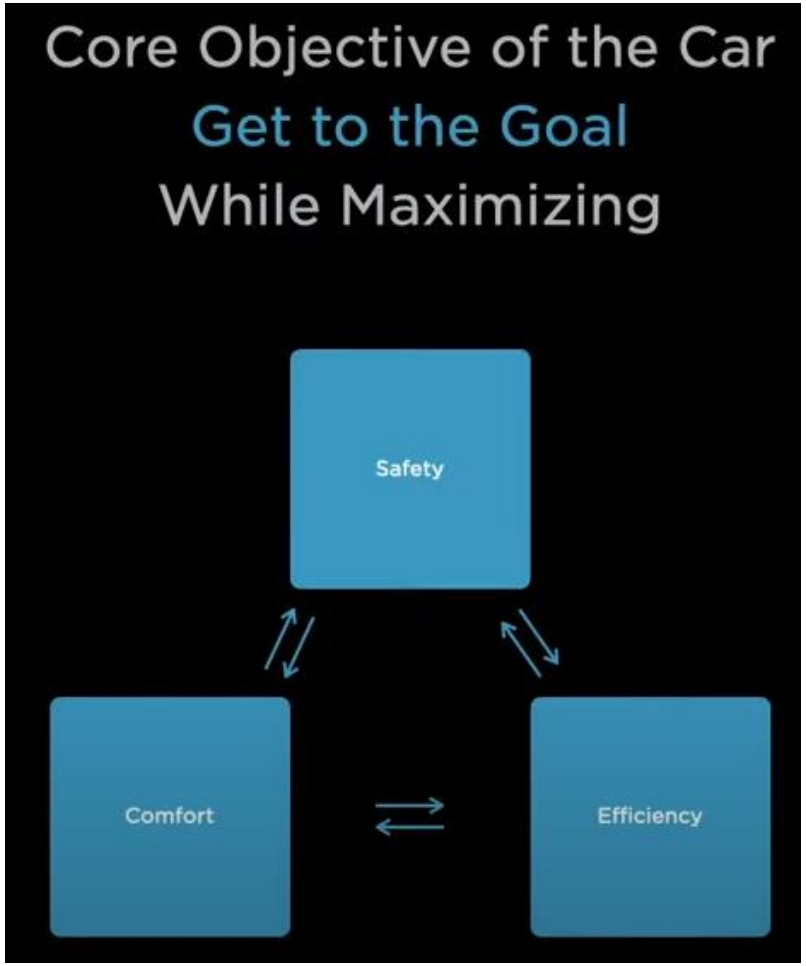


- Raw video images, rectification layer calibrates and common virtual camera, RegNets process features at different scales, fuse the multi-scale information with BiFPN, Transformer module to re-represent in a vector space, Feature Queue introduces time or space, Video queue maps into a two-dimensional plane, and then continues into the branching structure of the hydra head. Trunks and heads for the different tasks. Enhancements include fusing space or time earlier in the pipeline. Outputs are dense rasters, which are expensive to post-process in the car, not ideal given the latency requirements.

PUTTING IT ALL TOGETHER



PLANNING AND CONTROL

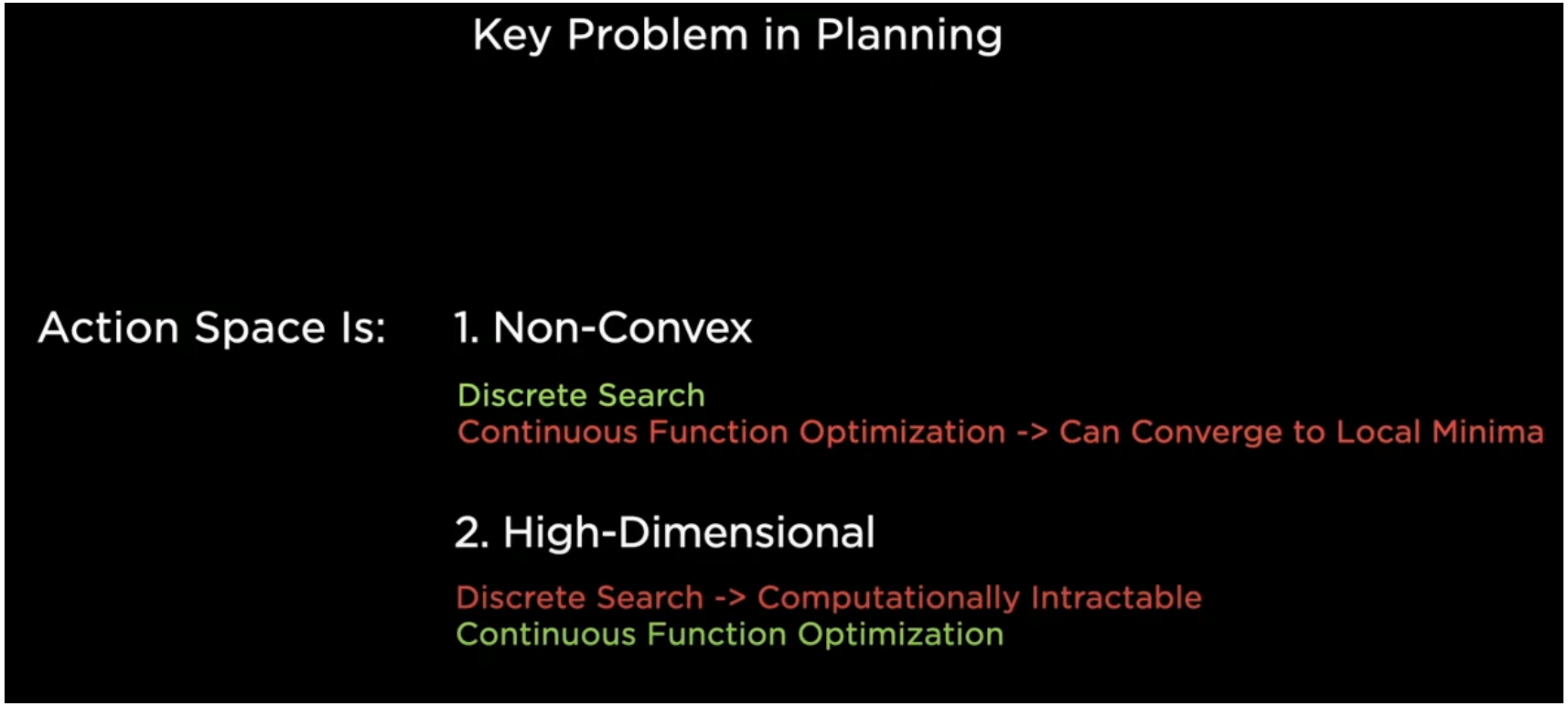


PLANNER

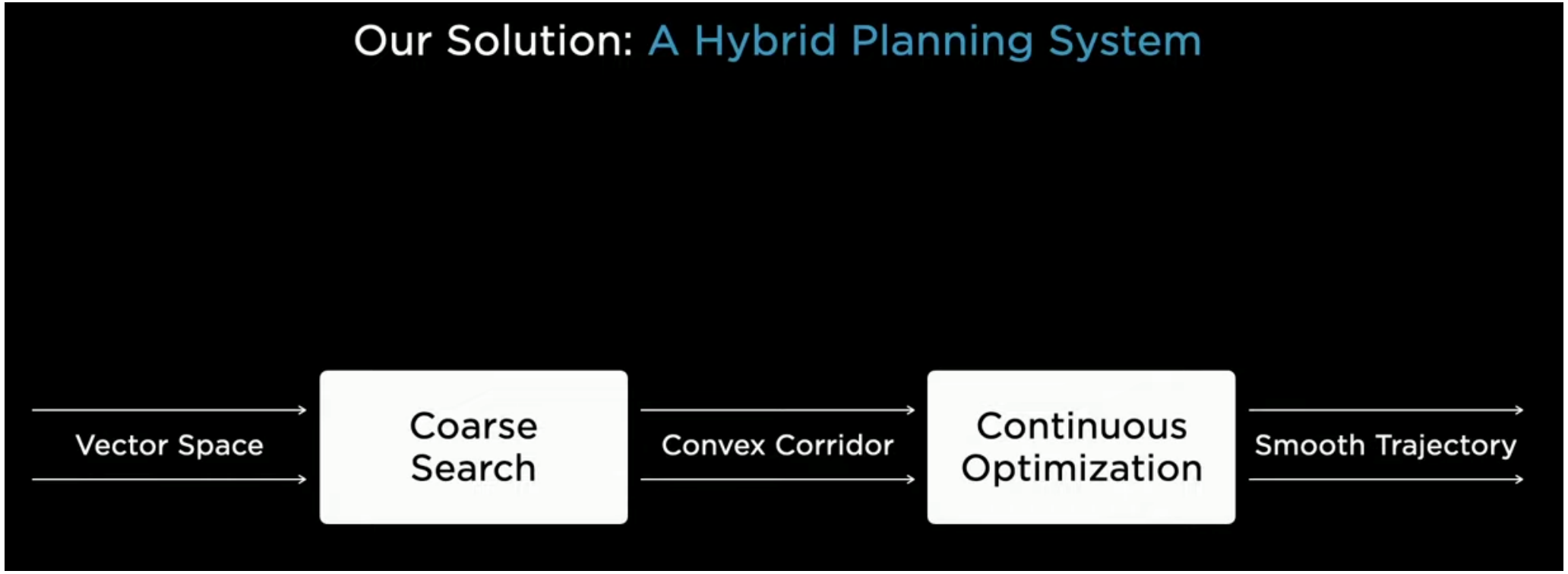
- Even early on, the planning software was good at highway driving, but city driving offers many more challenges- crossing vehicles and pedestrians behaving strangely.

PLANNING

- Non-Convex – multiple possible solutions that can be independently good, globally consistent solution is tricky and pockets of local minima that planning can get stuck in.
- High-Dimensional – the car needs to plan out 10-15 seconds. It must plan position, acceleration, etc.



GO HYBRID

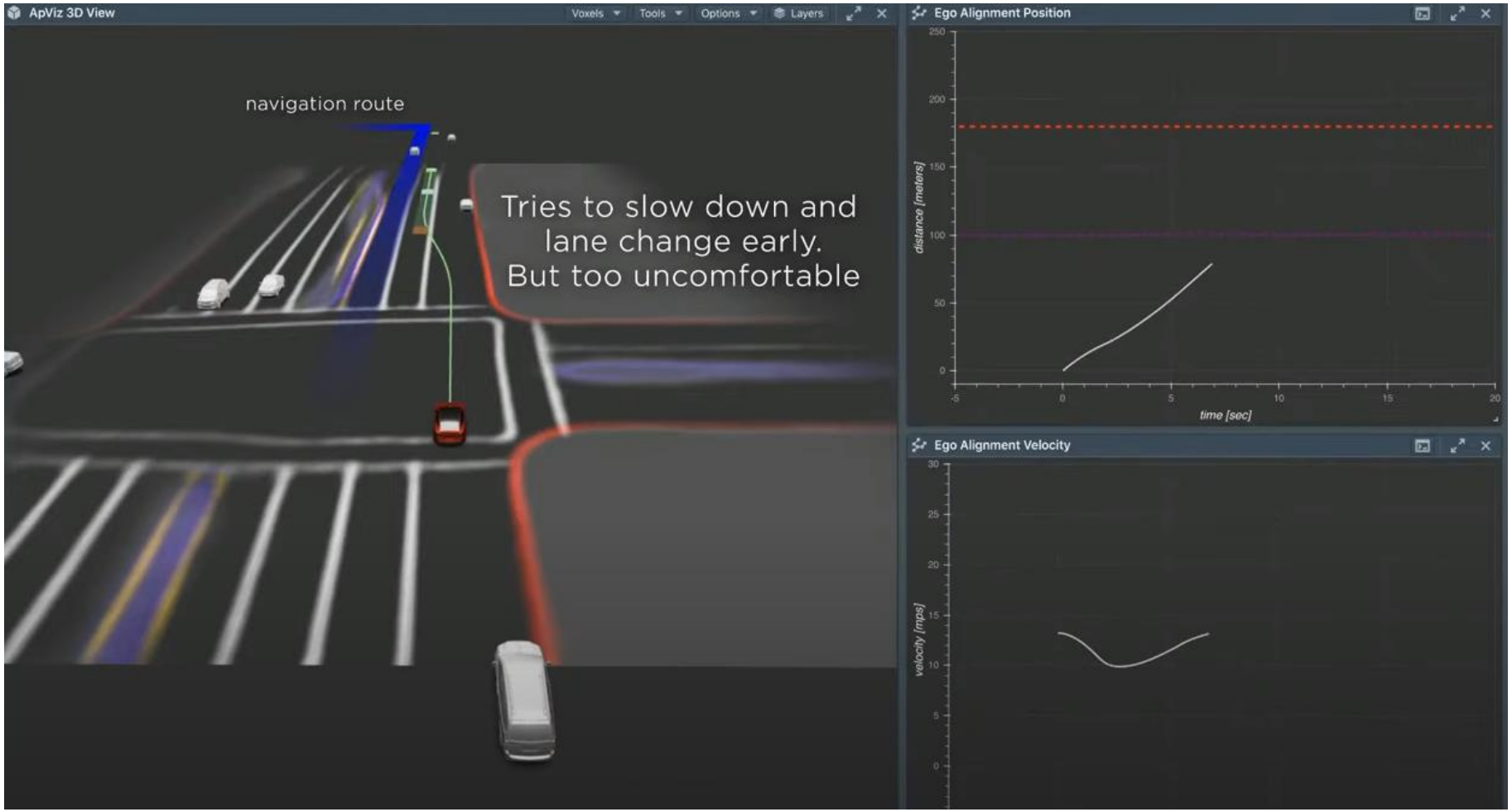


- Hybridb *****

EXAMPLE LANE CHANGE

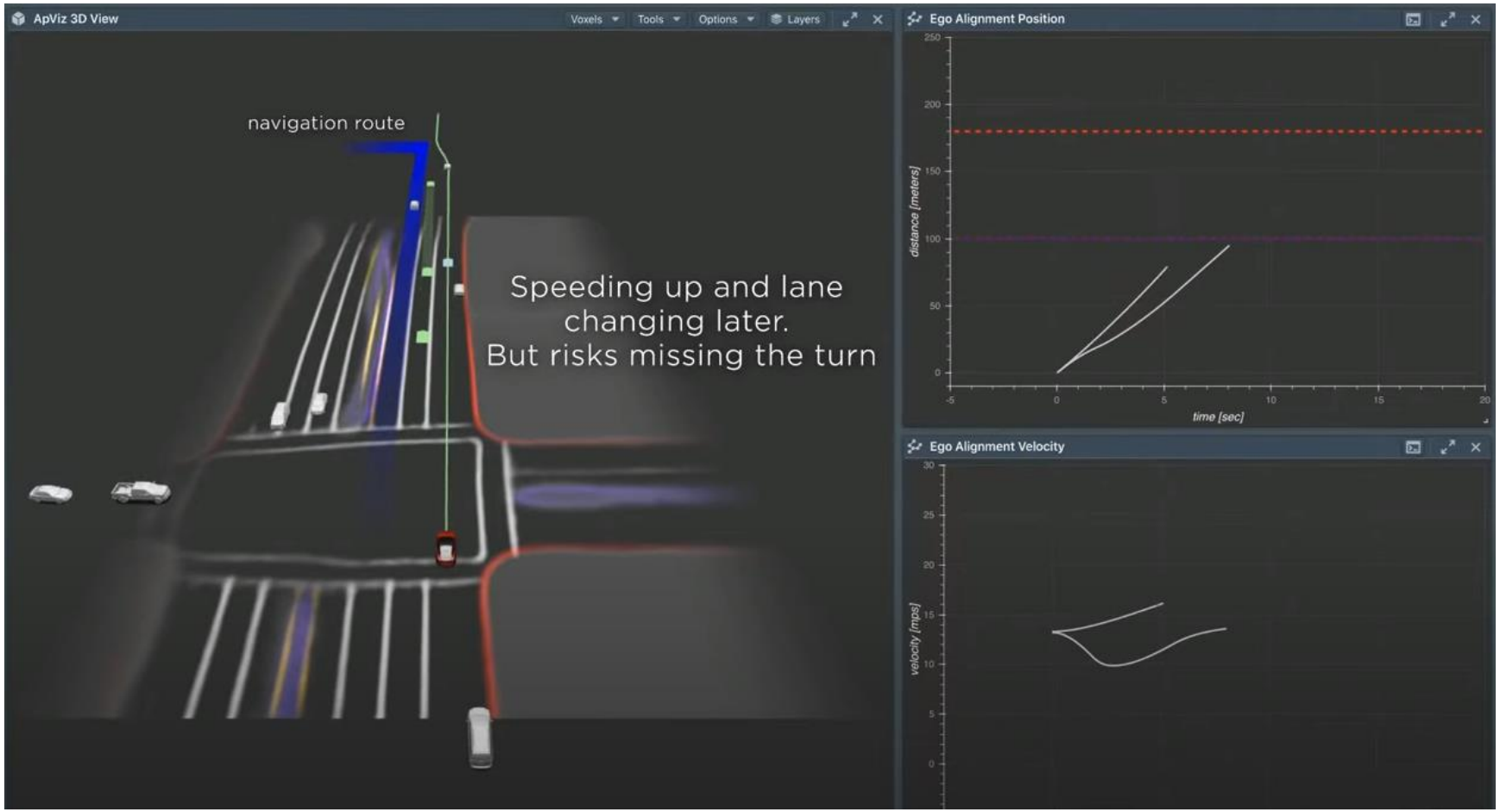
- The Tesla planning outlined and in the next example, planning two back-to-back lane changes to make a left turn. In the first example, the car has to break to harshly to affect a left-hand turn.

LANE PLAN ONE



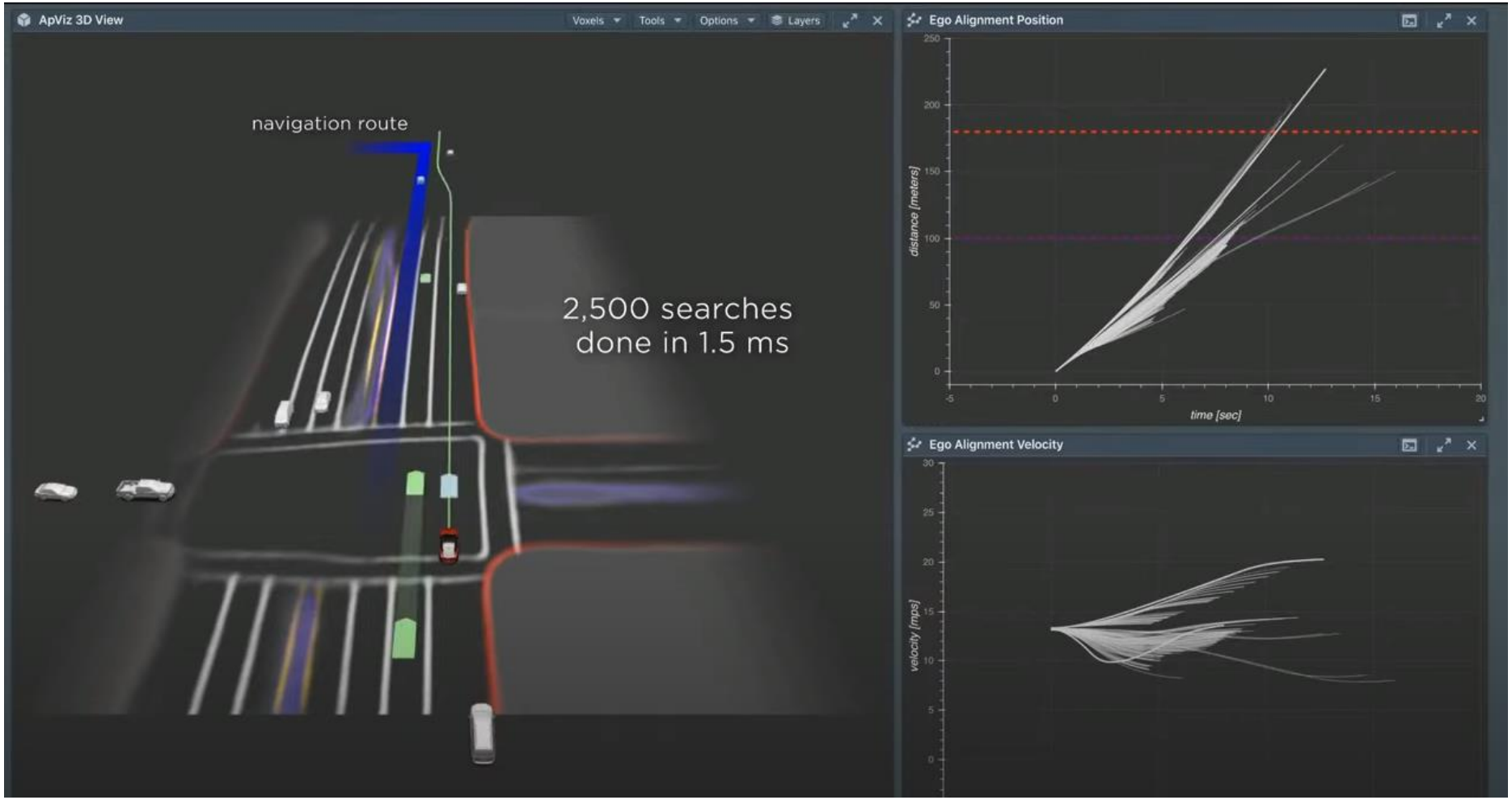
- Next it speeds up too fast, and risks missing the turn altogether.

LANE PLAN TWO



- The system searches and evaluates many options because they are physics based models, the futures are easy to simulate.
- An optimal trajectory is chosen to balance comfort and safety while easily making the turn.

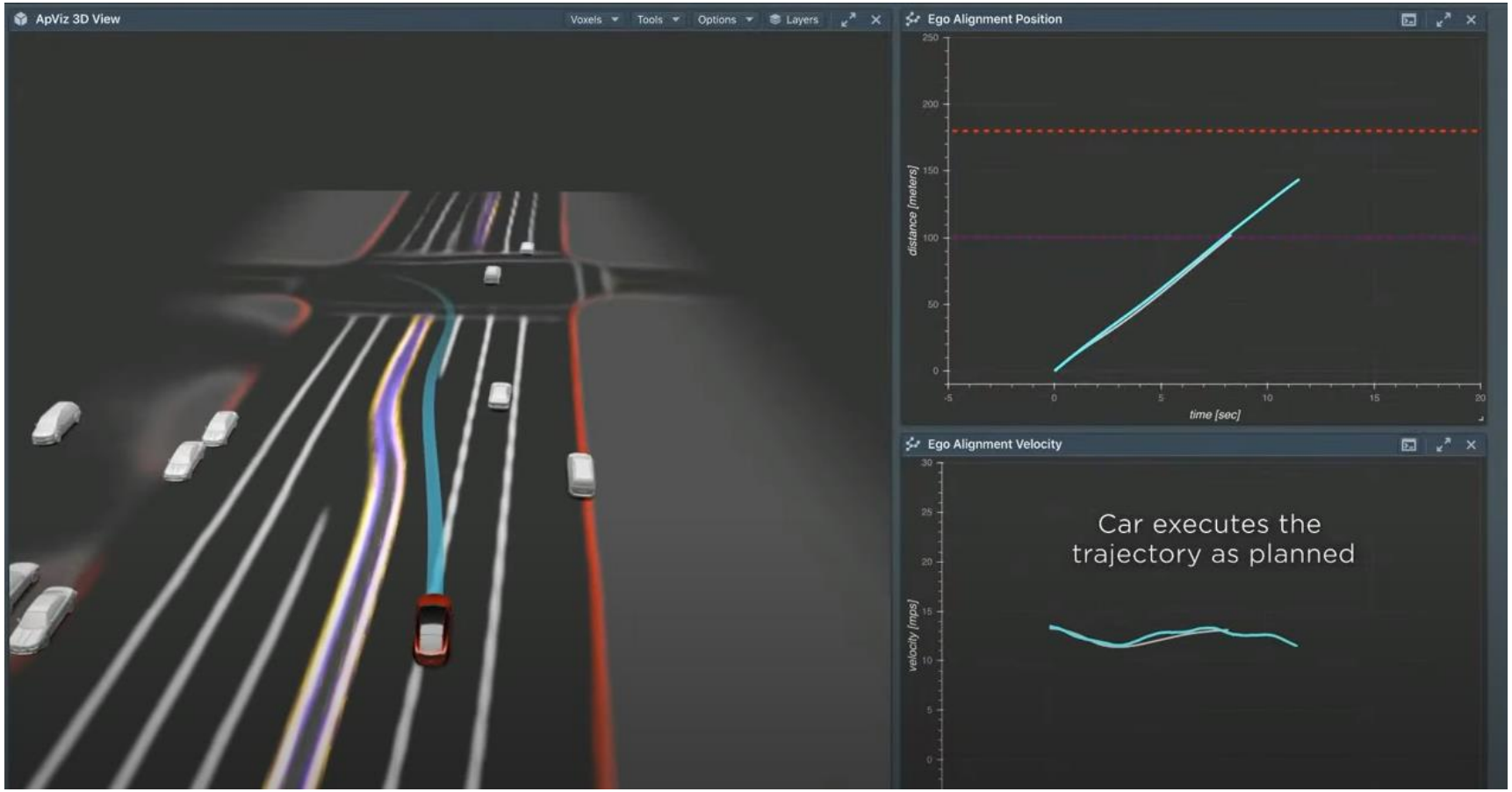
LANE PLAN THREE



- A plan is selected and the outcome is one where the velocity and position align nicely with the predictions.

MULTI-AGENT PLANNING

- When driving alongside other agents, Tesla Autopilot needs to play for itself and every other agent jointly.
- This is accomplished by running the Autopilot planner for every single object in the scene.

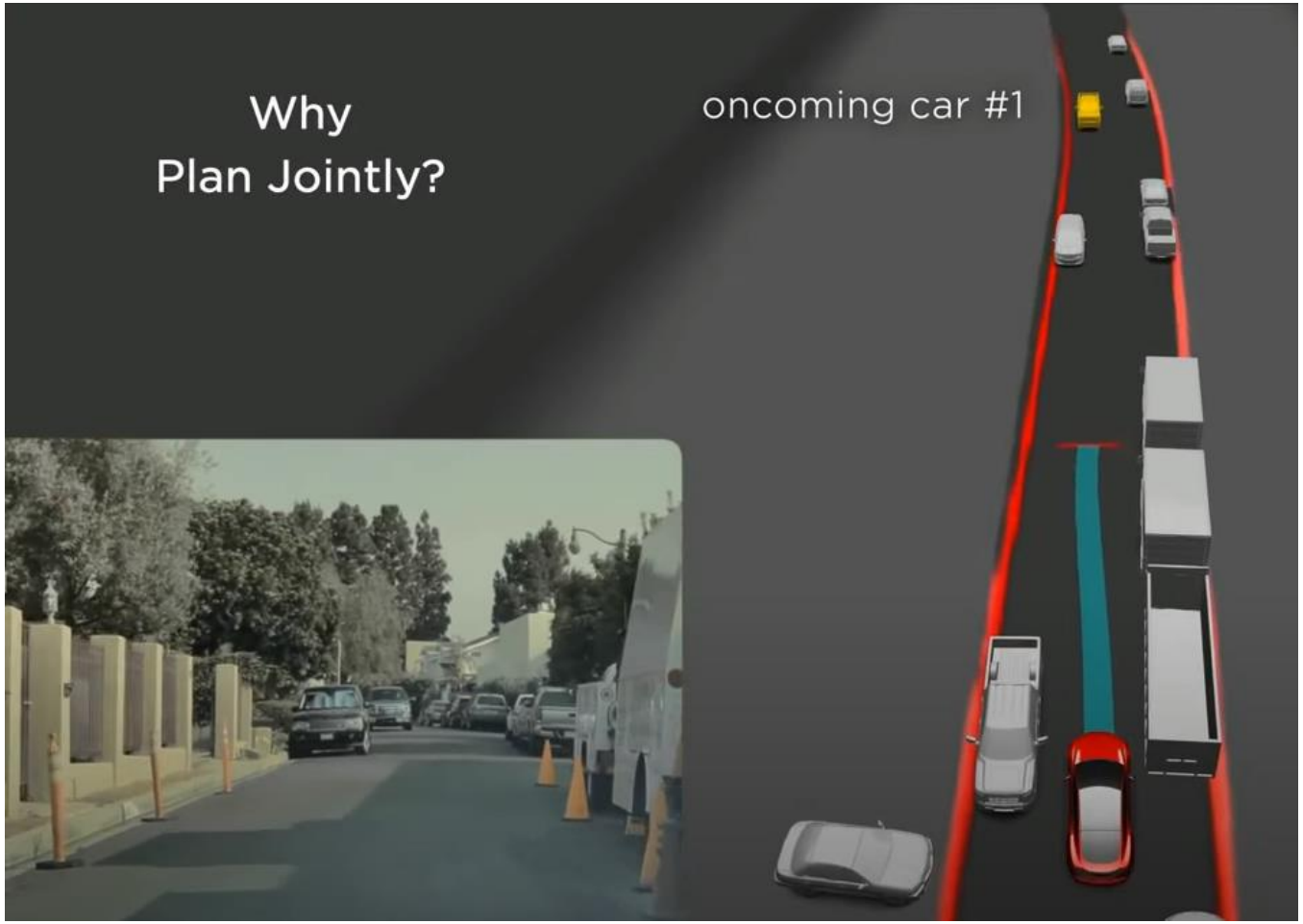


PLAN JOINTLY

- When driving alongside other agents, the system need to plan for itself and every other agent jointly and optimize for the scenes traffic flow.
- This is accomplished by running the Autopilot planner for every single object in the scene.

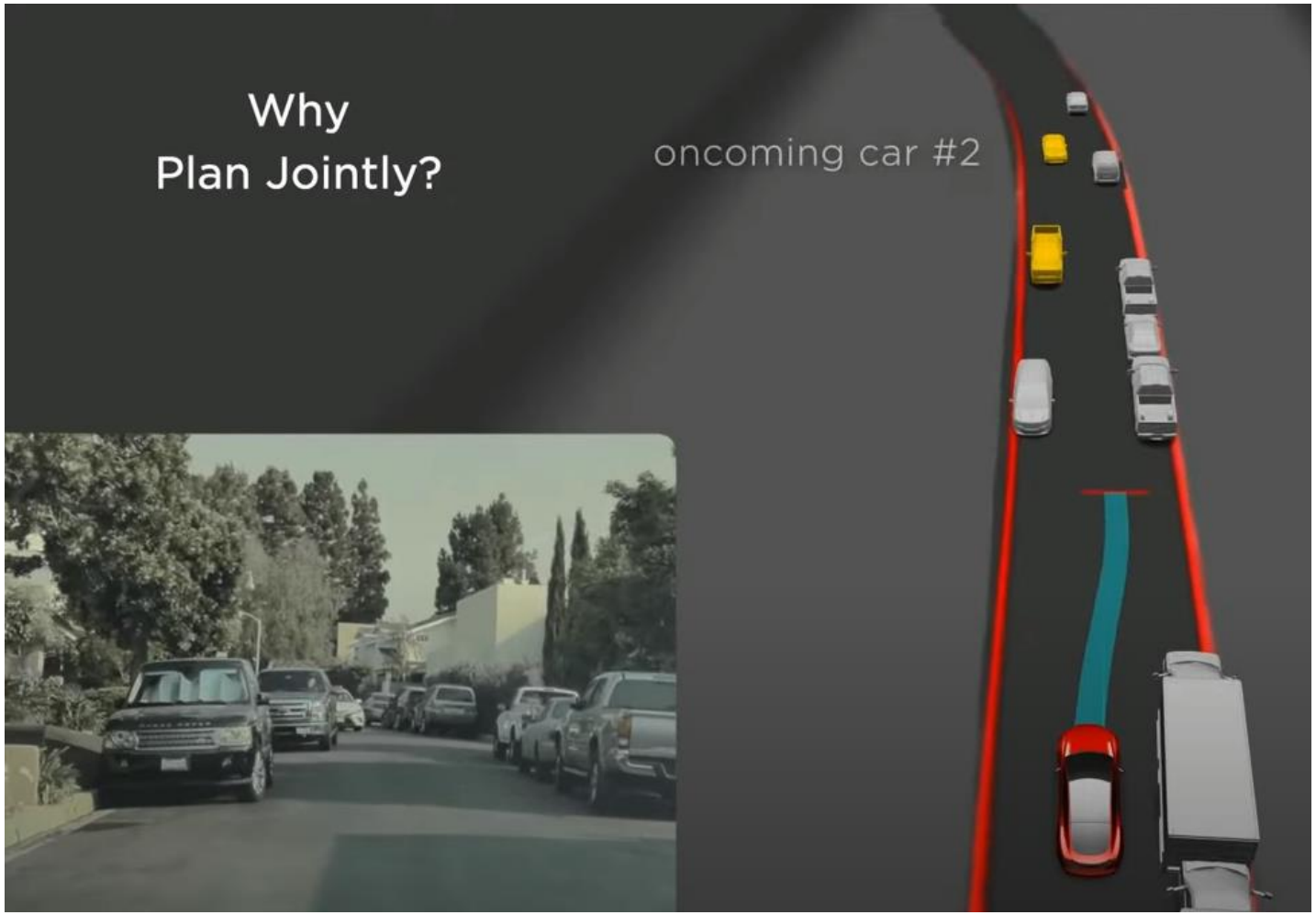
TIGHT LOCATION, CAR APPROACHING

- Autopilot reasons that an approaching car has slow enough velocity that it can pull over and allow the Tesla to continue slowly forward and pass.



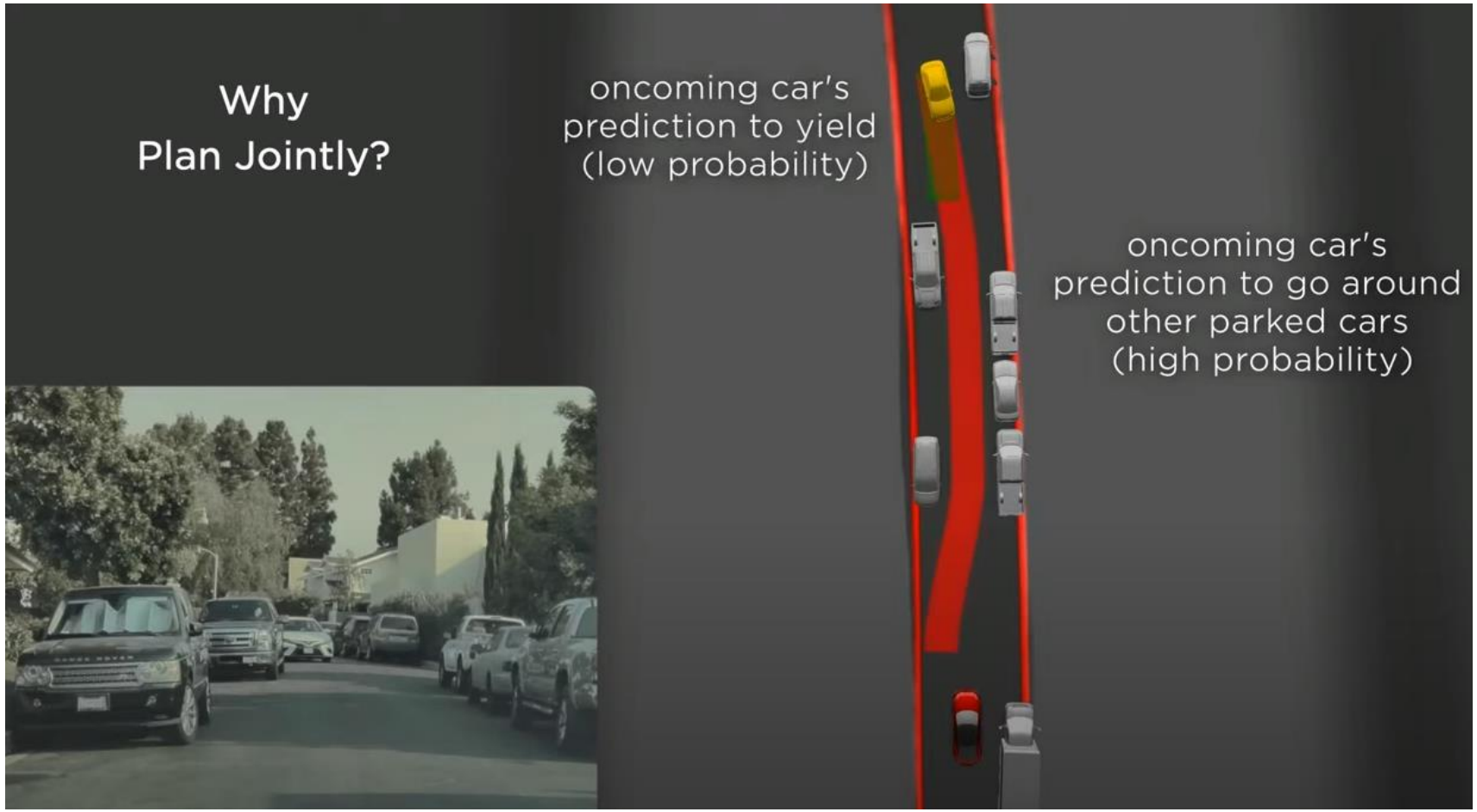
SECOND CAR

- When suddenly a second car comes into view, the Autopilot determines that it is moving very quickly and it needs to adjust its plan.



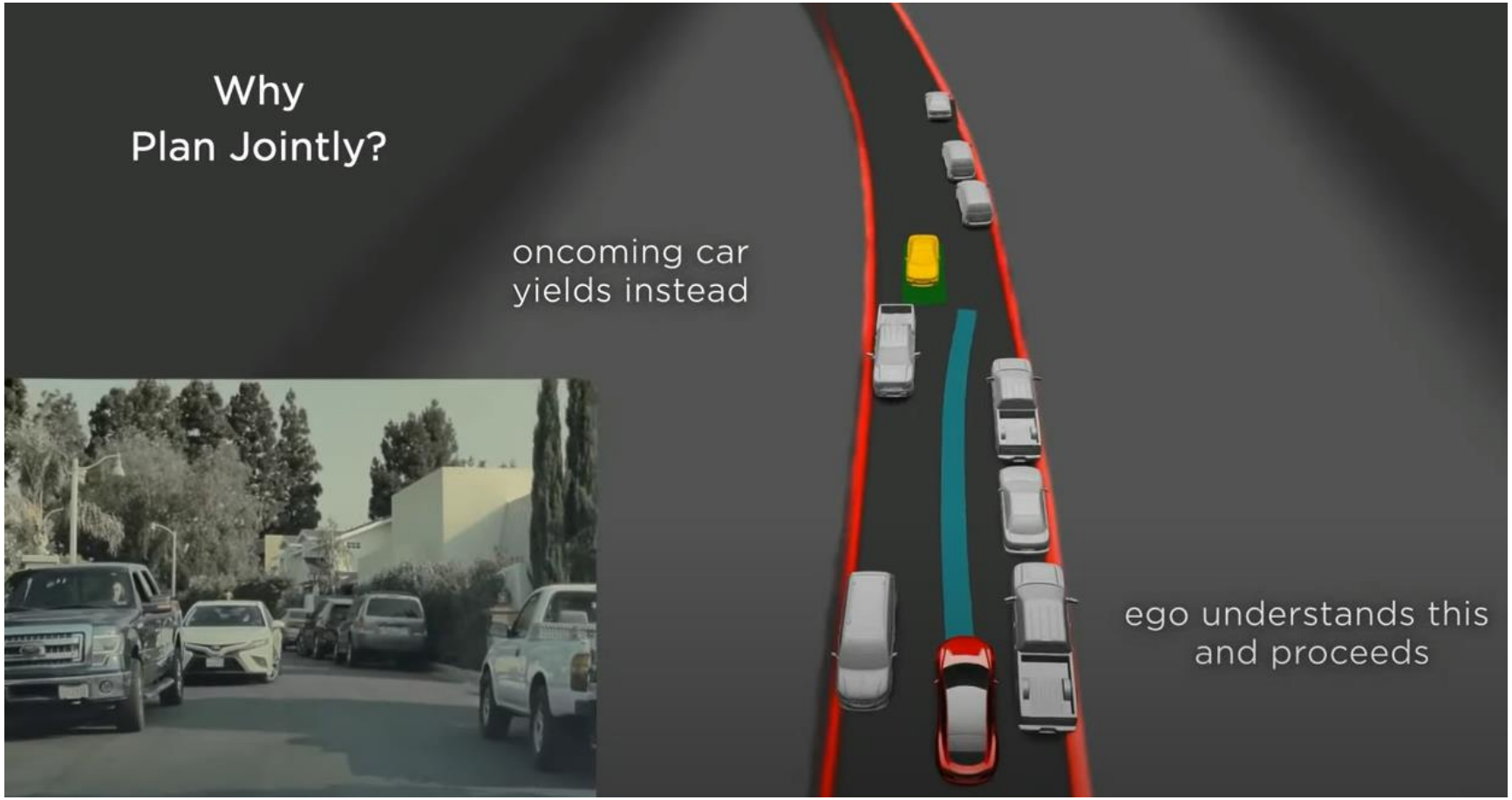
PREDICTING OUTCOMES

- Because of its velocity, there is a higher probability that the car will continue moving toward the Tesla and not yield. Without knowing what is in the mind of the driver, Autopilot calculates multiple future outcomes. Autopilot determines to pull over because there is space.



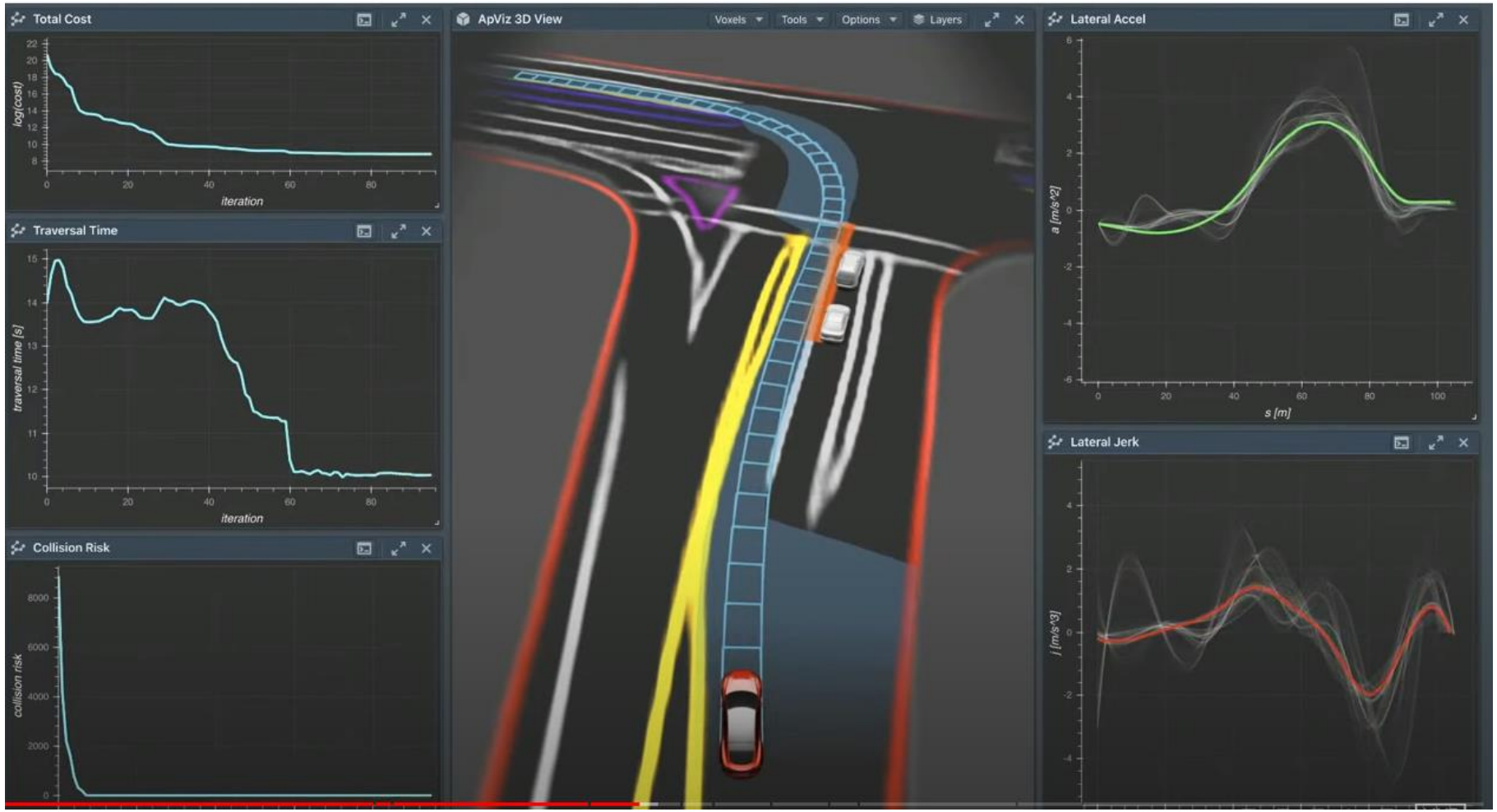
PLAN SWITCH

- As Autopilot pulls over, it determines, based on the yaw and velocity of the oncoming car that it has decided to pull over.
- The selected plan is switched to move forward, if Autopilot did not do this, it would be too timid and would not be a practical self-driving car.



OPTIMIZING FOR SAFETY AND COMFORT

- The Autopilot algorithms strive to produce an outcome that is both safe and comfortable.

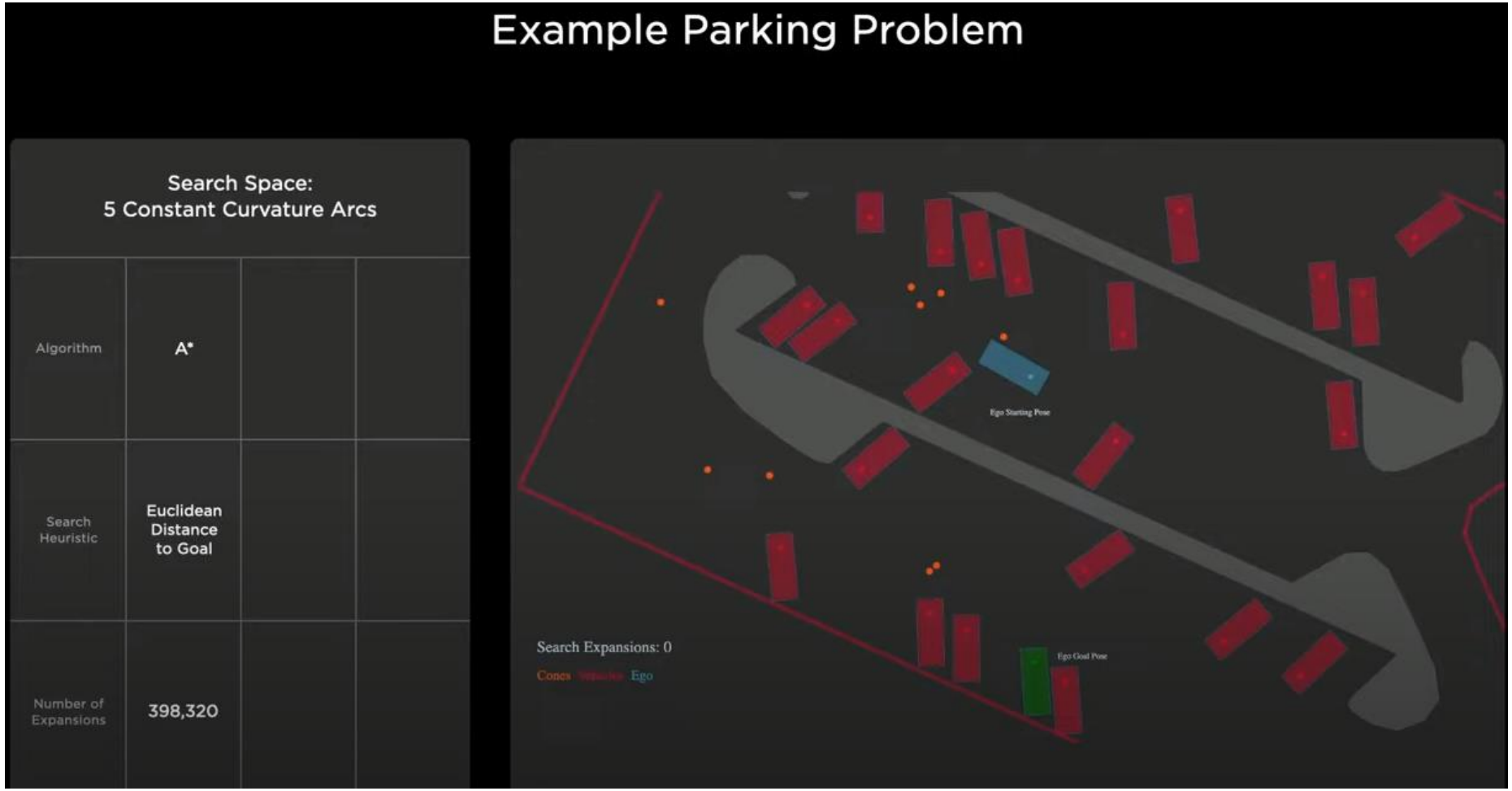


DIFFICULT DRIVING

- More obstructions, many pedestrians, harsh braking, etc. It will be difficult to scale up the system to efficiently handle these situations.

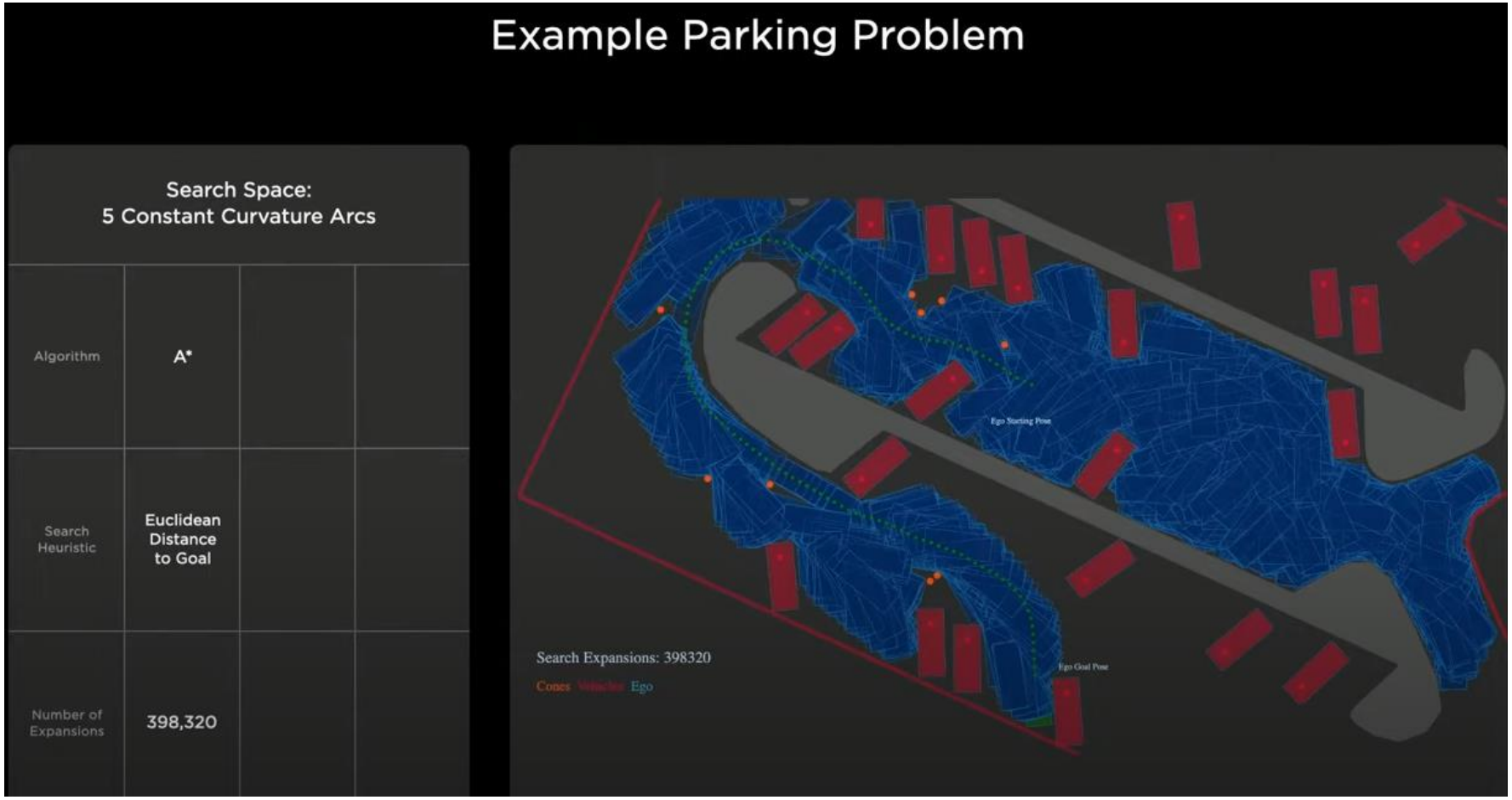
PARKING

- A simpler example demonstrates all of the challenges facing the vehicle. Here a vehicle (blue) must navigate its way to the parking spot (green). It must avoid cones (orange) and parked cars (red).



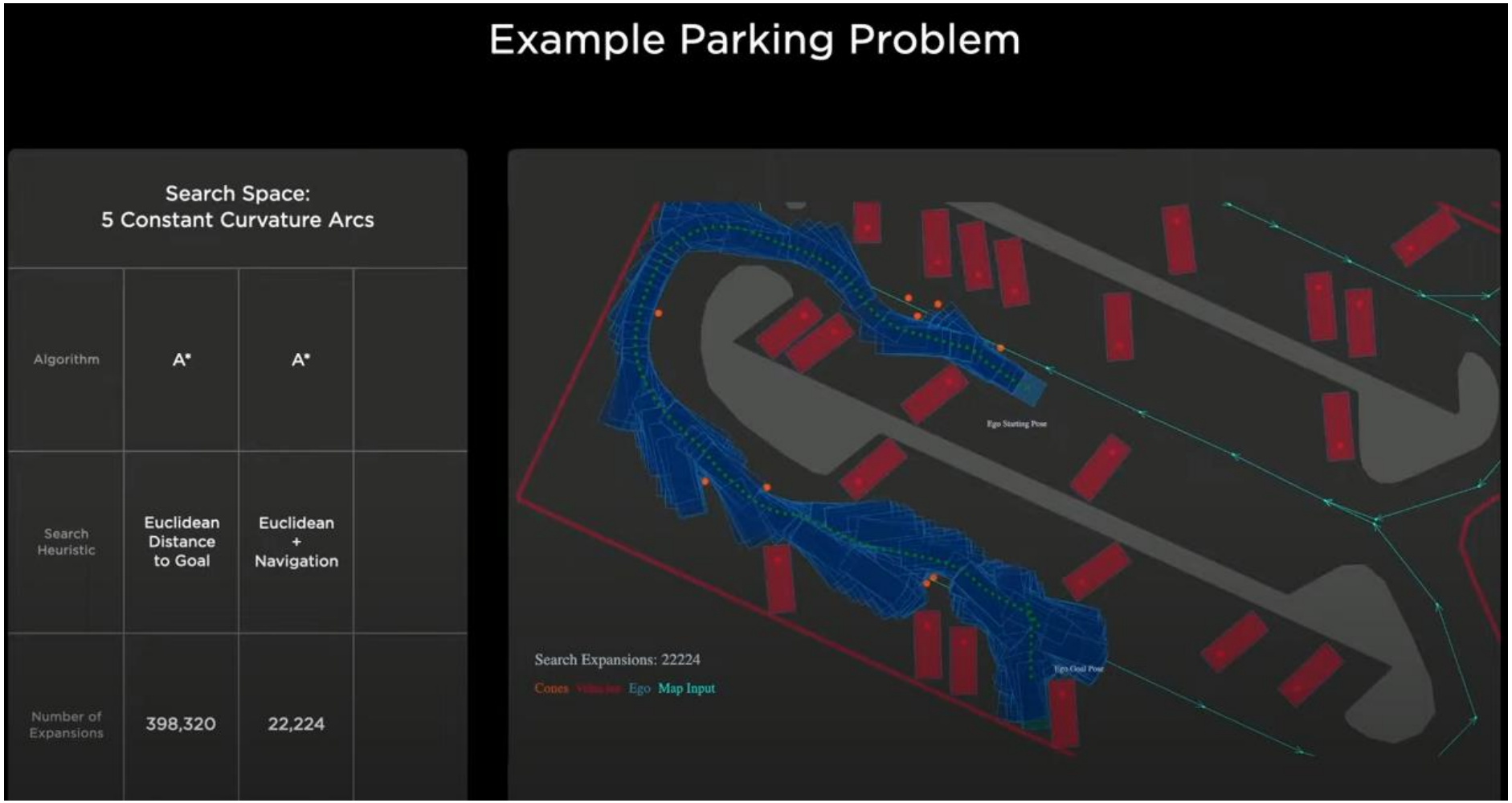
EUCLIDEAN DISTANCE

- Euclidean distance search requires 400K nodes to resolve the parking problem. Every time there is collision, it must back up and try again.

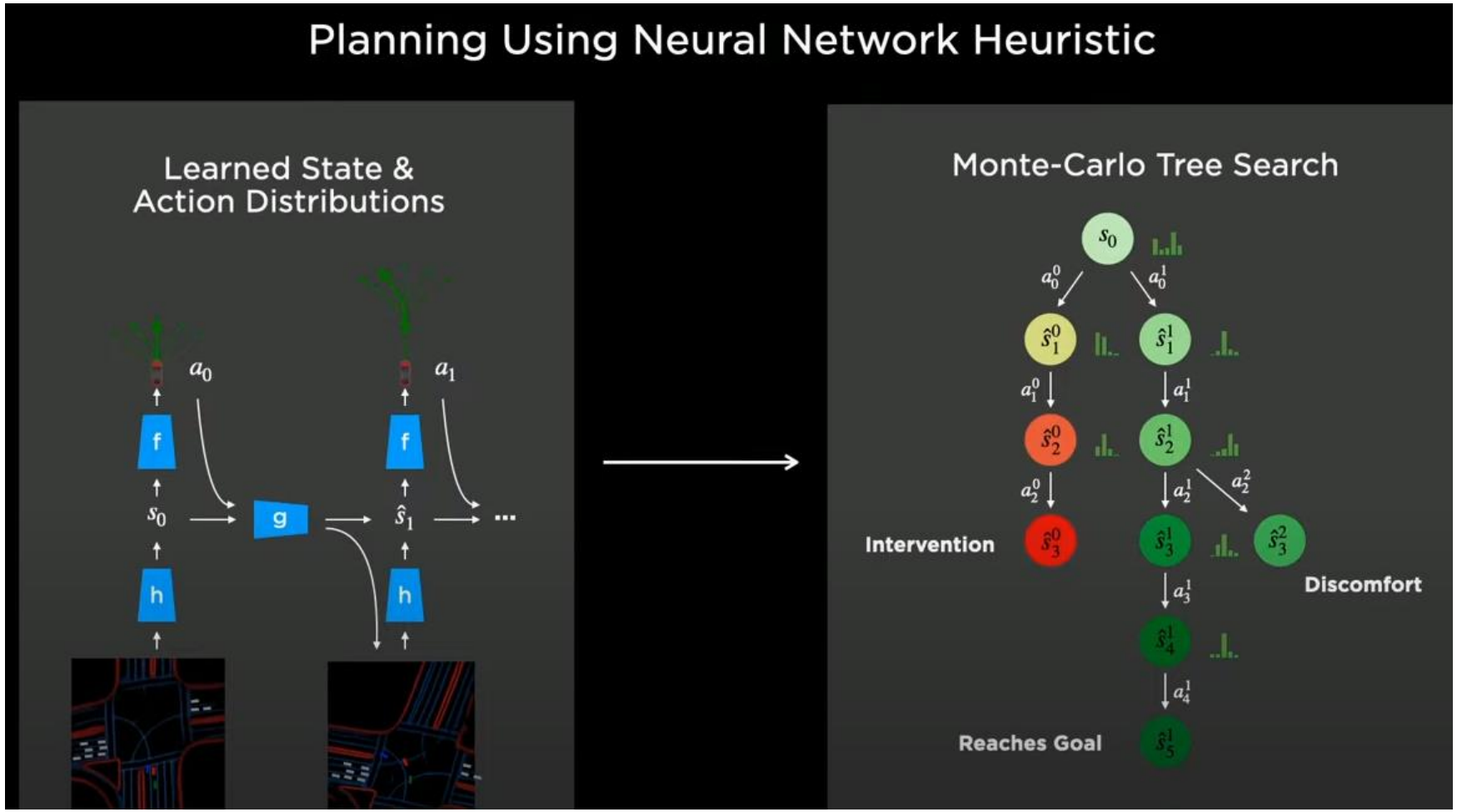


NAVIGATION ROUTE

- Adding a navigation route, helps substantially, but still requires collision and back-up and recalculation. Still requires 22K nodes.

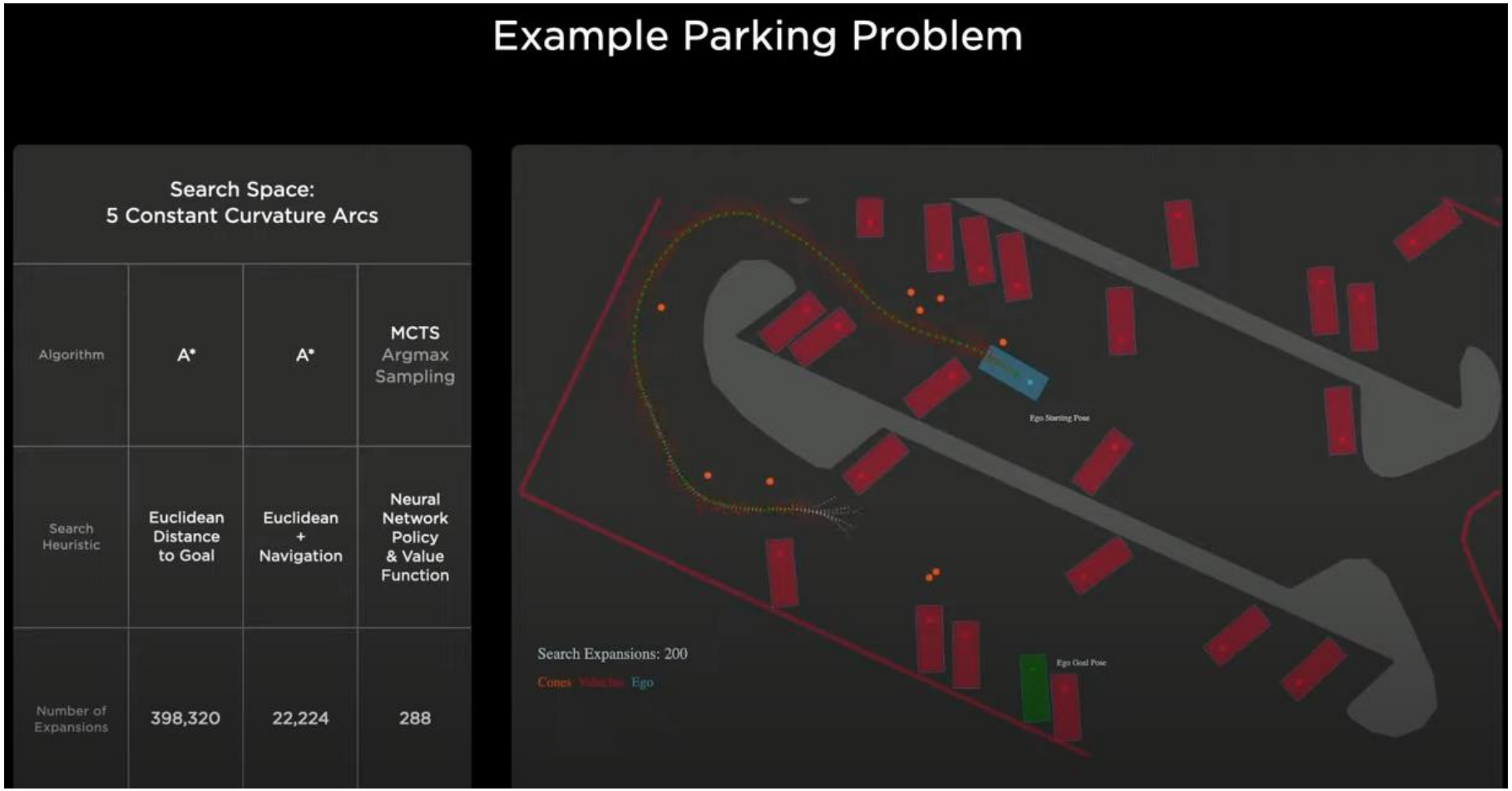


- Many techniques can be applied to help the search, but it is difficult to design a globally optimum heuristic.



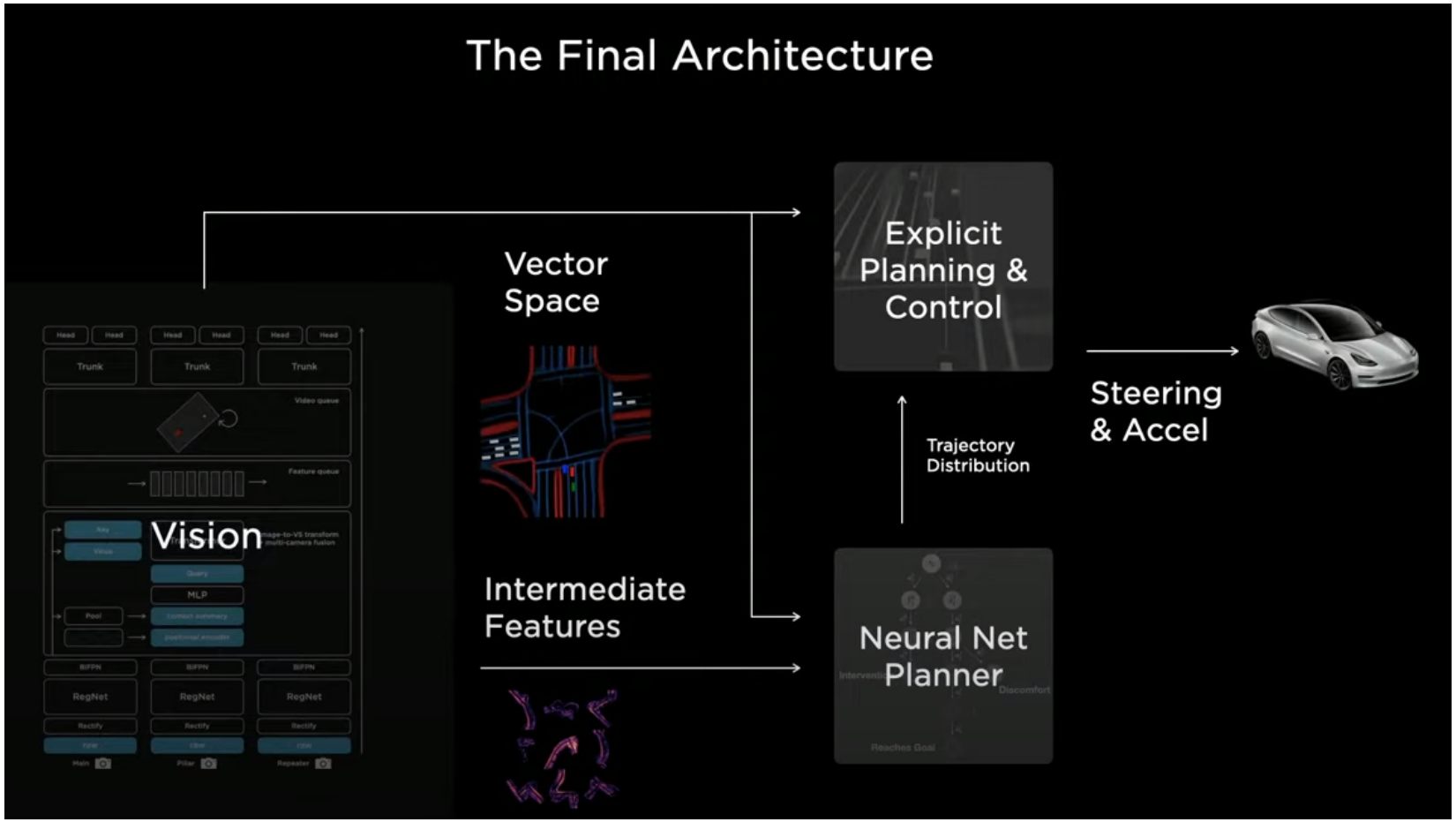
- The approach that was used to solve Go and Atari games using NN that produce state and action distributions that can be plugged into various cost functions. (Cost functions: collision, comfort). The plan is able to make progress toward the goal in one shot and doesn't use a navigation heuristic. The NN is able to consider the global context of the scene and produces a value function which effectively guides it to the global minima (destination) and does not get it stuck in any local minima- using only 288 nodes.

- asd


Example Parking Problem
Search Space:
5 Constant Curvature Arcs
Algorithm
A*
A*
MCTS
Argmax
Sampling
Search
Heuristic
Euclidean
Distance
to Goal
Euclidean
+
Navigation
Neural
Network
Policy
& Value
Function
Number of
Expansions
398,320
22,224
288
Ego Starting Pose
Search Expansions: 200
Ego Goal Pose
Ego

REVIEW

- The vision system crushes the visual inputs down into a vector space, etc. Etc.



THANK YOU

PART TWO WILL BE PRESENTED NEXT TIME WE MEET.