EECS 4422/5323 Computer Vision

Spatiotemporal Vision 2

Calden Wloka

18 November, 2019

Announcements

- Assignment 2 due today by midnight
- Midterm marks were sent out on Friday

Some Notes on Course Grades

- Grade adjustments are unlikely
  - We are over the size threshold for grade scrutiny
  - We have an unusual distribution which is top-heavy

Some Notes on Course Grades

- Grade adjustments are unlikely
  - We are over the size threshold for grade scrutiny
  - We have an unusual distribution which is top-heavy
- We still have 40% of the grade to go (counting Assignment 2)

Some Notes on Course Grades

- Grade adjustments are unlikely
  - We are over the size threshold for grade scrutiny
  - We have an unusual distribution which is top-heavy
- We still have 40% of the grade to go (counting Assignment 2)
- Turn things in! Even a failing mark of 50% is far better than a zero

Outline

- LSTMs
- Tracking
  - Challenges
  - Visualization
  - Dynamic Filters
  - Kalman Filter

Long Short-Term Memory (LSTM) Modules

As we mentioned earlier, LSTMs are a popular method of adding recurrence to deep networks.

Recurrence is useful for many vision problems, but carries particular importance in video processing (we want information from earlier frames to influence our processing of current frames).

Note: Information and images in the next several slides is adapted from Chris Olah's work.

"Standard" Recurrence

A recurrent neural network feeds information from higher in its hierarchy back toward earlier layers.

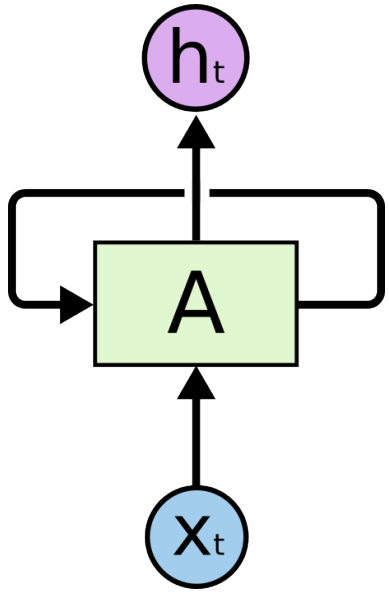


Image source: Olah, 2015

"Standard" Recurrence

This can be viewed as equivalent to running a chain of networks, with prior output serving as part of the network input.

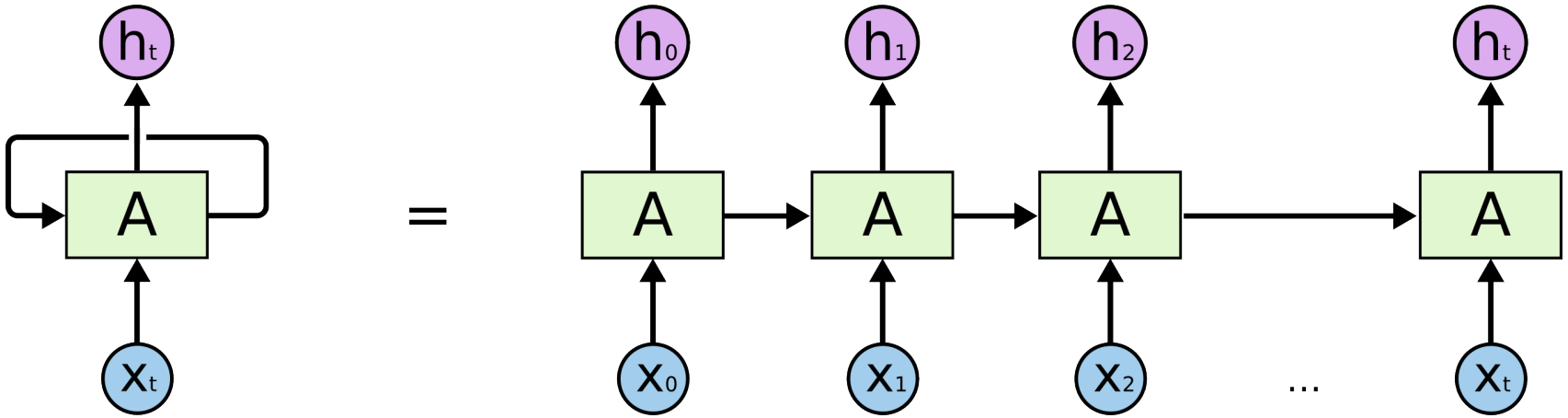


Image source: Olah, 2015

How to Link Far Removed Information?

It is clear that information from one or two time steps removed can easily affect the output of a given time step, but what if the connection is over many time steps? The link tends to get weaker as events are further removed, and even if we want to keep information around it is challenging to do so without adversely affecting other processing.

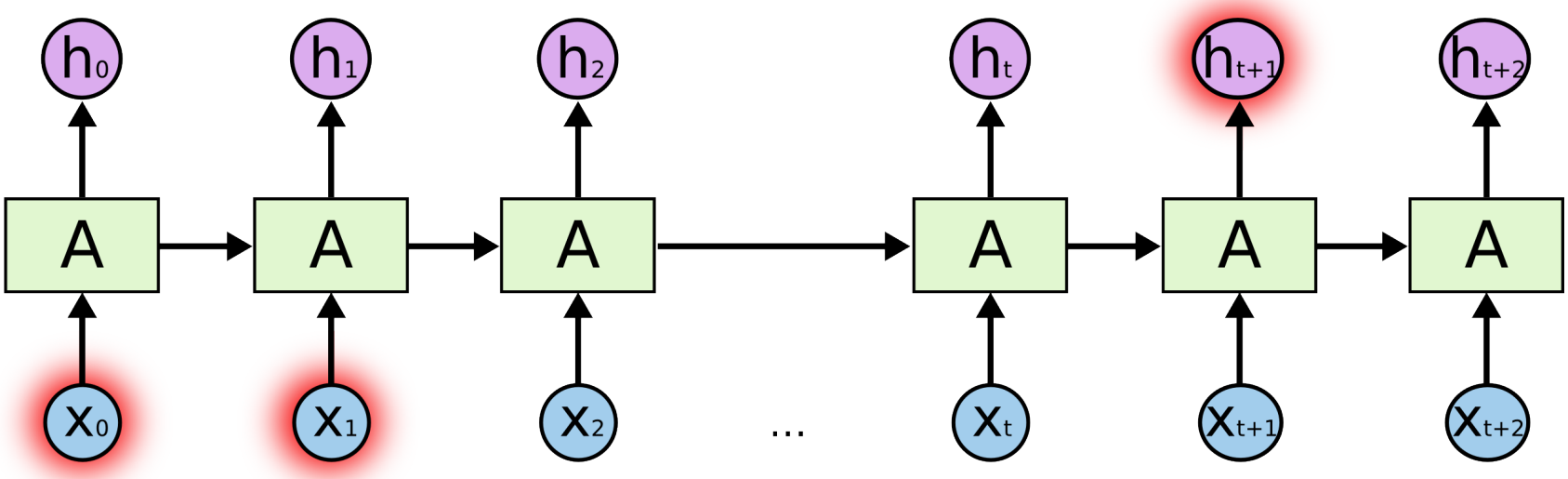


Image source: Olah, 2015

LSTM: Add Some Control Over Information Retention

Rather than simply feeding information directly through, an LSTM unit adds a series of *gates* and an additional information track to control the flow and retention of information.

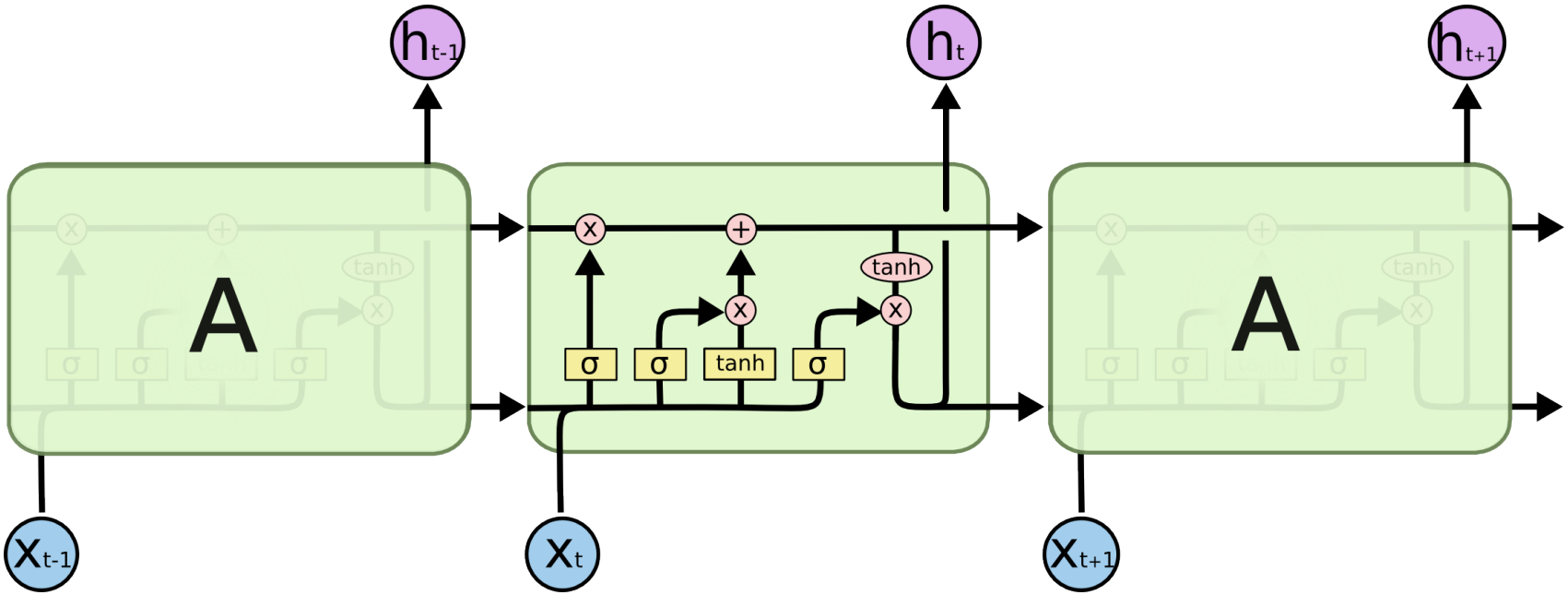


Image source: Olah, 2015

LSTM: The Cell State

Each LSTM unit has an associated "cell state", which carries information through time associated with that particular LSTM unit.

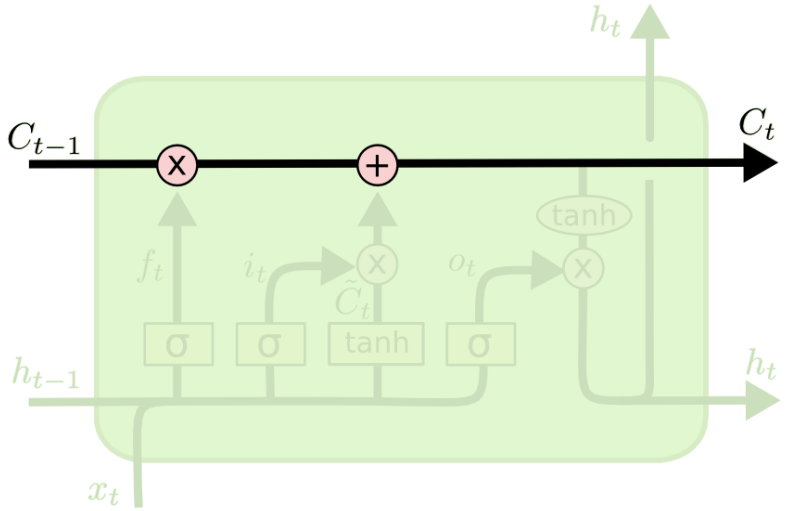
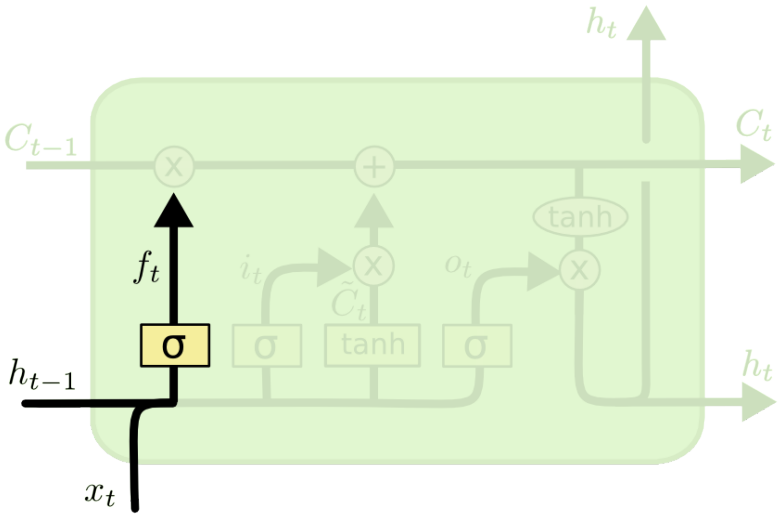


Image source: Olah, 2015

LSTM: Forgetting

The first gate is the "forget" gate, which combines information from the previous output (h_{t-1}) and the current input (x_t) to output sigmoid function which is multiplied by the incoming cell state.

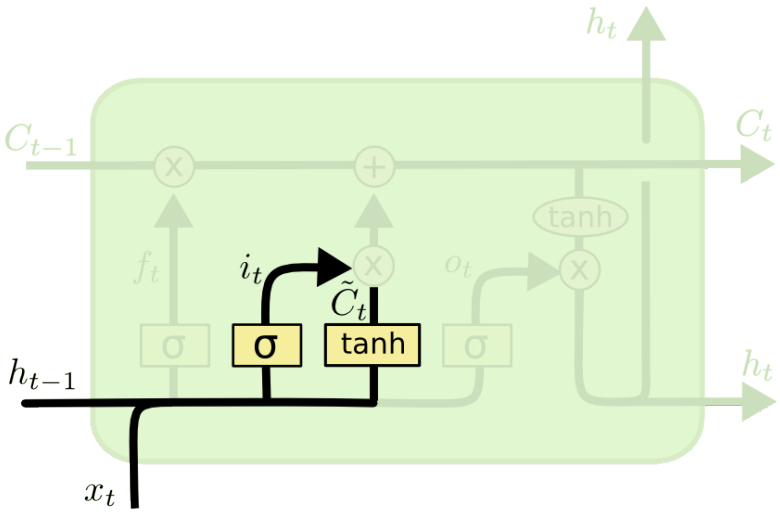


$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

Image source: Olah, 2015

LSTM: Remembering

After we have reduced the cell state through the forget gate, we need to determine what new information will be added to update it. This uses two different non-linearities: the sigmoid layer outputs how much of each new cell state feature to let through, and the tanh layer outputs the transformation of the input and previous output vectors into a new candidate cell state.



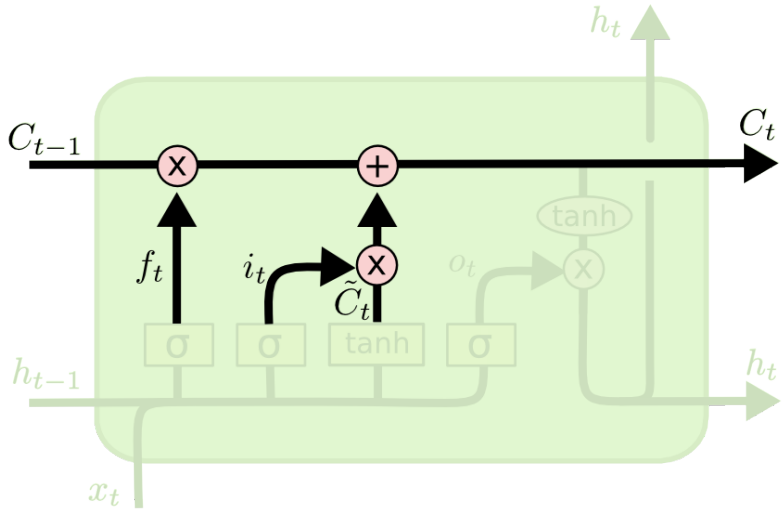
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

Image source: Olah, 2015

LSTM: Modifying the Cell State

Here we compute our new cell state based on the combined effects of the forgetting and remembering gates.

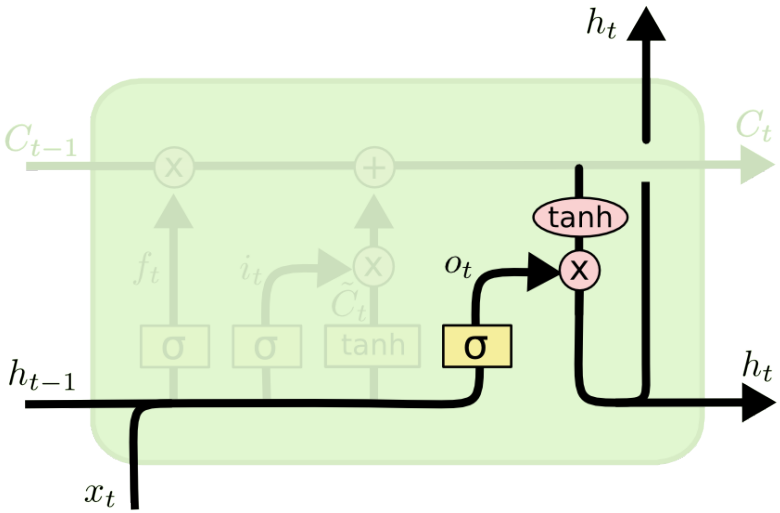


$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Image source: Olah, 2015

LSTM: The Output

Once the cell state is updated, we need to compute the actual output of our unit. The output is an interaction between the cell state and the input/previous output vector.



$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

Image source: Olah, 2015

LSTM Closing Thoughts

- LSTMs are one way to model information carry-over through time
- There are a number of common variants of the LSTM not covered here

A Common Problem in Computer Vision: Tracking

A common problem in video processing is *object tracking*.

For some set of targets, we want to be able to track a given instance of a target across all frames it is visible in.

Example Video

Some Common Problem Formulations

- Active (*e.g.* person following robot) vs. Passive (*e.g.* security camera)

Some Common Problem Formulations

- Active (*e.g.* person following robot) vs. Passive (*e.g.* security camera)
- Number and type of objects

Some Common Problem Formulations

- Active (*e.g.* person following robot) vs. Passive (*e.g.* security camera)
- Number and type of objects
  - Single object tracking

Some Common Problem Formulations

- Active (*e.g.* person following robot) vs. Passive (*e.g.* security camera)
- Number and type of objects
  - Single object tracking
  - Object class tracking

Some Common Problem Formulations

- Active (*e.g.* person following robot) vs. Passive (*e.g.* security camera)
- Number and type of objects
  - Single object tracking
  - Object class tracking
  - Multi-object tracking

Tracking can be Difficult

- Target may move unpredictably

Tracking can be Difficult

- Target may move unpredictably
- Target features may not be constant

Tracking can be Difficult

- Target may move unpredictably
- Target features may not be constant
- Occlusions may obscure part or all of a target

Tracking can be Difficult

- Target may move unpredictably
- Target features may not be constant
- Occlusions may obscure part or all of a target
- Other instances of a given object class may be present

Some Example Tracking

Example tracking, MDNet, Nam & Han, 2015.

Example person following, Chen *et al.*, 2017.

Dealing with target changes

Most object trackers operate with the primary goal of distinguishing a target from background.

A useful assumption is that, though objects may change through time, the changes from frame to frame should be small enough that we don't lose track of our target.

Dealing with target changes

Most object trackers operate with the primary goal of distinguishing a target from background.

A useful assumption is that, though objects may change through time, the changes from frame to frame should be small enough that we don't lose track of our target.

We can then update our detector to discriminate the target's appearance based on the most recent frames.

Detecting dynamic objects

Differentiating between target and background essentially boils down to a 2-class classification problem. There are many different types of classifier which can be used, with the choice often dictated by some combination of hardware available, execution speed, challenge of the task, and accuracy requirements.

- CNN-based trackers

Detecting dynamic objects

Differentiating between target and background essentially boils down to a 2-class classification problem. There are many different types of classifier which can be used, with the choice often dictated by some combination of hardware available, execution speed, challenge of the task, and accuracy requirements.

- CNN-based trackers
  - Online learning

Detecting dynamic objects

Differentiating between target and background essentially boils down to a 2-class classification problem. There are many different types of classifier which can be used, with the choice often dictated by some combination of hardware available, execution speed, challenge of the task, and accuracy requirements.

- CNN-based trackers
  - Online learning
  - Pre-trained deep features, only updating readout layer

Detecting dynamic objects

Differentiating between target and background essentially boils down to a 2-class classification problem. There are many different types of classifier which can be used, with the choice often dictated by some combination of hardware available, execution speed, challenge of the task, and accuracy requirements.

- CNN-based trackers
  - Online learning
  - Pre-trained deep features, only updating readout layer
- Correlation in feature space (*e.g.* HoG features)

A brief side note: HoGs and Visualizatio

Histogram of Gradients (HoG) features are a commonly used feature set in non-deep learning based object tracking, as they can be quickly computed and are relatively powerful.
However, like many computer vision methods, they can be somewhat hard to wrap your head around because of the amount of information which is being represented and simultaneously concatenated to create the feature representation.

The Usefulness of Visualization

We've mentioned visualization before in the context of deep networks, but this is a general aspect of computer vision research which is worth noting.

Designing a good visualization is non-trivial; it typically requires a solid understanding of how a particular method operates in order to determine how best to render that information to a user through images.

The Usefulness of Visualization

We've mentioned visualization before in the context of deep networks, but this is a general aspect of computer vision research which is worth noting.

Designing a good visualization is non-trivial; it typically requires a solid understanding of how a particular method operates in order to determine how best to render that information to a user through images.

A good visualizer, however, can be extremely useful! Visually inspecting what your method is doing is a great debugging tool, and can also sometimes lead to new insights for tweaks and improvements to a given model.

HoG Examples

HoGs are quite useful, but can sometimes lead to unexpected failures.

Image source: Vondrick *et al.*, 2015

Why is that water being picked up as a car?

Put on our HOGles

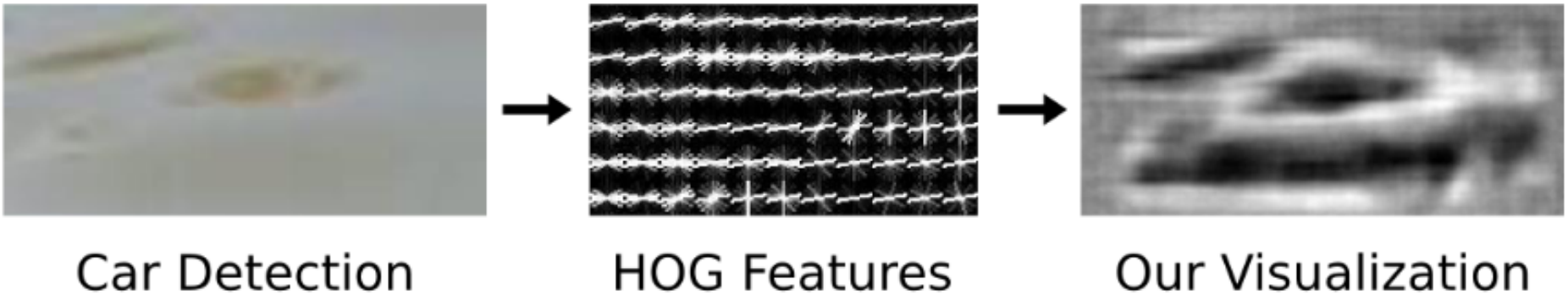


Image source: Vondrick *et al.*, 2015

By visualizing how the features are being represented, we can better understand why the method is failing and potentially think of ways to mitigate the problem.

Uncertainty through time

When trying to estimate information through time, after initialization we don't start from scratch, but each time step adds the possibility for change.

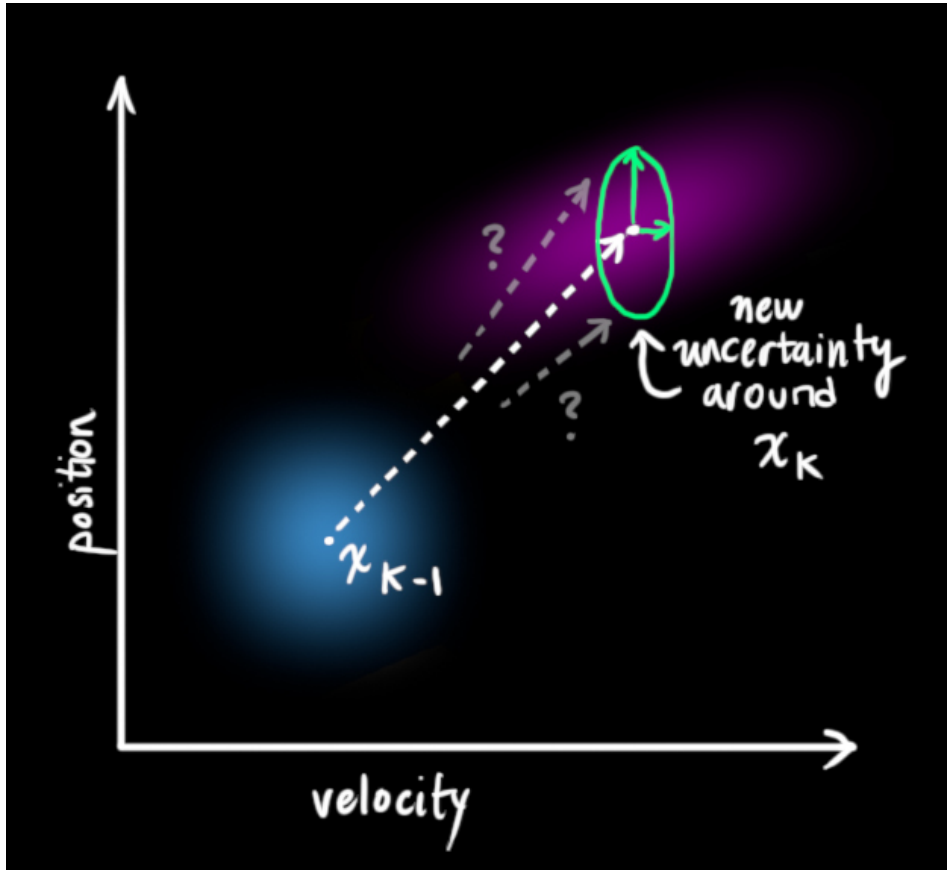


Image Source: Babb, 2015

Adding a new measurement

We can model an incoming reading (*e.g.* the location predicted by our tracker) with its own degree of uncertainty.

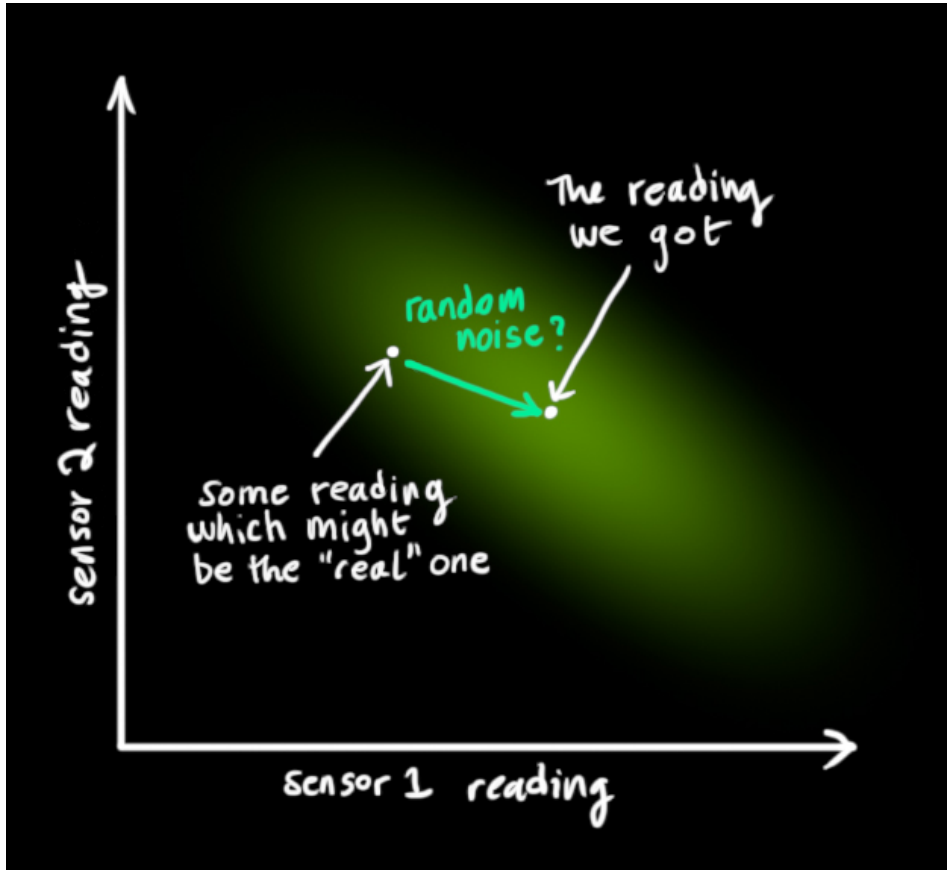


Image Source: Babb, 2015

Combining prior knowledge with new measurements

If pink represents our distribution based on our knowledge from previous frames, and green the distribution from our new reading, then we can multiply them to get the new distribution.

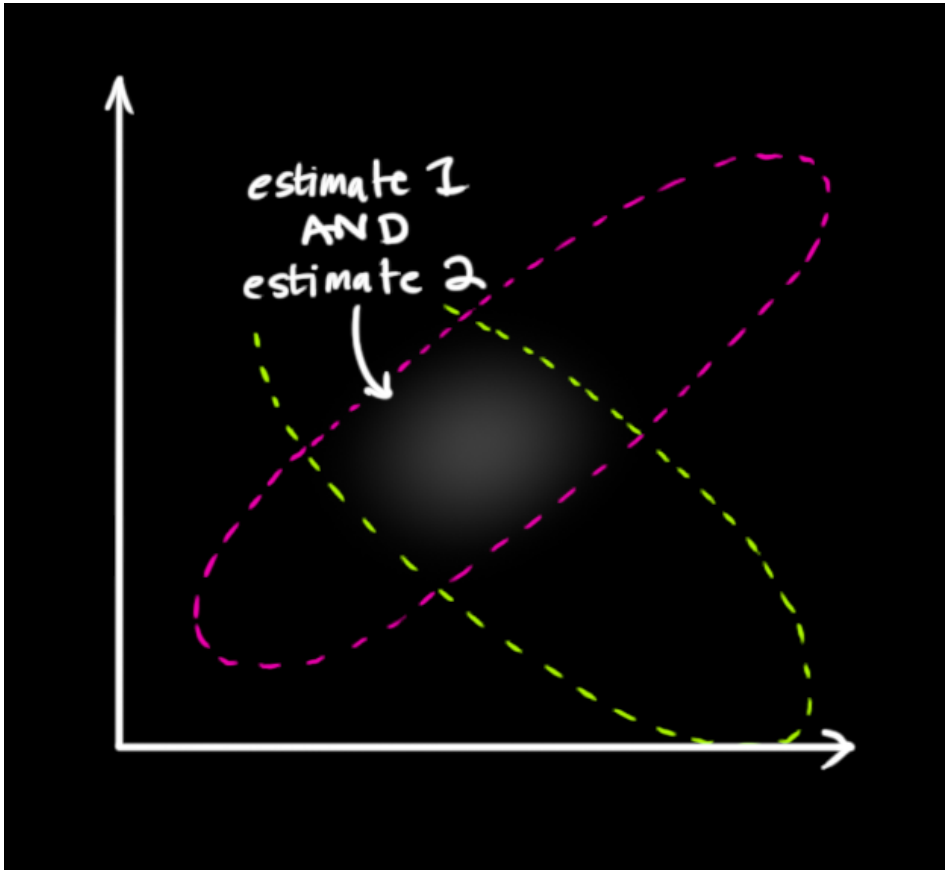


Image Source: Babb, 2015

The Kalman filter assumes Gaussian distributions

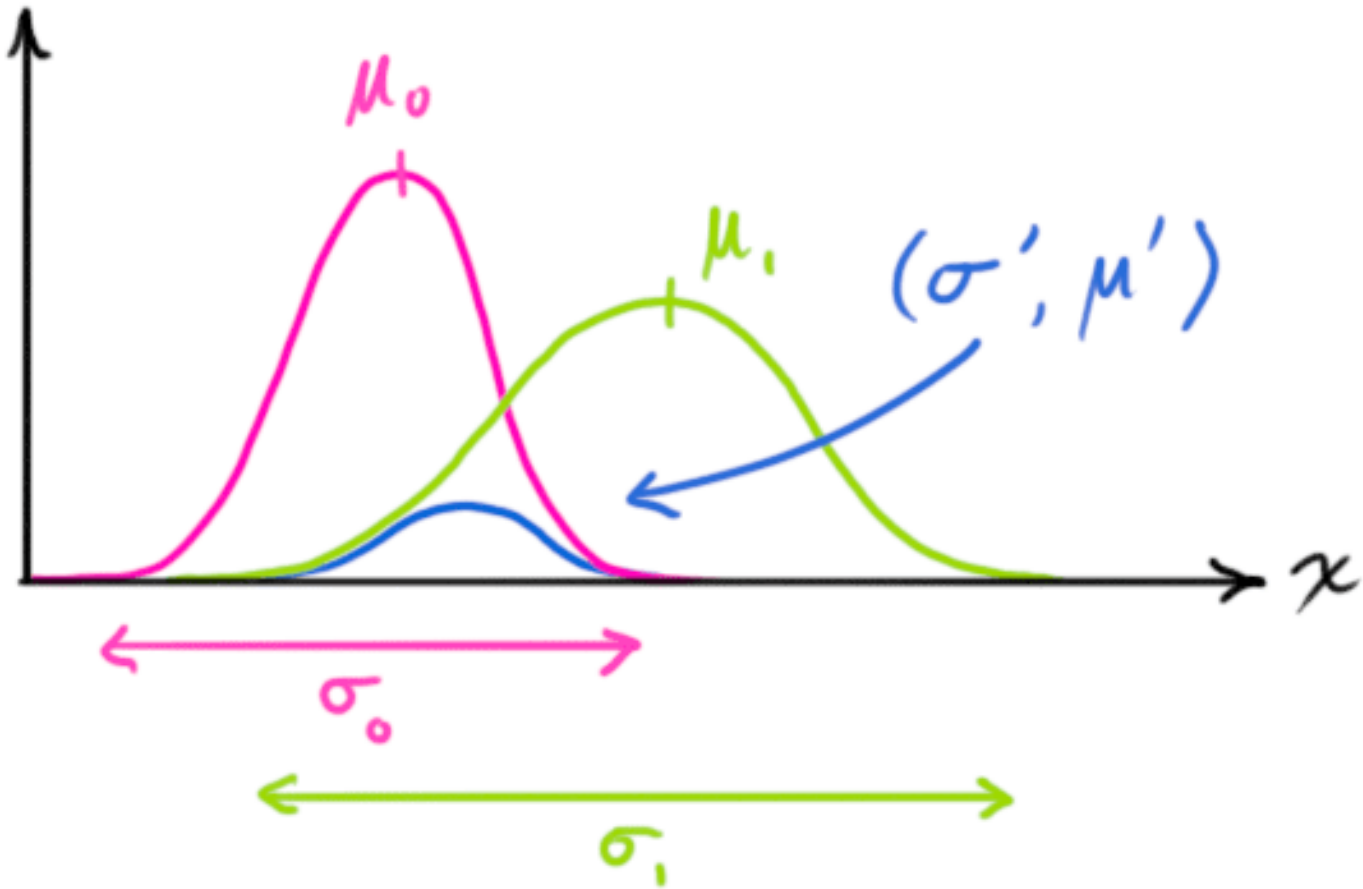


Image source: Modified from Vondrick *et al.*, 2015

Use of a Kalman filter tends to making smoother predictions across frames

Simple online and realtime tracking (SORT, Bewley, 2016) is a framework for multiple object tracking.

Example pedestrian tracking without a Kalman filter (Musk, 2017)

Example pedestrian tracking with a Kalman filter