# Multiple Cars Tracking Basing on Kalman Filter

ECE 251B –Digital Signal Processing Ⅱ

**Team #13**
**Pengluo Wang; Kaiyu Zhang; Shihao Luo**

UC San Diego
Electrical and Computer Engineering

# Contents

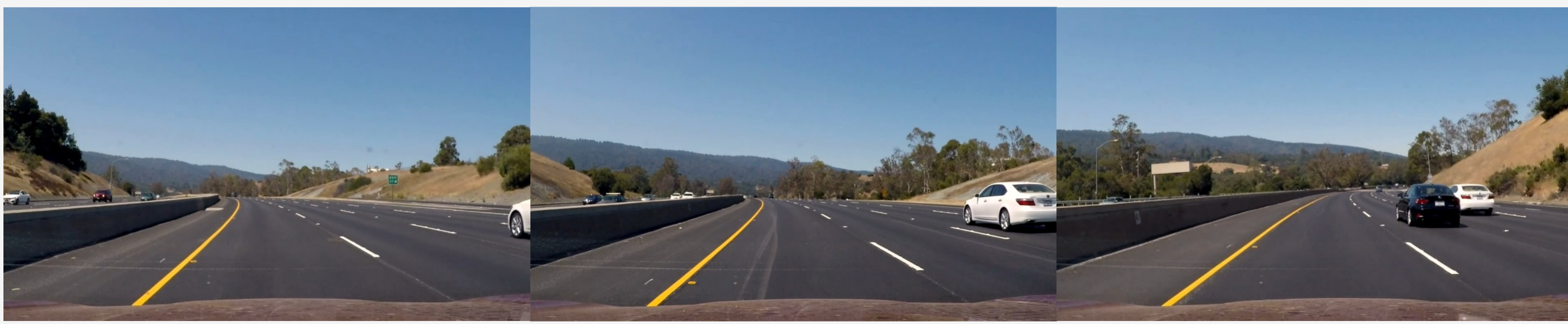
# Problem

- Background
  - The automatic driving is one of the most important technical achievement, which still needs more promotion.
  - Implement a set of algorithm to combine car detection and tracking together, to provide convenience for car driving.

- What we did
  - Use different methods implement the car detecting and tracking, and add boxes marking the multiple cars in the video.
  - We choose Tensorflow Object Detection API for car detection.
  - After getting the location information of detected cars, we make prediction using Kalman filter, and combine the prediction & measurement values as the real tracking value, to finish whole algorithm.

# Dataset

- Video data
  - The data we use is a set of video taken by the dashcam in front of our car, which is real and reliable.
  - The data was taken in our daily life, including different situations: no cars, one car, more than one cars in the visual area.
  - The cars in the video have several typical behaviors including intersection and overtaking.



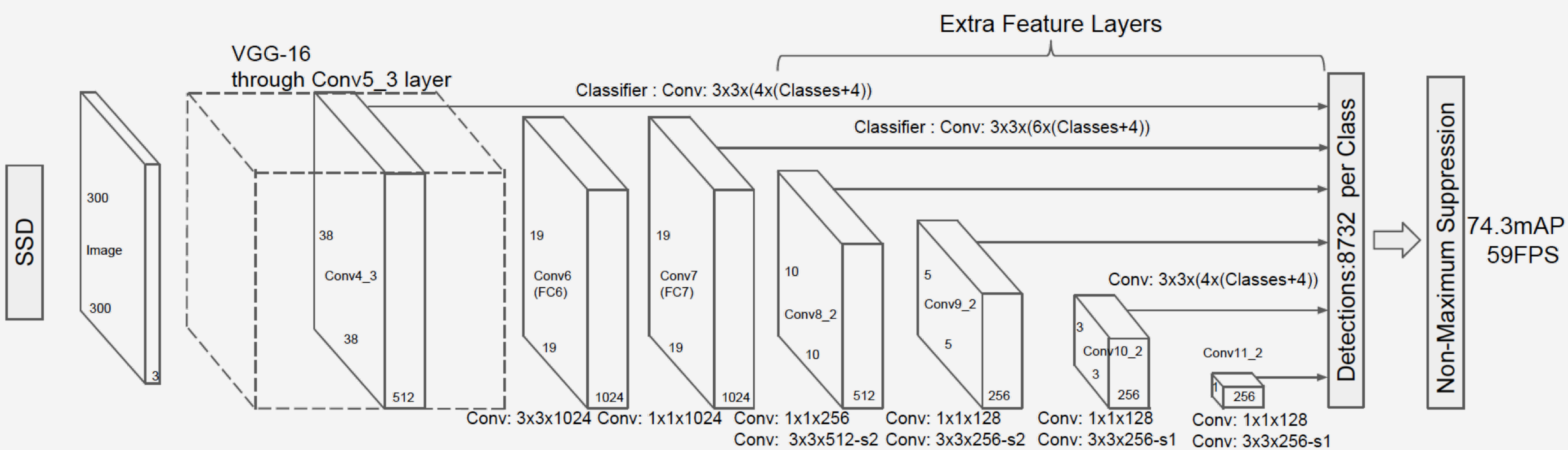
(a) (b) (c)

# Method Introduction – Detection

- Tensorflow object detection API
  - An open source framework built on top of TensorFlow that makes it easy to construct, train and deploy object detection models.
  - To use this build-in model, we applies SSD with MobileNet, a Single-Shot-Detection with MobileNet architecture trained on COCO dataset.
- Single-Shot-Detection (SSD)
  - SSD algorithm is an object detection algorithm that directly predicts the coordinates and categories of the objection with bounding box.
  - SSD is much faster than those traditional detection models without losing the accuracy, by adding several separate prediction filters.
- SSD advantages
  - High Speed
  - High Accuracy
  - Can be applied with low resolution and video data

# Method Introduction – Detection

- SSD – basic frames
  - Using a small convolutional filter to predict object categories and offsets in bounding box locations.
  - Using separate predictors (filters) for different aspect ratio detections.
  - Applying these filters to multiple feature maps from the later stages of a network in order to perform detection at multiple scales.
  - Train the data with ground truth boxes, and the obtained model can be used for detection.



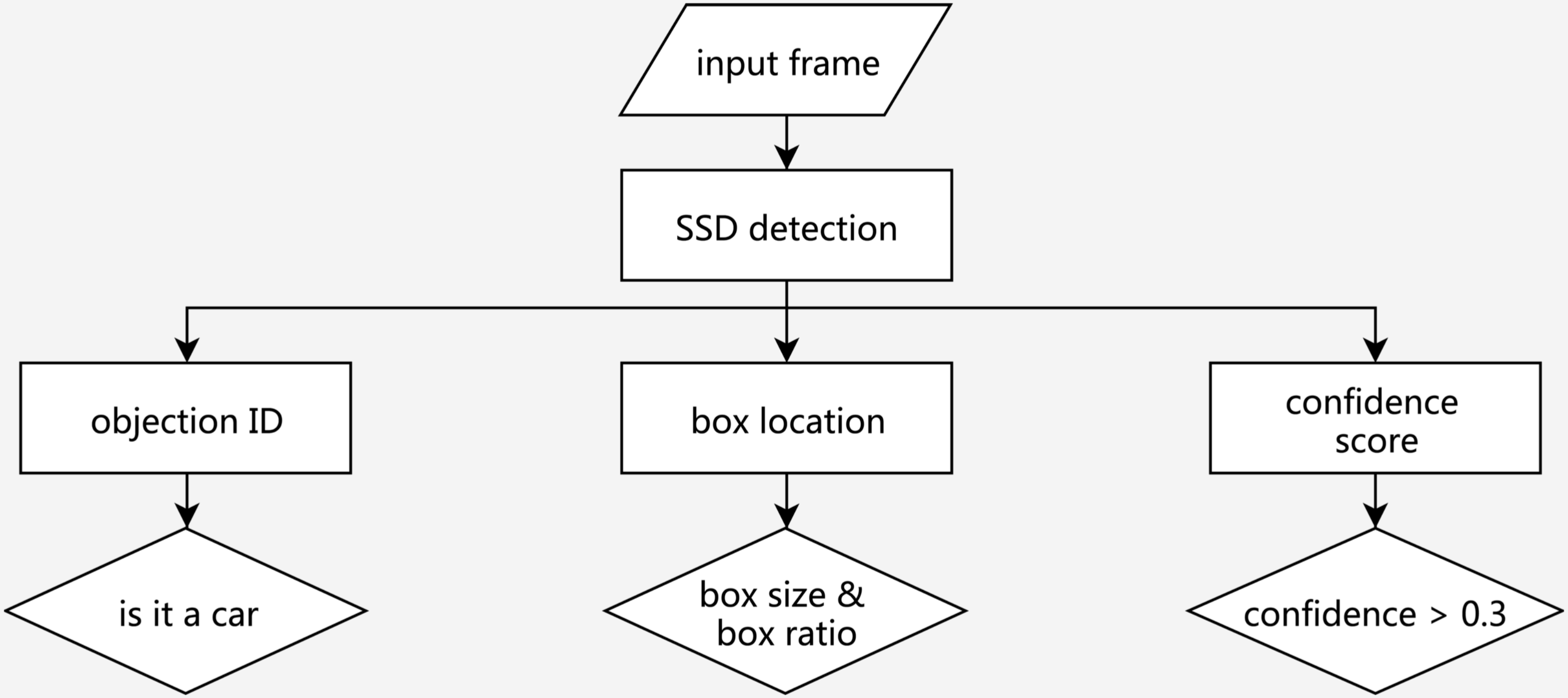
# Method Introduction – Detection

- SSD – outcome
  - This model has been trained with COCO dataset, which can detect the following objects related to transportation:

| Label | Objection |
|---|---|
| 1 | Person |
| 2 | Bicycle |
| 3 | Car |
| 4 | Motorcycle |
| 5 | Bus |
| 6 | Train |
| 7 | Traffic Light |
| 8 | Stop Sign |

# Method Introduction – Detection

- Flow chart



# Method Introduction – Detection

- SSD – outcome
  - Using a test image, we have the following outcome:

Detection result

# Method Introduction – Tracking

- Kalman filter
  - An adaptive filter, with following important benefit features:
    - Predict of object's future location;
    - Correction of the prediction based on new measurements;
    - Reduction of noise introduced by inaccurate detections;
    - Facilitating the process of association of multiple objects to their tracks.
  - Update process
    - Prediction: uses previous states to predict the current state.
    - Correction: uses the current measurement, such as detection bounding box location, to correct the state.

# Method Introduction – Tracking

- Prediction
  - Notations
    - **x**: process state
    - **P**: estimate error(noise) covariance
    - **A**: state transition matrix
    - **Q**: process noise covariance
    - **w**: process noise, $w \sim N(0, Q)$
  - Equations
    - $x_k = Ax_{k-1} + w$
    - $P_k = AP_{k-1}A^T + Q$
    - $z_k = Hx_k + v_k$
- Correction
  - Notations
    - **H**: measurement function (matrix)
    - **z**: measurement state
    - **R**: measurement noise covariance
    - **K**: Kalman gain
    - **v**: measurement noise, $v \sim N(0, R)$
  - Equations
    - $K_k = P_k H^T (HP_k H^T + R)^{-1}$
    - $P_k = (I - KH)P_{k-1}$
    - $x_k = x_{k-1} + K(z_{k-1} - Hx_{k-1})$

# Method Introduction – Tracking

- Kalman filter implementation – initialization
  - Suppose current state $x_t = [p_t \quad v_t \quad a_t]^T$

$$p_t = p_{t-1} + v_{t-1}\Delta t + \frac{1}{2}a_t\Delta t^2$$
$$v_t = v_{t-1} + a_t\Delta t$$

  - Write into Matrix Form as update function $x_t = Ax_{t-1}$, where

$$A = \begin{bmatrix} 1 & \Delta t & \nabla t/2 \\ 0 & 1 & \Delta t \\ 0 & 0 & 1 \end{bmatrix}$$ is the state transition matrix

  - In the project, dimension of $A$ is 12×12, since there are four locations to be tracked, which is left, down, right and up, representing the four corners of the bounding box.

# Method Introduction – Tracking

- Kalman filter implementation – initialization
  - Initial the measurement transition matrix $H$, given that the detector only outputs the coordinate (not velocity and acceleration)

$$H = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}$$

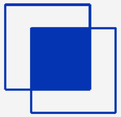
$$z_t = Hx_t + v_t$$

  - Initialize noise covariance Q, R:
    - Initial Q, R randomly.
    - The value of R shows the confidence of the confidence of prediction

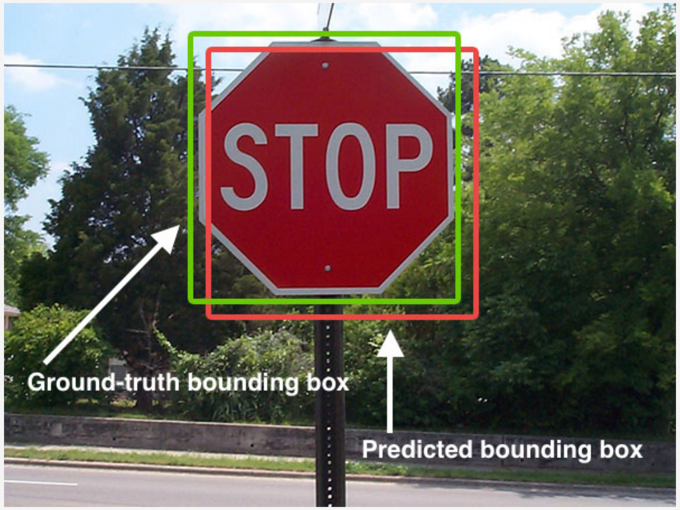
# Intersection-over-Union ( IoU )

- IoU
  - A parameter shows the accuracy of detection and prediction.
  - Can be understood as the accuracy rate for each image

$$\mathrm{IoU} = \frac{\mathrm{DetectionResults} \cap \mathrm{TrackingResults}}{\mathrm{DetectionResults} \cup \mathrm{TrackingResults}}$$



  - For example:



# Detection-to-Tracker Assignment

- Linear-Assignment
  - One of the fundamental combinatorial optimization in order to find a maximum weight matching (or minimum weight perfect matching) in a weighted bipartite graph.
  - Implemented by package *sklearn* in Python.
  - Produce the max matching between detection and tracking box in our project. The results are stored in list matched, unmatched_detections, and unmatched_trackers.

| | $D_1$ | $D_2$ | $D_3$ | $D_4$ |
|---|---|---|---|---|
| $T_1$ | 0 | 0 | 1 | 0 |
| $T_2$ | 1 | 0 | 0 | 0 |
| $T_3$ | 0 | 0 | 0 | 0 |

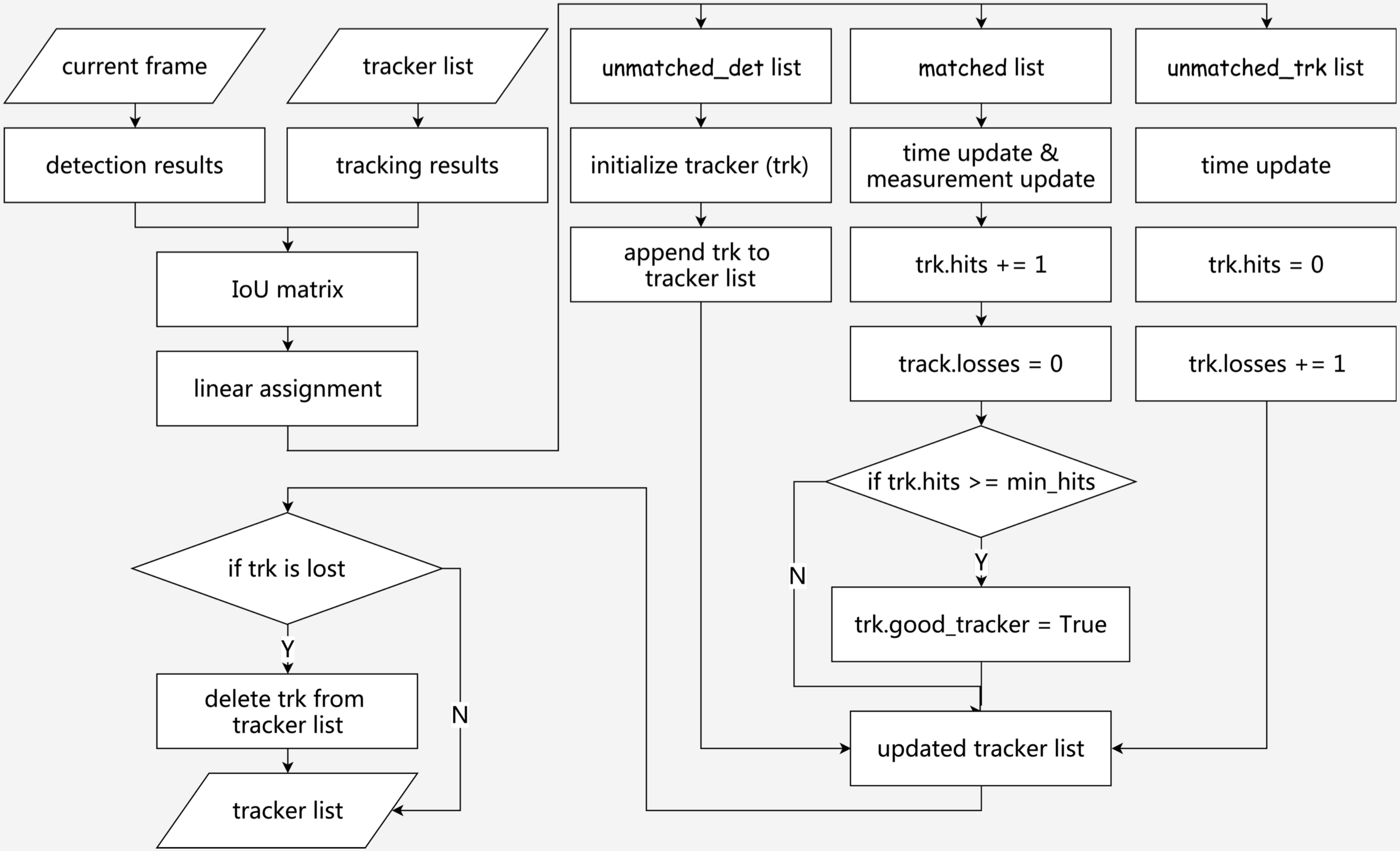
→ IoU matrix

# Pipeline

- Sequential processing
  - Process each frame of the input video, implemented by package MoviePy.
  - Deals with matched, unmatched_detections, and unmatched_trackers obtained from linear assignment of the IoU matrix for each frame.

- False alarm & missed detection
  - Introduce two important parameters, min_hits and max_losses: min_hits stands for the minimum number of consecutive matches to establish a tracker; max_loss stands for the maximum number of consecutive losses to delete a current track.
  - Introduce a variable to discriminate false alarm and real target, a tracker is called *good tracker* if it has reached min_hits.

# Pipeline Diagram



# Results – Frame

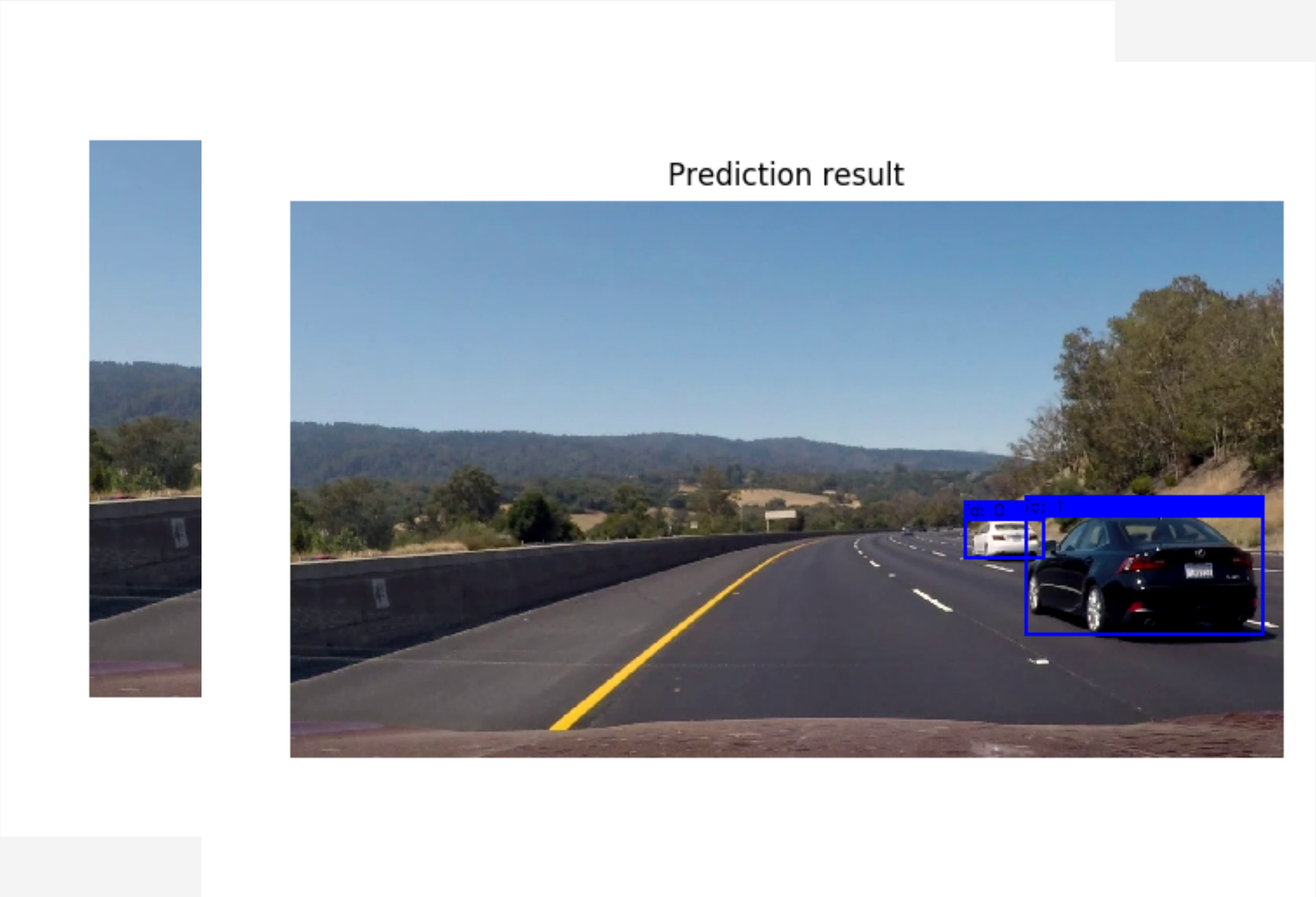
Detection result

# Results – Frame

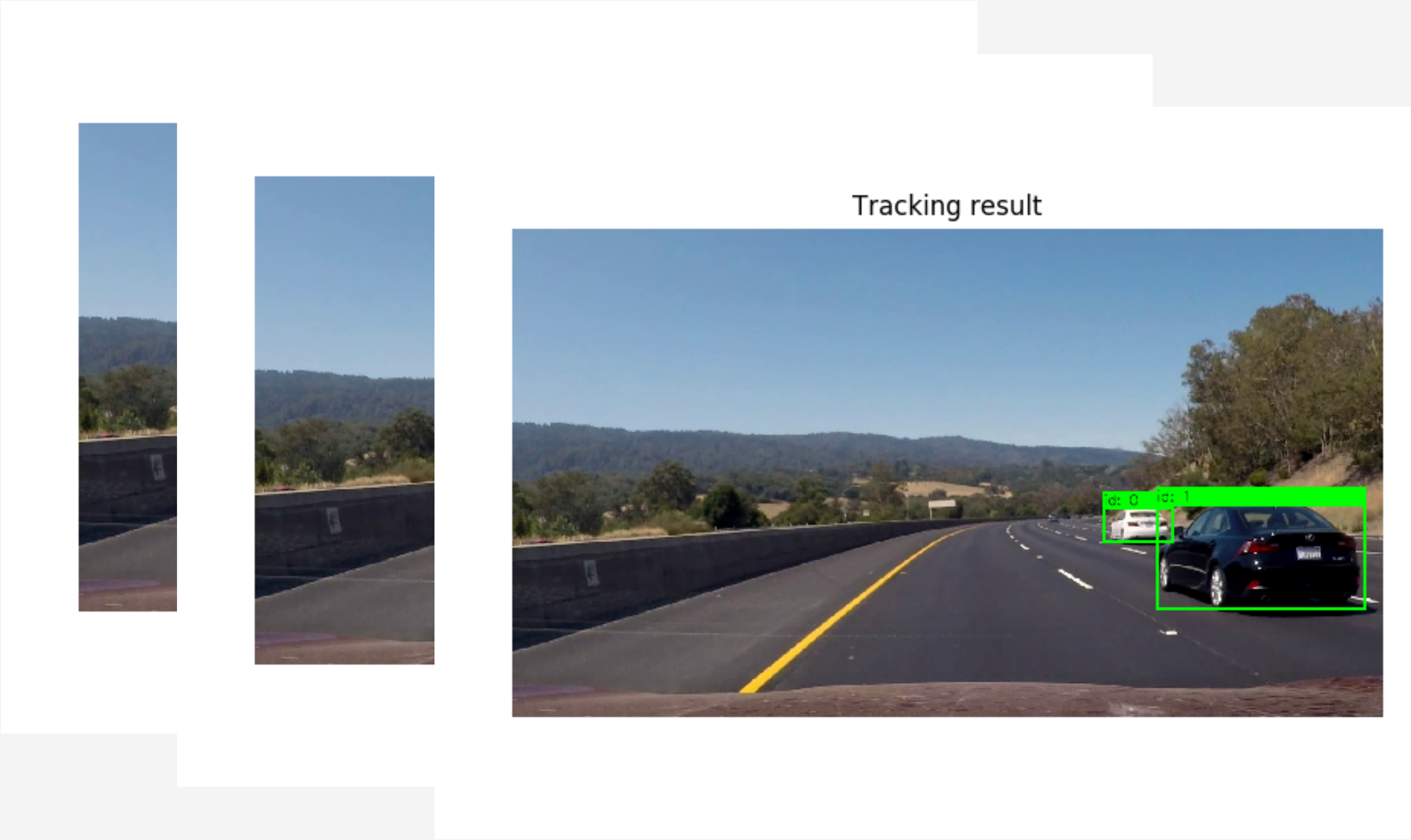
# Results – Frame

# Results – Frame

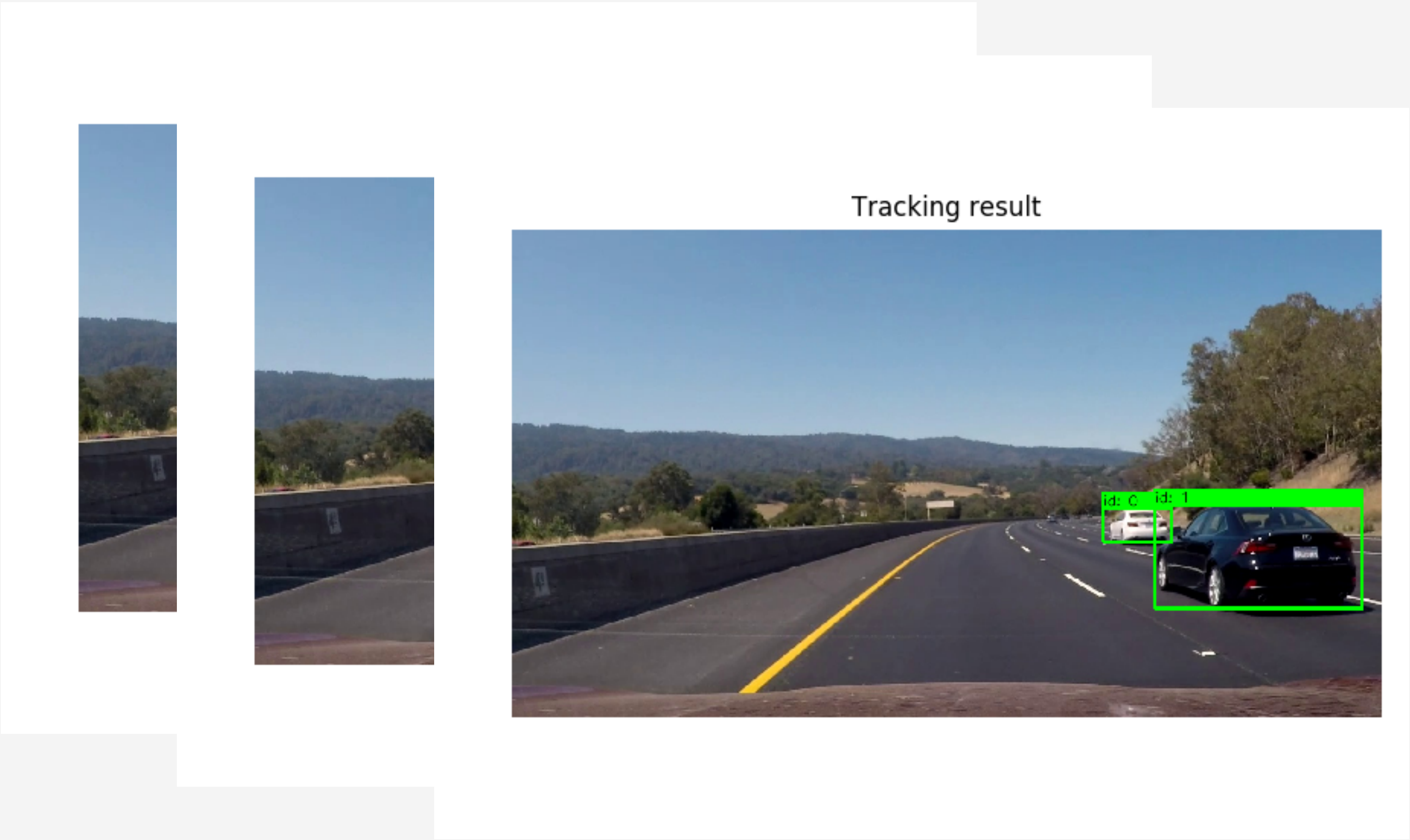
# Results – Frame



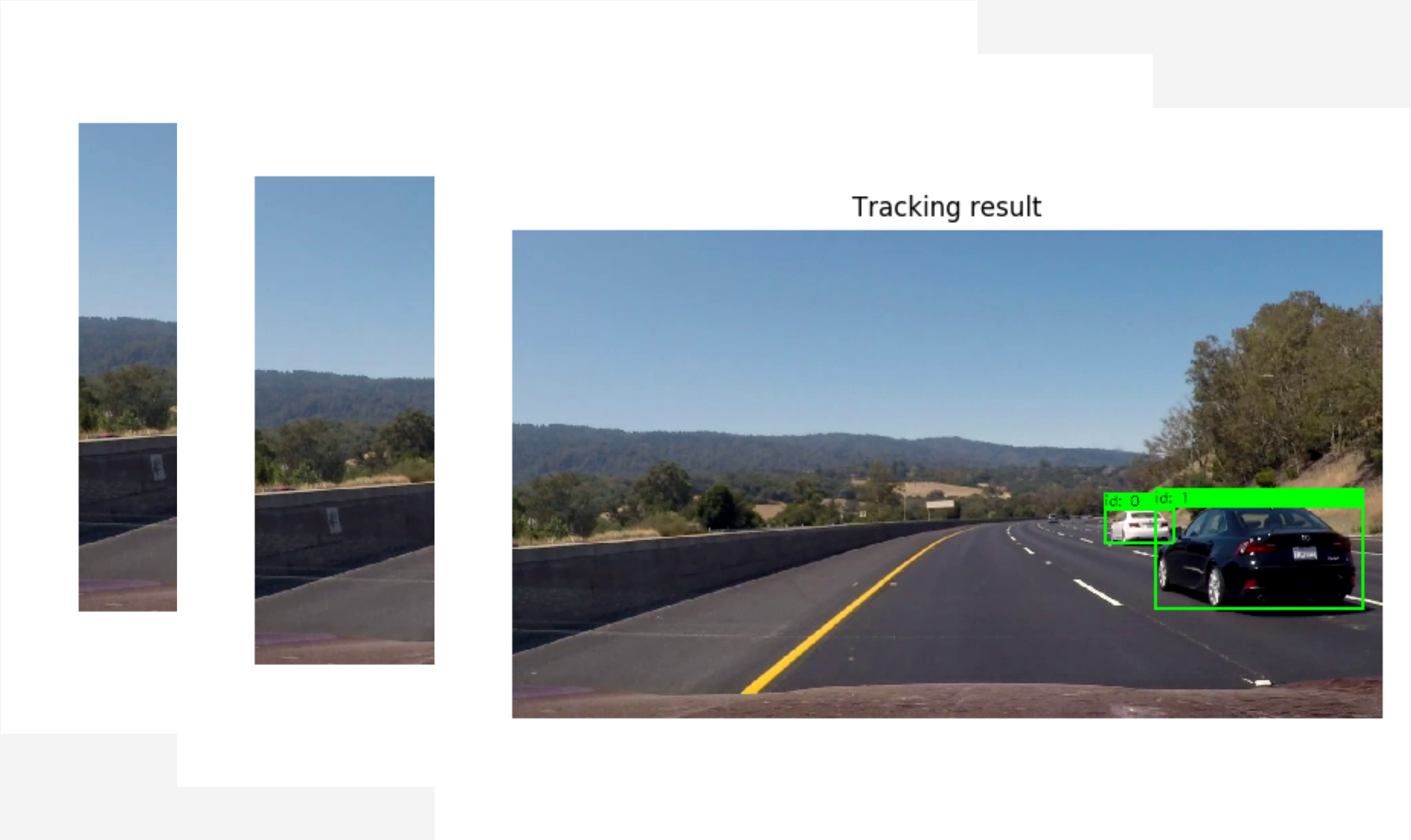
# Results – Frame



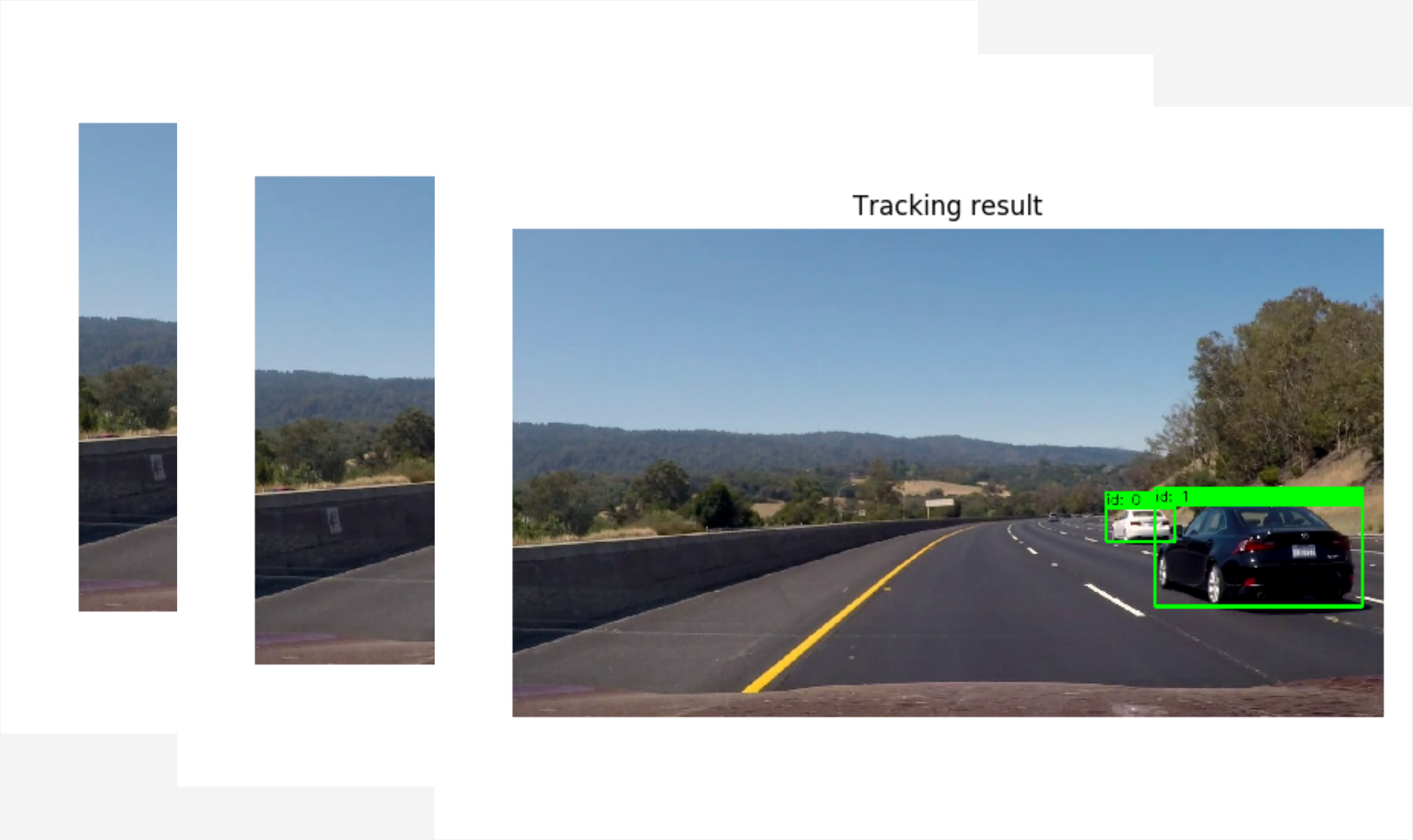
# Results – Frame



# Results – Frame



# Results – Frame



# Results – Video

# Reference


**[1]. Greg Welch, and Gary Bishop,** *An Introduction to the Kalman Filter,* [**University of North Carolina at Chapel Hill Chapel Hill], NC 27599-3175, July 24,2016**

**[2]. Wei Liu, Dragomir Anguelov, Dumitru Erhan, and Christian Szegedy,** *SSD: Single Shot MultiBox Detector,* **[cv.CS], Dec 29, 2016**

**[3]. Jonathan Huang, Vivek Rathod, Chen Sun, et al,** *Speed/accuracy trade-offs for modern convolutional object detectors,* **[cv.CS], Nov 30, 2016**

# Thank You!

# Q & A