Self-Driving Cars & Localization

Jeremy Cohen
Jun 26, 2018 · 7 min read

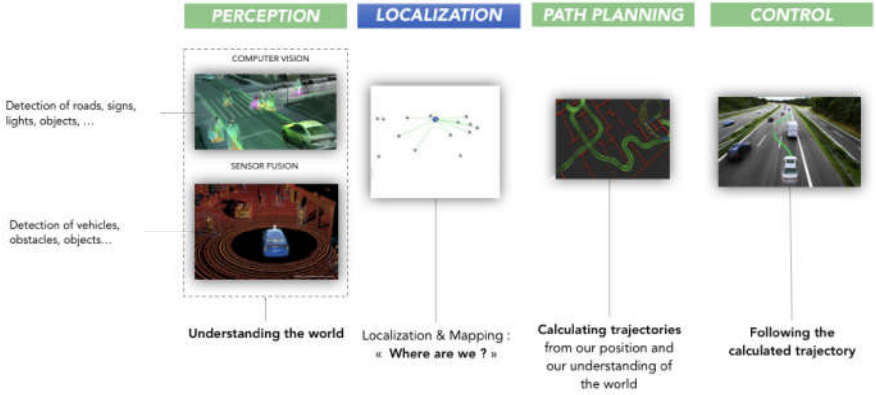
In a self-driving car car, GPS (Global Positioning Systems) use trilateration to locate our position.

source

In these measurements, there may be an error from 1 to 10 meters. This error is too important and can potentially be fatal for the passengers or the environment of the autonomous vehicle. We therefore include a step called **localization**.

> *Localization is the implementation of algorithms to estimate where is our vehicle with an error of less than 10 cm.*

> *This article follows articles AI … And the vehicle went autonomous and Sensor Fusion.*



> ***Localization*** *is a step implemented in the majority of robots and vehicles to locate with a really small margin of error. If we want to make decisions like overtaking a vehicle or simply defining a route, we need to know what's around us (sensor fusion) and where we are (localization). Only with this information we can define a trajectory.*

• • •

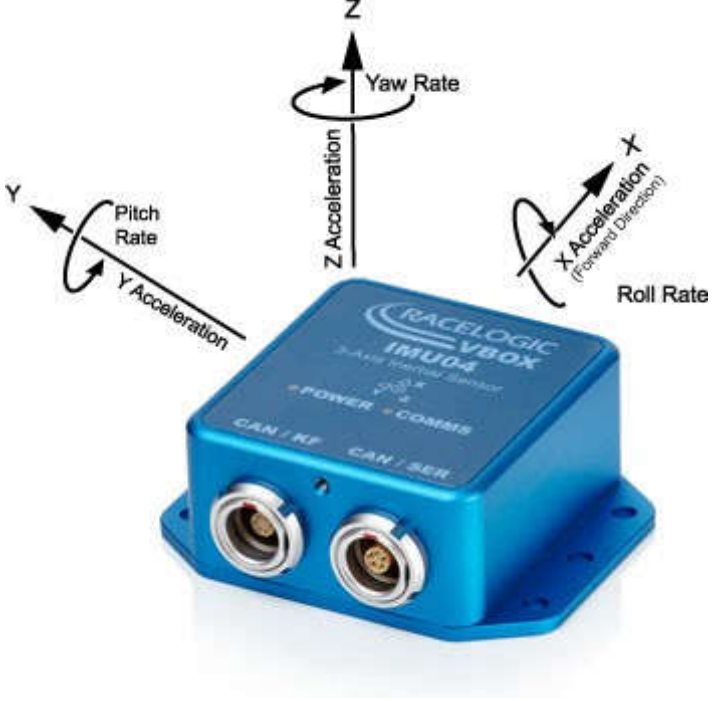
How to locate precisely?

There are many different techniques to help an autonomous vehicle locate itself.

- **Odometry**—This first technique, odometry, uses **a starting position** and **a wheel displacement calculation** to estimate a position at a time t. This technique is generally very inaccurate and leads to an accumulation of errors due to measurement inaccuracies, wheel slip, ...
- **Kalman filter**—The previous article evoked this technique to estimate the state of the vehicles around us. We can also implement this to **define the state of our own vehicle**.
- **Particle Filter—**The Bayesian filters can also have a variant called particle filters. This technique compares the observations of our sensors with the environmental map. **We then create particles around areas where the observations are similar to the map.**
- **SLAM**—A very popular technique if we also want to estimate the map exists. It is called SLAM (Simultaneous Localization And Mapping). In this technique, we estimate **our position** but **also the position of landmarks**. A traffic light can be a landmark.

Sensors

- **Inertial Measurement Unit (IMU)** is a sensor capable of defining the **movement of the vehicle** along the yaw, pitch, roll axis. This sensor calculates **acceleration** along the X, Y, Z axes, **orientation**, **inclination**, and **altitude**.



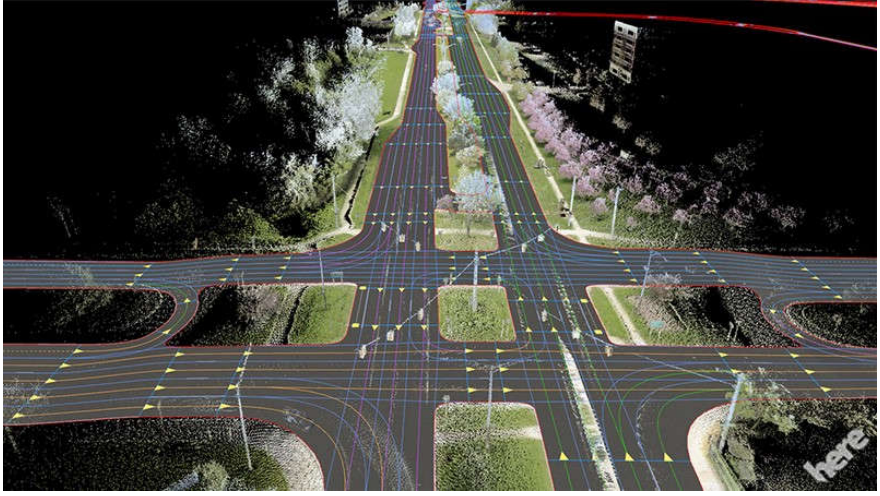
Inertial Measurement Unit (source)

- **Global Positioning System (GPS)** or NAVSTAR are the US system for positioning. In Europe, we talk about Galileo; in Russia, GLONASS. The term **Global Navigation Satellite System (GNSS)** is a very common satellite positioning system today that can use many of these subsystems to increase accuracy.

Vocabulary

We will introduce several words in this article :

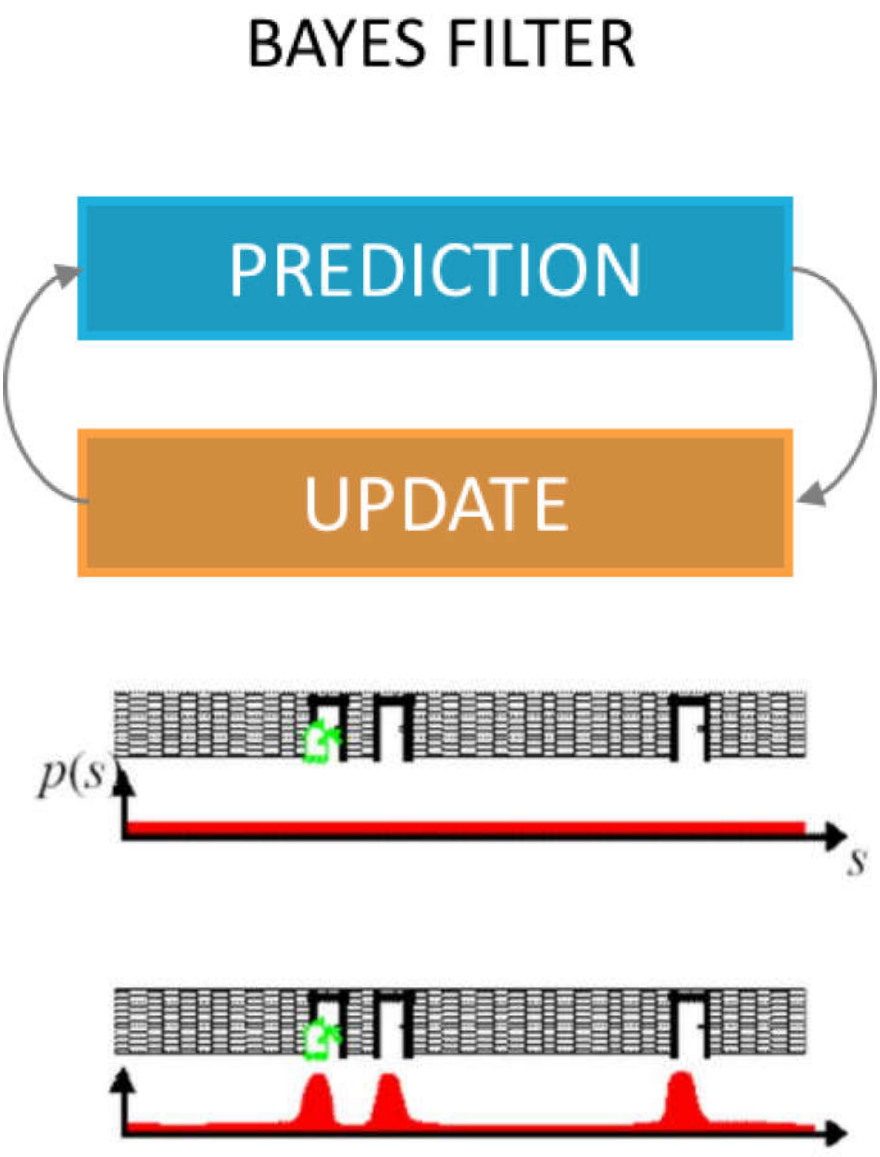
- **Observation**—An observation can be a measurement, an image, an angle …
- **Control**—This is our movements including our speeds and yaw, pitch, roll values retrieved by the IMU.
- **The position of the vehicle**—This vector includes the (x, y) coordinates and the orientation θ.
- **The map**—This is our landmarks, roads … There are several types of maps; companies like Here Technologies produce **HD Maps**, accurate maps centimeter by centimeter. These cards are produced according to the environment where the autonomous car will be able to drive.



. . .

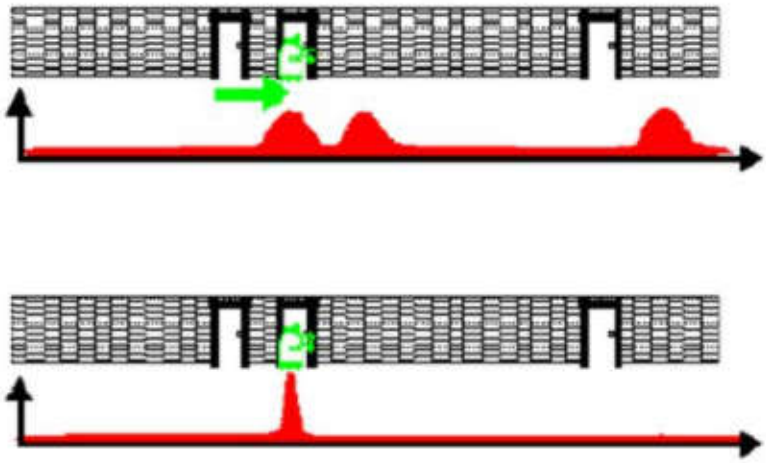
Kalman Filters

Explained in the previous article, a Kalman filter can estimate the state of a vehicle. As a reminder, this is the implementation of the Bayes Filter, with a prediction phase and an update phase.



Initialization and 1st measurement (source)

Our first estimate is an **equal distribution across the area**.
We then have a **measurement** telling us that we are **located next to a door**.
Our distribution then changes to **give a higher probability to areas located near doors**.



Motion and 2nd measurement (source)

We then perform a **motion**, our probabilities are shifted with greater uncertainties.

We take a new measurement, telling us that we **are next to a door again**.

The only possibility is to be located near the middle door.

> *In this example of a Kalman filter, we were able to locate ourselves using a few measurements and a comparison with the map. It is essential to know the map (including information only the 1st door has an adjacent door) to make deductions. This technique makes it possible not to use an initial position, which is preferable to the technique using odometry.*

. . .

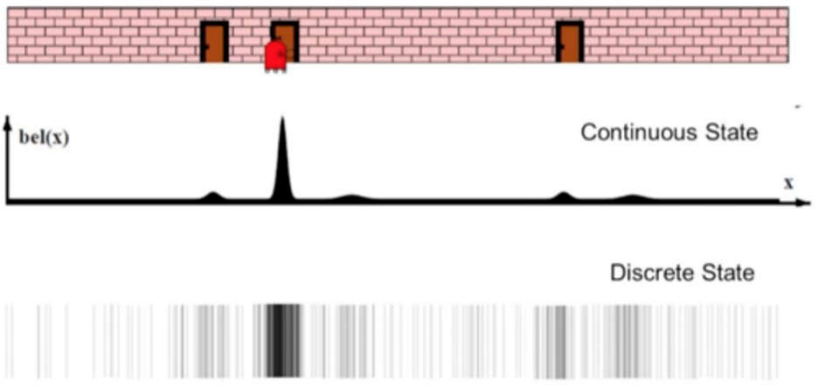
Particle Filters

A **Particle Filter** is another implementation of the **Bayes Filter**.

In a Particle Filter, we create **particles throughout the area** defined by the GPS and we **assign a weight to each particle**.

> *The weight of a particle represents the probability that our vehicle is at the location of the particle.*

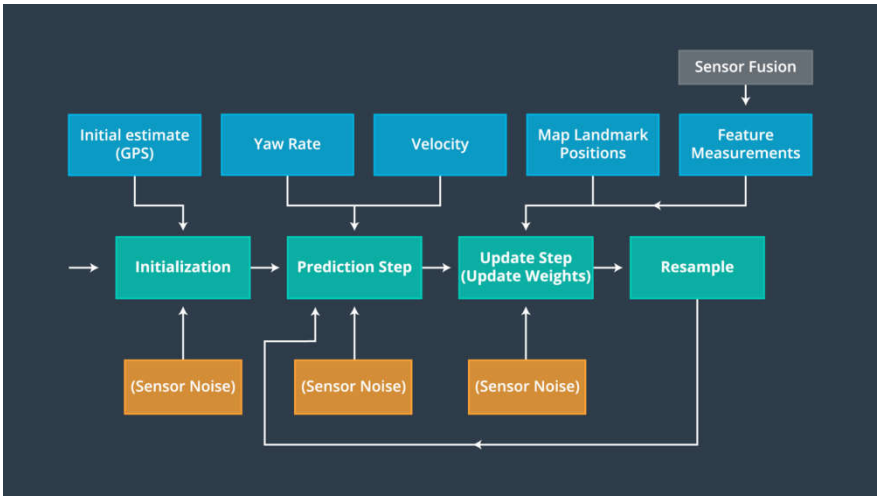
Unlike the Kalman filter, we have our probabilities are not continuous values but discrete values, we talk about weights.



Kalman Filter vs Particle Filter (source)

Algorithm

The implementation of the algorithm is according to the following scheme. We distinguish **four stages** (Initialization, Prediction, Update, Sampling) realized with the **help of several data** (GPS, IMU, speeds, measurements of the landmarks).



Localization algorithm

- **Initialization**—We use an **initial estimate** from the GPS and add noise (due to sensor inaccuracy) to **initialize a chosen number of particles. Each particle has a position (x, y) and an orientation θ.**
  This gives us a particle distribution throughout the GPS area with equal weights.

- **Prediction**—Once our particles are initialized, we make a first prediction taking into account our speed and our rotations. In

every prediction, our movements will be taken into account. We use equations describing x, y, θ (orientation) to describe the motion of a vehicle.

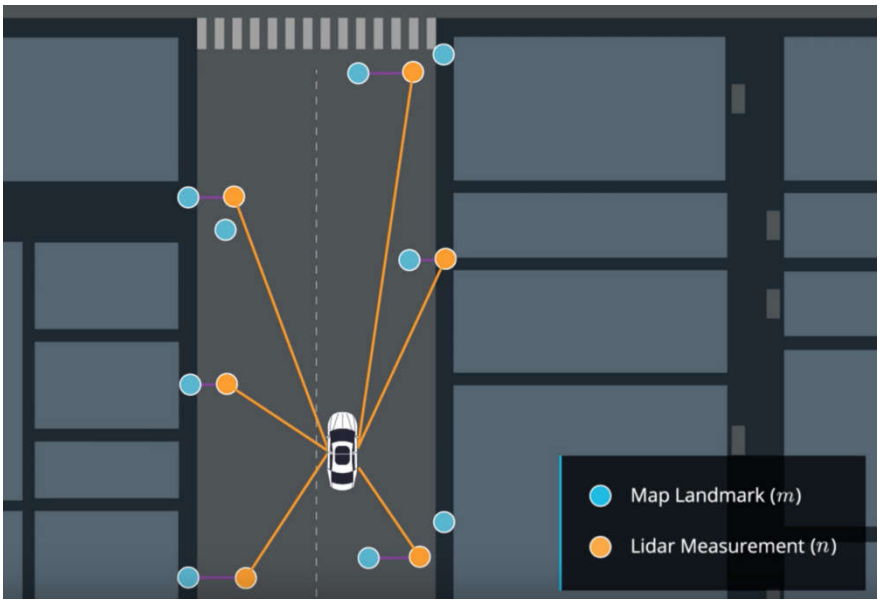
$$x_f = x_0 + \frac{v}{\dot{\theta}}[sin(\theta_0 + \dot{\theta}(dt)) - sin(\theta_0)]$$

$$y_f = y_0 + \frac{v}{\dot{\theta}}[cos(\theta_0) - cos(\theta_0 + \dot{\theta}(dt))]$$

$$\theta_f = \theta_0 + \dot{\theta}(dt)$$

Motion equations

- **Update**—In our update phase, we first realize a match between our **measurements n** and the **map m**.



Match between measurements and map

We use **sensor fusion data** to determine **surrounding objects** and then **update our weights** with the following equation :

$$P(x,y) = \frac{1}{2\pi\sigma\_x\sigma\_y} e^{-(\frac{(x-\mu_x)^2}{2\sigma_x^2} + \frac{(y-\mu_y)^2}{2\sigma_y^2})}$$

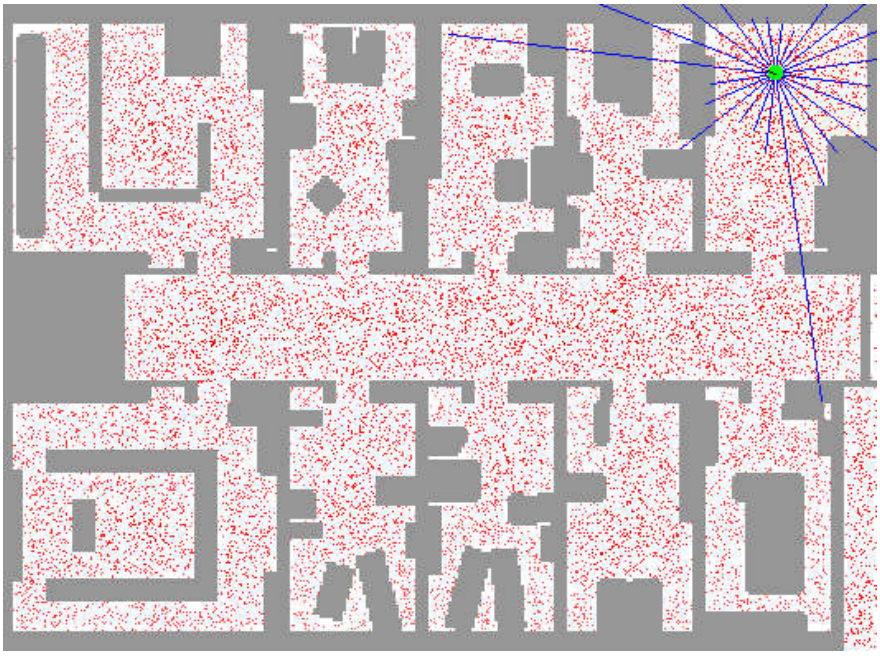
Update

In this equation, for each particle:
- σx and σy are our **uncertainties**
- x and y are the **observations** of the landmarks
- μx and μy are the **ground truth coordinates** of the landmarks coming from the map.

In the case where the error is strong, the exponential term is 0, the weight of our particle is 0 as well. In the case where it is very low, the weight of the particle is 1 standardized by the term $2\pi.\sigma x.\sigma y$.

- **Resampling**—Finally, we have one last stage where **we select the particles with the highest weights** and destroy the least likely ones.
  The higher the weight, the more likely the particle is to survive.

The cycle is then repeated with the most probable particles, we take into account our displacements since the last computation and realize a prediction then a correction according to our observations.



Particle filter in action

> *Particle filters are effective and can locate a vehicle very precisely. For each particle, we compare the measurements made by the particle with the measurements made by the vehicle and calculate a probability or weight. This calculation can makes the filter slow if we have a lot of particles. It also requires having the map of the environment where we drive permanently.*

Results

> *My projects with Udacity taught me how to implement a Particle Filter in C ++. As in the algorithm described earlier, we implement localization by defining 100 particles and assigning a*

weight to each particle through measurements made by our sensors.

In the following video, we can see :

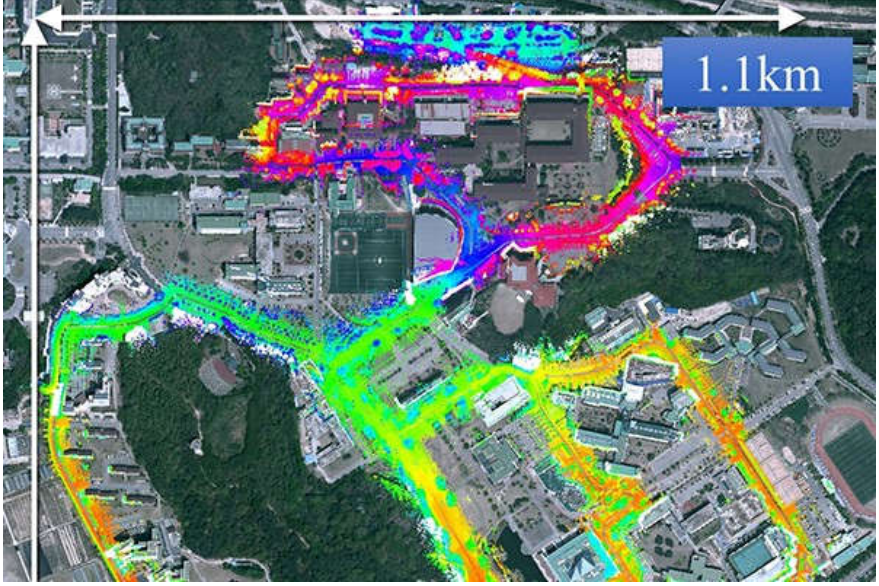
- A **green laser** representing the **measurements from the vehicle**.
- A **blue laser** representing the **measurements from the nearest particle** (blue circle).
- A **particle** locating the **vehicle** (blue circle).
- The **black circles** are our **landmarks** (traffic lights, signs, bushes, …) coming from the map.

Particle Filter Implementation

SLAM (Simultaneous Localization And Mapping)

Another very popular method is called **SLAM**, this technique makes it possible to **estimate the map** (the coordinates of the landmarks) in addition to estimating the coordinates of our vehicle.

To work, we can with the Lidar find walls, sidewalks and thus build a map. SLAM's algorithms need to know how to recognize landmarks, then position them and add elements to the map.

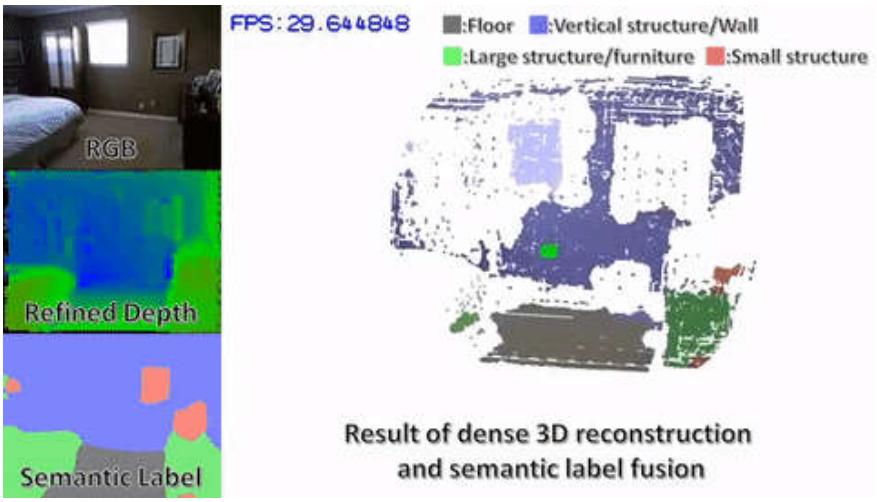


SLAM created map (source)

Conclusion

Localization is an essential topic for any robot or autonomous vehicle. If we can locate our vehicle very precisely, we can drive independently. This subject is constantly evolving, the sensors are becoming more and more accurate and the algorithms are more and more efficient.

SLAM techniques are very popular for outdoor and indoor navigation where GPS are not very effective. Cartography also has a very important role because without a map, we cannot know where we are. Today, research is exploring localization using deep learning algorithms and cameras.



source

> *Now that we are localized and know our environment, we can discuss algorithms for creating trajectories and making decisions!*

Jeremy Cohen.

→ Connect on LinkedIn

Discover this article in French

Next on Autonomous Navigation, Controllers and Final Integration

. . .

References

- Udacity Self-driving Car Nanodegree Program

GitHub Project for Localization

- Particle Filter Implementation : particle-filters