Hochschule Ravensburg-Weingarten

Lidar and Radar Systems
Project Report 3

Sensor Fusion: Estimate Object Velocity and Predict New Position

Created by:

Sahil Arora
(Matriculation Number: 32953)

February 12, 2021

Contents

1 Introduction

When we think of an autonomous application such as a self-driving car, a wide range of sensors come to mind that can be employed to do the job. On one side, you may have seen prototypes of Autonomous Taxis by Waymo with a large black colored protruding LiDAR sensor on top of it [Way]. On other hand, RADAR is now also being used heavily in automobile industry to serve as an assistance device to driver.

We learnt that Machine Learning algorithms using Neural Networks can be used on the given dataset to teach the car to detect objects (in our case, other automobiles on the road) [BR20]. One such approach is using a Bird's Eye View (BEV) 2D transformation and predict the Bounding Boxes using learnt networks. However, as some incidents show, the current developments are not enough to regard autonomous cars on the same level as human drivers. Following a fatal crash involving an autonomous test vehicle by Uber [Boa], it can be stated with huge safety weightage that an autonomous vehicle must be able to flawlessly predict objects (such as cars, bicycles, pedestrians) around it, must be able to precisely predict the velocity profile and future positions of the object to make a good judgement and take an action accordingly.

In this project report, we try to describe the steps used to estimate the velocity of an object in Section 2. We consider that ego vehicle is mounted with a camera, a LiDAR and a RADAR sensor. We use the RADAR observations to predict the radial velocity of object. Section 3, describes how we can use fused information from different sensors to predict the full velocity vector using Kuhn-Munkres Algorithm and make predictions of objects' new positions. Lastly in Section 4, we try to quantify the error between predicted and real positions. It doesn't make much sense to start from scratch in the area of object detection in our project, because already several high performance algorithms are available. One such algorithm is described by [Che+17] using **feature level fusion** on Image, Lidar FOV(Field of View) and BEV transformation where a Convolutional Neural Networks outputs 3D bounding boxes (results as shown in Fig 1.1).

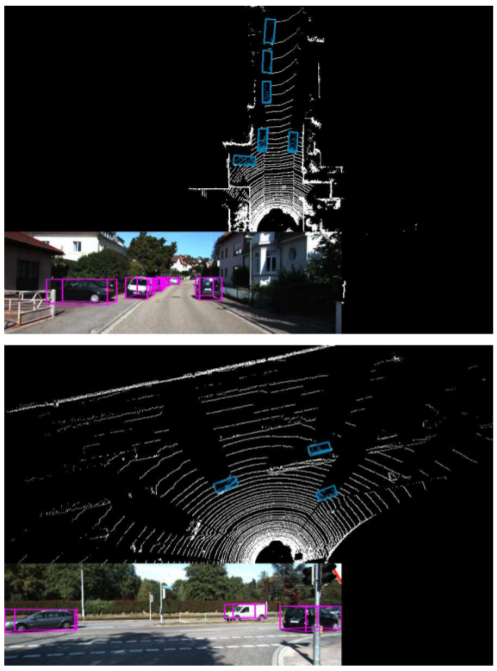


Fig 1.1: 3D Bounding Box prediction using Camera and Lidar data fusion by [Che+17].

2 Radial Velocity by RADAR

RADAR sensor works by emitting Electromagnetic Waves and sensing the reflected ones to compute the time of flight and properties like distance, velocity of the obstacle it hit [Wikc]. This means that using the transmitter and receiver combination, we can measure the emitted frequency (f_e) and the received (reflected) frequency (f_d). As we are sending and receiving electromagnetic waves, we can use the **Doppler Effect** [Wika], as described in Sec 2.1.

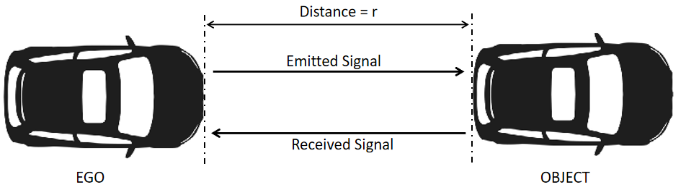
2.1 Velocity of single Echo using Doppler Effect

Let us assume that the distance between the ego vehicle (transmitter) and the object (reflector) is r, as shown in Fig 2.1A. As the signal goes back and forth, the total distance travelled by signal is $2r$. Since, we are dealing with Electromagnetic Waves, we can represent them as complex numbers in phasor forms [ER21a], as shown in Fig 2.1B. Here, the angular velocity of phasor rotation is represented as ω and the initial phase angle as ϕ_0. From elementary physics, we know that:

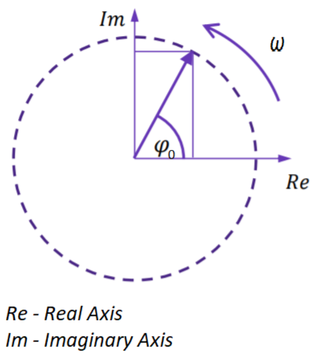
$$\omega = 2\pi f \tag{1}$$

where f is the frequency of wave. The phase angle (ϕ_t) can be written as a function of time as:

$$\phi_t = \omega t + \phi_0 \tag{2}$$



(a) Fig 2.1A Ego and Object



(b) Fig 2.1B Phasor Diagram [ER21a]

Taking the first derivative of equation (2) and combining with eq (1), we can write:

$$\frac{d\phi_t}{dt} = \omega = 2\pi f \tag{3}$$

Simplifying equation (3) we can write it as:

$$f = \frac{1}{2\pi}\frac{d\phi_t}{dt} \tag{4}$$

Further, let us say that the wavelength of the wave is denoted by λ. Based on total distance travelled ($2r$), we can write the number of length segments covered by the wave as:

$$N = \frac{2r}{\lambda} \tag{5}$$

Here each length segment equals to a full phase revolution of 2π radian angle. Following that, the total angle covered in N length segments is equal to :

$$\phi = 2\pi N = \frac{4\pi r}{\lambda} \tag{6}$$

Substituting the value of ϕ from equation (6) into equation (4) gives us :

$$f = \frac{1}{2\pi}\frac{d(\frac{4\pi r}{\lambda})}{dt} = \frac{2}{\lambda}\frac{dr}{dt} \tag{7}$$

where dr/dt can be written as velocity v_r. The equation (7) can be re-written as:

$$v_r = \frac{\lambda f_d}{2} \tag{8}$$

Using this formula, care must be taken that frequency is for the wave received by the RADAR receiver (denoted as f_d, called as Doppler Frequency). The velocity obtained is only the radial velocity, denoted as v_r. With the computed velocity v_r, we can only detect a part of the object's trajectory. This can be problematic in some configurations as shown in Fig 2.1C, where the tangential movement is not detected.

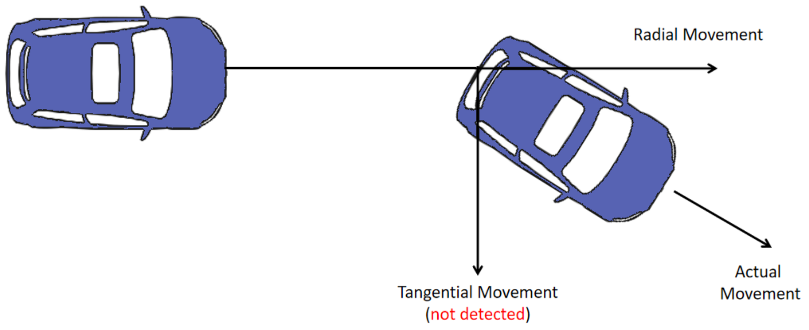


Fig 2.1C: Undetected tangential movement of Object.

The equation can be further simplified into terms of f_e and f_d, if wavelength is substituted as $\lambda = c/f_e$, c being speed of light and f_e is emitted wave's frequency:

$$v_r = \frac{cf_d}{2f_e} \tag{9}$$

2.2 Velocity of Full Object

RADAR reflections can vary heavily from measurement to measurement. Using just one point-observation to estimate the radial velocity could give result which are far from the actual value. To precisely compute the velocity of the whole object, we need to use all the RADAR observations that are available in the bounding box of that object. The velocity of the whole object could be computed as the average value of all the above RADAR observations. At the same time, we have to look for potential outliers and they need to be excluded before the averaging process.

Given a working object classifier that outputs 3D bounding boxes for the detected objects, we can use Field of View transformation to get a list of RADAR measurements that correspond to that object. Let us say, we get a list (denoted as V_{list}) of atleast 5 points corresponding to one object.

$$V_{list} = \{v_r^1, v_r^2, v_r^3, ...v_r^k\} \ ; \ k \geq 5$$

Some values of radial velocities from this list may be outliers, for eg: We get not only the reading from any given large vehicle, but due to large under-body clearance, we may also get readings from some static targets around the vehicle. Previous work done in [Mei+17] (as in Fig 2.1D) show outlying RADAR readings (marked as b) for object B that are not part of the object. The upper part of Fig 2.1D shows the camera image and lower part shows BEV of RADAR data. We will use the concept of Interquartile Range (IQR) to classify the points as outliers.

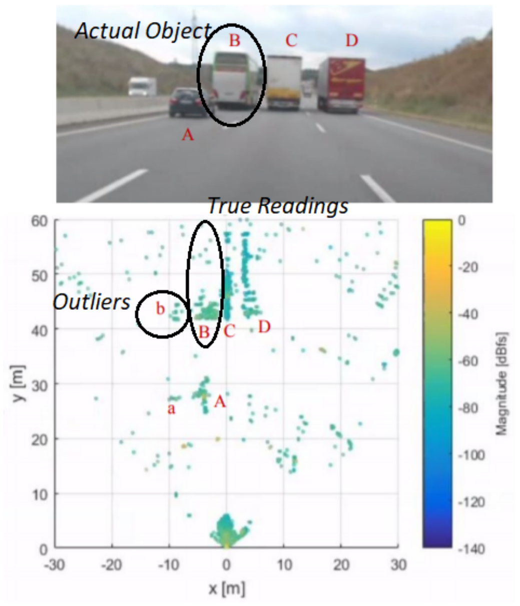


Fig 2.1D: Outlying RADAR points for large vehicle. [Mei+17]

Interquartile Range (IQR): IQR is the measure of statistical dispersion, defined as the difference of Q3 (third Quartile) and Q1 (first Quartile) [Wikb]. Quartiles are medians of parts of the ranked-data list. In order to find medians, we first arrange the given data into a ascending order. Given an *even* $2n$ or *odd* $2n + 1$ number of values (in our case $\geq$5), Quateriles are defined as following:

- Q1 = Median of n smallest numbers
- Q3 = Median of n largest numbers

$$IQR = Q3 - Q1$$

Outliers: Any point which does not fall in the range of $R' = [Q1 - 1.5 * IQR,\ Q3 + 1.5 * IQR]$ is classified as a potential oulier. Such point must be removed from the list as they would mess up the calculation. After removing the outliers, average is taken for the remaining points to give the velocity of the full object.

$$v_r(avg) = \frac{1}{k'} \sum_{m=1}^{k'} v_r^m \ \forall \ v_r^m \in R'$$

where k' is the count of valid points.

3 Object Tracking and Position Prediction

As seen in Section 2, we used RADAR measurements to estimate the object's radial velocity. However, prediction of future positions with radial velocity may not be a good idea. To overcome this problem, we use a **tracking algorithm** that matches corresponding objects in two different recordings [ER21b]. As mentioned before, we will skip the object detection algorithm and assume that we have a fully trained model that outputs 3D bounding boxes. This means that, given our ego vehicle and its co-ordinate frame (as shown in Fig 3), we have information of the width (w), length (l) and center point $p = (x, y)$ of the detected object's bounding box. With this information using elementary trigonometry, we can compute the rotation angle (ψ) of the object. From Sec 2.2, we also have the radial velocity (v_r) of the object. We sum up all the information we have related to the detected object (d_i) as a vector below:

$$d_i = (p_i, w_i, l_i, \psi_i, v_r^i)$$

To compare two arbitrary detections d_i and d_j, we define the differences in all components of vector d as below:

- Euclidean distance of center points : $p_{ij} = d(p_i, p_j)$
- Difference (absolute) in width : $w_{ij} = |w_i - w_j|$
- Difference (absolute) in length : $l_{ij} = |l_i - l_j|$
- Difference (absolute) in angle : $\psi_{ij} = |\psi_i - \psi_j|$
- Difference (absolute) in radial velocity : $v_{ij} = |v_r^i - v_r^j|$

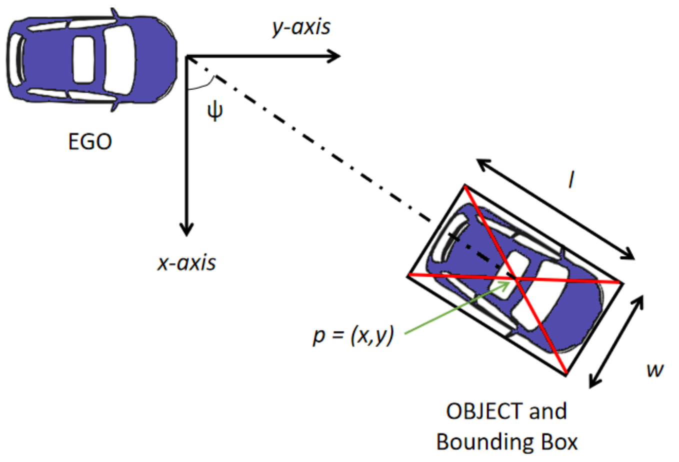


Fig 3: Bounding Box position in Ego co-ordinate system.

3.1 Tracking Algorithm : Kuhn-Munkres

We use Kuhn-Munkres algorithm to match corresponding objects in two different recordings by defining a **cost function** for the match [Mun57]. Let us say that we have two lists of detected objects, L_0 and L_1 in two different recordings, R_0 and R_1.

$$L_0 = (d_{01}, d_{02}, d_{03}, ...d_{0m}) \ ; \ L_1 = (d_{11}, d_{12}, d_{13}, ...d_{1n})$$

Normalization: We realize that the differences of two recordings can be very different values and they need to be normalized before moving ahead with calculations. To normalize, we find the maximum values of each of the above 5 computations (Euclidean distance and differences) and then individual elements are divided by the found maximum value [ER21b]. Below equations show the maximum of each computation in the recording list L_0 and L_1.

- $d_{max} = max\{d(p_{0i}, p_{1j})\ \forall\ 0 \leq i \leq m\ and\ 0 \leq j \leq n\ \}$
- $w_{max} = max\{|w_{0i} - w_{1j}|\ \forall\ 0 \leq i \leq m\ and\ 0 \leq j \leq n\ \}$
- $l_{max} = max\{|l_{0i} - l_{1j}|\ \forall\ 0 \leq i \leq m\ and\ 0 \leq j \leq n\ \}$
- $\psi_{max} = max\{|\psi_{0i} - \psi_{1j}|\ \forall\ 0 \leq i \leq m\ and\ 0 \leq j \leq n\ \}$
- $v_{max} = max\{|v_{0i} - v_{1j}|\ \forall\ 0 \leq i \leq m\ and\ 0 \leq j \leq n\ \}$

Below equations show the conversion of original computation to normalized computations. After normalization, the new values would range from 0 to 1.

$$P_{ij} = \frac{p_{ij}}{d_{max}};\ W_{ij} = \frac{w_{ij}}{w_{max}};\ L_{ij} = \frac{l_{ij}}{l_{max}};\ \Phi_{ij} = \frac{\psi_{ij}}{\psi_{max}};\ V_{ij} = \frac{v_{ij}}{v_{max}};$$

Cost Function: Now we can define the cost function between two detections, d_{0i} from recording R_0 and d_{1j} from recording R_1, as described in equation below:

$$f(d_{0i}, d_{1j}) = \lambda_p P_{ij} + \lambda_w W_{ij} + \lambda_l L_{ij} + \lambda_\Phi \Phi_{ij} + \lambda_v V_{ij}$$

where the λ values are weights, using which a selected feature can be given more importance than the rest. Now that we have defined the cost function, we can compute if detections in recording R_0 match to any of those in recording R_1. Let us assume that we detected 3 objects in each recording. We can combine all the cost functions into a **cost matrix**, which in this case will be of size 3 by 3 as shown in Table 3.1.

$f(\cdot,\cdot)$	d_{00}	d_{01}	d_{02}
d_{10}	$f(d_{00}, d_{10})$	$f(d_{01}, d_{10})$	$f(d_{02}, d_{10})$
d_{11}	$f(d_{00}, d_{11})$	$f(d_{01}, d_{11})$	$f(d_{02}, d_{11})$
d_{12}	$f(d_{00}, d_{12})$	$f(d_{01}, d_{12})$	$f(d_{02}, d_{12})$

Table 3.1: Cost Matrix

To get a better understanding, we take some example values. The Table 3.1 is transformed to Table 3.1B with calculated values (arbitrary examples). Now we subtract the minimum element of a column from all the values of that corresponding column. This means that one value per column will become 0, as seen in Table 3.1C.

We select the pairs with zero cost value in the matrix as matching objects. For our examples, we see a match in d_{11} and d_{01}. But it some cases, we may get multiples matches for the same object, just like here we have d_{12} matching to both d_{00} and d_{02}. Only one of them could be correct and for this we refer the Table 3.1B again. Here, cost of pairing d_{00} with d_{12} is lesser than that of d_{02}. On this basis, we decide d_{00} as the match for d_{12}. Since objects may disappear and new objects may appear in subsequent recordings, not all of the

$f(\cdot,\cdot)$	d_{00}	d_{01}	d_{02}
d_{10}	5	6	13
d_{11}	3	1	11
d_{12}	1	8	4

$f(\cdot,\cdot)$	d_{00}	d_{01}	d_{02}
d_{10}	4	5	9
d_{11}	2	0	7
d_{12}	0	7	0

Table : 3.1B on left, Table: 3.1C on right

objects will be matched. We see the same in our example, object d_{02} has no match implying that it gets lost in recording R_1 and object d_{10} has no match implying that it is seen for first time in recording R_1. In addition to this matching process, we could also define a threshold to keep objects like d_{02} alive for certain number of recording. This way, it can be re-tracked if it got occluded in some frames for a short time.

Using Kuhn-Munkres algorithm, we can follow objects and track the distance they travelled in X and Y direction. This also means that we can calculate the velocities V_x and V_y, thereby giving us the full velocity, as opposed to just radial velocity by RADAR in Sec 2.

3.2 Predict Object's Position in next Recording

We initially defined the detections with a vector of information of the width (w), length (l), center point $p = (x, y)$, roation angle (ψ_i) and radial velocity (v_r).

$$d_i = (p_i, w_i, l_i, \psi_i, v_r^i)$$

After the tracking and full velocity measurement by Kuhn-Munkres algorithm, this vector can be appended with new information. We can now add (v_x) and (v_y) to the list. Also, we can include the rate of change of angle ($\Delta\psi_i$). As a result, the updated data vector (X) is defined as below:

$$X_i = (x_i, y_i, w_i, l_i, \psi_i, v_x, v_y, \Delta\psi)^T$$

In case an object is seen in the recording for the first time, we initialize the terms v_x, v_y and $\Delta\psi$ as zeros. We assume the x and y velocities and the rate of change of rotation to be constant. With the available information, we can use the vector X_i to predict the position and orientation in next recording X_{i+1} [ER21b]. Considering the time gap between two recordings as Δt, we can compute the following:

- New x-Position : $x_{i+1} \;=\; x_i + \; v_x \Delta t$
- New y-Position : $y_{i+1} \;=\; y_i + \; v_y \Delta t$
- New Rotation angle : $\psi_{i+1} \;=\; \psi_i + \; \Delta\psi \Delta t$
- Width, length, velocity components and rate of change of rotation angle remain same.

Corresponding to the above calculations, we can write the new position and orientation as a matrix multiplication:

$$X_{i+1} \;=\; A \cdot X_i$$

where A is matrix of size 8 by 8.

$$X_{i+1} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & \Delta t & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & \Delta t \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x_i \\ y_i \\ w \\ l \\ \psi_i \\ v_x \\ v_y \\ \Delta\psi \end{pmatrix} \tag{10}$$

We could use a reduced form of the equation (10), as described below. We omit the terms that remain constant (such as width, length, angular rotation change rate). Angular rotation could also be emitted, because given the co-ordinate system and the position of bounding box, we can compute the angle of object w.r.t. ego using elementary trigonometry.

$$X_1 = \begin{pmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x_0 \\ y_0 \\ v_x \\ v_y \end{pmatrix} = \hat{A} \cdot X_0 \tag{11}$$

With the above equation (11), we can denote the position of the object in next recording, as depicted in Fig 3.2 below.

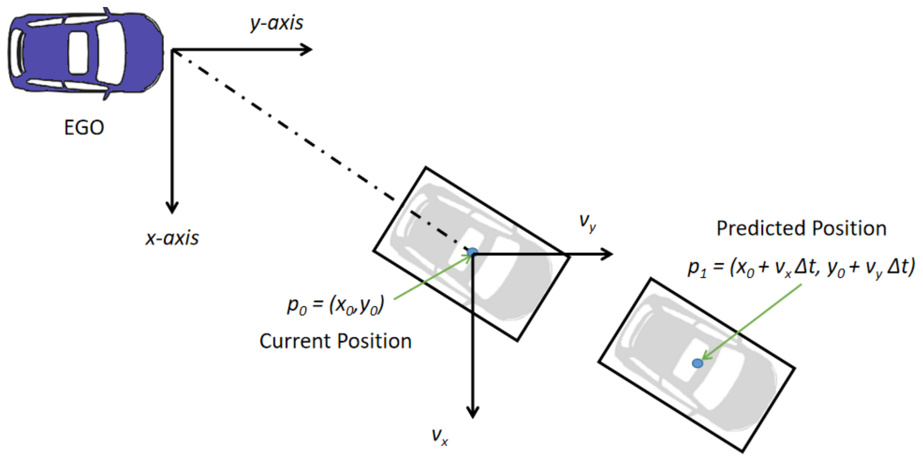


Fig 3.2: Position prediction in next Recording.

4 Error: Real vs Prediction

In Section 3, we described how to predict the position of a tracked object in new recordings. We understand that this prediction is based on the information extracted from the current recording. It is likely that the prediction may not completely match the true data of the next recording. In this case, it will be helpful to define an **error term** between the predicted and the real data. The position and velocity after time Δt is computed by equation (11).

$$X_1 = (x_0 + v_x\Delta t,\ y_0 + v_y\Delta t,\ v_x,\ v_y)^T$$

Let the real position as read by object detector be $Z_1 = (x_1, y_1)^T$. Note that Z_1 contains only information of position, where as in X_1 velocities are also included. In order to compare Z_1 and X_1, we have to transform X_1 vector to same size as that of Z_1. We define a matrix H (called the **observation model**) that enables us to do the comparison [ER21b]. The difference D is denoted by : $D = Z_1 - H \cdot X_1$. In this case, we define H as a 2 by 4 matrix.

$$D = Z_1 - H \cdot X_1 = \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} - \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} x_0 + v_x\Delta t \\ y_0 + v_y\Delta t \\ v_x \\ v_y \end{pmatrix} \tag{12}$$

$$D = \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} - \begin{pmatrix} x_0 + v_x\Delta t \\ y_0 + v_y\Delta t \end{pmatrix} \tag{13}$$

The difference D is called **Innovation**. This will be equal to zero if and only if our prediction is perfect and no further improvement is need. However, this will hardly the case in real conditions. To make the system more robust, we can implement Kalman Filter algorithm to refine our predictions based on, current predicted value and the true recorded data [Kal60]. The performance of a Kalman filter depends on how closely the real behaviour of a sensor can be represented. But, this algorithm only converges if the distributions we try to represent are actually close to a Gaussian distribution.

Articles

[BR20] F. Berens and Hochschule Ravensburg Weingarten. “Lecture notes in Lidar and Radar Systems, How to Astyx Complex Yolo”. In: (Nov. 2020).

[ER21a] S. Elser and Hochschule Ravensburg Weingarten. “Lecture notes in Lidar and Radar Systems, Chapter 5: Distance, Velocity and Angle”. In: (Jan. 2021).

[ER21b] S. Elser and Hochschule Ravensburg Weingarten. “Lecture notes in Lidar and Radar Systems, Chapter 7: Object Tracking and Kalman Filter”. In: (Jan. 2021).

[Che+17] Xiaozhi Chen et al. “Multi-View 3D Object Detection Network for Autonomous Driving”. In: (Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR),2017).

[Kal60] R.E. Kalman. “A new approach to linear filtering and prediction problems”. In: (Journal of basic Engineering,1960).

[Mei+17] F. Meinl et al. “An experimental high performance radar system for highly automated driving”. In: (2017 IEEE MTT-S International Conference on Microwaves for Intelligent Mobility (ICMIM),2017), pp. 71–74.

[Mun57] J. Munkres. “Algorithms for the assignment and transportation problems”. In: (Journal of the society for industrial and applied mathematics,1957).

Webpages

[Boa] National Transport Safety Board. *Preliminary Report Highway HWY18MH010.* URL: `https://www.ntsb.gov/investigations/AccidentReports/Reports/HWY18MH010-prelim.pdf`. (accessed: 12.02.2021).

[Way] Waymo. *Waymo.* URL: `https://waymo.com/`. (accessed: 10.12.2020).

[Wika] Wikipedia. *Doppler Effect.* URL: `https://en.wikipedia.org/wiki/Doppler_effect`. (accessed: 12.02.2021).

[Wikb] Wikipedia. *Interquartile Range.* URL: `https://en.wikipedia.org/wiki/Interquartile_range`. (accessed: 12.02.2021).

[Wikc] Wikipedia. *RADAR.* URL: `https://en.wikipedia.org/wiki/Radar`. (accessed: 20.01.2021).