Geolocating Traffic Signs using Large Imagery Datasets

Pedersen, Kasper F.; Torp, Kristian

Published in:
Proceedings of 17th International Symposium on Spatial and Temporal Databases, SSTD 2021

DOI (link to publication from Publisher):
10.1145/3469830.3470900

Creative Commons License
CC BY 4.0

Publication date:
2021

Document Version
Publisher's PDF, also known as Version of record

Link to publication from Aalborg University

Citation for published version (APA):
Pedersen, K. F., & Torp, K. (2021). Geolocating Traffic Signs using Large Imagery Datasets. In *Proceedings of 17th International Symposium on Spatial and Temporal Databases, SSTD 2021* (pp. 34-43). Association for Computing Machinery (ACM). https://doi.org/10.1145/3469830.3470900

Geolocating Traffic Signs using Large Imagery Datasets

Kasper F. Pedersen
Aalborg University
Dept. of Computer Science
Aalborg, Denmark
kasperf@cs.aau.dk

Kristian Torp
Aalborg University
Dept. of Computer Science
Aalborg, Denmark
torp@cs.aau.dk

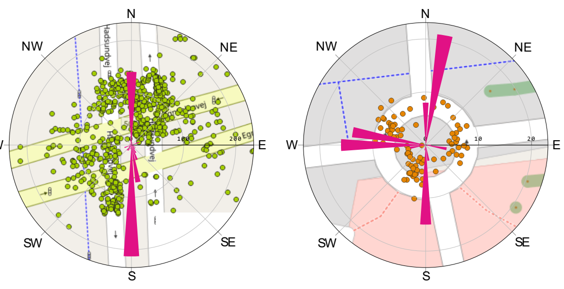
ABSTRACT

Maintaining a database with the type, location, and direction of traffic signs is a labor-intensive part of asset management for many road authorities. Today there are high-quality cameras in cell-phones that can add location (EXIF) metadata to the images. This makes it efficient and cheap to collect large geo-located imagery datasets. Detecting traffic signs from imagery is also much simpler today due to the availability of several high-quality open-source object-detection solutions. In this paper, we use the detection of traffic signs to find both the location and the direction of physical traffic signs. Five approaches to cluster the detections are presented. An extensive experimental evaluation shows that it is important to consider both the location and the direction. The evaluation is done on a novel dataset with 21,565 images that is available free for download. This includes the ground-truth location of 277 traffic signs and all source code. The conclusion is that traffic signs are detected with an F_1 score of 0.8889, a location accuracy of 5.097-meter (MAE), and a direction accuracy of $\pm 11.375°$(MAE). Only data from two trips are needed to get these results.

KEYWORDS

GPS, imagery, traffic sign, clustering

ACM Reference Format:
Kasper F. Pedersen and Kristian Torp. 2021. Geolocating Traffic Signs using Large Imagery Datasets. In *17th International Symposium on Spatial and Temporal Databases (SSTD '21), August 23–25, 2021, virtual, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3469830.3470900



(a) D15.3 in an Intersection (b) D11.3 in a Roundabout

Figure 1: Location and Direction of Traffic Signs

1 INTRODUCTION

High-resolution cameras are available in many consumer products such as cell phones and action cameras. Such devices make it simple to collect very large imagery datasets while driving. Further, a number of open-source software products from major companies such as Facebook, Microsoft, and Google have made general-purpose object-detection networks [12, 16, 18] available to small organizations. This combination of hardware and software makes it possible to detect traffic signs from an imagery dataset and use these detections to annotate maps with new information. In this way, it is possible to automate the labor-intensive process of keeping the placement of traffic signs in a road network up-to-date.

To annotate a map with the placements of traffic signs these have to be accurately detected, e.g., is it a speed limit or a yield sign. Several papers have looked at this issue [3, 22, 24]. However, it is also very important to know an accurate location of a traffic sign and in which direction the traffic sign is pointing, e.g., are you entering or leaving a 60 km/h zone. In this paper, we assume that the traffic signs have been accurately detected and we focus on the geolocation and direction of the traffic signs, both of which can be derived from standardized EXIF metadata [4] available from most cameras, e.g., a GPS location and a timestamp.

A major problem in geolocating traffic signs is that the quality of the GPS location can vary significantly, e.g., due to poor GPS signal quality or bad placement of the camera in the vehicle. This also negatively affects the quality of the direction of travel as the GPS signal is a component in computing it.

The accuracy of location and direction of traffic signs are particularly important in the parts of the road network where there are many traffic signs of the same type. This is illustrated using real-world data for an intersection in Figure 1a and a roundabout in Figure 1b. In these figures, a dot represents the computed location of a traffic sign from imageries. The wind roses show a count of the direction of the traffic signs using a 10° resolution. As can be seen, the traffic signs are detected in directions that are closely related to the directions of the roads. Note that the distribution of directions reflects the cars that have collected the imagery dataset not the traffic in general.

In Figure 1a there are four physical traffic signs. Due to the large number of detections, this is hard to determine from the locations alone (the dots). Detections from different physical traffic signs of the same type are naturally close due to the limited spatial extent of an intersection. Adding the direction in the prediction of the physical traffic signs can help significantly.

In this paper, we look at how to accurately determine the location and the direction of traffic signs. For this purpose, we have collected

SSTD '21, August 23–25, 2021, virtual, USA
© 2021 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-8425-4/21/08.
https://doi.org/10.1145/3469830.3470900

a very large imagery dataset and created the ground truth for a total of 277 traffic signs of 12 different types. The contributions of the paper can be summarized as follows.

We cluster the traffic-sign detections using off-the-shelf clustering algorithms and make a detailed comparison of a general clustering algorithm (DBSCAN) [10] to a specialized clustering algorithm (FCM4DD) [13].

We do an extensive experimental evaluation of five different approaches to geolocate traffic signs. This evaluation is based on a novel, large, real-world imagery dataset. We make a comparison to the ground-truth location and direction provided.

We release the data used in this paper for other researchers to freely download and use in their work to hopefully later improve on the results presented [21].

The main conclusion is that we can cluster traffic-sign detections with an F_1 score of 0.8889. The location of traffic signs can be predicted with a 5.097-meter position accuracy (MAE) and a direction accuracy of $\pm 11.375^\circ$ (MAE). It requires only data from two trips to reach this accuracy (two trips one camera, or one trip two cameras).

The remaining part of the paper is organized as follows. In Section 2 related work is presented. This is followed by a description of the data foundation in Section 3. The five clustering approaches are described in Section 4. Results are presented in Section 5. Section 6 concludes the paper and points to directions of future research.

2 RELATED WORK

Compass clustering [14] is a clustering method that relates existing imagery found in web services like Flickr, to points-of-interests (POIs) such as famous buildings or outdoor art. This method takes advantage of the EXIF metadata found in imagery. Specifically, metadata related to the location and the direction of the camera is utilized. A critique raised in [14] concerns how existing approaches neglect the camera direction and thereby base the relationship solely on distances. A key contribution of [14] is therefore the ability to utilize both a location and a direction to relate imagery to POIs. A notable difference between compass clustering and the work presented in this paper is the viewing angle of the POIs. Attractions often have a 360° viewing angle where traffic signs are limited to a 180° at most. Further, the location of the attraction is known whereas we predict the location and direction of traffic signs.

Fuzzy C-means for Directional Data (FCM4DD) [13] is an algorithm that aims at clustering directional data. As the name suggests, FCM4DD is based on the Fuzzy C-means (FCM) [9] clustering algorithm. FCM4DD is capable of clustering spherical data as well as object directions. As FCM, a predetermined c value (hyperparameter) is required to determine the number of clusters that should be created. The paper compares its contribution to two other directional clustering algorithms and finds similar accuracy, however, FCM4DD has a better runtime performance. Because FCM4DD can be used to cluster object directions, this paper explores if it an algorithm also suitable for clustering traffic signs.

The paper [15] focuses on improving parking-slot detection with a novel clustering algorithm called Directional-DBSCAN (D-DBSCAN). As the name suggests, D-DBSCAN is a customized version of the DBSCAN clustering algorithm. Its objective is to group points that collectively form a straight line. Points that do not form straight lines are marked as noise. Similar to [15], we focus on clustering data based on both locality and directionality using the DBSCAN algorithm. However, we assume that each traffic-sign detection (point) has a direction. Therefore, the concept of directionality differ significantly from what is presented in [15] as they compute a direction of the straight line constructed from multiple points.

Both [6] and [8] focus on traffic sign detection, classification, and mapping, using Google Street View (GSV) imagery. As a result, the papers have a broad system perspective involving computer vision (object detection and object classification) clustering, and mapping. We solely focus on the task of converting traffic sign detections into traffic sign predictions using five different clustering approaches. Further, we use data extracted from imagery captured by consumer-grade hardware and do not rely on GSV imagery.

Similar to [6] and [8] GoMap [20] is a system that locates traffic signs by utilizing road-level imagery. The road imagery is captured by consumer-grade hardware such as smartphones and action cams. Relevant metadata from the imagery is stored in a data warehouse. For detecting traffic signs in the imagery GoMap uses a trained RetinaNet [17] model. The traffic sign detections are clustered using DBSCAN [10] such they can be visualized on a web frontend. In general, GoMap consists of four major system components, namely a data warehouse, object detector, clustering, and a web frontend. The GoMap paper has a broad system perspective whereas this paper focuses solely on the clustering aspect using the detections computed by the system presented in the GoMap paper.

3 DATA FOUNDATION

This section presents how data is collected, processed, and utilized. All the data presented including the imagery is freely available on GitHub [21].

3.1 Imagery Foundation

The basic data foundation is a set of high-resolution images, mostly 12M pixels in a 4K×3K format. All images are captured by consumer-hardware such as action cameras and cell phones that support the standard EXIF format [5] for embedding metadata directly into the images.

To automate the data collection, only time-lapse imagery from trips is used, i.e., the driver only interacts with the hardware at trip start/end and not while driving. Note that all images are collected by drivers with another main purpose than data collection, e.g., driving to work. We measure the size of data collected on a specific road by the number of trips on this road. These trip counts only reflect the driving patterns of the vehicles collecting the imagery and not the traffic in general. This can be seen in 1a where there are more observations in the north-south direction than the east-west direction. The latter is the main road.

3.2 Traffic-Sign Foundation

An existing approach is used to detect the traffic signs on the images. Object detection is outside the scope of this paper, for details, please see [20]. We focus on determining the location and the direction of

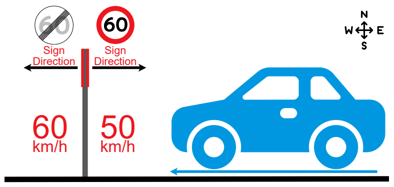


Figure 2: Direction of Vehicle and Traffic Sign

the traffic signs detected. Relevant EXIF tags are used for both of these purposes.

The EXIF *GPSLatitude* tag and the EXIF *GPSLongitude* tag are used to locate the camera. To get the location of the traffic sign, the thin-lens model [11] is used. This model computes the distance between the camera and the detected traffic sign. To compute these distances we use the EXIF tags *FocalLengthIn35mmFilm*, *ExifImageWidth*, and *ExifImageLength*.

The *GPSImgDirection* EXIF tag can be used to compute the direction of the detected traffic sign. Even though there is provided support for the *GPSImgDirection* tag the cameras used for the data collection do not always populate it. Therefore a camera direction is approximated using the GPS trip data. Such an approximation is well-suited if the camera is pointing in the same direction as the vehicle is driving. In this study, cameras are always mounted front-facing in the windscreen. This allows the image direction to be computed from two consecutive GPS points.

The traffic-sign location is found by first computing the distance between the camera and the traffic sign (the thin-lens model). This distance is used as the radius in a circle that has the camera location as its center. Then the traffic-sign location is computed by finding the intersection between this circle and a line string (or vector) from the circle center in the camera direction. For further details see [20].

In the paper, we use 21,565 images distributed among 12 traffic-sign types as shown in Table 1.

3.3 Ground-Truth Data

A ground-truth traffic-sign location is recorded manually using the QField [19] app. In this app, a satellite map allows. e.g., intersections and roundabouts to be used as reference points for locating the traffic signs more accurately. A similar approach is used by the Danish Road Directorate to map traffic-sign locations. In a manual post-processing step, each ground truth location is annotated a traffic sign direction. The direction is deduced using QGIS [1] and the Azimuth Measurement plugin [23].

We define the direction of a traffic sign as the Azimuth angle of the road segment of which the traffic sign is intended to be observed from. This direction is relative to the imprinted side of the traffic sign and thus in the opposite direction of which vehicles are driving. The direction of the vehicle and the traffic-sign is illustrated in Figure 2. When the vehicle is traveling West (270°) it sees traffic signs with the direction East (90°)

Figure 3 shows the area of Aalborg that contains the ground-truth data. In the right-hand corner of Figure 3 an up-close section of the map is shown. The hardware used for the ground-truth data collection is a Samsung Galaxy S20 Plus and a OnePlus 7 Pro. All

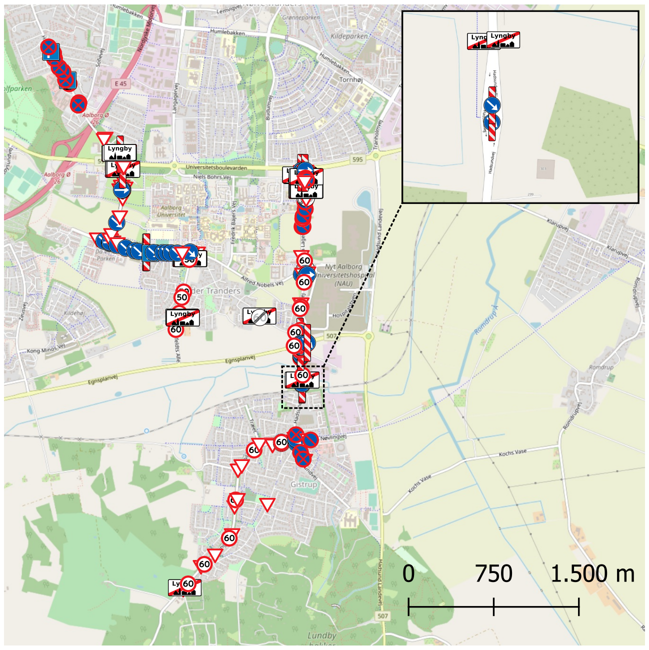


Figure 3: Map Showing Ground Truth

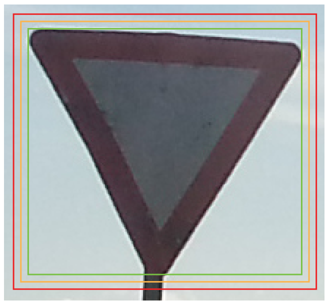


Figure 4: MBRs with Varying Accuracy

ground-truth traffic signs are mapped on-site and are assessed to be within one meter of accuracy. In total 277 ground-truth traffic-signs locations and directions are collected.

To evaluate the output accuracy of each clustering approach, both ground truth and detection data must be available. This limits the dataset to 21.565 traffic-sign detections. To avoid noisy data produced by the traffic-sign detector we remove inaccurate traffic-signs, i.e., all detections with a confidence score < 0.8 are removed. Simple experiments showed that 0.8 is a reasonable threshold. A total of 10.826 traffic-sign detections remains after the confidence score filtration.

All traffic sign detections with a camera distance larger than 30-meters are also removed. The reason for this is shown in Figure 4. Here the green Minimum-Bounding Rectangle (MBR) is correct, the orange MBR has a 2-pixel error, and the red MBR has a 4-pixel error. As the distance between the camera and traffic sign increases this influences the estimation of the traffic sign's location.

Figure 5 quantifies the effect of pixel error with respect to distances for the thin-lens model. The x-axis is the distance between

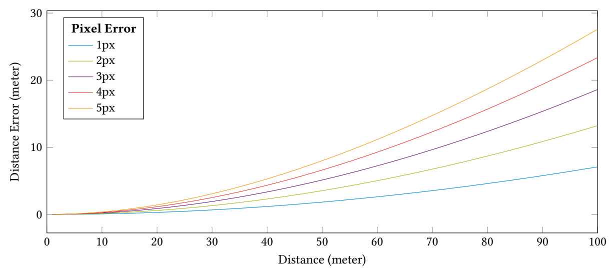


Figure 5: Absolute Error Increase with the Distance

	Ground Truth			Points		
Sign	Train	Val.	Test	Train	Val.	Test
↘	22	11	14	1,632	230	262
▽	41	14	15	1,550	826	737
60	23	7	7	627	288	269
⊗	14	4	4	244	34	159
➡	11	6	8	128	24	68
Lyngby	12	4	2	193	23	22
⊘	8	3	2	89	37	32
60	4	2	2	71	9	28
Lyngby	10	3	3	32	20	7
P	2	2	2	2	16	3
▮	7	2	2	65	17	54
50	2	1	1	12	7	11
Total	156	59	62	4,645	1,531	1,652

Table 1: Train, Validation, and Test Split Distribution

the vehicle and the detected traffic sign. The y-axis shows the distance error in meters. The size of the pixel error varies from one to five. As an example, a traffic sign detected from a 100-meters with a 4-pixel error is placed with an inaccurate of up to 20-meters.

Figure 5 shows that a 30-meters distance threshold allows an MBR error of up to 5 pixels while maintaining an accuracy within 5 meters. This is considered reasonable. This additional filter results in 8,005 traffic sign detections.

Finally, through human inspection, 177 detections are removed due to incorrect classification and 49 are reclassified. This leaves 7,828 traffic-sign detections distributed among the 12 classes shown in Table 1. We do this manual filtering because the focus of this paper is to geolocate traffic signs, the human inspection is intended to remove noise that would not otherwise be present if using a perfect object detector.

3.4 Dataset Splitting

Following good machine learning practices, the dataset is split into training, validation, and testing sets. Mapping each traffic-sign detection individually to the ground truth is a very labor-intensive task. Instead, we split the dataset using the following rules.

(1) *Domain:* Data is split on a per-class basis
(2) *Separation:* A dataset split can only be performed when a spatial separation between two or more clusters are apparent
(3) *Relation:* Ensure all truths relating to a cluster is represented in the same type of set, i.e. training, validation, and testing

The splitting is still manually, However, the task is significantly simplified because it uses spatial regions that contain many traffic-sign detections and not the individual detections. The details of this manual splitting are discussed in the following using the examples shown in Figure 6

A cluster should represent only a single class (traffic-sign type). Thus, clustering is performed once for each type in the dataset, i.e., 12 times. Therefore, the *domain* rule can be enforced without inducing any bias.

For assessing the clustering approaches accurately, it is important to retain the complexity of the original dataset in the training, validation, and testing sets. To retain this complexity the *separation* rule is introduced. Figure 6a and 6b the detections are not well-separated for the ➡ and ▽ signs. If the detections in these examples are split into separate sets, it becomes easier for the clustering approaches presented in Figure Section 4 because the clusters then become well-separated in the experiments, even though this is not the case for the real-world data. In these two cases all, detections are put into the same set.

Figure 6c shows an example where spatial separation is good in the real-world data, simply because the distance between the physical traffic signs is bigger than it is in Figure 6a and Figure 6b. In such cases, the three groupings of detections can be put into each of sets: training, validation, or test.

In this study, clustering performance is evaluated by matching a traffic sign prediction with a ground truth. This implies that a strong relationship exists between clusters and their corresponding ground truth. Obeying the *relation* rule means that no such relationship is broken during the splitting process. As an example, we cannot split any of the groupings of detections in 6c. All detections in a single grouping must be put into the same set.

The aim is a 60:20:20 split ratio. However, with the splitting rules imposed to avoid being biased, it is not possible to do the splitting with exactly these ratios. The final split ratio ended up being 56:21:23. Table 1 shows how the data is distributed between the training, validation, and testing sets.

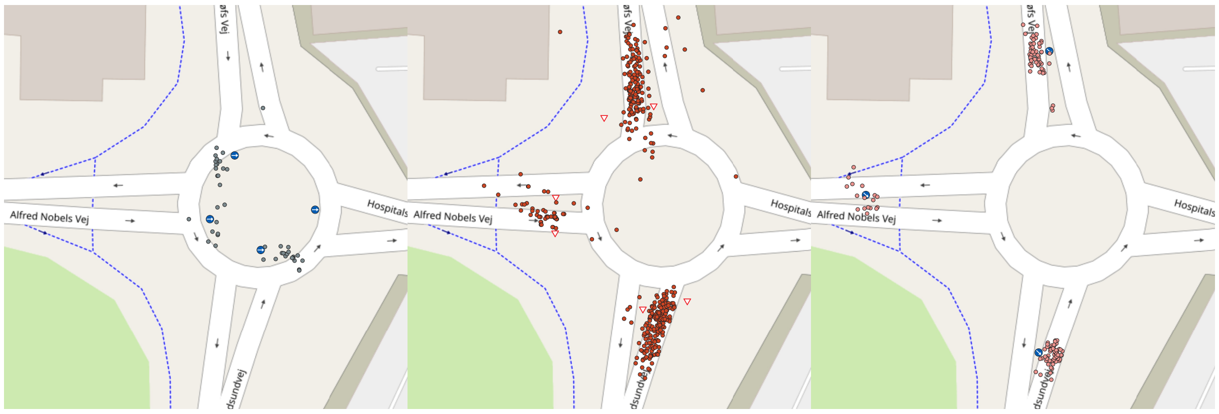
4 CLUSTERING TRAFFIC SIGNS

In this section, five approaches to clustering traffic-sign detections are presented. The goal is to find the location and the direction of the physical traffic sign with the highest accuracy. We first describe what is common to all (or most) of the clustering approaches. Next, each clustering approach is described in detail.

4.1 Overall Idea

All five approaches use off-the-shelf clustering algorithms to cluster the detections. Four out of the five approaches cluster on both the location and the direction. The fifth approach only uses location and acts as a baseline.

The clustering is always done on a per traffic-sign type basis, e.g., first we cluster ▽, then ⊘, and so on. Note that the order of traffic-sign type clustering does not matter.



(a) Testing Split (b) Validation Split (c) Validation Split

Figure 6: Dataset Split Examples with Ground Truth and Detections

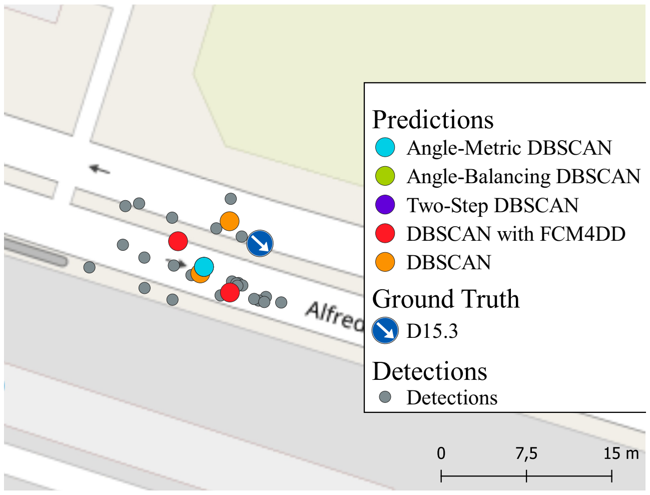


Figure 7: Finding a Traffic Sign Location

The approaches are applicable for directional-based locations in general, i.e., where the data foundation is $\langle id, type, latitude, longitude, direction \rangle$, where *id* is a unique ID for the detection, *type* is the type of the traffic sign, (*latitude*,*longitude*) give a location, and *direction* is a compass direction.

The domain for *direction* is an angle from $0°$ to $359°$. Then this domain wraps, i.e., $360° \equiv 0°$. Therefore, additional clustering steps are needed to cluster detections on both the location and the direction.

All five clustering approaches find a cluster center and a direction. This corresponds to the physical location of a traffic sign and its direction, see Figure 2. The location is computed by finding the geometric center (centroid) of the *Multipoint* enclosing the cluster points. The centroid is used assuming the distribution of the computed locations follows a normal distribution and then using the *central limit theorem*.

Figure 7 is a concrete example of cluster centers produced by each of the five approaches. The three approaches, Angle-Metric DBSCAN, Angle-Balancing DBSCAN, and Two-Step DBSCAN all produce the exact same cluster center, the cyan dot. DBSCAN and DBSCAN with FCM4DD results in two clusters each. The reason for this is explained in subsection 5.1.

The traffic-sign direction is computed by dividing the circle into n equally sized sectors. In this study $n = 36$ such each sector spans $10°$, i.e, $\{[355°, 5°), [5°, 15°), \ldots, [345°, 355°)\}$. Simple experiments showed this is a reasonable sector size.

Each detection is assigned to a sector based on its direction. The sector containing the most detections is selected for estimating the direction of the physical traffic sign. In the case multiple sectors have the same count, the first is selected. From the sector selected, the angle to the midpoint of the sector arc is used as the traffic sign direction. As an example, if the sector $[355°, 5°)$ has the most detections, the traffic-sign direction is set to $0°$.

4.2 DBSCAN

Density-based spatial clustering of applications with noise (DBSCAN)[10] is a widely used and robust clustering algorithm for spatial data. The algorithm takes two parameters *Eps* that is a threshold for when a point is considered noise and *MinPts* that is the minimum number of points allowed in a cluster.

DBSCAN clusters detections based on the Euclidean distance between these. As mentioned in subsection 4.1, the direction domain wraps at $360°$. This domain can therefore not directly be used with the DBSCAN algorithm. The direction domain needs to be transformed before it can be used.

To examine if this transformation of the direction domain improves the quality of the clustering, the first approach only uses the two dimensions $\langle latitude, longitude \rangle$. The approach serves as the baseline for the four other approaches.

4.3 Two-Step DBSCAN

A way to use the direction in the clustering of traffic-sign detections is to split the clustering into two steps. The first step clusters the data spatially, i.e., uses the location. The second step uses the output from the first step to further cluster the detections with respect to a maximum allowed angle difference, i.e., uses the direction.

The Two-Step DBSCAN approach uses the DBSCAN algorithm for both steps. The two steps require three parameters: *Eps*, *MinPts*, and *Max. Angle*. The first two parameters are the same as for the DBSCAN approach. The latter is a threshold for when a point is considered noise based on the direction. The *MinPts* parameter is shared between invocations of the DBSCAN algorithm. The Two-Step DBSCAN approach is described in Algorithm 1

Algorithm 1: Generic Two-Step DBSCAN

```
input  : $TS$, $Eps$, $MinPts$, $MaxAngle$, $DistFunc$
output : A set of clusters
1 $Clusters \leftarrow \emptyset$
2 $SC \leftarrow$ DBSCAN($TS$, $Eps$, $MinPts$)
3 forall $sc \in SC$ do
4     $AC \leftarrow$ DBSCAN($sc$, $MaxAngle$, $MinPts$, $DistFunc$)
5     $Clusters \leftarrow Clusters \cup AC$
6 return $Clusters$
```

The input is a set of traffic-sign detections (*TS*), the three parameters discussed above, and a distance function (*DistFunc*). The output is a set of clusters, a simple example of the output is shown in Figure 7.

In line 2, a default Euclidean distance function is used with the DBSCAN algorithm. This function only considers the location, i.e., latitude and longitude. This first step of the algorithm results in a set of clusters (*SC*).

line 3 to line 5 loops of the clusters found. In line 4, the DBSCAN algorithm uses the distance function *DistFunc* provided as part of the input. *DistFunc* only considers angular data, i.e., directions. The set of clusters found (*AC*) are added to the final output (*Clusters*)

In the Two-Step DBSCAN approach, *DistFunc* operates in Euclidean space and does not wrap angles. We deal with this later.

The order of the two-step clustering in Algorithm 1 is important. The first step must be clustering based on the location and the second step the sub-clustering based on direction. The reason for this is that detections can be spread over a wide spatial area. However, the direction is constrained to domain $[0°, 360°)$. Experiments showed that switching the two steps always gives worse results than reported in Section 5.

4.4 Angle-Metric DBSCAN

The Angle-Metric DBSCAN approach is similar to the Two-Step DBSCAN approach with the exception that the distance function used for clustering in the second step. The pseudo-code presented in Algorithm 1 is reused for the Angle-Metric DBSCAN approach. However, the value for the parameter *DistFunc.* is changed to the *AngleDist* function shown below.

Algorithm 2: *AngleDist* Function

```
input  : $angle_1$, $angle_2$
output : A angular distance
1 return $|(angle_2 - angle_1 + 180°) \bmod 360° - 180°|$
```

The *AngleDist* function wraps the angle, e.g., *AngleDist*(350°, 10°) is 20° and not 340°.

4.5 DBSCAN with FCM4DD

This approach is significantly different from the four other approaches presented in this paper because it uses the specialized FCM4DD algorithm [9] for clustering the direction.

As the name indicates, FCM4DD specializes in clustering angular data. Like the two previous approaches (Two-Step DBSCAN and Angle-Metric DBSCAN) we retain the two-step approach to clustering. However, in the second step, DBSCAN is swapped with FCM4DD. The latter can be seen in line 4 of Algorithm 3.

FCM4DD takes four parameters, namely c which decides how many clusters should be created, m which is the weighting fuzziness parameter, *MinImpr* which defines a threshold for when an optimal solution is found, and *MaxIter*, which defines a threshold for a maximum number of iterations that should be performed to find an optimal solution.

Algorithm 3: DBSCAN with FCM4DD

```
input  : $TS$, $Eps$, $MinPts$, $c$, $m$, $MaxIter$, $MinImpr$
output : A set of clusters
1 $Clusters \leftarrow \emptyset$
2 $SC \leftarrow$ DBSCAN($TS$, $Eps$, $MinPts$)
3 forall $sc \in SC$ do
4     $AC \leftarrow$ FCM4DD($sc$, $c$, $m$, $MaxIter$, $MinImpr$)
5     $Clusters \leftarrow Clusters \cup AC$
6 return $Clusters$
```

A difference between Algorithm 1 and Algorithm 3 is in the input where the FCM4DD algorithm takes more parameters. Also, line 4 is different. Here Algorithm 3 uses the FCM4DD algorithm.

4.6 Angle-Balancing DBSCAN

The three previous approaches that use both the location and the direction of detections cluster all data twice due to the two-step implementation. This section presents a single-step approach called Angle-Balancing DBSCAN. To do single-step clustering requires a different approach to handling directions (angles). This difference is illustrated in Figure 8

Figure 8 shows the locations of three detections labeled A, B, and C using dashed circles. The direction of each detection is shown using arrows, e.g., A to the East and C to the North. If the directions are used to map the detections to a circle $[0°, 360°)$ (the green arc) we get the solid circles labeled A', B', and C' at 0°, 45°, and 90°, respectively.

Looking at Figure 8 A and C seem close. However, A and C are only close from a location perspective. Seen from a direction perspective A' and C' are far apart. We therefore need to go from three dimensions (latitude, longitude, and direction) to four dimensions: two dimensions from the location and two dimensions from the converted direction, i.e., look at the direction as another (2D) location.

Equation 1 defines how to calculate a location from a direction. A', B', and C' are the output of Equation 1 given the locations A, B, and C.

Now a distance can be calculated between any pair of A', B' and C'. Note how the arc length of $A'B'$ ($\frac{\pi}{4}$) is half of the arc length of

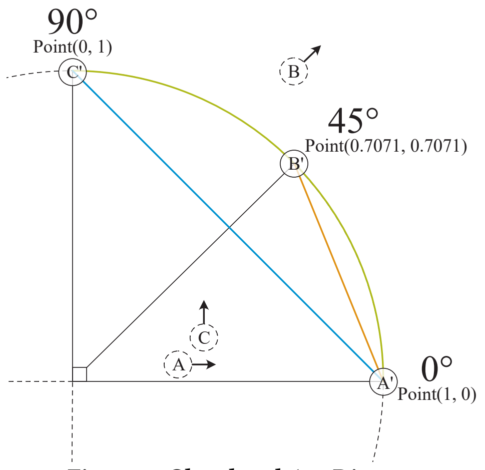


Figure 8: Chord and Arc Distance

$A'C'$ ($\frac{\pi}{2}$). This relation is not true for the chord lengths (Euclidean distances) $A'B'$ (0.7654) and $A'C'$ ($\sqrt{2}$). Therefore, the magnitude of chord lengths cannot be directly compared to each other. However, chord lengths can be compared in terms of =, < and >, as anything $> \sqrt{2}$ is also $> \frac{\pi}{2}$ (90°) and anything $< \sqrt{2}$ is also $< \frac{\pi}{2}$ (90°). In conclusion, DBSCAN can be used directly to cluster on four dimensions using a Euclidean distance function.

$$Point = \langle \cos(direction),\ \sin(direction) \rangle \tag{1}$$

$$c^2 = a^2 + b^2 - 2ab * \cos(C) \tag{2}$$

$$c = \sqrt{2 - 2 * \cos(C)} \tag{3}$$

One remaining problem is that the four dimensions longitude, latitude, cos($direction$) and sin($direction$) do not have the same unit. Therefore, the four dimensions are scaled such distances measured between all four dimensions weigh in evenly.

Similar to *Eps* defining a maximum allowed distance from any cluster point to its nearest neighbor, a *MaxAngle* is needed to define the maximum allowed directional difference from any direction to its nearest directional neighbor. Such *MaxAngle* parameter needs to be converted to a chord length to compare it with other calculated chord lengths. For this conversion, The Law of Cosines Equation 2, is used, where C is the *MaxAngle*. Because we operate on a unit circle Equation 2 can be simplified to Equation 3.

Angle-Balancing DBSCAN is described in Algorithm 4. Note that the input is the same as in Algorithm 1 In line 2 the $ScaleFactor$ is calculated. This factor is used to balance the directional coordinates, $XDir$ and $YDir$ in line 4 and line 5, respectively. In line 6 the four-dimensional data points are created. These points are then clustered using the DBSCAN algorithm in line 7 using a Euclidean distance function.

5 RESULTS

All results are computed on a server with an Intel I9 Processor (9900KF), 32GB 3.20GHz DDR4 RAM, 1 TB PCIe SSD hard drive, and an RTX2080Ti graphics card with 11 GB GDDR6 RAM. The graphics card is not used for clustering.

Algorithm 4: Angle-Balancing DBSCAN

```
input  : $TS$, $Eps$, $MinPts$, $MaxAngle$
output : A set of clusters
1 $Data \leftarrow \emptyset$
2 $ScaleFactor \leftarrow \frac{Eps}{\sqrt{2-2*\cos(MaxAngle)}}$
3 forall $ts \in TS$ do
4     $XDir \leftarrow \cos(ts.dir) * scaleFactor$
5     $YDir \leftarrow \sin(ts.dir) * scaleFactor$
6     $Data \leftarrow Data \cup \{\langle ts.lng,\ ts.lat,\ XDir,\ YDir \rangle\}$
7 return DBSCAN($Data$, $Eps$, $MinPts$)
```

5.1 Tuning Hyperparameters

As described in Section 3, the dataset is divided into training, validation, and test sets. The training set is used in conjunction with Optuna [2] to tune the hyperparameters. Optuna's built-in Tree-structured Parzen Estimator (TPE) [7] sampler is used for the parameter tuning. This setup is used for each of the five clustering approaches discussed in Section 4.

The hyperparameters are optimized with respect to the F_1 score as it describes both the *recall* and the *precision*. The objective of this paper is to recall a high percentage of the traffic signs without too many false positives, hence using the F_1 score. The tuning process is stopped when the validation loss has not improved for 1,000 iterations. In total, all five approaches are tuned within an hour. Table 2 lists the hyperparameters yielding the best results on the validation set. The maximum number of iterations varies from 3,168 for DBSCAN with FCM4DD to 3,648 for Angle-Metric DBSCAN.

DBSCAN has a significantly lower *Eps* value compared to the four other approaches. The lower *Eps* value is assumed to be caused by its inability to filter on the direction, and thereby leading to more strict spatial separation. Such low *Eps* can induce undesirable spatial separation. A concrete example of undesirable spatial separation can be seen in Figure 7. Here the low *Eps* value separates the detections into two clusters. This results in the two incorrectly plotted cluster centers for the DBSCAN approach.

As described in subsection 4.5 FCM4DD requires additional hyperparameters. *MaxIter* and *MinImpr* are both thresholds for improving the runtime performance of the FCM4DD algorithm. However, during the hyperparameter optimization, there has been no objectives to minimize these values. Therefore, lower values may exist yielding similar results with better runtime performance. Further, FCM4DD takes a parameter c that determines the number of clusters that should be found. As seen in Table 3 a value two has proven most desirable for such c parameter. However, Figure 7 illustrates that a predefined c value results in suboptimal solutions if the true cluster count differs from the selected c value, in this case, one instead of two.

5.2 Determining the Ground Truth

In order to calculate the metrics shown in Table 3, predictions need to be mapped to their corresponding truths. Algorithm 5 shows how a prediction of the location of a single, physical traffic sign is mapped to a ground truth. In line 2 the location of a prediction is buffered by $MaxDist$. Because a location is a point, the buffer creates

Name	Eps	MinPts	MaxAngle	c	m	MaxIter	MinImpr
Angle-Metric DBSCAN	10.7	2	27.5	-	-	-	-
Angle-Balancing DBSCAN	11.0	2	40.8	-	-	-	-
Two-Step DBSCAN	13.5	2	34.2	-	-	-	-
DBSCAN	3.9	2	-	-	-	-	-
DBSCAN with FCM4DD	10.5	2	-	2	8.3	180	0.4820

Table 2: Hyperparameters Settings

Algorithm 5: Find the Ground Truth

```
input  : $Prediction, Truths, MaxDist$
output : A truth or $null$
1  $BC \leftarrow null$
2  $SS \leftarrow Buffer(Prediction.Loc, MaxDist)$
3  $C \leftarrow Truths.Query(SS)$
4  forall $c \in C$ do
5      if $c.Visited$ then
6          continue
7      $X \leftarrow \frac{AngleDist(Prediction.Dir,\ c.Dir)}{180}$
8      $Y \leftarrow \frac{\text{Dist}(Prediction,\ c)}{MaxDist}$
9      $c.Score \leftarrow X + Y$
10     if $BC = null \lor c.Score < BC.Score$ then
11         $BC \leftarrow c$
12 if $BC \neq null$ then
13     $BC.Visited \leftarrow true$
14 return $BC$
```

a circle with a *MaxDist* radius. This circle acts as the search space. The *Query* in line 3 is a spatial range query that uses the search space *SS* and returns a set of candidate truths *C*. All candidates in *C* are considered possible truths for the prediction in question.

The task of finding the best candidate *BC* from the candidate set is defined from line 4 to 11. Finally, if a candidate is found it gets marked as visited on line 13. This ensures that each ground truth is used at most one time.

5.3 The Ground Truth for Specific Scenarios

The result combining the estimated physical location of a traffic sign and the ground-truth is shown in Figure 9 for four typical road-network scenarios where the same traffic sign appears multiple times in close in ground-truth data. The legend from Figure 7 is reused. We focus mostly on the location as it is easy to illustrate, direction is covered in detail in the next section. Please note that the scale is the same for all four sub-figures.

Figure 9a shows that Angle-Balancing DBSCAN approach for a round-about. The clusters are well-separated and the physical locations are estimated well. The approach has problems if two traffic signs of the same type (here ▽) are placed on both sides of the road near each other. More generally if multiple traffic signs of the same type are close both in the location and the direction this approach may miss one of the traffic signs.

Figure 9b shows the result for another common road-network scenario using the DBSCAN with FCM4DD approach clustering ⑥⓪ traffic signs. Due to the cluster-size hyperparameter (c) this approach always finds two cluster centers. This gives the correct result for the south leg of the intersection where there are two physical traffic signs close together. In the other cases, it gives an extra traffic sign with no ground truth. This is the opposite problem of what is shown in Figure 9a.

Figure 9c shows a result using the Angle-Metric DBSCAN approach where there are multiple traffic signs of the same type (here ⑤⓪) along a road. The Angle-Metric DBSCAN approach finds the sign close to the ground truth because the clusters are well-separated. The area of the ground-truth is a medium-size city (Aalborg) and suburban areas and the results shown in Figure 9c are typical for this road-network scenario.

Figure 9d shows the result for the Two-Step DBSCAN approach in a scenario similar to what is shown in Figure 9b. The locations of the traffic signs (here ●) are estimated well. Note that this double detection is not an error there are two traffic signs at the same physical location but pointing in opposite directions. This shows the benefit of using the direction in the clustering.

5.4 Location and Direction Accuracy

The results in the previous section are based on four specific scenarios. This section generalizes the results.

To avoid bias towards the validation set, the hyperparameters presented in Table 2 are used on the test set resulting in Table 3. The table is ordered by F_1 score such the best performing approach is listed first.

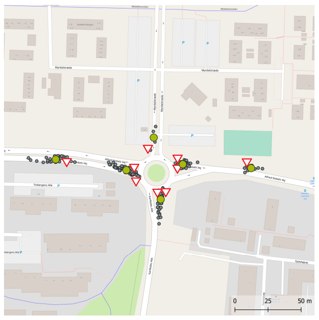
The four best approaches all take the direction into account where the three best do not require a known cluster size (c value in Table 2). Therefore, even though FCM4DD does handle angles, it seems like there exists no single c value that will fit the entire dataset well.

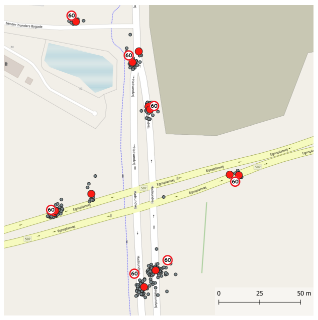
In general, the three best approaches have a low false-positive rate with precision ranging from 0.89 to 0.92. Further, the mean direction error is $\pm 11.38°$ and the mean locational error is at most 5.12 meters.

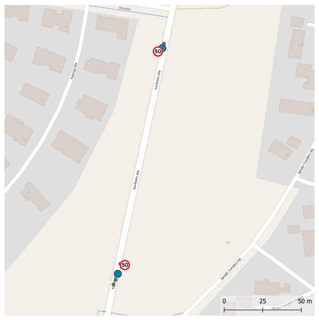
5.5 Number of Trips Required

The imagery data foundation described in Section 3 is based on trips driven by drivers with other main purposes than to collect data. It is therefore relevant to look at how the five approaches perform with a varying number of trips.

Figure 10 presents such an experiment and is conducted as follows for each of the five approaches.


(a) Angle-Balancing DBSCAN


(b) DBSCAN with FCM4DD


(c) Angle-Metric DBSCAN

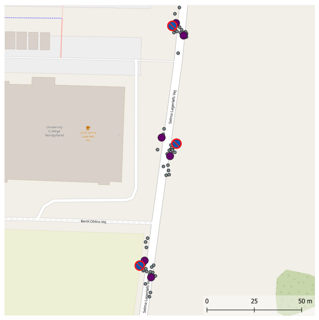
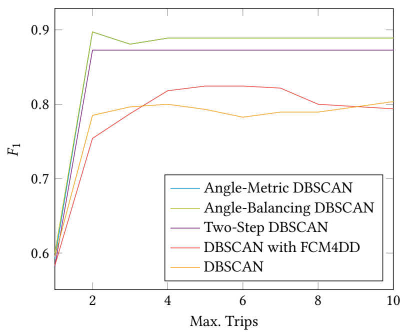

(d) Two-Step DBSCAN

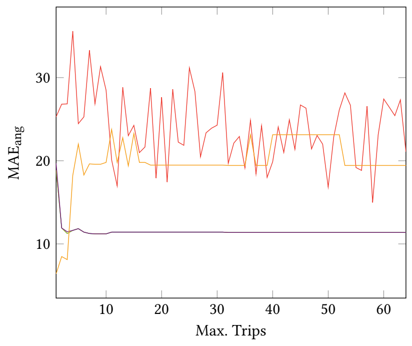
Figure 9: Detections, Computed Locations, and Ground Truth for Four Road-Network Scenarios

Name	F_1	Recall	Precision	MAE_{ang}	RMSE_{ang}	MAE_{loc}	RMSE_{loc}
Angle-Metric DBSCAN	0.8889	0.8571	0.9231	±11.375°	±35.566°	5.097 m	5.895 m
Angle-Balancing DBSCAN	0.8889	0.8571	0.9291	±11.375°	±35.566°	5.110 m	5.906 m
Two-Step DBSCAN	0.8727	0.8571	0.8889	±11.375°	±35.566°	5.123 m	5.922 m
DBSCAN	0.8036	0.8036	0.8306	±19.756°	±52.384°	5.878 m	6.840 m
DBSCAN with FCM4DD	0.7813	0.8929	0.6944	±31.080°	±63.707°	6.138 m	7.112 m

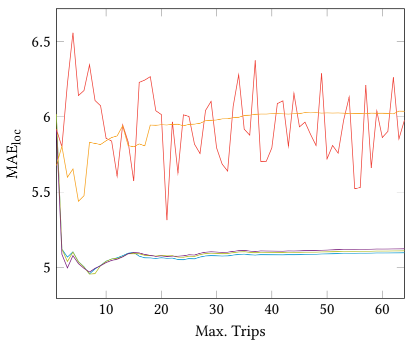
Table 3: Location and Direction Accuracy



(a) F_1 score relation to No. of Trips



(b) MAE_{ang} relation to No. of Trips



(c) MAE_{loc} relation to No. of Trips

Figure 10: Effect of Number of Trips

(1) Find all detectable physical traffic signs, i.e.m clusters
(2) Set n equal to the maximum trips found in a cluster
(3) Remove the newest trip from clusters with more than n trips
(4) Calculate F_1 score, MAE_{ang}, MAE_{loc}, RMSE_{ang} and RMSE_{loc}
(5) Subtract one from n and go to step 3 if $n > 0$

The legend found in Figure 10a applies to all three sub-figures in Figure 10. Figure 10a shows that only two trips are needed to get the results shown in Figure Table 3. The imagery from two trips can be collected driving a single trip with two cameras. The optimal number of trips is partly dataset dependent and may be higher than two for a more dense dataset.

Therefore, locating traffic signs on a significant number of roads using is highly practicable. It is worth noting that everything beyond two trips does not improve the F_1 score significantly. For illustration purposes, Figure 10a only shows up to 10 trips. However, tests have been done with up to 60 trips.

Figure 10b and Figure 10c show how the location and direction accuracy changes with the number of trips being added. However, both DBSCAN with FCM4DD and the unmodified DBSCAN have variance in the accuracy depending on the number of trips. The top three approaches and are more robust with respect to number of trips required.

The difference between MAE_{ang} and RMSE_{ang} in Table 3 is caused by a few outlies. For the sake of readability outliers have been excluded from Figure 11. However, all percentiles are lower than the MAE_{ang} supporting the few outlier argument. Therefore, the MAE_{ang} is a robust measure of the angular accuracy.

6 CONCLUSION

In this paper, five approaches to cluster traffic-sign detections are described, tuned, and evaluated. The approach uses the existing DBSCAN and FCM4DD algorithms.

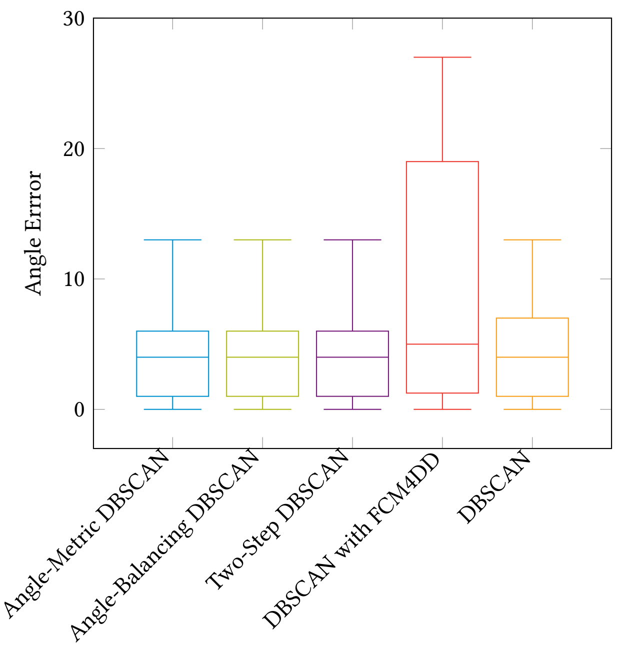


Figure 11: Angle Errors

The three best approaches are all based on the DBSCAN algorithm and produce very similar results. All five approaches reach their upper bound F_1 score with data from only two trips. Using more than two trips has a limited impact on the F_1 score.

Traffic signs can be placed with a locational accuracy of 5.097-meter (MAE) and a directional accuracy of $\pm$ 11.375° (MAE). It is important to consider both the location and direction of the detections, particular in intersection and round-about where traffic signs of the same type are close.

The specialized clustering algorithm FCM4DD performs worse than the general DBSCAN algorithm. The Angle-Balancing DBSCAN is ranked second but only requires one-pass over the data. The two-pass Angle-Metric DBSCAN is only marginally better. This paper also shows how a large imagery dataset can be collected, split, and used for finding both the location and the direction of traffic signs. The dataset is handled in a cost effective manner and includes imagery from both rural and urban areas.

All data used in this paper is available on GitHub. This includes the imagery dataset, the object detections, and the ground truth data. In addition, the source code for the clustering is made available.

Future work includes more advanced ways to place the cluster center, i.e, the physical location of the traffic sign. This location typically follows patterns related to the road network, e.g., to the right of the road, a few meters from where the roads in an intersection meet. Another direction of future work is dealing with the multiple traffic signs of the same type that are close both in location and direction.

REFERENCES

[1] 2021. Welcome to the QGIS project! https://www.qgis.org/en/site [Online; accessed 1. Jul. 2021].

[2] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. 2019. Optuna: A Next-generation Hyperparameter Optimization Framework. In *Proceedings of the 25rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.*

[3] Alvaro Arcos-Garcia, Juan A Alvarez-Garcia, and Luis M Soria-Morillo. 2018. Evaluation of deep neural networks for traffic sign detection systems. *Neurocomputing* 316 (2018), 332–344.

[4] Camera & Imaging Products Association. 2019. *Exchangeable image file format for digital still cameras: Exif Version 2.32.* Technical Report. Camera & Imaging Products Association.

[5] Camera & Imaging Products Association. 2019. *Exchangeable image file format for digital still cameras: Exif Version 2.32.* Technical Report.

[6] Vahid Balali, Armin Ashouri Rad, and Mani Golparvar-Fard. 2015. Detection, classification, and mapping of US traffic signs using google street view images for roadway inventory management. *Visualization in Engineering* 3, 1 (2015), 1–18.

[7] James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. 2011. Algorithms for hyper-parameter optimization. In *25th annual conference on neural information processing systems (NIPS 2011)*, Vol. 24. Neural Information Processing Systems Foundation.

[8] Andrew Campbell, Alan Both, and Qian Chayn Sun. 2019. Detecting and mapping traffic signs from Google Street View images using deep learning and GIS. *Computers, Environment and Urban Systems* 77 (2019), 101350.

[9] Joseph C Dunn. 1973. A fuzzy relative of the ISODATA process and its use in detecting compact well-separated clusters. (1973).

[10] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise.. In *Kdd*, Vol. 96. 226–231.

[11] W. Fulton. 2020. Calculate Distance or Size of an Object in a Photo Image. https://www.scantips.com/lights/subjectdistance.html.

[12] Ross Girshick. 2015. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision.* 1440–1448.

[13] Orhan Kesemen, Özge Tezel, and Eda Özkul. 2016. Fuzzy c-means clustering algorithm for directional data (FCM4DD). *Expert systems with applications* 58 (2016), 76–82.

[14] Yuri Almeida Lacerda, Robson Gonçalves Fechine Feitosa, Guilherme Álvaro Rodrigues Maia Esmeraldo, Cláudio de Souza Baptista, and Leandro Balby Marinho. 2012. Compass clustering: A new clustering method for detection of points of interest using personal collections of georeferenced and oriented photographs. In *Proceedings of the 18th Brazilian symposium on Multimedia and the web.* 281–288.

[15] Soomok Lee, Daejin Hyeon, Gikwang Park, Il-joo Baek, Seong-Woo Kim, and Seung-Woo Seo. 2016. Directional-DBSCAN: Parking-slot detection using a clustering method in around-view monitoring system. In *2016 IEEE Intelligent Vehicles Symposium (IV).* IEEE, 349–354.

[16] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. 2017. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision.* 2980–2988.

[17] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. 2017. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision.* 2980–2988.

[18] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. 2016. Ssd: Single shot multibox detector. In *European conference on computer vision.* Springer, 21–37.

[19] OPENGIS.ch. [n.d.]. About QField. https://qfield.org/

[20] Kasper F Pedersen and Kristian Torp. 2020. Geolocating Traffic Signs using Crowd-Sourced Imagery. In *Proceedings of the 28th International Conference on Advances in Geographic Information Systems.* 199–202.

[21] Kasper F. Pedersen and Kristian Torp. 2021. GomapClustering. https://github.com/fromm1990/GomapClustering [Online; accessed 1. Jul. 2021].

[22] Domen Tabernik and Danijel Skočaj. 2019. Deep learning for large-scale traffic-sign detection and recognition. *IEEE transactions on intelligent transportation systems* 21, 4 (2019), 1427–1440.

[23] webgeodatavore. 2021. azimuth_measurement. https://github.com/webgeodatavore/azimuth_measurement [Online; accessed 1. Jul. 2021].

[24] Jianming Zhang, Manting Huang, Xiaokang Jin, and Xudong Li. 2017. A real-time chinese traffic sign detection algorithm based on modified YOLOv2. *Algorithms* 10, 4 (2017), 127.