CHAPTER 4
APPLICATION 1: ROAD MAP REFINEMENT

The experiments presented in Chapter 3 all involved artificial problems; we used standard data sets in conjunction with constraints that were randomly generated from the true data labels. Those constraints were therefore completely reliable, in terms of guiding the clustering algorithm towards a result that was consistent with the labels used for evaluation.

However, we are also very interested in testing these methods in the context of real-world problems. Can real (and potentially noisy) domain knowledge be successfully represented as instance-level constraints, and will those constraints enable significant improvements in clustering accuracy? In this chapter, we first explore the application of intelligent clustering to the problem of automatic road map refinement (Section 4.1). Section 4.2 examines the performance of the regular k-means algorithm on this problem. In Section 4.3, we describe how to encode heuristic domain knowledge about this problem as a set of instance-level hard constraints. As we will show, the unconstrained k-means algorithm performs very poorly compared to COP-KMEANS, which has access to additional domain knowledge about the problem (Section 4.4). In the two chapters that follow, we will demonstrate the utility of our methods on two other real-world problems.

4.1 Digital Road Maps

Digital road maps currently exist that are used in several applications, such as generating personalized driving directions. However, these maps contain only coarse information about the location of a road. Map accuracy, in terms of how close the map points are to the true location of the road, is low. For example, Navigation Technologies, Inc., (NavTech) offers maps with individual points that are up to 15 meters off from the true road location (Navigation Technologies, 1996). This level of accuracy, while sufficient for generating directions for a human driver to follow, is inadequate for more sophisticated applications.

Our goal in this application is to refine digital maps by improving their accuracy and enhancing the fine details. We will produce maps that are annotated not only with the location of the road, but also with the location of individual road lanes. Maps of this nature enable the automatic generation of more precise driving directions as well as a host of more advanced applications, such as alerting a driver who inadvertently drifts from the current lane.

4.1.1 Using GPS Data to Automatically Refine Maps

One method for producing enhanced maps would be to manually re-survey each road segment on the map and to update the road maps by hand to include the new information. This would not only be extremely tedious and expensive but would also take a long time to complete. Instead, we propose using an automated method to analyze actual automobile driving patterns and extract information about the location of road lanes. This approach is low-cost, fast, and allows for easy updates to the road maps. Further, if a road lane is closed due to construction or some other obstruction, our analysis methods can detect that that lane is no longer in use and update the current status of the maps accordingly.

Our approach to this problem is based on the observation that drivers tend to drive within lane boundaries (rather than, for example, straddling two lanes). Over time, lanes should correspond to "densely traveled" regions, in contrast to the lane boundaries, which should be "sparsely traveled." Consequently, we hypothesized that it would be possible to collect data about the location of cars as they drive along a given road, and then cluster that data to automatically determine where the individual lanes are located.

4.1.2 Experimental Methodology

The data set we will be using consists of observations of several cars driving on the same network of roads. Each car's position was sampled approximately once per second using differential GPS (Global Positioning System) receivers affixed to the top of the vehicle. The DGPS receiver recorded the longitude, latitude, and altitude of the vehicle, as well as some information about how reliable each sample was. All data was taken from I-280 in the Palo Alto, CA, region. We define a *traverse* as a single pass by the same car from one end of a road segment to the other end. Consequently, each vehicle may have made more than one traverse of a given road.

Road segmentation. To facilitate processing, we segmented the data generated by each driver by matching it to NavTech's coarse road map segments. A road segment is a continuous section of a road that is terminated on each end where it intersects with other roads. Very long stretches of road may also be subdivided into separate segments by the NavTech map. Although NavTech's data points are not good enough to pinpoint the exact location of a road, they are good enough to allow us to segment the data streams into individual stretches of road.

Data format. We then converted the data into the following format. Each data point is represented by two features: its distance along the road segment and its perpendicular offset from the road centerline.[1] For evaluation purposes, we asked the drivers to indicate which lane they occupied and any lane changes. This allowed us to label each data point with its correct lane.

4.2 K-means Performance on Lane Finding

The k-means clustering algorithm (MacQueen, 1967) is a standard choice for clustering in Euclidean space. K-means has demonstrated good performance on a variety of problems, from image segmentation (Marroquin and Girosi, 1993) to speech recognition (Fontaine et al., 1994), and is known to converge to a local optimum (Selim and Ismail, 1984). It is therefore a sensible first choice for clustering our GPS data.

However, a straightforward application of k-means yields poor results on this problem. For example, Figure 4.1 shows the data set collected from several traverses of the same road segment. The horizontal axis is the distance along the road (in meters) and the vertical axis is the centerline offset. Figure 4.2 shows the output of the regular k-means algorithm on that road segment. We provided the correct value of k (number of lanes, 4) as input to k-means.

[1] The centerline parallels the road but is not necessarily located in the middle of the road. Schroedl et al. (2000) provide full details on the centerline computation.

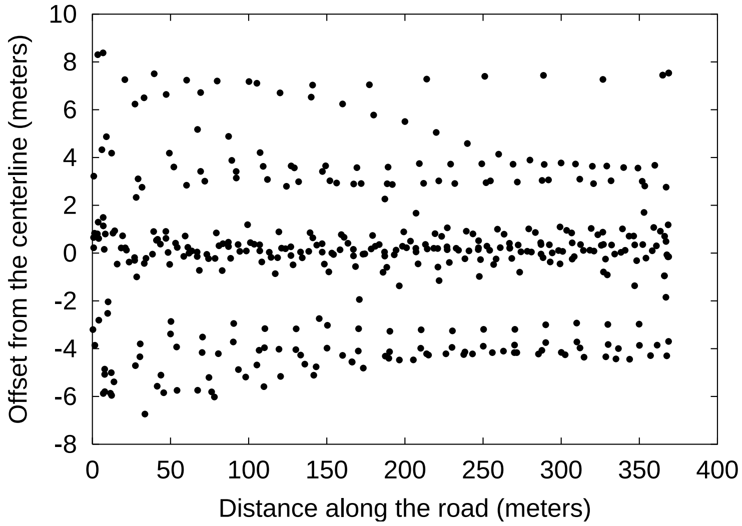

Figure 4.1: Raw data for a sample road segment; the x-axis is the distance along the road (in meters) and the y-axis is offset from the road centerline

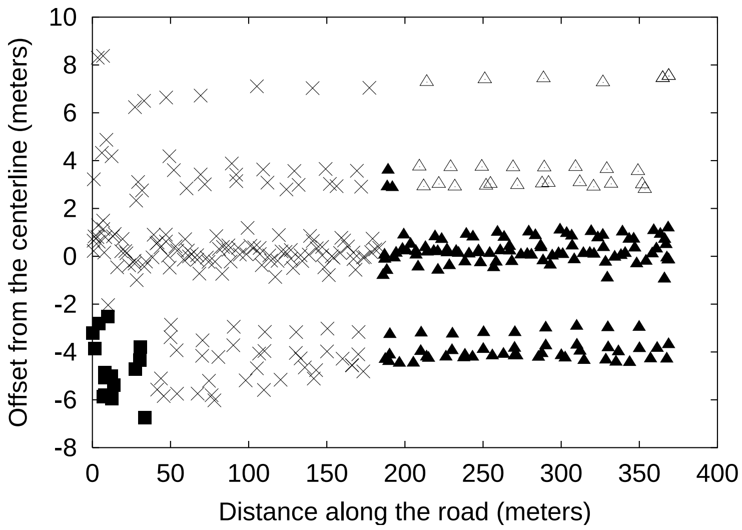

Figure 4.2: K-means output partition for the sample road segment in Figure 4.1

The points assigned to each of the resulting four lane clusters are represented by different symbols. This algorithm does a poor job of identifying the four lanes, and the mistakes it makes are representative of its general behavior on this problem. K-means by default seeks clusters that are spherical in the feature space, which results in clusters that span multiple lanes in the y direction and do not span the entire length of the road in the x direction.

To alleviate this problem, we could manually modify the distance calculation so that distance in the y direction is more heavily weighted than distance in the x direction is. Alternatively, we could re-scale all of the feature values to place more emphasis on the y distance than on the x distance. From the viewpoint of the k-means algorithm, either method would have the effect of compressing the feature space horizontally. However, the first approach would have us tweaking the distance calculation differently for every problem the k-means algorithm encounters. The second approach also has a major drawback: it requires an intimate knowledge of the clustering algorithm's bias so that the data can be pre-processed in ways that will improve performance. In contrast to both of these approaches, our method allows the algorithm to remain problem-independent. We do not need to make any internal changes to the algorithm to fit it to a specific problem, and we do not need to pre-process the data to make it fit the algorithm. Instead, we specify all problem-specific knowledge in a separate, isolated component (Con), and allow the algorithm to combine it automatically with the data.

With respect to the lane finding problem, we observe that the built-in bias of the k-means algorithm (towards spherical clusters) is inappropriate. However, this is because the k-means algorithm sees the data set only as a set of generic data points. It does not know that it is seeking road lanes, which have a specific characteristic shape. We know that road lanes tend to be elongated (along the road segment) and that they tend to be parallel to each other. In the following section, we will show how to convert our domain knowledge about the problem into useful instance-level constraints and report on the results of using those constraints with COP-KMEANS, the intelligent clustering version of k-means.

4.3 Generating Lane-finding Constraints

The constraint types defined in Chapter 3 provide us with a language to express our domain knowledge about the problem of lane finding. In this case, we focus on two domain-specific heuristics for generating constraints: trace contiguity and maximum separation.

Heuristic 1: *Trace contiguity* means that, in the absence of lane changes, all points a, b generated by the same vehicle in a single pass over a road segment should end up in the same lane. In other words, traces are contiguous in that all of the points from a single trace are linked. We convert this heuristic into a set of constraints in the following way:

$$\underset{a,b}{\forall}\ a.traverse = b.traverse \Rightarrow (a, b) \in Con_{=}$$

where each point is labeled according to its traverse of the segment. Again, the same car may have made more than one traverse of the same segment; in this case, points generated by the same car but on different passes will have different values for the *traverse* feature.

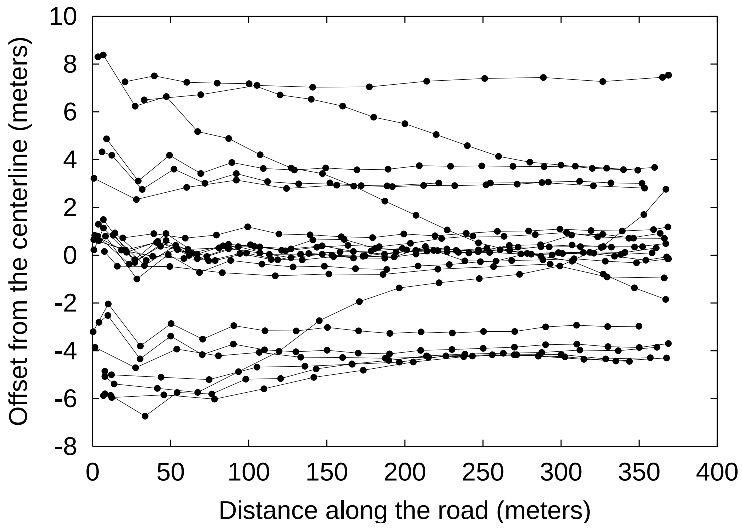

Figure 4.3: Domain knowledge encoded as pairwise links between points generated by the same traverse of the road segment

Figure 4.3 provides a conceptual representation of the relational trace contiguity information by linking each adjacent pair of points that came from the same traverse. We can see that two points may appear to be very far apart in this Euclidean space, yet still be generated by the same car. The transitivity of the must-link relation ensures that two such points will still end up in the same lane cluster.

Heuristic 2: *Maximum separation* refers to a limit on how far apart two points can be (perpendicular to the centerline) while still being in the same lane. If two points are within ten meters of each other in distance along the road segment, and their centerline offsets differ by at least four meters, then we generate a constraint that will prevent those two points from being placed in the same cluster:

$$\underset{a,b}{\forall}\ |a.x - b.x| < 10.0 \text{ and } |a.y - b.y| > 4.0 \Rightarrow (a, b) \in Con_{\neq}$$

This heuristic helps express the idea that lanes are parallel; the fact that two points are close together in the x direction has a different meaning than two points being close together in the y direction. Recall that the x and y axes do not correspond to a fixed set of coordinates in the real world. They simply represent "distance along the road" and "perpendicular offset from the centerline." The x axis follows the road centerline, which may be curving in the real Euclidean space.

We could also have included a set of links in Figure 4.3 between items deemed too far apart by the maximum separation heuristic, but this would make the figure difficult to interpret visually. The full set of constraints representing our domain knowledge about the problem, however, includes both $Con_{=}$ and $Con_{\neq}$.

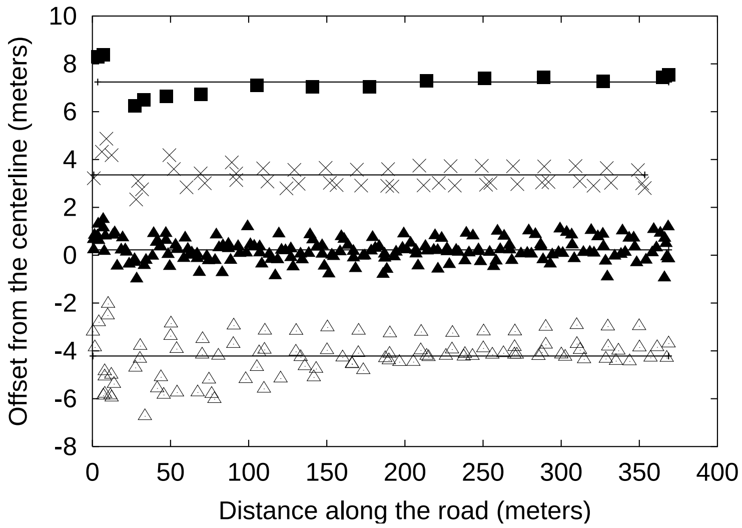

Figure 4.4: COP-KMEANS output for the sample road segment in Figure 4.1; lane centers are marked by solid lines

Figure 4.4 shows the correct partition of data points into lanes, according to the labels we obtained from the drivers. In addition, the final cluster (lane) centers appear as solid lines. Compare to Figure 4.2, which is the output generated by the k-means algorithm.

4.3.1 Lane Change Detection

The trace contiguity heuristic applies only in the absence of lane changes. If a driver changes lanes, we need to break that trace into a set of sub-traces so that no trace spans multiple lanes. This is something of a chicken-and-egg problem, since we do not yet know where the lane boundaries are, but we need to be able to detect lane changes! Since we are working with domain knowledge in the form of heuristics, however, we will content ourselves with an approximation method for detecting lane changes.

We define a *lane change* as a contiguous sequence of points $d_1, \ldots, d_m$, all generated on the same traverse, whose y feature values define a monotonic sequence such that $|d_1.y - d_m.y| >$ TOLSEP. TOLSEP is the maximum separation tolerance allowed for points within a single lane. This value is somewhat road-dependent; we used 2.0 meters for our experiments with the I-280 data.

The points in a lane change sequence $d_1, \ldots, d_m$ do not belong to a lane. They are transitional points signaling motion from one lane to another. Because it is not possible to unambiguously assign them to a lane, it is also not possible to assign them to a cluster. Therefore, after identifying a lane change sequence, we remove all of the points in that sequence from the data set for clustering purposes. This also severs the trace contiguity connection from the points that occur before the lane change with those that occur after the lane change (in terms of distance along the road segment). For example, Figures 4.2 and 4.4 have the lane change points removed.

4.3.2 Accuracy of Lane-finding Constraints

The constraints we used for our approach to this problem are, as previously noted, generated from heuristics. For example, the trace contiguity heuristic's accuracy is dependent on the accuracy of the lane change detection heuristic. We selected approximate values for TOLSEP, but those values may not always be precisely correct, due to noise in the GPS data and natural variations in road construction. Therefore, the constraints generated by these heuristics may not be completely reliable. This is important because we are treating each constraint as a hard restriction on the set of possible solutions. As we will see, the constraints are "reliable enough" to provide very high performance on this task. Nevertheless, it is important to evaluate the accuracy of the constraints, since that accuracy will affect the accuracy of our output. In the next section, we will accompany our discussion of experimental results with an analysis of the accuracy of the input constraints.

4.4 Experimental Results

This section describes the results we obtained when applying k-means and COP-KMEANS to the same set of 20 different road segments. In each case, the algorithms were required to select the best value for k. We first describe two important implementation details: how cluster centers are represented for this problem, and how to automatically select the best value for k. We then discuss our experiments and observe that COP-KMEANS, which has access to the heuristic domain knowledge described above, consistently performs much better than the regular k-means algorithm does. Finally, we analyze the errors made by COP-KMEANS and discuss the reliability of the heuristic constraint sets.

4.4.1 Cluster Center Representation

To better analyze performance in this domain, we modified the cluster center representation. The usual way to compute the center Ct_i of cluster C_i is to average all of its constituent points in the following way:

$$Ct_i = \frac{1}{|C_i|} \sum_{d \in C_i} d \tag{4.1}$$

where each d is a vector of feature values. Thus, the center of a cluster is itself a point; in this case, Ct_i would have two values and be represented by $(Ct_i.x, Ct_i.y)$. There are two significant drawbacks to using that representation for this problem. First, the center of a lane is a point halfway along its extent, which commonly means that points inside the lane but at the far ends of the road appear to be extremely far from the cluster center. Second, and more significantly for any practical use of this method, applications that make use of the clustering results need more than a single point to define a lane.

Instead, we represented the center of each lane cluster with a line segment parallel to the road centerline (see Figure 4.5). We express this lane representation with a tuple $\langle l, r, y \rangle$, which indicates the x coordinates of the left (l) and right (r) ends of the lane as well as the (constant) centerline offset of the lane, y:

$$Ct_i.l = \min_{d \in C_i} d.x \tag{4.2}$$

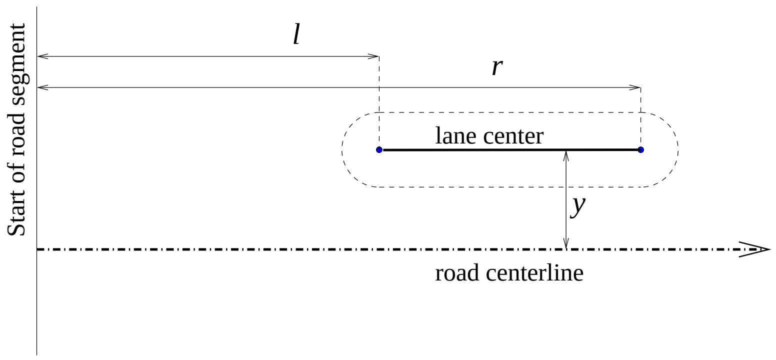

Figure 4.5: Representation of a cluster (lane) center. l and r are the x coordinates of the left and right ends of the lane, and y is the y offset of the lane from the centerline. Points that fall anywhere on the dashed line are considered equally distant from the center of the cluster.

$$Ct_i.r = \max_{d \in C_i} d.x \tag{4.3}$$

$$Ct_i.y = \frac{1}{|C_i|} \sum_{d \in C_i} d.y \tag{4.4}$$

Thus, the center of the cluster is a line segment from $(Ct_i.l, Ct_i.y)$ to $(Ct_i.r, Ct_i.y)$. This formulation more accurately models what we conceptualize as "the center of the lane," provides a better basis for measuring the distance from a point to its lane cluster center, and also provides useful output for other applications. To ensure a fair comparison, both the basic k-means algorithm and COP-KMEANS make use of this lane representation (for this problem).

K-means uses variance as its objective function, which relies on the ability to calculate the distance from a point to its assigned cluster. For two-dimensional data, the distance between a point d and a cluster C_i would be calculated as

$$dist(d, C_i) = \sqrt{(d.x - Ct_i.x)^2 + (d.y - Ct_i.y)^2}. \tag{4.5}$$

Due to our modification to the cluster center representation, we instead make use of the following method for calculating distance:

$$dist(d, C_i) = \begin{cases} \sqrt{(d.x - Ct_i.l)^2 + (d.y - Ct_i.y)^2} & d.x < Ct_i.l \\ \sqrt{(d.x - Ct_i.r)^2 + (d.y - Ct_i.y)^2} & d.x > Ct_i.r \\ |d.y - Ct_i.y| & \text{otherwise.} \end{cases} \tag{4.6}$$

This is a simple extension of the Euclidean distance to handle a line rather than a point. If d is to the left of the cluster's left endpoint, then we calculate the Euclidean distance from d to $(Ct_i.l, Ct_i.y)$. If d is to the right of the cluster's right endpoint, we calculate the distance from d to $(Ct_i.r, Ct_i.y)$. Otherwise, we calculate the vertical distance from $d.y$ to the cluster center $(Ct_i.y)$. For example, in Figure 4.5, any points that fall on the dashed line are considered equidistant from the cluster center.

It is interesting to note that even with this more expressive cluster center representation, the k-means algorithm continues to create very spherical clusters. In the absence of constraints, the k-means algorithm has no incentive to "grow" its clusters horizontally and to take advantage of this representation. Therefore, its greedy search never encounters solutions with very elongated clusters. In contrast, when clustering with constraints, the constraints provide exactly that kind of motivation.

4.4.2 Selecting k

In these experiments, both algorithms were required to automatically select the best value for the number of clusters, k. Each algorithm generated a partition for each value of k, from 1 to 5, and then selected the partition (and thus, the value of k) that it considered to be the best. In this problem domain, we know that there is a reasonable upper limit on k; in particular, the roads we were analyzing were all highway roads with a maximum of four lanes per road segment. Therefore, we can easily bound the possible range of values for k for this problem.

To enable the algorithms to select their best output partition, we used a second measure that trades off cluster cohesiveness against simplicity (in this case, measured as the number of clusters). More precisely, we define the *quality* of a partition by calculating the average squared distance from each point to its assigned cluster center and penalizing for the complexity of the solution:

$$quality(C_1, \ldots, C_k) = \frac{1}{n} \left(\sum_{d \in D} dist(d, C_d)^2\right) \cdot k^2. \tag{4.7}$$

The goal is to minimize this value. Note that this measure differs from the objective function used by k-means and COP-KMEANS while clustering (variance). In the language of Jain and Dubes (1988), quality is a relative criterion, while variance is an internal criterion. An internal criterion (objective function) allows an algorithm to select between different potential solutions generated when using the same set of input parameters. A relative criterion allows an algorithm to select between solutions generated on different runs, with different parameters (in this case, different values for k).

In the lane finding domain, the problem of selecting k is particularly challenging due to the large amount of noise in the GPS data. Each algorithm performed 30 randomly-initialized trials with each value of k (from 1 to 5). COP-KMEANS selected the correct value for k for all but one road segment, but k-means *never* chose the correct value for k (even though it was using the same method for selecting k). This is due to its strong bias towards spherical clusters; the most spherical cluster is obtained by assigning all points to a single cluster.

4.4.3 Results on I-280 Data

Table 4.1 presents accuracy results for both algorithms over 20 road segments. Accuracy was calculated using the Rand metric (see Section 3.5.1) and indicates the amount of agreement between the output partition (set of lanes) and the true lane labels. These results represent overall accuracy rather than held-out accuracy, since determining the right set of

Table 4.1: Lane finding performance (Rand index). Each algorithm was given the true value for k.

Segment (size)	K-means	COP-KMEANS	Constraints alone
1 (699)	49.8	100	36.8
2 (116)	47.2	100	31.5
3 (521)	56.5	100	44.2
4 (526)	49.4	100	47.1
5 (426)	50.2	100	29.6
6 (503)	75.0	100	56.3
7 (623)	73.5	100	57.8
8 (149)	74.7	100	53.6
9 (496)	58.6	100	46.8
10 (634)	50.2	100	63.4
11 (1160)	56.5	100	72.3
12 (427)	48.8	96.7	59.2
13 (587)	69.0	99.8	51.5
14 (678)	65.9	100	59.9
15 (400)	58.8	100	39.7
16 (115)	64.0	76.6	52.4
17 (383)	60.8	98.9	51.4
18 (786)	50.2	100	73.7
19 (880)	50.4	100	42.1
20 (570)	50.1	100	38.3
Average	**58.0**	**98.6**	**50.4**

constraints is part of the problem; the constraints are not artificially generated from the true labels. The number of data points for each road segment is also indicated. These data sets are much larger than the UCI data sets discussed in Section 3.5, providing a chance to test the ability of the algorithm to scale to larger problems.

As shown in Table 4.1, COP-KMEANS consistently outperformed the unconstrained k-means algorithm, attaining 100% accuracy for all but four data sets and averaging 98.6% overall. The unconstrained version performed much worse, averaging 58.0% accuracy. The clusters the latter algorithm produces often span multiple lanes and never cover the entire road segment lengthwise, even though it was given the correct value for k. As discussed above, lane clusters have a very specific shape: they are greatly elongated and usually oriented horizontally (with respect to the road centerline). Yet even when the cluster center is a line rather than a point, k-means seeks compact, usually spherical clusters. Consequently, it does a very poor job of locating the true lanes in the data.

Figure 4.2 showed the output of the regular k-means algorithm for road segment 15. The correct value of k was specified to obtain this output (without it, the algorithm selects $k = 1$). COP-KMEANS achieves 100% accuracy on this road segment (and it selects the proper value of k). The correct partition, which is identical to the COP-KMEANS output, was shown in Figure 4.4.

The final column in Table 4.1 is a measure of how much is known after generating the constraints and before doing any clustering. It shows that an average accuracy of 50.4% can be achieved using the background information alone. This demonstrates that neither general similarity information (k-means clustering) nor domain-specific information (constraints) alone perform very well, but that combining the two sources of information effectively (COP-KMEANS) can produce excellent results.

4.4.4 Analysis of COP-KMEANS Errors and Constraint Accuracy

An analysis of the errors made by COP-KMEANS on the lane-finding data sets shows that each mistake arose for a different reason. For road segment 12, the algorithm incorrectly included part of a trace from lane 4 in lane 3. This appears to have been caused by noise in the GPS points in question: they are significantly closer to lane 3 than lane 4. The same thing happened for a single point in road segment 13. Alternatively, the labels for this trace may themselves be noisy. On road segment 16, COP-KMEANS chose the wrong value for k (it decided on three lanes rather than four). This road segment contains relatively few data points (115, compared to the average of 534), which possibly contributed to the difficulty. Finally, for road segment 17, a small part of one trace was assigned to the wrong lane due to an incorrect lane-change detection. Since COP-KMEANS made so few errors on this data set, it is not possible to provide a more general characterization of their causes.

We mentioned above that quality of the output clusters obtained from COP-KMEANS will be sensitive to the reliability of the constraints it is given. In addition, since these constraints were generated heuristically, we may not expect them to be completely reliable. Consequently, we evaluated the accuracy of the constraints that were generated for each road segment by calculating how well they agreed with the true lane labels. We define two sets of *accurate* constraints, $Acc_=$ and $Acc_{\neq}$, which are subsets of $Con_=$ and $Con_{\neq}$ respectively, as follows:

$$Acc_{=} = \{(a, b) \in Con_{=} \mid a.lane = b.lane\} \quad (4.8)$$
$$Acc_{\neq} = \{(a, b) \in Con_{\neq} \mid a.lane \neq b.lane\} \quad (4.9)$$

where $a.lane$ is the true lane label for point a. We then calculate constraint accuracy:

$$\text{C-acc} = \frac{|Acc_{=}| + |Acc_{\neq}|}{|Con_{=}| + |Con_{\neq}|}. \quad (4.10)$$

For all but one of the road segments presented in Table 4.1, C-acc is 100.00%. The exception is segment 13, where the constraint set is 99.93% accurate. The overall constraint accuracy for the entire collection of road segments, which includes almost 2 million constraints, is 99.99%. In this case, our heuristic domain knowledge is very accurate; it is almost entirely consistent with the true data labels. In addition, for the single road segment with slightly inaccurate constraints, COP-KMEANS was still able to achieve 99.8% performance.

In fact, it is instructive to evaluate the sensitivity of COP-KMEANS to the accuracy of the input constraints. For example, if we modify the lane change detection heuristic to use TOLSEP = 3.0 meters, the accuracy of the constraints for segment 13 drops to 99.86%. Impact on COP-KMEANS performance is minimal, however; it drops from 99.797% to 99.795%.

4.4.5 Is K-means a Straw Man?

It might be argued that k-means is simply a poor choice of algorithm for this problem. For example, there are versions of k-means which are biased towards elliptical, rather than spherical, clusters, which might be expected to have better performance on this problem. However, selecting a different algorithm for each problem encountered has its drawbacks. First, we lose any advantages obtained by having a general solution to the clustering problem. For example, we must have access to a comprehensive suite of clustering algorithms, each with their own bias, and each of which must be maintained. Second, and more importantly, selecting the proper algorithm for a given problem requires specialized knowledge about clustering, each algorithm's bias, and the ability to decide what kind of bias is appropriate for the problem. Finally, a specialized algorithm with a suitable bias simply may not be available. We have access to a k-means algorithm that prefers ellipsoidal clusters, but we do not have one that prefers, for example, hyper-rectangular clusters. If that is the most appropriate bias for a given problem, then we must implement a new version of k-means to satisfy it. We therefore believe that a better alternative is to maintain a *general* clustering algorithm that can accommodate domain-specific knowledge input to *automatically specialize* by adapting or overriding its built-in bias.

In addition, the marked improvements we observed with COP-KMEANS suggest another advantage of this method: algorithm choice may be of smaller importance when constraints based on domain knowledge are available. For this task, even a poorly-performing algorithm can boost its performance to extremely high levels. In essence, it appears that domain knowledge can make performance less sensitive to which algorithm is chosen. This is good news for anyone wishing to make use of unsupervised learning methods without having the background to make informed decisions about which algorithm should be used.

4.4.6 Comparison to agglom

Rogers et al. (1999) previously experimented with a clustering approach that viewed lane finding as a one-dimensional problem. Their algorithm (agglom) only made use of the centerline offset of each point, ignoring the position of a point along the road segment. They used a hierarchical agglomerative clustering algorithm that terminated when the two closest clusters were more than a given distance apart (which represented the maximum width of a lane).

This approach is quite effective when there are no lanes that merge or split in a road segment, i.e., each lane continues horizontally from left to right with no interruptions. For the data sets listed in Table 4.1, their algorithm obtains an average accuracy of 99.4%, which is slightly higher than the results obtained with COP-KMEANS.[2] This is largely due to the impact of a single road segment (segment 16) for which COP-KMEANS chose the wrong value for k, as previously described.

However, all of these data sets were taken from a highway, where the number of lanes is constant over the entirety of each road segment. In cases where there are lane merges or splits, the one-dimensional approach may have difficulty representing the extent of a lane along the road segment. We expect that when processing data obtained from a larger variety of roads, including segments with lane merges and splits, COP-KMEANS and agglom may have performance that differs more greatly.

4.5 Summary

In this chapter, we applied the COP-KMEANS algorithm to a real-world problem. We demonstrated how to encode heuristic domain knowledge as a set of instance-level constraints and reported on the significant performance gains they provide. COP-KMEANS, which combines generic similarity information with problem-specific domain knowledge, is able to significantly outperform either source of information in isolation. We also analyzed the accuracy of these heuristic constraints and found them to be highly accurate, which contributes to the performance of COP-KMEANS. In the next chapter, we will see that COP-COBWEB also achieves performance increases when applied to a (different) real-world problem.

[2] A maximum merging distance of 2.5 meters was specified.