Semantic Segmentation: Methods, Implementation, and Comparative Analysis

An Implementation Study for COSC 544/444

Christian Baker-Davidson
Master of Applied Science
University of British Columbia
Kelowna, Canada
davidsonchristian@outlook.com

Aidan Morris
B.Sc. Computer Science Honours
University of British Columbia
Kelowna, Canada
aidanmorris810@gmail.com

Beth Ralston
B.Sc. Computer Science
University of British Columbia
Kelowna, Canada
bethmralston@gmail.com

Rory Hazlett
B.Sc. Computer Science
University of British Columbia
Kelowna, Canada
roryhazlett42@hotmail.com

***Abstract*—This paper presents a comprehensive study on semantic segmentation with a focus on enhancing object detection in urban driving environments, particularly for autonomous vehicles. We evaluate and compare multiple algorithms to determine their effectiveness in achieving accurate segmentation. Additionally, we address key implementation challenges and discuss the strategies employed to overcome them.**

***Index Terms*—Segmentation, Superpixels, SLIC, SVM, Object Detection.**

I. Introduction

Semantic segmentation is an important task in the field of computer vision, particularly within applications such as autonomous driving, where systems must interpret and understand complex visual scenes in real time with context. One area where this capability proves useful is in object detection and scene understanding, which enables self-driving vehicles to identify and respond to various elements in their environment, such as pedestrians, vehicles, and static objects.

The target problem for this project is to develop a semantic segmentation-based method to detect and accurately segment cars in urban driving environments using traditional computer vision techniques. Initially, our goal was to detect both pedestrians and vehicles, prioritizing pedestrian safety. However, as development progressed, we encountered significant challenges with our superpixel segmentation approach, particularly in consistently and accurately outlining the shapes of pedestrians in our dataset. These inaccuracies led us to narrow the focus of the project to car detection only, as superpixel segmentation demonstrated higher accuracy and reliability when applied to more well-defined objects like vehicles.

Solving this problem is crucial to improving the reliability of traditional computer vision methods in autonomous systems and to better understand what challenges arise when doing so and how to overcome them. A solution that combines computational efficiency with accurate car detection can contribute to safer and more accessible autonomous vehicles, especially in scenarios where lightweight or real-time processing is needed.

II. Literature Review

A. K-Means

K-Means is a method of clustering like entities to groups for further analysis [1]. The number of groups of entities must first be identified by the user. The number of groups needs to be identified carefully. If too many groups are used within the algorithm many entities of a single group may be split into multiple. If not enough groups are used, entities from two groups may be combined into a single group. Either of these cases would result in false positives or false negatives affecting the overall performance of the algorithm.

Initially points from a data set are randomly assigned to a fixed number of clusters determined by the user. The following two steps repeat until the change in magnitude position of previous to current centroids is below a predefined threshold. The cluster's centroid is then determined by taking the mean value of each of the entity's parameters. A distance function is then used to determine the distance of each entity to the closest cluster center. Figure 1 is an example of a distance function that can be used on entities with (x, y) parameters. This example represents a converged system with two clusters.

B. Simple Linear Iterative Clustering (SLIC):

Simple linear iterative clustering (SLIC) is a method for grouping pixels within an image into regions called superpixels [2]. These superpixels represent clusters of pixels that are similar in color and spatial location. Clustering pixels into superpixels can reduce the number of elements that need to be

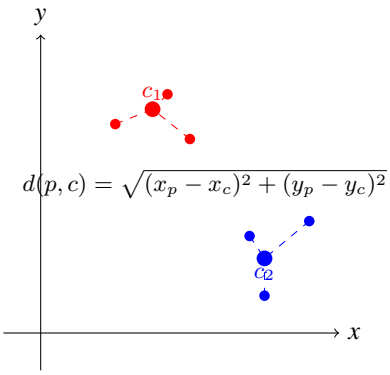


Fig. 1: Euclidean Distance in k-Means Clustering

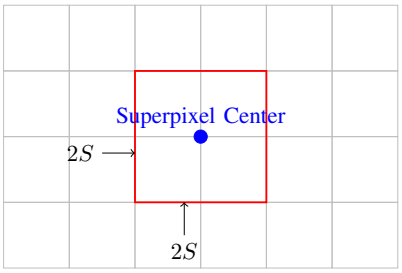
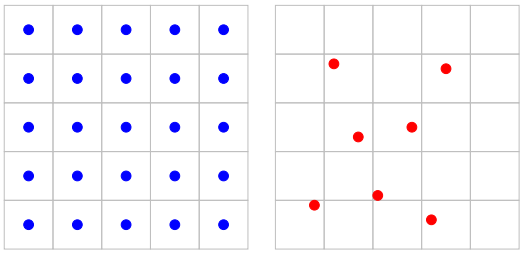


Fig. 2: Illustration of the $2S \times 2S$ search window in SLIC. The blue dot represents the superpixel center. Each square represents a $S \times S$ portion of the image.



(a) SLIC - Regular Grid Initialization (b) K-Means - Random Initialization

Fig. 3: Comparison of initial cluster placements: SLIC (left) uses a regular grid, while K-Means (right) starts with random centers.

processed by several orders of magnitude. The method is based on a modified version of K-Means. Instead of comparing every pixel to every possible cluster center, SLIC speeds up this process. SLIC only compares pixels within a limited window around each cluster center making the number of computations much smaller. By only focusing on the small region, SLIC ensures the distance to clusters that could not possibly be the closest to a pixel do not need to be computed. In addition to this optimization, SLIC also modifies k-means by using color information in its distance calculations.

The algorithm uses the CIELAB color space rather than RGB. Unlike RGB values, the CIELAB color space is designed to approximate human vision. The CIELAB color space splits the colors into lightness(L) and color channels (a and b). In the SLIC algorithm, using CIELAB allows for better differentiation between similar colors. CIELAB is required for SLIC segmentation processes, because of this property. The SLIC algorithm has a parameter m which controls how important spacital or color information is. Larger values of m result in more spatially compact pixels. The value of 'm' therefore needs to be tuned based on the algorithm's specific use case.

The SLIC algorithm initially converts the input image into CIELAB if it is not already in that color space, for reasons previously mentioned. The cluster centers are then initialized with an even spread across the image. They are placed in a grid arrangement with a spacing of S pixels. S is calculated based on the number of pixels in the image and the desired number of super pixels. During each iteration each pixel is assigned to its nearest cluster center based on the distance measure. Then, the cluster centers are updated by taking the average distance and color or the pixels assigned to the cluster. Again, only pixels in the $2 \times S$ window are considered The SLIC algorithm iterates until the sum squared distance between the clusters and their previous locations is below a threshold. This usually happens within 10 iterations.

Once the SLIC algorithm completes, [2] suggests that some post processing can improve the accuracy of the algorithm. Sometimes single pixels might not be connected to the main group of cluster pixels. These pixels are called 'orphaned' pixels. Once found, these disconnected pixels can be connected to the nearest cluster center.

The SLIC algorithm is designed to be a much more efficient version of k-means. It is both more computational and memory efficient when compared to the k-means algorithm. It achieves this by having a linear time complexity O(n) because of its use of a $2 \times S$ window. This drastically reduces the computational requirements compared to that of k-means O(mn). Where m is the number of pixels in an image and n is the desired number of superpixels. In addition to SLIC's memory efficiency, it also is much more memory efficient. SLIC only has to store O(n) floating point numbers for each cluster and O(m) integers for each pixel. This contrasts with the standard k-means implementation that might require more extensive memory to store the full distance matrices or additional data structures.

Unlike basic k-means, SLIC incorporates a parameter (m) that explicitly allows users to control the compactness of pixels. This means that depending on the needs of the task, the segmentation can be fine-tuned easily. In addition to this, SLIC initializes cluster centers on a grid rather than at random positions. This allows superpixels to be a much more even size.

SLIC's parameter (m) can be very beneficial, but this could also be a problem. If m is not properly tuned, it would result in pixels that are too compact, or too irregularly shaped. This differs from k-means as it does not have this parameter and thus does not have this particular problem. While the $2 \times S$ window search strategy is greatly computationally beneficial, it too could be a problem. Images with irregular patterns or complex boundaries could result in suboptimal

clustering. SLIC's clusters are initialized on a grid pattern. If the initial cluster locations do not match the images structures well, it could lead to suboptimal superpixels. More iterations, or additional post processing might be required to achieve optimal segmentation.

C. Support Vector Machines (SVM):

Support Vector Machines (SVMs) have been extensively employed in semantic segmentation tasks due to their strong theoretical foundations, excellent generalization capabilities, and effectiveness in handling high-dimensional data [3]. An SVM is a supervised machine learning algorithm that constructs an optimal hyperplane to separate data points belonging to different classes with maximal margin, thus facilitating robust classification even when the data distribution exhibits overlap or significant noise. A visualization of this can be seen in Figure . In semantic segmentation contexts, SVMs commonly utilize kernel methods, such as Radial Basis Function (RBF), polynomial, or linear kernels, to map input features into higher-dimensional spaces, allowing for effective separation of complex, nonlinear feature distributions [4].

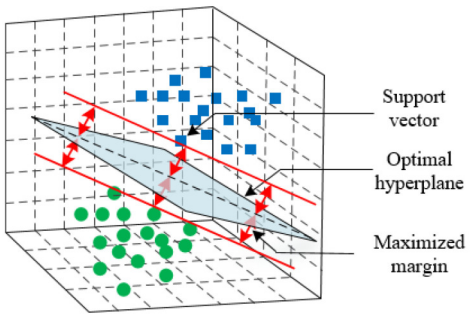


Fig. 4: Visualization of SVM forming a hyperplane through data.

Typically, when SVMs are applied to superpixel-based semantic segmentation, the feature vectors extracted from each superpixel—such as color histograms, texture measures, spatial statistics, and centroid locations—are input into the classifier. The SVM then assigns labels to superpixels based on these extracted features, which significantly reduces computational complexity compared to pixel-wise classification methods. The result is a substantial improvement in efficiency without dramatically sacrificing segmentation accuracy, particularly suitable for applications with moderate-sized datasets or limited computational resources [5].

One of the primary advantages of SVM-based segmentation methods is their robust generalization capability, especially valuable in scenarios with limited or noisy labeled data. They exhibit stable performance across varied datasets, owing to their margin-maximizing principle. Moreover, SVMs are computationally efficient during inference, making them particularly attractive for real-time or near-real-time segmentation tasks. Additionally, because SVMs depend explicitly on engineered feature sets, they offer interpretable classification outcomes and facilitate targeted feature engineering.

Despite their strengths, SVMs require careful selection and tuning of kernel functions and regularization parameters to achieve optimal performance, a process that can be computationally demanding and dependent on domain-specific expertise. Furthermore, the quality of segmentation heavily relies on the effectiveness of the extracted features; thus, poor feature selection or insufficient feature extraction can severely impact classification accuracy. Unlike deep learning methods, SVMs do not inherently capture high-level contextual or semantic information, potentially limiting their ability to discriminate between classes with subtle visual differences or complex spatial relationships. Consequently, SVM approaches may struggle in highly diverse or dynamically changing environments without additional contextual modeling or sophisticated feature enhancement techniques.

III. System Design

Our system, as seen in Figure 5, is made up of 5 basic modules: Preparation of data, SLIC feature extraction, K-means unsupervised classification, preparing training data, training and testing SVM supervised classification.

The first module downloads the dataset and organizes the ground truth labels to go with the images. The training images are then inputted into the SLIC feature extraction module which first converts the images into the LAB color space before sending it to the SLIC algorithm. As well, the SLIC algorithm takes in the parameters where the number of superpixels is 1024, the max number of iterations is 15, a threshold of 200 and whether to run it on the CPU or the GPU. The number of superpixels was chosen to be 1024 based on several different numbers and 1024 resulting with the best results. The max number of iterations and threshold were chosen based on the recommendation of the algorithm description. A connected components algorithm is also used to enforce connectivity and ensure that the SLIC clusters are consistent to help with the comparison to the ground truth image. The SLIC algorithm produces a matrix of cluster ids where each matrix point corresponds to a pixel in the image and an array of superpixel clusters which contains the information for each superpixel. The cluster information includes the cluster id, the center x,y value of the cluster and the average LAB values of the cluster. The system then breaks into two streams, an unsupervised classification system and a supervised classification system.

The results from the SLIC algorithm, along with the ground truth labels are passed into the training data preparation module. This module extracts the information from the superpixel clusters and the ground truth labels to create X and Y train. As well, linear binary patterns and Gabor filters were used to capture texture data. X Train contains the texture data, LAB mean, cluster centroid location, standard deviation of SLIC cell and the median value of SLIC cell. Y Train is the dominant label of the masked ground truth matrix using the SLIC cluster location as the mask. X and Y train are then passed into the SVM supervised classification training algorithm which uses a pipeline creation algorithm to create a SVM supervised classification system pipeline based on the

training image samples. When creating the pipeline a RBF kernel is used in the model fitting to handle non-linear decision boundaries by mapping the X train to a higher dimensional space. Regularization is also used during model fitting to help with minimizing the misclassification. The pipeline is then passed into the SVM testing module which uses the newly created pipeline to predicate the labels of the designated test images. An error percentage algorithm is run to compare the ground truth labels to the predicted labels to calculate the error rate of the SVM algorithm. The testing SVM supervised classification module finishes by displaying the final segmented and labeled images compared to the ground truth images.

K-Means is used for the unsupervised classifier and the module takes in the array of superpixel clusters from the SLIC module. The K-means clustering algorithm uses the LAB value of each superpixel cluster and uses K-means clustering to create mega clusters containing the SLIC superpixel clusters. A percentage algorithm is then used to compare how the mega clusters are made up to the ground truth pixel labels and to calculate the percentage that the dominant label makes up the mega cluster.

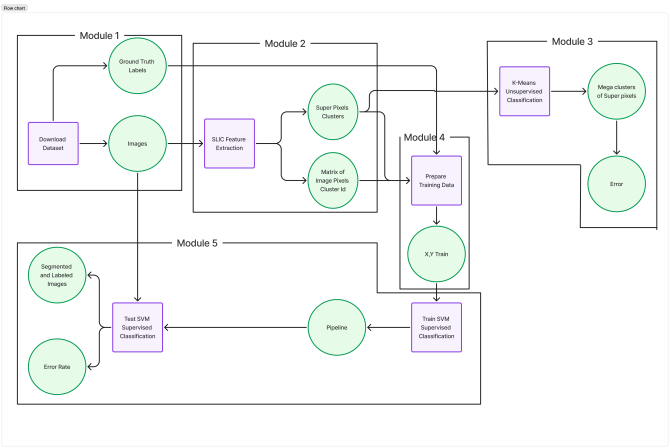


Fig. 5: The pipeline of data within our systems architecture.

IV. Results

For the results section, it is important to realize we utilized a Support Vector Machine (SVM) classifier equipped with a Radial Basis Function (RBF) kernel and regularization parameters to perform semantic segmentation as previously discussed. For this reason, slic clusters are labelled with their dominant label to encapsulate the entire object as well as speed up SVM classification as we are classifying the images in chunks. As will be demonstrated within this section, this leads to generally favourable outcomes with minimal processing times, however, occasionally it causes the prediction to be entirely inaccurate for a chunk. This integration of SLIC with SVM classification was able to achieve effective pixel-wise labelling by leveraging both spatial coherence and feature-based discriminability.

The data we utilized was sourced from the Cityscapes data set and contained over 3000 semantically labelled images [6]. The training data encompassed a comprehensive feature set derived from multiple image channels, systematically capturing critical statistical attributes. Specifically, each superpixel's feature vector included the mean, standard deviation, median, minimum, maximum pixel intensity values computed across all input channels as well as linear binary pattern and Gabor filters to capture texture. These statistical measures ensured robust representation of each superpixel's intrinsic characteristics. Moreover, spatial attributes, including the centroid coordinates of each superpixel, were incorporated to facilitate the classifier's ability to exploit spatial context in the labeling process. Additionally, color information was represented through mean LAB color values extracted for each superpixel, providing colorimetric context beneficial in distinguishing visually similar regions.

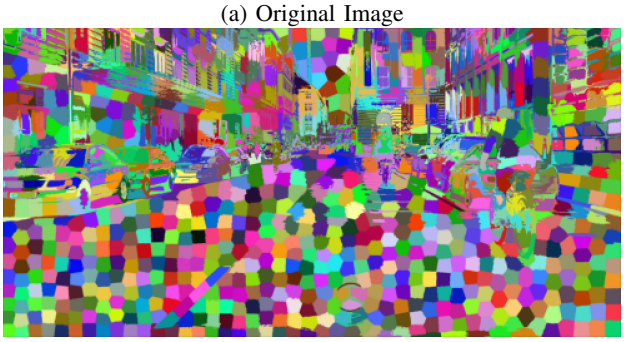
Ground truth labels were systematically generated through a process involving the identification of the dominant label within each SLIC superpixel. This was accomplished by employing the locations of the SLIC clusters as spatial masks on manually annotated reference images. The most prevalent ground truth label within each masked superpixel area was assigned as the superpixel's primary classification, establishing a direct correspondence between the classifier's predictions and meaningful semantic categories.

K-Means was used to group superpixels based on like pixels based on like Lab values. K-means is an unsupervied classification method, meaning it could not be direclty compared with the SVM classification method. Instead K-Means was used to group super pixels into clusters of clusters, or "mega" clusters. The accuracy of the classification was then measured. The ground truth value of each pixel is used to measure the largest group of correct pixels within an mega cluster. The ratio of correct pixels to total number of pixels is the accuracy of that given mega cluster.

V. Discussion

Initial tests of the SLIC segmentation algorithm demonstated that computation time would be greater than we would have liked. As seem in Table I found in Appendix A. The computation time for a single image lied somewhere in the neighborhood of $\sim$36 seconds in the cpu implentation. In order to optimally train our SVM implmentation many images needed to be segmented with many different sets of parameters. We determined that additional performance would be needed. The CPU implmentation was already full optimized, therfore the algorithm needed to be parallelized. CUDA was used to parallelize the algorithm on the GPU allowing for much better scalability with larger images and datasets. We experienced an almost $200\times$ speed up in computation time for each iteration. This took the time to compute an image from $\sim$36 second to only $\sim$174 ms. These numbers found using both a large image (2048x1024px) and with a large number of superpixels (1024).

(a) Original Image



(b) Segmented image overlayed on Original Image

Fig. 6: Superpixel segmented image compared to original image. 1024 superpixels, 15 max iterations, m = 4, and threshold of 200 were used to produce these results. Random colors have be assigned to differentiate between the different superpixels.

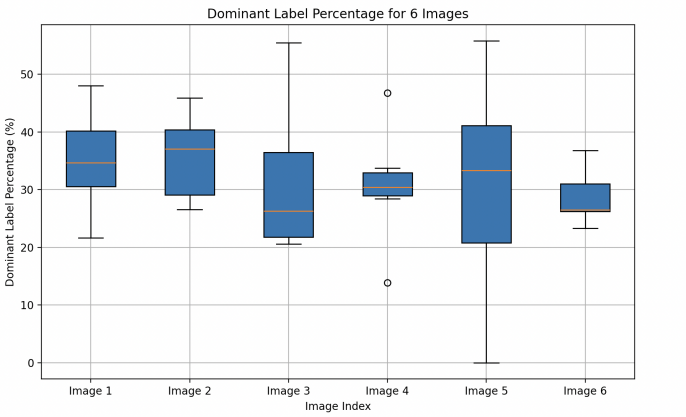
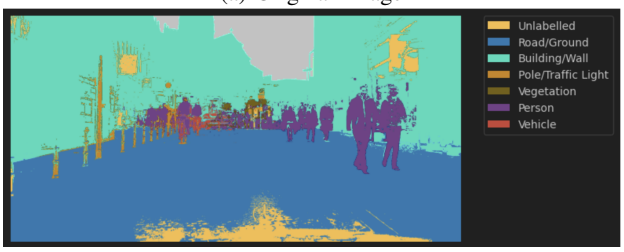


Fig. 7: The spread of the dominant label percentage in 6 images using k-means.

The Table II in Appendix A demonstates how the SLIC algorithm scales with varying image sizes. The computation times scales linearly with the number of pixels present in the image. An image half the size (a forth the number of pixels) has a forth the computational time. This demonstates slic's window optimization nicly.

Table III in Apprendix A contains the average count of the dominant label per sum of labels for each mega cluster within an image. As seen in the table the most accurate number of

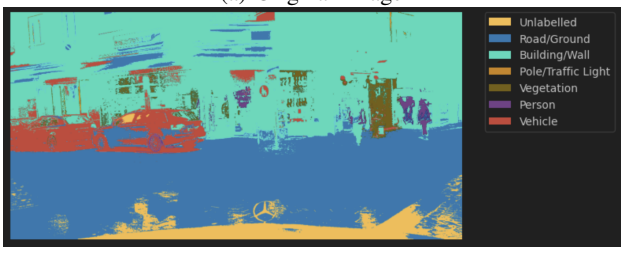
(a) Original Image



(b) Predicted Classes for the Image

Fig. 8: Semantically classified image compared to original image with an excellent outcome with an accuracy of 90.78%.

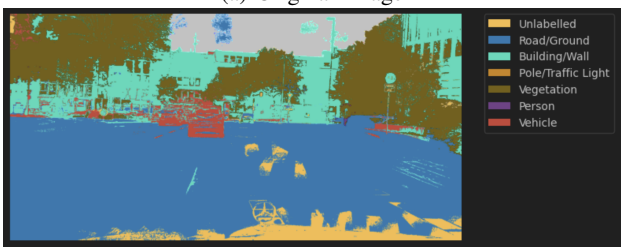
(a) Original Image



(b) Predicted Classes for the Image

Fig. 9: Semantically classified image compared to original image with a very poor outcome of 47.55%.

(a) Original Image



(b) Predicted Classes for the Image

Fig. 10: Semantically classified image compared to original image with an average outcome of 82.73%.

mega clusters is when the amount of mega clusters is set to 32 where it reaches a maximum of 42% accuracy. This is important because the dominant label is what each mega cluster is labeled as, therefore this algorithm will have low accuracy as each mega cluster is classified poorly.

The performance of the proposed segmentation methodology was rigorously evaluated on a selected test set of 100 diverse images, representative of various complexities and scene conditions. Quantitative analysis revealed an overall pixel-level segmentation accuracy of 80.9%, indicating a considerable capacity of the SVM classifier combined with SLIC-based segmentation to discriminate effectively among several semantic classes. A good outcome of this can be seen in Figure 8.

Further qualitative evaluation was conducted to understand performance trends and limitations. The classifier demonstrated notable success in accurately delineating the external boundaries of vehicles, achieving satisfactory segmentation results particularly around vehicle outlines. However, detailed inspection identified occasional issues with accurately labelling the midsection areas of vehicles as seen in Figure 9. These interior regions, which tend to have homogeneous textures and color distributions, posed significant challenges, suggesting the existing statistical and color features were insufficient to differentiate these interior portions clearly.

While the obtained average pixel-wise accuracy of 80.9% might initially appear modest, visual inspection of segmented images reveals a notably high level of practical effectiveness. The numerical accuracy alone does not fully convey the strengths of the segmentation approach employed. Crucially, the classification errors observed provide significant practical advantages, manifesting predominantly within semantically related categories. For instance, the road surface was consistently and accurately segmented, clearly differentiated from surrounding elements such as buildings, vehicles, and pedestrians. Misclassifications typically occurred among similar obstacle categories—for example, human figures might occasionally be classified incorrectly as vehicles or buildings, yet they were rarely confused with non-obstacle categories like roads.

From the perspective of an obstacle detection system, this characteristic is particularly advantageous, as the segmentation method reliably distinguishes traversable from non-traversable regions. Consequently, even when misclassifications occur, they predominantly remain confined to obstacles, preserving the critical ability of the system to accurately identify and avoid potential hazards. Thus, despite an accuracy metric that may appear merely acceptable, the practical implications of the classifier's performance indicate a robust and highly beneficial segmentation outcome for real-world applications. An example of this can be seen in a below average accuracy classification in Figure 10.

Overall, while the results show promise in leveraging SLIC and SVM methods for semantic segmentation, they clearly indicate areas for further refinement, particularly regarding feature selection and the integration of advanced spatial-contextual cues to improve class-level discrimination.

VI. Future Work

Several promising directions have been identified to further enhance and extend the effectiveness of our semantic segmentation approach. First, optimizing feature extraction efficiency remains an important priority. Future research should explore computational strategies to reduce feature dimensionality, improve the speed of feature computation, and experiment extensively with alternative feature sets. Incorporating texture-based features, advanced edge detection metrics, and higher-order spatial relationships could substantially enhance classifier discriminability and performance.

Addressing existing classification inaccuracies, especially concerning human figures occasionally misclassified as vehicles or buildings, represents another crucial area for improvement. This problem may be mitigated through targeted feature engineering, specifically tailored to distinguish subtle yet discriminative visual characteristics unique to pedestrians and vehicles. Additionally, introducing context-sensitive classification methods, possibly through deep learning-based semantic or instance segmentation approaches, could further help alleviate such misclassification errors.

Finally, the expansion of classification capabilities beyond the current seven broad semantic categories is an important avenue for future investigation. Refining the segmentation approach to differentiate more granular subclasses, such as distinguishing among various vehicle types including cars, vans, motorcycles, trucks, and buses, would significantly enhance its utility. Accomplishing this would likely require an expanded dataset with more detailed annotations and poten-

tially the adoption of more advanced multi-class or hierarchical classification frameworks.

By pursuing these improvements, future studies have the potential to greatly enhance both the accuracy and practical applicability of semantic segmentation in diverse real-world scenarios.

REFERENCES

[1] Dr. Mohamed Shehata. Course introduction, 2025.

[2] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Süsstrunk. Slic superpixels compared to state-of-the-art superpixel methods. *IEEE transactions on pattern analysis and machine intelligence*, 34(11):2274–2282, 2012.

[3] Farid Melgani and Lorenzo Bruzzone. Classification of hyperspectral remote sensing images with support vector machines. *IEEE Transactions on geoscience and remote sensing*, 42(8):1778–1790, 2004.

[4] Preetham Kumar and B Vijayakumar. Brain tumour mr image segmentation and classification using by pca and rbf kernel based support vector machine. *Middle-East Journal of Scientific Research*, 23(9):2106–2116, 2015.

[5] Zhengqin Li and Jiansheng Chen. Superpixel segmentation using linear spectral clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[6] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3213–3223. IEEE, 2016.

APPENDIX A
ADDITIONAL EXPERIMENTAL RESULTS

This section provides additional experimental results for CPU and GPU processing times.

Image Paths						
City	**Berlin**	**Bielefeld**	**Bonn**	**Leverkusen**	**Mainz**	**Munich**
File ID	000095_000019	000000_006603	000012_000019	000053_000019	000003_016360	000385_000019
CPU Iteration Times (ms)						
Iteration #	**Berlin**	**Bielefeld**	**Bonn**	**Leverkusen**	**Mainz**	**Munich**
1	2170.86	2142.29	2145.32	2199.20	2169.88	2178.08
2	2218.03	2198.77	2203.04	2209.27	2195.35	2194.48
3	2262.79	2192.03	2239.31	2200.66	2227.41	2219.44
4	2262.61	2198.10	2257.14	2258.53	2249.18	2216.08
5	2264.32	2228.12	2282.92	2248.89	2280.71	2226.54
6	2276.45	2212.93	2294.92	2250.42	2288.57	2225.07
7	2303.07	2210.06	2306.12	2262.97	2267.70	2249.44
8	2311.49	2232.91	2299.65	2257.57	2269.30	2247.76
9	2311.85	2258.50	2320.20	2256.00	2284.38	2253.37
10	2321.14	2255.94	2322.64	2261.63	2293.20	2252.66
11	2314.81	2240.36	2313.76	2252.61	2292.47	2258.75
12	2301.62	2248.96	2322.17	2262.76	2302.27	2248.63
13	2330.65	2240.79	2316.72	2273.71	2299.28	2258.07
14	2322.37	2255.82	2323.45	2265.37	2296.29	2258.01
15	2305.26	2256.26	2315.41	2272.38	2295.07	2266.01
Total (ms)	36449.95	35524.54	36429.56	35913.23	36143.13	35729.40
Avg (ms)	2285.16	2224.79	2284.18	2248.80	2267.40	2236.83
GPU Iteration Times (ms)						
Iteration #	**Berlin**	**Bielefeld**	**Bonn**	**Leverkusen**	**Mainz**	**Munich**
1	15.81	9.89	9.79	9.99	10.00	10.00
2	15.72	9.86	9.95	9.96	9.97	9.94
3	15.60	9.88	9.89	9.94	9.95	9.96
4	15.42	9.90	9.87	9.91	9.92	9.92
5	15.38	9.91	9.86	9.89	9.91	9.93
6	15.24	10.02	9.99	10.05	10.04	10.06
7	15.30	10.10	10.02	10.12	10.14	10.15
8	15.21	10.16	10.07	10.19	10.20	10.22
9	15.20	10.19	10.10	10.22	10.23	10.25
10	15.28	10.21	10.14	10.27	10.28	10.30
11	15.25	10.23	10.18	10.31	10.32	10.33
12	15.19	10.30	10.21	10.36	10.38	10.40
13	15.22	10.35	10.24	10.41	10.42	10.45
14	15.20	10.37	10.26	10.48	10.50	10.52
15	15.18	10.40	10.27	10.53	10.54	10.56
Total (ms)	253.72	174.01	174.75	176.49	175.39	176.22
Avg (ms)	15.40	10.43	10.34	10.54	10.51	10.56
Avg Speedup Factor: 199.89×						

TABLE I: CPU and GPU Iteration Times for Image Processing. The following parameters were used: 1024 superpixels, m = 4, 15 max iterations and a threshold of 200.

Image: /berlin_000095_000019_leftImg8bit.png				
Subdivisions and Image Dimensions				
Subdivision Level	**Width**	**Height**		
0	2048	1024		
1	1024	512		
2	512	256		
3	256	128		
Iteration Times (ms) for Different Subdivisions				
Iteration	**0**	**1**	**2**	**3**
1	15.91	3.26	0.70	0.33
2	15.69	2.36	0.66	0.31
3	15.68	2.37	0.65	0.30
4	15.63	2.39	0.66	0.28
5	15.60	2.39	0.65	0.28
6	15.57	2.34	0.65	0.28
7	15.55	2.33	0.65	0.28
8	15.52	2.30	0.65	
9	15.40	2.29	0.65	
10	15.37	2.31	0.66	
11	15.35	2.31	0.65	
12	15.33	2.29	0.65	
13	15.32	2.27	0.67	
14	15.36	2.31		
15	15.37	2.36		
Total Time (ms)	**255.34**	**42.84**	**10.66**	**2.87**

TABLE II: Iteration times for different subdivision levels of the image. The following parameters were used: 1024 superpixels, m = 4, 15 max iterations and a threshold of 200.

Number of Mega Clusters	Image 1	Image 2	Image 3
2	23.18%	25.75%	29.48%
4	31.28%	29.18%	34.90%
8	34.01%	26.47%	33.32%
16	40.27%	30.47%	38.48%
32	41.96%	32.68%	37.97%
64	40.86%	31.09%	38.86%
128	33.59%	31.06%	30.88%

TABLE III: K-Means run with different number of target (Mega) clusters. Tests were run on three different images.