Unstructured Road Detection and Steering Assist Based on HSV Color Space Segmentation for Autonomous Car

1st A.A. Mahersatillah
Postgraduate Student of Department of Electrical Engineering
Hasanuddin University
Makassar, Indonesia
suradiaam18d@student.unhas.ac.id

2nd Z. Zainuddin
Department of Informatics Engineering
Hasanuddin University
Makassar, Indonesia
zahir@unhas.ac.id

3rd Y. Yusran
Department of Electrical Engineering
Hasanuddin University
Makassar, Indonesia
yusran@unhas.ac.id

***Abstract*— One of the important things in a self-driving car (SDC), also known as an autonomous vehicle (AV) is detecting the road so that it remains in the right lane. Therefore this paper aims to be able to detect roads, especially unstructured roads based on the results of the HSV color space segmentation on the road, then produce car position information from the center of the lane (center offset) which is a parameter in making the decision to move the car's steering wheel to return to the center of the lane. In marking the edge of the roadside, the method used is Hough transform based on the resulting edge line using an edge detector, then the coordinates of the left and right curb lines which represent the width of the road. The results of this paper indicate that the system's ability to distinguish between road and non-road areas in several sections with an average percentage of 99.59% for accuracy, 99.49% for precision, and 98.84% for recall and the system's ability to mark the left and right edge of the roadside is very good with an average percentage reaches 99.27% and the percentage error and accuracy obtained in providing information on the position of the car from the center (center offset) based on the actual value and prediction results are 16.05% and 84.14%.**

Keywords—autonomous car, road detection, hsv, steering assist

I. INTRODUCTION

An autonomous car or commonly known as a car without a driver is a car that can analyze or recognize conditions that exist on the road by placing a series of sensors that are spread throughout the body of the car. Some of the capabilities of autonomous cars include being able to detect vehicles and estimate their distance, can detect traffic signs, accelerate or automatic braking, and so on, to make this car drive like it is controlled by humans.

The idea of a vehicle that can drive itself is much further than Google's research today. The concept of an autonomous car originated from Futurama, an exhibition at the 1939 New York World Exhibition. This exhibition is a vision for the next 20 years that contains an automatic highway system and illustrates how the United States can be connected in a wide spread network of highways and highways [1].

In designing an autonomous car technology, one of the important things is the ability to detect road lanes which are a parameter so that the car stays on the right lane. So far, the types of roads detected are structured roads, namely roads that have borders and markings. However, the fact is that not all roads have a border or marking line, so the system can't detect the correct lane if it passes through the road area.

Fig. 1. Differences between unstructured road and structured road.

The purpose of this paper is how to design a system that can distinguish between road and non-road areas in various road conditions and perform roadside boundary marking using just a single camera.

II. RELATED WORKS

Detect roadside boundaries by recognizing the situation of the road boundary using the Convolutional Neural Network. There are various classes used as parameters as road conditions (*i.e., white lines, blur white lines, sidewalks and white lines, blur sidewalks and white lines, sidewalks, sidewalks and grass, grass and white lines, grass and blur white lines, sidewalks and white lines*). There are 25 classes used and reach an accuracy level of more than 90% [2].

Road detection system based on RGB color histogram filtering and boundary classification to recognize roads and non-roads. Furthermore, using edge detector (Canny) to detect lines then unite these lines which are assumed to be the road boundaries using Hough transform. The result of this research can detect roads and non-roads well but has not been tested for various lighting conditions [3].

Road detection system by applying threshold technique using the Otsu method. Furthermore, applying the method of IPM (Inverse Perspective Mapping) which is a technique that can see images in various points of view, in this case, the author integrates Bottom-Up to scan and selects non-zero pixels and finally uses Hough transform to adjust two straight lines that represent the boundary left and right. The results of

this research indicate that there was no error detection on 80 road images located in Santiago [4].

The difference between this paper and the previous paper is how to differentiate between the road and non-road areas which is more complex and modern called *semantic segmentation* because this method has been through a training process in various road conditions. This paper will also discuss how to determine the position of the vehicle from the center of the lane (center offset) which will be a system parameter in making the decision to move the car's steering wheel back to the middle of the lane, which previous research did not discuss about this.

III. PURPOSED METHOD

A. Data Acquisition

The data in this paper are roads taken on several roads in Gowa Regency, South Sulawesi, Indonesia. The data collection process began in August using a webcam with a scale (16:9) and a resolution of 1280 x 720 which is placed in the center of the car's dashboard. When retrieving the data, the author also measures the width of the road and the position of the car from the center of the lane as the actual conditions that will be compared with the system measurement results.

Fig. 2. Illustration of data acquisition.

B. Road Detection

The first process that is carried out is to resize the image to a smaller scale of 800 x 450 to minimize the size and reduce the execution time. Furthermore, to classify the classes in the frame by giving different color labels to each pixel is called the *semantic segmentation* technique. The classes are road, sky, building, sidewalk, grass, and so on for the street scene. Various pre-trained models can be used to classify classes in a frame such as Coco, ImageNet, NYU Depth V2, Cityscapes, SUN, OpenSurfaces, PascalContext, ADE20K, and others. In this study, the ADE20K *pre-trained* model was used with a total of 3,169 classes were annotated. This model has a high level of annotation complexity than the models mentioned above [5,6].

Fig. 3. Semantic Segmentation.

Next, convert the image to the HSV color space, then take the street color value in the form of an array on the *semantic segmentation* result image. The array value contains the minimum and maximum values in each component of the color model.

HSV (Hue, Saturation, Value) is one of the popular models used in image processing, especially in terms of object segmentation based on color rather than the RGB (Red, Green, Blue) model because this model is closer to the visual way the human eye distinguishes color sensations. The value in the HSV color space indicates the intensity or brightness of a color that varies from 0 to 100. This device-independent color space can be applied in image analysis and object recognition [7]. Besides, the RGB model has a high correlation between the R, G, and B components [8].

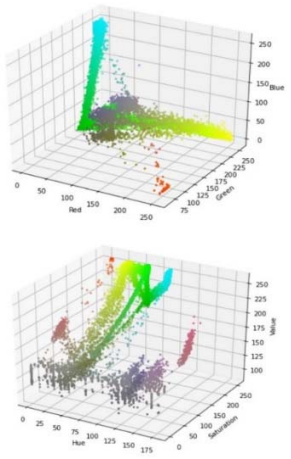


Fig. 4. Comparison of HSV and RGB color space.

Figure 4 is a comparison of the RGB and HSV color spaces from the results of the semantic segmentation process in Figure 3. As we can see that the street color in the image is gray, therefore based on Figure 4, gray is more localized in the HSV color space rather than RGB, making it easier to determine the minimum and maximum street color values.

The next step is to separate the road and non-road areas based on the minimum and maximum value of the road color using the *trackbar*, to make it easier to see the results in *realtime*.

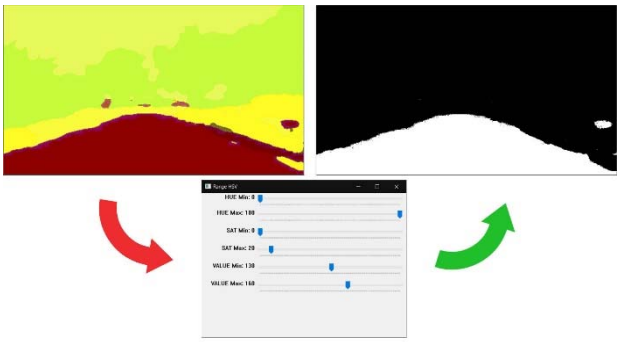


Fig. 5. Thresholding image.

Figure 5 shows the result of the separation between the road and non-road colors based on the color value of the semantic segmentation process. In this case, the color range obtained for the road area is as follows:

- Hue Min = 0
- Hue Max = 180
- Saturation Min = 0
- Saturation Max = 20
- Value Min = 130
- Value Max = 160

The next process is to detect the edge of the road from the result in figure 5 to create a line that represents the roadside using Canny. Several operators can be used to detect edges, namely (Roberts, Sobel, and Prewitt), but the Canny operator has a detection rate that is more maximal than the others both from the inside and outside edges of an image object because Canny is a modified result of the Sobel's technique by checking the horizontal and vertical pixel intensity [9,10].

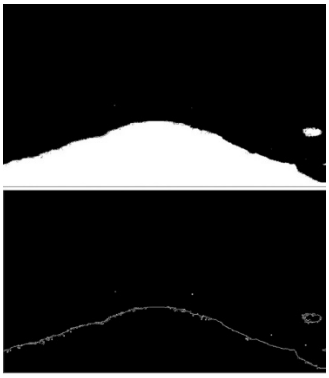


Fig. 6. Edge detection.

The results of the edge detection process still have a lot of noise, therefore it is necessary to carry out an erosion process, where the output pixel value is the minimum value of all pixels in its environment. Next, do a dilation process, where the output pixel value is the maximum value of all pixels in its environment. In this case, the result of the erosion process. So that the result roadside edges look smooth.

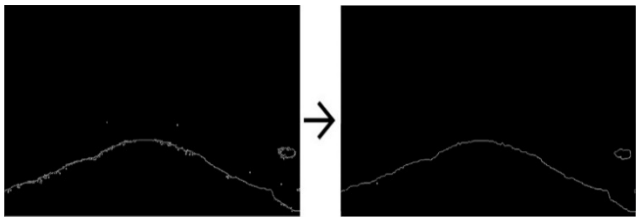


Fig. 7. Noise reduction.

After the noise has been reduced, that there are still parts other than roads that are detected as lines, therefore ROI (Region of Interest) needs to be regulated only on the road area to minimize unnecessary lines. To set ROI there are 3 points (*i.e., left, center, and right*) vertically which will form a triangle. Of course, the value entered adjusts the camera angle at the time of data collection. For this case, the values are as follows:

- left = 400px
- center = 210px
- right = 400px.

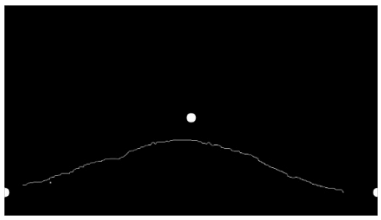
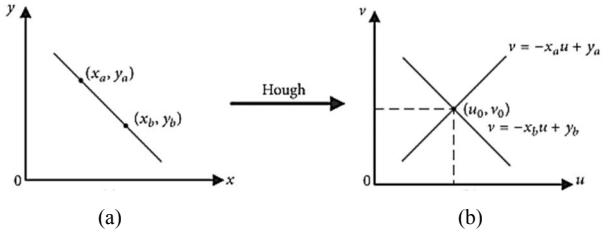


Fig. 8. Set ROI road area.

At this stage, the system will convert the detected pixel points using the Canny operator, especially in the ROI section, into a straight line using Hough transform. The Hough transform method or commonly known as *edge linking* is a popular technique used for image analysis, computer vision, and digital image processing to create straight lines and circles. In detecting edges, the resulting pixels seldom fully characterize the edges due to noise, image cracking, or other effects, resulting in discontinuity of the edges themselves. So that using Hough Transform can connect the broken pixels so that they become one unit and become a solid line [11].

In visualizing lines, the method used in the OpenCV's library is *cv2.HoughLinesP()* which has several parameters in it as follows:

- **Image** is pixel points generated using edge detector.
- **Rho** is the resolution of parameter r in pixels. In this case, use 1 pixel.
- **Theta** is the resolution of the parameter θ in radians. In this case, use 1 degree (CV_PI / 180).
- **Threshold** is the minimum number of intersection points for detecting lines. In this case, use 10.
- **MinLineLength** is the minimum number of points that can form a line. Lines with fewer than this number will be ignore. In this case, use 100.
- **MaxLineGap** is the amount between two points to be considered in the same line. In this case, use 50.



(a) (b)

Fig. 9. The basic principle of Hough transform [12].

In its implementation using (1).

$$\rho = x\cos(\theta) + y\sin(\theta), \quad (1)$$

where (*x, y*) are non-zero pixel coordinates in binary image [12].

ρ = the distance between the x-axis and the fitting line.

θ = the angle between the x-axis and the normal line. Value range θ is ± 90°.

Next, the system will create a straight line on the left and right (e.g. blue and green) using the *cv2.Line()* function is based on the pixel points or lines obtained in the previous process. The left line is defined with the *coordinates_left_line_1* and *coordinates_left_line_2*, while the right line is defined with the *coordinates_right_line_1* and the *coordinates_right_line_2*, which if connected has a height of 160px.

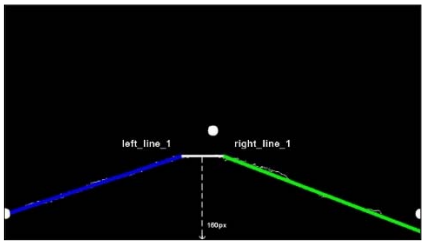


Fig. 10. Roadside edge detection.

When seen in Figure 10, *coordinate_left_line_1* and *coordinates_right_line_1* do not yet have a line-height of 160px, because *coordinates_left_line_2* and *coordinates_right_line_2* have not reached the lower limit of the frame because the ends of the lines are outside the frame.

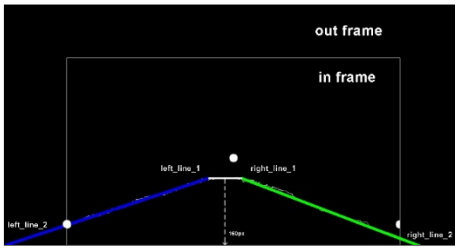


Fig. 11 Roadside edge detection detail.

In figure 11, the *coordinates_left_line_2* and *coordinates_right_line_2* are outside the image which has reached the lower limit of the image. The distance between these two coordinates will represent the current road width. The distance between *coordinate_left_line_1* and *coordinate_right_line_1* represents the width of the road in front. The final step is to connect the four coordinates into one unit and color it as example red using the *cv2.fillPoly()* function, then combine the detection results with the original image using the *cv2.addWeighted()* function.

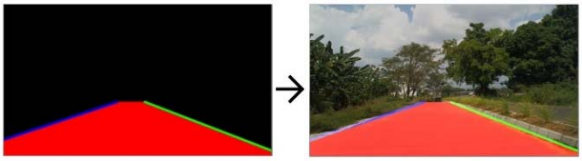


Fig. 12. All road areas detected.

C. Center offset

The scenario in this paper is that the car can maintain its position to remain in the center. At this stage, the system will determine how much the car shifts to the left or right from the center of the lane (center_offset) when driving. First, determine the center position of the car (center_of_car), second, determine the center position of the road (center_of_road) and the third, convert from meters to pixels for the x-axis of the measured road width (xm_per_pix) using (2,3,4).

$$center\_of\_car = \frac{width\ of\ frame\ (800)}{2} \tag{2}$$

$$center\_of\_road = \frac{left\_line\_2 + right\_line\_2}{2} \tag{3}$$

$$xm\_per\_pix = \frac{width\ of\ road}{left\_line\_2 + right\_line\_2} \tag{4}$$

After getting the four information above, then the position of the car can be determined using (5) and display it in text form using the *cv2.putText()* function, and create a white marker line for the center position of the lane from the prediction results using the *cv2.Line function()*.

$$center_{offset} = \left(center_{of_{car}} - center_{of_{road_{now}}}\right) * xm_{per_{pix}} * 100 \tag{5}$$

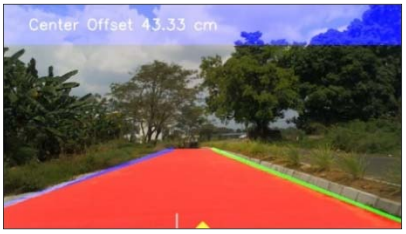


Fig. 13. Final result of road detection.

In Figure 13, the final result where the road has been detected and display information about the position of the car from the center. If the value is positive, then the car is shifting to the right, else if the value is negative, then the car is shifting to the left. These values will be the parameters in deciding to move the car's steer back to the center of the lane. Also shown in the picture is a yellow triangle showing the center position of the car.

IV. Discussion and Evaluation

In terms of knowing the road and non-road areas, the method used in measuring system performance is the *confusion matrix* which contains information in comparing the classification by the system and the actual classification. There are some categories in terms of classification (*i.e., binary, multi-class, multi-label, and hierarchical*). In this case, the type of classification used is binary because in essence there are only 2 data processed, which are road and non-road.

There are some terms in seeing the performance of the system using *confusion matrix*, i.e. TP (True Positive), TN (True Negative), FP (False Positive) and, FN (False Negative) [13].

TABLE I. Confusion Matrix

Data Class	Classified as pos	Classified as neg
Positive	True Positive (TP)	False Negative (FN)
Negative	False Positive (FP)	True Negative (TN)

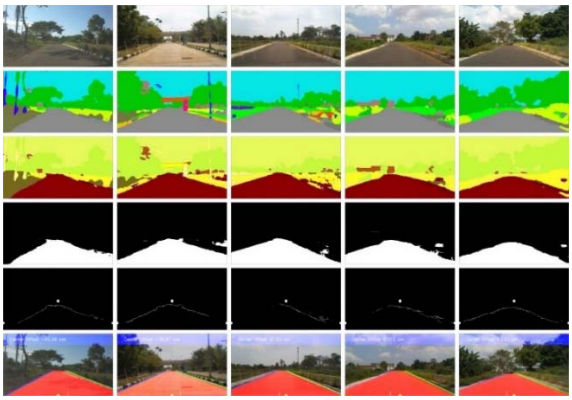
TP (True Positive) is positive data (road) that is predicted to be true, meaning that the actual data is a road and the system predicts it is indeed true. TN (True Negative) negative (non-road) data that is predicted to be true, meaning that the actual data is not a road and the system predicts that the data is indeed not a road. FP (False Positive) is positive data (road) which is predicted to be wrong, meaning that the actual data is a road but the system predicts that the data is not a road. FN (False Negative) is negative data (non-road) which is predicted wrong, meaning that the actual data is not a road but the system predicts that the data is a road. After getting the value of the four terms, the system can get the value of *accuracy*, *precision*, and *recall* using (6,7,8).

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} * 100 \quad (6)$$

$$Precision = \frac{TP}{TP+FP} * 100 \quad (7)$$

$$Recall = \frac{TP}{TP+FN} * 100 \quad (8)$$

Figure 14 is the result of several examples of roads that show the sequence of the process from the beginning to the end. As for seeing the performance of the system in terms of distinguishing between the road and non-road areas, it is necessary to calculate the number of pixels from the actual condition (fig. 14 on the first row) and the predicted result (fig. 14 on the second row).



(a) (b) (c) (d) (e)

Fig. 14. Pipeline of some images.

TABLE II. EVALUATION IN FIGURE 14 TO FIND OUT DIFFERENT OF ROADS AND NON ROADS

Fig	TP	FP	TN	FN	Acc %	Prec %	Rec %
14(a)	77,841	127	269,067	847	99.72	99.83	98.92
14(b)	94,832	664	248,459	299	99.72	99.30	99.68
14(c)	82,907	919	251,362	1,550	99.26	98.90	98.16
14(d)	83,491	208	257,589	1,532	99.49	99.75	98.19
14(e)	78,151	246	270,002	572	99.76	99.68	99.27
Average					99.59	99.49	98.84

If to measure system performance based on the ratio of accuracy in marking the edge of the roadside, then the test method used is *recall* by counting the actual number of pixels on the road (fig. 14 on the first row) and the number of pixels on the predicted road (fig. 14 on the last row).

TABLE III. EVALUATION IN FIGURE 14 TO PREDICT THE ROAD AREA

Fig	TP	FN	Recall
14(a)	77,741	1,130	98.56%
14(b)	89,630	13	99.98%
14(c)	80,723	314	99.61%
14(d)	82,588	1,178	98.59%
14(e)	77,615	286	99.63%
Average			99.27%

To find out the position of the car from the center of the lane (center offset) is strongly influenced by the roadside boundary marking process. In this case, a comparison is made between the results *center_offset* in actual conditions and the predicted results of the system. Next will be calculated *Absolute Error, Relative Error, Percentage Error, and Accuracy* using (9, 10, 11, 12). If the percentage result is a little, the prediction result of the system is also fairly good.

$$Abs.Error = differences\ (actual - measured) \quad (9)$$

$$Rel.Error = \frac{Abs.Error}{Actual} \quad (10)$$

$$Percentage\ Error = Rel.Error * 100 \quad (11)$$

$$Acc. = \frac{low\ value}{high\ value} * 100 \text{ (actual and measured)} \quad (12)$$

TABLE IV. EVALUATION IN FIGURE 14 TO PREDICT CENTER OFFSET

Fig	Road (cm)	Actual (cm)	Meas.(cm)	Abs. Error	Rel. Error	% Error	% Acc.
14(a)	590	58	45.26	12.74	0.21	21.96	78.03
14(b)	622	31	33.63	2.63	0.08	8.48	2.17
14(c)	579	54.5	37.16	17.34	0.31	31.81	68.18
14(d)	590	65	63.12	2.12	0.03	3.26	97.10
14(e)	582	51	43.48	7.52	0.14	14.74	85.25
Average						16.05	84.14

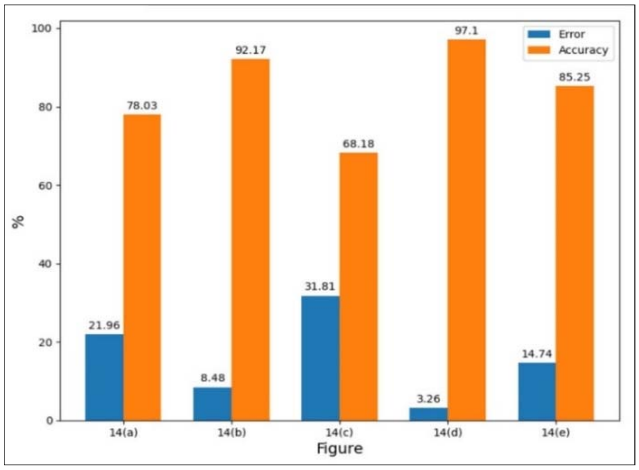


Fig. 15. Comparison of error and accuracy in determining the center offset.

Aside from the above figures, several types of conditions have been tested (*i.e* corners, intersections, and roads with two lanes).

(a) Corner (b) Junction

Fig. 16. Experimental results on road corner and road junction.

The experimental results in Figure 16(a) show that the system has not been able to detect well at a road corner, because the detection process applied is a straight line. Meanwhile, in Figure 16(b) there is also a detection error at the end of the road, due to the cut off of the edge line produced by the *edge detector* at the intersection area.

Fig. 17. Experimental results on two lanes.

Figure 17 shows that the system can properly detect two-lane roads by connecting coordinates (3) and (13).

$$center\_of\_road\_front = \frac{left\_line\_1 + right\_line\_1}{2} \quad (13)$$

As previously explained, this method is very good for roads that tend to be straight.

CONCLUSION

Based on the results obtained at the evaluation stage, it can be concluded that this system is very good in detecting unstructured road areas using the ADE20K *pre-trained* model in sunny weather conditions with an average percentage of 99.59% for accuracy, 99.49% for precision and 98.84% for recall, and the system's ability to mark the average road area reaches 99.27% as well as the percentage error and accuracy obtained in providing information on the position of the car from the center of the lane (center offset) based on actual values and results predictions of 16.05% and 84.14%.

FUTURE WORK

For further development, it is hoped that the next researchers make their *pre-trained* models that only focus on predicting road and non-road areas so that there is no need to classify other classes and can be more flexible in marking roadside boundaries by following the curvature of the road so that the results are well too and keep stable under different conditions.

ACKNOWLEDGMENTS

This work was supported by Computer Based System Laboratory, Department of Informatics, the Faculty of Engineering, Hasanuddin University. We thank Ayoola Olafenwa, creator of PixelLib, who is very expert in performing segmentation of objects in images and videos.

REFERENCES

[1] History of Autonomous Driving." [Online]. Available: https://futurama.io/history-of-autonomous-driving/ [Accessed: 17-Apr-2020].

[2] H. Komori, and K. Onoguchi, "Driving Lane Detection Based on Recognition of Road Boundary Situation," 2018 Int. Conf. Digit. Image Comput. Tech. Appl. DICTA 2018, pp. 1–8, 2019, doi: 10.1109/DICTA.2018.8615784.

[3] M. D. E. Munajat, D. H. Widyantoro, and R. Munir, "Road Detection System Based On RGB Histogram Filterization and Boundary Classifier," ICACSIS 2015 - 2015 Int. Conf. Adv. Comput. Sci. Inf. Syst. Proc., pp. 195–200, 2016, doi: 10.1109/ICACSIS.2015.7415163

[4] Z. Ying, and G. Li, "Robust Lane Marking Detection Using Boundary-Based Inverse Perspective Mapping" 2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 1921–1925, 2016. doi: 10.1109/ICASSP.2016.7472011.

[5] B. Zhou, H. Zhao, X. Puig, S. Fidler, A. Barriuso, and A. Torralba, "Scene parsing through ADE20K dataset," Proc. - 30th IEEE Conf. Comput. Vis. Pattern Recognition, CVPR 2017, vol. 2017-Janua, pp. 5122–5130, 2017, doi: 10.1109/CVPR.2017.544.

[6] B. Zhou et al., "Semantic Understanding of Scenes Through the ADE20K Dataset," Int. J. Comput. Vis., vol. 127, no. 3, pp. 302–321, 2019, doi: 10.1007/s11263-018-1140-0.

[7] P. Ganesan, V. Rajini, B. S. Sathish, and K. B. Shaik, "HSV Color Space Based Segmentation Of Region Of Interest In Satellite Images," 2014 Int. Conf. Control. Instrumentation, Commun. Comput. Technol. ICCICCT 2014, pp. 101–105, 2014, doi: 10.1109/ICCICCT.2014.6992938.

[8] L. Beran, P. Chmelar, and L. Rejfek, "Image Processing Methods Usable for Object Detection on the Chessboard," MATEC Web Conf., vol. 75, pp. 8–13, 2016, doi: 10.1051/matecconf/20167503004.

[9] S. Singh and R. Singh, "Comparison of various edge detection techniques," 2015 Int. Conf. Comput. Sustain. Glob. Dev. INDIACom 2015, vol. 9, no. 2, pp. 393–396, 2015, doi: 10.14257/ijsip.2016.9.2.13.

[10] M. Khairudin, and D. Irmawati, "Comparison Methods Of Edge Detection For USG Images," Proc. - 2016 3rd Int. Conf. Inf. Technol. Comput. Electr. Eng. ICITACEE 2016, pp. 85–88, 2017, doi: 10.1109/ICITACEE.2016.7892416.

[11] M. Li, Y. Li, and M. Jiang, "Lane Detection Based on Connection of Various Feature Extraction Methods," Advances in Multimedia., vol. 2018, 2018, doi: 10.1155/2018/8320207.

[12] S. Singh and A. Datar, "Edge Detection Techniques Using Hough Transform," Int. J. Emerg. Technol. Adv. Eng., vol. 3, no. 6, pp. 333–337, 2013.

[13] M. Sokolova and G. Lapalme, "A Systematic Analysis of Performance Measures for Classification Tasks," Information Processing Management., vol. 45, no. 4, pp. 427–437, 2009, doi: 10.1016/j.ipm.2009.03.002.