International Journal of Advanced Culture Technology Vol.10 No.2 252-259 (2022)
DOI https://doi.org/10.17703/IJACT.2022.10.2.252

IJACT 22-6-34

Study on Image Processing Techniques Applying Artificial Intelligence-based Gray Scale and RGB scale

[1]Sang-Hyun Lee, [2]Hyun-Tae Kim

[1]Asociate professor, Department of Computer Engineering, Honam University, Korea
[2]Student, Department of Computer Engineering, Honam University, Korea
[1]leeesang64@honam.ac.kr, [2]gusxo0560@naver.com

Abstract

Artificial intelligence is used in fusion with image processing techniques using cameras. Image processing technology is a technology that processes objects in an image received from a camera in real time, and is used in various fields such as security monitoring and medical image analysis. If such image processing reduces the accuracy of recognition, providing incorrect information to medical image analysis, security monitoring, etc. may cause serious problems. Therefore, this paper uses a mixture of YOLOv4-tiny model and image processing algorithm and uses the COCO dataset for learning. The image processing algorithm performs five image processing methods such as normalization, Gaussian distribution, Otsu algorithm, equalization, and gradient operation. For RGB images, three image processing methods are performed: equalization, Gaussian blur, and gamma correction proceed. Among the nine algorithms applied in this paper, the Equalization and Gaussian Blur model showed the highest object detection accuracy of 96%, and the gamma correction (RGB environment) model showed the highest object detection rate of 89% outdoors (daytime). The image binarization model showed the highest object detection rate at 89% outdoors (night).

Keywords: *YOLOv4-tiny, Gaussian Distribution, Binarization, COCO Dataset, Artificial Intelligence, Otsu Algorithm*

1. INTRODUCTION

Artificial intelligence is used in fusion with image processing technology using a camera. Image processing technology is a technology that processes objects in an image received from a camera in real time, and is used in various fields such as CCTV and medical image analysis [1]. In such a moving picture, noise is generated under dark or cloudy conditions due to the influence of the external environment, and this is corrected through image processing. It is used to find feature points in the image corrected by image processing or to recognize objects through the camera by identifying objects. In particular, it is used as an important technology of artificial intelligence. A digital image corresponds to two dimensions. Computer processing of images requires understanding them as two-dimensional functions, which are sampled, quantified and processed. Therefore, the process of digitizing an image is a process necessary for a computer to recognize an image.

On March 18, 2018, an accident occurred in which an Uber test self-driving car struck and killed a woman

Manuscript received: May 28, 2022 / revised: June 5, 2022 / accepted: June 9, 2022
Corresponding Author: gusxo0560@naver.com
Tel: +82-62-940-5285, Fax: +82-62-940-5285
Student, Dept. of Computer Engineering, Honam University., Korea

riding a car due to a problem caused by an error in detection among technologies using artificial intelligence. The accident occurred because pedestrians were crossing the crosswalk outside the crosswalk, so it was recognized that the area did not require pedestrian attention [2]. Another example is the use of Pixellot's AI camera during the broadcast of the Scottish Championship match on 03 November 2020. However, an error occurred in that the camera recognized and tracked the head of the referee rather than tracking the soccer ball [3]. Due to this problem, if the recognition speed of image processing is low, the accuracy of security monitoring may be lowered, and a malfunction may occur by recognizing noise in the image. Therefore, accurate recognition rate and recognition speed are very important in real-time image processing technology [4-5].

In this paper, image processing algorithm techniques such as normalization and Gaussian model are applied to improve the recognition rate of real-time object detection. In this paper, the 1-Stage Detector method and the YOLOv4 model, which have relatively low accuracy, were selected to compare the improvement of object recognition rate using the image processing algorithm technique [6-7]. We also use the COCO Dataset to perform artificial intelligence training.

For the test, real-time shooting images were used for comparison by frame, and five algorithms (Normalization, Gaussian distribution, Binarization, Histogram Equalization, Gradient) were applied to the gray scale image., image processing is performed using Gamma correction. Through this process, the recognition rate and accuracy of real-time object detection are compared.

2. RESEARCH METHOD

In this paper, we intend to apply image processing technology in two situations: a black-and-white environment and an RGB environment. In this study, the original image and 8 image detection results applied with image processing algorithm are compared in three environments: indoor, daytime, and night (indoor, outdoor(Day time), outdoor(Night)).

The image processing method applied in this paper is divided into Gray Image Processing Algorithm and RGB Image Processing Algorithm. Gray Image Processing Algorithm is an image processing algorithm applied in a black-and-white environment, and performs 5 types of image processing: Normalization, Gaussian distribution, Binarization, Histogram Equalization, and Gradient. RGB Image Processing Algorithm is an image processing algorithm applied in RGB environment to process color images. Here, three types of image processing are performed: Histogram Equalization, Gaussian blur, and Gamma correction.

The normalization algorithm is an algorithm that changes the contrast of an image by distributing it evenly over the entire area using the ratio of the minimum and maximum values of the pixels of the image clustered in a specific part of the image histogram [8].

$$c(x,y) = (c(x,y) - Min)\frac{nMax - nMin}{Max - Min} + nMin \quad (1)$$

The image normalization algorithm is shown in Equation (1). In this case, $c(x, y)$ is the comparison value of the coordinates to be processed, *Min* and *Max* are the minimum and maximum values of the histogram of the existing image, and *nMin* and *nMax* are the minimum and maximum values of the image, respectively.

The Gaussian distribution is reduced to a left-right symmetrical bell-shaped shape around the mean [9].

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{\left(-\frac{x^2+y^2}{2\sigma^2}\right)} \quad (2)$$

Equation (2) describes the Gaussian distribution formula in two-dimensional space. The variable value is the distance from the origin of the horizontal axis of the *x* axis, the *y* axis is the distance from the origin of the vertical axis, and σ is the standard deviation of the Gaussian distribution.

Gaussian blur is an "image blur filter algorithm" that uses a Gaussian distribution for each pixel in an image, with numbers getting larger as they get closer to the median and smaller as they get further away.

When the Gaussian distribution formula of Equation (2) is applied to a two-dimensional image, a surface that is a concentric circle at the center point of the contour is generated. The distribution values are used to build a convolution matrix such that new pixel values that change are set to the weighted average of that pixel's neighbors. The value of the existing pixel has the highest Gaussian value, and the weight decreases as the distance between adjacent pixels increases. This has a blur effect where the borders and edges are well preserved.

In this study, one of the binarization algorithms, Otsu's Algorithm, was selected. Otsu's algorithm divides pixels into two classifications by randomly setting a threshold and repeats the task of finding the intensity distribution. Through this, it is an algorithm that selects the threshold value with the most uniform intensity distribution of the two classifications.

$$T = \underset{t \in \{0,1,\cdots,L-1\}}{argmin} v_{within}(t)$$
$$v_{within} = \omega_0(t)v_0(t) + \omega_i(t)v_i(t) \tag{3}$$

Equation (3) is an Otsu's Algorithm formula, and is based on the fact that between-class variance is maximized when *within variance* is minimized in the overall variance. However, the total variance only needs to be done once, but once the formula is developed, it takes a long time to derive a formula for calculating the mean. To solve this, when the calculation method of Equation (4) is applied, the previous average value is used as it is and the recursive method is used, so the calculation speed is increased [10-11].

$$\omega_i(t) = \sum_{i=i+1}^{L-1} \hat{h}(i)$$
$$\mu_i(t) = \frac{1}{\omega_i(t)} \sum_{i=i+1}^{L-1} i\hat{h}(i) \tag{4}$$
$$v_i(t) = \frac{1}{\omega_i(t)} \sum_{i=i+1}^{L-1} \hat{h}(i)\big(i - \mu_1(t)\big)^2$$

The RGB Image Processing Algorithm used in the RGB environment performs three types of image processing: Histogram Equalization, Gaussian blur, and Gamma correction.

Histogram equalization is used to flatten the histogram by "reconstructing the image's histogram similarly to the histogram of the surrounding area" where the distribution is concentrated.

$$H(v) = round\left(\frac{cdf(v) - cdf_{min}}{(M*N) - cdf_{min}} * (L-1)\right) \tag{5}$$

In Equation (5), *cdf(v)* represents the cumulative function of the histogram, *cdfmin* is the cumulative

minimum value, *M*N* is the total number of pixels multiplied by the width and height of the image. And *L* is a value indicating a distributed area and *H(v)* is a value of a histogram to which equalization is applied [12].

In Equation (6), *cimg* is an image to which gamma correction is applied, *img* is an original image, and γ is an arbitrary gamma value. If Equation (6) is written as a program code, it becomes the calculation expression of Equation (7). Unlike the formula in (6), the reason for dividing 255 is because the gamma value is defined in the range between 0 and 1, so first divide by 255 to calculate the color, and then multiply it again to express the color.

$$cimg = img^{\gamma * \frac{1}{\gamma}} \quad (6)$$

$$cimg = ((img/255)^{(1/g)}) * 255 \quad (7)$$

YOLOv4 is a one-step detection method, and the accuracy is relatively lower than that of the two-step detection method, but it appears faster. Frames Per Second (FPS) is important for real-time object detection, but since object detection is performed using an image processing algorithm in this paper, we want to compensate for speed by performing additional calculations that affect FPS. The YOLOv4-Tiny model is less accurate, but has a faster detection rate because "it has relatively fewer computational layers than YOLOv4 [13].

3. IMPLEMENTATION

In this paper, the research proceeds in the same order as Fig. 1. In addition, using the YOLOv4-tiny model, the original image and "changed values for the image to which the image processing algorithm is applied" are compared.

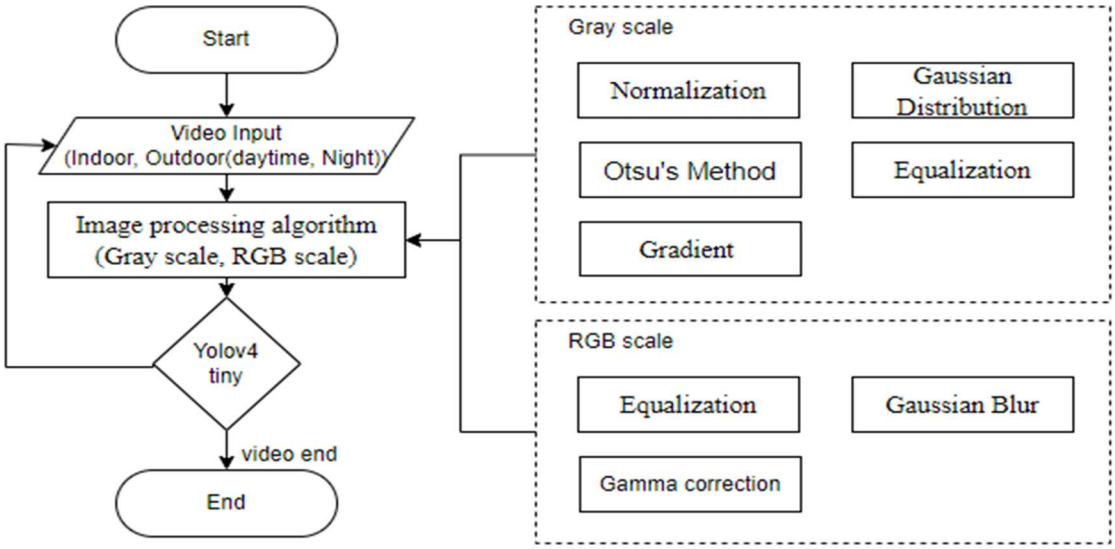

Figure 1. Image processing procedure

The dataset used in this study utilizes training and validation data from 2017 among the COCO datasets [14]. In this paper, the data format of the COCO dataset was changed to fit the YOLO environment, and 4 types of cars, buses, trucks, and people were selected out of 80 data types. The quantity and number of labeling data of Training Data and Validation Data are shown in Tab 1.

Table 1. Data set properties

COCO Dataset	count	COCO Dataset	count
Training Data	69,214	Validation Data	2,930
person	262,465	person	11,004
car	43,867	car	1,932
bus	6,069	bus	285
truck	9,973	truck	415

3.1 Design of the Proposed YOLOv4-tiny Model

The learning model structure of YOLOv4-tiny in Fig. 2 starts with image input, and learning proceeds, where the cuboid is the part that detects objects in the image. In the YOLOv4-tiny model, two types of image detection are performed, and image detection is carried out by using the method of detecting objects in a pyramid shape by dividing the image resolution.

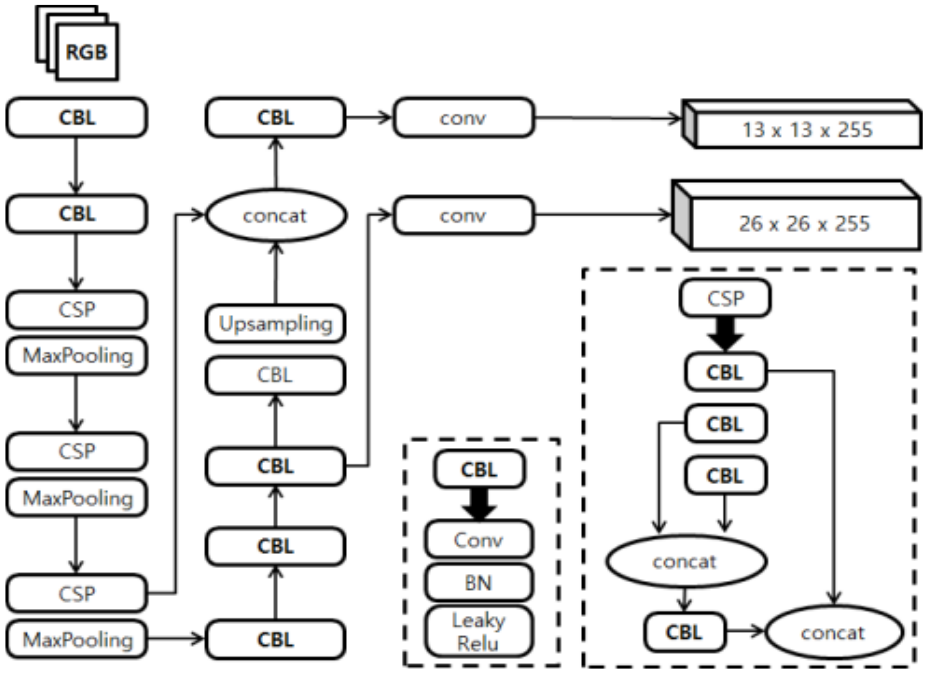

Figure 2. Structure of YOLOv4-tiny Model Applied to this Paper

The size of the convolutional filter of the YOLOv4-tiny model applied in this paper is (5+object quantity) * 3, so (5+4) * 3 = 27", the batch size is 64, and the image size is 416 was set for training.

When performing a test using the YOLOv4-tiny model, the video image is divided into frames and an object is detected using the image "applied with an image processing algorithm". Each image processing algorithm uses five image processing algorithms: Normalization, Gaussian Distribution, Image Binarization, Histogram Equalization, and Gradient for black and white images, and histograms for RGB images. It uses three image processing algorithms: Histogram Equalization, Gaussian Blur, and Gamma correction.

4. RESULT

The development environment of this paper was implemented in a hardware environment of operating system Ubuntu 20.04 LTS, CPU Intel i5-6600, GPU NVIDIA GTX1660 Super 6GB, RAM 12GB, and the software environment is Python 3.8.4, CUDA 11.4, cuDNN 8.2.2, torch Developed with 1.6.0, torchvision 0.7.0 and OpenCV 4.5.3.56.

Table 2 shows obj loss (object loss), box loss (bounding box loss), and cls loss (classification loss) as learning loss rates of the YOLOV4-tiny model applied in this study. Here, the obj loss does not have a big difference between learning after 6,000 steps and 10,000 steps, but the learning continues because the cls loss and box loss have a wide range of changes. After that, at 11,000 steps and 12,700 steps, the fluctuation range of the cls loss and box loss values was low, so learning was stopped.

Table 2. Training Loss

Division	obj loss	cls loss	box loss
1	0.1675	0.01034	0.08688
2,000	0.1396	0.003735	0.05249
4,000	0.1383	0.003666	0.05197
6,000	0.1379	0.003627	0.05186
		…	
10,000	0.1378	0.003607	0.05171
11,000	0.1382	0.003584	0.05154
12,700	0.1372	0.003602	0.05148

Table 3 shows the learning precision, recall, and mAP 0.5 (mean average precision) values, and it was confirmed that the values increased as the number of learning times increased from 2,000 steps. appeared high. In addition, the test is conducted using the 11,000-step learning model with the highest precision value.

Table 3. YOLOv4-tiny Test Values

Division	mAP 0.5:0.95	mAP 0.5	precision	recall
1	0	0	0	0
2,000	0.2807	0.5094	0.3399	0.6066
4,000	0.2883	0.5235	0.3467	0.6203
6,000	0.2926	0.5256	0.3459	0.6273
		…		
10,000	0.2953	0.5302	0.3474	0.6345
11,000	0.2952	0.5301	0.3508	0.6324
12,700	0.2944	0.5324	0.3497	0.6336

4.1 Results of the Proposed Image Processing Test

The environment of the image to be used for the object detection test was taken in three environments: indoor (Indoor), outdoor (daytime) taken at 9:12 am, and outdoor (night) shot at 7:01 pm. Each video was tested indoors at 13 seconds 380 frames, outdoors (daytime) at 14 seconds 437 frames and outdoors (nighttime) at 22 seconds 658 frames.

Table 4. Indoor Video Result

Image processing method	object detection rate	number of false positives
Original Image	96%	1
Normalization	95%	0
Gaussian Distribution	-	0
Image binarization	55%	0
Equalization	95%	0
Gradient	85%	0
Equalization(RGB Environment)	96%	1
Gaussian Blur(RGB Environment)	96%	1
Gamma correction(RGB Environment)	95%	1

Table 4 shows the image processing results of 238 frames among the test results of a moving picture file composed of 380 frames indoors. Among the nine algorithms, Equalization and Gaussian Blur model had the highest object detection accuracy of 96%, but the number of false positives was included one by one.

Table 5 shows the image processing result values of 264 frames among the test results of a video file composed of 437 frames outdoors (daytime). And it shows the image processing result of 558 frames among the test results of the video file composed of 658 frames outdoors (nighttime).

Table 5. Outdoor Video Results (Day and Night)

Image processing method	Outdoor(Daytime)		Outdoor(Night)	
	Object detection rate	Number of false positives	Object detection rate	Number of false positives
Original Image	87%	0	85%	0
Normalization	85%	0	87%	0
Gaussian Distribution	-	0	-	0
Image binarization	-	0	89%	0
Equalization	85%	0	87%	0
Gradient	84%	0	-	0
Equalization (RGB Environment)	88%	0	-	0
Gaussian Blur (RGB Environment)	88%	0	86%	0
Gamma correction (RGB Environment)	89%	0	88%	0

Table 5 shows that the Gamma correction (RGB Environment) model had the highest object detection rate of 89% in the outdoors (daytime), and the Image binarization model had the highest object detection rate of 89% in the outdoors (nighttime). Also, the number of false positives did not appear in all the models tested.

5. CONCLUSION

In this paper, image processing algorithm techniques such as normalization and Gauss model were applied to improve the recognition rate of real-time object detection. In addition, in order to confirm the improvement of the object recognition rate, the 1-Stage detector method with relatively low accuracy was selected. The artificial intelligence model was carried out by selecting the YOLOv4 model, and the data set was carried out

using COCO Dataset, a public data set.

The test uses real-time captured images and compares them by frames. For gray scale images, there are 5 algorithms (normalization, Gaussian distribution, Binarization, Histogram Equalization, and gradient). And RGB Scale, image processing was performed with three algorithms (Histogram Equalization, Gaussian blur, and Gamma correction) to compare the recognition rate and accuracy of real-time object detection. Among the nine algorithms, Equalization and Gaussian Blur model had the highest object detection accuracy of 96%, but the number of false positives was included one by one.

Among the algorithms applied in this present study, the Equalization and Gaussian Blur models showed the highest object detection accuracy, and the gamma correction model was judged to have the highest object detection rate outdoors. If the image algorithm applied in this paper is applied, data augmentation is possible for a task that lacks a single object or data set, and high accuracy is expected because the features of the object are further emphasized.

REFERENCES

[1] Y.-S. Kim, Y.Y. Yun and H.J. Cho, et al. "Research Trend of Adversarial Attack in Computer Vision Field". Communications of the Korean Institute of Information Scientists and Engineers, Vol. 39, No. 8, pp. 8-17, Aug 2021.

[2] https://www.yna.co.kr/view/AKR20180320003951075

[3] https://www.ndtv.com/offbeat/ai-camera-ruins-football-game-by-mistaking-referees-bald-head-for-ball-2319171

[4] J.H. Park, O.K. Ha and Y.K. Jun, "A Survey of Real-Time Object Recognition" in Proc. of the Korean Society of Computer Information Conference, pp. 35-36, Jan. 10, 2017

[5] M.R. Greene, A.P Botros and D.M. Beck, et. al "What you see is what you expect: rapid scene understanding benefits from prior experience" Attention, Perception, & Psychophysics, Vol. 77, No. 4, pp. 1239-1251, May 2015. DOI : https://doi.org/10.3758/s13414-015-0859-8

[6] B. Alexey, C.Y. Wang and H.Y.M. Liao, "YOLOv4: Optimal Speed and Accuracy of Object Detection" arXiv preprint arXiv:2004.10934. Apr 2020. DOI: https://doi.org/10.48550/arXiv.2004.10934

[7] Wang, Chien Y and A.B, et al. "Scaled-YOLOv4: Scaling Cross Stage Partial Network" In Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp.13029-13038. Feb, 2021. DOI : https://doi.org/10.48550/arXiv.2011.08036

[8] Tistory, https://sikaleo.tistory.com/83

[9] E.G Lukacs, "A Characterization of the Normal Distribution" The Annals of Mathematical Statistics, Vol. 13, No. 1 pp. 91-93, Mar 1942. DOI: 10.1214/aoms/1177731647

[10] H.M. Jang, E.R. Kim and S.C. Oh, et al. "Embedded Multi-LED Display System based on Wireless Internet using Otsu Algorithm" The Journal of the Institute of Internet, Broadcasting and Communication, Vol. 16, No. 6, pp. 329-336, Dec 2016. DOI : https://doi.org/10.7236/JIIBC.2016.16.6.329

[11] N. Otsu, "A Threshold Selection Method from Gray-Level Histograms" IEEE Transactions on Systems, Man, and Cybernetics, Vol. 9, No. 1, pp. 62-66, Jan 1979. DOI: 10.1109/TSMC.1979.4310076

[12] https://en.wikipedia.org/w/index.php?title=Histogram_equalization&oldid=1047624667

[13] Z. Jiang, L. Zhao and S. Li, et al. "Real-time object detection method based on improved YOLOv4-tiny" arXiv preprint arXiv:2011.04244. Dec 2020. DOI : https://doi.org/10.48550/arXiv.2011.04244

[14] T.Y. Lin, M. Maire and S. Belongie, et al. "Microsoft COCO: Common Objects in Context" European conference on computer vision. ECCV 2014, pp. 740-755, 2014. DOI: https://doi.org/10.1007/978-3-319-10602-1_48