IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received September 25, 2020, accepted September 27, 2020, date of publication September 30, 2020,
date of current version October 13, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3027839

Contaminated Facade Identification Using Convolutional Neural Network and Image Processing

JISEOK LEE[1], JOOYOUNG HONG[2], (Member, IEEE), GARAM PARK[1], HWA SOO KIM[3], (Member, IEEE), SUNGON LEE[4], (Member, IEEE), AND TAEWON SEO[1], (Senior Member, IEEE)

[1]School of Mechanical Engineering, Hanyang University, Seoul 04763, South Korea
[2]School of Mechanical and Aerospace Engineering, Seoul National University, Seoul 08826, South Korea
[3]School of Mechanical System Engineering, Kyounggi University, Suwon 16227, South Korea
[4]School of Electrical Engineering, Hanyang University, Ansan 15588, South Korea

Corresponding authors: Taewon Seo (taewonseo@hanyang.ac.kr) and Sungon Lee (sungon@hanyang.ac.kr)

This work was supported in part by the National Research Foundation of Korea (NRF) Grant funded by the Ministry of Science and ICT for the First-Mover Program for Accelerating Disruptive Technology Development under Grant 2018M3C1B9088331 and Grant 2018M3C1B9088332; and in part by Hanyang University under Grant HY-2019.

ABSTRACT In recent years, as number of new building getting larger, there has been an increased interest in the cleaning of exterior walls. Accordingly, there is a growing interest in automatic cleaning robots that move around the outer building façade. These robots are also required to apply different cleaning methods to remove various contaminants on the outer wall of the building. However, current surface contaminant detection systems can either detect only a single type of contaminant, or are not compact enough for installation on mobile platforms that move around the outer façade. As cleaning workers are able to distinguish various contaminants with the naked eye, we aim to solve this problem by developing a machine-vision system using convolutional neural networks (CNNs) and image processing methods. As it is a compact system that uses only a camera to take pictures and a processor to process the images, it is suitable for applications involving mobile platforms. Object-type contaminants such as avian feces are handled by the YOLOv3 module using the object-detection algorithm. Area-type contaminants such as rusty stains are processed using the color-detection module using the HSV color space, median filter, and flood fill algorithm. Particle-type contaminants such as dust are handled by the grayscale module, converting images to grayscale images and then comparing the average brightness with a reference that is provided in advance. This proposed machine vision system will detect objects, areas, and particle-type contaminants with a single image and some reference images provided in advance.

INDEX TERMS Contaminant detection, convolutional neural network, façade cleaning, image processing.

I. INTRODUCTION

In recent years, with the increase in the number of cases involving environmental pollutants such as sand dust or fine dust, the outer walls of buildings have tended to accumulate more dirt [1]. Accordingly, the market demand for solutions to clean these outer walls has increased. However, the work environment of the workers contracted to clean exterior walls of buildings is extremely dangerous, and requires them to be suspended on a rope during cleaning.

Therefore, many researchers have developed outer wall cleaning robots that replace the humans who perform this task manually [2]–[7]. Equipping these cleaning robots with systems that detect contaminants on outer walls will enable them to perform a greater variety of tasks. If the number and type of various contaminants in the cleaning area are detected in real time, it is possible to automatically spray the cleaning liquid that corresponds to the pollutant, and more efficient cleaning can be performed by varying the force of the brush

The associate editor coordinating the review of this manuscript and approving it for publication was Ehsan Asadi.

depending on the kind of contamination. It can also be used to keep the exterior wall clean by transmitting the current contamination status information of the building exterior to the building's management system.

In order to make such a contaminant detection system, we have looked for surface contamination detection methods. The contamination detector should be compact enough to fit on the robot, and be able to detect all types of contaminants (dust, dirt, birds, etc.) on the outside wall at the same time. A method of detecting liquid contaminant by irradiating an IR ray onto a surface and analyzing the reflected light can [8] get high accuracy in liquid detection. However, the contaminants on the outer wall of the building are not only liquid. It is difficult to use because the measurement results may various. There is also a surface contamination detection method using Mini Raman Lidar [9]. However, there is a problem that the system is too large to be applied to a mobile platform moving on the building façade

The purpose of this study is detecting the various types of contaminants in building façade such as avian feces, rusty stains, and dusts in a straight process in framework. Inspiring from that workers cleaning the outer walls of buildings detect these contaminants only by 'naked eye', we used machine vision to solve the problem.

We propose simple framework using various modules of common conventional image processing methods, setting a straight processing framework to detect various contaminants at once, modifying each method as little as possible. This method allows the framework to use the various basic image processing & some NN based methods easily, in a practical way.

In this study, the contamination factors of outer wall are largely divided into three types: Object-type, Area-type and Particle-type. We selected each type's detection method and merged into single sequential process to detect all types of contaminant at single image. An object-type contaminant is a contaminant whose size and shape are so irregular that it is difficult to cope with the image processing algorithm. Area-type contaminants are contaminants separated by areas of similar color, such as rust stain. Finally, Particle-type contaminants are the remaining contaminants except for the two contaminant types, usually dust and thin oil.

First, the YOLOv3 module detects the object-type contaminant using CNN. The color detection module detects Area-type using the HSV color space for the remaining part except for the detected area. Finally, except for all the detected areas, the grayscale module detects the amount of contaminants accumulated by using the difference in average brightness between the wall surface and the wall surface. Then, the system parameters of the system were robustly optimized according to user conditions through the Taguchi method. [23]–[25]

The paper is composed as follows. Section 2 introduces the overall algorithm. The overall system structure, the order of operations of each module, and the brief roles are described. Section 3 describes the role and details of the YOLOv3 module for detecting object-type contaminants. The use of the YOLO v3 algorithm, and the organization and validation of training data sets, will be discussed. Section 4 describes the role and details of the color detection module for detecting area-type contaminants. A detection method using HSV color space, color detecting, median filter, and flood fill is described. Section 5 describes a grayscale module that detects particle-type contaminants. This section explains how to convert images to grayscale used in this module, and indirect particle detection. Section 6 describes the system variable robust optimal design using the Taguchi method for the detection system. Section 7 describes the results of the overall performance test of the detection system. We discuss the result of experiment and framework itself in Section 8.

II. DETECTION METHOD OVERVIEW

Three modules inside the system process the images sequentially, and the results of each module can be seen in Figure 1. The YOLOv3 module detects the object-type contaminant and calculates the object type, quantity, and detection box. The type and quantity of the object are transmitted to the outside as the contamination info of the corresponding type, and the detection box information is transmitted to the next module so that the calculation is not performed on the already detected area of the contaminant. Next, color detection module detects the area-type contaminant and calculates the type, width and detection box. At this time, the inside of the detection box transmitted from the previous module is not examined. The type and section of the detected contaminant are transmitted to the outside as the contamination info of the corresponding type, and the detection box is sent to the next module for not to calculate again. Finally, the Grayscale transforms the image to grayscale for the remaining area. After calculating the average brightness of the transformed region, calculate the difference from the average brightness of the reference image without contaminant, and then transmit it to the outside as particle-type contamination info.

III. OBJECT-TYPE CONTAMINANT DETECTION

The YOLOv3 module is a module that has YOLOv3 [20] object detection algorithm inside. When the photographed image arrives, it passes through the internal DNN to extract the type, number, and detection box information of the previously learned object. In this case, information on the type and number is transmitted to the outside as Object-type contamination info, and the information of Object-type detection boxes is transmitted to the next module to prevent another module from calculating same contaminant again.

In recent years, Artificial Neural Network and CNN (Convolutional Neural Network) have developed rapidly, and now have the better ability to classify objects than human. [10] There are also many applications like detect rail defects [11], detect cracks in asphalt roads [12], cracks in concrete [13] and tile degradation [14]. There are two major methods that can be used: object detection and semantic segmentation. Both methods are powerful enough to get various

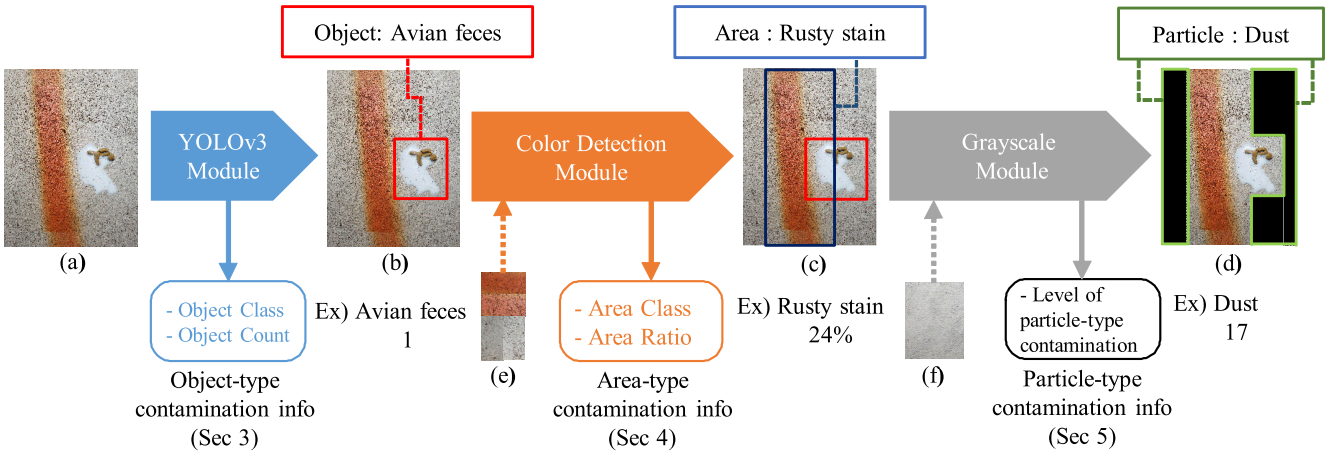


FIGURE 1. Contamination detection system overview diagram. Solid line is output, and dotted line is images provided in advance. (a) is the original image, (b) is result after pass through the YOLOv3 module, (c) is result after pass through the color detection module, (d) is converted image inside the grayscale module. (e) is predefined area and background sample images, (f) is redefined clean background sample image.

information in façade. For application in object detection, it can detect cracks by YOLOv2 which is object detection network structure [15]. For application in semantic segmentation, it can segment many façade objects efficiently [16], [17]. Hyperspectral image classification also can be used [37], [38]

In this study, we used object detection method to detect object-type contaminants. As this system will be installed in façade cleaning robot, the robot cannot pick and clean only the local parts like humans. Instead, it cleans the whole surface at once. Cleaning robots can only use the information that how many object-type contaminations in the region of interest. As segmenting the detected object by pixels is not required, we select the object detection algorithms.

Among the object detection algorithms using CNN such as Faster R-CNN [18], SSD [19], YOLO [20], in this article, we choose YOLOv3 [21], which has a relatively low accuracy but a relatively fast processing speed. Because we thought that there would be heat and power problems to install a high-power, high-performance processor such as a GPU under the condition of an outer wall cleaning robot. If GPU is installed though, get faster processing speed is more important issue.

A. YOLOv3 ALGORITHM

As you can see in Figure 2, there are two parameters inside the module: NMS Threshold and Confidence Threshold. The NMS threshold is used in the Non Maximum Suppression Algorithm (NMS) [26], which makes several detection boxes in a single object after passing through a Neural Network. During the NMS process, we remove other boxes with IoU [27] values below this threshold. Confidence Threshold removes detection boxes whose Confidence level is below a certain value determined in YOLOv3, and filters out object boxes that are detected incorrectly.

B. TRAINING

In this study, avian feces was selected as object-type contaminant, and both learning and verification were performed

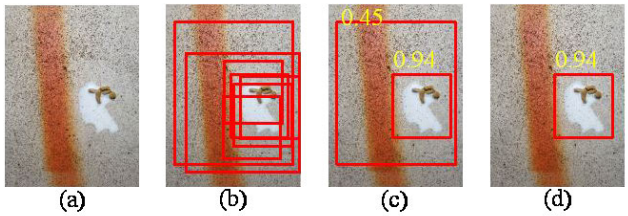


FIGURE 2. Each step image of YOLOv3 module process. (a) is original image. (b) is raw object detection boxes after pass through CNN. (c) is merged boxes through NMS algorithm, yellow number above is confidence level. (d) is final detection box after filtering the box with confidence threshold.

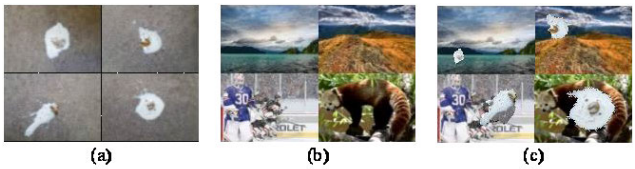


FIGURE 3. Used avian feces data set for training YOLOv3 module. (a) is original artificial avian feces image, (b) is a random background image to be composited, (c) is composited training set image.

using artificial avian feces. We used clay soil, white paint mixed with water to make artificial avian feces. As a background, acryl covered with concrete-patterned interior film was used. We trained Neural network inside YOLOv3 with 1306 training sets, 100 test sets, and 200 validation sets. 250 artificial avian feces made of white paint and clay were made on concrete patterned interior film, and 2000 photographs were taken from 8 directions. For data augmentation, After removing the background of 1606 photographs except for 396 defective photographs, the images were resized with arbitrary positions and sizes. Then composited with random wallpaper photos. Images used in training process can be seen in Figure 3, The network structure adopted from the basic structure of YOLOv3, and the transfer learning [25] method

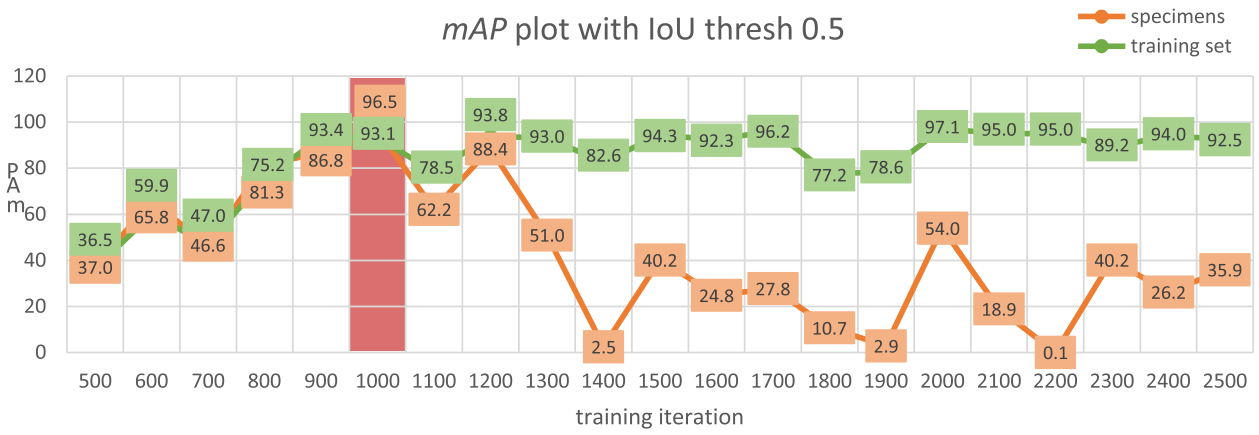


FIGURE 4. YOLOv3 module training results. Iteration 1000 selected.

TABLE 1. Result of training implementation.

Class number	1	Avian Feces
CPU used	Intel i5-9400F	
GPU used	RTX 2080	
precision	0.92	IoU thresh=0.25
recall	0.94	IoU thresh=0.25
F1-score	0.93	IoU thresh=0.25
AP	0.95	IoU thresh=0.5 Area-Under-Curve for unique Recall

is used in the training process, using pre-trained model (darknet53.conv.74). As shown in Figure 4, we selected the 1000th iteration weights with the highest mAP(mean Average Precision) to specimens used in parameter optimization at Section 7. [35] Result of 1000^{th} iteration weights on 200 test sets can be seen in Table 1.

IV. AREA-TYPE CONTAMINANT DETECTION

The color detection module extracts a region of a specific color in the image. we select HSV as the color space to be used in this article [28]. At this time, the color information and the area of the extracted region are primarily transmitted to the area-type contaminant info on the outside, and the detection box is created secondarily to the next module so that the contaminated area is not processed again.

Inside the color detection module, a background area and a color area sample are used as an input, and a hue range and a saturation threshold are calculated. A hue range and a saturation threshold is used to extract a color area using HSV color space. Figure 5 shows the process of detecting colors.

First, an image with RGB color space is converted to HSV color space. In this case, H (0-255 Hue) represents the direction in the color circle, S (0-255 Saturation) represents the saturation of the corresponding color, and V (0-255 Value) represents the brightness. Then, a Hue depth map is created by mapping the H value using a predefined hue range for a pixel having an S value equal to or greater than the predefined saturation threshold. In this case, the saturation threshold and hue range information are automatically determined by using the histogram by receiving the sample image of the background and color area. Since the median filter is used during the process, the system parameter: median filter size is used, and the parameter hue margin gives a margin to both sides of the defined hue range since it may look slightly different from the hue range sampled by the camera noise.

Next, we perform median blur and down sampling to remove the noise of the hue depth map and reduce the size of the photograph to speed up the flood fill process. The size of the median filter used during this process is the same as the size of the median filter used in the previous sample images, and the sampling interval of down sampling is the parameter-down sampling size. First, the area having depth values greater than 0 in the flag map is transmitted as Area-type contaminant info. Secondly, we wrap the area in the form of a detection box using the flood fill algorithm on the flag map and send it to the next module so that the contaminant is not processed again for the area already detected. During this process, the inside of the object-type detection box from the previous YOLOv3 module is not detected.

A. MAPPING HUE VALUE

First, the Color detection module converts the image into HSV color space and separate the Hue channel and Saturation channel. As Figure 5-(a) $\sim$ (d) shows, the saturation values above the saturation threshold are mapped into the value between 0-127 integer, saved as Hue depth map. the maximum value is 127 because the depth is stored in the second to eighth bits as the first bit is used as the flag bit during the flood fill process for an unsigned 8-bit storage per pixel. As shown in Figure 6, Hue range consists of upper limit

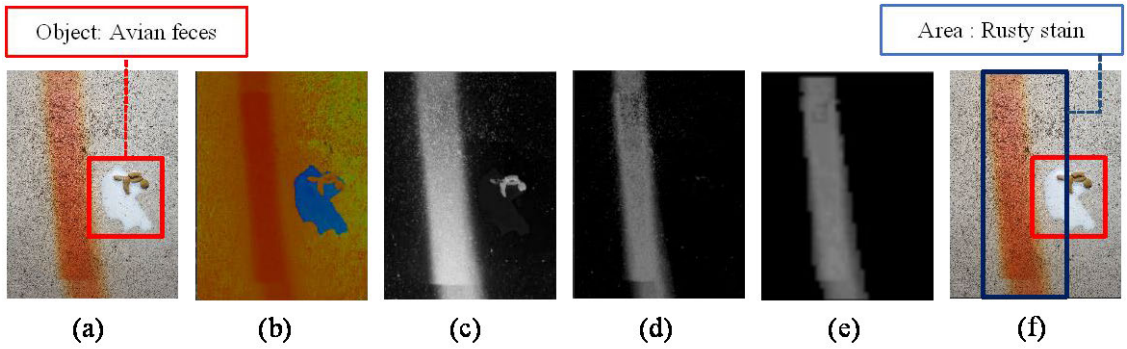


FIGURE 5. Each step image of Color detection module process. (a) is image with object detection box from module before. (b) is split hue channel from original image. (c) is hue depth map. (d) is median blurred hue depth map. (e) is flag map made by subsampling and median filtering the hue depth map. (f) is final detection boxes.

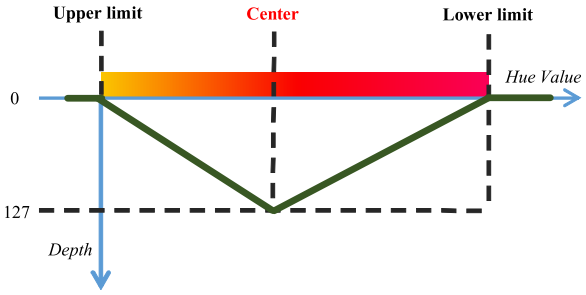


FIGURE 6. Each step image of Color detection module process. (a) is image with object detection box from module before. (b) is split hue channel from original image. (c) is hue depth map. (d) is median blurred hue depth map. (e) is flag map made by subsampling and median filtering the hue depth map. (f) is final detection boxes.

value, center value, and lower limit value. Since hue value is expressed as an integer between 0 and 255, it is a cyclic value. So, the center value or lower limit can be higher than the upper limit value. In this case, we calculated where the value '0'. when the value less than 0 is changed to 255, we add -255 and compare the result. After mapping, if the value is negative, add 255 to get original value. After mapping the Hue value through the hue range, map the value to range of 0-127 and uses it as the hue depth in the Hue depth map. The larger the Hue depth, the closer the color we want to extract.

B. DOWN SAMPLING WITH MEDIAN FILTER

After get Hue depth map, we use the Median filter to smooth out the area and reduce the camera noise. The median filter is a nonlinear filter that replaces the value of the pixel to median of the values of surrounding pixels. And it is effective for reducing salt-pepper noise [29]. However, there is a disadvantage that the processing time is longer among the image noise canceling filters such as Gaussian filter [30] and Bilateral filter [31]. Bigger the Median filter size, more the image noise reduced, but processing time increases. The length of one side of the median filter was selected as a system parameter: Median filter size. In the process of creating a detection box through the flood fill, larger the number of pixels in the image, longer the time takes. In this article, we tried to increase the processing speed by reducing the resolution of images through sub sampling. A noise filter and a sub sampling were performed at the same time by applying the Median Filter to the center of the square of a certain size. If the sampling interval is too narrow, the number of total pixels does not reducing well. If the interval is too wide, the number of pixels will be reduced but the accuracy of the detection box will be reduced also. Though the length of one side of the sub sampling square was selected as a system parameter: Sub sampling size. Through the above sub-sampling process, we remove the noise of the Hue depth map and reduce the resolution of the image to generate the flag map that can calculate the extracted area-type contaminant section. This process can be seen in Figure 5, (d) $\sim$ (e).

C. DETECTION BOX BY FLOOD-FILL ALGORITHM

The Flag map generated in the previous step shows the area of the specific color we were looking for in this module. The ratio of the detected color-area to the entire screen can be calculated by the ratio of the total number of pixels in the flag map and the number of pixels whose hue depth value is not 0. Also, in this study, the Flood fill algorithm using Queue was used to wrap the non-zero pixel set in the flag map with the detection box. The larger the size of the flag map, the better the resolution, but the slower the process.

Figure 7 shows the process of creating a detection box with the Flood fill algorithm. The depth value of one pixel of the flag map is consists of 8 bits: 1 bit used as flag information and the remaining 7 bits representing the hue depth of the pixel. If the flag value is 0, the pixel is not wrapped in the detection box. If the flag value is 1, the pixel is processed. In addition, the remaining 7-bit unsigned integer value represents hue depth in the range of 0 to 127.

The detailed operation of the flood fill algorithm is as follows. It starts at the beginning of the flag map and proceeds until it comes to pixels with depth values between 0 and 127. When it finds that pixel, it stops and starts flood filling. It checks the depth value of neighboring pixels, and if the depth is between 0 and 127, it keeps checking the

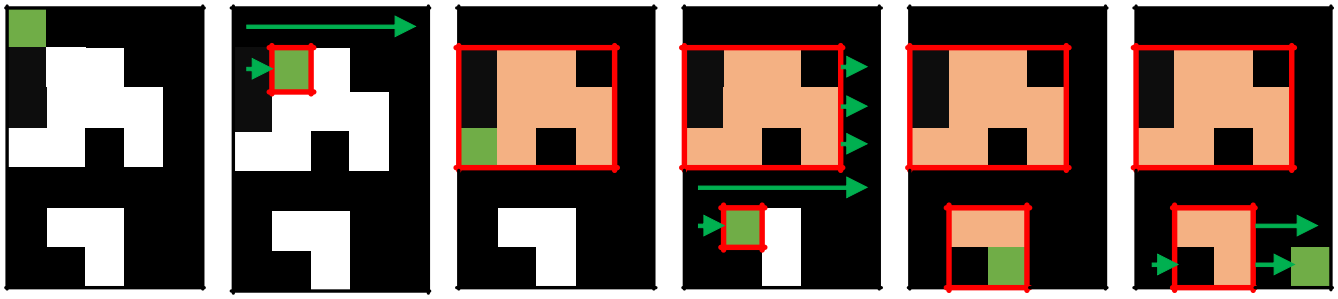


FIGURE 7. Procedure of make detection boxes with flood fill. The original depth map is shown in left image. First, Searching pixel that has proper depth. Set flag to 1 and searching other neighbor pixels until all connected pixels get into the detection box. Searching proper pixel again, make another detection box, doing the same procedure. End the procedure when reached end of the Flag map.

neighboring pixels of neighboring pixels. When checking for new neighboring pixels, it increases the size of the detection box according to the direction. if all neighboring pixels of certain pixel in the four directions is inspected, changed the flag value to 1 so that the inspected pixel is not checked again. When all neighboring pixels have been inspected, the detection box is saved and then moved forward. If another detection box is encountered before encountering a pixel with the appropriate depth again, it will cross over without inspecting inside that box. This process is repeated until the end of the flag map is reached to obtain detection boxes. Also, in Figure 5, the actual results can be confirmed by the pictures (e) to (f). Also, sudo code of the algorithm used in this article is shown in appendix.

D. HUE RANGE & SATURATION THERSHOLD DETERMINATION

Before creating the hue depth map by converting the image input to the color detection module into the HSV color space, we will determine hue range having the color we are looking for with an S value above the saturation threshold. A standard for this is needed. Since the hue value depends on the color you are looking for, you need to redefine the hue range if you want to extract a new color as you set a hue range that selects only one specific color. Therefore, before starting to detect the exterior wall, the module provides a background and color area sample that are prepared in advance so that the optimum hue range and saturation threshold can be found automatically. This makes it possible to automatically determine parameters even if the background changes or the color to be extracted is changing.

The process of automatically determining the hue range and saturation threshold using the background and color area samples is as follows. First, convert each sample image to HSV color space, and then remove the camera noise by using Median filter on Hue and Saturation channels. After that, the values of the Hue channel of the color area sample picture are sorted in ascending order, and then the lower 5% and the upper 5% are become both end values, and the average become center value. Create a hue range based on these three values, but when shooting different images on a real camera, colors may appear slightly different, so expand the range of hue ranges by the certain value to create a new hue range and set that value as a system parameter: hue margin. Next, calculate histograms for the saturation values of the background and colored area samples to calculate the saturation threshold. Then find the saturation threshold values that cover all possible saturation distributions on the colored area but not the background saturation distributions. First, get the highest saturation value of the background and the top 5% saturation value. Then, get the lowest saturation value and the lowest 5% saturation value from color area. Then set the saturation threshold in the middle of the other nearest saturation value without exceeding the lower 5% saturation value in the color area. If the saturation value of the lower 5% of the color area is lower than the saturation value of the upper 5% of the background, the latter value is set as the threshold. Even if both the background and the colored area are passed at the saturation threshold, pixels that do not have an H value in the hue range will eventually change to zero during the process of creating the hue depth map. Therefore, the goal is to create a threshold that can be distinguished as much as possible. Figure 8 shows the histogram and sample image used in this process.

V. PARTICLE-TYPE CONTAMINATION DETECTION

The grayscale module converts the photo to a grayscale image using a specific color space and then calculates how dark the image is using the overall difference in average brightness value from the reference. Assuming that the degree of darkening of the image is proportional to the degree of particle-type contaminant accumulation. The calculated difference is transmitted externally as Particle-type contaminant info. As shown in Figure 9, the calculations are performed for the remaining areas except for the detection box sent in the previous module, to prevent contamination of the average brightness value by other types of contaminant.

A. COLOR-TO-GRAYSCALE CONVERSION METHOD

There are many ways to convert an image to grayscale, but in this article, the most representative grayscale conversion methods, V (Value) of HSV color space, Y (Luminance) of YUV color space, and L (Lightness) of CIE L $*$ a $*$ b, are considered. The HSV color space is very intuitive for

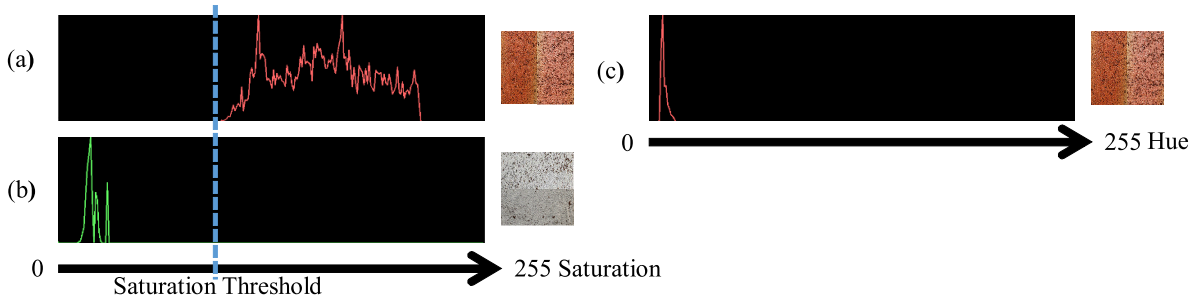


FIGURE 8. Automatically determine saturation threshold and hue range. Each right images are sample images. (a) is histogram of saturation in colored area samples, (b) is histogram of saturation in background samples, (c) is histogram of hue value in colored area samples.

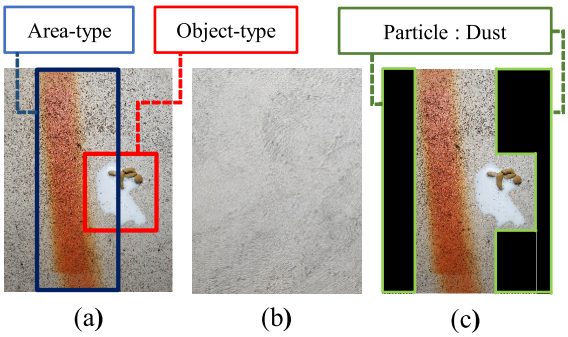


FIGURE 9. Each step image of Grayscale module process. (a) is input image from front module (color detection module). (b) is the reference image to calculate average brightness in advance. (c) is the selected section that outside of the detection boxes.

humans, and the V value represents the brightness of the color. The YUV color space was developed in the black-and-white television era. the *Y(luma)* value represents the luminance. YUV is divided into YCrCb and YPrPb according to the transmission method, In this study, we used ITU-R BT.601 standard. Finally, the CIE L ∗ a ∗ b color space has a perceptually uniform color distance, and L represents lightness.

Each value of RGB used in this article is expressed as an integer between 0 and 255. The method of converting the RGB color space to the V value of the HSV color space is as follows [32]:

$$V = Max(R, G, B) \tag{1}$$

The RGB to YUV(YCbCr) color space conversion method used in this article is as follows. [33]

$$Y = 0.257R + 0.504G + 0.098B \tag{2}$$

The conversion method of RGB to L value in CIE L ∗ a ∗ b color space is as follows [34]. First convert the RGB color space to the CIE XYZ color space.

$$X = 0.4303R + 0.3416G + 0.1784B \tag{3}$$
$$Y = 0.2219R + 0.7068G + 0.0713B \tag{4}$$
$$X = 0.0202R + 0.1296G + 0.9393B \tag{5}$$

Next, convert LAB XYZ to LAB L ∗ a ∗ b color space.

$$L = 116f\left(\frac{Y}{Y_n}\right) \tag{6}$$
$$f(q) = \sqrt[3]{q} \quad forq > 0.008856 \tag{7}$$
$$f(q) = \sqrt[3]{q} \quad forq \leq 0.008856 \tag{8}$$

In this case Y_n is derived from CIE XYZ, and the values of Y_n under illuminant U65 with normalization $Y = 100$.

B. BACKGROUND REFERENCE IMAGE

For the grayscale module to work, you need a clean sample of the outer wall with no contaminants in the same illumination conditions same as during operation. Details can be found in Figure 9 (c). We need to convert this sample to grayscale, calculate the average brightness of the photo, and make it a brightness reference. This reference needs to be calculated and stored only once when the system starts. During operation, the fixed brightness reference value is compared with the average brightness of the current image and the outputs the difference as particle-type contaminant info. Measure how dark the image is by accumulating particle-type contaminants on the façade. You can see the actual figure in Figure 9, (c).

VI. PARAMETER DESIGN

The system created in this article consists of three sequential modules. The system parameters from each module are shown in Table 2. We optimized the value of the system parameter using the Taguchi method like optimize other problems [22]–[24]. But in this article, each module sequentially affects the next module, but the results of subsequent modules do not affect the previous module. Therefore, the system parameters of the YOLOv3 module are optimized first to fix the values. Next, based on the fixed value, we proceed to optimize the next module, the color detection module. Finally, fix the system parameters of the color detection module and optimize the system parameters of the grayscale module. YOLOv3 and color detection module were processed by Taguchi method, and grayscale module was optimized by simple comparison experiment.

In the Taguchi method, we selected height and brightness as user conditions and separated them into 2 and 3 levels, respectively. All specimens used were manufactured by hand.

TABLE 2. Selected system variables.

System Parameter	YOLOv3 Module		Color detection Module			Grayscale Module
Parameter	NMS Threshold	Confidence Threshold	Hue margin	Median filter size	Sub sampling size	Color space

TABLE 3. System detection accuracy result.

Object-type (rms)	Area-type (rms)	Particle-type (mean)
$errIoU_c$	$errIoU_a$	$BTDC_{CS}$
0.3133	0.2474	0.967
Minimum IoU	Minimum IoU	
0.6867	0.7526	

Specimens should include the 3 contaminant type: avian feces representing object-type contaminants, rusty stains representing area-type contaminants, and dust representing particle-type contaminants, and also the background should be concrete walls. First, the background pasted by interior film printed with various concrete patterns and textures on a 300mm x 300mm, 2T formax board. Avian feces were made with white paint and brown clay, and rusty stains were painted with orange oil spray on the specimen to draw a color area. Finally, we evenly spread the coffee dust on the board to draw dust on the specimen surface. Section 2-D, Test bench and specimen, of this article [35] shows the test bench used in this experiment and result section.

The YOLOv3 module used$L_9(3^4)$orthogonal arrays in consideration of two variables and two correlations, and the color detection module considers $L_{27}(3^{13})$ orthogonal arrays in consideration of three variables and three correlations. After optimization of the previous two modules, the grayscale module experimented with dividing a system variable into 3 levels with the fixed values, comparing each 3 results to find best parameter value.

VII. RESULT

A. EVALUATION INDEX

To test the overall detection performance of the system, we used 144 specimen images used in the optimizing process, and the results are shown in Table 3. The performance indicators of each module are as follows: $errIoU_c$(error of IoU with Count) for the YOLOv3 module, $errIoU_a$ (error of IoU with Area) for the color detection module, $BTDS_{CS}$ (Brightness-To-Dust Correlation in current Color Space) for the grayscale module. A paper conducted optimization on this system, [35] shows more detailed calculation method of each performance evaluation indexes.

First, we defined the $errIoU_c$ as the evaluation index of the YOLOv3 module. Based on the mAP calculation method introduced in PASCAL VOC [36], we added the relation with count of detected objects. $errIoU_c$ is a real number from 0-1. Better the performance, closer the value to 0. We defined the function measures accuracy of count in object detection, CAR(Count Accuracy Detection) is as follows:

$$Onlyif\,(\alpha \geq \beta),\quad CAR\,(\alpha \geq \beta) = \left(\frac{\beta}{\alpha+\varepsilon}\right)^{(\alpha-\beta)/\alpha},\ \varepsilon = 2 \tag{9}$$

CAR value is always between 0 to 1. TB_c is number of true bounding boxes. DB_c is number of the detected bounding boxes. CB_c is number of the correct bounding boxes, which is most accurate bounding box to each of the true bounding box. Correct bounding box must have at least 0.5 IoU(Intersection over Union) value with true bounding box. The calculation method of $errIoU_c$ is as follows :

$$\begin{aligned} &If\,(DB_c \neq 0\ and\ TB_c \neq 0) errIoU_c = 1 - \mathrm{IoU} \\ &\qquad \times CAR(TB_c, CB_c) \times CAR(DB_c, CB_c) \\ &Elseif\,(DB_c = TB_c = 0)\ errIoU_c = 0 \\ &Elseif\,(DB_c = 0 and TB_c \geq 1)\ errIoU_c = 1 \\ &Elseif\,(DB_c \geq 1 and TB_c = 0)\ errIoU_c = 1 \end{aligned} \tag{10}$$

Second, we defined the $errIoU_a$ as the evaluation index of the color detection module. Because the information included in the area-type contaminant information is the ratio between the area of the image to the area-type contaminant. $errIoU_a$ is a real number from 0-1. Better the performance, closer the value to 0. Calculation method is as follows:

$$TA_r(\text{True Area Ratio}) = \frac{D_p}{T_p} \tag{11}$$

where D_p is the detected pixel count, and$T_p$is the total pixel count of image.

$$DA_r(\text{Detected Area Ratio}) = \frac{D_{fp} \times M^2}{TA_r} \tag{12}$$

where D_{fp} is the detected flag map pixel count, and M is the median filter size. We defined function that measures accuracy of area, AAR(Area Accuracy Ratio). The calculation method of AAR is as follows:

$$\text{If } (0 < \beta \leq 2\alpha)\ AAR\,(\alpha, \beta) = 1 - \left|\frac{\alpha - \beta}{\alpha}\right| \tag{13}$$

$$\text{Else } AAR\,(\alpha, \beta) = 0 \tag{14}$$

AAR value is always between 0 to 1. When $\alpha = \beta$, AAR value become maximum value 1. As P_t is processing time of the

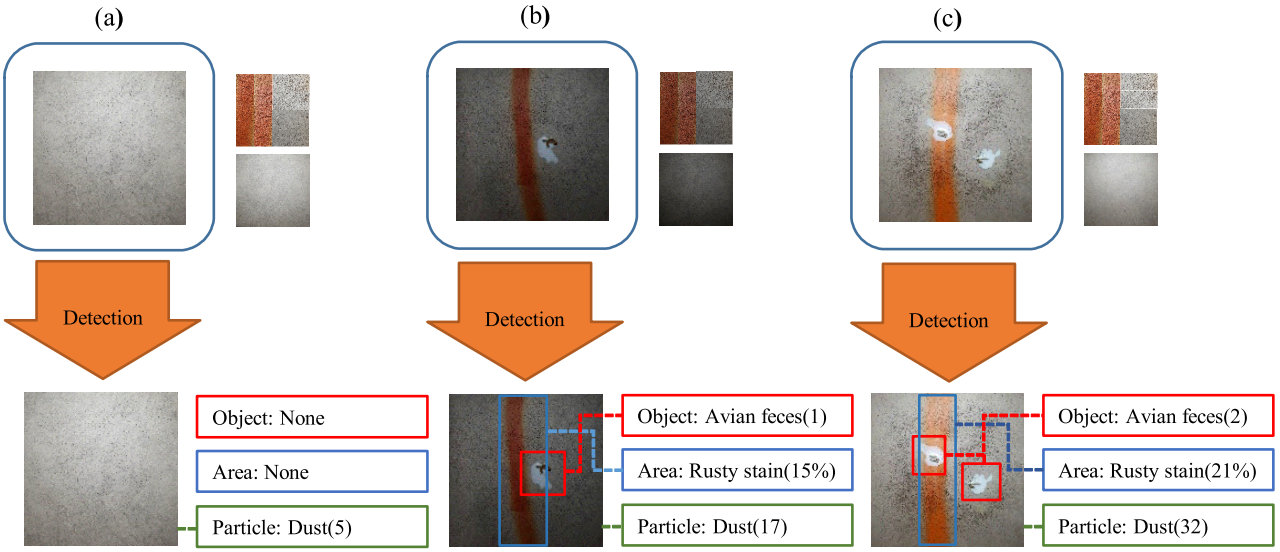


FIGURE 10. Detection results. Each of (a), (b), and (c) is a different specimens. In the box, the left image is the original image from the camera, the right upper image is the color area, and the & background sample for color detection. The right lower image is the brightness reference image for the grayscale module.

color detection module, the calculation method of the $errIoU_a$ is as follows :

$$\text{If } (P_t < 34ms)\ errIoU_a = 1 - IoU \times AAR\,(TA_r, DA_r) \tag{15}$$

$$\text{Else } errIoU_a = 1 \tag{16}$$

Third, we defined the $BTDS_{CS}$ as the evaluation index of the grayscale module. To ensure reliability in the value of the grayscale module, the results of the grayscale module should show similar trends depending on the number of particle-type contaminants. Therefore, when the number of particle-type contaminants increases constantly, we evaluate whether the result of the grayscale module also increases constantly. Therefore, by performing linear regression analysis using the least-square method for the result, we calculated the $BTDS_{CS}$. $BTDS_{CS}$ is a real number from 0-1. Better the performance, closer the value to 1. In the LSM(Least Square Method), the R-value(coefficient of determination) is calculated as follows:

$$\bar{x} = \frac{x_1 + x_2 + x_3}{3}, \bar{y} = \frac{y_1 + y_3 + y_3}{3} \tag{17}$$

$$\mathrm{a} = \frac{\sum_{i=1}^{3}(y_i - \bar{y})(x_i - \bar{x})}{\sum (x_i - \bar{x})^2}, b = \bar{y} - a\bar{x} \tag{18}$$

where $x_i = i$ as the specimens dust amount have ratio of 1:2:3, y_i is the result of grayscale module. Linear regression and R-value formula is:

$$f_i = ax_i + b \tag{19}$$

$$R^2 = 1 - \frac{\sum_i (y_i - f_i)^2}{\sum_i (y_i - \bar{y})^2} \tag{20}$$

the $BTDC_{CS}$(Brightness-to-Dust Correlation with specific Color Space) is as follows:

$$BTDC_{CS} = \frac{\sum_{j=1}^{N} R_j^2}{N} \tag{21}$$

where the R_j is R-value of j-th specimen from dust ratio 1:2:3, CS is the current Color Space used in grayscale calculation method. N is the total number of specimens

B. SYSTEM PERFORMANCE

The contaminant detection of specimens was performed using optimized system parameters. Figure 10 and Table 2 shows each modules performance index values. From the final SNR(Signal to Noise Ratio) in section 7 in paper [35], we can reverse calculate the root mean square of $errIoU_c$ and $errIoU_a$by formula: $\sqrt{10}^{-SNR/10}$

Object-type detection shows rms $errIoU_c$ of 0.3133. $errIoU_c$ is calculated with IoU and CAR function, as in formula (10). we can approximate minimum value of IoU:

$$\begin{aligned} As\text{IoU} &\geq \text{IoU} \times CAR(TB_c, CB_c) \\ &\quad \times CAR(DB_c, CB_c), (0 \leq CAR \leq 1) \\ \text{IoU} &\geq 1 - errIoU_c = 1 - 0.3133 = 0.6867 \end{aligned} \tag{22}$$

area-type detection shows rms $errIoU_a$ of 0.2474. $errIoU_c$ is calculated with IoU and CAR function, as in formula (14). we can approximate minimum value of IoU:

$$\begin{aligned} As\text{IoU} &\geq \text{IoU} \times AAR\,(TA_r, DA_r)\,, (0 \leq CAR \leq 1) \\ \text{IoU} &\geq 1 - errIoU_c = 1 - 0.2474 = 0.7526 \end{aligned} \tag{23}$$

and finally, particle-type detection shows mean $BTDS_{CS}$ of 0.967.

VIII. DISCUSSION

In this study, we propose the framework that detects the various contaminant in building façades by using the existing image processing methods of each type. Framework can detect each type of contaminants by processing sequentially with modules of each contaminant types. Existing various image processing methods were easily used, and each result can be checked within this framework. When the object detection module with the existing YOLOv3 algorithm was properly trained, the following Area-type contaminant detection module also showed a minimum IoU of 0.75. Particle-type detection module also showed good prediction rates for images that passed both modules.

But there's some limitations for this framework. First, the form of detection output of previous detection module is 'box'. In the experiment result, when using the specimen that object-type contaminant is above the area-type contaminant, the precision value of area-type detection module dropped below mean of experiment. This seems to because the current module processes for the rest of the previous module's output section. This is a problem that part to be detected in the current module are cut, in the process of excluding the output section of the previous modules. Second, the data set was not used the real objects. It will be difficult to YOLOv3 network trained using artificial avian feces to respond to real avian feces. Also rusty stains and dusts are not real. Third, the effectiveness of the grayscale module. This module is greatly influenced oy external lighting conditions. It can be applied in a laboratory environment, where is easy to control lightings, but hard to outdoor environment, where lighting conditions cannot be controlled due to sun.

There's some more points need to be improved. There are limitations that come from machine vision itself in the simple visible region. For example, if the color of the exterior wall and the color of the avian feces are similar, or the color of the rusty stain is similar, the camera has no way of distinguishing the two. In this case, in addition to object detection and methods using HSV color space, other methods are needed. Therefore, by measuring the curvature of the outer wall in real-time using a point cloud or depth camera, you can think of sensor fusion with the existing machine vision [39]. Also, in the case of a particle-type detector, it must be robust against changes in brightness as it is a device used outdoors. However, if a mechanical device for controlling the lighting condition is not accompanied, severe noise will occur in the measurement result.

We propose the following future work as this: first, converting the output form of every module from 'box' to 'pixel'. This is expected to increase the probability that the output section of the current module does not contain the section to be processed by the next module. To do this, it will be need to use image segmentation methods as well as object detection methods. Second, improving the quality and quantity of data set used for verification. The current data set is not made by real objects. When applying the results verified using a dataset drawn with art tools to a real environment, it will be difficult to expect precise result. The problem is, it is hard to find exact spot that every avian feces, rusty stains, dust are arranged on concrete wall. Using each real image, it is considered to recombine and create a data set as if were in the desired shape and location. Third is to study sensor fusion like a vision camera for the results of using areas other than visible light such as point cloud or depth camera. Since there is a limit to just looking at the camera, we expect to be able to supplement a lot using information about depth.

In summary, using each image processing method, it showed good detection result. However, there remains a need to improve the output form of the modules inside the framework and the quality of the data set.

Algorithm 1 Color Area Detection Box With Flood-Fill Algorithm

```
Detection Box by Flood-fill algorithm
import flag map as FM
import detection boxes array as DB_arr
Point Queue as PQ
FOR (x,y) = (1,1) -> (rows,cols)
  IF (x,y) is already in the detection boxes at DB_arr
    Jump (x,y) to right end of that box
  ENDIF
  IF FM(x,y) > 0
    Put (x,y) to PQ
    prepare a new detection box
    WHILE PQ is not empty
    dequeue fill_point at PQ
    check FM(east,west,north,south of fill_point) > 0
    IF FM(certain direction) > 0
        Put that point to PQ
        Change 'detected pixel' flag of that
        point to 'true'
        Expand new detection box in that direction
    ENDIF
    ENDWHILE
    Add new detection box to DB_arr
  ENDIF
ENDFOR
```

IX. APPENDIX

See Algorithm 1.

REFERENCES

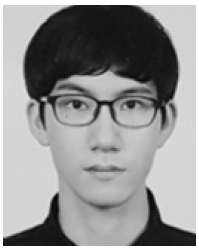
[1] *Skyscraper*. Accessed: Jul. 4, 2018. [Online]. Available: http://www.skyscrapercenter.com/countries

[2] T. Seo, Y. Jeon, C. Park, and J. Kim, "Survey on glass and Façade-Cleaning robots: Climbing mechanisms, cleaning methods, and applications," *Int. J. Precis. Eng. Manuf.-Green Technol.*, vol. 6, pp. 367–376, Mar. 2019.

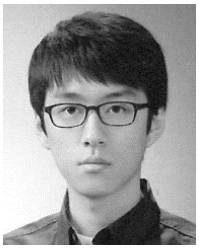
[3] N. Elkmann, M. Lucke, T. Krüger, D. Kunst, and T. Stürze, "Kinematics, sensors and control of the fully automated Façade-cleaning robot SIRIUSc for the Fraunhofer headquarters building, Munich," in *Advances In Climbing And Walking Robots*. Singapore: World Scientific, 2007, pp. 169–176.

[4] E. Gambao and M. Hernando, "Control system for a semi-automatic Façade cleaning Robot," in *Proc. Int. Symp. Automat. Robot. Construct.*, 2006, pp. 406–411.

[5] W. Wang, B. Tang, H. Zhang, and G. Zong, "Robotic cleaning system for glass facade of high-rise airport control tower," *Ind. Robot: Int. J.*, vol. 37, no. 5, pp. 469–478, 2010.
[6] O. Tokhi, H. Zhang, J. Zhang, W. Wang, R. Liu, and G. Zong, "A series of pneumatic glass-wall cleaning robots for high-rise buildings," *Ind. Robot: Int. J.*, to be published.
[7] S.-M. Moon, D. Hong, S.-W. Kim, and S. Park, "Building wall maintenance robot based on built-in guide rail," in *Proc. IEEE Int. Conf. Ind. Technol.*, Mar. 2012, pp. 498–503.
[8] R. Harig, R. Braun, C. Dyer, C. Howle, and B. Truscott, "Short-range remote detection of liquid surface contamination by active imaging Fourier transform spectrometry," *Opt. Express*, vol. 16, no. 8, pp. 5708–5714, 2008.
[9] A. J. Sedlacek, III., M. D. Ray, N. S. Higdon, and D. A. Richter, "Short-range noncontact detection of surface contamination using Raman lidar," *Proc. SIPE*, vol. 4577, pp. 95–104, Feb. 2002.
[10] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification," 2015, *arXiv:1502.01852*. [Online]. Available: http://arxiv.org/abs/1502.01852
[11] S. Faghih-Roohi, S. Hajizadeh, A. Nunez, R. Babuska, and B. De Schutter, "Deep convolutional neural networks for detection of rail surface defects," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2016, pp. 2584–2589.
[12] L. Zhang, F. Yang, Y. Daniel Zhang, and Y. J. Zhu, "Road crack detection using deep convolutional neural network," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2016, pp. 3708–3712.
[13] Y. J. Cha, W. Choi, and O. Büyüköztürk, "Deep learning-based crack damage detection using convolutional neural networks," *Comput.-Aided Civil Infrastruct. Eng.*, vol. 32, no. 5, pp. 361–378, 2017.
[14] P. H. and K. H. Shih Chi, "A deep learning application for detecting facade tile degradation," in *Proc. Int. Conf. Hum. Syst. Eng. Design: Future Trends Appl.*, 2019, pp. 26–32.
[15] V. Mandal, L. Uong, and Y. Adu-Gyamfi, "Automated road crack detection using deep convolutional neural networks," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Dec. 2018, pp. 5212–5215.
[16] H. Liu, Y. Xu, J. Zhang, J. Zhu, Y. Li, and C. H. S. Hoi, "DeepFacade: A deep learning approach to facade parsing with symmetric loss," *IEEE Trans. Multimedia*, early access, Feb. 3, 2020, doi: 10.1109/TMM.2020.2971431.
[17] R. Baumann, J.-F. Evers-Senne, and M. Strand, "Classification of 3D structures based on an object detection for facade elements in multiple views during the reconstruction process," in *Proc. IEEE Int. Conf. Ind. Cyber Phys. Syst. (ICPS)*, May 2019, pp. 230–235.
[18] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 91–99.
[19] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C. Y. Fu, and A. C. Berg, "SSD: Single shot multibox detector," *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 21–37.
[20] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 779–788.
[21] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, and K. Murphy, "Speed/Accuracy trade-offs for modern convolutional object detectors," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 7310–7311.
[22] G. S. Peace, *Taguchi Methods: A Hands-on Approach*. Reading, MA, USA: Addison-Wesley, 1993.
[23] J. T. Krishakant, B. Mohit, and R. Kumar, "Application of Taguchi method for optimizing turning process by the effects of machining parameters," *Int. J. Eng. Adv. Technol.*, vol. 2, no. 1, pp. 263–274, Oct. 2012.
[24] J. A. Ghani, I. A. Choudhury, and H. H. Hassan, "Application of Taguchi method in the optimization of end milling parameters," *J. Mater. Process. Technol.*, vol. 145, no. 1, pp. 84–92, 2004.
[25] S. Jialin Pan and Q. Yang, "A survey on transfer learning," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 10, pp. 1345–1359, Oct. 2010.
[26] A. Neubeck and L. Van Gool, "Efficient non-maximum suppression," in *Proc. 18th Int. Conf. Pattern Recognit. (ICPR)*, vol. 3, 2006, pp. 850–855.
[27] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2015, pp. 3431–3440.
[28] D. J. Bora, A. K. Gupta, and F. A. Khan, "Comparing the performance of L* A* B* and HSV color spaces with respect to color image segmentation," *Int. J. Emerg. Technol. Adv. Eng.*, vol. 5, pp. 192–203, Feb. 2015.
[29] R. Srinivas and S. Panda, "Performance analysis of various filters for image noise removal in different noise environment," *Int. J. Adv. Comput. Res.*, vol. 3, pp. 47–52, Dec. 2013.
[30] J. Weickert, *Anisotropic Diffusion in Image Processing*, vol. 1. Stuttgart, Germany: Teubner, 1998, pp. 59–60.
[31] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," in *Proc. 6th Int. Conf. Comput. Vis.*, Jan. 1998, pp. 839–846.
[32] V. Chernov, J. Alander, and V. Bochko, "Integer-based accurate conversion between RGB and HSV color spaces," *Comput. Electr. Eng.*, vol. 46, pp. 328–337, Aug. 2015.
[33] M. Podpora, G. P. Korbas, and A. Kawala-Janik, "YUV vs RGB-choosing a color space for human-machine interaction," in *Proc. Federated Conf. Comput. Sci. Inform. Syst.* vol. 3, 2014, pp. 29–34.
[34] C. Connolly and T. Fleiss, "A study of efficiency and accuracy in the transformation from RGB to CIELAB color space," *IEEE Trans. Image Process.*, vol. 6, no. 7, pp. 1046–1048, Jul. 1997.
[35] J. Lee, G. Park, Y. Moon, S. Lee, and T. Seo, "Robust design of detecting contaminants in Façade cleaning applications," *IEEE Access*, vol. 8, pp. 2869–2884, 2020.
[36] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The Pascal visual object classes (VOC) challenge," *Int. J. Comput. Vis.*, vol. 88, no. 2, pp. 303–338, Jun. 2010.
[37] F. Luo, L. Zhang, B. Du, and L. Zhang, "Dimensionality reduction with enhanced hybrid-graph discriminant learning for hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 58, no. 8, pp. 5336–5353, Aug. 2020.
[38] F. Luo, H. Huang, Y. Duan, J. Liu, and Y. Liao, "Local geometric structure feature for dimensionality reduction of hyperspectral imagery," *Remote Sens.*, vol. 9, no. 8, p. 790, Aug. 2017.
[39] Y. Tang, L. Li, C. Wang, M. Chen, W. Feng, X. Zou, and K. Huang, "Real-time detection of surface deformation and strain in recycled aggregate concrete-filled steel tubular columns via four-ocular vision," *Robot. Comput.-Integr. Manuf.*, vol. 59, pp. 36–46, Oct. 2019.



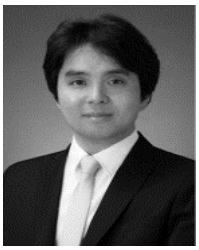
JISEOK LEE received the B.S. degree in mechanical engineering from Hanyang University, in 2019, where he is currently pursuing the M.S. degree in mechanical engineering. His research interest includes robot automatic control.



JOOYOUNG HONG (Member, IEEE) received the B.S. degree in mechanical engineering from Yonsei University, in 2018. He is currently pursuing the M.S. degree in mechanical engineering with Seoul National University. His research interest includes robot mechanism design.

GARAM PARK received the B.S. degree in mechanical engineering from Hanyang University, in 2019, where he is currently pursuing the M.S. degree in mechanical engineering. His research interest includes robot mechanism design.

HWA SOO KIM (Member, IEEE) received the B.S. and Ph.D. degrees in mechanical engineering from Seoul National University, South Korea, in 2000 and 2006, respectively. From 2007 to 2008, he was a Postdoctoral Researcher with the Laboratory for Innovations in Sensing, Estimation, and Control, University of Minnesota, Minneapolis, MN, USA. He is currently an Associate Professor with the Department of Mechanical System Engineering, Kyonggi University. His current research interests include design, modeling, and control of various mobile platforms.



SUNGON LEE (Member, IEEE) received the B.S. degree in mechanical design and product engineering from Seoul National University, Seoul, South Korea, in 1997, the M.S. degree in mechanical engineering from the Pohang University of Science and Technology (POSTECH), Pohang, South Korea, in 1999, and the Ph.D. degree in mechano-informatics from The University of Tokyo, Tokyo, Japan, in 2008. From 2010 to 2012, he was a Postdoctoral Research Fellow with the Center for Systems Biology, Massachusetts General Hospital, Harvard Medical School, Boston, MA, USA. From 1999 to 2015, he was a Senior Research Scientist with the Korea Institute of Science and Technology, Seoul. Since 2015, he has been with the School of Electrical Engineering, Hanyang University, Ansan, South Korea. He is currently an Associate Professor. His research interests include biomedical robotics, surgical robots, medical image processing, and motion compensation systems.

TAEWON SEO (Senior Member, IEEE) received the B.S. and Ph.D. degrees from the School of Mechanical and Aerospace Engineering, Seoul National University, South Korea. He was a Postdoctoral Researcher with the NanoRobotics Laboratory, Carnegie Mellon University, a Visiting Professor with the Biomimetic Millisystems Laboratory, University of California at Berkeley (UC Berkeley), and an Assistant Professor with the School of Mechanical Engineering, Yeungnam University, South Korea. He is currently an Associate Professor with the School of Mechanical Engineering, Hanyang University, South Korea. His research interests include robot design, analysis, control, optimization, and planning. He received the Best Paper Award of the IEEE/ASME TRANSACTIONS ON MECHATRONICS, in 2014. He is also working as a Technical Editor of IEEE/ASME TRANSACTIONS ON MECHATRONICS and an Associate Editor of IEEE ROBOTICS AND AUTOMATION LETTERS and *Intelligent Service Robotics*.