Heliyon 9 (2023) e13213

Contents lists available at ScienceDirect

Heliyon

journal homepage: www.cell.com/heliyon

Research article

Faster RCNN mixed-integer optimization with weighted cost function for container detection in port automation

Steven Bandong [a,b], Yul Yunazwin Nazaruddin [c,d], Endra Joelianto [b,c,d,*]

[a] Engineering Physics Doctoral Program, Faculty of Industrial Technology, Institut Teknologi Bandung, Bandung, Indonesia
[b] University Center of Excellence on Artificial Intelligence for Vision, NLP and Big Data Analytics, Institut Teknologi Bandung, Bandung, Indonesia
[c] Instrumentation and Control Research Group, Faculty of Industrial Technology, Institut Teknologi Bandung, Bandung, Indonesia
[d] National Center for Sustainable Transportation Technology, CRCS Building, 2nd Floor, Institut Teknologi Bandung, Bandung, Indonesia

A R T I C L E I N F O

Keywords:
Optimization
Mixed-integer PSO
Weighted cost function
Faster RCNN
Gantry crane control system
Container detection

A B S T R A C T

The development of port automation requires sensors to detect container movement. Vision sensors have recently received considerable attention and are being developed as AI advances, leading to various container motion detection methods. Faster-RCNN is a detection method that performs better precision and recall than other methods. Nonetheless, the detectors are set using the Faster-RCNN default parameters. It is of interest to optimized its parameters for producing more accurate detectors for container detection tasks. Faster RCNN requires mixed integer optimization for its continuous and integer parameters. Efficient Modified Particle Swarm Optimization (EMPSO) offers a method to optimize integer parameter by evolutionary updating the space of each candidate solution but has high possibility stuck in the local minima due to rapid growth of Gbest and Pbest space. This paper proposes two modifications to improve EMPSO that could adapt to the current global solution. Firstly, the non-Gbest and Pbest total position spaces are made adaptive to changes according to the Gbest and Pbest position spaces. Second, a weighted multiobjective optimization for Faster-RCNN is proposed based on minimum loss, average loss, and gradient of loss to give priority scale. The integer EMPSO with adaptive changes to Gbest and Pbest position space is first tested on nine non-linear standard test functions to validate its performance, the results show performance improvement in finding global minimum compared to EMPSO. This tested algorithm is then applied to optimize Faster-RCNN with the weighted cost function, which uses 1300 container images to train the model and then tested on four videos of moving containers at seaports. The results produce better performances regarding the speed and achieving the optimal solution. This technique causes better minimum losses, average losses, intersection over union, confidence score, precision, and accuracy than the results of the default parameters.

1. Introduction

Increased economic globalization is supported by fast and reliable international transportation services, advances in telematics, standardization, and trade liberalization [1]. Seaports are still the main route of international trade transportation due to the high

* Corresponding author. Instrumentation and Control Research Group, Faculty of Industrial Technology, Institut Teknologi Bandung, Bandung, Indonesia.

E-mail addresses: bandong.steven@gmail.com (S. Bandong), yul@itb.ac.id (Y.Y. Nazaruddin), ejoel@itb.ac.id (E. Joelianto).

https://doi.org/10.1016/j.heliyon.2023.e13213
Received 17 December 2021; Received in revised form 17 January 2023; Accepted 19 January 2023
Available online 26 January 2023

transportation volume that makes it impossible to use planes. Air transportation is used only 0.1–0.6% of the trade volume in Latin America, and 72–96% still uses sea transportation [1]. One of the critical processes in the sea trade route is the process of loading and unloading containers. Amid increasing volume and density of trade traffic, it can cause accidents in containers' loading and unloading process. The operator's improper operation of the gantry crane can result in huge losses because it involves complex processes such as stevedoring, delivery, and receiving, which are high risk for workers, cargo, and port facilities [2]. These errors lead to further consequences such as delays in cargo delivery and higher costs. It is in line with research by Ref. [1] which found that the process of moving containers contributes significantly to efficiency and cost reduction in international trade sea transportation.

Those risks elevate the necessity to build a gantry crane automation system for loading and unloading containers and increase the efficiency of seaports. Several studies have been published in the area of gantry crane automation [3–5]. Dynamic model and Lyapunov method had been applied to control container positioning by Gantry Crane [3]. The fuzzy-PID hybrid method was also applied to control the swing angle and container position [5]. In Ref. [4], a hybrid energy system was proposed to improve the performance of gantry crane control in the container loading process. The application of these control algorithms requires a position sensor and the swing angle of the container to be fed back to the control system.

The researchers used several methods to determine the location of containers, such as the use of wireless sensor networks (WSNs) [6] and container tracking using NBIoT [7]. However, its implementation is expensive because it requires the installation of intelligence elements in the container. The accelerometer has also been applied as a container position sensor, but the vibration caused false and inaccurate position detection. Other methods such as the use of Position Sensitive Detector (PSD) cameras and LED light, Charged-Coupled Device (CCD) cameras and lasers were also applied, but when there is interference from sunlight, dim light, and rain caused errors to increase [8].

The development of artificial intelligence has led to the possibility of camera-based object detection that has the potential to increase the automation of the gantry crane system. The intelligence obtained from this camera image is divided into two, namely traditional methods and deep learning methods. The traditional method of searching the features of the image is obtained from the programmer's intelligence. However, this method is difficult and does not show a significant accuracy improvement in recent years. Deep learning methods, especially CNN, can find image features through the network of neurons used. YOLO [9], MobileNet [10], SSD ResNet [11], Faster RCNN [12], Mask RCNN [13] methods are CNN-based object detection methods.

The Faster RCNN method applies a two-stage detector; the first stage is a proposal area called the Region Proposal Network (RPN). The second stage uses the RPN output to perform bounding box regression and class classification. Faster RCNN in various object detection scenarios shows better performance than other methods [14] applied Faster RCNN to detect pedestrians and had better accuracy than ACF. Faster RCNN was also used as a face detector and gave satisfactory results on various benchmark datasets such as the WIDER face dataset, FDDB and IJB-A [15]. Subsequent studies applied Faster RCNN to detect faces [16], PCB surface defects [17] and bubble detection [18]. Faster RCNN also provided better detection accuracy than SSD MobileNet, MobileNet FPN and SSD ResNet for container detection [19,20]. This paper applies Faster RCNN to detect container objects which will later be used for gantry crane automation. Faster RCNN parameter optimization is applied to obtain the optimal detector.

Several studies have attempted to optimize Faster RCNN. Bao et al. [21] used gaussian distribution as pre-processing data and random forest to reduce complexity in order to obtain Faster RCNN which is computationally faster. Lu et al. [22] developed a regional proposal optimization network (RPON) on the Faster RCNN network to recognize driver actions. Another researcher, Bai et al. [23], developed Faster RCNN by applying DRNet (Dense Residual Network) and RoI (Region of Interest) in the information extraction process. The development of these methods generally creates or incorporates new structures into the Faster RCNN in order to obtain better performance. In some circumstances, an optimization strategy is needed to minimize the loss of the Faster RCNN model that already exists and is widely available to the scholar or engineering society. In this paper, multiobjective optimization of the Faster RCNN using mixed-integer PSO is proposed as a solution.

Particle Swarm Optimization (PSO) is an optimization method inspired by the social behavior of bird groups when looking for food. PSO starts by generating random solutions called particles. Each particle will change its position depending on the change in velocity in each iteration. PSO has been widely applied as an optimization tool in various fields and gives satisfactory results. Sennan et al. [24] used PSO and Type 2 Fuzzy Logic to maximize the internet of things lifespan up to 10%–15%. Smooth path planning for mobile robots had also been successfully developed using PSO [25]. In the field of human health, PSO was applied as a feature reducer for SVM-based diabetes classification [26]. In the control system field, PSO was used as a PID controller tuning for container transfer and reduction of swing angle on a rubber tyred gantry crane (RTGC) [19,20]. PSO was also used to find optimal PID controller and virtual sensor for quadrotor [27] and to control benfield concentration of a stripper unit in a fertilizer plant [28]. PSO is considered a powerful optimization tool, easy to implement, and has a light computational load [29]. This advantage makes PSO very suitable for the optimization of image-based object detection, which generally requires a high computational load during the training process such as Faster RCNN.

Hyperparameter optimization of the Faster RCNN model has its problems because there are continuous and discrete parameters that need to be optimized by means of mixed-integer optimization method. PSO is generally an optimization method for continuous parameters. However, some researchers are endeavouring to develop it also for the optimization of integer or discrete parameters [30–32]. In Ref. [33], an integer parameter optimization method called EMPSO was developed. EMPSO has advantages in its ease of implementation based on the spatial evolution of each potential optimization solution throughout the PSO iteration process. However, this method uses Gbest and Pbest space multiplication against certain constants which causes both of them to grow too fast so they tend to be stuck at their local minimum as the optimization iteration increases.

To handle those limitations, this paper proposes an enhanced integer parameter optimization method by means of two improvements which are adaptive to changes according to the Gbest and Pbest position spaces and weighted cost function that will cause the

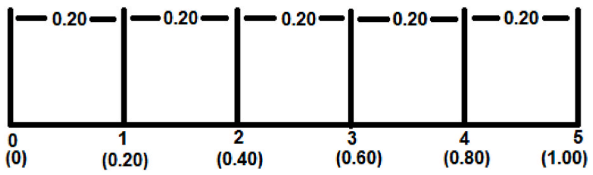


Fig. 1. Standard or initial space.

optimization has more possibility of achieving the global minimum. The improved EMPSO with adaptive position candidate space solution and weighted cost function is then implemented to solve the mixed-integer optimization problem of Faster RCNN. Moreover, Faster RCNN with multiobjective optimization is proposed based on its weighted minimum loss, average loss, and gradient loss during training. The paper contributes to the development of optimal Faster RCNN as a container detector in seaports, enhances the mixed-integer optimization method based on modified EMPSO, and proposes multiobjective optimization, which takes into account the weighting of the minimum loss, average loss, and gradient of loss. The result shows that this method provides a smaller minimum and average loss value than Faster RCNN's default parameter, leading to better detection results.

This paper begins with a general description of PSO and continues with the idea and simulation of the development of mixed-integer PSO in Section 2. Section 3 provides a brief description of the Faster RCNN. Section 4 contains the development of methods, experiments, and the analysis of results. The developed mixed-integer PSO method will be tested on nine optimization test functions and followed by the implementation of Faster RCNN optimization using the proposed mixed-integer PSO and the proposed weighted cost function. The detection accuracy of the obtained results will be calculated by detecting containers on four videos that have never been trained on the model.

2. Particle swarm optimization

2.1. Continuous PSO

PSO mimics the adaptability of fish and bird colonies in nature when looking for food through an information-sharing system. In its implementation, PSO will make a number of particles as candidates for optimal parameters. Each particle has a velocity vector value that will affect the change in particle value at each iteration when looking for optimal parameters. The solution of each particle will affect each other due to the existence of social parameters that build momentum for finding the optimal parameter. The PSO algorithm starts by determining the maximum number of iterations (t) and the number of particle populations (m). The number of parameters will be referred to as the number of positions (k); therefore, the position and velocity matrix of size ($m \times k$) is obtained. Furthermore, in each iteration process, the cost value for each particle is calculated. The position and velocity matrix values used in the next iteration are calculated using Eqs. (1) and (2), where p_{best} is the particle with the best position on the ith particle, p_{gbest} is the best particle for the entire population of particles, p_i is the ith particle, v_i is the ith particle velocity. Furthermore, it takes 4 values of w, μ, c_1, and c_2, namely inertia weight, random number between 0 and 1, cognitive weight, and social weight, to determine the portion of global and local searches in the computing process [24–26].

$$v_{ij}^{n+1} = w.v_{ij}^{n} + c_1.\mu.\left(p_{ijbest}^{n} - p_{ij}^{n}\right) + c_2.\mu.\left(p_{jgbest}^{n} - p_{ij}^{n}\right) \tag{1}$$

$$p_{ij}^{n+1} = p_{ij}^{n} + v_{ij}^{n+1} \tag{2}$$

$$n = 0, 1, 2, ..., t. \quad i = 0, 1, 2, ..., m. \quad j = 0, 1, 2, ..., k.$$

2.2. Integer or discrete PSO

PSO is generally used to optimize the continuous parameter. However, in most cases, optimization is required for integer or discrete parameters [34–36]. Faster RCNN has a continuous parameter as well as an integer that needs to be optimized. This paper developed an integer parameter optimization method, which refers to Ref. [33], which applies evolutionary strategies. This method selects the PSO particle value based on the mapping of the random number generation to space in the range 0–1. Space in the range 0–1 is divided into multiple space parts according to the number of potential integer values as the optimal solution. In each iteration, the space of each of these sections will be updated. The one chosen as the global optimal solution (Gbest) will update its space by multiplying it by the parameter c_3 ($c_3 = 1.5$) and the element selected as the optimal individual solution (Pbest) is multiplied by c_4 ($c_4 = 1.2$). Prospective solutions that are not Gbest and Pbest will have their space updated using the normalization method [33]. For example, if there is a solution set for integer parameters $\Omega_d = \{1, 2, 3, 4, 5\}$ and $|\Omega_d| = 5$, the standard or the initial space is shown in Fig. 1.

The integer EMPSO program will generate a random number and the selected parameter will be calculated for its cost function. For example, if in this iteration, the global optimal is 2 and the individual optimal is 5, then the space will change to the space that is shown in Fig. 2. In the next iteration, it will generate a random number between 0 and 1. This generated number will be in one of the space intervals in the image above. For example, if the result of random number generation is 0.257 then this is in the interval [0.153, 0.453]

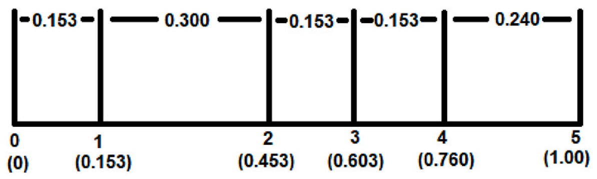


Fig. 2. Integer EMPSO adjusted space.

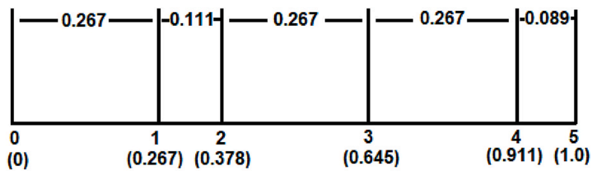


Fig. 3. The Proposed Integer PSO adjusted space.

so that the selected integer parameter is 2 [33].

Nevertheless, this method applies a space calculation for the Gbest and Pbest positions only using the multiplication to constant values c_3 and c_4 which will cause the Gbest position and Pbest position to grow fast and cause great opportunities for this position to be reelected. As a result of this fast growth, Gbest and Pbest candidates can be reelected because the space gets bigger as iterations grow. The growth of this space is essential so that Gbest and Pbest continue to exist in every iteration, but if it grows too fast, it will cause other potential solutions not to be selected. Then the optimization algorithm is stuck at the local minima and does not reach the global minima. Therefore, in this paper, an improvement had been made on the method of space calculation for each integer PSO iteration.

In this proposed method, the total position space that is not Gbest and Pbest is adaptive to changes according to the Gbest and Pbest position space. However, the space must be smaller than Gbest and Pbest space. It will allow sufficient space for non-Gbest and Pbest to be selected as PSO particles. Spaces that are not Gbest and Pbest are made gradually smaller as the number of iterations increases. The discrete variable on the j^{th} particle in the k^{th} iteration is $x_j^n = [x_{j1}^k, x_{j2}^k, ..., x_{jl}^k], x_{jd}^k \in \Omega_d$ where Ω_d is all discrete values that possible to be chosen as particles and $d = 1, 2, ...l$. The particle update process is described as follows:

1. Determine the number of possible discrete variable (integer) values as $|\Omega_d|$. For example, if $1 \le x_{j1}^k \le 5$ then $\Omega_d = \{1, 2, 3, 4, 5\}$ and $|\Omega_d| = 5$.
2. Calculate the initial spacing between all possible values $\lambda_{jd} = \frac{1}{|\Omega_d|}$.
3. Initialize random number between values 0–1. This random value is mapped to the discrete space value at step 2. The selected integer or discrete values are the initial particle value. Then calculate the cost function.
4. For example, the global optimal solution is the ith element of the set Ω_d and the optimal particle is the mth element, then the global best particle will be spaced using $\lambda'_{jd}[i] = c_3 \times \lambda_{jd}[i]$ and the best particle space will be updated using $\lambda'_{jd}[m] = c_4 \times \lambda_{jd}[m]$, where c_3 is the social-cognitive coefficient (used $c_3 = 1.5$) and c_4 is the self-cognitive coefficient (used $c_4 = 1.2$).
5. The Ω_d elements that are not Gbest and Pbest should have wide space at the initial iteration to give them a fair probability to be chosen as an optimal solution and then gradually get smaller as the number of iterations (k) increases. It is achieved by giving them total space (λ'_{left}) which is proportional to the space of Gbest ($\lambda'_{jd}[i]$)and Pbest ($\lambda'_{jd}[m]$) multiplied by the number of potential solutions ($|\Omega_d|$). To make their space get smaller as iterations increase, a division with the current number of iterations (k) is applied to them. A certain discount factor (in this case used 0.8 Eq. (3)) is applied to give the percentage of the space that is delivered to the non Gbest and Pbest. It also gives the ability in this method to reduce the Gbest space growth by giving it a value close to 1. The formula to calculate total space for non Gbest and Pbest is presented below (Eq. (3)).

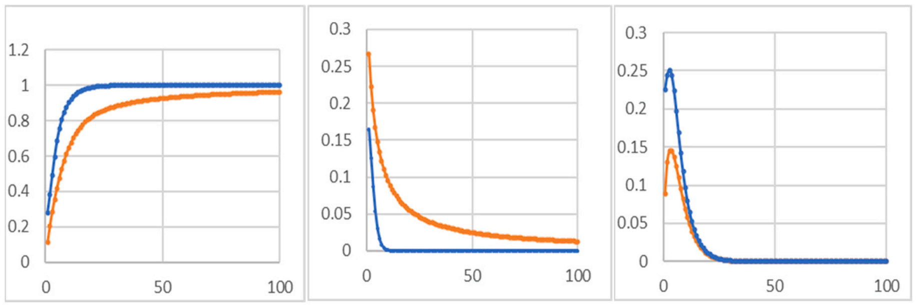
$$\lambda'_{left} = 0.8 \times \frac{\left(\lambda'_{jd}[i] + \lambda'_{jd}[m]\right) \times |\Omega_d|}{k} \tag{3}$$

Their individual non Gbest and Pbest space is then calculated by normalization (Eq. (4)).

$$\lambda'_{id}[j] = \frac{\lambda'_{id}[j]}{\sum_{j=1}^{|\Omega_d|} \lambda'_{id}[j]} \times \lambda'_{left}, (j = 1, 2, ..., |\Omega_d|, \text{except } i \text{ and } m) \tag{4}$$

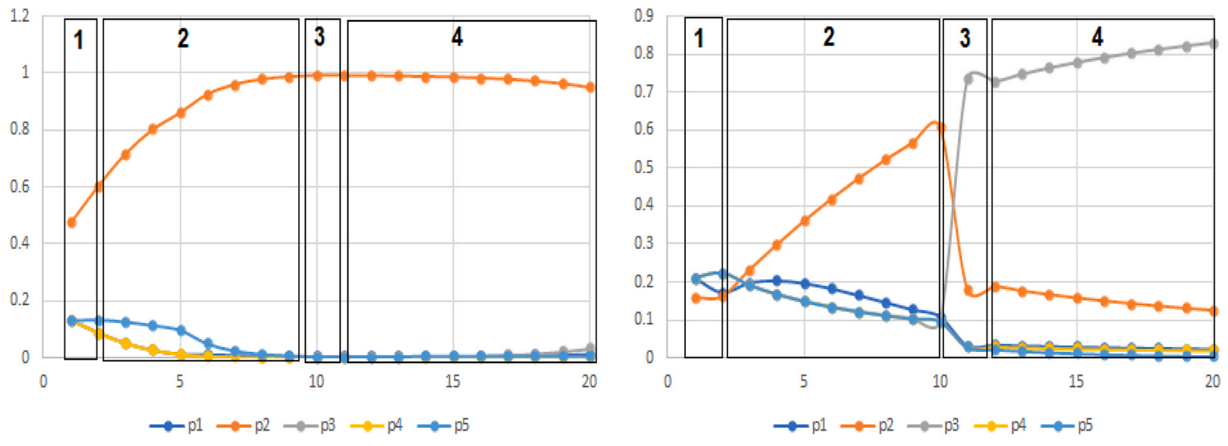
An example of the result of the proposed integer PSO update space from Fig. 1 and point 1 above is shown in Fig. 3 where parameter 2 is Gbest and 5 is Pbest.

Though in the first iteration the space of Gbest and Pbest is smaller but soon it will get wider as the iteration increase (see Fig. 4). It also gives the other potential solution more possibility in the first run. If these examples are extended the iteration to 100 iterations, a comparison of the changes in the space between the integer EMPSO and the proposed integer PSO will be obtained (see Fig. 4). Fig. 4



a. Gbest Solution b. Other parameters c. Pbest Solution

Fig. 4. Space adjustment of integer EMPSO (blue) and proposed integer PSO (orange), y-axis: solution space, x-axis: iteration. (For interpretation of the references to colour in this figure legend, the reader is referred to the Web version of this article.)



a. Integer EMPSO b. Proposed Integer PSO

Fig. 5. Four testing scenario at 20 iterations, y-axis: solution space, x-axis: iteration.

shows the space updating characteristics of the two methods if Gbest and Pbest are selected in long iterations. In Fig. 4a, it can be seen that the EMPSO Gbest space is rapidly approaching 1, which means that Gbest is consuming a lot of space quickly and this of course reduces the chances of another solution being selected in the next iteration. The proposed PSO integer is slower to increase so that it provides an opportunity for other solutions to be selected. This can also be seen in other parameters in Fig. 4b, namely the parameters that are not Gbest and Pbest. Space int EMPSO also approaches 0 faster so their chances of being selected are getting smaller. However, the proposed PSO integer provides more opportunities to be selected as a candidate solution in the next iteration. In the Pbest solution (Fig. 4c), these two methods show quite similar characteristics.

To further deepen the characteristics of these two methods, an algorithm test scenario is presented that represents the dynamics of its optimization. Five integer parameters are chosen to be tested on this scenario $\Omega_d = \{1, 2, 3, 4, 5\}$ which later will be denoted as $\{p1, p2, p3, p4, p5\}$. Here are some considerations:

1. The position of Gbest is the same as Pbest

This situation generally occurs at the beginning of the PSO iteration or when the new Gbest is selected, the Pbest position will be the same as Gbest.

2. Gbest position remains but Pbest changes

This situation describes an optimization process where Pbest changes but cannot get a better solution than Gbest.

3. New Gbest found, Gbest position is the same as Pbest position

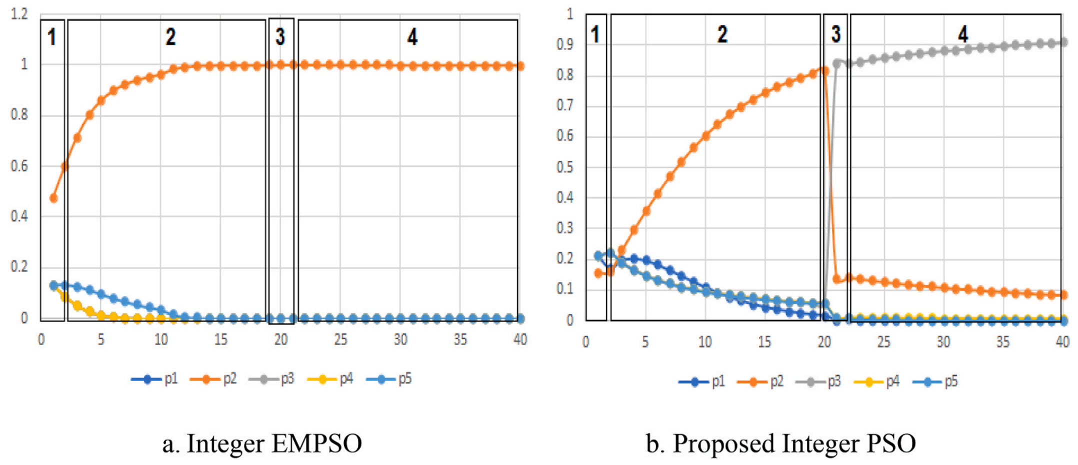


Fig. 6. Four Scenario testing at 40 iterations, y-axis: solution space, x-axis: iteration.

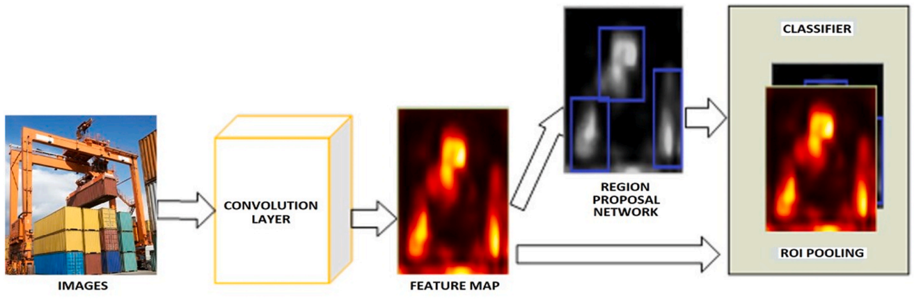


Fig. 7. Faster RCNN structure.

Similar to scenario 1 but this happens in the middle of the optimization process. It is important to know the change in space when in the middle of the iteration process a new Gbest is found.

4. Gbest position remains but Pbest changes

After that, the process continues on the search for another Gbest.

The test results of these 4 scenarios can be seen in Figs. 5 and 6. Figs. 5 and 6 show differences in space dynamics in an iteration range of 20 and 40 in the Integer EMPSO and Proposed Integer PSO methods. In scenario 1, the new Gbest and Pbest (*p2*) are selected so that the Gbest space increases. In scenario 2, the Pbest changes to *p1* but the Gbest (*p2*) remains the same, but the Gbest space grows very fast in scenario 2, the EMPSO Gbest is even close to the maximum value (Figs. 5a and 6a). As a result, the space value for the other parameters is reduced to close to 0. This will cause a small chance for other parameters to be selected as PSO particles so that other potential solutions are not likely to be obtained and optimization is stuck at the local minima. In the proposed PSO integer (Figs. 5b and 6b) the space increase is slower because it is given a total space of 80% (Eq. (3)) of the Gbest space used for other potential solutions. This allows the selection of other solutions but still maintains the weight of the Gbest space. The weight or probability of the Gbest space must be kept greater than the probability of other solutions so that in each iteration there is a Gbest particle.

In scenario 3, the Gbest value is changed to *p3* and the EMPSO integer method is difficult to reduce the previous Gbest space (Figs. 5a and 6a). This is because the previous Gbest space has been very close to 1 but the new Gbest space has previously fallen to close to 0. To increase the new Gbest space the EMPSO integer method uses a multiplier of $c_3 = 1.5$. However, if the previous space is close to 0 it will be very slow to increase the new Gbest space. In Fig. 5a the old Gbest space decreases in scenario 4 and is accompanied by an increase in the new Gbest space. However, this happened after 10 iterations after the new Gbest is found. On the other hand, the proposed integer PSO method (Figs. 5b and 6b) prioritizes the new Gbest space so that the new Gbest is more likely to be selected as a PSO particle in the next iteration. This can be seen when a new Gbest is obtained in scenario 3, the new Gbest space becomes larger. The old Gbest space will drop but will have a greater chance of being selected than the parameter that has not previously been Gbest because its space is wider. This is useful considering that the old Gbest may have characteristics that can lead to the selection of a more optimal Gbest in the next iteration.

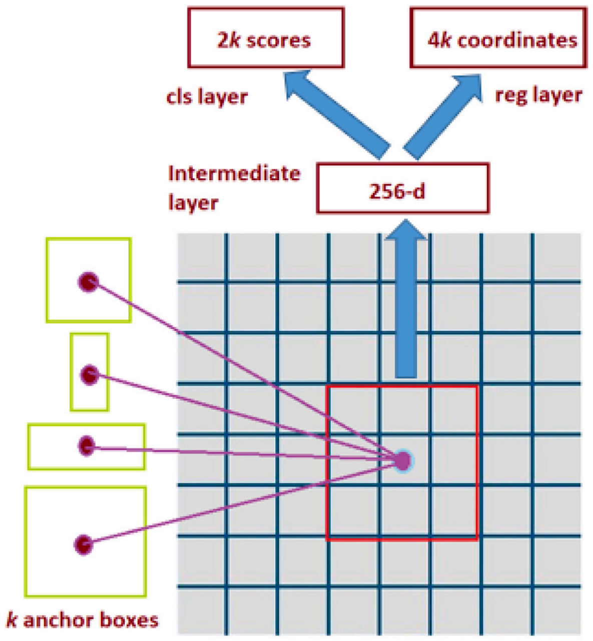


Fig. 8. Region proposal network (RPN).

Table 1
Comparison of New Proposed integer PSO and integer EMPSO for optimization of nine test functions.

No	Test Function	Target Position (x_1, x_2)	Target Cost $f(x)$	Optimization Method					
				New Proposed Int PSO			Int EMPSO		
				Gbest Pos	Gbest Cost	Iteration Target Reached	Gbest Pos	Gbest Cost	Iteration Target Reached
1	Auckley Function	(0,0)	0	(0,0)	0	17	(0,0)	0	63
2	Bukin Function N.6	(-10,1)	0	(0,0)	0.1	–	(-40,16)	0.3	–
3	Drop Wave Function	(0,0)	−1	(0,0)	−1	62	(0,0)	−1	21
4	Rastrigin Function	(0,0)	0	(0,0)	0	4	(0,0)	0	6
5	Sum of Different Powers	(0,0)	0	(0,0)	0	15	(0,0)	0	28
6	Trid Function	(2,2)	−2	(2,2)	−2	8	(2,2)	−2	27
7	Levy Function N.13	(1,1)	0	(1,1)	0	50	(2,1)	1	–
8	Three-Hump Camel	(0,0)	0	(0,0)	0	15	(2,-1)	0.867	–
9	Rosenbrock Function	(1,1)	0	(1,1)	0	88	(5,25)	16	–

3. Faster RCNN

Faster RCNN consists of two modules. The first module is a convolution network that proposes a region (Region Proposal network, RPN), and the second module is a Fast RCNN detector that uses the proposed region. The whole system is one unified network for object detection (Fig. 7). This method uses a neural network that has an 'attention' mechanism, namely, the RPN module that tells the Fast RCNN module where the object is likely to be.

A small network is shifted over the convoluted feature maps by the last convolutional layer to generate the proposed region. This small network takes as input an $n \times n$ spatial window of feature maps. Each sliding window is mapped to a feature with less dimensions. This feature is incorporated into two fully connected layers, namely the box regression layer (*reg*) and the box classification layer (*cls*) [14,15].

At each location of the sliding window, several regional proposals were simultaneously predicted, where the maximum possible number of proposals for each location was denoted as k. So the *reg* layer has a $4k$ output encoding the coordinates of k squares, and the *cls* layer outputs a $2k$ score estimating the object or non-object probability for each proposal. The proposal parameter k is created relative to a reference called the anchor. The anchor is centered in the observed sliding window and varies by scale and ratio (Fig. 8). By default, 3 scales and 3 ratios are generally used so that $k = 9$ anchors are obtained at each position shift. Therefore, the feature maps of the convolution process of size $W \times H$ obtained anchor $W \times H \times k$ in total.

Table 2
Faster RCNN continuous parameter.

Parameter	Range Parameter [xmin, xmax]	Default Parameter	Optimized Parameter
Scales	[(0.15,0.40,0.8,1.7), (0.35,0.60,1.2,2.2)]	(0.25, 0.5, 1.0, 2.0)	(0.282, 0.500, 0.905, 2.150)
Aspect_ratio	[(0.4,0.8,1.8), (0.6,1.2,2.2)]	[0.5, 1.0, 2.0]	(0.539, 1.145, 1.937)
Weight l2_regulizer first stage	[0,0.001]	0.0	0
stddev Initializer	[0.008,0.02]	0.01	0.019
first_stage_nms_score_threshold	[0,0.01]	0.0	0.003
first_stage_nms_iou_threshold	[0.65,0.75]	0.7	0.705
first_stage_localization_loss_weight	[1.8,2.2]	2.0	1.893
first_stage_objectness_loss_weight	[0.8,1.2]	1.0	0.884
dropout_keep_probability	[0.95,1]	1.0	0.982
Weight l2_regulizer 2nd stage	[0,0.001]	0.0	0.00059
Factor	[0.9,1.1]	1.0	1.081
score_threshold 2nd stage	[0,0.001]	0.0	0.0009
iou_threshold 2nd stage	[0.55,0.65]	0.6	0.593
second_stage_localization_loss_weight	[1.8,2.2]	2.0	1.855
second_stage_classification_loss_weight	[0.8,1.2]	1.0	0.952

Table 3
Faster RCNN integer/discrete parameter.

Parameter	Range Parameter	Default Parameter	Optimized Parameter
first_stage_max_proposals	[290,310]	300	293
initial_crop_size	[12,16]	14	12
maxpool_kernel_size	[1,4]	2	2
maxpool_stride	[1,4]	2	2
use_dropout	[False, True]	False	True
Uniform	[False, True]	True	True
max_detections_per_class	[90,110]	100	91
max_total_detections	[290,310]	300	299

After obtaining the RPN, the network used for object detectors is Fast RCNN. Each object proposal will be processed by the Region of Interest (RoI) layer, which extracts feature vectors with a fixed length. Each feature vector is processed by fully connected layers, which eventually gives two output layers: one that produces an estimate of the softmax probability over object class K, and the other layer produces four real numbers for each object class K. Each of these four values gives information about the position of the bounding box corresponding to the object class K for one of the K classes [14,15].

Scale and aspect ratio are examples of Faster RCNN parameters that need to be optimized. Other parameters need to be optimized to obtain a higher detection accuracy. Some of these parameters are continuous, and others are integer or discrete parameters, see Table 2 and Table 3. Therefore, it is necessary to apply mixed-integer optimization. This paper develops the PSO mixed-integer method with a weighted cost function applied to Faster RCNN to detect containers at the port. Further discussion of the proposed method can be found in the next section.

4. Methodology and experiments

4.1. Evaluation of the proposed integer PSO

The PSO integer method developed has been described in Section 2. This method will be used to optimize several non-linear test functions to test its ability to optimize integer parameters. Here are some of the used test functions.

1. Auckley Function (Eq. (5)) [37].

$$f(x,y) = -20\exp\left[-0.2\sqrt{0.5(x^2+y^2)}\right] - \exp[0.5(\cos 2\pi x + \cos 2\pi y)] + e + 20 \tag{5}$$

2. Bukin Function N.6 (Eq. (6)) [37].

$$f(x) = 100\sqrt{|x_2^2 - 0.01x_1^2|} + 0.01|x_1 + 10| \tag{6}$$

3. Drop Wave Function (Eq. (7)) [38].

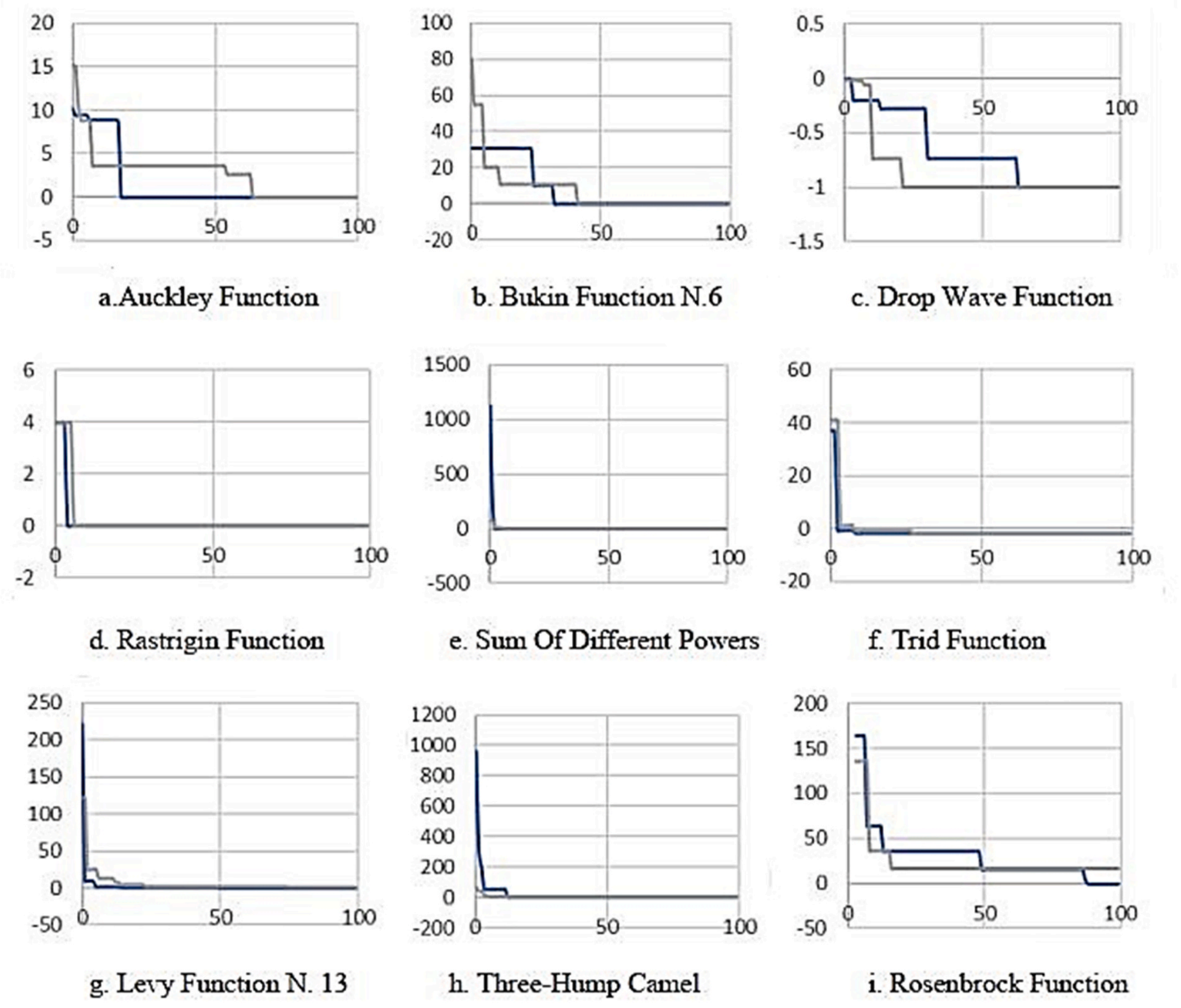


Fig. 9. Comparison of the proposed int PSO (Blue) and int EMPSO (Grey), y-axis: $f(x)$, x-axis: iteration. (For interpretation of the references to colour in this figure legend, the reader is referred to the Web version of this article.)

$$f(x) = -\frac{1+\cos\left(12\sqrt{x_1^2+x_2^2}\right)}{0.5(x_1^2+x_2^2)+2} \tag{7}$$

4. Rastrigin Function (Eq. (8)) [38].

$$f(x) = 10d + \sum_{i=1}^{d}\left[x_1^2 - 10\cos(2\pi x_i)\right] \tag{8}$$

5. Sum of Different Powers Function (Eq. (9)) [39].

$$f(x) = \sum_{i=1}^{d}|x_i|^{i+1} \tag{9}$$

6. Trid Function (Eq. (10)) [37].

$$f(x) = \sum_{i=1}^{d}(x_i-1)^2 - \sum_{i=2}^{d}x_i x_{i-1} \tag{10}$$

7. Levy Function N.13 (Eq. (11)) [40].

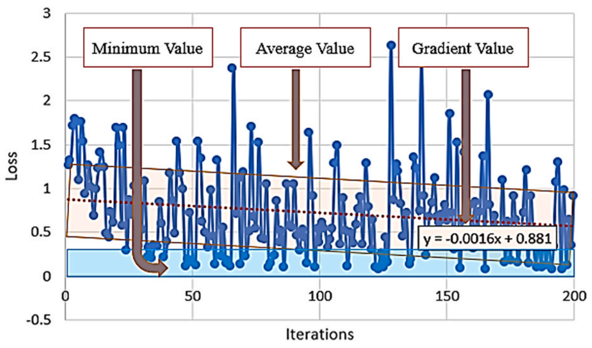


Fig. 10. Default parameter loss graph at steps 0-200.

$$f(x) = \sin^2(3\pi x_1) + (x_1 - 1)^2\left[1 + \sin^2(3\pi x_2)\right] + (x_2 - 1)^2\left[1 + \sin^2(2\pi x_2)\right] \tag{11}$$

8. Three-Hump Camel Function (Eq. (12)) [37].

$$f(x) = 2x_1^2 - 1.05x_1^4 + \frac{x_1^6}{6} + x_1x_2 + x_2^2 \tag{12}$$

9. Rosenbrock Function (Eq. (13)) [37].

$$f(x) = \sum_{i=1}^{d-1}\left[\beta\left(x_{i+1} - x_i^2\right)^2 + (x_i - 1)^2\right], \quad \left\{\beta \in \mathbb{R} : \ 100 \le \beta \le 105, \text{ in this case } \beta = 100\right\} \tag{13}$$

The test results of these test functions are presented in Table 1 and Fig. 9(a)–(i). From the 9 test functions, the proposed method gives 8 better optimization results than integer EMPSO (Table 1 No. 1–2 & 4–9; Fig. 9(a, b and d–i)). From these 8 advantages, 7 succeeded in achieving the target position and the number of iterations needed to achieve the target was smaller than integer EMPSO. Both methods cannot achieve the optimization target on the Bukin function (Table 1 No. 2; Fig. 9b), but the proposed method is superior because it has a cost value that is closest to the target. Even though the integer EMPSO drop wave function (Table 1 No. 3; Fig. 9c) is faster to reach the target cost, the proposed method also reaches the target cost in the 62nd iterations. On the other hand, integer EMPSO fails to reach the target in 4 test functions out of 9 (Table 1 No. 2 & 7–9; Fig. 9(b and g–i). This shows that the proposed method is more efficient and guarantees the target is achieved than integer EMPSO. This occurs due to the adaptive nature of the proposed space of the non-optimal solution that is updated according to the space of optimal position solution and gradually becomes smaller as the iterations increase. On the other hand, integer EMPSO when it encounters complex functions and many local minima will tend to increase its optimal space on local minima and eventually reduce its ability to explore other possible solutions and get stuck on local minima. Based on the overall test, it can be concluded that the proposed integer PSO method has better performance than the integer EMPSO method.

4.2. Faster RCNN optimization

Faster RCNN has been widely used as an object detection method. TensorFlow also released the faster RCNN object detection module. This object detection module is generally applied using the default parameter configuration, both continuous and integer parameters. In this paper, mixed-integer optimization will be applied to the Faster RCNN parameters contained in the configuration file of the TensorFlow Faster RCNN module. The object to be detected in this paper is the container inside the image data. One thousand three hundred container image data had been collected from google search. Python code from labelimg. py is used to label these images, which assigns the container's location in the image based on its minimum and maximum pixel value on the x-axis and y-axis. Those image data are then divided into 80% as data for training and 20% for testing.

Table 2 shows 20 continuous parameters and eight integer/discrete parameters that will be optimized. Due to the high computational load, optimization will be carried out on faster RCNN with 200 steps of training. Mixed-integer PSO was applied using ten particles and 50 iterations. The range of parameter optimization values can be seen in Tables 2 and 3. Before applying Faster RCNN optimization using PSO mixed-integer, it is necessary to determine the optimization cost function. The Faster RCNN training method generally uses the loss value as an indicator of the performance of the Faster RCNN. However, it is unfair to only use the loss value in the last step, 200, as an indication of the performance of a PSO particle. Fig. 10 shows the stochastic nature of the Faster RCNN loss value during the training process, where the value changes at each step. Therefore, a new communal assessment is proposed to measure the model's performance on the whole system. Three indicators are used, namely the minimum loss, average loss, and gradient loss (Fig. 10).

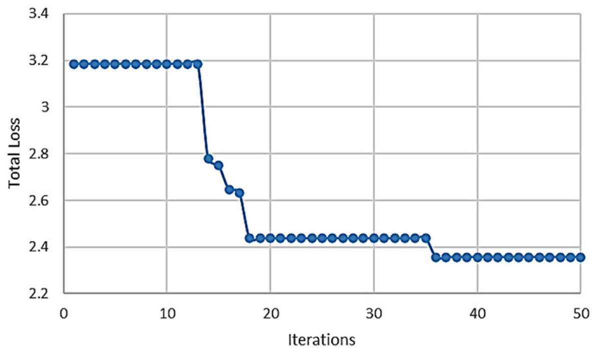


Fig. 11. Gbest history on Mixed-Integer PSO Optimization of Faster RCNN.

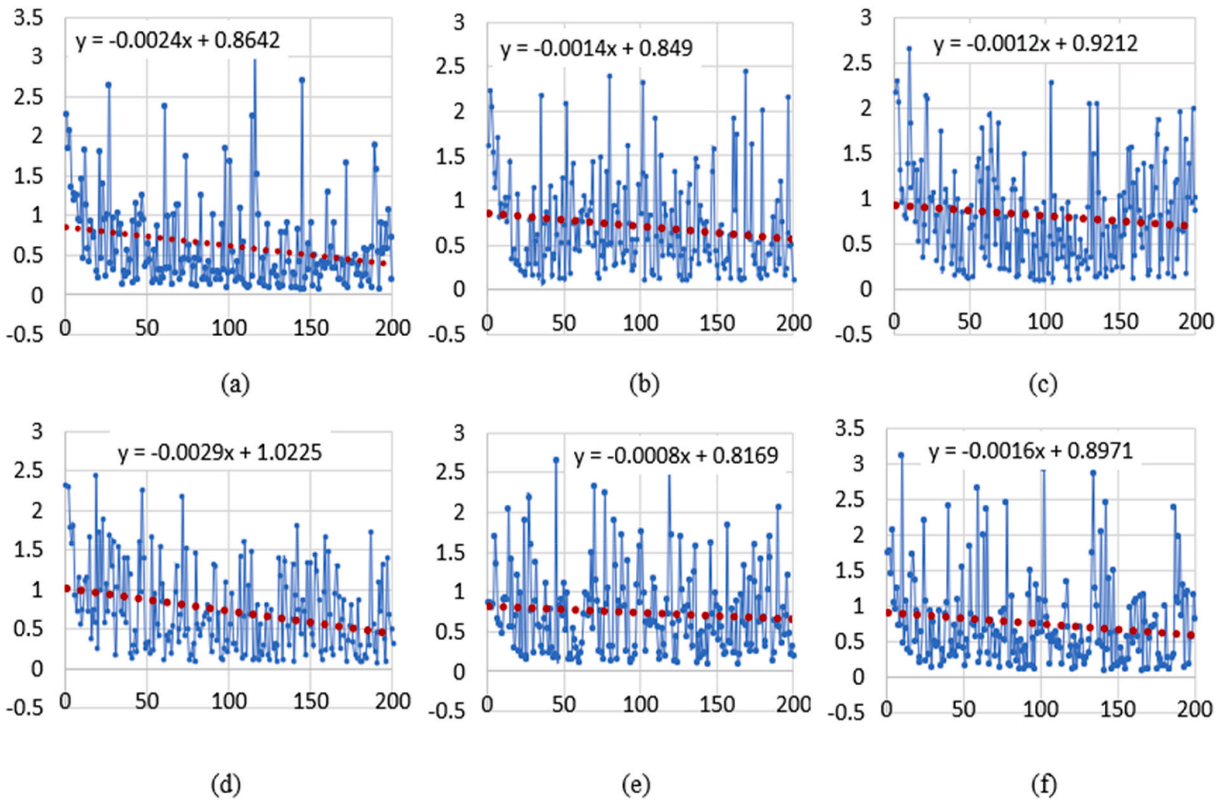


Fig. 12. Six Graphs of 200 steps default parameters, y-axis: loss, x-axis: iteration.

1. Minimum loss

The minimum loss indicates how capable the model is to achieve the smallest loss in a given range. Because the training loss tends to change with each step, it is necessary to look at the minimum value that represents the smallest loss achieved by the model.

2. Average loss

The loss value that changes at each step requires knowing the average loss in the training process. Average loss describes the general loss value achieved by the model. A small average value indicates that the loss system value is generally small and the data variation is smaller.

3. Gradient loss

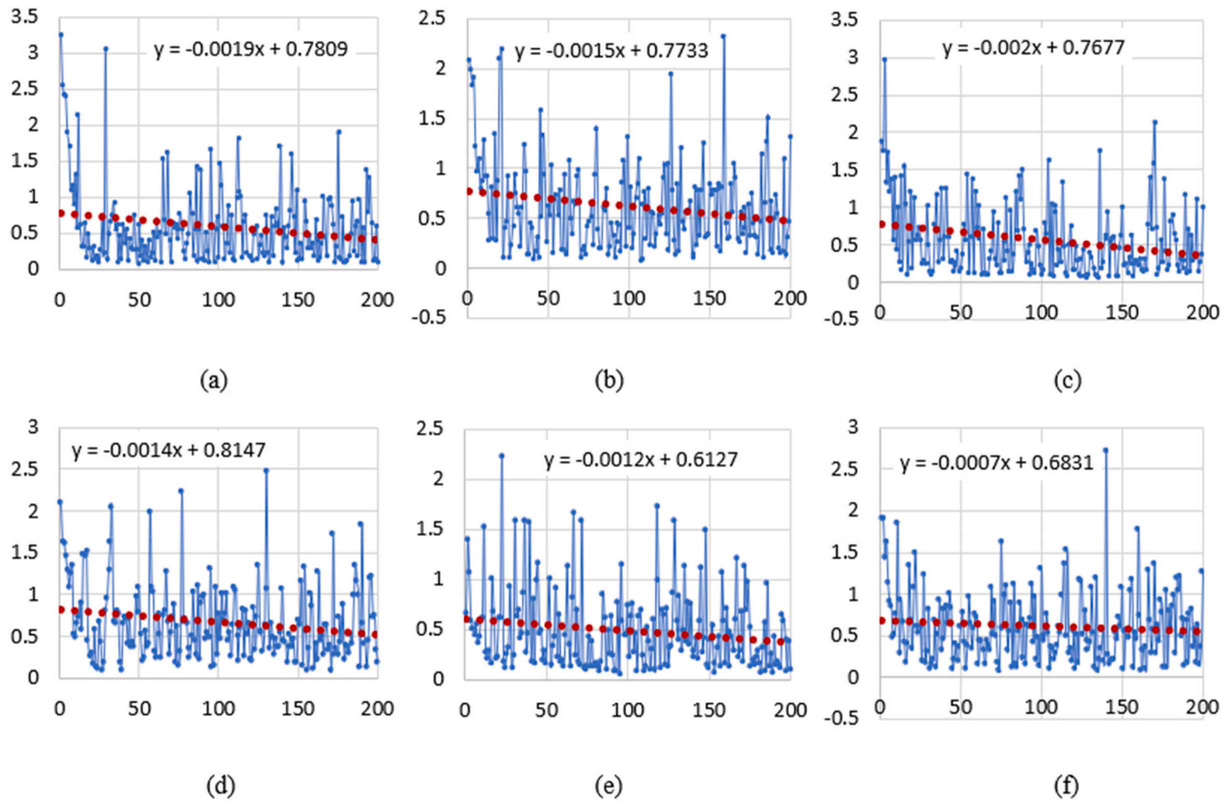


Fig. 13. Six Graphs of 200 steps optimized parameters, y-axis: loss, x-axis: iteration.

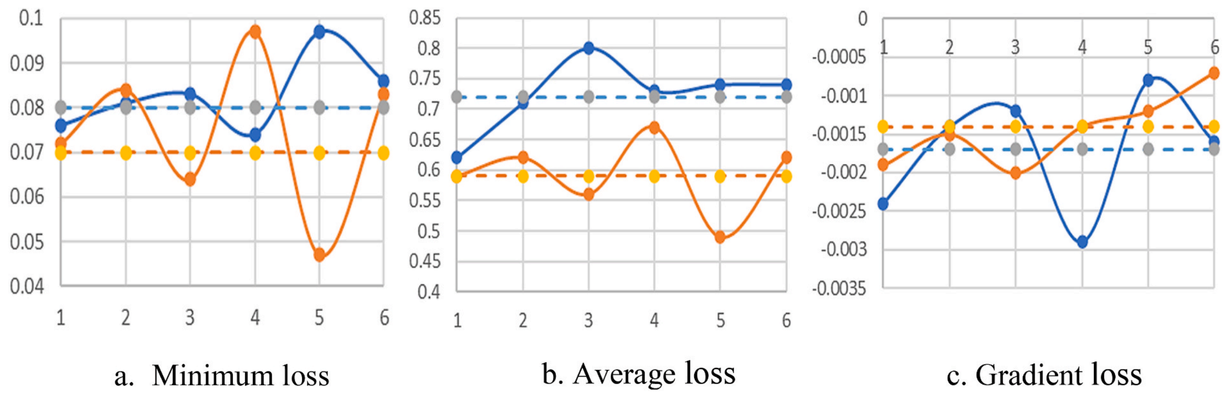


Fig. 14. Performance comparison of optimized (orange) and default parameter (blue), dashed line: average value, y-axis: loss, x-axis: experiment. (For interpretation of the references to colour in this figure legend, the reader is referred to the Web version of this article.)

The gradient loss provides information about the model's ability to reduce the loss value. A large negative loss gradient value indicates a large momentum in decreasing loss. It is assumed that if the loss gradient is a larger negative value, then the loss probability will be smaller in the next steps. A constraint value is also applied on the selection of Gbest and Pbest that they must have a negative gradient value.

These 3 objectives are indicators that need to be minimized so that this problem becomes a multiobjective mixed-integer optimization. Based on these 3 indicators, a cost function for optimization is proposed which is formulated as follows (Eq. (14)).

$$Total\ loss = w_m \frac{m}{m_{def}} + w_a \frac{a}{a_{def}} + w_g \frac{g_{def}}{g} \tag{14}$$

where:

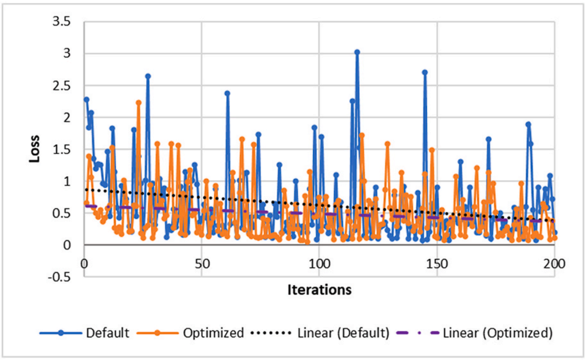


Fig. 15. Comparison result of default and optimized parameters.

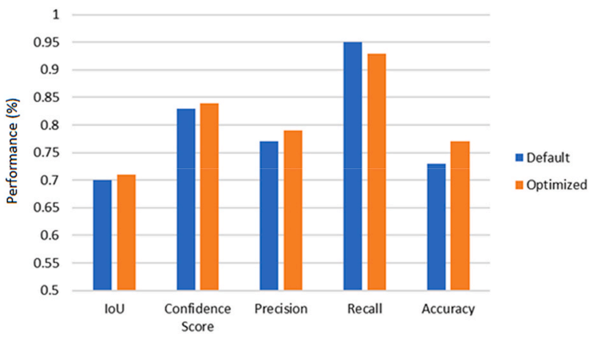


Fig. 16. Detection performance comparison.

m = minimum loss
a = average loss
g = gradient loss
m_{def} = minimum loss when using default parameter
a_{def} = average loss when using default parameter
g_{def} = gradient when using default parameter
w_m = minimum loss weight
w_a = average loss weight
w_g = gradient loss weight

Each indicator is compared against its default performance value in order to obtain a ratio value that eliminates the difference in the magnitude of each indicator. For example, the values of $m = 0.1$, $a = 0.8$ and $g = -0.0001$, all three have different value ranges. Only adding these losses value will reduce the effect of m to the total loss, even more g, because they are generally smaller than a. Therefore, it is necessary to normalize so that these three values are in the same value range. The case is solved by dividing it against performance when using default parameters.

In addition, each indicator is given a weight to give the researcher the freedom to choose which objective is preferred. If the optimization goal is to minimize the average value as the main objective, then a larger weight is given to the average loss ratio element. Without the use of weight, it will cause one element to be minimized but by compromising the value of the other which tends to be unwanted by the researcher. The use of weight allows researchers to give priority to the optimization of the three indicators. The maximum value for each weight is 1. In this paper, the order of priority is the minimum loss value ($w_m = 1$), average loss ($w_a = 0.85$), and gradient loss ($w_g = 0.60$).

In addition to the cost function, it is also necessary to pay attention to the procedure for the optimization process. Faster RCNN optimization is an optimization that involves deep learning architecture. The Faster RCNN training process also inherits the nature of deep learning which is based on the principle of probability and is stochastic. In contrast to the general method, for example, optimization of PID controller parameters, which always gives the same cost function results when using the same combination of the proportional gain, integral time and derivative time on the same dynamic system, Faster RCNN will provide different cost function values even if tested using the same parameter. Therefore, the value of the cost function in one test can not be used as a description of the detection performance for a certain combination of parameters. To achieve a better representative cost function on a set of parameters, the average value will be taken on several tests. In this paper, only two testings were used, and then the average of these two

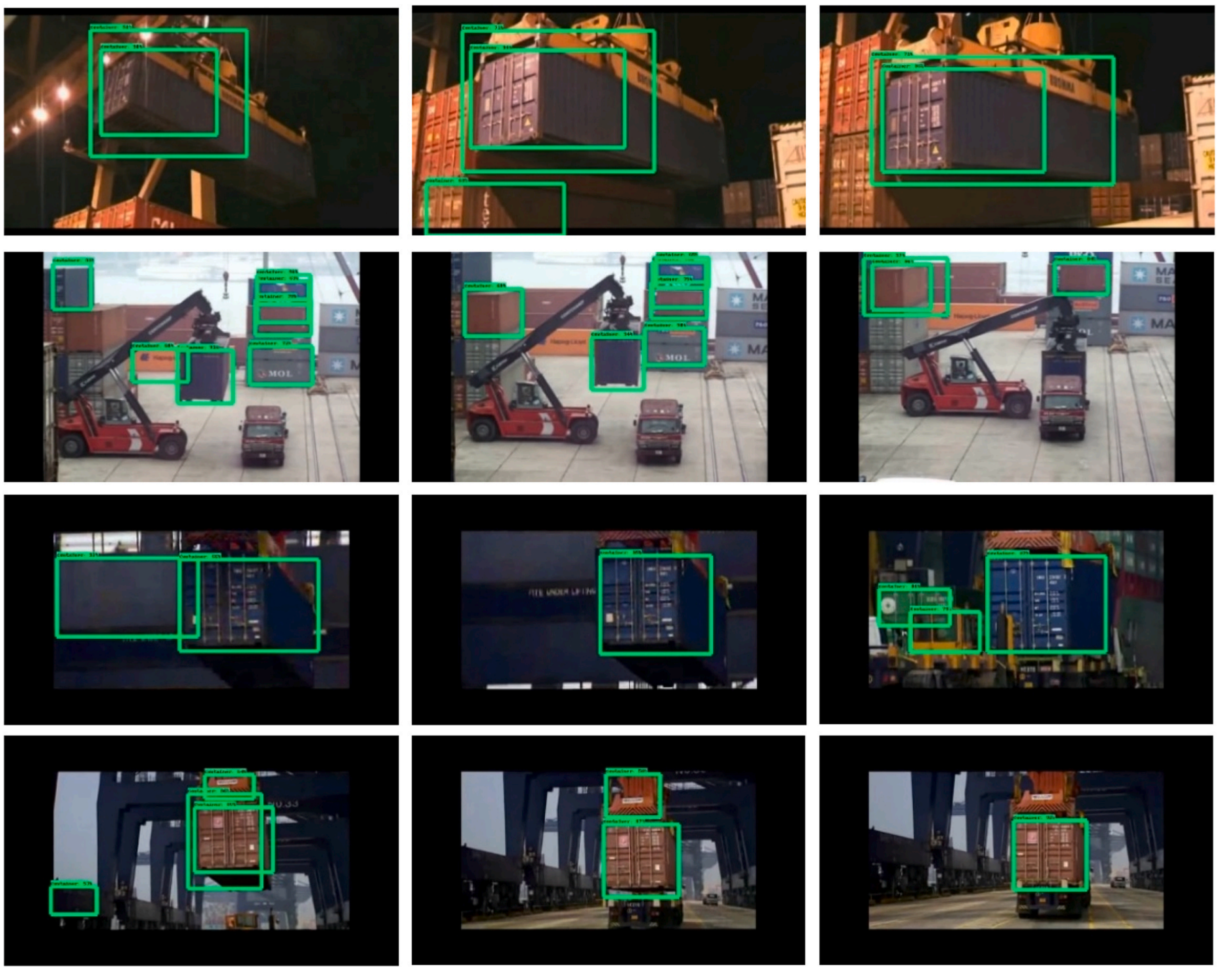


Fig. 17. Detection result at 2000 step using default parameters (1st–4th rows: 1st–4th different video; 1st– 3rd columns: 3 frames from each video).

costs was taken. So, when the mixed-integer PSO creates a particle combination of Faster RCNN parameters to be tested, a combination of parameters will be trained twice so that two values of m, a and g are obtained and then averaged so that a more robust cost function value is obtained representing the performance of Faster RCNN in that parameter combination.

To avoid selecting the wrong Gbest, a constraint is applied as the selection criteria for Gbest by assuming that the optimal parameter must have a negative loss gradient in the 200-step training range. This treatment is applied so that the optimization momentum is maintained for the next steps, namely, there is a consistent decrease in the loss in the following steps. Therefore, a potential solution that even though it gives a small minimum and average loss but has a positive gradient will not be selected as Gbest.

Fig. 11 shows the Gbest history during the optimization process using mixed-integer PSO. This method has succeeded in showing a decrease in the loss value in the continuation of the iteration stages. Optimal Gbest is obtained in the 36th iteration where the optimal loss obtained is 2.38. Tables 2 and 3 show the changes in the previous default parameter values which are then optimized within the given parameter range.

After obtaining the optimal parameters, the test was carried out 6 times and compared the results with those using the default parameters. The test was carried out 6 times to see the optimization performance, in general, considering the stochastic character of Faster RCNN in the training process. Three indicators, namely minimum loss, average loss, and gradient value will be a measure of the performance of the two models. The results are shown in Figs. 12, 13, and 14. All Fig. 12(a–f) and 13(a–f) give negative gradient which confirm their ability to minimize the loss. Fig. 12(c–f) shows the start of a high red regression line at a value of 1.0, and in Fig. 12 (a, b), the value is close to 1.0. Fig. 13(a–f) shows a better start to the training process; namely, the value of all the regression line tends to start below 1.0, even Fig. 13e, f at 0.5. Besides that, during the 200 steps training process, Fig. 12(a–f) often touches a loss value above 1.5; pay attention to the number of points that exceed the loss value of 1.5. Only images 12 (a, d) are relatively few, which exceeds the 1.5 loss. Contrast when compared to the training process in Fig. 13(a–f), which all give smaller loss values and few touch values above 1.5.

Fig. 14 a shows that the minimum loss value in the optimized parameter (m_O) is smaller $m_O = \{$ 0.047, 0.064, 0.072, 0.083, 0.084, 0.097) than the default $m_D = \{0.074, 0.076, 0.081, 0.083, 0.086, 0.097\}$. Based on the average loss value (Fig. 14b), the optimized

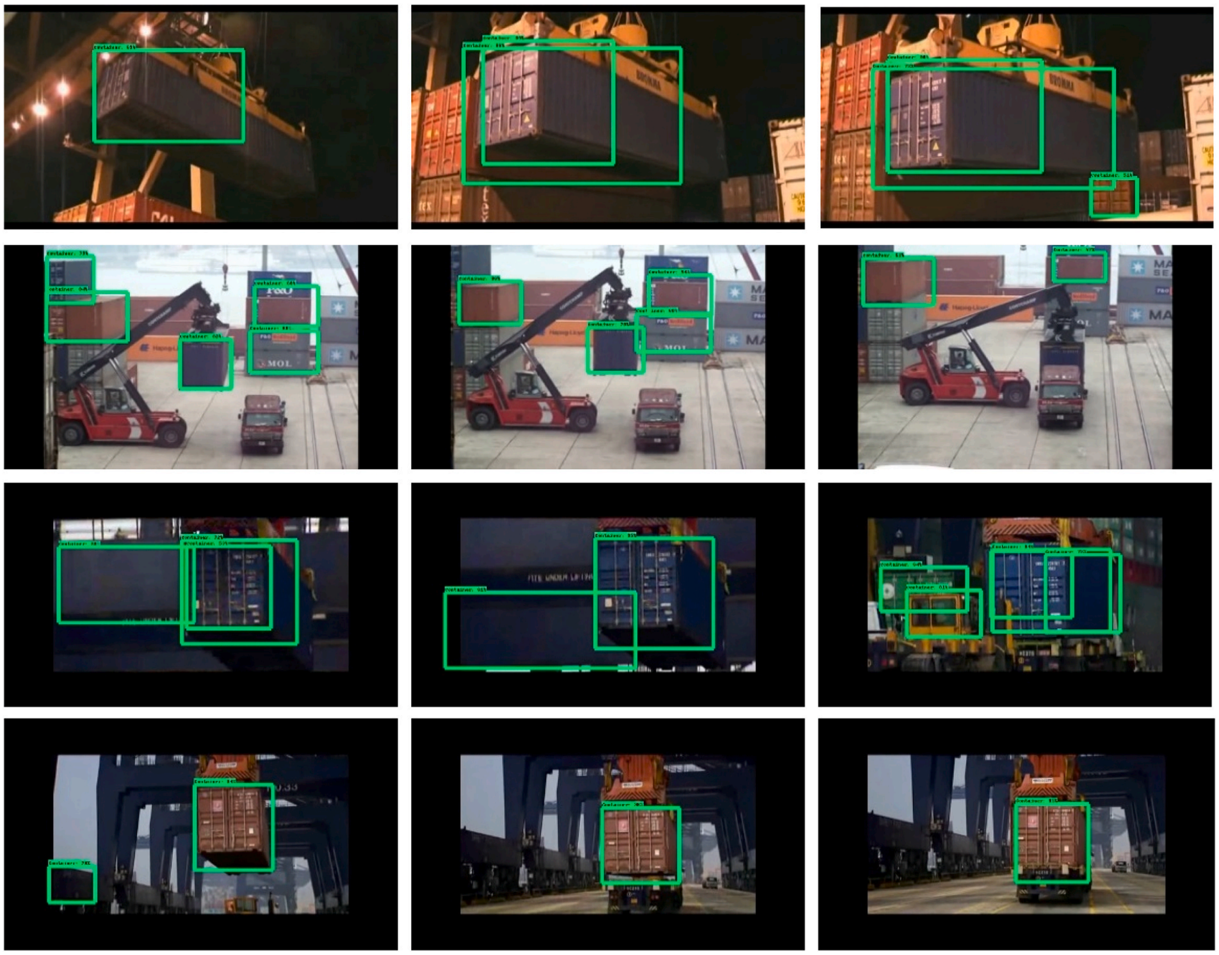


Fig. 18. Detection result at 2000 step using optimized parameters (1st–4th rows: 1st–4th different video; 1st–3rd columns: 3 frames from each video).

parameter (a_O) also gives a smaller result value a_O = {0.49, 0.56, 0.59, 0.62, 0.62, 0.67} compared to the default average loss parameter a_D = {0.62, 0.71, 0.73, 0.74, 0.74, 0.80}. In terms of the gradient loss value (Fig. 14c), the default parameter has a lower negative gradient value g_D = {-0.0008, −0.0012, −0.0014, −0.0016, −0.0024, −0.0029} than the gradient optimized parameter g_O = {-0.0007, −0.0012, −0.0014, −0.0015, −0.0019, −0.0020}. This is because when optimization is given a smaller priority or weight on gradient loss optimization. The minimum loss and average loss values are given higher priority/weight so that the optimization results in both indicators provide better values than the default parameters.

The average value of the minimum loss and average loss (Fig. 14; dashed line) is also smaller in the optimized parameter. Similar results are also seen in Figs. 12 and 13 where the distribution of loss values using optimized parameters is smaller than the default one. Fig. 15 shows a comparison between the best-optimized parameters and the best when using the default parameters. Optimized parameters can touch minimum error values that are smaller than the default parameters. In addition, because the average value is smaller, generally the optimized parameters loss values are spread below the default parameter loss value. This shows that the optimization results provide better loss characteristics than the default parameters.

Tests on the optimization results are also applied to 4 videos [41–44] that had never been trained on this Faster RCNN model. These videos are chosen based on their environment where located at the seaport that fit to the application of container detection for the seaports automation. There are also variations of light intensity from low light at the night (Figs. 17–20, 1st row), medium (Figs. 17–20, 3rd-4th row), to high at the daylight (Figs. 17–20, 2nd row). Observing Figs. 17–20, each row of the figure is the result of detection from one video. Three random samples are collected from each video which are then compared to their detection capabilities at 2000 steps and 5000 steps of training. These samples represent different point of view which enrich the testing scenario. In Figs. 17–20, 1st row 1st column figure gives the view of container from below, 1st row 2nd-3rd column & all column in 3rd row give the side view of container, 2nd row 1st-3rd column give the view of container from afar (especially 3rd column gives a partially occluded container view), and all columns in 4th row give the front view. In step 2000, the results provided are not very stable, there are some overlapping boxes because the number of steps is only 2000. The default and optimized models give similar results, both had difficulties distinguishing between the blue container and the blue box background in video 3 (Figs. 17 and 18, 3rd row).

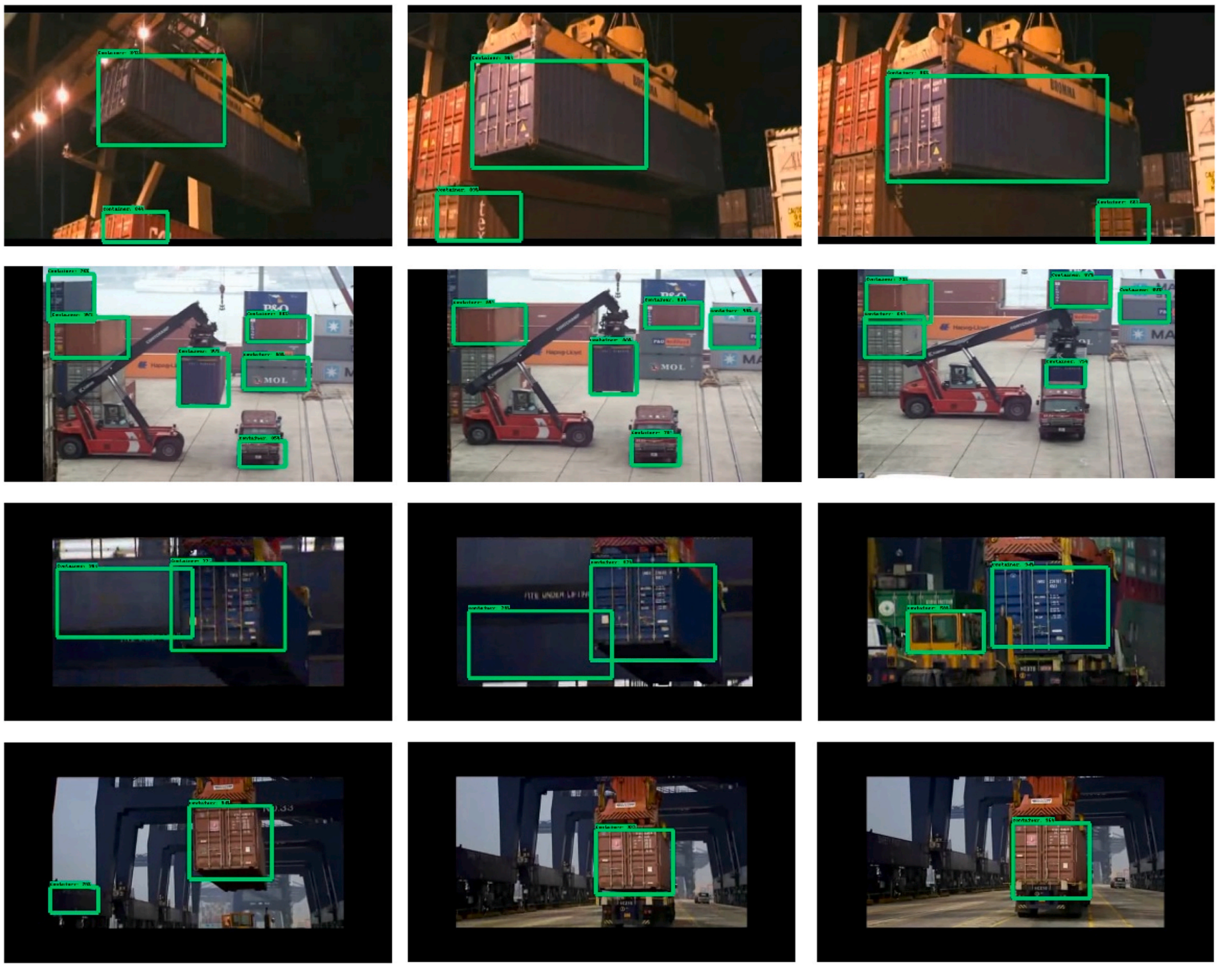


Fig. 19. Detection result at 5000 step using default parameters (1st–4th rows: 1st–4th different video; 1st– 3rd columns: 3 frames from each video).

However, there are several other errors in the default model detection results that do not occur in the optimized model. In the default model, some objects are not containers but detected as containers (Fig. 17 2nd row 1st column, 4th row 1st column, 4th row 2nd column; Fig. 19 2nd row 2nd column, 4th row 1st column). Although the optimization model also produce some errors, there were more errors in the default model. The errors in the optimized model also occurred in the default model. Therefore, the obtained results shows that the optimized model detection results are better than the default model. On the other hand, the model is optimized using mixed-integer PSO at step 200 but it was still tested well at steps 2000 and 5000. For better results, the model can be optimized at steps 2000 and 5000 as well. Future research can test the detection accuracy when optimized at a higher step.

The comparison of the accuracy of the detection results in the four videos above can also be expressed into several accuracy parameters such as Intersection over Union (IoU), confidence score, precision, recall and accuracy. IoU is calculated by dividing the intersection area between the predicted bounding box BBp and the bounding box BBgt by the area of union between them (Eq. (15)). Precision is a model's ability to identify only relevant objects. It is the percentage of correct positive predictions (Eq. (16)). A model's recall is its ability to find all relevant cases (all ground-truth bounding boxes) (Eq. (17)). It is the proportion of correct positive predictions out of all possible ground truths. The model's confidence scores reflect how certain the box contains an object is and how accurate it believes the box is that it predicts. If there is no object in that cell, the confidence scores should be zero. Confidence score is the output of the model calculated by the faster RCNN network. Accuracy is an evaluation metric to count the number of correct predictions made by a model (Eq. (18)) [45].

$$IoU = \frac{\sigma\left(BB_p \cap BB_{gt}\right)}{\sigma\left(BB_p \cup BB_{gt}\right)} \tag{15}$$

$$Precision = \frac{TP}{TP + FP} \tag{16}$$

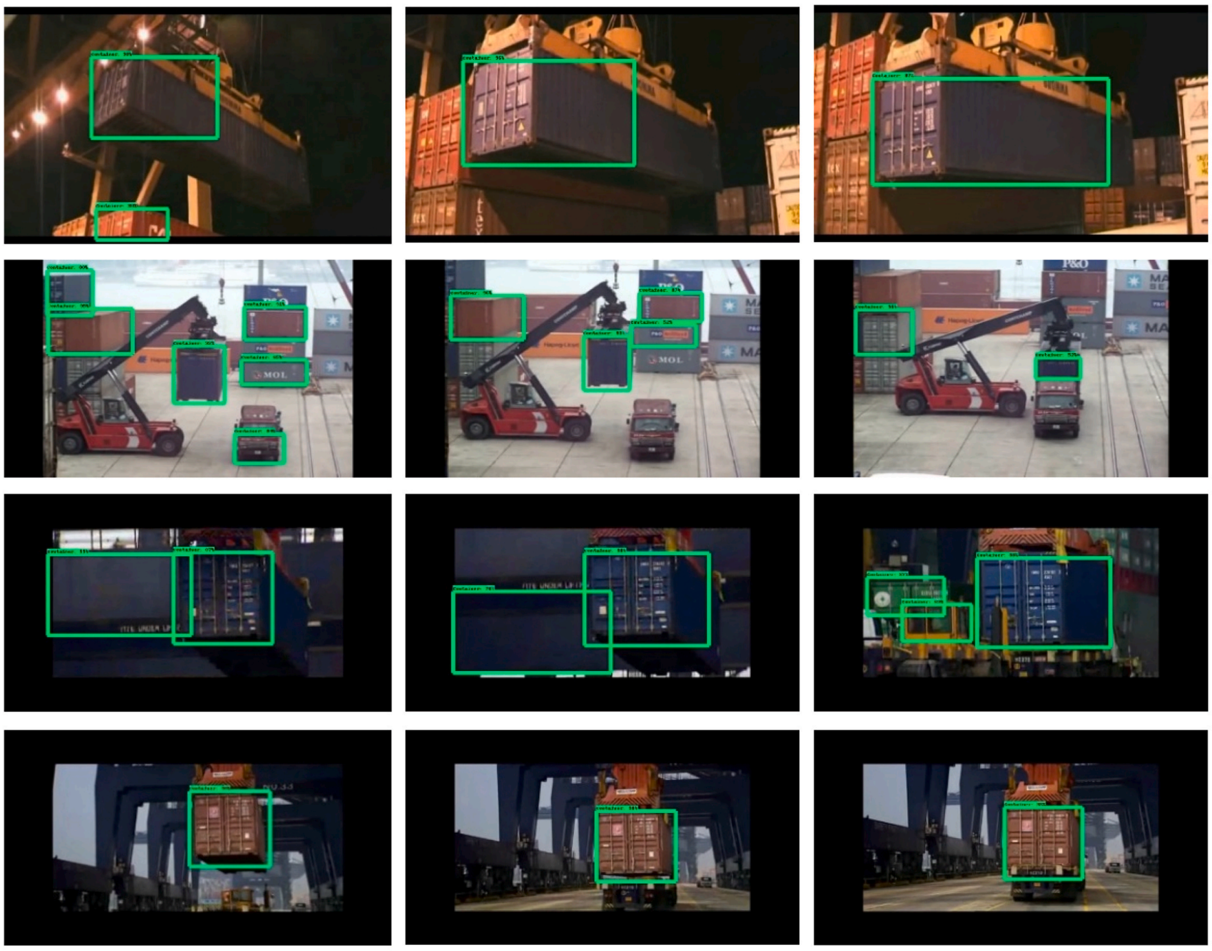


Fig. 20. Detection result at 5000 step using optimized parameters (1st–4th rows: 1st–4th different video; 1st–3rd columns: 3 frames from each video).

Table 4
Faster RCNN Optimization results.

Video	Number of Frames	Default					Optimized				
		a	b	c	d	e	a	b	c	d	e
1	838	0.66	0.83	1.00	0.89	0.89	0.67	0.70	0.85	0.89	0.81
2	1294	0.55	0.79	0.88	0.93	0.83	0.58	0.80	0.96	0.85	0.88
3	250	0.78	0.81	0.60	1.00	0.60	0.80	0.89	0.64	1.00	0.70
4	154	0.82	0.90	0.61	1.00	0.61	0.81	0.97	0.70	1.00	0.70
Average		0.70	0.83	0.77	0.95	0.73	0.71	0.84	0.79	0.93	0.77

Where: a: IoU (%), b: Confidence Score (%), c: Precision (%), d: Recall (%), e. Accuracy (%).

$$Recall = \frac{TP}{TP + FN} \tag{17}$$

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{18}$$

where σ is a function for calculating area. The predicted bounding box is denoted by BB_p, while the ground truth bounding box is denoted by BB_{gt}. True Positive (TP) refers to the number of times a detector successfully identifies an item, while False Positive (FP) refers to the number of times a detector incorrectly detects an object. False Negative is the total number of events in which the detector erroneously identifies ground truth as non-objects (FN). True Negative (TN) is the total number of times the detector correctly detects a non-object as a non-object.

IoU is generally used as a hyperparameter threshold criterion [46,47] at the training stage which determines whether the anchor (see Section 3) created is an object based on overlapping areas with ground truth. This is a common strategy in object detection where each ground truth is assigned one or more anchors and if the IoU exceeds a certain threshold then the anchor is said to contain an object. In this paper, IoU is also used as a hyperparameter as can be seen in Table 2 rows 7 and 14.

In spite of this, IoU information gives interesting insight as a performance or evaluation metric [45,48] because it provides an idea of how much percentage of the boundary box has correctly localized the object. In other words, IoU aids in categorizing a detection as correct or incorrect. Some studies have also used IoU as a performance parameter [49,50]. Furthermore, as a hyperparameter, IoU is used as a threshold criterion, therefore this is only tested on training data. By applying IoU as a performance metric, the localization accuracy of the bounding box on the container image in the test data can be evaluated.

Table 4 and Fig. 16 show the comparison of container detection accuracy. The average accuracy value shows that the optimized Faster RCNN provides better IoU, confidence score, precision, and accuracy values than the default parameter detection results. The results confirm the capability of the proposed method to optimize Faster RCNN for container detection.

5. Conclusion

The paper proposed a method for mixed-integer optimization using PSO with a weighted cost function on Faster RCNN to detect moving containers for port automation. PSO Gbest and Pbest space evolution were developed, which could avoid being trapped in the local minima for integer optimization. The enhanced integer optimization of PSO was tested to nine test functions and provided better optimization results. After that, the combination of continuous and integer parameters optimization led to a mixed-integer optimization method. This method was then applied to the Faster RCNN mixed-integer optimization problems to gain optimal container detection for port automation. An improved cost function was proposed for the optimization based on weighted minimum loss, average loss, and its gradient value. The optimization results using the proposed mixed-integer PSO with weighted cost function demonstrated better accuracy in terms of minimum and average loss, IoU, confidence score, precision, and accuracy than the default parameters. Future research will be devoted to apply this proposed container detector scheme to build a gantry crane automation system at the seaport.

Author contribution statement

Steven Bandong; Endra Joelianto: Conceived and designed the experiments; Performed the experiments; Analyzed and interpreted the data; Contributed reagents, materials, analysis tools or data; Wrote the paper.

Yul Yunazwin Nazaruddin: Conceived and designed the experiments; Analyzed and interpreted the data; Contributed reagents, materials, analysis tools or data; Wrote the paper.

Funding statement

Mr. Steven Bandong was supported by Indonesia Endowment Fund for Education (LPDP – Lembaga Pengelola Dana Pendidikan) (2021–2023), University Center of Excellence Artificial Intelligence on Vision, NLP & Big Data Analytics (2021).

Endra Joelianto was supported by ITB Research Program (2020 & 2022).

Yul Yunazwin Nazaruddin was supported by Doctoral Dissertation Research Scholarship (2022).

Data availability statement

Data will be made available on request.

Declaration of interest's statement

The authors declare no conflict of interest.

References

[1] R.J. Sanchez, J. Hoffmann, A. Micco, G.V. Pizzolitto, M. Sgut, G. Wilmsmeier, Port efficiency and international trade: port efficiency as a determinant of maritime transport costs, Marit. Econ. Logist. 5 (2) (2003) 199–218.
[2] M.A. Budiyanto, H. Fernanda, Risk assessment of work accident in container terminals using the fault tree analysis method, J. Mar. Sci. Eng. 8 (6) (2020) 466.
[3] W. He, S.S. Ge, Cooperative control of a nonuniform gantry crane with constrained tension, Automatica 66 (2016) 146–154, https://doi.org/10.1016/j.automatica.2015.12.026.
[4] S.-M. Kim, S.-K. Sul, Control of rubber tyred gantry crane with energy storage based on supercapacitor bank, IEEE Trans. Power Electron. 21 (5) (2006) 1420–1427, https://doi.org/10.1109/TPEL.2006.880260.
[5] M.I. Solihin, Wahyudi, A. Legowo, Fuzzy-tuned PID anti-swing control of automatic gantry crane, J. Vib. Control 16 (1) (2010) 127–145.
[6] S. Abbate, M. Avvenuti, P. Corsini, B. Panicucci, M. Passacantando, A. Vecchio, An integer linear programming approach for radio-based localization of shipping containers in the presence of incomplete. Proximity information, IEEE Trans. Intell. Transport. Syst. 13 (3) (2012) 1404–1419, https://doi.org/10.1109/tits.2012.2188518.
[7] S. Kavuri, D. Moltchanov, A. Ometov, S. Andreev, Y. Koucheryavy, Performance analysis of onshore NB-IoT for container tracking during near-the-shore vessel navigation, IEEE Internet Things J. (2020), https://doi.org/10.1109/jiot.2020.2964245, 1–1.

[8] Y. Hirofumi, K. Satoshi, H. Hiromitsu, M. Noriaki, K. Masato, Development of Hoisting Load Position Sensor for Container Handling Cranes, Mitsubishi Heavy Industries, Ltd. Technical Review, 2001. Vol. 38, No.2.
[9] B. Rahmat, Y.V. Via, A. Wasian, I.Y. Purbasari, N.K. Sari, W. Wurjani, S. Bandong, E. Joelianto, P.I. Siregar, Video-based container tracking system using deep learning, in: Cyber Physical, Computer and Automation System: A Study of New Technologies, 2021, pp. 81–90.
[10] A.G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, H. Adam, Mobilenets: Efficient Convolutional Neural Networks for Mobile Vision Applications, 2017 arXiv preprint arXiv:1704.04861.
[11] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778.
[12] M. Lokanath, K.S. Kumar, E.S. Keerthi, Accurate object classification and detection by Faster RCNN, IOP Conf. Ser. Mater. Sci. Eng. 263 (5) (2017), 052028.
[13] Y. Yu, K. Zhang, L. Yang, D. Zhang, Fruit detection for strawberry harvesting robot in non-structural environment based on Mask-RCNN, Comput. Electron. Agric. 163 (2019), 104846.
[14] Byeon, H. Yeong, K. Keun-Chang, A performance comparison of pedestrian detection using faster RCNN and ACF, in: 6th IIAI International Congress on Advanced Applied Informatics (IIAI-AAI), IEEE, 2017.
[15] Huaizu Jiang, Erik Learned-Miller, Face detection with the faster RCNN, in: 12th IEEE International Conference on Automatic Face and Gesture Recognition (FG 2017), IEEE, 2017, pp. 650–657.
[16] C. Zhang, X. Xu, D. Tu, Face Detection Using Improved Faster RCNN, 2018 arXiv preprint arXiv:1802.02142.
[17] B. Hu, J. Wang, Detection of PCB surface defects with improved faster-RCNN and feature pyramid network, IEEE Access 8 (2020) 108335–108345.
[18] T. Haas, C. Schubert, M. Eickhoff, H. Pfeifer, BubCNN: bubble detection using Faster RCNN and shape regression network, Chem. Eng. Sci. 216 (2020), 115467.
[19] S. Bandong, Y.Y. Nazaruddin, E. Joelianto, Container detection system using CNN based object detectors, in: 2021 International Conference on Instrumentation, Control, and Automation (ICA), IEEE, 2021, pp. 106–111.
[20] S. Bandong, M.R. Miransyahputra, Y. Setiaji, Y.Y. Nazaruddin, P.I. Siregar, E. Joelianto, Optimization of gantry crane PID controller based on PSO, SFS, and FPA, in: 2021 60th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE), IEEE, 2021, pp. 338–343.
[21] J. Bao, S. Wei, J. Lv, W. Zhang, Optimized Faster RCNN in real-time facial expression classification, Vol. 790, No. 1, in: IOP Conference Series: Materials Science and Engineering, IOP Publishing, 2020, March, 012148.
[22] M. Lu, Y. Hu, X. Lu, Driver action recognition using deformable and dilated faster RCNN with optimized region proposals, Appl. Intell. 50 (4) (2020) 1100–1111.
[23] T. Bai, Y. Pang, J. Wang, K. Han, J. Luo, H. Wang, H. Zhang, An Optimized faster RCNN method based on drnet and roi align for building detection in remote sensing images, Rem. Sens. 12 (5) (2020) 762.
[24] S. Sennan, S. Ramasubbareddy, S. Balasubramaniyam, A. Nayyar, M. Abouhawwash, N.A. Hikal, T2FL-PSO: type-2 fuzzy logic-based particle swarm optimization algorithm used to maximize the lifetime of internet of things, IEEE Access 9 (2021) 63966–63979.
[25] B. Song, Z. Wang, L. Zou, An improved PSO algorithm for smooth path planning of mobile robots using continuous high-degree Bezier curve, Appl. Soft Comput. 100 (2021), 106960.
[26] D.K. Choubey, S. Tripathi, P. Kumar, V. Shukla, V.K. Dhandhania, Classification of diabetes by kernel based SVM with PSO, Recent Adv. Comput. Sci. Commun. (Formerly: Recent Pat. Comput. Sci.) 14 (4) (2021) 1242–1255.
[27] Y.Y. Nazaruddin, A.D. Andrini, B. Anditio, PSO based PID controller for quadrotor with virtual sensor, IFAC-PapersOnLine 51 (4) (2018) 358–363.
[28] Y.Y. Nazaruddin, B. Anditio, A.D. Andrini, Integration of PSO-based virtual sensor and PID to control benfield concentration of a stripper unit in a fertilizer plant, in: 2019 12th Asian Control Conference (ASCC), IEEE, 2019, pp. 364–369.
[29] J. Kennedy, R.C. Eberhart, Y. Shi, Swarm Intelligence, Morgan Kaufmann Publishers Inc., San Francisco, CA, 2001.
[30] M. Gomez-Gonzalez, A. López, F. Jurado, Optimization of distributed generation systems using a new discrete PSO and OPF, Elec. Power Syst. Res. 84 (1) (2012) 174–180.
[31] C.J. Liao, C.T. Tseng, P. Luarn, A discrete version of particle swarm optimization for flowshop scheduling problems, Comput. Oper. Res. 34 (10) (2007) 3099–3111.
[32] X.H. Zhi, X.L. Xing, Q.X. Wang, L.H. Zhang, X.W. Yang, C.G. Zhou, Y.C. Liang, A discrete PSO method for generalized TSP problem, vol. 4, in: Proceedings of 2004 International Conference on Machine Learning and Cybernetics (IEEE Cat. No. 04EX826), 2004, August, pp. 2378–2383.
[33] Y. Sun, Y. Gao, An efficient modified particle swarm optimization algorithm for solving mixed-integer nonlinear programming problems, Int. J. Comput. Intell. Syst. 12 (2) (2019) 530–543.
[34] Y. Hou, G. Hao, Y. Zhang, F. Gu, W. Xu, A multi-objective discrete particle swarm optimization method for particle routing in distributed particle filters, Knowl. Base Syst. 240 (2022), 108068.
[35] A.S. Majid, E. Joelianto, Optimal sensor deployment in non-convex region using discrete particle swarm optimization algorithm, in: 2012 IEEE Conference on Control, Systems and Industrial Informatics, IEEE, 2012, pp. 109–113.
[36] Z. Miao, P. Yong, Y. Mei, Y. Quanjun, X. Xu, A discrete PSO-based static load balancing algorithm for distributed simulations in a cloud environment, Future Generat. Comput. Syst. 115 (2021) 497–516.
[37] M. Jamil, X.S. Yang, A literature survey of benchmark functions for global optimisation problems, Int. J. Math. Model. Numer. Optim. 4 (2) (2013) 150–194.
[38] M. Molga, C. Smutnicki, Test functions for optimization needs, Test functions for optimization needs 101 (2005) 48.
[39] H. Pohlheim, Examples of objective functions, Retrieved 4 (10) (2007) 2012.
[40] S. Hernández, G. Duran, J.M. Amigó, General Algorithmic Search, 2017 arXiv preprint arXiv:1705.08691.
[41] Agmachine, Automatic container landing systems (ACLAS) [video]. YouTube. https://www.youtube.com/watch?v=dKo2IF6lfaA, 2009.
[42] Automos, Reach stacker unload container [Video], https://www.veoh.com/watch/v14221373552qBkWD6, 2022.
[43] Automos, Reach stacker load container to truck [Video], https://www.veoh.com/watch/v1422137372PAm7kEW, 2022.
[44] Thunfisch 1967, Container laden und laschen auf einem Frachter [Video]. YouTube. https://www.youtube.com/watch?v=F55Zcvo0F8k, 2009.
[45] R. Padilla, S.L. Netto, E.A.B. da Silva, A survey on performance metrics for object-detection algorithms, in: 2020 International Conference on Systems, Signals and Image Processing (IWSSIP), IEEE, 2020.
[46] J. Hollister, R. Vega, M.H.B. Azhar, Automatic identification of non-biting midges (Chironomidae) using object detection and deep learning techniques, in: ICPRAM, 2022, pp. 256–263.
[47] K. Kim, H.S. Lee, August. Probabilistic anchor assignment with iou prediction for object detection, in: European Conference on Computer Vision, Springer, Cham, 2020, pp. 355–371.
[48] H. Rezatofighi, N. Tsoi, J. Gwak, A. Sadeghian, I. Reid, S. Savarese, Generalized intersection over union: a metric and a loss for bounding box regression, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, IEEE, 2019, pp. 658–666.
[49] R. Kanza, Y. Zhao, Z. Huang, C. Huang, Z. Li, Enhancement: SiamFC tracker algorithm performance based on convolutional hyperparameters optimization and low pass filter, Mathematics 10 (9) (2022) 1527.
[50] O. Rukundo, October. Effect of the regularization hyperparameter on deep-learning-based segmentation in LGE-MRI, vol. 11897, in: Optoelectronic Imaging and Multimedia Technology VIII, SPIE, 2021, pp. 227–231.