Robust Real-time Detection of Abandoned Objects using a Dual Background Model

Hyeseung Park[1], Seungchul Park[1*] and Youngbok Joo[1]
[1] School of Computer Science and Engineering, Korea University of Technology and Education
Cheonan, Chungnam 31253 – Republic of Korea
[e-mail: hs200park@koreatech.ac.kr, scpark@koreatech.ac.kr, ybjoo@koreatech.ac.kr]
*Corresponding author: Seungchul Park

Received Septebmer 9, 2019; revised November 28, 2019; accepted December 25, 2019; published February 29, 2020

Abstract

Detection of abandoned objects for smart video surveillance should be robust and accurate in various situations with low computational costs. This paper presents a new algorithm for abandoned object detection based on the dual background model. Through the template registration of a candidate stationary object and presence authentication methods presented in this paper, we can handle some complex cases such as occlusions, illumination changes, long-term abandonment, and owner's re-attendance as well as general detection of abandoned objects. The proposed algorithm also analyzes video frames at specific intervals rather than consecutive video frames to reduce the computational overhead. For performance evaluation, we experimented with the algorithm using the well-known PETS2006, ABODA datasets, and our video dataset in a live streaming environment, which shows that the proposed algorithm works well in various situations

Keywords: Abandoned object detection, dual background model, illumination change, PETS2006, ABODA

A preliminary version of this paper was presented at APIC-IST 2019, and was selected as an outstanding paper.

http://doi.org/10.3837/tiis.2020.02.017 **ISSN : 1976-7277**

1. Introduction

Most of existing video surveillance systems are monitored by humans and store the videos on the server. They have problems such as high labor demands, increased traffic and storage capacity, and surveillance failures caused by human mistakes or absence, etc. Therefore, the demand for smart video surveillance services that can automatically detect specific events is increaing. In particular, There is a growing interest in automatic abandonment detection such as suspicious abandoned object detection, garbage dumping detection, and illegal parking detection, etc. Although there have been many studies on the detection of abandoned objects, there are still several challenging issues such as occlusions, illumination changes, owner's re-attendance, and long-term abandonment[1, 2]. In recent years, implementing video analysis functions on edge devices, rather than on high-performance servers, has become an important challenge[3, 4]. It is to solve the network traffic load, latency overhead, and the high construction cost of the video analysis center, which means that the costs for the detection must be reduced sufficiently so that analysis functions can be implemented in edge devices.

Most of the existing studies for abandoned object detection are based on the background model using various background subtraction techniques. They extract a candidate stationary object from the model in their own way and then track the state of the object for a predetermined period analyaing consecutive foreground information generated from the model. If the tracking is successful, they determine the object as an abandoned object[2]. However, due to the nature of the background subtraction technique, there is a problem that the foreground is gradually absorbed into the background. It is also vulnerable to occlusions and illumination changes. So it is difficult and computation-intensive for a surveillance system to accurately track a stationary object for a long time in various environments.

In the proposed algorithm, a stable, unattended, and non-human object is extracted from a dual background model and is selected as a candidate stationary object. At this time, a template of the object, which consists of its position, height, width, and contour data, is created and registered. In addition, an owner of the object is determined by using a back-tracing technique and the color histogram of the owner's image is also registered as a part of the template. It is used to check the re-attendance of the owner while tracking the object. When the timer for abandonment determination expires, the registered template is used for the presence authentication of the candidate stationary object to verify that the object still exists in the current video frame. If the presence authentication is successful, the candidate stationary object is finally determined to be an abandoned object.

The template registration and presence authentication methods make it possible to track the state of a candidate stationary object over time even if its foreground is getting absorbed into the background in the middle of tracking. Once the template of a candidate stationary object is registered, it is not necessary to continuously check the state of the object until the end of timer because the algorithm determines the next or final state of the object in the presence authentication, which can reduce computational costs. That is the main reason why the proposed algorithm can be robust to complex situations such as occlusions, illumination changes, and long-term abandonment. The algorithm also analyzes a video at specific frame intervals instead of every frame. The frame interval to analyze is specified in the TFI(Tracking Frame Interval). Applying adequate TFI can also reduce video analysis computation costs for real-time detection. This paper can detect abandoned objects by analyzing only one frame per 20 frames at 30 fps(frame per second) video, that is, with TFI value 20 in experiments. We

experimented with the algorithm using our own video dataset as well as some popular datasets like PETS2006, ABODA in a live video streaming environment using a typical desktop, which show that the proposed algorithm can accurately detect abandoned objects not only in normal cases but also in complex situations such as occlusions, illumination changes, owner's re-attendance, removal, and so on.

2. Related Work

In this paper, the proposed algorithm detects abandoned objects using the PETS2006 abandoned object criteria[5, 6]. The criteria for PETS2006 abandoned objects are as follows.

1) Temporal rule: An object is considered unattended when it is left by its owner. Then, it should not be re-attended within time T=30 seconds to become an abandoned object.

2) Spatial rule: Only if the distance between the owner and the object should be greater than a predefined distance D = 3 m, the object is considered an unattended object.

Most of the algorithms for abandoned object detection use the background subtraction techniques that generate a foreground image by subtracting a background image from an input image and extract a stationary object from the foreground. It is very complicated to find a stationary object from a foreground image and determine that the object is abandoned. There have been many studies for the detection of abandoned objects using a single background subtraction model. One of the disadvantages is that it is difficult to select and track a stationary object among all the foregrounds generated from a single background model[1, 2].

A dual background subtraction model consists of two single background models with different background absorption rates. It can easily obtain a stationary object from the difference of two foreground images generated by each sub-model[7]. The stationary object remains as a part of the foreground in the long-term model with a low absorption rate, which updates the model slowly. However, it is absorbed quickly into background in the short-term model that has a relatively high absorption rate and updates faster. Then naturally, only the stationary objects remain in the difference of the short-term and long-term foreground images. However, determining a candidate stationary object from only one difference foreground image has a problem that the detection accuracy can be poor due to the incomplete characteristics of the background subtraction model. Several follow-up studies have been conducted to develop the performance of detecting an abandoned object by analyzing the pixel state of the continuous long-term and short-term foreground images for a sufficient time[6, 8]. They track the pixel state of the foreground images until the timer expires, and determine that the group of pixels maintained until the end of the timer is an abandoned object. One of the problems with these techniques is that tracking is getting difficult when another object occludes a candidate stationary object. In addition, both the short-term and long-term background subtraction models gradually absorb the stationary object revealed as a foreground into the background, which could lead to poor tracking accuracy over time. Once the absorption is complete, the tracking can be impossible. Besides, there is a computational disadvantage in that the state of all pixels of every frame must be tracked continuously.

In the triple background subtraction model, short-term, medium-term, and long-term background subtraction models are used[9]. Since the stationary objects are never absorbed into the background and always remain as the foreground in the long-term model, it is very

effective for the detection of the long-term abandoned objects that have been abandoned for quite a long time. It is also helpful to solve the occlusion problem. However, the first problem with this technique is the high computational cost for keeping track of the state of every pixel in successive foreground images produced by three background models until the end of the timer. The second problem is that the long-term background model does not absorb the stationary foreground objects into the background. Therefore it cannot adequately accommodate background changes caused by illumination changes. If illumination changes happen, foreground objects remain un-absorbed in the long-term foreground and a large number of stationary objects are wrongly generated. Recently, deep learning techniques have been attempted to some researches[10]. However, these techniques have problems that the range of objects that can be identified by the deep learning model is limited and the computational costs are high. This paper presents a new algorithm for abandoned object detection based on a dual background model. The algorithm can handle some challenging issues such as occlusions, illumination changes, long-term abandonment and owner's re-attendance as well as general detection of abandoned objects with low computational costs.

3. Abandoned Object Detection based on Presence Authentication

3.1 Framework of the Proposed Algorithm

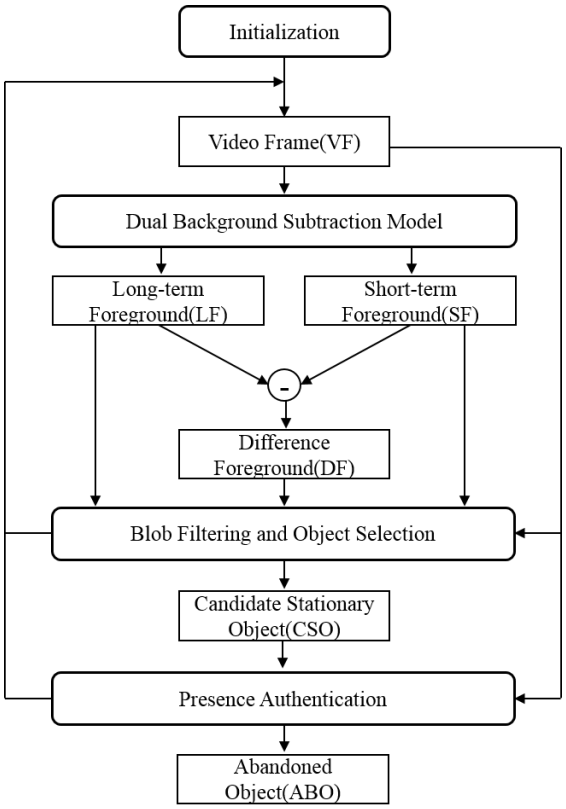
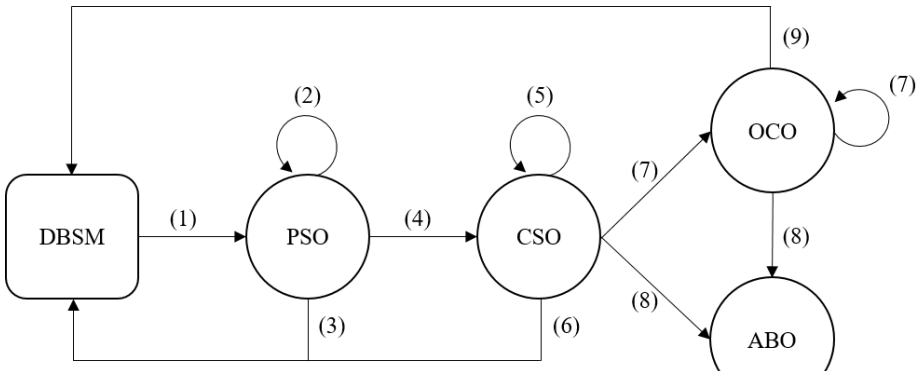


Fig. 1. Framework of the proposed algorithm

Fig. 1 shows the framework of the proposed algorithm. The most fundamental task in detecting an abandoned object placed by its owner and kept stationary for a certain period is to extract candidate stationary objects. In this paper, we use a dual background model that has proved the effectiveness of extracting stationary objects in terms of implementation complexity and computational costs in existing studies. We use the KNN[11] background subtraction technique to build a single background model in this study. The dual background model consists of two single background model with different learning rates and produces short-term foreground(SF) images and long-term foreground(LF) images. Then, the algorithm generates a difference foreground(DF) image of a SF and a LF. In the initialization phase, we set the TFI for low computational cost and the background learning rates of the sub-models of the dual background model.

Existing studies consistently analyze the foreground information until the determination of abandoned objects. However, the proposed algorithm analyzes the foreground blobs of the DF for only a short period of time when verifying the stability of a stationary object. The algorithm selects a stable, un-attended, and non-human object as a candidate stationary through blob filtering and object selection process. At this time, the template of the object is registered and the timer for abandonment determination is started. When the timer expires, the presence authentication process is performed to check the existence of the registered object in the current video frame, which determines the next or final state of the object. If the candidate stationary object successfully passes the presence authentication, it is finally determined as an abandoned object.

3.2 State Transition of a Stationary Object



1) Meaningful Blob Extraction from Difference Foreground
2) Not Stable || Attended
3) Human || Moved
4) Stable && Unattended && Not Human && Presence-authentication Succeeded
5) Unattended && Abandonment Determination Timer Not Out
6) Re-attended
7) Abandonment Determination Timer Out && Presence-authentication Failed
8) Abandonment Determination Timer Out && Presence-authentication Succeeded
9) Occlusion Timer Out || Removed

DBSM : Dual Background Subtraction Model, PSO : Pre-Stationary Object,
CSO : Candidate Stationary Object, ABO : ABandoned Object, OCO : OCcluded Object

Fig. 2. State transition diagram of a stationary foreground object

Fig. 2 shows the state transition of a stationary object in the proposed algorithm. A foreground blob is obtained in the DF through the blob filtering process. First, it enters the Pre-Stationary

Object(*PSO*) state, instead of directly changing to the Candidate Stationary Object(*CSO*) state. Because of the incompleteness of the background subtraction model, some blobs can be generated by noise or the like. Since they might exist temporarily in the DF, we need to verify the stability of each of them. In this paper, a *PSO* is stable if it has existed for a certain period of time at the same position in the DF. If a stable *PSO* is also not a stationary human, the algorithm extracts the contour data of the object from the current video frame and finds its owner using a back-tracing technique. When a stable, non-human *PSO* is getting unattended, the presence authentication for the *PSO* is performed to check the similarity of the initial *PSO* image and the current image of the *PSO* area. If they are considered identical, it is changed to a *CSO* and the template including the owner histogram is registered. From this point, we only check whether the object is re-attended until the timer expires, that is 30 seconds, without continuously analyzing the current state of the object. At the end of the timer, the presence authentication for the *CSO* is performed to verify the existence of the object in the current video frame. According to the result, the state can be changed to the *ABO*(ABandoned Object), *OCO*(OCcluded Object), or removed.

3.3 Blob Filtering, Illumination Change Handling and Extracting Candidate Stationary Objects

Blob filtering is performed first to identify the meaningful blobs in the LF, SF, and DF generated from the dual background model. The blobs smaller than the minimum threshold (this is initially set to a sufficiently small value, but it can be adjusted according to the measurable objects of input image, for example, 1/10 of the size of a detectd pedestrian) in the DF are considered meaningless like noise and discarded. From this point, we only focus on the meaningful blobs. They are used for handling illumination changes or extracting stationary objects in this paper.

Sudden illumination changes caused by such as turning on the lights or passing clouds in the sky, etc. can greatly affect the entire image, which creates a large number of blobs with various sizes in both the LF and SF. The short-term model with a high learning rate can quickly adapt to changes. In other words, the foreground blob of the SF is also quickly absorbed into the background. However, the foreground blob of the LF lasts for quite a long time in the following LFs, which produces many stationary objects in the DF. Since most of the stationary objects are wrongly generated by the illumination changes, the false positive rate can be increased. Therefore, it is important to handle them. First, we need to know whether an illumination change occurs. The proposed algorithm recognizes the occurrence of an illumination change when a huge blob larger than the maximum threshold(for example, more than 1/4 of the full image size in our experiments.) is detected in the LF and SF. When illumination changes occur, we must adapt to those situations quickly. In other words, the affected foreground blobs, especially in the LF, should be absorbed quickly into the background. To solve this, we adjust the background learning rates of both the long-term and short-term models to the same maximum value by compulsion, so that the affected foreground blobs can be absorbed into the background quickly and simultaneously. At this time, the TFI is adjusted to a predetermined small value for a while so as to adapt more quickly, and returned to the original value when adaptation is over. Finally, it is also important to know the end of the adaptation. If both images of the long-term and short-term models are completely identical to each other, that is, if the number of foreground blobs of the DF is zero, we consider that the illumination change adaptation is over. After the adaptation, the absorption rates of each model are returned to their original values to continue the object state tracking. The proposed method can easily handle illumination changes. But the problem is that it initializes all existing

foreground information, which makes them disappear in the LF, SF, and DF. Therefore, studies heavily dependent on foreground information for tracking an object are not suitable to apply this method. However, we also solve the problem by using the template registration and presence authentication methods, which is explained in the next chapter.

After the blob filtering, the remaining blobs in the DF are determined to be *PSO*s. To verify the stability of a *PSO*, the proposed algorithm checks whether the PSO appears more than the specified number of times(5 in our experiments) in the subsequent DFs. To determine whether the current *PSO* is the same as the previous *PSO* at the same location, we compare the size values of each *PSO*. Once the stability of the *PSO* is confirmed, we check if the *PSO* is not a stationary human. We applied the HOG[12] for human identification. Once the *PSO* is determined to be stable and not a human, the foreground blob comparison is no longer performed. From this point, it is only necessary to check whether the stable *PSO* is un-attended, that is, the distance between the *PSO* and the owner is more than a certain threshold. When it is getting un-attended, the presence authentication for the *PSO* is performed to verify the similarity of the initial *PSO* image and the current image of the *PSO* area. If they are considered identical, the state is changed to the *CSO*. In summary, a stable, non-human, and un-attended *PSO* becomes a *CSO*.

3.4 Template Registration for Candidate Stationary Object

When our algorithm chooses a *CSO*, the template of the object is registered. The template of a CSO consists of the following information.

$$CSO\ \ template = \{Position, Height, Width, maxContour, New\_maxContour, BG\_maxContour, OwnerHistogram\}$$

Most of the attributes of the *CSO* template inherit the attributes of the previous *PSO*, but a new attribute is registered when a *CSO* is selected. *Position*, *height*, and *width* attributes represent the *PSO* area in the video frame, which constructs a region of interest for tracking the object. When a *PSO* is determined to be stable, the proposed algorithm extracts several contours of the *PSO* area in the current video frame. *maxContour* is the largest contour data (e.g. shape, area, length, vertices). Similarly, *BG_maxContour* represents the largest one among the contours of a background image. It can be extracted from the previous long-term background(LB) image corresponding to when the *PSO* is created first in the DF, because the foreground of the object is not yet absorbed into the background at that time. The template inherits these attributes. However, *New_maxContour* attribute contains the largest contour data of the *CSO* area and is extracted when a *PSO* becomes a *CSO*. Normally, *maxContour* and *New_maxContour* have the same value. But they can have different values in some particular cases such as when the *PSO* is partially removed or occluded, which is determined by the *PSO* presence authentication. The detail of the *PSO* presence authentication is explained in the next chapter. The reason for using the largest contour, not others such as the whole contours and color histogram, is that it is more robust to illumination changes. When an illumination change occurs, there could be small or massive changes in the detail of the object. For example, the direction, shadows, and the color of the object, etc. can be affected. Color histogram comparison, a simple way to calculate the similarity of two images, works well under normal circumstances but is very vulnerable to illumination changes. Therefore, we need something less vulnerable to the changes to compare two object images of different times. For this, we use the texture information of the object. Contour, an element of edges, has the advantage of measuring and utilizing specific areas because it is the closed curve. The contours of the object

area contain the entire object and parts of the background. As the light changes, the direction and size of the shadow can be changed, which affects the internal details of the object. Therefore, the contours inside the object are relatively more vulnerable to illumination changes. However, the largest contour, which is very likely to be the external shape of the object, shows robustness to illumination changes. That is why we use the largest contour for the template registration and presence authentication. We use the shape, area, length, and the number of vertices of a contour for comparison.

In most cases, people may leave their objects temporarily, but soon return to the objects. In terms of the state transition diagram in **Fig. 2**, the state of an unattended object becomes attended. We should not assume that an object is attended simply because a random person is near the object. We only need to respond to the access of an owner. Therefore, the owner of an abandoned object must be found. We also need to check if the owner is re-attending to the object before the timer expires. The proposed algorithm uses a back-tracing technique to find an owner by checking for moving objects near the *PSO* in the previous frame corresponding to when the *PSO* if first generated in the DF. If no human is found in that frame, the algorithm searches for the owner in the previous frames. If finding the owner fails for a set number of attempts, the object is not considered abandoned by a human. Therefore, the algorithm should remove it from the *CSO* list. If the owner is found, the histogram of the owner image is stored in the template. The owner histogram is used to verify whether the human attending to the object is the owner or not. The reason for verifying the owner by the simple histogram comparison, not by the largest contour comparison, is that the owner's shape keeps changing, unlike the stationary object.

3.5 Presence Authentication

Presence authentication can be performed for the *PSO* and *CSO*, which verifies the existence of a target object in the current video frame. Performing the presence authentication on every frame is computationally intensive. Therefore, the proposed algorithm performs the presence authentication only when specific events such as the template registration, timer expiration, etc. occur. The foreground information is not necessary in the presence authentication because the method only needs the template and the current image of the *PSO* or *CSO* area. The simplified version of the *PSO* presence authentication algorithm is as follows.

```
if (matching_score(PSO.maxContour, Current_maxContour of PSO.area) ≥ Th_Contour),
then    /* PSO still exists */
        { CSO.maxContour = PSO.maxContour;
          CSO.New_maxContour = PSO.maxContour;}
else
if (matching_score(PSO.BG_maxContour, Current_maxContour of PSO.area) ≥ Th_Contour),
then    /* fully removed object */
        return;
else     /* partially removed or occluded */
        { CSO.maxContour = PSO.maxContour;
          CSO.New_maxContour = Current _maxContour of PSO.area }
```

The *PSO* presence authentication runs when a *PSO* is determined to be stable, un-attended, and non-human. It is the process of verifying that the *PSO* has existed since it is created, and

also registering the template before starting the timer for abandonment decision. First, the largest contour of the *PSO* area in the current video frame, hereinafter '*Current_maxContour*' is extracted from the current video frame, and then it is compared with the *PSO.maxContour* extracted when verifying the stability of the *PSO*. If they are identical, that is, if the matching score is higher than the threshold(*Th_Contour*), it logically means the *PSO* still exists in the video. Then the state of *PSO* is changed to the *CSO*. In this case, the *CSO.maxContour* and *CSO.New_maxContour* both inherit the *PSO.maxContour*. If they don't match each other, the *Current_maxContour* is compared with the *PSO.BG_maxContour* extracted from the background image. If they are equal to each other, we consider that the *PSO* is fully removed from the video and stop further tracking immediately. If it doesn't match either the *PSO.maxContour* or *PSO.BG_maxContour*, we assume that the *PSO* is partially removed or occluded at the present time. In this case, we store the *Current_maxContour* in the *New_maxContour* attribute of the *CSO*, which adds an option to compare in the presence authentication for the *CSO* after the timer expires. If the timer expires, the presence authentication for the *CSO* is executed. The *CSO* presence authentication algorithm is as follows.

```
if (matching_score(CSO.maxContour | CSO.New_maxContour, Current_maxContour of
CSO.area) ≥ Th_Contour),
 then    /* abandoned object */
         { alarm ABO_detection;remove the CSO from CSO_list;}
else
if (matching_score(CSO.BG_maxContour, Current _maxContour of CSO.area) ≥ Th_Contour),
then     /* removed object */
         {remove the CSO from CSO_list;}
else      /* occluded object */
         {repeat presence authentication at next turn}
```

First, the current maxContour of the *CSO* area, also hereinafter '*Current_maxContour'* is extracted in the current video frame. It is then compared with both the *maxContour* and *New_maxContour* attributes of the template. If the *Current_maxContour* matches the *CSO.maxContour*, it also logically means that the object has been stationary for more than n seconds(30 seconds according to the PETS2006 temporal rule) since it was unattended. Otherwise, if it does not match the *CSO.maxContour* but matches the *CSO.New_maxContour*, we consider that the object still exists in the video but was partially removed before the *CSO* was confirmed. Although, the *CSO* is determined as an abandoned object in these cases. If the *Current_maxContour* doesn't match either the *CSO.maxContour* or *CSO.New_maxContour* attributes of the template, it is compared with the *BG_maxContour* to check whether the CSO has been removed before the end of the timer. If they are considered identical, we remove the *CSO* from the *CSO* list. If the *Current_maxContour* is not even the same as the *BG_maxContour*, this means that the *CSO* is currently occluded by another object. Therefore, in this case, the presence authentication runs again after a predetermined time passes. If the occlusion disappears during the period, the *CSO* would be finally determined to be an *ABO* or removed object. If it is still occluded, we repeat the presence authentication a pre-determined number of times. If the occluding object remains stationary, it can be detected as a new *CSO*.

4. Experimentation Results

4.1 Experimentation Environment

We have tested the real-time detection capability of the proposed algorithm by using a live streaming video input from a RTSP server as seen in **Fig. 3**. The client system used a common desktop of a 3.3 GHz Xeon E3 CPU. For verification, we used PETS2006[5], ABODA[13] and used our dataset(1920x1080 / 29.97 FPS) for experiments in more complex situations. We experimented with applying TFI 20, that is, one frame per 20 consecutive video frames, to show that it is possible to detect the objects abandoned by pedestrians using as few video frames as possible. When the rapid foreground absorption is necessary, such as the initialization, illumination changes, etc., the event is automatically detected in the blob filtering and then TFI 5 is applied until the absorption is completed. For the KNN model deployed as the background subtraction model in this study, the values of history size for determining the background absorption rate was set to the short-term model 50 and the long-term model 500. When the algorithm detects the illumination change, both values are set to one, respectively, to enable rapid background absorption.

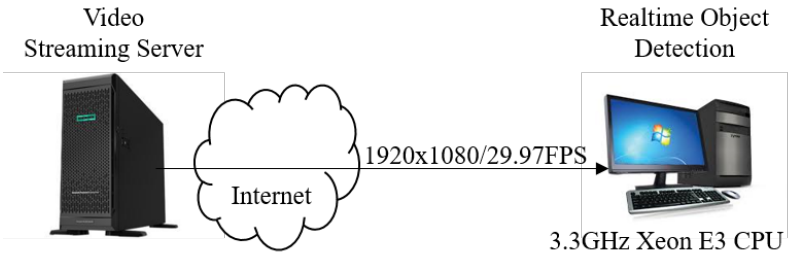


Fig. 3. Experimentation environment

4.2 Experimentation for PETS2006 and ABODA Datasets

We performed experiments using PETS2006 and ABODA datasets to verify the basic abandoned object detection performance of the proposed algorithm. **Table 1** shows the test results for PETS2006.

Table 1. PETS2006 test results

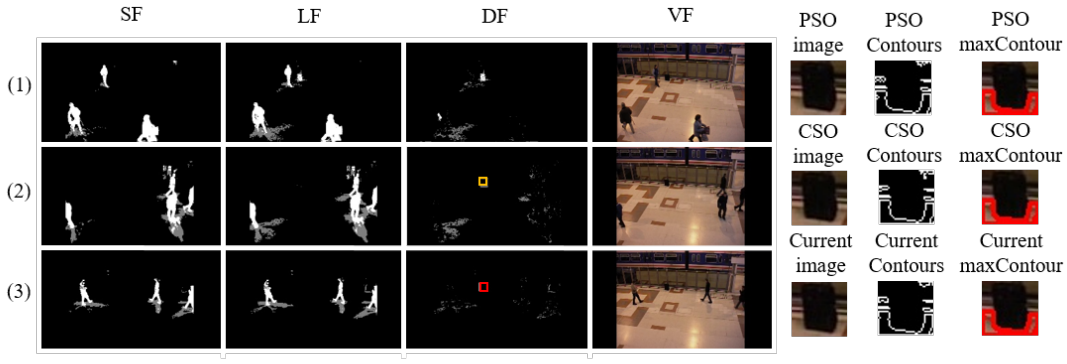
Video	GT	TP	FP	FN	Find Owner
S1	1	1	0	0	1
S2	1	1	0	0	1
S3	1	1	0	0	-
S4	1	1	0	0	1
S5	1	1	0	0	1
S6	1	1	0	0	1
S7	1	1	0	0	1

*GT, TP, FP, FN denote ground truth, true positive, false positive, and false negative, respectively

The proposed algorithm accurately detects abandoned objects for all test scenarios of PETS2006 and also accurately detects the owner of the object for each scenario. S3 shows an owner putting his bag on the ground, standing nearby the bag for a while, then picking up the bag and disappearing. Based on our algorithm, a stable, attended *PSO* is first created and

changed to the un-attended state when the owner disappears. Since the object disappears with the owner, the algorithm knows that the *Current_maxContour* becomes the same as the *PSO.BG_maxContour* through the presence authentication for the *PSO* and the algorithm stops further tracking. So the result of the scenario S3 is that there is no abandoned object, therefore, no owner.

Fig. 4 shows the process of detecting the abandoned object in S7 which has the highest difficulty in the PETS2006 dataset. In this scenario, a *PSO* becomes un-attended after 28 seconds have passed since the *PSO* was created. Note that at the time (2) of **Fig. 4** when the *PSO* is changed to a *CSO*, almost the entire foreground of the target object in the LF as well as in the SF is already absorbed into the background. However, since we already have the necessary information such as the region of interest, the largest contour, etc. for the PSO presence authentication, there is no problem if the foreground disappears. Therefore, the state of the *PSO* can be changed to a *CSO* when the *PSO* is un-attended. In this scenario, since the *PSO* presence authentication succeeds, the *New_maxContour* attribute has the same value as the *maxContour* attribute of the *CSO* template. After a *CSO* is confirmed, the owner does not re-attend to the object though many people are passing by near the *CSO*. Therefore, the algorithm ensures that the person near the *CSO* is not the owner by checking the owner's re-attendance. When the timer expires, the *CSO* is finally determined to be an *ABO* by the *CSO* presence authentication at the time (3) of **Fig. 4**, because the *Current_maxContour* matches the *maxCoutour* of the template.



(1) t : PSO determined, (2) t+28s : CSO determined, (3) t+58s : ABO determined
SF : Short-term Foreground, LF : Long-term Foreground, DF : Difference Foreground, VF : Video Frame

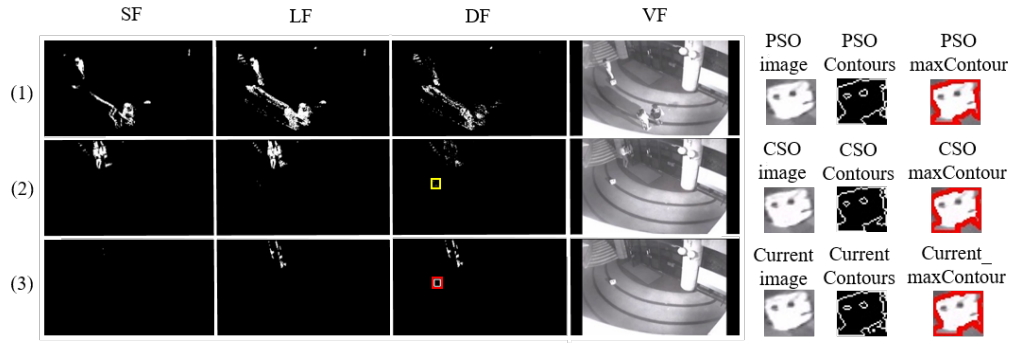
Fig. 4. Abandoned object detection for PETS2006 S7

Table 2 shows the experimental results for the ABODA dataset. The ABODA dataset contains scenarios that do not meet the abandonment determination criteria of PETS2006. So, we excluded V11 which does not comply with the PETS2006 abandonment rules from both temporal and spatial viewpoints. We also experimented by shortening the abandonment determination time condition to 10 seconds instead of 30 seconds due to the short video length problems of ABODA.

Table 2. Comparison of ABODA test results

Video	GT	TP/FP[ours]	TP/FP[ABODA authors']
V1(outdoor)	1	1/0	1/0
V2(outdoor)	1	1/0	1/0
V3(outdoor)	1	1/0	1/0
V4(outdoor)	1	1/0	1/0
V5(night)	1	1/0	1/1
V6(light switching)	2	2/0	2/0
V7(light switching)	1	1/0	1/1
V8(light switching)	1	1/0	1/1
V9(indoor)	1	1/0	1/0
V10(indoor)	1	1/0	1/0
V11(crowded)	1	-	1/3

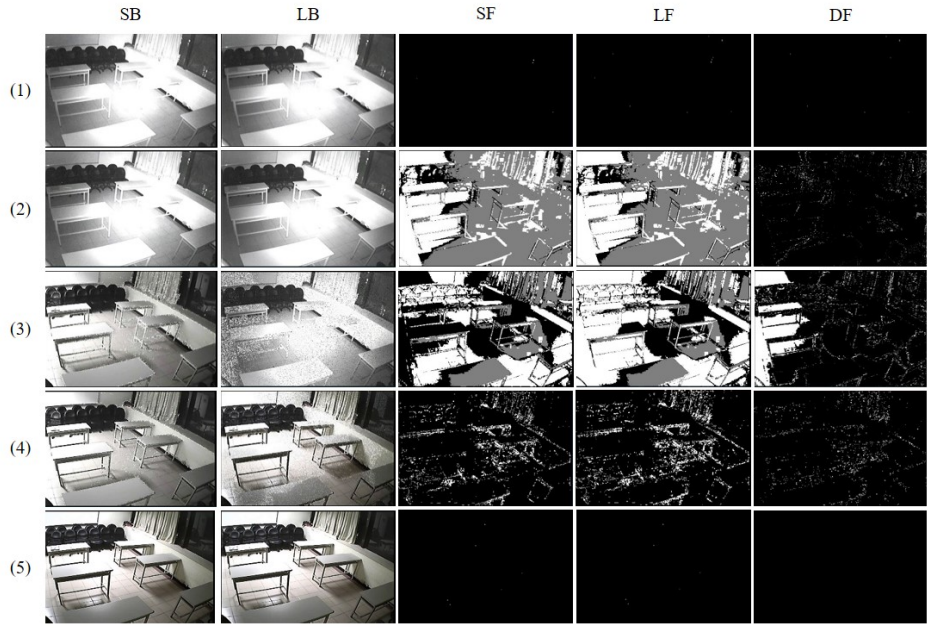
Even the algorithm proposed by the authors of ABODA dataset[6], as shown in **Table 2**, has false-positive errors detecting wrong the objects caused by sudden illumination changes in the scenarios(V5, V7, V8). Since our algorithm can recognize illumination changes rapidly and absorb the foreground objects into the background immediately, we don't suffer from that kind of problem. In V5 of ABODA, the human shadow can be detected because people loiter in a tight space for a quite long time. In this scenario, people are moving little by little, thus their shadows move too. Then, because the difference foreground blobs of the same size are not constantly detected in the same position, the object is filtered by verifying the *PSO* stability without changing to the *CSO* state. So eventually, as we can see in **Fig. 5**, only the luggage abandoned by its owner is determined to be an abandoned object.



(1) t : PSO determined, (2) t+25s : CSO determined, (3) t+35s : ABO determined
SF : Short-term Foreground, LF : Long-term Foreground, DF : Difference Foreground, VF : Video Frame

Fig. 5. Abandoned object detection in night[ABODA V5]

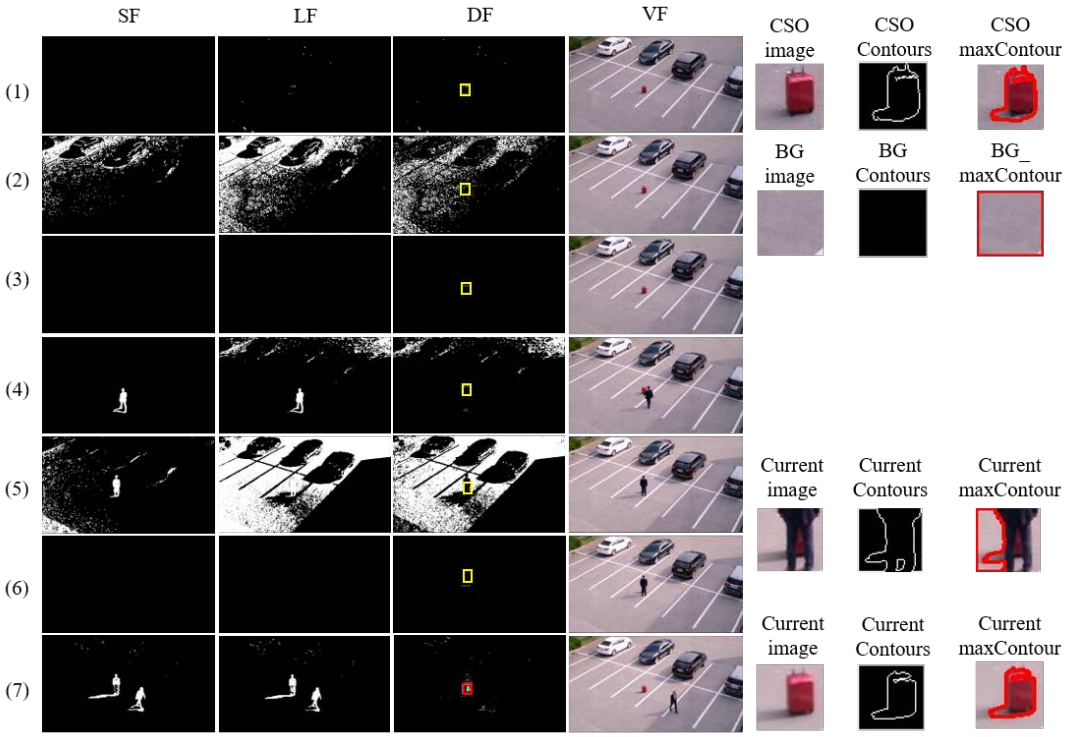
Fig. 6 shows how the proposed algorithm adapts quickly to an illumination change caused by turning on the light in ABODA V7. We can see that when an illumination change is detected, the long-term model adapts to the change almost the same as the short-term model does. As mentioned above, we regard that the LF and SF images are completely identical, if there are no blobs detected in the DF, which means that the adaptation is over. Then the absorption rates of each model are returned to their original value to continue the object tracking. The adaptation took less than 2 seconds in our experiments.



(1) t : Illumination Change, (5) t+2s : After Adaptation
SB : Short-term Background, LB : Long-term Background,
SF : Short-term Foreground, LF : Long-term Foreground, DF : Difference Foreground

Fig. 6. Illumination change handling[ABODA V7]

The ABODA's illumination change scenarios only deal with illumination changes occurring before an object is unattended. They don't cover the cases that the illumination changes occur after an object is unattended. Because illumination changes affect existing foreground information, it is important to be robust to the changes that occur in the middle of tracking. We applied our scenario as shown in **Fig. 7** to verify the robustness of abandoned object detection when the multiple illumination changes occur after abandonment and occlusion happen together. We can see that the *CSO* still exists after the 1st illumination change occurs in (3) of **Fig. 7**. And also, the proposed algorithm can verify that the *CSO* is occluded by another object right after the 2nd illumination change in (6) of **Fig. 7**. The occlusion case will be explained more in the next section. Since the 1st presence authentication failed, the algorithm repeats the presence authentication after the pre-determined time(10 seconds in the experiment) passes. And in (7) of **Fig. 7**, we can see that the *CSO* is determined to be an *ABO* because the *maxContour* of the template is considered equal to the *Current_maxContour* in the presence authentication. This shows that the largest contour of an object can remain robust even when illumination changes occur.



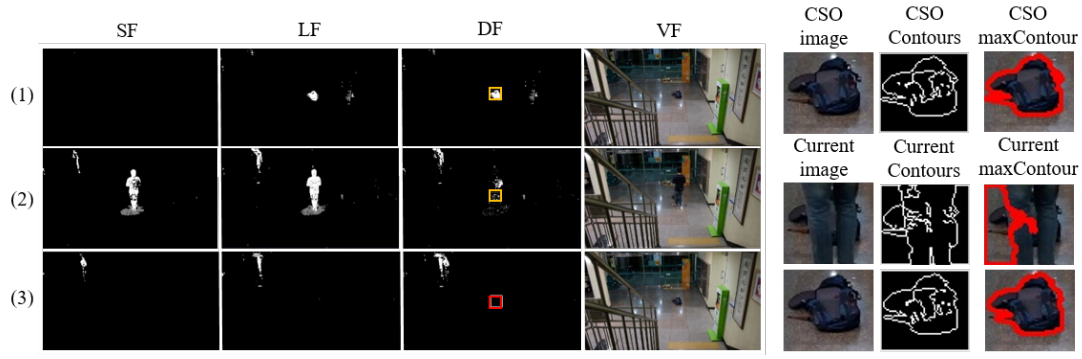
(1) t : CSO Extraction, (2) t+9s : 1st Illumination Change, (3) t+11s : After Adaptation,
(4) t+24s : A Human(Not Owner) is Attending, (5) t+27s : 2nd Illumination Change,
(6) t+30s : 1st Presence Authentication ; Occlusion, (7) t + 40s : 2nd Presence Authentication ; ABO Detection
SF : Short-term Foreground, LF : Long-term Foreground, DF : Difference Foreground, VF : Video Frame

Fig. 7. Detection of an object abandoned before illumination changes

4.3 Discrimination of Occluded, Removed, and Abandoned Objects

Once a *CSO* is selected, the proposed algorithm determine the next or final state of the object after n seconds except when the owner reattends. In other words, we are not interested in occlusions that occur before the presence authentication. We only consider the occlusion that occurs at the end of the timer. If the *Current_maxContour* is identical to either the *maxContour* or *New_maxContour* of the template, a *CSO* is finally determined to be an *ABO* in the presence authentication. Otherwise, if the *Current_maxContour* is not the same as the *BG_maxContour*, its state is changed to an occluded object(*OCO*). If they are identical, we regard that the object has been removed before the timer expires.

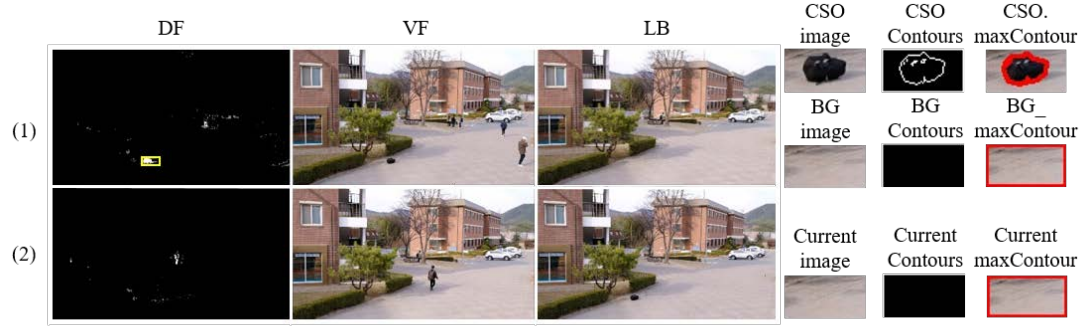
Fig. 8 shows the experiment of detecting occlusion at the time of abandonment determination, using our own video data. The proposed algorithm extracts the *Current_maxContour* for the presence authentication at t + 30s after the *CSO* is confirmed at t. Since the *CSO* is occluded by another object(human) at this time, the *Current_maxContour* is different from both the *maxContour* and the *New_maxContour* of the template. The current maxContour data is also different from the *BG_maxContour* of the template. Therefore, the *CSO* state is changed to an *OCO* state and the presence authentication is repeated at the next determination time. We performed the next *CSO* presence authentication after 30 seconds following the PETS2006 abandonment temporal rule. If the *Current_maxContour* matches the *maxContour* or the *New_maxContour*, the *CSO* is finally determined to be an *ABO*. In this scenario, it matches the *maxContour* of the *CSO* template.



(1) t : CSO determined, (2) t+30s : occluded, (3) t+60s : ABO determined
SF : Short-term Foreground, LF : Long-term Foreground, DF : Difference Foreground, VF : Video Frame

Fig. 8. Abandoned object detection in occlusion situation

Fig. 9 shows the case where an object disappears in the middle of tracking. The *maxContour* and *BG_maxContour* attributes of the template at t. The *Current_maxContour* is different from the *maxContour*. But it is the same as the *BG_maxContour*. So we consider that the *CSO* has already been removed before the presence authentication at time t + x. Therefore, in this case, the *CSO* is determined to be a removed object.



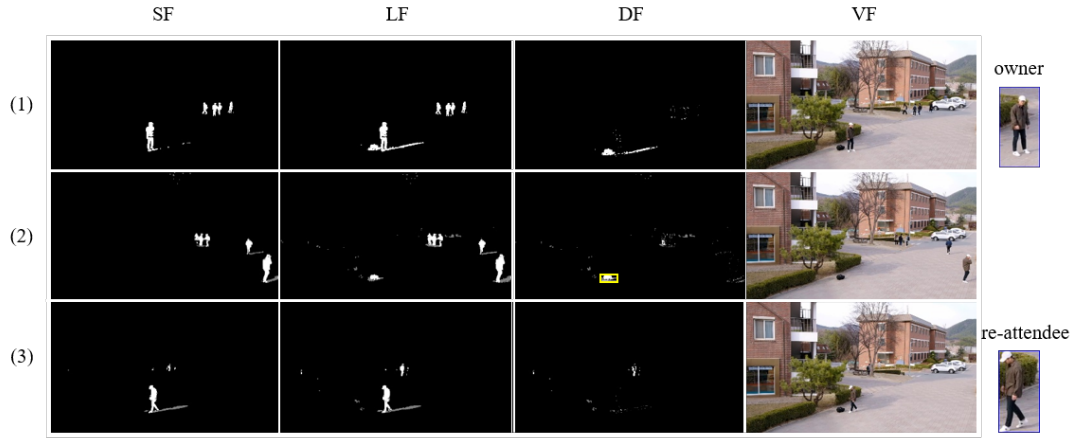
(1) t : CSO determined, (2) t+x : Object removed
DF : Difference Foreground, VF : Video Frame, LB : Long-term Background

Fig. 9. Removed object detection

4.4 Finding Owner and Checking Owner's Re-attendance

The proposed algorithm finds the owner of an abandoned object and includes the histogram of the owner in the template. The algorithm detects an owner's re-attendance if a moving object which size is larger than the minimum threshold is detected near the *CSO* and it is a human and the color histogram of the object is the same as that of the template. If the owner is detected near the *CSO*, we discard the *CSO* from the list. **Fig. 10** shows the experiment of finding an owner and the owner's re-attendance. By using the back-tracing technique, the owner is found at the time t at which the *PSO* is determined. And it is registered as a part of the template at the time t + 11s. Then, when a human appears on the LF image at the time t + 30s, the proposed algorithm checks the owner's re-attendance on the current video frame by comparing the human histogram with the registered owner histogram. If the owner has re-attended, the *CSO* is no longer unattended, and it is removed from the *CSO* list. We have tested owner detection

for PETS2006 and ABODA datasets, and confirmed that the proposed algorithm correctly finds the owner for all scenarios.



(1) t : PSO determined, (2) t+11s : CSO determined, (3) t+30 : Owner's Re-attendance
SF : Short-term Foreground, LF : Long-term Foreground, DF : Difference Foreground, VF : Video Frame

Fig. 10. Owner and re-attendance detection

5. Conclusion

In this paper, we present a new algorithm for detecting an abandoned object based on the dual background model. A background subtraction technique, which is one of the effective ways to extract foreground information from images, is used for the background modeling. However, there is a problem that the foreground is gradually absorbed into the background over time. Another problem is that it is very vulnerable to illumination changes. In this paper, we present an effective illumination change handling technique which detects an illumination change and adapts quickly. But, it also has a problem that the existing foreground information disappears after the adaptation. To solve this problem, we suggest the template registration and presence authentication. In the proposed algorithm, the template of a candidate stationary object extracted from a dual background subtraction model is registered, and it is used to authenticate the presence of the object only if specific events such as the timer expiration, etc. occur. We showed that the algorithm accurately detects abandoned objects not only in normal cases but also in complex situations such as occlusions, illumination changes, owner's re-attendance, and removal. The algorithm is less computation-intensive because it doesn't need to continuously track the target object until the end of timer. The final state is determined by the presence authentication comparing the template and the largest contour of the region of interest in the current video frame. The tracking frame interval(TFI) which is the frame interval to be analyzed, is also helpful to reduce the computational cost of the algorithm. In most of the experiments, our algorithm accurately detect abandoned objects applying TFI 20, one frame per 20 frames.

References

[1] C. Cuevas, R. Martinez, and N. Garcia, "Detection of stationary foreground objects: a survey," *Computer Vision and Image Understanding*, 152, pp. 41-57, 2016. Article (CrossRef Link)

[2] E. Luna, J. C S Miguel, D. Ortego, and J. M Martinez., "Abandoned object detection in video-surveillance: survey and comparison," *Sensors*, 18(12), pp. 1-32, 2018. Article (CrossRef Link)

[3] N. Chen, Y. Chen, E. Blasch, H. Ling, Y. You, ans X. Ye, "Enabling Smart Urban Surveillance at The Edge," in *Proc. of 2017 International Conference on Smart City*, 2017. Article (CrossRef Link)

[4] S. Y. Nikouei, Y. Chen, S. Song, R. Xu, B. Y. Choi, and T. T. Faughnan, "Smart Surveillance as an Edge Network Service: from Harr-Cascade, SVM to a Lightweight CNN," in *Proc. of 4th International Conference on Collaboration and Internet Computing (CIC)*, 2018. Article (CrossRef Link)

[5] PETS 2006 Benchmark Data. Available: http://www.cvg.reading.ac.uk/PETS2006/data.html

[6] K. Lin, S.-C. Chen, C.-S. Chen, D.-T. Lin, and Y.-P. Hung "Left-Luggage Detection from Finite-State-Machine Analysis in Static-Camera Videos," in *Proc. of 22nd International Conference on Pattern Recognition*, 2014. Article (CrossRef Link)

[7] F. Porikli, Y. Ivanov, and T. Haga, "Robust abandoned object detection using dual foregrounds," *EURASIP Journal on Advances in Signal Processing*, pp. 1-11, 2008. Article (CrossRef Link)

[8] K. Lin, S.-C. Chen, C.-S. Chen, and D.-T. Lin, "Abandoned object detection via temporal consistency modeling and back-tracing verification for visual surveillance," *IEEE Trans. On Information Forensics and Security*, Vol. 10, No. 7, pp. 1359-1370, 2015. Article (CrossRef Link)

[9] C. Cuevas, R. Martinez, D. Berjon, and N. Garcia, "Detection of stationary foreground objects using multiple nonparametric background-foreground models on a finite state machine," *IEEE Trans. On Image Processing*, Vol. 26, No. 3, pp. 1127-1142, 2017. Article (CrossRef Link)

[10] S. Smeureanu and R. T. Ionescu "Real-Time Deep Learning Method for Abandoned Luggage Detection in Video," in *Proc. of Conference: 26th European Signal Processing Conference (EUSIPCO)*, 2018. Article (CrossRef Link)

[11] Z. Zivkovic and F. Van Der Heijden, "Efficient adaptive density estimation per image pixel for the task of background subtraction," *Pattern Recognition Letters*, Vol. 27, Issue 7, pp. 773-780, 2006. Article (CrossRef Link)

[12] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proc. of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005.

[13] ABODA Dataset. Available : http://imp.iis.sinica.edu.tw/ABODA/index.html

Hyeseung Park received the B.S. degree in Computer Engineering and Science in 2012 and M.S degree in Electronics and Computer Engineering and Science from Hanyang University in 2014, respectively. He is currently a Ph.D. student in Computer Engineering and Science at Korea University of Technology and Education from 2016. His current research interests include image processing, deep learning and blockchain technology.

Seungchul Park received the B.S. degree in Computer Science and Statistics from Seoul National University in 1985 and M.S. degree in Computer Science from Korea Advanced Institute of Science and Technology, and Ph.D. in Computer Engineering from Seoul National University, Korea, in 1996. Before joining Korea University of Technology and Education as a professor in 2004, he worked for Electronics and Telecommunications Research Institute, IBM Korea, Hyundai Electronics(current SK Hynix), and Hyundai Networks, as a software and network engineer. His current research interests include Blockchain, network security, and multimedia communications.

You ngbok Joo received his B.S. and M.S. in Computer Science in 1991 and 1993, respectively from Yonsei University, Seoul, Korea and received M.S. and Ph.D. in Computer Science and Engineering in 1997`and 2000, respectively from the University of New South Wales, Sydney, Australia. From February 1991 to January 1993, He was an assistant researcher at Samsung Electronics, Ki-Heung, Korea. From March 1997 to June 2000, he was a researcher at Platypus Technology, Sydney, Australia. From July 2000 to October 2001, he was a senior researcher at Q-vis Ltd., Perth, Australia. From December 2001 to June 2002, he was a visiting researcher at the University of Western Australia, Perth, Australia. From July 2002 to May 2006, he was a senior researcher at Lynx Engineering Consultant, Perth, Australia. From June 2006 to 2008, he was a research professor at Kyungpook National University, Daegu, Korea and Yonsei University, Seoul, Korea. From 2009, he has been an associate professor at Korea University of Technology and Education. His current research interest is pattern recognition, face detection and recognition, automatic vision inspection.