International Journal of Grid and Distributed Computing
Vol. 11, No. 8 (2018), pp.1-12
http://dx.doi.org/10.14257/ijgdc.2018.11.8.01

A Study on the Virtual Machine-based Anti-Theft System Running on IoT Devices

Jaehyun Kim and Yangsun Lee*

Dept. of Computer Engineering, Seokyeong University
16-1 Jungneung-Dong, Sungbuk-Ku, Seoul 136-704, KOREA
**yslee@skuniv.ac.kr*

Abstract

The Internet of Things (IoT) is a next-generation innovation technology that integrates embedded sensors and communication functions into devices and connects them to the Internet. These IoT devices are used in various fields such as business, medicine, artificial intelligence, and the like. As the service and the market of the IoT devices are increased, the platform service based on the IoT devices is attracting attention.

In this paper, we design and implement a theft prevention system that runs on the IoT devices. Theft protection system consists of an antitheft application and a virtual machine system to execute it. The application program consists of an image recognition module that receives camera image from inside the system, an object detection module that detects an object in the image, and a theft confirmation module that compares the past object with the current object to identify the stolen article. In addition, the virtual machine system, which is an execution environment, implements the OSMU (One Source Multi Use) by executing an application program developed in various languages without restriction of a platform or an operating system. Such a virtual machine system based antitheft system has the advantage of high portability and independence of operating system and platform because it can be executed when there are devices with the IoT devices.

Keywords: *Internet of Things (IoT), Virtual Machine System, Theft Prevention System, Image Recognition, Object Detection, Object Identification, OSMU (One Source Multi Use)*

1. Introduction

The Internet of Things (IoT) is a next-generation innovation technology that integrates embedded sensors and communication functions into devices and connects them to the Internet. These IoT devices are used in various fields such as business, medicine, artificial intelligence, and the like. As the service and the market of the IoT devices are increased, the platform service based on the IoT devices is attracting attention [1].

In this paper, we design and implement a theft prevention system that runs on the IoT devices. Theft protection system consists of an antitheft application and a virtual machine system to execute it. The application program consists of an image recognition module that receives camera image from inside the system, an object detection module that detects an object in the image, and a theft confirmation module that compares the past object with the current object to identify the stolen article. A virtual machine system consists of a compiler that converts an application into an assembly file that is an intermediate code, an assembler that converts the assembly file into a binary file that can be executed by the virtual machine, and a virtual machine that executes the binary file in an interpreter manner. The virtual machine system has the advantage of implementing One Source Multi Use (OSMU) because

Received (March 13, 2018), Review Result (June 16, 2018), Accepted (June 20, 2018)
* Corresponding Author

ISSN: 2005-4262 IJGDC

the application program developed in various languages can be executed independently without restriction of a platform or an operating system. Therefore, the theft prevention system based on the IoT devices can be executed if there is an device with the IoT devices, and so it has the advantage of being platform independent because it is highly portable and operates in the virtual machine system.

2. Related Studies

2.1. IoT-Cloud Fusion Virtual Machine System

The IoT-Cloud fusion virtual machine system is a virtual machine system that provides high-performance cloud computing power to the low-performance IoT device. This system can provide the computing power of a high-performance cloud server when running computationally intensive tasks, which can compensate for the drawback of virtual machine systems, and accommodate applications written in various programming languages by maintaining the platform-independent execution environment, which is advantage of existing smart virtual machine system[2-3]. Figure 1 shows the overall structure of the IoT-Cloud fusion virtual machine system.

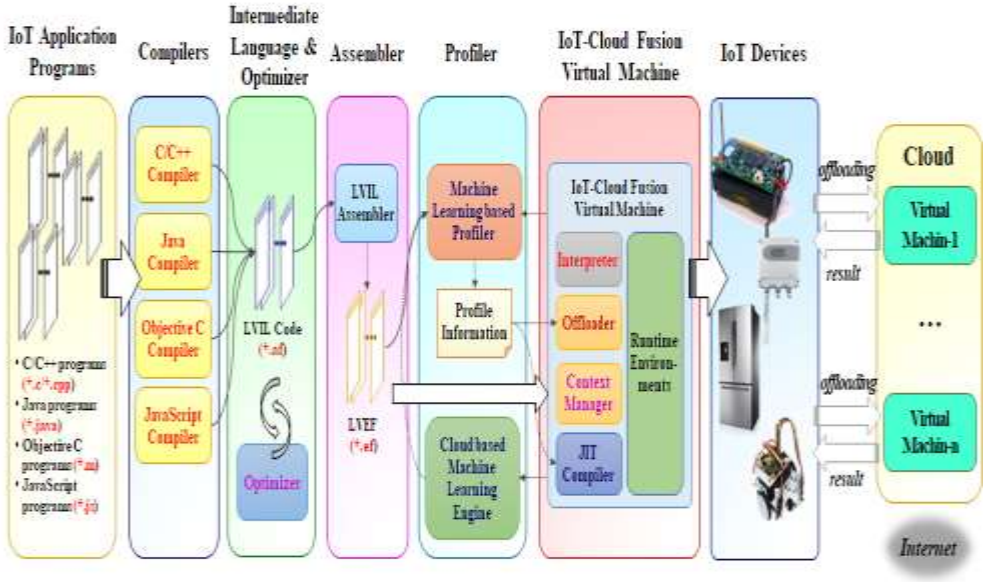

Figure 1. Overall Structure of the IoT-Cloud Fusion Virtual Machine System

The configuration of the IoT-Cloud fusion virtual machine system consists largely of a compiler, an optimizer, an assembler, a profiler, and a lightweight virtual machine. The compiler generates assembly files by compiling applications written in various languages, and the optimizer optimizes the intermediate code of the assembly file generated by the compiler to improve execution efficiency.

The assembler translates the assembly file into an executable file, which is a binary code type executable in the virtual machine, and the profiler analyzes the complexity of the function by traversing the executable file generated by the assembler on a function by function. Function complexity is used to determine offloading when an executable file is executed on a virtual machine by delegating highly computationally intensive tasks to a high-performance cloud server.

A lightweight virtual machine is a virtual machine for a low-performance IoT device that can be ported to a configuration suitable for IoT devices through a structured design,

and consumes less memory when executing an application while maintaining performance because it uses less memory when loading instructions.

2.2. Object Recognition

Object recognition is a technique for recognizing objects inside a camera image. Objects inside a video image are recognized using techniques such as contour detection and color detection.

Contour detection method [4] is one of techniques used in image processing, image analysis, image pattern recognition, and computer vision technology. It captures the point where the brightness of the image changes suddenly and detects the contour of the object. Ideally, a connection curve representing an object boundary can be created, but it is not always possible to obtain an ideal contour in a complex image.

The color detection technique is a technique of distinguishing objects by comparing the color distribution of the image of the region estimated as the object of the image. This technique can compare images from various color models, regardless of the color model, but can't detect when the colors are repositioned or blended. Since contour detection and color detection techniques have advantages and disadvantages, they can be more accurately recognized objects by supplementing them and maintaining their advantages [5-6].

3. Theft Prevention System

This paper designs and implements a theft prevention system based on a virtual machine execution system running on the IoT devices. Figure 2 shows the overall configuration of the theft prevention system.

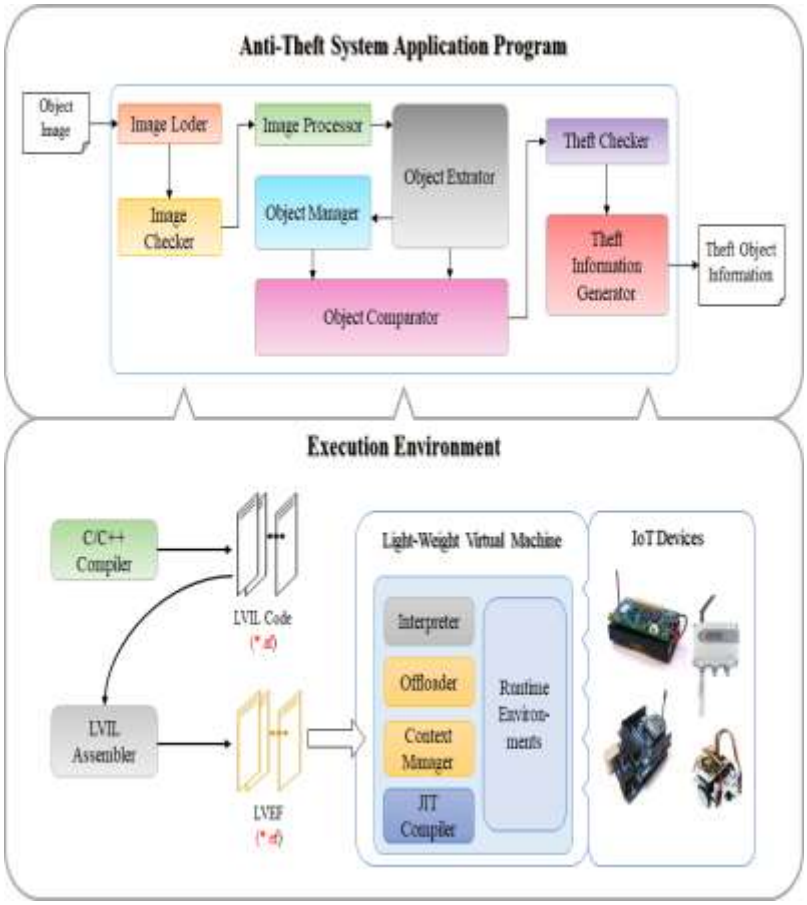

Figure 2. Configuration of the Theft Prevention System

Theft protection system consists of an antitheft application and a virtual machine system to execute it. The theft protection application program detects the object in the real-time video image using the object recognition technique and compares it with the detected object in the previous video image to detect the stolen object. The virtual machine system converts the application program into a format executable by the virtual machine for execution in the virtual machine, and executes it in an interpreted method. This theft prevention system was designed based on the virtual machine system with reference to the existing model[7-8].

3.1. Theft Protection Application Program

The anti-theft application program is an application operating in a IoT based virtual machine system, and outputs a stolen article information by inputting a real-time image image in which an object exists. The application program is largely classified into an image recognition module for loading a video image into the system, an object detection module for detecting an object in a video image, and a theft confirmation module for comparing the past object and the current object to identify the stolen article[9-10].

3.1.1. Image Recognition Module

The image recognition area consists of an image loader and an image checker. It is a preparatory step for recognizing the images captured by the IoT device and preparing the subsequent operations. The image loader converts the real-time image into an input two-dimensional matrix. The image checker analyzes the noise of the image converted from the image loader to the two-dimensional matrix and the difference from the previous image to determine whether the input image is a normal image. Figure 3 shows the configuration of the image checker system.

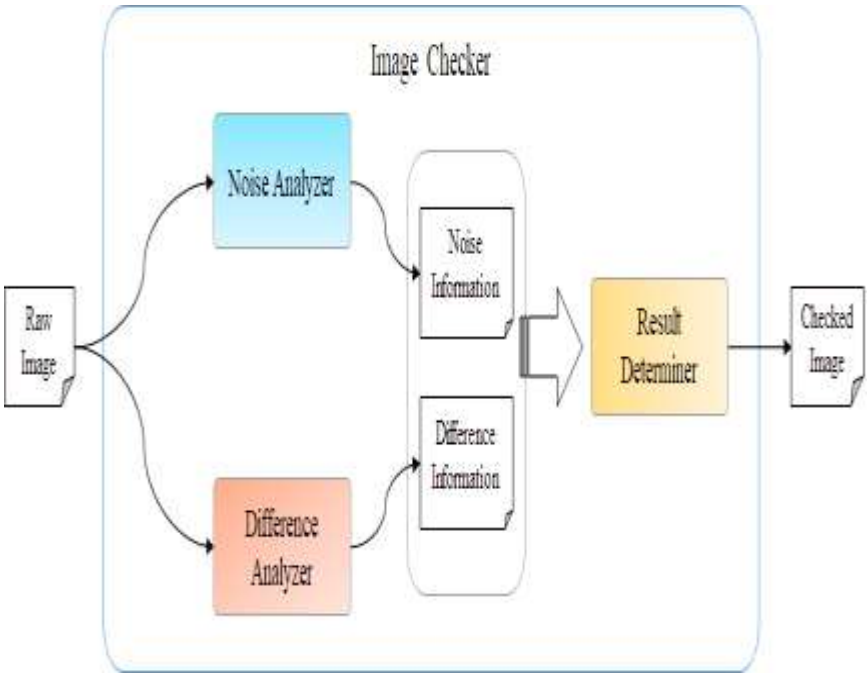

Figure 3. Configuration of the Image Checker System

The noise analyzer analyzes the noise of the input image by analyzing the noise of the real-time image, that is, the raw image. The values analyzed in the noise analyzer are used as a normal image judgment criterion.

The difference analyzer analyzes the difference between the real-time image and the previous image, and analyzes the linked value or difference between the current input image and the immediately preceding situation image. These difference numbers can determine situations such as sudden camera equipment errors.

The result determiner determines whether the current image is a normal image, that is, an image that can be used, by inputting the noise figure and difference value derived from the noise analyzer and the difference analyzer.

3.1.2. Object Detection Module

The object detection module consists of an image processor, an object extractor, and an object manager. Image processing, contour detection, and shape detection are performed to detect objects in the image. Figure 4 shows a flowchart of the object detection algorithm.

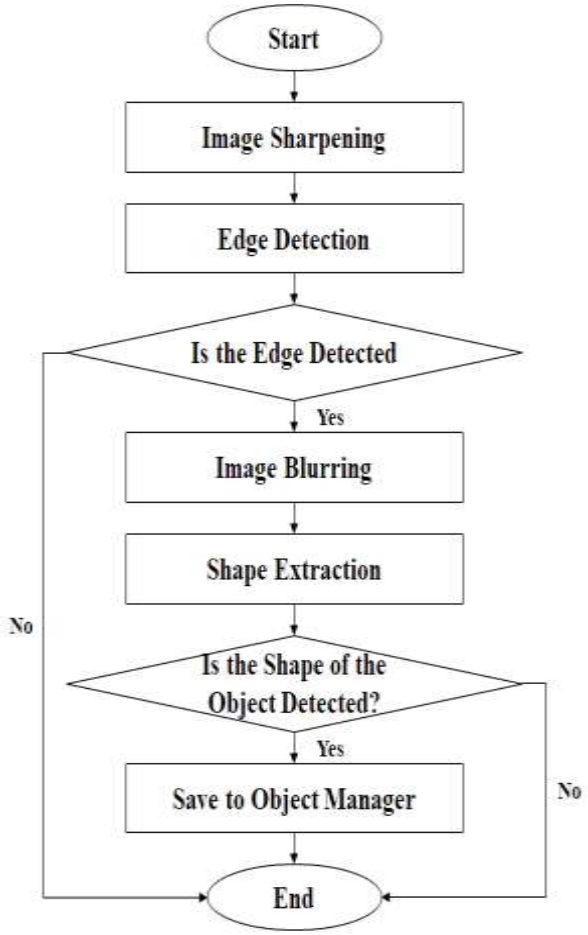

Figure 4. Flowchart of the Object Detection Algorithm

The image processor processes the image so that it is easy to detect objects in the image. Contour detection method for object detection uses contour difference to detect a contour, so contrast must be clear for accurate results. For this purpose, this paper uses color histogram equalization, sharpening and blurring. Color histogram equalization improves the image contrast with low contrast by making the distribution of the brightness values of the image uniform, and is executed first in the image processor.

Image sharpening is a process of sharpening the boundary of the object in the image. It is performed after the histogram equalization to reduce the error of the contour detection and shape detection technique for object detection. Image blurring is a process of smoothly

processing an image by removing noise from the image, which is executed after contour detection, and removes noise that is increased during the image sharpening process.

The object extractor extracts the contour, shape, position, and size information of the image object processed in the image processor. The information extracted from the object extractor is transferred to the object manager and managed, and it is used for the theft check.

3.1.3. Theft Check Module

The theft check module consists of an object comparator, a theft checker, and a stolen information generator, and checks the occurrence of theft by comparing the real image and the previous image. The object comparator compares the information of the object managed by the object manager in the object detection module with the information of the object extracted in real time from the object extractor to find the difference and convert it into the theft occurrence number.

The theft checker compiles flag information indicating that a theft situation has occurred and information of the object where theft occurred, when the theft occurrence value analyzed by the object comparator exceeds the threshold value, and transmits the information to the theft information generator. The stolen information generator draws a red rectangle indicating the theft situation on the stolen object position inside the image based on the information received from the stolen checker. Figure 5 shows a flowchart of the theft check algorithm.

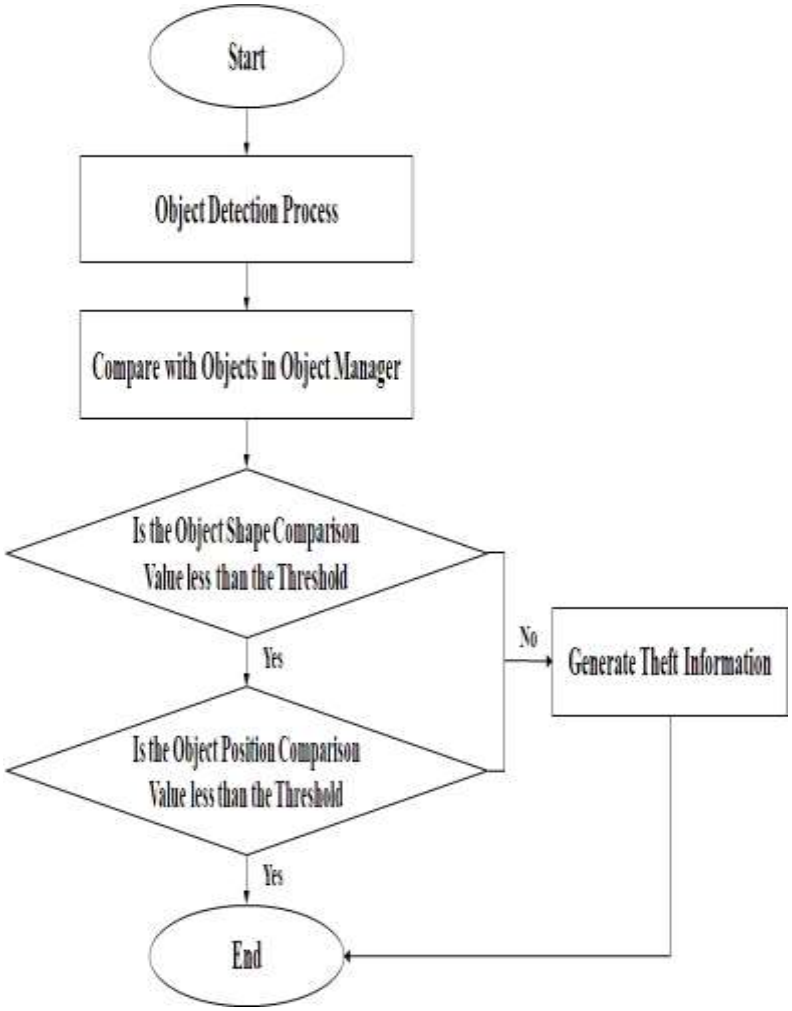

Figure 5. Flowchart of the Theft Check Algorithm

3.2. Virtual Machine System

3.2.1. C/C++ Compiler

In order for the anti-theft applications to run, they must be compiled for the execution environment. The C / C ++ compiler in this paper converts the anti-theft application into an assembly file consisting of the intermediate language LVIL for execution on the virtual machine. Figure 6 shows the LVIL code inside the assembled assembly file.

Anti-Theft.af

```
%%CodeSectionStart
    %FunctionStart
        .func_name      &main
        .opcode_start
                proc    0       1       1
                ldp
                calls   323
                 ...
    %Label      ##7
                ldp
                calls   322
                ldp
                 ...
    %Label      ##8
                ldp
                calls   326
                ujp     ##5
    %Label      ##6
                ret
    .opcode_end
  %FunctionEnd
...
```

Figure 6. LVIL Code for Anti-Theft Applications

3.2.2. LVIL Assembler

The assembly file converted by the compiler must be converted to an executable in binary code format for execution in the virtual machine. The LVIL assembler converts the LVIL code inside an assembly file into a binary code format that can be executed by a virtual machine. Because the binary code is not human readable, the LVIL assembler creates a file that records the conversion of the binary code file to binary code. Figure 7 shows how the LVIL code of the stolen application is converted to binary code.

3.2.3. Lightweight Virtual Machine

A lightweight virtual machine is a virtual machine for low-performance IoT devices that interpretively executes binary code generated by the LVIL assembler. A lightweight virtual machine consists largely of an execution layer, an offloading layer, a runtime environment layer, a portable library layer, and a just-in-time (JIT) compiler. The lightweight virtual machine of this paper deals only with offloading cloud computing techniques and local execution except JIT compiler[11-14]. Figure 8 shows the configuration of the lightweight virtual machine.

Anti-Theft.out

```
---- Header Section ----
        InitFunctionAddress : 0xFFFFFFFF
        InitFunctionName :
        EntryPointAddress : 0x00000000
        EntryFunctionName : &main
---- End of Header Section ----
        ...
---- Function Table [1] ----
        &main [188] P0 (static)
        [CS:0x00000000]   proc      0x00000000
        0x0001    0x0001
        [CS:0x0000000A]   ldp
        [CS:0x0000000C]   calls     0x00000143
        [CS:0x00000012]   not.i
        [CS:0x00000014]   fjp       0x00000022
        [CS:0x0000001A]   ldc.i     0x00000000
        [CS:0x00000020]   retv.i
                          ...
        [CS:0x000000AC]   ldp
        [CS:0x000000AE]   calls     0x00000146
        [CS:0x000000B4]   ujp       0x0000006E
        [CS:0x000000BA]   ret
---- End of Function Table ----

---- Label Table [9] ----
        Name : ##0
        CS Offset : 0x00000022

        Name : ##1
        CS Offset : 0x00000022
        ...
---- End of Label Table ----
        ...
```

Figure 7. Binary Code Conversion Process of Theft Application

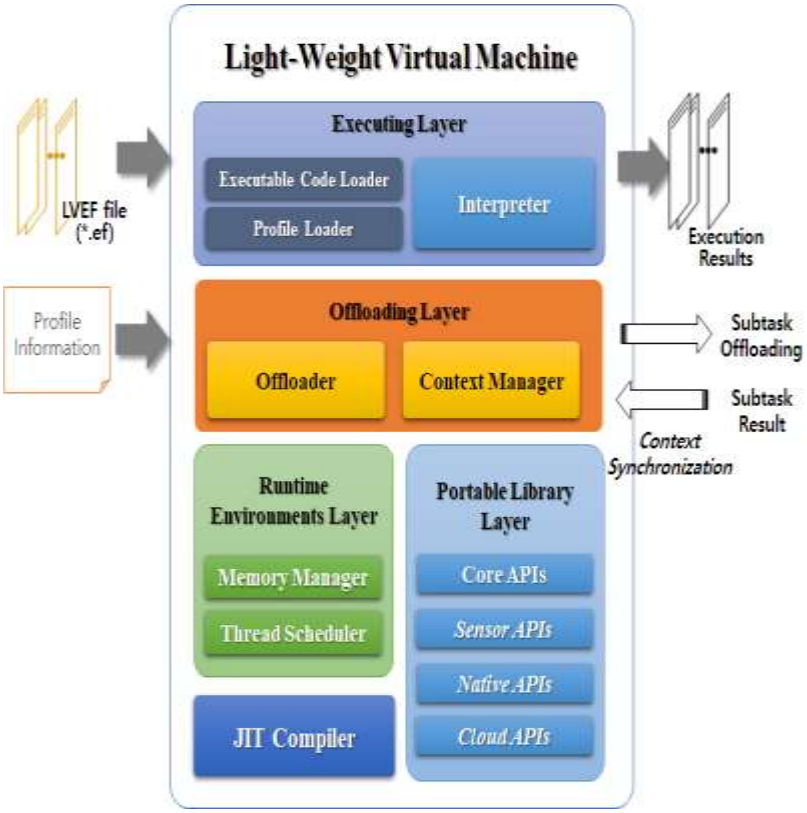

Figure 8. Configuration of the Lightweight Virtual Machine

The execution layer of the lightweight virtual machine is a layer that executes binary code generated by the LVIL assembler, which consists of an executable code loader, a profile loader, and an interpreter. The executable code loader and profile loader loads the binary code for executing the executable and the profile information analyzed in the profiler for offloading execution into the memory space inside the virtual machine.

The interpreter sequentially executes the executable code loaded from the executable code loader and determines offloading execution to delegate high-complexity tasks to a high-performance server based on profile information.

The runtime environment layer consists of a memory manager and a thread scheduler. The memory manager manages the memory used when the program is executed in the interpreter, and the thread scheduler schedules the thread, which is an execution flow in the program, in priority order. The portable library layer is a layer that provides an interface to support the sensor, native execution, etc. of the IoT devices.

In this paper, we have added an interface for the implementation of the anti-theft system to the portable library. The anti-theft system implementation interface was implemented as a wrapper function to support native execution.

4. Experimental Results and Analysis

In this section, we conduct an experiment that assumes the theft situation to confirm the execution of the anti-theft system using the IoT devices, and confirm the execution of the anti-theft system contour detection, article detection, and theft detection. Table 1 shows the execution environment of the anti-theft system.

Table 1. Execution Environment of the Anti-Theft System

Device	RaspberryPi 3 model B
Processor	Broadcom BCM2837 1.2GHz
Memory	1GB RAM
O.S	Raspbian

The anti-theft system detects the contours by processing images to detect objects in real-time video images. Figure 9 shows the result of detecting the outline in real-time image.

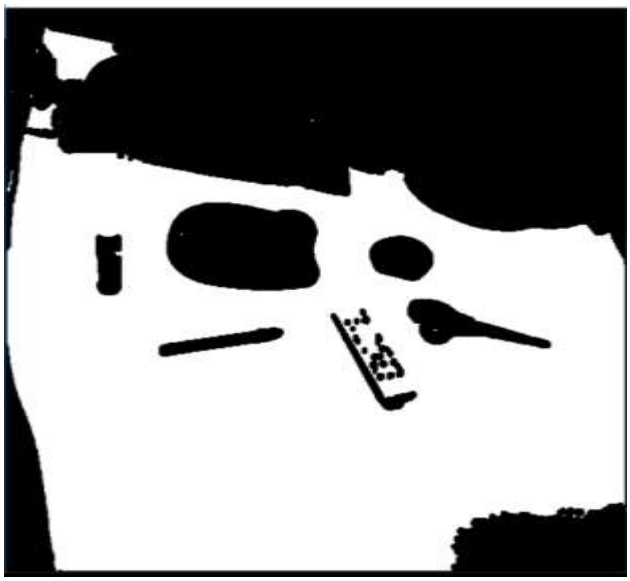

Figure 9. Outline Detection Result

The outline detection result in Figure 9 is used as a basis for detecting the object by the anti-theft system. Figure 10 shows the result of the detection of the object by the anti-theft system based on the outline detection result.

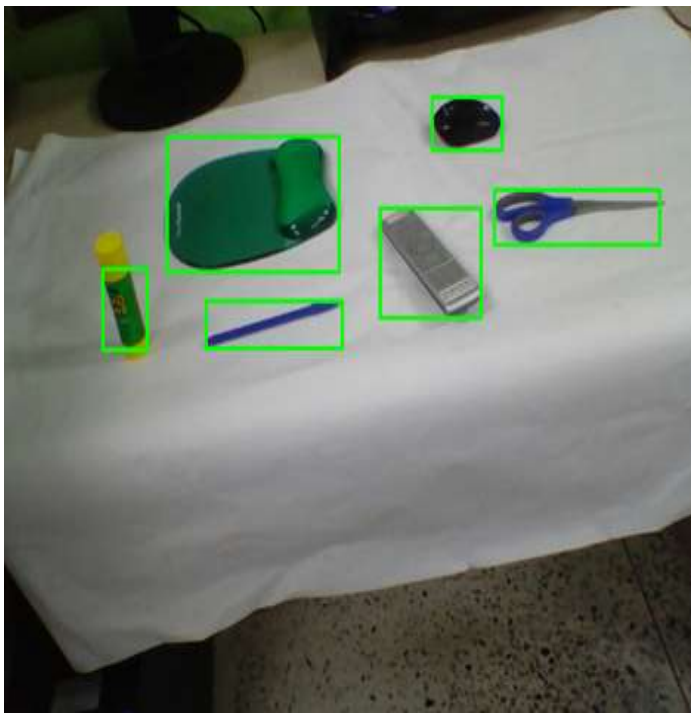

Figure 10. Object Detection Result

The anti-theft system compares the detected objects in Figure 10 with the detected objects in the subsequent real-time video to detect the items that are stolen. Figure 11 shows the results of the theft detection of the theft prevention system. In Figure 11, a stolen item is indicated by a red square to indicate the occurrence of the theft.

Figure 11. Stolen Object Detection Result

5. Conclusions and Further Researches

In this paper, we designed and implemented a theft prevention system based on a virtual machine system running on the IoT devices. The theft prevention system implemented in this paper is divided into an application program and an execution environment. The application program consists of an image recognition module, an object detection module, and a theft confirmation module. The execution environment layer consists of a compiler, an assembler, and a virtual machine.

The application layer recognizes and processes the video image, and detects and compares the object using contour detection and shape extraction. The execution environment compiles and assembles the application to create an executable file of the virtual machine, and the virtual machine executes it in an interpreted way. This virtual machine system has advantages of high portability and platform independence because application programs developed in various languages are implemented without platform or operating system constraints and implement OSMU (One Source Multi Use).

The theft prevention system implemented in this paper lacks means to notify the occurrence of theft, and the virtual machine execution system is currently only capable of local execution. Therefore, we will be notified of the occurrence of the theft by interworking with the IoT device and the smart phone in the further study. In addition, the offloading technique [7-9] will be applied to judge the amount of computation of the anti-theft system, and to execute a large computational load on a high-performance cloud server. To increase a local execution performance, JIT (Just-In-Time) Compiler [10] will be used so that the load-intensive tasks in the virtual machine can be executed in the native environment.

Acknowledgements

This research was supported by Seokyeong University in 2018.

References

[1] C. Michael, L. Markus and R. Roger, “The Internet of Things”, McKinsey Quarterly, **(2014)**.
[2] Y. Lee and Y. Son, “A Study on the Smart Virtual Machine for Executing Virtual Machine Codes on Smart Platforms”, International Journal of Smart Home, SERSC, vol. 6, no. 4, **(2012)**, pp. 93-105.
[3] Y. Lee and Y. Son, “A Study on the Smart Virtual Machine for Smart Devices”, Information -an International Interdisciplinary Journal, International Information Institute, Japan, vol. 16, no. 2, **(2013)**, pp. 1465-1472.
[4] D. Marr and E. Hildreth, “Theory of Edge Detection”, Proceedings of the Royal Society of London, Series B, Biological Sciences, vol. 207, no. 1167, **(1980)**, pp. 187-217.
[5] S. Birchfield, “Elliptical Head Tracking Using Intensity Gradient and Color Histograms”, IEEE Conf. Computer Vision and Pattern Recognition, **(1998)**, pp. 232-237.
[6] D. Comaniciu, “Kernel-Based Object Tracking”, IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 25, no. 5, **(2003)**, pp. 564-575.
[7] M. Heo, “U-Antitheft System: Intelligent Notebook Theft Prevention System using USN Module)”, The Journal of Korea Information Technology Convergence Society, vol. 3, no. 1, **(2010)**, pp. 11-24.
[8] K. Lee and Y. Jung, “Implementation of Real-Time Security System by using Dual Camera”, The Journal of Korea Academia-Industrial cooperation Society, vol. 10, no. 1, **(2009)**, pp. 158-164.
[9] D. Sim and U. Lim, “Digital Image Processing”, Hanbit Publishing Network, **(2017)**.
[10] C. Kim, H. Jung and H. Yang, “Digital Image Processing”, Life and Power Press, **(2016)**.
[11] Y. Son and Y. Lee, “Offloading Method for Efficient Use of Local Computational Resources in Mobile Location-Based Services Using Clouds”, Mobile Information Systems, Netherlands Hindawi Publishing Corp., vol. 2017, **(2017)**, pp. 1-9.
[12] K. Yang, S. Ou and H. H. Chen, “On Effective Offloading Services for Resource-Constrained Mobile Devices Running Heavier Mobile Internet Applications”, IEEE Comm. Magazine, vol. 46, no. 1, **(2008)**, pp. 56-63.
[13] K. Kumar, “A Survey of Computation Offloading for Mobile Systems”, Mobile Networks and Applications, vol. 18, no. 1, **(2013)**, pp. 129-140.
[14] A. Krall, “Efficient JavaVM Just-in-Time Compilation”, Proceedings of the 1998 International Conference on Parallel Architectures and Compilation Techniques, **(1998)**, pp. 54-61.

Authors

JaeHyun Kim, he received the B.S. degree from the Dept. of Mathematics, Hanyang University, Seoul, Korea, in 1986, and M.S. and Ph.D. degrees from Dept. of Statistics, Dongguk University, Seoul, Korea in 1989 and 1996, respectively. He was a chairman of Dept. of Internet Information 2002-2007. Currently, he is a member of the Korean Data & Information Science Society and a Professor of Dept. of Computer Engineering, Seokyeong University, Seoul, Korea. His research areas include mobile programming, cloud system and big data analysis.

Yangsun Lee, he received the B.S. degree from the Dept. of Computer Science, Dongguk University, Seoul, Korea, in 1985, and M.S. and Ph.D. degrees from Dept. of Computer Engineering, Dongguk University, Seoul, Korea in 1987 and 2003, respectively. He was a Manager of the Computer Center, Seokyeong University from 1996-2000, a Director of Korea Multimedia Society from 2004-2018, a General Director of Korea Multimedia Society from 2005-2006, a Vice President of Korea Multimedia Society in 2009, and a Senior Vice President of Korea Multimedia Society in 2015-2016. Also, he was a Director of Korea Information Processing Society from 2006-2014 and a President of a Society for the Study of Game at Korea Information Processing Society from 2006-2010. And, he was a Director of HSST from 2014-2018. Currently, he is a Professor of Dept. of Computer Engineering, Seokyeong University, Seoul, Korea. His research areas include smart system solutions, programming languages, and embedded systems.