# Method Resource Sharing in On-Premises Environment Based on Cross-Origin Resource Sharing and its Application for Safety-First Constructions

Kohei Arai[1], Kodai Norikoshi[2], Mariko Oda[3]
Information Science Department, Saga University, Saga City, Japan[1]
Information Network Department, Kurume Institute of Technology, Kurume City, Japan[1, 2, 3]

***Abstract*—The method of resource sharing in an on-premises environment based on Cross-Origin Resource Sharing (CORS) is proposed for security reasons. However, using CORS entails several risks: Cross-Site Request Forgery (CSRF), difficulties in secure configuration, handling credentials, controlling complex requests, and restrictions associated with using wildcards. (1) To mitigate these risks, the following countermeasures are proposed: (2) Use CSRF tokens and the "SameSite" attribute. (3) Minimize preflight requests by allowing only specific origins. (4) Use the "withCredentials" flag or set the "Access-Control-Allow-Credentials" header on the server. (5) Handle custom headers by adding the required headers to CORS settings. (6) Specify a specific origin in the "Access-Control-Allow-Origin" header instead of using wildcards. Additionally, applying CORS for safety-first constructions, which helps raise awareness of dangerous actions in construction fields, is also being explored.**

***Keywords*—Cross-Origin Resource Sharing: CORS; CSRF (Cross-Site Request Forgery); SameSite; withCredentials flag; Access-Control-Allow-Credentials header; safety first constructions**

## I. Introduction

Cross-Origin Resource Sharing (CORS) is a security feature of web browsers that uses additional HTTP headers to control how web applications can access resources located on different origins. This mechanism allows secure data exchange between different origins, enabling browsers and servers to communicate safely. CORS prevents malicious websites from accessing other sites' data without explicit permission.

When CORS fails, JavaScript cannot determine the specific error due to security restrictions. Instead, developers must check the browser's console for detailed error information. Although CORS was introduced to address security issues, it presents several challenges:

CSRF (Cross-Site Request Forgery) Risk: Allowing cross-origin requests can increase the risk of CSRF attacks, where an attacker could exploit the victim's browser to perform unintended actions on behalf of an authenticated user. Careful management of cross-origin requests is essential.

Difficulty in Secure Setup: Proper CORS configuration is required on both the server and client sides. Incorrect configurations can lead to security vulnerabilities.

Credential Handling: Browsers do not include authentication information (such as cookies or HTTP authentication) in cross-origin requests by default. Enabling this requires careful configuration on both the server and client sides.

Controlling Complex Requests: Handling complex requests, such as preflight requests or custom headers, requires meticulous configuration to ensure security.

Restrictions Associated with Using Wildcards: Using wildcards in the Access-Control-Allow-Origin header grants access to all origins, reducing security. Specifying a specific origin is preferable for enhanced security.

In this paper, we propose countermeasures to mitigate these risks. We also explore the application of secure CORS in a local development environment. Specifically, we develop a web-based system that runs in browsers to provide construction workers with YouTube content highlighting dangerous actions as a safety-first measure. By limiting CORS to trusted domains and allowing only necessary methods and headers, we minimize associated risks. This ensures construction workers can view safety videos on their smartphones before starting work.

The following sections outline the research background and related work in Section II, detail the proposed countermeasures for CORS risks in Section III. The web application system for local content delivery is given in Section IV. Results, conclusion and future research work is given in Section V, VI and VII respectively.

## II. Research Background and Related Research Works

### A. Research Background

Considering the occurrence of occupational accidents in the construction industry in recent years, although the number of such accidents in Japan has been decreasing over the long term, approximately 300 people still die each year. The most common fatalities and injuries in the construction industry are falls, accounting for approximately 40% of fatalities and 30% of all injuries. From 2017 to 2021, 162 accidents occurred while working at heights using ladders, stepladders, etc. [1]. The primary causes of these accidents are improper handling or carelessness by users, such as not using appropriate lifting

equipment, leaning over and falling, and inadequate environmental preparation around work sites.

To prevent such accidents, we have developed and verified a web application aimed at preventing falls while working at heights using ladders, stepladders, etc. The web application allows workers to check and select their daily tasks and displays relevant videos and text instructions. These videos are stored and managed on a video distribution platform. After development, workers can access and use the URL on their mobile devices. Additionally, an administrator's web application tracks the number of views of the videos played on the workers' web application to ensure it is being used appropriately. The database stores the ID of each video, which is required to obtain the number of video views.

### *B. Related Research Works*

With the application of BIM/CIM principles, various initiatives using CIM models have been proposed. Instead of traditional construction briefs and design drawings, we use highly expressive CIM models with VR goggles and tablet devices to promote a three-dimensional understanding of construction work [2]. VR technology is employed not only to visualize work progress and discrepancies between design drawings and construction drawings but also to simulate dangerous locations and movements. Consequently, robust security features for web applications are essential.

Several research works focus on detecting dangerous actions:

A comparative study on discrimination methods for identifying dangerous red tide species using wavelet-based classification methods [3].

A method for detecting dangerous actions by cars using wavelet Multi-Resolution Analysis (MRA) based on the appropriate support length of the base function [4].

Research on pedestrian safety through eye contact between autonomous cars and pedestrians [5].

In addition, research on web application services and systems includes:

A wearable computing system with input and output devices based on eye-based Human-Computer Interaction (HCI), enabling location-based web services [6].

Numerical representation of websites of remote sensing satellite data providers and its application to knowledge-based information retrieval with natural language processing [7].

A mashup-based e-learning content search engine for mobile learning using Yahoo! Search and web APIs [8].

A web-based data acquisition and management system for GOSAT validation Lidar data analysis [9].

Improvements to the web-based data acquisition and management system for GOSAT validation Lidar data analysis [10].

A method for Web GIS systems applicable to assimilation model database constructions [11].

These research efforts demonstrate the integration of advanced technologies in both dangerous action detection and web application services, highlighting the need for secure, efficient, and user-friendly systems.

## III. COUNTERMEASURES FOR CORS RISKS

CORS (Cross-Origin Resource Sharing) is a mechanism for controlling resource access from different origins (domains, protocols, ports) in web browsers. Below is an illustration of the CORS concept:

Client (browser): The browser in which the user opens the web page.

Website A (Origin A): The site where the page is hosted, e.g., https://example.com.

Website B (Origin B): An external site, e.g., https://api.example.com.

Request: JavaScript on Website A sends an HTTP request to Website B to retrieve data.

Preflight request: Before the actual request, the browser sends an OPTIONS request to check if the web server is authorized.

Verifying CORS headers: The web server responds with CORS headers. If they are not present or incorrect, the browser denies the request.

Data retrieval: If the server permits, the original request is sent, and the data is retrieved.

This flow controls cross-origin requests and improves security. Proper CORS configuration is necessary on both the server and client sides.

Important Notes on the CORS Specification:

Same-origin policy: For security reasons, the browser restricts direct access from one origin to another. CORS helps to overcome this limitation.

Preflight requests: These are made before the actual request if it contains unsafe methods (e.g., POST, PUT) or certain headers.

Requirement of CORS headers: The server must set appropriate CORS headers, including Access-Control-Allow-Origin (specifying allowed origins), Access-Control-Allow-Methods, and Access-Control-Allow-Headers.

Handling credentials: By default, browsers do not include credentials in cross-origin requests. If needed, set the withCredentials flag and enable credential handling on both server and client.

Restrictions on using wildcard origins: While a wildcard (*) in Access-Control-Allow-Origin allows access from all origins, specifying a specific origin is more secure.

By considering these aspects, you can implement CORS properly and build secure web applications. CORS uses additional HTTP headers to instruct the browser to grant a web application running in one origin access to specific resources in a different origin. When a web application requests a resource

from a different origin, the browser performs a cross-origin HTTP request.

Problems of CORS and Their Countermeasures:

Risk of CSRF attacks: Use CSRF tokens and the SameSite attribute.

Difficulty in configuring secure CORS: Minimize preflight requests by allowing only specific origins.

Handling credentials: Use the withCredentials flag or set the Access-Control-Allow-Credentials header on the server.

Controlling complex requests: Handle custom headers or add the required headers to CORS settings.

Limitations of using wildcards: Specify a specific origin in the Access-Control-Allow-Origin header instead of using wildcards.

By addressing these countermeasures, the proposed web application services are designed to be secure and efficient.

## IV. Web Application System for Content Providing in a Local Environment

### A. Development Environment

At construction sites, KY (Kiken Yochi, or hazard prediction) activities are conducted to improve workers' safety awareness. The purpose of these activities is to predict potential dangers before starting work, enabling workers to take countermeasures and prevent accidents. Creating a safe worksite environment and preventing accidents are crucial for construction companies. To support effective and non-burdensome KY activities, we have developed safety education videos (onsite hazard prediction videos) with features for presentation and for checking and managing workers' safety awareness and behavior.

We used Docker [12], FastAPI [13], and React [14] to develop the web applications. After checking the day's work details, workers can access the hazard prediction video page via the worker web application. The site hazard prediction videos, hosted on YouTube, will be played. The playback count information is accessible from the administrator's web application, allowing supervisors to issue warnings to workers who have not watched the videos.

There are two web applications: one for workers and one for supervisors. The development environment is as follows:

OS: Windows 11

Editor: Microsoft Visual Studio Code

Languages Used: TypeScript, CSS

Server: Vercel [15]

Virtual Environment: Docker Engine v24.0.6 [16]

Additional Languages: JavaScript, Python, HTML, Dockerfile

The administrator web application was developed using a containerized virtual environment with Docker. The container setup is divided into three parts: one for the server, one for the database, and one for the web application.

### B. Web Applications for Workers

Additionally, during the development of the web application for workers, we used a virtual environment provided by Python without containerizing it, to verify actual operation on mobile devices.

The application includes the following functions:

Login Function: This function has a high priority and is used to create and register a worker's user account and log in. It also helps track the usage status of workers. By registering each user, the application can count the number of views for each user, thereby monitoring their engagement.

Danger Video Viewing and Precaution Display: This function also has a high priority. Workers can select and watch hazard prediction videos. Precautions related to the work are displayed for workers to review. Additionally, when a video is selected, the application counts the views and updates the relevant table to keep track of this information.

### C. Web Applications for Supervisors

On the other hand, regarding functions, we consider checking the number of views of dangerous videos by each worker, managing dangerous videos, and implementing a login function. Here are the details:

View Count of Dangerous Videos: This function has high priority. It involves checking how many times a worker has viewed a dangerous video. By ensuring that workers have watched appropriate videos for the day's tasks, the system can effectively monitor video usage. This function is essential for assessing worker engagement with safety content.

Management of Dangerous Videos: This function has medium priority. It involves accessing the YouTube channel page where dangerous videos are stored and managed. The system posts videos for workers to watch, with channels set to limit access.

Login Function: This function also has high priority. It allows for the creation and registration of worker user accounts and facilitates logging in. The login function provides insight into the usage status of workers. By registering each user, the system counts the number of video views per user, providing comprehensive usage statistics.

Given that workers are expected to view dangerous videos at job sites, the application layout is designed exclusively for mobile devices like smartphones and tablets. In contrast, administrators are assumed to review all information in a control room, so the layout is tailored for PC screens.

### D. Functionalities of Web Applications

Fig. 1 illustrates the overall system diagram of the proposed web application. The operational requirements for the server (cloud-provided system) are detailed below:

Server Operation Mode: The server will operate in Autopilot mode on Google Kubernetes Engine (GKE) [17]. In Autopilot mode, Google Cloud automatically manages and scales nodes,

and you only pay for the resources required to run your workloads. This reduces server operating costs and management efforts. Notably, since nodes are managed by GKE, there are no charges for unused node capacity, system pods, operating system costs, or unscheduled workloads.

Server Network Mode: The server will adopt the VPC native cluster [18] for network mode. VPC native clusters assign IP addresses from the VPC network to nodes and Pods, improving network performance and security. This setup also facilitates communication with other resources within the VPC network.

Server Configuration: Web Server: Deployed as a container using FastAPI, a fast and modern Python web framework ideal for developing RESTful APIs [19].

Database: The database container utilizes Cloud SQL [20], Google Cloud Platform's fully managed relational database service supporting database engines such as MySQL and PostgreSQL.

Application Container: Cloud Run is used as the container platform for the application. Cloud Run is GCP's serverless container platform, enabling you to run container images using any language or library.

Security and Availability:

Server security and availability are maintained according to GCP's best practices.

Containers are stored in encrypted storage and communicate using SSL/TLS.

Containers are distributed across multiple zones and regions to enhance resilience against failures and disasters.

This configuration ensures secure, scalable, and cost-effective operation of the web application on Google Cloud Platform.

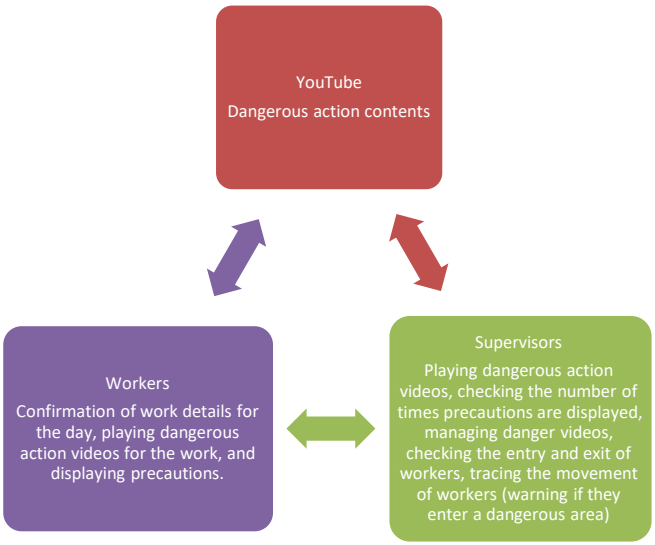


Fig. 1. Functional requirements.

### E. Network Configurations

Regarding network configuration, CORS is essential for web applications running in a local development environment. CORS (Cross-Origin Resource Sharing) uses additional HTTP headers to instruct the browser to allow a web application running in one origin to access specific resources in another origin. When a web application requests a resource from a different origin, the browser performs a cross-origin HTTP request.

For instance, if the front-end JavaScript code of a web application served from https://website-1.com makes a request to https://api-server.com/data-info using XMLHttpRequest, CORS ensures that this request is allowed if the appropriate CORS headers are set.

The same-origin policy restricts web applications to requesting resources only from the origin they are loaded from. CORS is implemented to relax this restriction securely. Fig. 2 illustrates an example of the CORS operation flow.

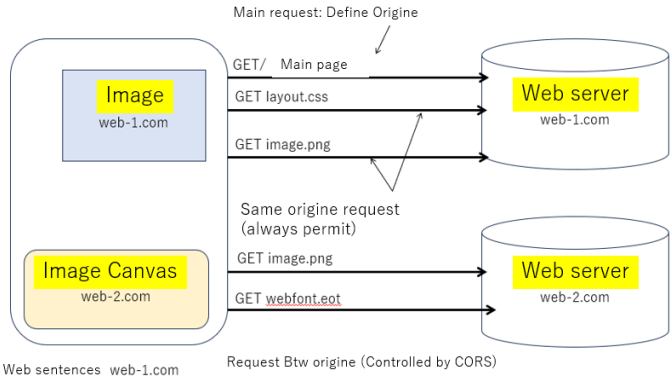


Fig. 2. CORS operations.

CORS works by adding new HTTP headers that allow servers to specify which origins are permitted to access their resources from a web browser. Additionally, browsers use the HTTP OPTIONS request method for certain HTTP methods (especially those other than GET and POST, and those with specific MIME types) that can have side effects on server data. This preflight request asks the server to indicate the methods supported before sending the actual request with proper authorization.

The server can also instruct the client whether it should include credentials (such as cookies or HTTP authentication) in the request. Therefore, these aforementioned considerations are necessary for implementing CORS securely.

## V. Development Results

To prevent falls while working at heights, we integrated a web application for workers that displays dangerous videos and precautions, a database for managing these resources, and a system to track the number of video views per worker. We developed and studied a cloud computing system for this purpose.

The web application for workers allows them to use a mobile device to select tasks and view danger videos and precautions. The administrator's web application monitors the number of video views from the worker's interface to ensure appropriate usage. Additionally, an authentication function was introduced to enhance security and reliability in system development, particularly in security-critical environments.

This initiative aims to prevent falls while working at heights, which is the primary objective of this research. We achieved the development of features such as "confirmation of the number of times each worker has viewed a dangerous video" and a "login function." Upon signing up and signing in, users are directed to the main page where they can check task details and view statistics on video usage by workers. As an example of the developed web applications, the supervisor's main page is shown in Fig. 3.

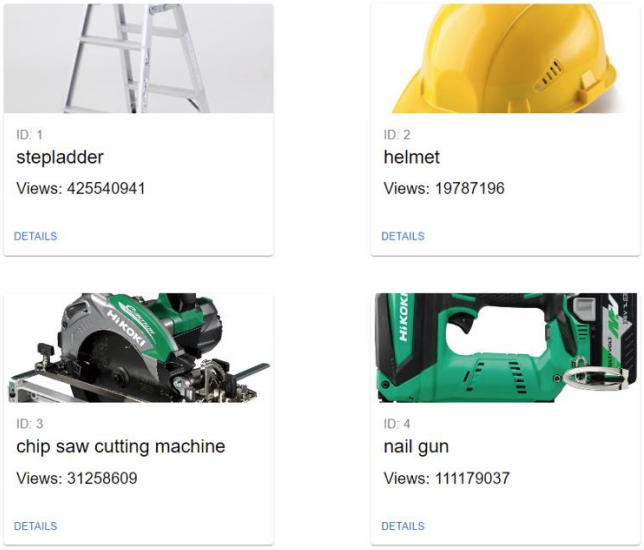


Fig. 3. Main page for the supervisor.

Fig. 4 shows an example of awareness video contents of dangerous action on stepladder in Japanese for workers' safety-first.


Fig. 4. Example of awareness video contents of dangerous action on stepladder in Japanese for workers' safety-first.

## VI. Conclusion

A method for resource sharing in an on-premises environment using Cross-Origin Resource Sharing (CORS) is proposed for security reasons. However, using CORS presents several risks: CSRF (Cross-Site Request Forgery) risks, difficulties in secure configuration, handling credentials, controlling complex requests, and restrictions associated with using wildcards. To mitigate these risks, the following countermeasures are recommended:

Use CSRF tokens and the SameSite attribute.

Minimize preflight requests by allowing only specific origins.

Use the withCredentials flag or set the Access-Control-Allow-Credentials header on the server.

Handle custom headers or add required headers to CORS settings.

Specify a specific origin in the Access-Control-Allow-Origin header instead of using wildcards.

Additionally, the application of these measures in safety-first construction scenarios, aimed at increasing awareness of dangerous actions on construction sites, is being explored.

Through the integration of the proposed web application system, it was found that the major risks associated with CORS can be mitigated with these countermeasures. By using a risk-avoided CORS configuration, a web application system focused on safety for construction workers was developed. This system includes features such as tracking the number of times each worker has viewed a dangerous video and a login function. Once users sign up and log in, they are directed to the main page where they can check the number of views for each task and track the number of views by each worker.

## VII. Future Research Works

Since the developed web application could not be tested in the field, we were unable to obtain feedback from the actual users. User feedback is crucial for improving the UI, so collecting data from actual usage is necessary. Additionally, we were unable to develop a login function or create a container for the worker's web application. Therefore, enhancing the quality of actual operations, including implementing future security measures and deployment strategies, is necessary.

For the administrator's web application, there are still issues to address, such as "developing pages for video management" and "deploying on the cloud." While an authentication function has been introduced as a security measure, it has not yet been tested in a real-world environment. Consequently, its resistance to potential attacks still needs to be evaluated.

## Acknowledgment

The authors would like to thank to Professor Dr. Hiroshi Okumura and Professor Dr. Osamu Fukuda of Saga University for their valuable discussions.

## References

[1] https://www.kensaibou.or.jp/index.html, https://www.nite.go.jp/jiko/chuikanki/mailmagazin/2022fy/vol414_221011.html

[2] https://www.kkr.mlit.go.jp/plan/happyou/thesises/2023/lbhrsn000000m6ag-att/a1684912390134.pdf

[3] Kohei Arai, Comparative study on discrimination methods for identifying dangerous red tide species based on wavelet utilized classification methods, International Journal of Advanced Computer Science and Applications, 4, 1, 95-102, 2013.

[4] Kohei Arai, Tomoko Nishikawa, Method for car in dangerous action detection by means of wavelet Multi-Resolution Analysis based on appropriate support length of base function, International Journal of Advanced Research in Artificial Intelligence, 2, 4, 13-17, 2013.

[5] Kohei Arai, Akihiro Yamashita, Hiroshi Okumura, Pedestrian safety with eye contact between autonomous car and pedestrian, International Journal of Advanced Computer Science and Applications IJACSA, 10, 5, 161-165, 2019.

[6] Kohei Arai, Wearable computing system with input output devices based on eye-based Human Computer Interaction: HCI allowing location-based web services, International Journal of Advanced Research in Artificial Intelligence, 2, 8, 34-39, 2013.

[7] Kohei Arai, Numerical representation of web sites of remote sensing satellite data providers and its application to knowledge-based information retrievals with natural language, International Journal of Advanced Research in Artificial Intelligence, 2, 10, 26-31, 2013.

[8] Kohei Arai, Yahoo! Search and web API utilized mashup-based e-learning content search engine for mobile learning, International Journal of Advanced Research on Artificial Intelligence, 4, 6, 1-7, 2015.

[9] H. Okumura, S. Takubo, T. Kawsaki, I.N. Abdulah, T. Sakai, T. Maki, Kohei Arai, Web based data acquisition and management system for GOSAT validation Lidar data analysis, Proceedings of the SPIE Vol.8537, Conference 8537: Image and Signal Processing for Remote Sensing, Paper #8537-43, system, 2012.

[10] Hiroshi Okumura, Shoichiro Takubo, Takeru Kawasaki, Indra Nugraha Abdulah, Osamu Uchino, Isamu Morino, Tatsuya Yokota, Tomohiro Nagai, Tetu Sakai, Takashi Maki, Kohei Arai, Improvement of web-based data acquisition and management system for GOSAT validation Lidar data analysis (2013), SPIE Electronic Imaging Conference, 2013.

[11] Kohei Arai, Method for Web. GIS System Applicable to Assimilation Model Database Constructions, Proceedings of the Future Technology Conference 2021, 2021.

[12] Docker,Docker: Accelerating container application development, [online] https://www.docker.com/ja-jp/ (accessed January 1, 2024).

[13] FasstAPI, [online] https://fastapi.tiangolo.com/ja/ (accessed January 1, 2024).

[14] React,React, [online] https://ja.react.dev/(accessed January 1, 2024).

[15] Build and deploy the best Web experiences with The Frontend Cloud – Vercel, [online]URL, https://vercel.com/.

[16] Docker Engine v24.0.6 https://docs.docker.com/engine/release-notes/24.0/.

[17] Google Kubernetes Engine https://www.cloudskillsboost.google /course_templates/2.

[18] VPC native cluster https://cloudacademy.com/course/advanced-cluster-options-gke-3500/routes-based-vs-vpc-native/.

[19] RESTful APIs https://blog.hubspot.com/website/what-is-rest-api.

[20] cloudsql gcp https://console.cloud.google.com/marketplace/product/google-cloud-platform/cloud-sql?pli=1&project=serene-bonbon-368602.

## AUTHORS' PROFILE

Kohei Arai, He received BS, MS and PhD degrees in 1972, 1974 and 1982, respectively. He was with The Institute for Industrial Science and Technology of the University of Tokyo from April 1974 to December 1978 also was with National Space Development Agency of Japan from January, 1979 to March, 1990. During from 1985 to 1987, he was with Canada Centre for Remote Sensing as a Post Doctoral Fellow of National Science and Engineering Research Council of Canada. He moved to Saga University as a Professor in Department of Information Science on April 1990. He was a councilor for the Aeronautics and Space related to the Technology Committee of the Ministry of Science and Technology during from 1998 to 2000. He was a councilor of Saga University for 2002 and 2003. He also was an executive councilor for the Remote Sensing Society of Japan for 2003 to 2005. He is a Science Council of Japan Special Member since 2012. He is an Adjunct Professor of University of Arizona, USA since 1998. He also is Vice Chairman of the Science Commission "A" of ICSU/COSPAR since 2008 then he is now award committee member of ICSU/COSPAR. He wrote 87 books and published 710 journal papers as well as 650 conference papers. He received 66 of awards including ICSU/COSPAR Vikram Sarabhai Medal in 2016, and Science award of Ministry of Mister of Education of Japan in 2015. He is now Editor-in-Chief of IJACSA and IJISA. http://teagis.ip.is.saga-u.ac.jp/index.html

Kodai Norikoshi, He received BE degree from Kurume Institute of Technology in 2024.

Mariko Oda, She graduated from the Faculty of Engineering, Saga University in 1992, and completed her master's and doctoral studies at the Graduate School of Engineering, Saga University in 1994 and 2012, respectively. She received Ph.D(Engineering) from Saga University in 2012. She also received the IPSJ Kyushu Section Newcomer Incentive Award. In 1994, she became an assistant professor at the department of engineering in Kurume Institute of Technology; in 2001, a lecturer; from 2012 to 2014, an associate professor at the same institute; from 2014, an associate professor at Hagoromo university of International studies; from 2017 to 2020, a professor at the Department of Media studies, Hagoromo university of International studies. In 2020, she was appointed Deputy Director and Professor of the Applied of AI Research Institute at Kurume Institute of Technology. She has been in this position up to the present. She is currently working on applied AI research in the fields of education.