Software Support for Skeleton Recognition and Monitoring People with Privacy

Yoshihisa Nitta
Department of Computer Science
Faculty of Liberal Arts
Tsuda University
nitta@tsuda.ac.jp

Yuko Murayama
Department of Computer Science
Faculty of Liberal Arts
Tsuda University
murayama@tsuda.ac.jp

Abstract

In this research, we have developed an open source software tool which makes it easy for a user to get skeleton information of people in a live image by use of Kinect for Windows V2. Our tool is a set of library software which provides users with easy coding to get body information and face recognition. The tool has been distributed widely and used by several users already for their research work such as robotics. In this paper, we propose possible use cases such as a remote monitor system for elderly care with privacy as well as a monitor system for shelters at disaster.

1. Introduction

It is popular for one to obtain skeleton information out of the images and moving pictures in motion captures.

The traditional way for skeleton recognition has been conducted by use of marker based systems [1][2][3]. A subject needs to put on markers around his/her body and the locations of markers are tracked and detected out of images and motion pictures taken by camera. Tracking would be done various ways such as optical motion picture and the use of an infrared camera. While marker-based systems have been used extensively, they are expensive.

Recently a more economical markerless systems, Kinect for Windows V2 [4] (hereinafter called "Kinect V2") is available. Woolford [5] suggests that such a system, the Kinect V2, it may be useable compare to a traditional clinical system, in a healthcare environment because the target person does not need to put on any marker device so that tracking would be done without making physical contact with that person. The Kinect V2 developed by Microsoft is a device with many functions such as skeleton tracking, face tracking and voice direction acquisition. Human body data can be obtained with sufficient accuracy without contact with a special marker on the human body.

Leap motion devices are markerless as well. Unfortunately they are supposed to be used only for hands [6].

The original code API for C++ with the Kinect for Windows SDK 2.0 consists of too many methods; 54 kinds of *Interface*, and the total number of methods is 277 [7]. We had opportunities for our university students to program Kinect V2 using the SDK, but they faced difficulties and could not be completed easily. The cause of difficulty was primarily due to the dependent function calls and troublesome memory management. So we felt the need to propose a slightly larger, more appropriate granularity API such that the number of methods would be reduced.

In this research, we have developed a class library *NtKinect* [8] [9] for C++ so that it is easier for programming to use the Kinect V2. The library has been released as an Open Source of MIT license.

With the library, one can easily perform skeleton tracking (Body Framework information) as well as face recognition [10][11]. The tool has been distributed widely and used by several users already for their research work such as robotics and computer art systems [12][13][14].

As an application of this library, we propose a watching over service is promising. There are various watching systems as classified in [15] . For watching services for the elderly, there are services that do not use IT technology, for example, post officers or electric and water inspectors visit homes at regular intervals to check the resident's health status [16]. However, in order to respond quickly to emergency situation, it is preferable to use IT technology. The Ramrock system [17] detects loitering and fall of an elderly alone for automatic warning, but sendnig images of a surveillance camera may cause privacy problems. In consideration of privacy, there are some methods using sensors other than camera to watch more loosely. In the system of Zojirushi [18], the usage records of a pot is sent twice a day, but such information is too indirect to detect emergency state. The system that watches the elderly

URI: http://hdl.handle.net/10125/49915
ISBN: 978-0-9981331-1-9

by attaching sensors to furnitures such as a bed [19] has been studied, but it is inevitably a relatively large system. In the system of Philips [20] where a watched person wears a pendant type sensor for fall detection, there is a troublesome disadvantage that the sensor must be worn at all times. In the system of Fujitsu [21] which uses "sound analysis" of daily sound such as coughing and snoring to confirm the safety and estimates the health condition of elderly, the privacy is kept, but the accuracy of grasping the state of the elderly from the sound might become a problem.

From the above consideration, the method of recognizing the skeleton with non-attached sensor and directly grasping the human motion and posture is a superior method to watch people with privacy, and has a great advantage for other methods.

Furthermore, the recognized skeleton data has a very small size compared to camera images , so it is practical even if the data is sent over a narrow bandwidth network. This means a major advantage in terms of communication cost, especially when monitoring a large number of places, detacting a person's appearance and notifying it. This can be used for management of a place to be a shelter in case of a disaster.

This paper reports our work on the library software as and proposes some possible use cases of the Kinect V2 such as a remote monitor system for elderly care with privacy as well as a monitor system for shelters at disaster. The next section introduces the design and implementation of our library software. Section 3 shows an example of skeleton recognition program to express the granularity of our library's API. Section 4 shows the configuration of a system that distributes skeleton and face recognition results to a network as a server. Section 5 proposes two applications of the library for the remote monitor system. Section 6 gives some conclusions.

2. The design of the library for Kinect V2

In order to make best possible prospect of programming development using the function of Kinect V2, we have newly developed C++ class library *NtKinect*.

The design policies for our library, NtKinect, are as follows:

- we use the "Kinect for Windows SDK 2.0" as the development base
- we take the object-oriented approach to manage devices
- we make use of the Standard Template Library (hereinafter called "STL") to manage the collection-class data
- we use the Open Source Computer Vision (OpenCV) data structure
- we use mutex and our library supports multi-threading
- it should be easy to make Dynamic Link Library (DLL)

In our library, all the sensors of a Kinect V2 are managed with one object. Data from sensors will be acquired by one method invocation and saved in the object. Each sensor is initialized at the first method call to get the data.

The number of data for body information detected by Kinect V2 changes from moment to moment, so that the data would be managed best as "Collection" data by the STL.

The STL is a library for C++ programming language and provides some facilities including *containers*. We deal with data from Kinect V2 sensors with Collection, such as *pair*, *vector* and *array* of STL's sequence containers, not with simple array. In the Collection, the size may be changed at any time and each element is assigned on the heap memory.

OpenCV is useful for image processing and image recognition [22]. It becomes even easier to use in C++ from version 2 or later. If one managed the data obtained from Kinect V2 with OpenCV data structure, the data could be passed directly to the OpenCV functions.

The OpenCV data structure, *cv::Mat*, holds image data of Kinect V2, such as color, depth (distance), bodyindex, and infrared. Since each sensor uses a different coordinate system, we need to provide the functions for coordinate transformation (See Fig. 1 and Table. 1).

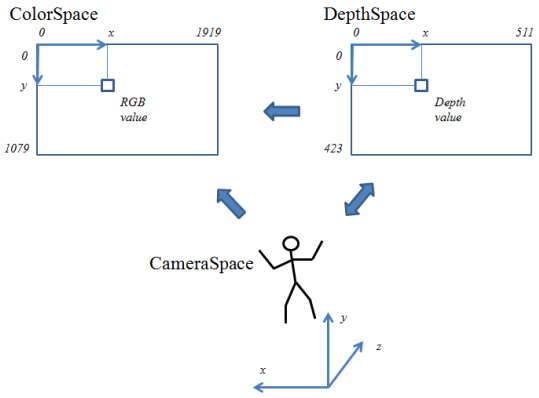

Figure 1. Data Representation of cv::Mat and Coordinates Transformation

Parallel processing may well be required for the following operations:

Table 1. Data and Coordinates System of Kinect V2

Coordinates System	Types of Data
ColorSpace	Color Image
DepthSpace	Depth Image, BodyIndex Image， Infrared Image
CameraSpace	Skeleton Information

- data acquisition from sensor devices
- the processing of acquired data

We provide a method that performs mutual exclusion control using "mutex" as well as thread-safe methods, so that the above two operations could be processed in a parallel way.

Due to our design policies, the NtKinect Object and the acquired data from sensors are allocated on the heap memory. Accordingly it becomes possible to make the developed program into DLLs so that it would be easier to use those programs in a different language and development environment.

We implemented a library based on the design policy described above. Method API is shown in Table 2 and member variables are shown in Table 3. In those tables, arguments of methods and the types of some member variables are omitted.

Presumably the program repeats the following operation: data is stored in a variable in the object by one method call. Data would be accessed through variables from outside the object.

The current version of our library supports Kinect V2 skeleton recognition, face recognition, detailed face recognition (HDFace), gesture recognition, acquisition of sound and its direction, and speech recognition.

3. Development of a program with NtKinect

In this section, we show how one can use our library to write a program in C++ for skeleton recognition. A sample program is shown in Fig. 2, and its execution result is Fig. 3. The program of Fig. 2 recognizes the skeleton and draws the joint position at the corresponding location on the camera image. For skeleton recognition, 25 joints positions per person are obtained in 3-dimensional coordinates, and their positions are superimposed and displayed on the 2-dimensional color image take by a camera. As can be seen from Fig. 2, the programs using Kinect V2 can be described more concisely than the traditional ones [23]. Therefore, the ovarall system is easy to understand for everyone, and effective reuse becomes possible. Of course, it can also be used to build a watching system.

The operation of the program is as follows.

- Call the *kinect.setRGB()* method to get the color 2-dimensional image taken by a camera.
- Call the *kinect.setSkeleton()* method to recognize skeleton.
- Since the positions of the joints are expressed in the CameraSpace coordinate system, they are converted to the ColorSpace coordinate system and then rectangles are drawn on the color image.
- Joint data has meaning only when its *TrackingState* member variable has the value other than *TrackingState_NotTracked*.
- Display the camera image in which the joints are drawn in red.

4. Skeleton Data Server

There is a need for a system that automatically recognizes the existence and state of people and communicates the result to a remote place. For this needs, skeleton recognition is considered to be a very useful means. The reasons are as follows.

- A system that watches people remotely is necessary, but since it is human beings being watched, privacy problems tend to occur. Abstracting human image data into skeleton information will protect their privacy.
- It is a laborious task in manpower to pick out only the images of people from many images. If it is possible to automatically select only the image of people and send it together with the information of the time and place, it is possible to save labor for monitoring.

We set up a system that sends out skeleton recognition result over the Internet using the *NtKinect* library. This server system is composed of a Note PC (Windows 10) with the Internet connection and Kinect V2 connected . The model of the system is shown in Fig. 4.

The server system keeps getting human skeleton information and voice within the distance between 50 cm and 450 cm from the front of Kinect V2. The acquired data will be sent in response to the

Table 2. Methods of NtKinect.

Category	Type	Method Name	Details (variable to be changed)
RGB	void	setRGB	set Color Image to *rgbImage*
Depth	void	setDepth	set Depth Image to *depthImage*
BodyIndex	void	setBodyIndex	get Body Index Image to *bodyIndexImage*
Infrared	void	setInfrared	set Infrared Image to *infraredImage*
Skelton	void	setSkeleton	recognize Skeleton and set to *skeleton, skeletonId, skeletonTrackingId*
Palm	pair<int,int>	handState	Recognize Hand State
Face	void	setFace	Recognize Face and set to *facePoint, faceRect, faceDirection, faceProperty*
Face	void	setHDFace	Recognize HDFace and set to *hdfaceVertices, hdfaceTrackingId*, *hdfaceStatus*
Audio	void	setAudio	set Audio data to *beamAngle, beamAngleConfidence, audioTrackingId*
Audio	void	drawAudioDirection	draw audio beam direction
Audio	bool	isOpendAudio	flag for recording or not-recording
Audio	void	opendAudio	start recording
Audio	void	closeAudio	end recording
Speech	void	setSpeechLang	Set language and words for recognition
Speech	void	startSpeech	Start speech recognition
Speech	void	stopSpeech	Stop speech recognition
Speech	bool	setSpeech	Recognize Speech and set to *recognizedSpeech, speechTag|Item|Confidence*
Gesture	void	setGestureFile	register Gesture difinition file
Gesture	void	setGesture	Recognize Gesture and set to *discreteGesture, discreteGestureTrackingId*, *continuousGesture, continuousGestureTrackingId*

Table 3. Member variables of NtKinect.

Category	Type	Name	Details
RGB	cv::Mat	rgbImage	Color Image
Depth	cv::Mat	depthImage	Depth Image
BodyIndex	cv::Mat	bodyIndexImage	BodyIndex Image
Infrared	cv::Mat	infraredImage	Infrared Image
Skeleton	vector<vector<Joint>>	skeleton	Skeleton Information
Skeleton	vector<int>	skeletonId	Skeleton's BodyIndex
Skeleton	vector<INT64>	skeletonTrackingId	Skeleton's TrackingID
Face	vector<vector<PointF>>	facePoint	Position of Face Parts
Face	vector<cv::Rect>	faceRect	Boundingbox of Face
Face	vector<cv::Vec3f>	faceDirection	Face Direction
Face	vector<vector<DetectionResult>>	faceProperty	Face Properties
Face	vector<UINT64>	faceTrackingID	Face's skeletonTrackingID
HDFace	vector<vector<CameraSpacePoint>>	hdfaceVertices	Position of HDFace Parts
HDFace	vector<UINT64>	hdfaceTrackingId	HDFace's skeletonTrackingId
HDFace	vector<pair<int,int>>	hdfaceStatus	HDFace properties
Audio	float	beamAngle	Audio Beam's Direction
Audio	float	beamAngleConfidence	beamangle Confidence
Audio	UINT64	audioTrackingId	Speaker's skeletonTrackingId
Speech	bool	recognizedSpeech	Flag for Speech Recognition
Speech	wstring	speechTag	Tag of Recognized Word
Speech	wstring	speechItem	Item of Recognized Word
Speech	float	speechConfidence	Confidence
Speech	float	confidenceThreshold	Threshold
Gesture	vector<pair<...,float>>	discreteGesture	Discrete Gesture and it's confidence
Gesture	vector<UINT64>	discreteGestureTrackingId	Discrete Gesture's skeletonTrackingId
Gesture	vector<pair<...,float>>	continuousGesture	Continuous Gesture and it's progress
Gesture	vector<UINT64>	continuousGestureTrackingId	Continuous Gesture's skeletonTrackingId

```
#include <iostream>
#include <sstream>
#include "NtKinect.h"
using namespace std;
void doJob() {
  NtKinect kinect;
  while (1) {
    kinect.setRGB();
    kinect.setSkeleton();
    for (auto person: kinect.skeleton) {
      for (auto joint: person) {
        if (joint.TrackingState == TrackingState_NotTracked) continue;
        ColorSpacePoint cp;
        kinect.coordinateMapper->MapCameraPointToColorSpace(joint.Position,&cp);
        cv::rectangle(kinect.rgbImage,cv::Rect((int)cp.X-5,(int)cp.Y-5,10,10),cv::Scalar(0,0,255),2);
      }
    }
    cv::imshow("rgb", kinect.rgbImage);
    auto key = cv::waitKey(1);
    if (key == 'q') break;
  }
  cv::destroyAllWindows();
}
int main(int argc, char** argv) {
  try {
    doJob();
  } catch (exception &ex) {
    cout << ex.what() << endl;
    string s;
    cin >> s;
  }
  return 0;
}
```

Figure 2. Recognition of Skeleton with NtKinect (C++). Kinect's Programs can be described more compactly than the traditional ones.

Figure 3. Execution Result of the Program in Fig. 2

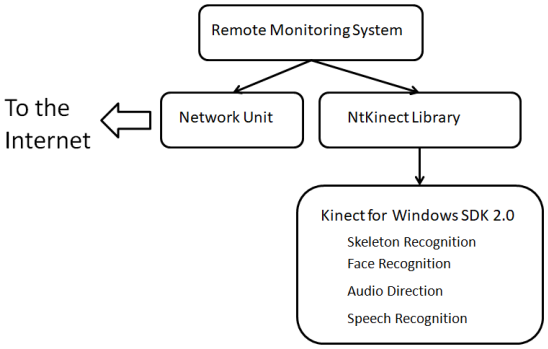

Figure 4. Remote Monitoring System

requests from the client. The data includes as skeleton information as well as the information such as which skeleton has sounded or made a specific movement most recently. The client is authenticated with its IP Address or a password.

5. Proposed Applications of Kinect V2

In this section, we present two possible applications which make use of Kinect V2 so that our library would be useful. One is a remote monitor system for elderly care with privacy and the other is a monitor system for shelters at disaster to keep track automatically of the people. We describe our proposal in the following subsections.

5.1. A remote monitor system for elderly care

We have been facing a demographic problem of increasing population of elderly in Japan. A remote monitoring for their safety and security has been implemented to monitor those who live alone. As mentioned in Section 1, several attempts have been made with IT technology [17] [18] [19] [20] [21]. While these attempts are useful, there remain privacy issues and accurate grasp of people's state to be solved.

To find accidents and abnormalities by recognizing

the skeleton, we could monitor the elderly for their safety and security with care for their privacy. Indeed, monitoring with the posture and gesture by skeleton recognition together with speaking sound and audio direction, we could watch over the safty and security from remotely as well. The outline of the system is shown in Fig. 5.

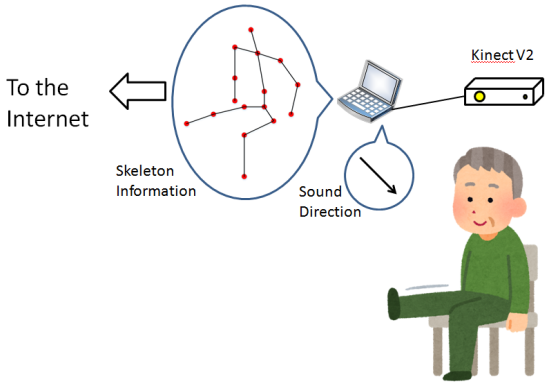

Figure 5. Watching the elderly with Privacy

5.2. Detecting the status of shelters at disaster

We have had various natural disasters almost every month in Japan[24]. One of them is the sudden volcano explosion with Mt. Ontake in Japan on September 27, 2014, causing a catastrophe in which 58 people were killed and 5 were still missing[25].

A lot of people evacuated to nearby mountain lodges in the event of the disaster, but the administrative side took time to grasp which evacuees are in which mountain lodges.

We suggested the needs for information processing at disaster by taking examples of the Great East Japan Earthquake at March 11, 2011 [26].

At large-scale disasters, it is highly crucial to grasp the information urgently on who or how many people are evacuated in what shelters. If such information is collected manually, there would be a delay in rescue and emergency response. If it is possible to detect the possible existence of people automatically, there will be a high possibility that the rescue and response will be improved dramatically [27].

Moreover, public places such as community centers and halls are likely to be utilized as evacuation centers at disaster. If we monitor the people around in those buildings not only at disaster but also at normal situations, there might be a privacy issue coming up that people feel that they are monitored by identified who they are. On the other hand, with our system with "skeleton information," we can detect the existence of people without collecting personal information so that privacy may well be preserved presumably.

Another issue is the amount of data to be transferred over the network for communication with our proposed monitoring method with Kinect V2 compared to the use of traditional monitor with live streaming or images. Since one joint position is represented by three float numbers, when expressing skeleton information of a human with 25 joints, it can be represented by 4 bytes $\times$ 3 $\times$ 25 = 300 bytes. We presume it absolutely useful in case of disaster with limited bandwidth available for communications [28].

6. Conclusion

In this paper, we report the development a libray software, NtKinect, which makes it easier to use a Kinect V2 device and the possible applications to use such a device so that the library software would be used to a great extent.

The library has been released as an Open Source of MIT license since July 2016 and used widely for various applications such as in collaborative robotics and computer art systems. With our library, it is possible to automatically recognize the existence of human beings and his/her postures and gestures, so the system could be used for monitoring the elderly people as well as detecting victims for situation awareness at a disaster.

Considering the generalization of low-speed mobile communication for IoT in the near future, it seems that utilizing the result of skeleton recognition becomes increasingly important. With recent mobile communication systems for Internet of Things (IoT), it is possible to use a network which is inexpensive but slower. LPWAN (Low-Power Wide-Are Network) [29] is such a wireless telecommunication wide area network to provide long range communications at a low bit rate among connected objects. For example, SIGFOX [30] data rates is 100 bps, and LoRaWAN [31] data rates range from 0.3 kbps to 50 kbps. With the 5G mobile communication system (the fifth generation mobile networks), which is coming into use widely, since IoT communication becomes popular in many applications, it will be certainly necessary to communicate with as small traffic volume as possible. According to such trend, our proposed use of skeleton information to detect the existence of people could be useful.

References

[1] Sementille,A.C., Loureno, L.E., Brega,J.R.F. and Rodello, I.: *A motion capture system using passive markers*, Proc. of the 2004 ACM SIGGRAPH international conference on

Virtual Reality continuum and its applications in industry (VRCAI '04), pp.440-447, 2004

[2] Barber, A., Cosker, D., James , O.,Waine, T. and Patel, R.: *Camera tracking in visual effects an industry perspective of structure from motion*, ACM Proc. of the 2016 Symposium on Digital Production (DigiPro '16) pp.45-54, 2016.

[3] Schrder, M., Maycock, J. and Botsch, M.: *Reduced marker layouts for optical motion capture of hands*, Proc of the 8th ACM SIGGRAPH Conference on Motion in Games (MIG '15), pp.7-16, 2015.

[4] Microsoft: *Meet Kinect for Windows*, https://developer.microsoft.com/en-us/windows/kinect (2017/06/08 access).

[5] Woolford, K.: *Defining accuracy in the use of Kinect v2 for exercise monitoring*, ACM, Proc. of the 2nd International Workshop on Movement and Computing (MOCO '15), pP.112-119, 2015. 2015.

[6] Avola, D, Cinque, L, Levialdi, S, Petracca, A, Placidi, G, Spezialetti, M: *Markerless Hand Gesture Interface Based on LEAP Motion Controller* Proc. of the 20th International Conference on Distributed Multimedia Systems: Research papers on distributed multimedia systems, distance education technologies and visual languages and computing, pp.27-29, 2014.

[7] Microsoft: *Kinect for Windows SDK C++ Reference*, https://msdn.microsoft.com/en-us/library/dn791993.aspx (2017/06/08 access).

[8] Nitta, Y.: *NtKinect - C++ Class Library for Kinect V2*, the 172-th conference SIG Human-Computer Interaction, IPSJ, 2017.

[9] Nitta, Y.: *NtKinect - Kinect V2 C++ Programming with OpenCV on Windows10*, http://nw.tsuda.ac.jp/lec/kinect2/index-en.html (2017/06/08 access).

[10] Nitta, Y.: *NtKinect - How to recognize human skeleton with Kinect V2*, http://nw.tsuda.ac.jp/lec/kinect2/KinectV2_skeleton/index-en.html (2017/06/08 access).

[11] Nitta, Y.: *NtKinect - How to recognize human face with Kinect V2 in ColorSpace coordinate system*, http://nw.tsuda.ac.jp/lec/kinect2/KinectV2_face/index-en.html (2017/06/08 access).

[12] Tsuchiya, M., Itoh, T., Nitta, Y.: *Interactive Light Painting System Using Human Recognition*, NICOGRAPH 2016, P-4, The Society for Art and Science, 2016.

[13] Tsuchiya, M. Itoh, T. Nitta, Y: *An Interactive System for Light-Art-Like Representation of Human Silhouettes*, WISS 2016, P-213, JSSST, 2016.

[14] Tsuchiya, M., Itoh, T., Nitta, Y.: *An Interactive System for Light-Art-Like Representation of Human Silhouettes*, Interaction 2017, IPSJ, 2017.

[15] Tokunaga, S. SAIKI, S. Matsumoto, S. Nakamura, M: *A Classification Method of Remote Monitoring Service for Elderly Person*, Vol. 113, No.469, p.169-174, 2014.

[16] Japan Post: *Watching Service of Post Office*, http://www.post.japanpost.jp/life/mimamori/index.html (2017/08/29 access) *(in Japanese)*.

[17] Ramrock: *Care Support System "Ramrock System"*, http://www.ramrock.co.jp/ (2017/08/29 access) *(in Japanese)*.

[18] Zojirushi: *MIMAMORI Hot Line*, http://www.mimamori.net/ (2017/08/29 access) *(in Japanese)*.

[19] Bradford, D. Freyne, J. Karunanithi, M.: *Sensors on My Bed: The Ups and Downs of In-Home Monitoring*, vol.7910, Lecture Notes in Computer ScienceSpringer Berlin Heidelberg 2013.

[20] Philips: *Medical Alert Service*, http://www.lifeline.philips.com/ (2017/08/29 access).

[21] Fujitsu : *"The deciding factor is IoT and sound", Elderly monitoring service to the next step* http://www.fujitsu.com/jp/innovation/digital/life/dl-contents/2017/topics-05/ (2017/06/08 access) *(in Japanese)*.

[22] OpenCV: *OpenCV library* http://opencv.org/ (2017/06/08 access).

[23] Microsft: *Kinect for Windows SDK 2.0, Programming Guide, Body tracking*, https://msdn.microsoft.com/en-us/library/dn799273.aspx (2017/09/03 access).

[24] Cabinet Office, Government of Japan: *Disaster Management in Japan*, http://www.bousai.go.jp/1info/pdf/saigaipanf_e.pdf (2017/09/02 access).

[25] Wikipedia: *2014 Mount Ontake eruption* https://en.wikipedia.org/wiki/2014_Mount_Ontake_eruption (2017/09/02 access).

[26] Murayama, Y., Sasaki, J. and Nishioka, D.: *Experiences in Emergency Response at the Great East Japan Earthquake and Tsunami*, HICSS-49, P.146-151, 2016.

[27] Van de Walle, B., Turoff, M. and Hiltz, S.R. eds: *Information systems for emergency management*, M.E. Sharpe, 2009.

[28] Saito, Y., Fujihara, Y. and Murayama, Y.: *A Study of Reconstruction Watcher in Disaster Area*, Proc. ACM CHI2012, pp.811-814, 2012.

[29] Wikipedia: *LPWAN*, https://en.wikipedia.org/wiki/LPWAN (2017/09/02 access).

[30] SIGFOX : *Radio Technology Keypoints*, https://www.sigfox.com/en/sigfox-iot-radio-technology (2017/06/08 access).

[31] LoRa Alliance: *LoRa Alliance Technology*, https://www.lora-alliance.org/what-is-lora (2017/09/02 access).