Title	Experiential information presentation about objects using AR-VR system
Sub Title	
Author	高, 翔一(Gao, Xiangyi) 小木, 哲朗(Ogi, Tetsurō)
Publisher	慶應義塾大学大学院システムデザイン・ マネジメント研究科
Publication year	2019
Jtitle	
JaLC DOI	
Abstract	
Notes	修士学位論文. 2019年度システムエンジニアリング学 第291号
Genre	Thesis or Dissertation
URL	https://koara.lib.keio.ac.jp/xoonips/modules/xoonips/detail.php?koara_id=KO40002001-00002019-0007

Master's Dissertation 2019

Experiential Information Presentation About Objects Using AR-VR System

Xiangyi Gao
（Student ID Number：81734541）

Supervisor Prof. Tetsuro Ogi

September 2019

Graduate School of System Design and Management,
Keio University
Major in System Design and Management

SUMMARY OF MASTER'S DISSERTATION

Student Identification Number	81734541	Name	Xiangyi Gao

Title

Experiential Information Presentation About Objects Using AR-VR System

Abstract

With AR technology, digital information can be overlaid onto the physical world, and information can be placed near objects to emphasized which object the information belongs to. Although AR strengthen the relationship between objects and information, the physical space to display information is limited. An effective way to deliver information through AR system should be proposed. Virtual reality (VR) is an effective method to deliver information. In this research, we propose an object detection-based AR-VR system, which delivers VR information after user selecting objects in AR. With this system, user can not only experience a VR environment for information, but also understanding which object the information is describing. As an implementation, tourism promotion purpose system was developed and exhibited. Users can select local specialties through object detection AR and experience the VR worlds which describe those objects. The evaluation indicates that this system is effective as an information system.

Key Word(5 words)

Deep learning, Object detection, AR, VR website, User interface

Table of Contents

List of Figures

CHAPTER 1: INTRODUCTION

1.1 Background

1.1.1 Overview of Augmented Reality

Augmented reality (AR) is a revolutionary information delivering technology which overlays virtual information on the physical world. AR delivers information differently from the traditional way. An AR system have the following properties [1]:

- combines real and virtual objects in a real environment;
- runs interactively, and in real time;
- registers (aligns) real and virtual objects with each other.

Instead of displaying digital contents independently on a device screen, AR presents digital information in the real word, often near to the object which strongly related to the information. For example, when a person meets a fruit he or she never seen before, instead of searching for it on the internet using a mobile device, he or she can directly see the introduction about the fruit on top of it as an AR information. Other than the convenience, AR technology provides strong connection between object and information which helps users to understand the information intuitively. Even though displaying digital information on top of real object sounds like science fiction concept, the fact is that AR has the potential to play a big role in the next generation information system in the near future [2]. AR technology have been applied various industries.

AR technology is utilized to create learning experiences [3]. In a research of augmented reality enhanced artifacts, the system can recognize parts of the Minkisi artifact and augmented with

information, making the object observable with information [4]. In a research of augmented remote laboratory, the system offers a greater sense of realism, the same physical laboratory settings can be used for different experiments, and easier experiment verification and generation since [5].

In manufacturing industries, AR technology can also be applied. AR can be used as a comparison tool in the automotive industry to compare actual parts with correct construction data. With the help of AR, real parts can be check whether they are manufactured with appropriate design [6]. In aerospace industries, AR is used in servicing projects, such as space station filter change and engine maintenance [7].

1.1.2 Object Detection Based AR system

One of the major types of AR system is object detection-based AR system. This AR system is triggered by object detection mechanisms and presents information in the form of AR to introduce the detected objects. The system consists of two main functions which are object detection and information presentation. The common mechanisms to detect object are marker-based, location-based, sensor-based, and deep-learning-based. The common formats of information presented are texts, images, 3D objects. Figure 1 shows a marker-based AR system in which the actual panel is registered as a marker, and when it is detected, an image related to the panel will be displayed in AR. Figure 2 shows a location-based AR system in which recommended place to visit near current location will be displayed in AR [8]. Figure 3 shows a deep learning-based AR system in which introduction texts about detected animals will be displayed [9].

Figure 1. AR image triggered by marker

Figure 2. Place recommendation triggered by current location

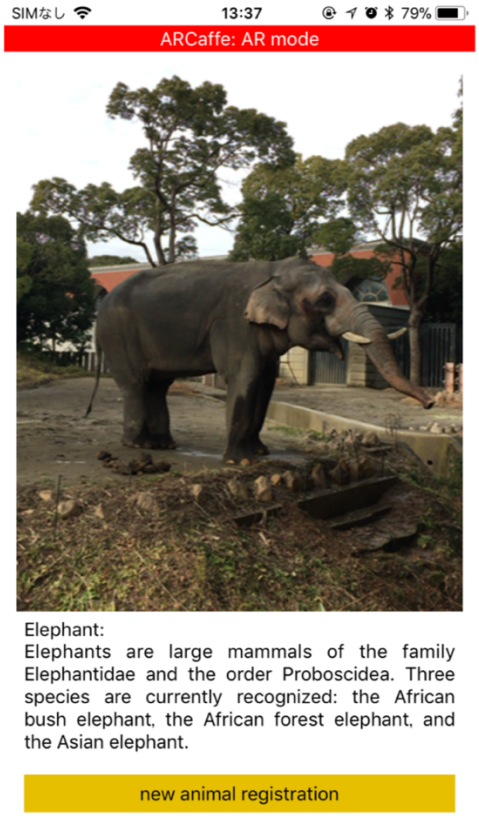

Figure 3. Text information triggered by detected animals

1.2 Problem Statement

Even though object detection-based AR system strengthens the relationship between information and objects by displaying digital information close to physical objects, the space for information presentation in AR system becomes limited. In object detection-based AR system, information has to be placed near the detected object to lead users to intuitively understand that the piece of information is describing the object. Thus, information must be concise enough to maintain the connection to objects. Concise information often leads to missing information which wanted to be delivered. The problem statement of this research is how to present comprehensive information while maintaining the connection between information and described objects.

CHAPTER 2: PROPOSED METHOD

We propose using virtual reality (VR) to deliver comprehensive information about objects. VR is a simulated digital environment created by computer where users can immerse into. VR usually provide users a 360-degree view and users can experience and explore the virtual environment. A large amount of information can be integrated into a VR scene since VR provides a completely virtual environment. Another advantage of using VR is that the experience in VR can deliver information which is hard to describe in words. For example, the best way to tell a person how beautiful a place is, is to take that person to the place and let him or her to have a look.

In our method, we combine AR and VR technology to achieve experiential information delivering about objects, in which users can comprehensively understand the information and also recognize the subject of the information.

2.1 Related Work

We propose using VR to deliver information about objects, thus the effectiveness of VR as information needs to be investigated. Researches about this topic have been conducted, and the results indicate that VR is an ideal tool for information delivering.

In a comparative study of VR education, an A/B test was held where two groups of participants attend a lecture about Stonehenge [10]. The first group was lectured through university lecture style PowerPoint slides, and the second group learned through VR content. The overall result of the time-constrained assessment which consists of questions covering information mentioned in the lecture indicates that participants who learned through VR performed better when answering those questions. Furthermore, participants claimed that learning in VR was a more positive experience compared to learning with text material.

In the research on art appreciation learning via VR, authors examined the usefulness of VR in art appreciation [11]. They conducted a comparative experiment of teaching using VR and existing presentation media, which are a textbook, a poster, a replica, and a tablet. The experiment results indicate that VR is the most preferred media for art presentation.

CHAPTER 3: SYSTEM CONSTRUCTION

3.1 System Concept

The system detects objects, and present what are recognized in AR. Then when users decide which object information they want to see, the system presents information about that object in VR. Users using this system will see objects in front of them are recognized and select one of the objects they are interested in. Then they can immerse into the VR environment which describe that object and receive information. For example, when a user interested in an object in front of him or her which is a bottle of water, they can choose that object and immerse into VR mountain spring which is the source of that bottle of water. In this case, user can actually see the beautiful spring where the water comes from.

The function flow is shown in figure 4. The system first detects objects and then present detected object information in AR. After user interaction, the system present VR information.

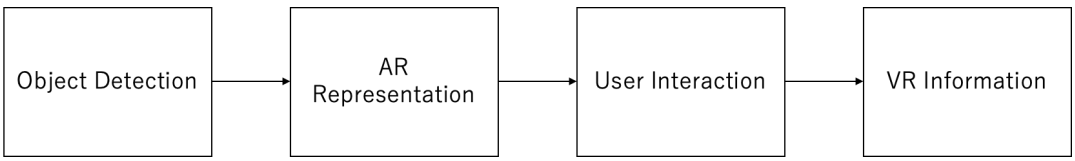

Figure 4. System function flow

3.2 Object Detection Development

3.2.1 Mechanism Selection

One of the major functions of this system is object detection. As we discussed in CHAPTER 1, there are several object detection mechanisms used for AR systems. Before developing object detection function, we investigated which mechanism is suitable for our system.

In **marker-based** object detection [12], [13], [14], a computer recognizes an object by its image pattern. In this mechanism, a specific 2D pattern is recognized. Since the objects which we want computer to recognize are real 3D objects, and usually have different appearances, a pattern match method cannot function as we expect. A walkaround way to utilize pattern matching is to prepare a separate piece of 2D image which has simple pattern, usually a QR code like image, and place it near to the object. When objects have their own 2D marker, it become possible for computer to identify each object by recognizing markers near each object. The disadvantage of using this mechanism is that we need to prepare marker for each object and place them. This becomes unrealistic when the number of recognizable objects increase.

In **location-based** object detection [15], [16], [17], a computer recognizes an object by its location. In this mechanism, objects positions are defined by their geo-location which are registered beforehand. When system with GPS moves close to registered locations, corresponding AR information will show up. This mechanism is usually used to display information for unmovable objects such as buildings and sightseeing spots. The disadvantages of using location-based method is that the system will recognize the location but no the actual object, when the object is moved from designed location, the AR information still appears at the location instead of following the object.

In **sensor-based** object detection [18], [19], [20], a computer recognizes an object by detecting

the sensor signal placed close to the object. Usually radio frequency identification (RFID) tag is used for item identification. Multiple RFID tags can be detected accurate and precisely to identify objects. However, similar to marker-based method, to use this mechanism, we need to prepare tags for each object and place them. This becomes unrealistic when the number of detectable objects increase.

In **deep learning-based** object detection [9], [21], [22], a computer recognized an object by running its image through trained machine learning model. With this mechanism, a computer detects objects without extra information set with them, only with object appearance. Furthermore, objects can be recognized from different angle, and objects which are never seen by computer vision before, as long as that category of object is trained with datasets. Even though the training process requires large datasets, the well-trained model has high accuracy on recognizing objects. Since computer will recognize objects from its vision, when multiple objects exist in the view, the precise position of objects can be presented.

Comparing object detection mechanisms, we decide to implement our system with deep learning-based object detection. The marker-based and sensor-based mechanisms both require extra components to link with objects physically. Location-based mechanism cannot detect moving objects, which will largely limit our target objects. On the other hand, deep learning-based mechanism only needs objects appearance for detection, which can be applied to detect various objects without too much limitation.

3.2.2 Deep learning-based object detection algorithm

3.2.2.1 Deep learning introduction

Deep learning is a class of machine learning, which is based on artificial neural networks [23], [24]. Deep learning has been applied to different field such as image recognition, natural language processing, speech recognition, and etcetera with its architectures such as deep neural networks,

convolutional neural networks, recurrent neural networks.

The basic understanding of deep learning is that computers compute parameters in defined neural networks with provided datasets, following defined mathematical optimization methods.

Artificial neural networks are computing structures which are inspired by biological neural networks. For example, Figure 5 is a neural network which has 3 inputs, and 2 outputs, with 1 hidden layer in the middle.

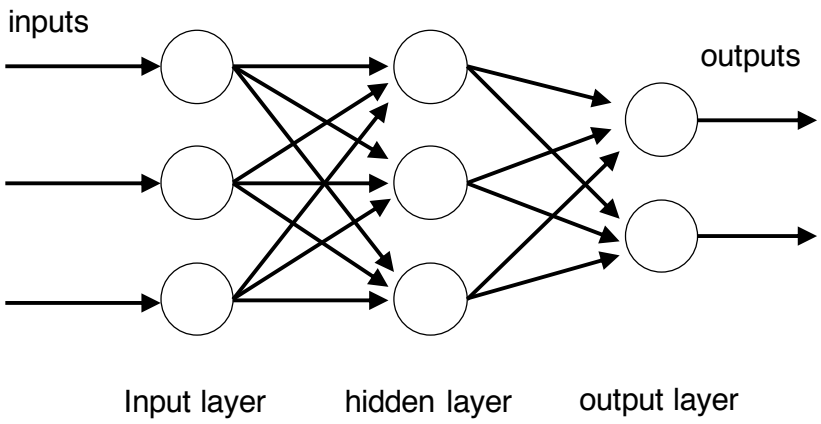

Figure 5. Example neural network

The nodes presented in hidden layers are artificial neurons which are mathematical functions as shown in Figure 6.

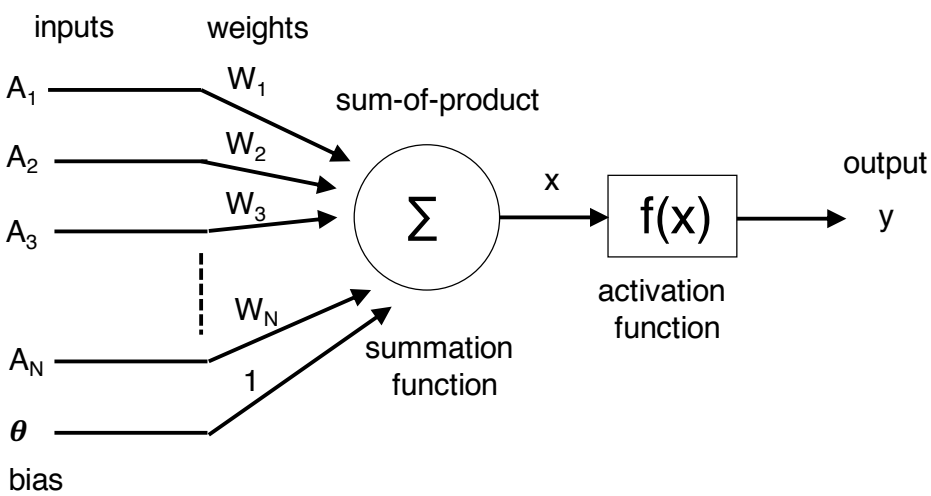

Figure 6. Artificial neuron

For each neuron, inputs are multiplied by weights and summing up with bias. Then the sum-of-product is calculated through activation function, and the results becomes input to next neurons. Since the fact that neural networks consist of neurons which are mathematical functions, neural networks are large mathematical functions. When the number of inputs and hidden layers becomes large, a neural network becomes "deep", in other words, a giant mathematical function. With given inputs and mathematical functions, computers can calculate the outputs. Same process happens in deep learning-based image recognition. When an image is given, pixels of the image become inputs, and computer calculates through given mathematical function to give outputs which indicate, for example, whether the image is a cat or not cat. A trained neural network is the mathematical function in this process.

Only defining the structure of a neural network is not enough for computation, because the parameter values are not defined yet. Parameters in neural network such as weights and bias, are all called weights. Computing values of weights in neural network is the training process of deep learning. To compute what values should parameters in a mathematical function be, computers need to know example inputs and outputs which are called datasets, and also the optimization procedure which is called loss function. The loss function defines how the neural network updates parameters to reduce the error in prediction.

Given a neural network structure, datasets, procedures to compute weights in neural network, computers can produce a neural network with weights which is trained by the datasets. Thus, computers can recognize images by calculating through neural networks.

Since in our research, computer vision is used for object detection, an important class of neural network is needed to be introduced which is Convolutional neural network (CNN). CNN usually consists of convolution layers and pooling layers.

A convolution layer is a layer which do convolution operation [25]. A convolution is a linear

operation which multiply inputs with a set of weights, similar to regular neural network introduced above. The multiplication is performed between inputs and two-dimensional arrays which is called filters or kernels. The values in a filter is considered to be weights. An example of what filters do in a convolution operation is shown in figures below.

$$\begin{pmatrix} 0 & -0.5 & 0 \\ -0.5 & 3 & -0.5 \\ 0 & -0.5 & 0 \end{pmatrix}$$

Figure 7. Example filter which sharpen an input image

Figure 8. Example input and output a sharpen filter

As shown in figures above, a filter can output an image enhancing the features in original image. In above example, edges are sharpened through filtering. A convolution layer often consists of multiple filters, so inputting one image, a convolution layer can produce multiple images which are filtered with different weights.

A pooling layer a layer which down sample inputs. An example is shown as figure below, which is a max pooling process, which down sample input image by filtering with 2x2 window and

taking the maximum values as the outputs.

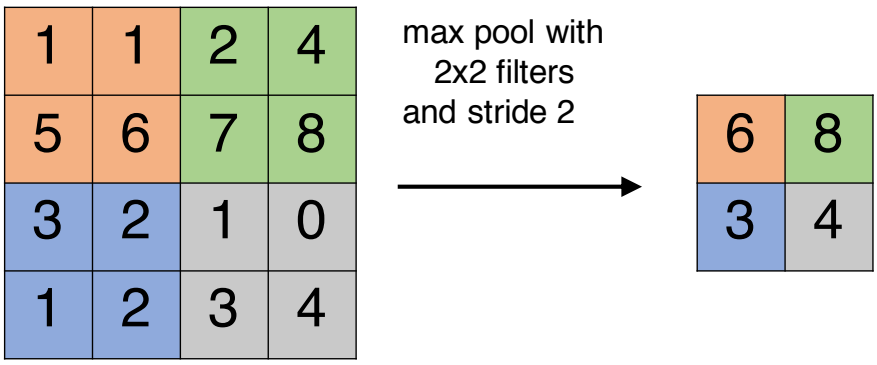

Figure 9. Example pooling

A CNN is usually combination of convolution layers and pooling layers. Convolution layers extract features from an inputs image, and pooling layers down samples outputs from convolution layers. The training process is similar to regular neural network we discussed above. With dataset and optimization process, parameters in filters are computed.

3.2.2.2 Algorithm Selection

CNN is a commonly used image recognition algorithm today. However, image recognition and object detection are close but different topic. In image recognition, computer tells what kind of object the whole image is representing. On the other hand, in object detection, computer tells what objects are in an image, and their position and size. Current object detection algorithms also utilize CNN. Examples of image recognition and object detection are shown below.

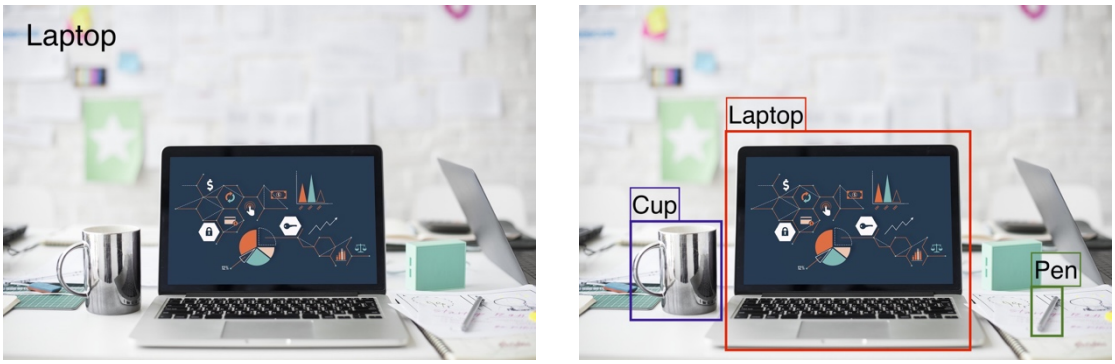

Figure 10. Image recognition and object recognition

To implement our research concept, object detection algorithm is required. There are several state-of-the-art object detection algorithms.

In **R-CNN** (Regions with CNN features), an input image is proposed into regions which are pieces of the image, and those pieces of image becomes inputs to CNN separately [26]. Thus, in this algorithm, computer does not consider the input image as a whole. Instead, computer runs through CNN for each region of the image, and the output indicates what objects are in which region. Methods to propose regions have been studied and improved as R-CNN, Fast R-CNN [27], Faster R-CNN [28].

In **YOLO** (You Only Look Once), an input image runs through a single CNN which predicts bounding boxes and the class of these boxes [29]. The object detection is realized by looking once at the input image, and this is where the name of this algorithm comes from. A square image is input into the YOLO model, and the outputs is also a square grid information, where each grid is "responsible" for a corresponding area of the input image. Each grid contains information about bounding boxes which centered in the grid area, each bounding box information contains, 4 coordinates defining the box area and 1 confidence value defining how confident an object is inside the box. Other than bounding box information, the classes probabilities of detected object are also in each grid. An image of a 7x7 grid output, which consists of 2 bounding boxes information and probabilities of 20 classes which can be detected, is shown in figure 11.

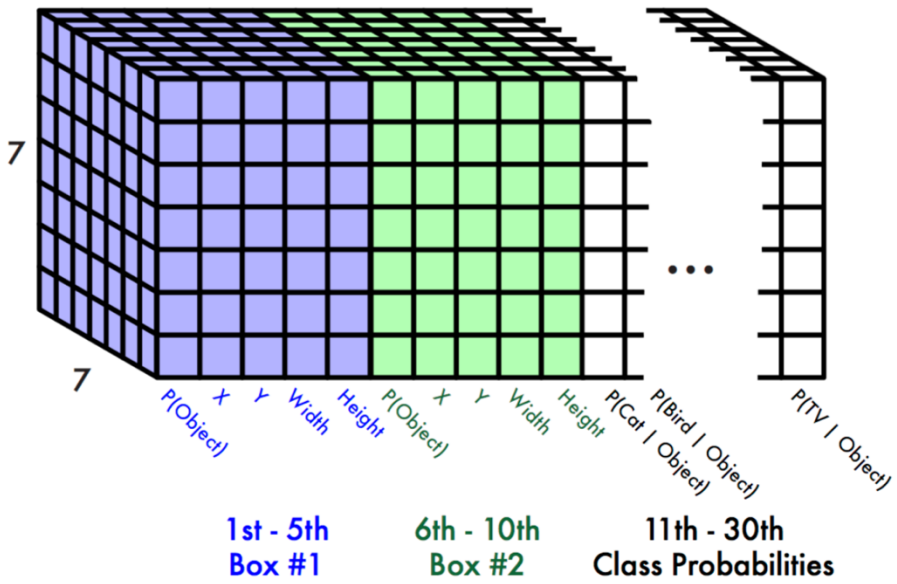

Figure 11. YOLO 7x7 output

The core idea of YOLO is packing all information about object position, size, and class into one single 3-dimentional array as output. Thus, to detect objects, CNN is only computed once with YOLO. On the other hand, algorithms like R-CNN require computing through CNN multiple times. As a result, the advantage of YOLO is fast, and the advantage of R-CNN is accurate.

YOLOv2 is an improved version of YOLO which have slightly different structure [30]. In first version of YOLO, the output array contains bounding boxes information and class probabilities for each grid. On the other hand, output of YOLOv2 contains class probabilities for each bounding box, as shown in figure 12.

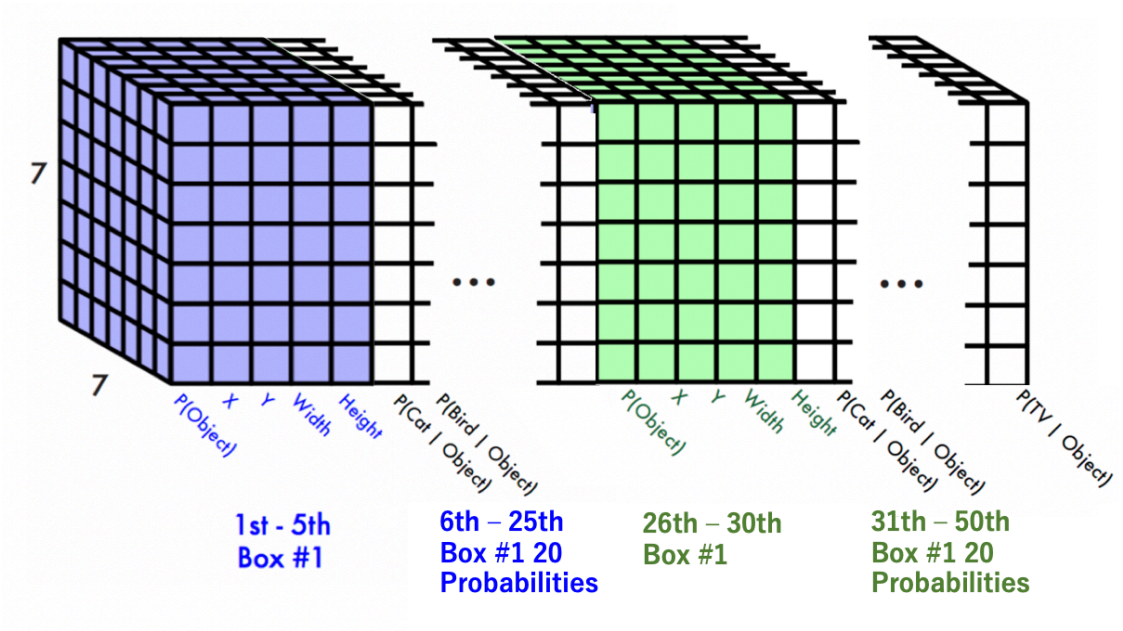

Figure 12. YOLOv2 7x7 output

Our research focuses on AR function which requires high refresh rate, so that system can detect objects in real time. YOLOv2 becomes our choice to implement. CNN structure in YOLOv2 algorithms can be varied, and there is a structure called Tiny YOLO which have a smaller number of layers in CNN compared with full version of YOLOv2. Tiny YOLO is less accurate compared to full YOLOv2 since it has simpler CNN structure, but it is much faster. Due to this fact, we decide to implement our object detection function using Tiny YOLO function.

3.2.2.3 YOLO Model Development

① Development Environment

CPU

- Architecture: x86_64
- CPU op-mode(s): 32-bit, 64-bit
- Byte Order: Little Endian
- CPU(s): 4
- On-line CPU(s) list: 0-3
- Thread(s) per core: 1
- Core(s) per socket: 4

- Socket(s): 1
- NUMA node(s): 1
- Vendor ID: GenuineIntel
- CPU family: 6
- Model: 79
- Stepping: 1
- CPU MHz: 1199.980
- BogoMIPS: 6983.80
- Virtualization: VT-x
- L1d cache: 32K
- L1i cache: 32K
- L2 cache: 256K
- L3 cache: 10240K
- NUMA node0 CPU(s): 0-3

GPU

- Product Name: TITAN X (Pascal)
- Product Brand: GeForce

Operation System

- DISTRIB_ID=Ubuntu
- DISTRIB_RELEASE=14.04
- DISTRIB_CODENAME=trusty
- DISTRIB_DESCRIPTION="Ubuntu 14.04.5 LTS"

Software Environment

- Python 3.6.3
- Tensorflow_gpu-1.3.0
- Cuda compilation tools, release 8.0, V8.0.61
- Nvidia driver version 418.39

② Dataset Preparation

To train a neural network, we need to prepare a dataset. For object detection, the dataset will be images and annotations which contain object classes and their coordinates for each image. Images of objects which we want the system to detect should be prepared. For example, if we want the system to detect "person", images of person or persons should be prepared. The more image data, the better object detection model will be trained. Variety of those data is also important. Images of different "persons" in different conditions and situations should be prepared for training. Otherwise if only images of person in red t-shirt are used for training, the model will only identify person with red t-shirt as "person".

For annotation, we need to make a file containing objects information for each image. A convenient tool **labelImg**[1] is ready for us to use. The user interface of labelImg is shown as figure 13.

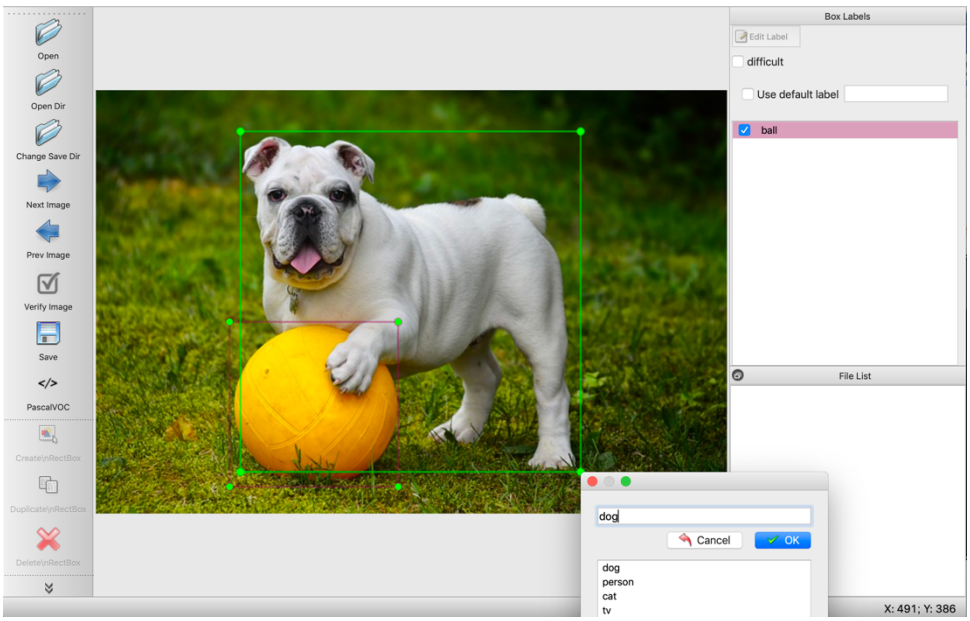

Figure 13. UI of labelImg

[1] https://github.com/tzutalin/labelImg.git

By drawing bounding boxes and inputting objects names, the annotations are recorded. These annotations will be saved as an XML file containing objects names, and their coordinates. In the example of figure 13, the image includes a dog and a ball. To annotate them, we need to drag bounding box on the dog and input “dog” and drag bounding box on the ball and input “ball”. The bounding coordinates of the dog and the ball will be saved as an XML file which is shown in figure 14. The file name indicates the corresponding image file which the XML belongs to, and it will help to pair the image and the XML file in training process. A bounding box can be described with two coordinates which are upper left point and lower right point of the box. In a XML file, the two coordinates are recorded for each object class as xmin, ymin, xmax, and ymax with object name.

```xml
<annotation>
    <folder>Downloads</folder>
    <filename>english-bulldog-562723_640.jpg</filename>
    <path>/Users/gao/Downloads/english-bulldog-562723_640.jpg</path>
    <source>
        <database>Unknown</database>
    </source>
    <size>
        <width>640</width>
        <height>429</height>
        <depth>3</depth>
    </size>
    <segmented>0</segmented>
    <object>
        <name>ball</name>
        <pose>Unspecified</pose>
        <truncated>0</truncated>
        <difficult>0</difficult>
        <bndbox>
            <xmin>135</xmin>
            <ymin>234</ymin>
            <xmax>306</xmax>
            <ymax>401</ymax>
        </bndbox>
    </object>
    <object>
        <name>dog</name>
        <pose>Unspecified</pose>
        <truncated>0</truncated>
        <difficult>0</difficult>
        <bndbox>
            <xmin>146</xmin>
            <ymin>42</ymin>
            <xmax>492</xmax>
            <ymax>389</ymax>
        </bndbox>
    </object>
</annotation>
```

Figure 14. Inside an XML file

③ Modeling

To train a model from our dataset, we utilize deep learning frameworks. The original YOLO algorithm is implemented in Darknet, an open source neural network framework written in C and CUDA. We want to implement our model to hardware other than PC, which we will discuss later, however, Darknet format model is not compatible with the format we want. A more commonly used framework should be used, and Tensorflow, an open source framework written in Python, C++ and CUDA, is our choice. Fortunately, the work to translate YOLO from Darknet formate to Tensorflow formate have already done in **Darkflow**[2]. With Darkflow, we can train our own YOLO model in Tensorflow format.

To train a YOLO model, the dataset will be place under corresponding directories. Since the number of object classes to be detected varies, modification on the configuration file which defines YOLO network structure is required. The detailed procedure to train a YOLO model is described in appendix.

The input size of YOLO model has width of 416 pixels and height of 416 pixels. The training data have various image sizes. Thus, during the training process, training data images will be resized into 416x416 pixels images and their corresponding object coordinates will also be resized according to the original image size and keeping the aspect ratio. While prediction process, an input image will be resized into 416x416 pixels, and the output coordinates of objects will be resized according to the original input size.

[2] https://github.com/thtrieu/darkflow.git

3.3 VR Contents Development

3.3.1 Mechanism Selection

Local VR applications can be developed with game engines such as Unity. Since these engines are powerful, VR applications with large varieties and good qualities can be developed. The created VR applications are designed to be processed locally, so they will be saved locally as part of our system and waiting to be processed. Often local VR applications have big file sizes and take a lot of data storage space to save them. Considering scalability of our system, if we develop our VR contents as local applications, as our detectable objects increase, the storage space taken by VR contents corresponding to each object class also increases. This implementation method is not preferred not only because it will require large space of storage, but also the editability of VR contents. The concept of this research is to deliver VR contents as information, and if we develop the VR contents locally, the information cannot be modified easily.

Developing remotely accessible VR applications which do not require to store before using the system is a solution. WebVR is a web technology to build VR application as web applications, and the created VR content can be accessed like regular websites via accessing URLs. Even though WebVR is not designed to develop complex and high-quality VR application as game engines do, the VR applications created with WebVR can be accessed through internet and does not require our system to store them beforehand. WebVR is a JavaScript API for creating immersive VR experiencein browsers, and it can work on mobile phones and VR headsets such as HTC VIVE, Oculus Rift and etc. Researches have been conducted utilizing WebVR technology which promises the usability of WebVR [31], [32], [33].

3.3.2 VR Websites Development

To create VR websites utilizing WebVR technology, we used A-Frame, a web framework to build

VR websites. A-Frame is a three.js framework based on top of HTML, and it can be developed from plain HTML file without having to install anything. We used Glitch, an online code editor that hosts and deploys for free, to develop our A-Frame websites. Users should be able to interact with VR contents. When users turn their head to different angles and look at different direction, the view should also change according to user's action. It requires the application to communicate with the device which is displaying VR contents, and A-Frame handles it. A-Frame retrieve sensor data from devices through brewers which support WebVR. With A-Frame, we can define and create various components to construct virtual worlds. Components can be not only pre-defined component such as boxes, spheres, and cylinders, but also imported 3D models. We can change their position, size, color as we want. Another important factor of A-Frame is that since it is a web framework, many web technologies can be applied to enhance the VR world. For example, we can link websites to components we created, and users can access to other websites when interact with components.

Even though A-Frame provides plenty functions to construct VR websites, in this research we will focus on making simple VR worlds which are constructed with 360- degree videos which will be played in loop.

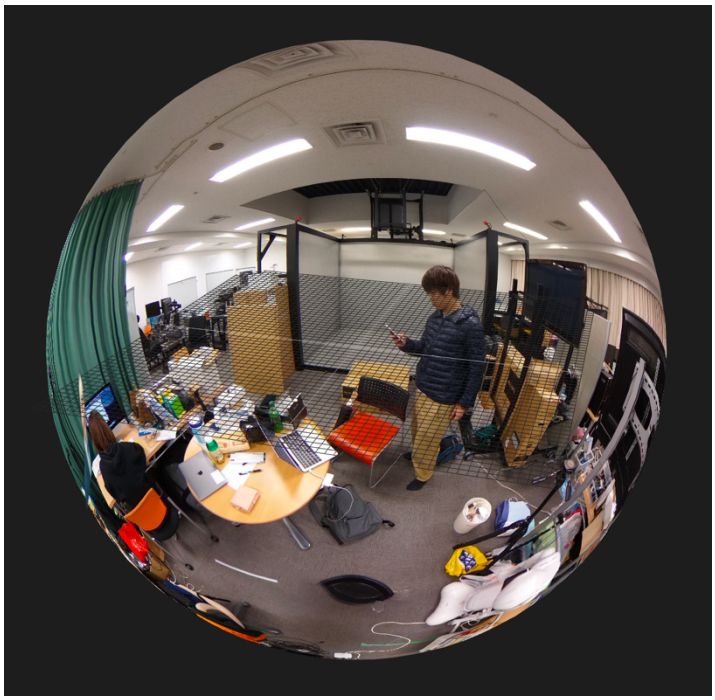

Figure 15. 360-degree video image

3.3.3 360-degree Video Making

The background environments are important to make people immerse into virtual worlds. Our purpose is to create VR contents for information about actual objects, and 360-degree video is utilized to describe objects. We used THETA V to take 360-degree videos to create VR websites.

Figure 16. THETA V

The specs[3] of THETA V regarding 360-degree video is shown below.

Video resolution/frame rate/bit rate	4K, H264: 3840×1920/29.97fps/56Mbps 2K, H264: 1920×960/29.97fps/16Mbps
Microphone	4ch
Internal memory/Number of photos that can be recorded, time	Video (time per recording): Max. 5 minutes/25 minutes Video (total recording time): Approx. 40 minutes (4K, H.264)/approx. 130 minutes (2K, H.264)
Object distance	Approx. 10cm - ∞ (from front of lens)
ISO sensitivity (standard output sensitivity)	Video: ISO64 - 6400
Compression method	Video: MP4 (Video: MPEG-4 AVC/H.264, H.265, Audio: AAC-LC (mono) + Linear PCM (4ch spatial audio))

To match the view of most users, 360-degree videos should be taken at height about the average height of an adult, and also perpendicular to ground. We used a camera stand for THETA to satisfy these requirements.

[3] https://theta360.com/en/about/theta/v.html

3.4 System Integration

3.4.1 Hardware Selection

AR glasses such as HoloLens are devices that present information in AR. Users wearing them can view digital information overlaid on real world while seeing through optically. AR glasses are often equipped with camera to scan the real world so digital components can interact with real objects, for example a virtual box can be placed on a real table. Even though AR glasses provide high-quality AR experience, they are not capable of displaying VR contents because of the optical see-through. Immersive VR environments require users to be shut down from the real world. None of the current AR glasses can perform both AR and VR function.

VR HMDs (head mounted displays) are devices for VR experience. There are two types of VR HMDs, wired and wireless. Wired HMDs such as Oculus Rift must be connected to PCs to display VR contents. PCs do the computation and output to wired HMDs. With computational power provided by PCs, users wearing wired HMDs can play large VR games. Even though wired VR HMDs can provide high-quality VR experience, they are lacking mobility due to the wire connected to PCs. We want our system to be movable, so users can walk around to have VR experience about different objects. Wireless HMDs such as Oculus Go, which are also called standalone HMDs, do not have wire connect them to PCs, instead, they are standalone device. They cannot process VR applications which require high computational power, but they can be used at any place, in other words, they have high mobility. Unfortunately, current wireless HMDs are not equipped with high resolution camera to perform video see-through which is essential for AR.

Mobile device-based HMDs can realize the functions we want to implement. A mobile device-based HMD consists of a mobile device, usually a smartphone, and a headset device. Users

wearing the headset with smartphone placed inside can view the smart phone screen with two eyes. Object detection AR can be realized by the combination of smartphone camera and its computational power; VR contents can be displayed on the smartphone screen and viewed through headset. In our research an iPhoneXs and a Google Cardboard v2 headset HOMIDO PRIME (figure 17) is utilized to implement our system. Figure 18 shows a user wearing the HMD.

HOMIDO PRIME[4]

 **Figure 17. HOMIDO PRIME**	Features: - Custom made VR lenses: 110° FOV - Wear glasses compatible - Compatible with most recent smartphones (Android & iPhone) - Optical settings: IPD - Interchangeable face contact foam - Wireless

[4] https://homido.com/shop/prime

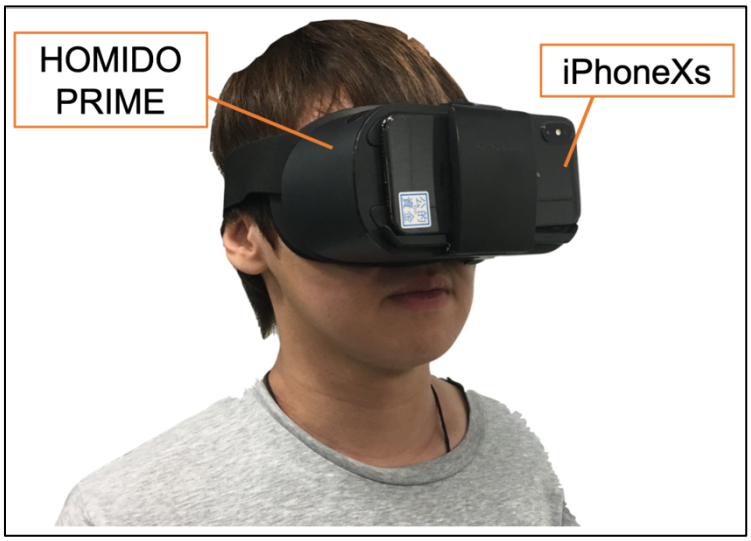

Figure 18. User wearing the HMD

3.4.2 Application Development

Development Environment

- macOS 10.13.5
- Xcode 9.3
- Swift 4.1
- iOS 12.1.2

In this research, a combination of mobile device and VR headset is utilized to make a head mounted display (HMD) to bring users to VR environments from physical objects. The HMD is built with an iPhoneXs and a head mount device HOMiDO PRIME that conforms to the Google Cardboard v2 which has a button to tap the mobile device screen. The system is implemented as an iOS application. As described in Figure 19, the system consists of two phases of operation. The first phase is object detection with AR output, where a user with HMD can see-through with camera captured images, and detection results of objects are shown in the user's view as AR tagged bounding boxes. The second phase is selection and VR information presentation, where the user can select one of the detected objects through aiming the cursor and pressing the button

on head mount device, and a WebVR website linked to the selected object is accessed. Then the user can explore the VR world for information. When the button is pressed again, the system changes back to the first phase again.

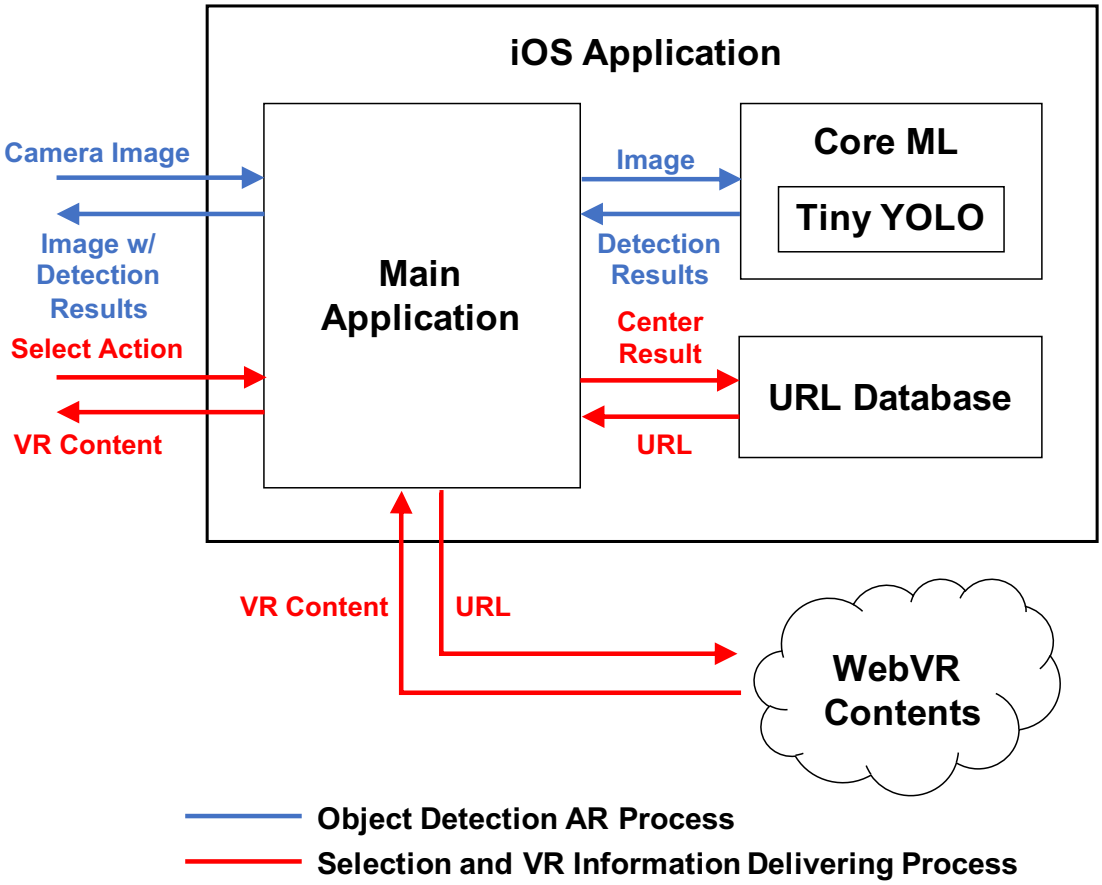

Figure 19. System layout

3.4.2.1 Deep Learning Model in iOS App

The deep learning model introduced in section 4.2.2.3 cannot be implemented into an iOS application directly. Conversion to CoreML format is required. Apple's Core ML is a framework for implementing machine leaning model into iOS application.

Our trained model has Tensorflow format, and to convert it into CoreML format, we used **tf-coreml**[5], a Tensorflow to CoreML converter.

[5] https://github.com/tf-coreml/tf-coreml.git

3.4.2.2 Object Detection AR process

An open source project[6] can be referenced. To realize video see-through, camera captures images and display them side-by-side with 30 frames per second. Along with the video see-through thread, another thread which is responsible for object detection is processing. While images are displayed on the screen, the image saved in image buffer is retrieved and runs through object detection process. The image is resized into 416x416 pixels because the Tiny YOLO model's input size is 416x416 pixels. Then the resized image becomes input of Tiny YOLO model. After computation, it outputs matrix which contains possible existing objects' position, size and probability information. The output is then decoded into coordinates with corresponding objects probabilities. We filter out bounding boxes which have confidence scores lower than 0.3 to ensure bounding boxes with high uncertainty does not show up. The predicted bounding boxes are in the coordinate space of the model's input image which is a square image of 416x416 pixels. Since the actual image captured and displayed on the screen has 1920 to 1080 aspect ratio, we need to resize the bounding boxes according to the ratio. The bounding boxes in correct coordinate space are then displayed side-by-side on top of the displayed side-by-side image. The object names are also displayed at the top of bounding boxes. The object detection AR view is shown in figure 20.

[6] https://github.com/hollance/YOLO-CoreML-MPSNNGraph

Figure 20. Object Detection AR View

3.4.2.3 Selection and VR information delivering process

When an object bounding box bounds the center coordinate of the image, the object is chosen to be selected. A yellow circle cursor located at the center of the image helps users to aim object they want to select. The actual selection action happens when users press the button on the head mount device which can trigger a tap action on screen. The selection action is confirmed by tapping on screen, thus user users wearing HMD can perform object selection without taking it off. VR information delivering process is trigged by selection. The selected object name is them matched in URL database and the corresponding WebVR website is accessed. WebKit, a web engine, is used in the application to browse websites, and it is capable to WebVR, so users can immerse into the VR environment and interact with it. The tap action triggered in VR mode will reactivate see-through process, so the users can come back to the real world when button pressed.

3.4.3 Program Function Flow

To shift between two phases of operation, two swift files are created, and the detailed function flow is shown below.

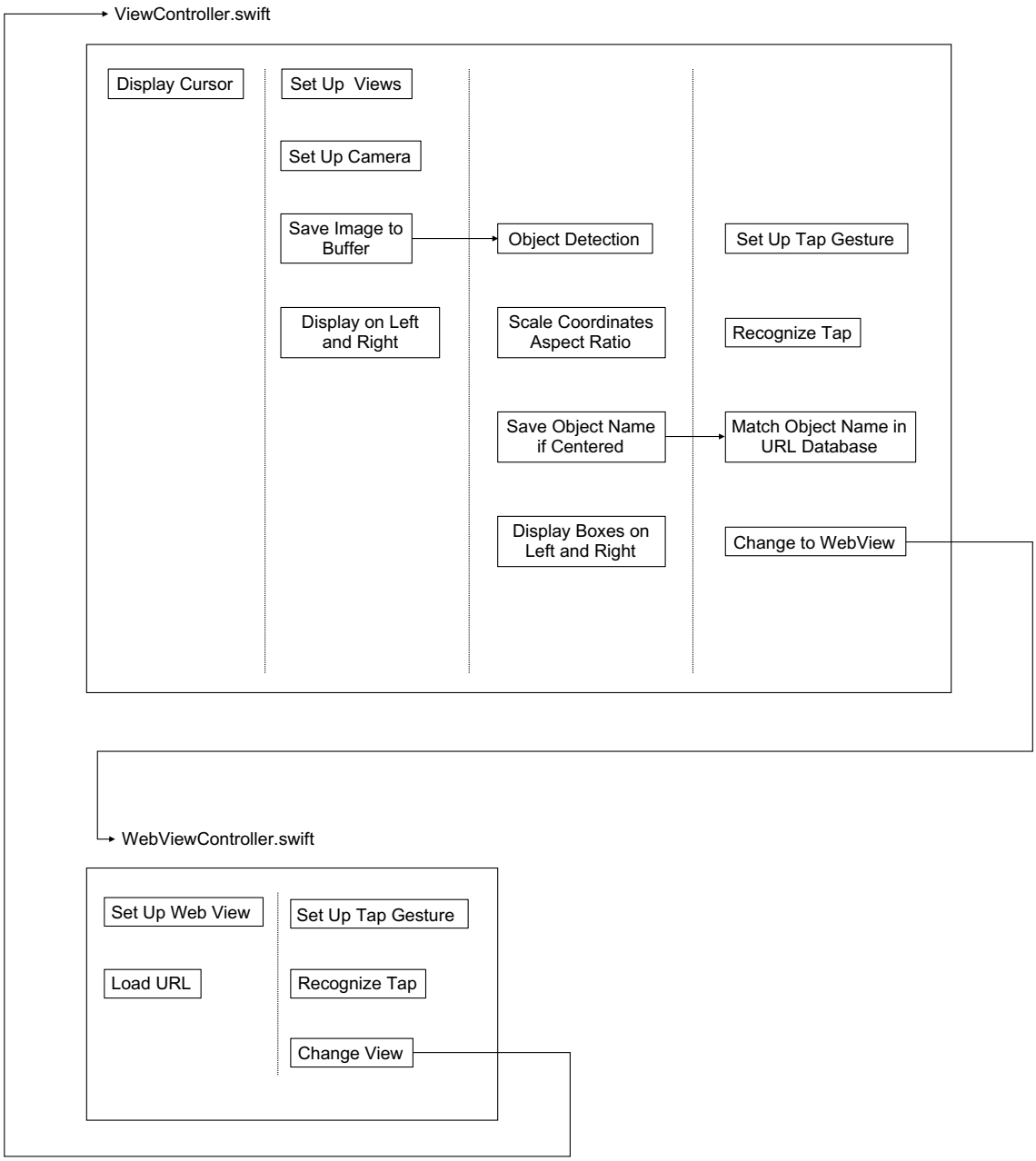

Figure 21. Program Function Flow

In ViewController.swift, there are four threads. In the first thread, two yellow cursors are displayed on the screen for left and right view. In the second thread, left and right views are set, and camera is activated to capture images. Then the captured image is saved to the pixel buffer, and the save image is displayed both on left and right views. In the third thread, the image saved in the buffer runs through object detection and the box coordinates are scaled according to the aspect ratio of views. The detected object name is saved if its bounding box is located at the center. Then detected objects' boxes and names are displayed on left and right views. In the fourth thread, tap gesture is set up and when it recognizes a tap action, the saved object name is matched in the URL database. Then the view changes to web view.

In WebViewController.swift, the web view is set up and the matched URL is accessed. The tap gesture is also set up and when it recognizes a tap action, the view changes to the first view.

CHAPTER 4: PROTOTYPE EVALUATION

4.1 Prototype

A prototype system was developed to verify the functionality. The pre-trained neural network model Tiny YOLO which can recognize 20 classes of objects is used. Although we can build websites for all 20 classes, we chose to develop 3 websites for 3 classes, which are “bottle”, “car”, and “person”. The use cases are when a person becomes interested in a bottle of water and curious where the water comes from; when a person find a cool car passing by and want to take a look inside; when a person sees a professor and wondering what his laboratory looks like.

For “bottle”, the linked website is built with a 360-degree image of a beautiful spring in a mountain and description texts about the water source as shown in Figure 23. For “car”, the linked website is built with a 360-degree image taken from inside of a car and description texts about mechanical features as shown in Figure 24. For “person”, the linked website is built with a 360-degree image of his working environment which is a laboratory in this prototype and description texts as shown in Figure 25.

Figure 22. Prototype Object Detection AR

Figure 23. VR spring when "bottle" selected

Figure 24. VR car interior when "car" selected

Figure 25. VR laboratory when "person" selected

4.2 Evaluation

A user experience evaluation experiment of the prototype was held during an exhibition. The prototype mentioned in section 5.1 was used and a bottle of natural spring water, a model car, and a picture of professor were put on a table. We asked participants wearing HMD to select objects which they were interested in and see the VR worlds. The evaluation was conducted through a survey. 30 participants answered the survey questionnaire after experiencing the system. The age and gender variance distribution of participants are shown in Figure 26 and 27. The participants in good variance answered four questions about object recognition AR experience, four questions about VR websites experience, and three questions about the whole system function experience.

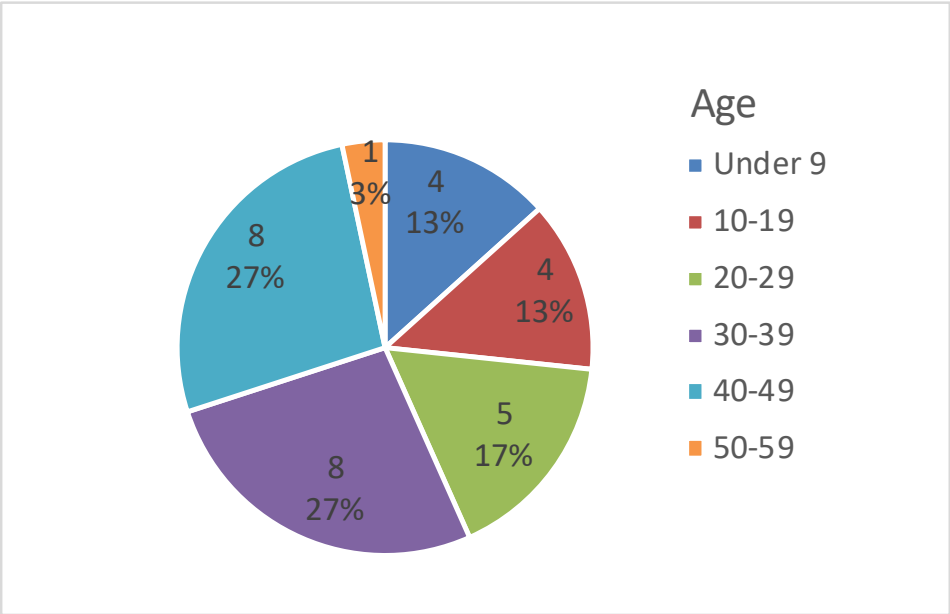

Figure 26. Age Variance

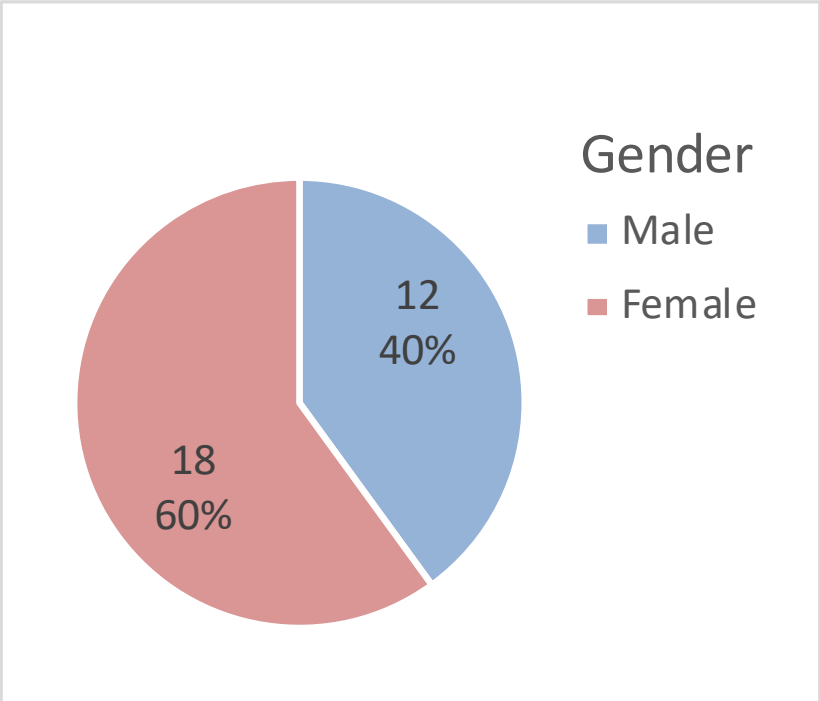

Figure 27. Gender Variance

Same four questions about object detection AR and VR contents experiences were answered: whether the user did not feel sickness, spatial unawareness, delay, and narrow visual field. The answers were in five levels Likert scale (-2=Disagree, -1=Slightly Disagree, 0=Neither,

1=Slightly Agree, 2=Agree). The evaluation results are similar for both functions. The results are shown in Figure 28 and Figure 29.

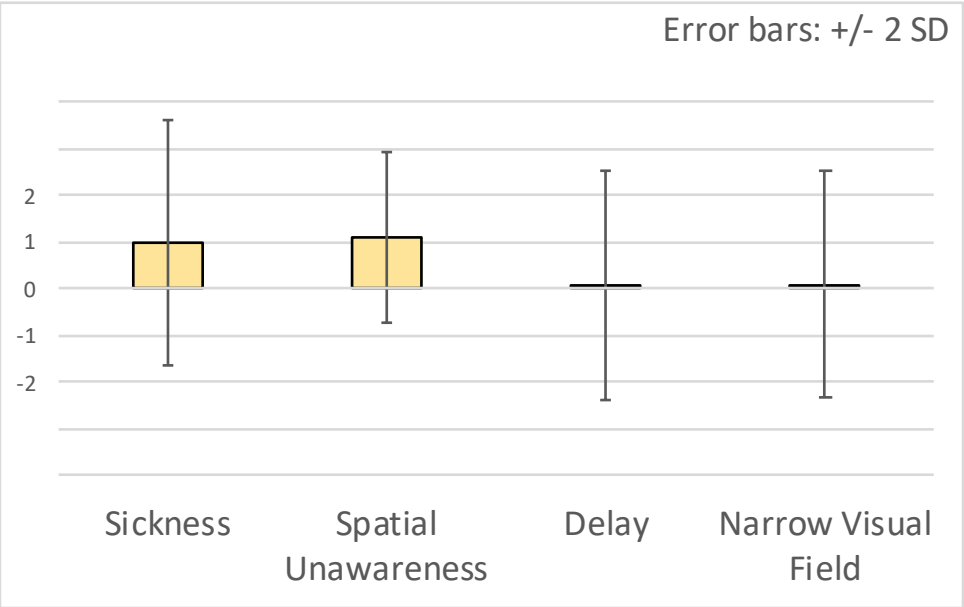

Figure 28. Object detection AR evaluation

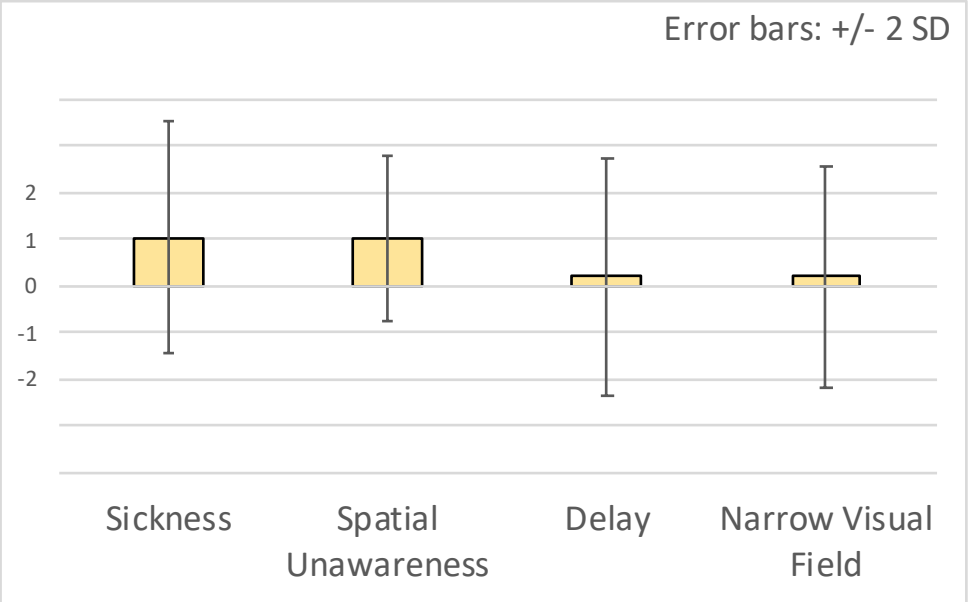

Figure 29. VR contents evaluation

For object detection AR, the positive mean results of sickness and spatial unawareness indicate that users are comfortable with video see-through of this system, and they can recognize the spatial relationship between objects. The slightly positive mean results of delay and narrow visual

field indicate compromises of using video see-through are tolerable. The overall user experience of video see-through with object recognition on the smartphone is good. However, since the standard deviations of all four criteria are not small, the user experience of video see-through AR on the smartphone depends on individuals.

For VR websites, the positive mean results of sickness and spatial unawareness indicate that users are comfortable with experiencing VR contents delivered through websites on smartphone and they can recognize the spatial relationship between information components created inside VR contents. The slightly positive mean results of delay and narrow visual field indicate the processing power and hardware limitation of the smartphone are tolerable when providing VR experience. The overall VR experience on the smartphone through websites is good. However, since the standard deviations of all four criteria are not small, the user experience of VR websites on the smartphone depends on individuals.

Three questions about the entire system experience were answered: whether the system is effective as information system, whether the system is interesting, and willingness of using the system in the future. The answers were in five levels Likert scale (-2=Disagree, -1=Slightly Disagree, 0=Neither, 1=Slightly Agree, 2=Agree). The results are shown in Figure 30. The positive mean and low standard deviation result for each question indicate that most users think the object detection-based AR-VR System is effective and interesting as an information system, and they are willing to see more information presented in VR through object detection AR.

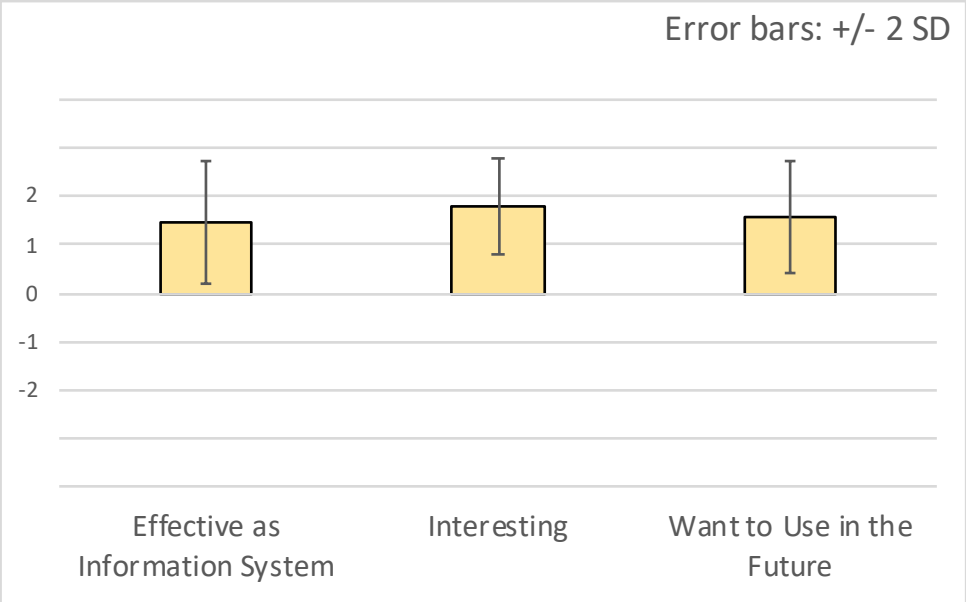

Figure 30. System experience evaluation

CHAPTER 5: PRACTICAL IMPLEMENTATION

A 2-day promotion event was held at Haneda Airport to introduce Shintomi-cho, a town in Miyazaki prefecture, Japan and our system was exhibited as part of the event. The event was held by Shintomi-cho's local companies and governments to promote the town. The purpose of the event is to market local products, promote tourism, and most importantly make more people to know Shintomi-cho. The most famous local specialty is the Shintomi-lychee, a locally farmed lychee with high-quality. Since the Japanese lychee is very rear, and valuable, the main role of the event was the lychee, and the organizers want people to first be interested in the lychee and then become interested in Shintomi-cho. The event booth consisted of town information space, local products selling space, and our experiential AR-VR system space. The actual booth view is shown as figure 31.

Figure 31. Event booth

At the local products selling space, lychees were both for sample tasting and sale. People visiting the booth can have free lychees for tasting. Our system was developed to be able to detect the real lychees, and people with sample lychees can use our system to see information about the lychees. Other than the lychee, Shintomi-cho also has beautiful beach and monthly morning market which can be presented to attract tourists. We implemented three pairs of objects and VR information. when a person sees a lychee, which is the local specialty, he or she will be taken to the actual lychee farm in Shintomi-cho where the lychee was picked; when a person sees the beach scene printed panel, he or she will be taken to the beach of Shintomi-cho; when a person sees the morning market scene printed panel, he or she will be taken to the monthly held morning market in Shintomi-cho.

5.1 System Development

We trained the model to detect three objects which are Shintomi-cho's lychee, beach scene printed panel, and morning market scene printed panel. For each object class, 1000 images are used to train the model. We built three VR contents which are actual Shintomi-cho lychee farm, beach, and morning market, corresponding to the detectable objects. The three detectable objects are shown as figure 32.

Figure 32. Detectable objects: lychee, beach panel and market panel

For lychee, a VR world is created with 360-degree video of the actual lychee farm as shown in figure33. The video length is 1 minute 2 seconds. Users will immerse into the lychee farm and experience the place. At the start of the VR content, a Shintomi-cho person standing at the center of the farm will introduce the lychee, and then the scene will change to the view from different position in the lychee farm while reaming the introduction talk. Users can observe and experience the lychee farm while listening to information about the lychee.

Figure 33. VR lychee farm

For beach, a VR world is created with 360-degree video of the actual beach as shown in figure 34. The video length is 25 seconds. Users who immersed into VR beach, can view the beautiful beach under sunshine. Users can see the ling coastline while hearing the sound of waves.

Figure 34. VR beach

For market, a VR world is created with 360-degree video of the actual marketplace event as shown in figure 35. The video length is 56 seconds. The VR contents shows different shops at the market and users can observe the entire street where the marketplace was held. The crowd's sound will also enhance the immersive experience.

Figure 35. VR market

5.2 Evaluation

The evaluation was conducted through a survey and 60 participants answered the survey questionnaire after experiencing the system. The age and gender variance distribution of participants are shown in Figure 36 and 37. The participants in good variance answered four questions about object recognition AR experience, four questions about VR contents experience, and three questions about over all experience.

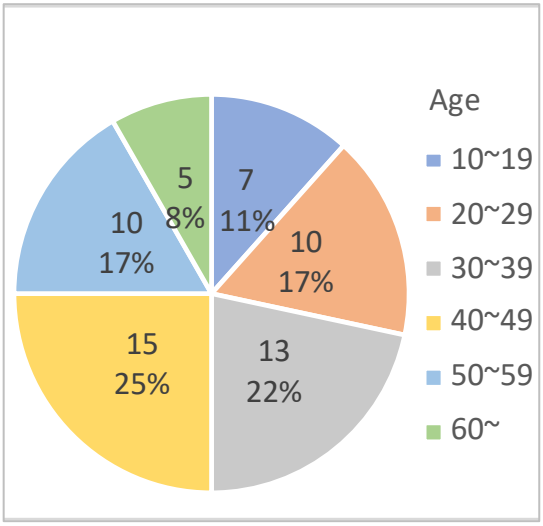

Figure 36. Event age variance

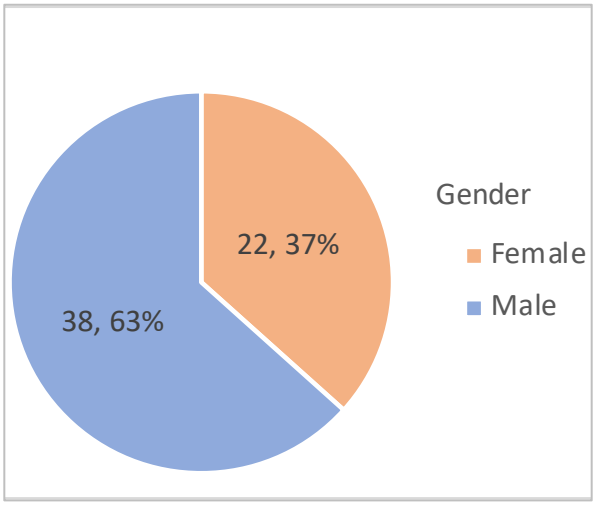

Figure 37. Event gender variance

5.2.1 Object Detection AR and VR Contents Evaluation

Same four questions about object detection AR and VR contents experiences were answered: whether the user did not feel sickness, spatial unawareness, delay, and narrow visual field. The answers were in five levels Likert scale (-2=Disagree, -1=Slightly Disagree, 0=Neither, 1=Slightly Agree, 2=Agree). The evaluation results are similar for both functions. The results are shown in Figure 38 and Figure 39.

The positive mean results indicate that users are comfortable with both AR and VR functions of this system regarding to sickness, spatial unawareness, delay, and narrow visual field. However, since the standard deviations of all four criteria for both functions are not small, the user experience of object detection AR and VR on the smartphone depends on individuals.

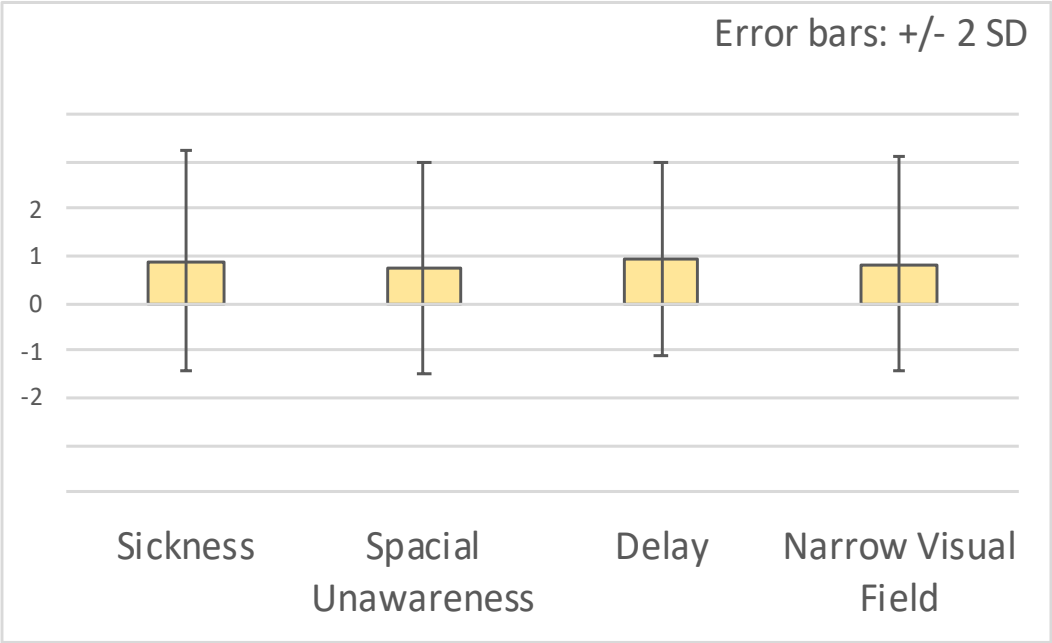

Figure 38. Event object detection AR experience evaluation

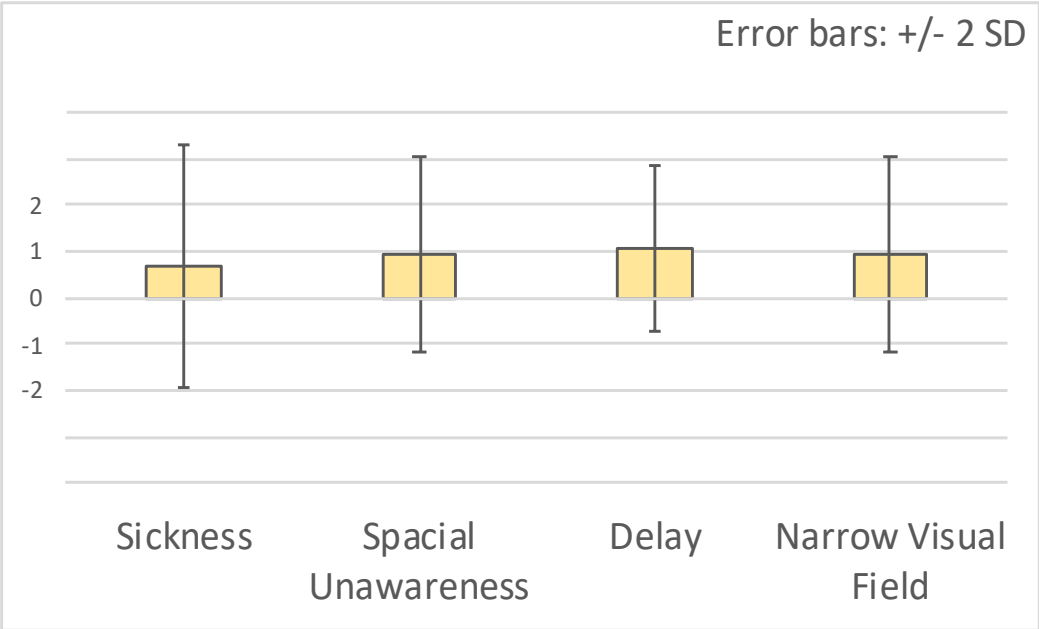

Figure 39. Event VR contents experience evaluation

5.2.2 System Experience Evaluation

Three questions about the entire system experience were answered: whether the system is effective as information system, whether the system is interesting, and willingness of using the

system in the future. The answers were in five levels Likert scale (-2=Disagree, -1=Slightly Disagree, 0=Neither, 1=Slightly Agree, 2=Agree). The results are shown in Figure 40. The positive mean and low standard deviation result for each question indicate that most users think the object detection-based AR-VR System is effective and interesting as an information system, and they are willing to see more information presented in VR through object detection AR.

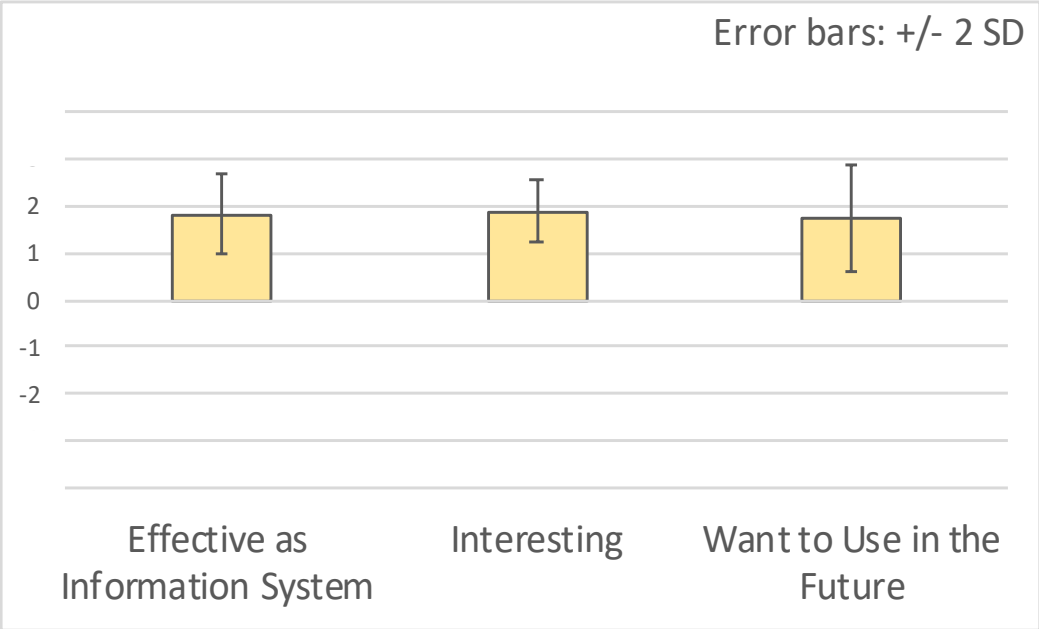

Figure 40. Event system experience evaluation

Other than the questionnaires, we also interviewed participants for comments. Many participants commented that they did not know lychees are also planted in Japan, and it is their first time to learn what a lychee farm looks like. Since they could view the VR contents triggered by the actual lychee they have, they understood the lychee that they were holding came from a well-organized lychee farm and it also proved that the lychee was authentic Japanese lychee. This experience stimulated them to actually purchase lychees.

A person who likes surfing commented that Shintomi-cho's beach must be a good place for surfing, because in the VR beach, he could feel the wave. He could not wait to actually go to Shintomi-cho to visit the beach. This type of understanding could only be delivered through VR,

because VR made him immerse into the scene.

Some people who had been to Shintomi-cho got excited while seeing the VR morning market. They commented that the VR experience provided them chance to revisit the place they had been to and find out the street did not change a lot since their last visit.

CHAPTER 6: FUTURE WORK

In this research, the mobile device is utilized because it satisfies both hardware and software requirements to implement our concept. We used the camera to capture images and display them on the screen to realize video see-though function. There are several improvements can be done regarding see-though AR function.

In our system, one camera captures images and display them side-by-side and the user will only see-through the world in two-dimensional, which losses depth information. To realize more realistic see-through experience, three-dimensional image output is needed, and using two cameras to capture images corresponding to two eyes of human can solve this problem. Even though the two-dimensional video see-through we developed had relatively good evaluation, improving it into three-dimensional video see-through will lead to better user experience.

Time lag in video see-through is unavoidable because we need to wait the camera to capture images and display them on the screen, which is considered to be a factor which will cause VR sickness. Optical see-through is preferred to solve this problem and users can observe the real world through optical glasses. Optical see-through devices are immerging in recent years such as HoloLens.

Regarding the VR experience, the mobile device is not produced specifically for VR experience and it has relatively narrow visual field compared to VR headsets for example, Oculus Rift. VR headsets usually have higher resolution display which can show higher quality VR contents. To

improve the system user experience on VR information presentation, a higher quality VR device can be used.

The transform from AR to VR can be considered more. In our system, due to the limitation of hardware and software, the visual field of see-through and VR contents are different, and when the AR view transforms into VR view, users will notice obvious changes in their view. Also, the view will transition from AR to VR directly. Further studies can be done to investigate the transition from AR to VR, for example, animation between AR view and VR view.

As mentioned in related studies in chapter 3, even though VR is a better format of information compared with text information, there is also advantages of using text. When details needed to be delivered, text information will perform better. Thus, for our system, further investigation on effectiveness of using VR contents as output of object detection AR system on different target objects can be conducted.

CHAPTER 7: CONCLUSION

In this research, we proposed object detection-based AR-VR system which utilize VR as an information delivering method to enhance the object detection-based AR system. When user interested in an object, the user can select the object with box tagged, and immerse into the VR environment for information about the object. The system is implemented on mobile device-based head mounted display as a mobile application. To implement the object detection function, deep-learning image recognition algorithm YOLO is applied to our system after mechanism comparison. To output VR contents, customized VR websites are accessed. A prototype and a practical implementation were developed to evaluate the system and the effectiveness and potential of this system are verified.

CHAPTER 8: ACKNOWLEDGEMENT

I would like to thank Professor Ogi, who guided me throughout the thesis. He has provided valuable advices and a lot of support. I would also like to thank Professor Shirasaka, who provided helpful feedback and comments.

I would like to thank staff of Shintomi-cho and Haneda Airport who supported me throughout the evaluation event.

I thank all participants for answering the survey which helped me to conduct a meaningful evaluation.

CHAPTER 9: REFERENCES

[1] Azuma, R., et al. "Recent Advances in Augmented Reality." *IEEE Computer Graphics and Applications*, vol. 21, no. 6, 2001, pp. 34–47.

[2] Bastug, Ejder, et al. "Toward Interconnected Virtual Reality: Opportunities, Challenges, and Enablers." *IEEE Communications Magazine*, vol. 55, no. 6, 2017, pp. 110–117.

[3] Santos, Marc Ericson C., et al. "Augmented Reality Learning Experiences: Survey of Prototype Design and Evaluation." *IEEE Transactions on Learning Technologies*, vol. 7, no. 1, 2014, pp. 38–56.

[4] Simeone, Luca, and Salvatore Iaconesi. "Anthropological Conversations: Augmented Reality Enhanced Artifacts to Foster Education in Cultural Anthropology." *2011 IEEE 11th International Conference on Advanced Learning Technologies*, 2011.

[5] Andujar, Jose Manuel, et al. "Augmented Reality for the Improvement of Remote Laboratories: An Augmented Remote Laboratory." *IEEE Transactions on Education*, vol. 54, no. 3, 2011, pp. 492–500.

[6] Nolle, Stefan, and Gudrun Klinker. "Augmented Reality as a Comparison Tool in Automotive Industry." *2006 IEEE/ACM International Symposium on Mixed and Augmented Reality*, 2006.

[7] Regenbrecht, H., et al. "Augmented Reality Projects in the Automotive and Aerospace

Industries." *IEEE Computer Graphics and Applications*, vol. 25, no. 6, 2005, pp. 48–56.

[8] Sekiguchi, Kentaro. "Museum Exhibition Linking with the Real World by Using Augmented Reality Technology." *Keio Gijuku Daigaku Daigakuin Shisutemu Dezain Manejimento Kenkyuka*, Mar. 2014, pp. 101.

[9] Takesue, Yusuke. "Development of AR System Based on Deep Learning and Operation Using Gamification." *Keiō Gijuku Daigaku Daigakuin Shisutemu Dezain Manejimento Kenkyūka*, Mar. 2018.

[10] Slavova, Yoana, and Mu Mu. "A Comparative Study of the Learning Outcomes and Experience of VR in Education." *2018 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, 2018.

[11] Usui, Shoko, et al. "Development and Evaluation Experiment of Display Media Using VR for Art Appreciation Learning." *2017 6th IIAI International Congress on Advanced Applied Informatics (IIAI-AAI)*, 2017.

[12] Fiala, M. "ARTag, a Fiducial Marker System Using Digital Techniques." *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR05)*.

[13] Zhang, Xiang, et al. "Visual Marker Detection and Decoding in AR Systems: a Comparative Study." *Proceedings. International Symposium on Mixed and Augmented Reality*.

[14] Mohring, M., et al. “Video See-Through AR on Consumer Cell-Phones.” *Third IEEE and ACM International Symposium on Mixed and Augmented Reality*, 24 Jan. 2005.

[15] Narzt, et al. “Pervasive Information Acquisition for Mobile AR-Navigation Systems.” *Proceedings DARPA Information Survivability Conference and Exposition MCSA-03*, 2003.

[16] Vossiek, M., et al. “Wireless Local Positioning - Concepts, Solutions, Applications.” *Radio and Wireless Conference, 2003. RAWCON 03. Proceedings*.

[17] Paucher, Remi, and Matthew Turk. “Location-Based Augmented Reality on Mobile Phones.” *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Workshops*, 2010.

[18] Kunkel, Sebastian, et al. “A Concept for Infrastructure Independent Localization and Augmented Reality Visualization of RFID Tags.” *2009 IEEE MTT-S International Microwave Workshop on Wireless Sensing, Local Positioning, and RFID*, 2009.

[19] Sun, Jianqiang, et al. “RF-ISee: Identify and Distinguish Multiple RFID Tagged Objects in Augmented Reality Systems.” *2016 IEEE 36th International Conference on Distributed Computing Systems (ICDCS)*, 2016.

[20] Xie, Lei, et al. “TaggedAR: An RFID-Based Approach for Recognition of Multiple Tagged

[21] Objects in Augmented Reality Systems.” *IEEE Transactions on Mobile Computing*, vol. 18,

no. 5, 2019, pp. 1188–1202.

[22] Lin, Cheng-Hung, et al. “A Novel Campus Navigation APP with Augmented Reality and Deep Learning.” *2018 IEEE International Conference on Applied System Invention (ICASI)*, 2018, doi:10.1109/icasi.2018.8394464.

[23] Lalonde, Jean-Francois. “Deep Learning for Augmented Reality.” *2018 17th Workshop on Information Optics (WIO)*, 2018.

[24] Szegedy, Christian, et al. “Going Deeper with Convolutions.” *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[25] Girshick, Ross, et al. “Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation.” *2014 IEEE Conference on Computer Vision and Pattern Recognition*, 2014.

[26] Girshick, Ross. “Fast R-CNN.” *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015.

[27] Ren, Shaoqing, et al. “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks.” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, 2017, pp. 1137–1149.

[28] Redmon, Joseph, et al. “You Only Look Once: Unified, Real-Time Object Detection.” *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[29] Redmon, Joseph, and Ali Farhadi. “YOLO9000: Better, Faster, Stronger.” *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[30] Butcher, Peter W. S., and Panagiotis D. Ritsos. “Building Immersive Data Visualizations for the Web.” *2017 International Conference on Cyberworlds (CW)*, 2017.

[31] Gunkel, Simon, et al. “WebVR Meets WebRTC: Towards 360-Degree Social VR Experiences.” *2017 IEEE Virtual Reality (VR)*, 2017.

[32] Pakkanen, Toni, et al. “Interaction with WebVR 360° Video Player: Comparing Three Interaction Paradigms.” *2017 IEEE Virtual Reality (VR)*, 2017.

CHAPTER 10: APPENDICES

10.1 Tiny YOLO Modeling Procedure

- Download and Install Darkflow from https://github.com/thtrieu/darkflow.git

  pip install -e .

- Move the training dataset to corresponding directories. Images to './images/' directory and annotations to './annotations/'.

- Copy and rename one of the configuration files that is going to be used. A configuration file contains neural network structure information. Configuration files are located in './cfg/', and 'tiny-yolo-voc.cfg' is our choice. We renamed the copy of it to 'tiny-yolo-new.cfg'.

- Change several parameters in configuration file. If 2 classes of objects are annotated in dataset, change classes to 2.

  classes=2

  Change filters number to boxes_number*(boxes_info+classes number). 5 is the default box number and we keep this. Boxes_info are P(Object), X, Y, Width, Height, thus 5. We change filters to 5*(5+2), which is 35.

  Filters = 35

tiny-yolo-new.cfg

```
[net]
batch=64
subdivisions=8
width=416
height=416
channels=3
momentum=0.9
decay=0.0005
angle=0
saturation = 1.5
exposure = 1.5
hue=.1
```

```
learning_rate=0.001
max_batches = 40100
policy=steps
steps=-1,100,20000,30000
scales=.1,10,.1,.1

[convolutional]
batch_normalize=1
filters=16
size=3
stride=1
pad=1
activation=leaky

[maxpool]
size=2
stride=2

[convolutional]
batch_normalize=1
filters=32
size=3
stride=1
pad=1
activation=leaky

[maxpool]
size=2
stride=2


[convolutional]
batch_normalize=1
filters=64
size=3
stride=1
pad=1
activation=leaky

[maxpool]
size=2
stride=2

[convolutional]
batch_normalize=1
filters=128
size=3
stride=1
pad=1
activation=leaky
```

```
[maxpool]
size=2
stride=2

[convolutional]
batch_normalize=1
filters=256
size=3
stride=1
pad=1
activation=leaky

[maxpool]
size=2
stride=2

[convolutional]
batch_normalize=1
filters=512
size=3
stride=1
pad=1
activation=leaky

[maxpool]
size=2
stride=1


[convolutional]
batch_normalize=1
filters=1024
size=3
stride=1
pad=1
activation=leaky

###########

[convolutional]
batch_normalize=1
size=3
stride=1
pad=1
filters=1024
activation=leaky

[convolutional]
size=1
stride=1
```

```
pad=1
filters=35
activation=linear

[region]
anchors = 1.08,1.19,    3.42,4.41,    6.63,11.38,    9.42,5.11,    16.62,10.52
bias_match=1
classes=2
coords=4
num=5
softmax=1
jitter=.2
rescore=1

object_scale=5
noobject_scale=1
class_scale=1
coord_scale=1

absolute=1
thresh = .5
random=1
```

- Modify ‘labels.txt’ file and specify names of classes. Write down “dog” and “ball”, if our dataset has dogs and balls annotated.
- Download corresponding pre-trained weights to ‘./bin/'. In our case, ‘tiny-yolo-voc.weights’ is downloaded
- Create a python script defining training options.

```
import numpy as np

from darkflow.net.build import TFNet
import cv2


options = {"model": "./cfg/tiny-yolo-new.cfg",
                "load": "./bin/tiny-yolo-voc.weights",
                "batch": 8,
                "epoch": 30,
                "gpu": 1.0,
                "train": True,
```

```
                "annotation": "./data/annotations/",
                "dataset": "./data/images/"}

tfnet = TFNet(options)
tfnet.train()
tfnet.savepb()
```

Set “model” to the configuration file we created

Set “load” to the weights we downloaded

Set “batch” to 8, where batch size defines the number of samples that will be propagated once

Set “epoch” to 30, where epoch defines how many times a dataset will be trained through

Set “gpu” to 1.0 to enable GPU hardware

Set “train” to true to claim it is a training process

Set “annotation” to the directory with XML files

Set “dataset” to the directory with training images

10.2 Example A-Frame Website Code

```
<html>
  <head>
    <meta charset="utf-8">
    <title>Welcome to Shintomi</title>
    <meta name="description" content="Welcome to Shintomi">
    <meta name="apple-mobile-web-app-capable" content="yes">
    <script src="https://aframe.io/releases/0.8.0/aframe.min.js"></script>


    <script>
      AFRAME.registerComponent('auto-enter-vr', {
  init: function () {

    this.el.sceneEl.enterVR();

  }
});
    </script>
```

```
  </head>
  <body>

    <a-scene vr-mode-ui="enabled: false" auto-enter-vr>
      <a-videosphere rotation="0 -90 0" material="side:back" src="#video" >
      </a-videosphere>


      <a-entity camera rotation = "0 0 0" id="cameraview"   look-controls wasd-controls
>
      </a-entity>
      <a-assets>
        <!-- Single source video. -->
        <video id="video" style="display:none"
                autoplay loop crossorigin="anonymous" playsinline webkit-playsinline>
          <!-- MP4 video source. -->
          <source type="video/mp4"
                src="https://cdn.glitch.com/61d7c84c-8655-4084-adc1-
8b85826f1391%2Flychee.mp4?1558864370678" />
        </video>
      </a-assets>
  </a-scene>
<script>
  var v = document.querySelector('#video');
  v.play();
</script>
  </body>
</html>
```

A VR website can be created with a single .html file by including A-Frame in the <head>.

```
<head>
  <script src="https://aframe.io/releases/0.8.0/aframe.min.js"></script>
</head>
```

We used A-Frame version 0.8.0.

Since our system requires direct enter to VR mode, additional JavaScript is included.

```
<script>
      AFRAME.registerComponent('auto-enter-vr', {
  init: function () {

    this.el.sceneEl.enterVR();

  }
});
    </script>
```

A camera entity is created to display the VR world.

```
<a-entity camera rotation = "0 0 0" id="cameraview"    look-controls wasd-controls >
        </a-entity>
```

Different contents can be created in VR website. In the example, a 360-degree video is included. Since we want our system to play directly instead of waiting for users interact, we need to set the video to play automatically and play in loop.

```
<a-videosphere rotation="0 -90 0" material="side:back" src="#video" >
</a-videosphere>
<a-assets>
        <!-- Single source video. -->
        <video id="video" style="display:none"
                autoplay loop crossorigin="anonymous" playsinline webkit-playsinline>
          <!-- MP4 video source. -->
          <source type="video/mp4"
                src="https://cdn.glitch.com/61d7c84c-8655-4084-adc1-
8b85826f1391%2Flychee.mp4?1558864370678" />
        </video>
</a-assets>


<script>
  var v = document.querySelector('#video');
  v.play();
</script>
```

10.3 iOS application code

WebViewController.swift

```
//
//   WebViewController.swift
//
//
//   Created by XIANGYI GAO on 2018/09/09.
//   Copyright © 2018年 XIANGYI GAO. All rights reserved.
//
import Foundation
```

```
import UIKit
import WebKit

class WebViewController: UIViewController, UIGestureRecognizerDelegate {
//      @IBOutlet weak var myWebView1: WKWebView!
    var myWebView1 = FullScreenWKWebView()
    override func viewDidLoad() {
        super.viewDidLoad()
        let config = WKWebViewConfiguration()
        config.allowsInlineMediaPlayback = true
        config.mediaTypesRequiringUserActionForPlayback = []
        self.myWebView1 = FullScreenWKWebView(frame: CGRect(x: 0, y: 0, width:
812, height: 375), configuration: config) // configと共にインスタンス化
        self.myWebView1.isOpaque = false;
        self.view.addSubview(self.myWebView1)
        // Do any additional setup after loading the view.
        self.myWebView1.load(URLRequest(url:URL(string:webName)!))

        let tapGesture:UITapGestureRecognizer = UITapGestureRecognizer(
            target: self,
            action: #selector(ViewController.tapped(_:)))

        // デリゲートをセット
        tapGesture.delegate = self
        self.myWebView1.addGestureRecognizer(tapGesture)
    }

    func gestureRecognizer(_ gestureRecognizer: UIGestureRecognizer,
shouldRecognizeSimultaneouslyWith otherGestureRecognizer: UIGestureRecognizer) ->
Bool{
        return true
    }

    @objc func tapped(_ sender: UITapGestureRecognizer) {
        self.myWebView1.loadHTMLString("<html/>", baseURL: nil)
        performSegue(withIdentifier: "goToSel", sender: self)
```

```
        print("View Tap")
    }

    override func didReceiveMemoryWarning() {
        super.didReceiveMemoryWarning()
        // Dispose of any resources that can be recreated.
    }

    override func viewWillAppear(_ animated: Bool) {
        super.viewWillAppear(animated)
        UIDevice.current.setValue(UIInterfaceOrientation.landscapeRight.rawValue,
forKey: "orientation")
    }
    override var shouldAutorotate: Bool {
        return false
    }
    override var supportedInterfaceOrientations: UIInterfaceOrientationMask {
        return.landscapeRight
    }
    override var preferredInterfaceOrientationForPresentation: UIInterfaceOrientation{
        return.landscapeRight
    }
}
class FullScreenWKWebView: WKWebView {
    override var safeAreaInsets: UIEdgeInsets {
        return UIEdgeInsets(top: 0, left: 0, bottom: 0, right: 0)
    }
}
```

ViewController.swift

```
//
//   ViewController.swift
//   tinyYoloVR
//
//   Created by XIANGYI GAO on 2018/04/27.
```

```
//   Copyright © 2018年 XIANGYI GAO. All rights reserved.
//

import UIKit
import AVFoundation
import Vision
var webName : String = "…"
var flag = 0

class ViewController: UIViewController,
AVCaptureVideoDataOutputSampleBufferDelegate,
UIGestureRecognizerDelegate,
URLSessionTaskDelegate {

    @IBOutlet weak var boxView: UIView!
    @IBOutlet weak var cameraView: UIView!

    var input:AVCaptureDeviceInput!
    var output:AVCaptureVideoDataOutput!
    var session:AVCaptureSession!
    var camera:AVCaptureDevice!
    var imageView:UIImageView!
    var firstView:UIImageView!
    var secondView:UIImageView!
    var pixelBuffer: CVPixelBuffer!
    var boundingBoxes = [BoundingBox]()
    var boundingBoxes1 = [BoundingBox]()
    var colors: [UIColor] = []
    var request: VNCoreMLRequest!
    let yolo = YOLO()
    var count = 0
    var offcount = 0
    var result : String = "…"
    var judge : String = "…"
    var objectNotAimed : Bool = true
    var firstSetUpDisplay: Bool = true
```

```
    let dispatchQueueSIS = DispatchQueue(label: "ObjectDetection")

    override func viewDidLoad() {
        flag = 0
        super.viewDidLoad()

        // Display two yellow circle cursors
        if firstSetUpDisplay == true{
            drawCursorCircles()
        }
        // Set up Tap Gesture
        setUpTapGesture()

        // Set up Bounding Boxes
        setUpBoundingBoxes()

        // Set up Screen Display
         if firstSetUpDisplay == true{
        setupDisplay()
        }
        firstSetUpDisplay = false
        // カメラの設定
        setupCamera()

       dispatchQueueSIS.async {
        sleep(UInt32(1))
        while(flag == 0){
        usleep(50000)
        self.setUpVision()
        try?                                    VNImageRequestHandler(cvPixelBuffer:
self.pixelBuffer).perform([self.request])
        }
        }

    }
```

```
    func drawCursorCircles(){
        let screenWidth = self.cameraView.bounds.width        // Get device screen width
        let screenHeight = self.cameraView.bounds.height    // Get device screen height
        // Draw two yellow cirlces
        let CircleDraw = circleDraw(frame: CGRect(x: 0, y: 0, width: screenWidth, height:
screenHeight))
        self.view.addSubview(CircleDraw)
        CircleDraw.isOpaque = false
    }

    func setUpTapGesture(){
        let tapGesture:UITapGestureRecognizer = UITapGestureRecognizer(
            target: self,
            action: #selector(ViewController.tapped(_:))
        )
        // デリゲートをセット
        tapGesture.delegate = self;
        // View に追加.
        self.view.addGestureRecognizer(tapGesture)
    }

    @objc func tapped(_ sender: UITapGestureRecognizer){

        if judge == "新富ライチ" {
            flag = 1
            webName = "https://lychee.glitch.me"
            performSegue(withIdentifier: "goToWeb", sender: self)
        }
        if judge == "朝市" {
            flag = 1
            webName = "https://asaichi.glitch.me"
            performSegue(withIdentifier: "goToWeb", sender: self)
        }
        if judge == "富田浜" {
            flag = 1
            webName = "https://umibe.glitch.me/"
```

```
                performSegue(withIdentifier: "goToWeb", sender: self)
            }
        }
        func setUpBoundingBoxes() {
            for _ in 0..<YOLO.maxBoundingBoxes {
                boundingBoxes.append(BoundingBox())
                boundingBoxes1.append(BoundingBox())
            }

            // Make colors for the bounding boxes. There is one color for each class,
            // 20 classes in total.



            let color0 = UIColor(red: 0.2, green: 0.6, blue: 0.2, alpha: 0.5) // alpha = 0.5
            colors.append(color0)
            let color1 = UIColor(red: 0.8, green: 0.5, blue: 0.2, alpha: 0.5) // alpha = 0.5
            colors.append(color1)
            let color2 = UIColor(red: 0.2, green: 0.6, blue: 0.8, alpha: 0.5) // alpha = 0.5
            colors.append(color2)

        }

        func setUpVision() {
            guard let visionModel = try? VNCoreMLModel(for: yolo.model.model) else {
                print("Error: could not create Vision model")
                return
            }

            request = VNCoreMLRequest(model: visionModel, completionHandler:
visionRequestDidComplete)

            // NOTE: If you choose another crop/scale option, then you must also
            // change how the BoundingBox objects get scaled when they are drawn.
            // Currently they assume the full input image is used.
            request.imageCropAndScaleOption = .scaleFill
        }
```

```
    func visionRequestDidComplete(request: VNRequest, error: Error?) {
        if let observations = request.results as? [VNCoreMLFeatureValueObservation],
            let features = observations.first?.featureValue.multiArrayValue {

            let boundingBoxes = yolo.computeBoundingBoxes(features: features)
            showOnMainThread(boundingBoxes)
        }
    }
    func showOnMainThread(_ boundingBoxes: [YOLO.Prediction]) {
        DispatchQueue.main.async {

            self.show(predictions: boundingBoxes)
            for box in self.boundingBoxes {
                box.addToLayer(self.boxView.layer)

            }
            for box1 in self.boundingBoxes1 {
                box1.addToLayer(self.boxView.layer)

            }
        }
    }
    func show(predictions: [YOLO.Prediction]) {
        objectNotAimed = true
        for i in 0..<boundingBoxes.count {

            if i < predictions.count {
                let prediction = predictions[i]

                // The predicted bounding box is in the coordinate space of the input
                // image, which is a square image of 416x416 pixels. We want to show
it
                // on the video preview, which is as wide as the screen and has a 4:3
                // aspect ratio. The video preview also may be letterboxed at the top
                // and bottom.
```

```
                let width = cameraView.bounds.width/2
                let height = cameraView.bounds.width * (1920/1080) / 2
                let scaleX = width / CGFloat(YOLO.inputWidth)

                let scaleY = height / CGFloat(YOLO.inputHeight)
                let top = (cameraView.bounds.height - cameraView.bounds.width *
(1920/1080)) / 2

                // Translate and scale the rectangle to our own coordinate system.
                var rect = prediction.rect
                rect = rect.applying(CGAffineTransform(rotationAngle:
CGFloat(1*Double.pi/2)))
                rect.origin.x = (rect.origin.x*scaleX + (3*width/2))
                rect.origin.y *= scaleY
                rect.origin.y += top
                rect.size.width *= scaleX
                rect.size.height *= scaleY


                var rect1 = rect
                rect1.origin.y += (cameraView.bounds.height/2)-top

                // Show the bounding box.
                let label = String(format: "%@", labels[prediction.classIndex])
                let color = colors[prediction.classIndex]

                if (rect.origin.x < cameraView.bounds.width/2) && ((rect.origin.x +
rect.size.width) > cameraView.bounds.width/2) && (rect.origin.y <
(cameraView.bounds.height - height)/2) && ((rect.origin.y + rect.size.height) >
(cameraView.bounds.height - height)/2){
                    objectNotAimed = false
                    judge = labels[prediction.classIndex]
                }


                boundingBoxes[i].show(frame: rect, label: label, color: color)
```

```
                boundingBoxes1[i].show(frame: rect1, label: label, color: color)
            } else {
                boundingBoxes[i].hide()
                boundingBoxes1[i].hide()
            }
        }
        if objectNotAimed{
            judge = "null"
        }
    }
    /* View 切替時の呼び出し */
    override func viewDidDisappear(_ animated: Bool) {
        // camera stop メモリ解放
        session.stopRunning()
        for output in session.outputs {
            // session.removeOutput((output as? AVCaptureOutput)!)
            session.removeOutput(output)
        }

        for input in session.inputs {
            // session.removeInput((input as? AVCaptureInput)!)
            session.removeInput(input)
        }

        session = nil
        camera = nil
    }

    /* 画像表示に関する設定 */
    func setupDisplay(){
        //スクリーンの幅
        //1920 1080
        //812 375
        let screenWidth = cameraView.bounds.width;
        //スクリーンの高さ
        let screenHeight = cameraView.bounds.width * (1920/1080);
```

```
        // プレビュー用のビューを生成
        imageView = UIImageView()
        firstView = UIImageView()
        secondView = UIImageView()
        imageView.frame = CGRect(x: 0, y: 0, width: screenWidth, height:
            screenHeight)
        firstView.frame = CGRect(x: screenWidth/4, y: (cameraView.bounds.height -
screenHeight)/2 , width: screenWidth/2, height:
            screenHeight/2)
        secondView.frame = CGRect(x: screenWidth/4, y: cameraView.bounds.height/2,
width: screenWidth/2, height:
            screenHeight/2)
    }

    /* カメラ入力、ビデオ出力とセッションの接続 */
    func setupCamera(){
        // AVCaptureSession: キャプチャに関する入力と出力の管理
        session = AVCaptureSession()

        // sessionPreset: キャプチャ・クオリティの設定
        // session.sessionPreset = AVCaptureSessionPresetHigh
        session.sessionPreset = AVCaptureSession.Preset.high

        // 背面・前面カメラの選択 iOS10 での変更
        camera =

AVCaptureDevice.default(AVCaptureDevice.DeviceType.builtInWideAngleCamera,
                                        for: AVMediaType.video, position: .back)

        // カメラからの入力データ
        do {
            input = try AVCaptureDeviceInput(device: camera) as
AVCaptureDeviceInput
        } catch let error as NSError {
```

```
            print(error)
        }
        // 入力をセッションに追加
        if(session.canAddInput(input)) {
            session.addInput(input)
        }

        // AVCaptureVideoDataOutput:動画フレームデータを出力に設定
        output = AVCaptureVideoDataOutput()
        // 出力をセッションに追加
        if(session.canAddOutput(output)) {
            session.addOutput(output)
        }

        // ピクセルフォーマットを 32bit BGR + A とする
        output.videoSettings = [kCVPixelBufferPixelFormatTypeKey as AnyHashable
as!
            String : Int(kCVPixelFormatType_32BGRA)]

        // フレームをキャプチャするためのサブスレッド用のシリアルキューを用意
        output.setSampleBufferDelegate(self, queue: DispatchQueue.main)
        output.alwaysDiscardsLateVideoFrames = true



        session.startRunning()

        // device をロックして設定
        do {
            try camera.lockForConfiguration()
            // フレームレート
            camera.activeVideoMinFrameDuration = CMTimeMake(1, 30)
            camera.unlockForConfiguration()
        } catch _ {
        }
    }
```

```
    /* ビデオキャプチャ毎に呼ばれる処理 */
    func captureOutput(_ captureOutput: AVCaptureOutput, didOutput sampleBuffer:
        CMSampleBuffer, from connection: AVCaptureConnection) {
        // キャプチャした sampleBuffer から UIImage を作成
        let image:UIImage = self.captureImage(sampleBuffer)
        pixelBuffer = CMSampleBufferGetImageBuffer(sampleBuffer)

        //
            self.imageView.image = image
            self.firstView.image = image
            self.secondView.image = image
            // UIImageView をビューに追加
            self.cameraView.addSubview(self.firstView)
            self.cameraView.addSubview(self.secondView)


        ////////////////////////////////
//          result = result.components(separatedBy: ":").first!
//          judge = (result == "Penguin")
//          print(result)
//          message.text = result

    }
    /* カメラからの画像キャプチャ */
    func captureImage(_ sampleBuffer:CMSampleBuffer) -> UIImage{

        // Sampling Buffer から画像を取得
        let                    imageBuffer:CVImageBuffer                    =
CMSampleBufferGetImageBuffer(sampleBuffer)!

        // pixel buffer のベースアドレスをロック
        CVPixelBufferLockBaseAddress(imageBuffer,
CVPixelBufferLockFlags(rawValue:
            CVOptionFlags(0)))
```

```
        let baseAddress:UnsafeMutableRawPointer =
            CVPixelBufferGetBaseAddressOfPlane(imageBuffer, 0)!

        let bytesPerRow:Int = CVPixelBufferGetBytesPerRow(imageBuffer)
        let width:Int = CVPixelBufferGetWidth(imageBuffer)
        let height:Int = CVPixelBufferGetHeight(imageBuffer)

        // 色空間
        let colorSpace:CGColorSpace = CGColorSpaceCreateDeviceRGB()

        //let bitsPerCompornent:Int = 8
        let newContext:CGContext = CGContext(data: baseAddress, width: width, height:
            height, bitsPerComponent: 8, bytesPerRow: bytesPerRow, space: colorSpace, bitmapInfo:

CGImageAlphaInfo.premultipliedFirst.rawValue|CGBitmapInfo.byteOrder32Little.rawValue)!

        let imageRef:CGImage = newContext.makeImage()!
        let resultImage = UIImage(cgImage: imageRef, scale: 1.0, orientation:
            UIImageOrientation.right)
        return resultImage
    }


    /* メモリ不足時の呼び出し */
    override func didReceiveMemoryWarning() {
        super.didReceiveMemoryWarning()
        // Dispose of any resources that can be recreated.
    }

}
```