Learning to automatically spectate games for Esports using object detection mechanism

Ho-Taek Joo[a,1] (hotaek87@gm.gist.ac.kr), Sung-Ha Lee[b,1] (shlee0414@gm.gist.ac.kr), Cheong-mok Bae[a] (cmbae0307@gm.gist.ac.kr), Kyung-Joong Kim[a,b] (kjkim@gist.ac.kr)

[a] School of Integrated Technology, Gwangju Institute of Science and Technology, 123 Cheomdangwagi-ro, Buk-gu, Gwangju, 61005, South Korea
[b] AI Graduate School, Gwangju Institute of Science and Technology, 123 Cheomdangwagi-ro, Buk-gu, Gwangju, 61005, South Korea
[1] HT Joo and SH Lee contributed equally to this work.

Corresponding Author:
Kyung-Joong Kim
School of Integrated Technology, Gwangju Institute of Science and Technology, 123 Cheomdangwagi-ro, Buk-gu, Gwangju, 61005, South Korea
Tel: (062) 715-5345
Email: kjkim@gist.ac.kr

Learning to automatically spectate games for Esports using object detection mechanism

Ho-Taek Joo[a,1], Sung-Ha Lee[b,1], Cheong-mok Bae[a], Kyung-Joong Kim[a,b,*]

[a]School of Integrated Technology, Gwangju Institute of Science and Technology, 123 Cheomdangwagi-ro, Buk-gu, Gwangju, 61005, South Korea
[b]AI Graduate School, Gwangju Institute of Science and Technology, 123 Cheomdangwagi-ro, Buk-gu, Gwangju, 61005, South Korea

Abstract

Human game observers, who control in-game cameras and provide viewers with an engaging experience, are a vital part of electronic sports (Esports) which has emerged as a rapidly growing industry in recent years. However, such a professional human observer poses several problems. For example, they are prone to missing events occurring concurrently across the map. Further, human game observers are difficult to afford when only a small number of spectators are spectating the game. Consequently, various methods to create automatic observers have been explored, and these methods are based on defining in-game events and focus on defined events. However, these event-based methods necessitate detailed predefined events, demanding high domain knowledge when developing. Additionally, these methods cannot show scenes that contain undefined events. In this paper, we propose a method to overcome these problems by utilizing multiple human observational data and an object detection method, Mask R-CNN, in a real-time strategy game (e.g., StarCraft).By learning from human observational data, our method can observe scenes that are not defined as events in advance. The proposed model utilizes an object detection mechanism to find the area where the human spectator is interested. We consider the pattern of the two-dimensional spatial area that the spectator is looking at as the object to find, rather than treating a single unit or building as an object which is conventionally done in object detection. Consequently, we show that our automatic observer outperforms both current rule-based methods and human observers. The game observation video that compares our method and the rule-based method is available at https://www.youtube.com/watch?v=61JIfSrLHVk.

Keywords:
Esports, Spectators, Automatic Observer, Mask R-CNN, StarCraft

1. Introduction

Multiple humans called observers control camera movement in Esports to provide an engaging and comprehensive viewing experience for spectators. These observers play an important role in Esports because only a portion of the game screen is shown to spectators, not the entire game screen, and the observer must control the camera to select the most interesting and important part. However, because multiple events occur simultaneously across the map, human observers may miss important events. Furthermore, because such professional human observers require extensive domain knowledge and observing experience, they are difficult to afford in small tournaments (Lie et al. (2019)) or personal spectating. As a result, the demand for automatic observers has grown.

Therefore, various automated observing methods have been developed in pursuit of artificial observers. These methods can be classified into two types: rule-based methods and learning-based methods. Rule-based cameras can be found in games such as Defense of the Ancients 2 (Dota 2), League of Legends (LoL), and StarCraft (Mattsson et al. (2015)). Typically, the camera follows predefined important events across the map in rule-based methods. There have been several studies conducted on learning-based methods. The learning-based methods are usually found in Dota 2. The learning-based methods use various approaches such as predicting team-fight in advance (Tot et al. (2021)), classifying types of predefined events (Phang (2014)), or ranking the future importance of champions and focusing on the champion using predefined event importance value (Lie et al. (2019)).

Thus far, both rule-based and learning-based methods are based on events and must predefine events and the importance of each event, necessitating extensive domain knowledge when developing. Furthermore, they cannot observe events that have not been specifically defined. Furthermore, the significance of events can change depending on the current state of the game, making the observer unreliable. Additionally, as events become more varied, such as in real-time strategy games similar to StarCraft which have significantly more units and strategies than Dota 2, these event-based methods become even less viable because the events become more difficult to define.

In this paper, we first propose an approach to learning observation styles from human game observation data. Unlike event-based approaches mentioned previously, the observation module in our approach can observe the most important scenes from the vast search space without requiring to predefine events in detail, thus covering situations that are ambiguous or difficult

*Corresponding author.
Email addresses: hotaek87@gm.gist.ac.kr (Ho-Taek Joo), shlee0414@gm.gist.ac.kr (Sung-Ha Lee), cmbae0307@gm.gist.ac.kr (Cheong-mok Bae), kjkim@gist.ac.kr (Kyung-Joong Kim)
[1]HT Joo and SH Lee contributed equally to this work.

to define clearly.

To achieve this goal, unlike previous learning-based approaches, our framework used human data to learn human observation patterns in general rather than predefining each event and finding it. We used data from multiple humans watching the same game instead of a single human observation per game. This is because the observer aims to observe the most engaging situation, but a single human does not represent the interest of spectators in general.

Furthermore, we propose the use of object detection mechanisms as a method to learn human observational data. Applying object detection means that the proposed method focuses on two-dimensional spatial regions. Hence, while event-based learning only focused on the actions of in-game objects (e.g., attacking units, creating units in buildings, and upgrading attacks/defenses in buildings), the proposed method focused on patterns within a region that the humans watch. Generally, an object detection (Zhao et al. (2019)) algorithm is the method for determining what targets are in an image and where they are in an image. In the case of a game, the target is objects, such as player units or buildings. However, in this paper, the pattern of the area humans are watching (viewport) is instead considered as the target object. This is because, in game-spectating, the overall patterns of observation, including places without buildings or units, are more important than the presence of specific units or buildings.

In summary, we collect StarCraft in-game human observational data from participants (N=25) who played or watched StarCraft for at least two hours per week. Using the collected data, we proposed a framework for creating an automatic observer using an object detection mechanism and modified an algorithm to better fit our problem. The proposed framework can be applied to various games that can obtain the game state, such as a minimap, and the area viewed by a spectator is part of the game state such as viewport (e.g., StarCraft, Dota 2, LoL).

However, unlike a typical object detection problem, our problem lacks ground truths for each situation because the most interesting region in the game is determined by the individual interests and preferences of human observers. To compensate for this difference, we developed the concept of the region of common interest (ROCI), which is the region in which humans have commonly observed interests. Specifically, ROCI refers to the regions where the viewports of human observers overlap each other. Furthermore, by applying additional loss when the model predicted an area that was not ROCI, the overall performance of the automatic observer was improved. As for the object detection algorithm, Mask R-CNN (He et al. (2020)) was used.

The contributions of our study are as follows:

- This is the first automatic observer in-game that does not require predefining events specifically, therefore requiring far less domain-specific knowledge.
- We propose a novel method for creating an automatic observer using object detection mechanisms and human spectating data, which is being explored for the first time in observer research.
- Furthermore, the proposed method outperforms existing rule-based methods, human, and Mask R-CNN without ROCI by defining and emphasizing ROCI.
- We propose a new evaluation metric for automatic observers and benchmark the various existing observers including rule-based, imitation learning, and human observation.

The remainder of this paper is structured as follows. Section 2 discusses related work and background. Section 2.3 describes the problem in detail, using statements about Automatic Observer and StarCraft. Section 4 presents the proposed method and approaches. Detailed environmental settings and procedures for experiments are stated in Section 5. Section 6 compares quantitative and qualitative evaluations with other approaches mentioned in this study. Finally, Section 7 concludes our work.

2. Background and Related Work

This section introduces the existing automatic observer papers, the object detection algorithm used in our proposed framework and how it was developed, the StarCraft game, the platform we benchmarked, and the artificial intelligence techniques used in this StarCraft game.

2.1. Automatic Observer

An observer is a subject who controls in-game cameras to provide viewers with a spectating experience. The observer displays significant or interesting events to viewers for them to gain a better understanding or engaging experience of the game. Any Esports match requires an observer, and generally, multiple professional observers are required to broadcast Esports.

Several attempts have been made to develop an observer that controls the camera automatically, without human intervention because such professional observers are often prohibitively expensive in personal or small settings, get tired of observing multiple games, and make mistakes. Despite the growing interest in Esports, studies on the spectating experience have received insufficient attention (Lie et al. (2019)). In addition, the number of existing studies on automatic observers is small compared to the role of observers in Esports spectating.

We briefly summarized research on automatic observers of the past in Table 1. Several games include built-in automatic camera systems. Dota 2 and League of Legends, for example, have an automatic observer called a directed camera in spectator mode that follows important events across the map. While not yet implemented in the game, in StarCraft, few rule-based automatic observers have been developed such as AIIDE (Artificial Intelligence and Interactive Digital Entertainment), StarCraft AI Competition[2] and SSCAIT (Student StarCraft AI Tournament[3]) observers. They were developed because of the

[2]https://www.cs.mun.ca/ dchurchill/starcraftaicomp/
[3]https://sscaitournament.com/

Table 1: Previous works about automatic observers.

Works	Game	Event-based	Method	Comments
Phang (2014)	Dota 2	Y	Learning-based	Hybrid of heuristic and machine learning (classification) method
AIIDE	StarCraft	Y	Rule-based	Focus camera on predefined events and priorities
Mattsson et al. (2015)	StarCraft	Y	Rule-based	Focus camera on predefined events and priorities
Lie et al. (2019)	Dota 2	Y	Learning-based	Find and predict important events using AdaRank
Tot et al. (2021)	Dota 2	Y	Learning-based	Predict team-fight using deep neural network
Ours	StarCraft	N	Learning-based	Learn observing patterns of humans using object detection

need to publicly stream bot games as public interest grows. To accomplish this, the camera must follow interesting events to make the stream appealing to viewers. Both AIIDE and SSCAIT examine a game screen based on these predefined interesting game events, as well as the priorities of each defined event. SSCAIT, on the other hand, has more detailed rules than AIIDE. SSCAIT also employs a timer to determine the minimum and maximum time between switching scenes (Mattsson et al. (2015)).

Aside from rule-based methods, several works on automatic observers that use deep learning or machine learning have been investigated. These studies are primarily focused on Dota 2. Existing studies must manually define each event and its event importance hierarchy before using deep learning or machine learning methods to predict or classify a few predefined events such as team fights, farming, and ambushing (Phang (2014); Tot et al. (2021)). However, while these event-driven learning approaches have been widely used in the past, they have limitations in that observers cannot observe events that are not defined, and the importance of events that are defined can change depending on the context of the game. These methods are even less suitable for an environment with a greater variety and number of events such as StarCraft. Because Real-Time Strategy (RTS) games similar to StarCraft have far more units and unit types than Multiplayer Online Battle Arena (MOBA) game Dota 2, events are more diverse, increasing the likelihood of undefined or poorly defined events. Additionally, because these methods are developed in Dota 2, importing to other game platforms can cause problems because the events are largely different and need redefining. The performance of these methods could vary greatly depending on what events and importance are defined.

2.2. Object Detection and Localization

Object detection is a sub-field of computer vision that detects specific objects in digital images and videos. In detail, object detection is a method that combines the recognition of the existence of an object (object recognition), what the object is (object classification), and where the object (object localization) is located in the image. The working principle of object detection is to find a region with an object (this region is called Region of Interest (ROI)) and then feed the ROI into the Convolution Neural Networks (CNNs) to perform object classification and box regression. The representative works of this object detection include regions with Convolutional Neural Networks features (R-CNN) (Girshick et al. (2014)), Fast R-CNN (Girshick (2015)), Faster R-CNN (Ren et al. (2017)), PointRend (Kirillov et al. (2020)), and Mask R-CNN (He et al. (2020)).

R-CNN is the method that combines the selective search (Uijlings et al. (2013)) method widely used in region proposals with CNNs for object detection. This method extracts 2,000 "bounding boxes" (regional proposals) that are likely to have objects in the image. Additionally, after making the extracted "bounding boxes" the same size, pass them to CNNs to classify what kind of object the extracted regions are. Fast R-CNN is an algorithm that greatly accelerates R-CNN computation. R-CNN locates ROI through selective search, and each region is processed by the CNN, resulting in slow computation. Fast R-CNN, on the other hand, sends the entire image to CNN, including the ROI region discovered through selective search. The ROI discovered through selective search is then projected based on the size of the feature map passed through the CNN. Fast R-CNN enabled learning as a single model, including classification and bounding box regression, using this method. The layer prior to ROI pooling in Mask R-CNN is the same as in Faster R-CNN, and then a branch that predicts the mask is added to make segmentation predictable. Mask R-CNN enabled object detection by segmenting in pixels, as opposed to the object detection model that predicted objects only with boxes.

2.3. StarCraft

StarCraft was one of the most commercially successful games on the global market between the 1990s and 2000s. RTS games, such as StarCraft, were considered to be among the most challenging tasks for AI techniques to solve in the last decade. RTS games typically have multiple subtasks, such as gathering resources, training units, constructing buildings, and combating enemies, to achieve a long-term goal, victory. To handle RTS game-playing by AI agents, it should consider these tasks and systemic constraints such as making real-time decisions.

The BroodWar Application Programming Interface[4][5] (BWAPI) is an open-source C++ framework that interacts with StarCraft. It enables the creation of custom-built AI players and their incorporation into StarCraft games. StarCraft has been used as a testbed of game AI research such as Hidden

[4]https://github.com/bwapi/bwapi
[5]https://bwapi.github.io/

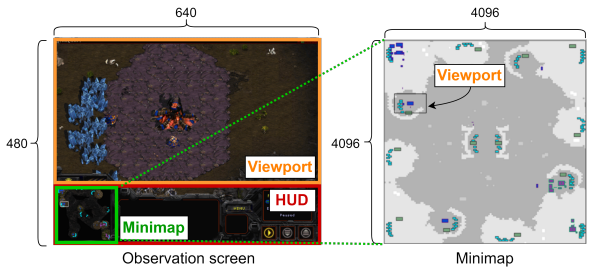


Figure 1: Screen information for StarCraft. The observation screen (left) is the actual screen of StarCraft, and Minimap (right) is an enlarged screen of the picture on the left.

Markov Model (Hostetler et al. (2012)), Tree Search (Cho et al. (2013)), Bayesian Network (Synnaeve & Bessière (2012)), Reinforcement Learning (Wender & Watson (2012)) (Wang et al. (2021)), Imitation Learning (Oh et al. (2017)), and others since the release of BWAPI. Several international StarCraft AI competitions (IEEE Conference on Games, SSCAIT, and AIIDE) have been organized to promote the development of AI players for the RTS games using BWAPI (Čertický et al. (2019)).

3. Problem Definition for Observing the Game

In this section, we defined the viewport problem in automatic observers and introduced what is critical to look at in viewports and how difficult it is to create automatic observers in a Starcraft game comparing Dota2.

3.1. Viewport Problem

The fundamental problem in many games, including popular Esports games similar to Dota 2 or LoL, is that spectators can only see a portion of the screen, known as the viewport, of the entire game. Consequently, selecting a small area where spectators are interested from a large observation space is required. This issue also exists in StarCraft, which our method used as a research platform. More detailed explanations about screens in StarCraft to aid understanding of the platform are as follows.

In Figure 1, we describe how the screens of StarCraft are structured based on example game screens. The "observation screen" refers to the screen that the observer shows to the spectators. This screen is composed of a "viewport" and HUD (Head-Up Display) at the bottom of the screen. The "minimap" shows abstracted information for the entire game state. It assists the observer in grasping the overall situation such as unit movements, battlefield conditions, and fog of war. As the "observation screen," screen shows, observers in games cannot inspect the entire situation at once. Thus, observers should determine the region that is expected to make audiences most likely to be currently interested in the entire game state and display it to audiences via the viewport.

3.2. Finding Region of Interest

The problem we defined is to automatically find the ROI that spectators are most likely to be excited about from the entire screen for each frame of a game replay. Hence, our goal for each frame of the game is to find the most exciting area that spectators want to watch. This can be accomplished by collecting human observing data, which each human viewed in an area that they consider exciting, and learning the pattern of the viewport where spectators watched in each situation. The detailed method can be found in the following section.

3.3. Complexity of Providing an Observation Screen in StarCraft

While our framework can be applied to other games that can get representative states of the game and if the observation area is part of that state of the game, such as Dota 2 or LoL, our framework used StarCraft as a research platform. Providing an observing screen in StarCraft is a difficult task. The majority of existing studies on automatic observers using deep learning have provided Dota 2 observing screens. However, because Dota 2 is a five-vs-five game with each player controlling a single champion, the camera is usually focused on ten players in those studies. While StarCraft is typically played in one-on-one matches, unlike Dota 2, a player can control up to 200 units. Comparing the complexity of StarCraft with units in games similar to LoL or Dota, the number of units in StarCraft is approximately 40 (400/10) times greater than those in Dota 2 and LoL. Additionally, given the various types of units and buildings, the events that can happen in StarCraft are more diverse.

4. Proposed Framework for the Automatic Observer

This section introduces our proposed overall framework. In the overall architecture section, it is explained in general through Figure 2, and detailed explanations are given in the following subsection.

4.1. Overall Architecture

In this paper, we propose a framework for the automatic observer based on human observational data, and Figure 2 depicts the overall architecture of our proposed framework.

The proposed framework first requires replay files containing in-game raw data. Replay files contain a set of actions that the game engine can use to reproduce accurate events (Justesen & Risi (2017)) and are now available in various games (e.g., StarCraft, LoL, Dota 2, Fortnite, etc.).

Then, the process of extracting in-game features is then carried out using the raw data from the collected replay files. In-game features, such as each unit or building in the game, serve to represent the state of the game and are used as inputs for creating automatic observers.

Unlike previous automated observer research, the proposed method involves creating an automatic observer using human observational data rather than expert-level domain knowledge to create rules for in-game camera control. As a result, rather than experts developing rules for automatic observers, our

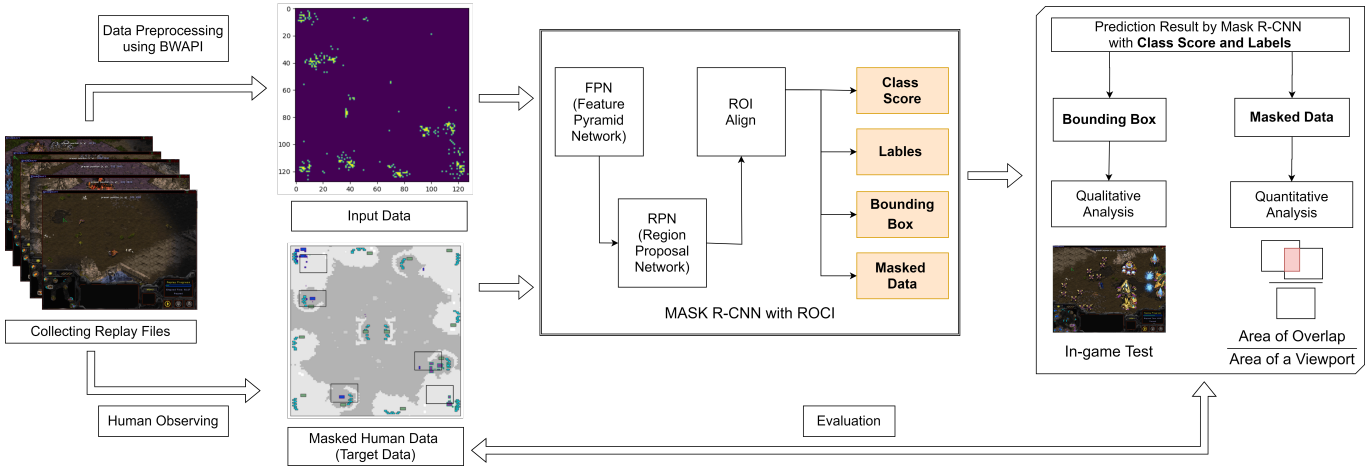


Figure 2: Overall architecture for the proposed framework

method relies on human observational data. These game observation data can be obtained through video data on YouTube[6], Twitch[7], etc., or by recruiting participants to observe the game. The human observational collected data is used as target data to create an automatic observer and must be transformed into part of the overall game state. "Masked Human Data" in Figure 2 provides an example of five people watching the game.

Our proposed method searches for a ROI that corresponds to a subset of the entire region using an object detection mechanism. The in-game features are used as input data of Mask R-CNN, one of the representative algorithms of object mechanism, and the pre-processed human observational data is divided into "labels," "boxes," and "masked data" for Mask R-CNN to be used as target data.

Mask R-CNN model was performed in our framework using these input data and target data, emphasizing the common area of human observational data.

As a result of the model, "class scores," "labels," "boxes," and "masked data" are outputs, and our framework uses these values for evaluation. The evaluation is divided into two parts: a quantitative evaluation designed in the proposed framework and a qualitative evaluation by replaying the actual StarCraft game based on the predictions of automatic observers.

4.2. Input Data and Human Observational Data

With the replay file, the proposed framework does two things. The first is to extract in-game features to create an automatic observer and the second is to collect human observational data from participants. It allows users to watch the game by loading the replay file and controlling the observers by themselves during the replay. **Input data** Selecting input features from in-game raw data in Figure 3 to a neural network model on the game's screen is required to create an automatic observer. When preprocessing in-game raw data for making input features, it must consider whether to include all individual units

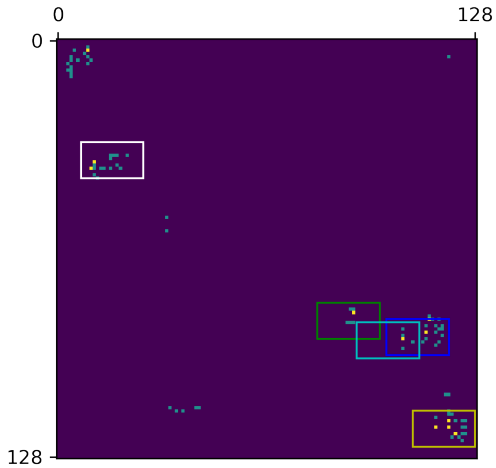


Figure 3: Examples of raw data and human observational data in StarCraft with map size (128×128). Each cell represents the accumulated and normalized number of in-game units or buildings in that location (purple background denotes empty places and dots that are closer to yellow color denote more units/buildings placed) and each rectangular box represents the partial screen of the map multiple participants are viewing.

and buildings on the map or group them based on characteristics such as ground unit and air unit. Because the raw data, such as units and buildings, differs between games, so do the in-game features.

Human observational data is used as target data in our framework, which uses the object detection mechanism. The target data for the object detection model requires labels indicating which objects are present in a given image, masked data corresponding to the locations of the objects, and the bounding box coordinates of each masked data.

Thus, we gathered participants who watched or played StarCraft for at least 2 hours a week and gave them access to the replay files (details are provided in the subsection 5.2.) We had multiple people watching the same replay and recorded which screens they were watching.

In Figure 3, each human views different scenes even though they watch the same frames in the same game because each

[6]https://www.youtube.com/
[7]https://www.twitch.tv/

Table 2: Comparison table between the Conventional Mask R-CNN and the proposed method

	Conventional Mask R-CNN	Proposed Method with Mask R-CNN
Objects	Objects in the given image (e.g., In StarCraft, workers, attack units, and buildings are recognized as objects.)	Recognize the screen that the participants watch as an object (the number of objects in a given image is the number of participants).
Labels	Types of objects given image (e.g., In Starcraft, workers, attacking units, and buildings are each assigned a label. For example, the background is labeled as zero, and workers are labeled as one. Attacking units are labeled as two, and buildings are labeled as three.).	There are two types of labels. Screens watched by the participants are labeled as one, and screens not watched by the participants are labeled as zero.
Masked data	The pixel coordinates corresponding to the object are filled with 1. The rest pixel is 0.	All regions that the participants watch are filled with 1. The rest are 0, not individual objects.

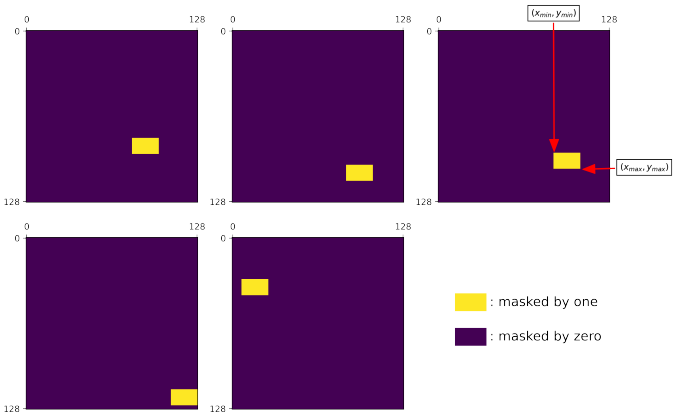


Figure 4: Masked data from human observational data and instance segmentation

human has unique preferences. Therefore, the proposed method aims to find scenes that Esports audiences can enjoy by using the collected data of participants to identify consistent common points from the game screen that multiple participants watch.

The main advantage of utilizing human observational data is by learning from the human spectating data; defining events and the importance of each event is no longer necessary. Therefore, generally less domain knowledge is required than in previous event-based methods. Additionally, the observer can focus on the interesting situation even if the corresponding event is not defined.

4.3. Finding Common Interests of Humans with Mask R-CNN

4.3.1. Object Detection Mechanism

The role of the observer is to find specific ROI for the spectators across the entire map. This process of finding a viewport, which is a portion of the entire screen, is similar to the object detection mechanism's principle of determining what objects are in the image and where they are located in the image. Similarly, in the object detection algorithm, the region where the object exists is ROI in the image and is a part of the entire image. Thus, in the object detection algorithm, finding the ROI corresponding to the mask in the image is analogous to finding the ROI corresponding to a portion of the entire StarCraft screen.

However, while the existing object detection algorithm detects individual objects, our method identifies a region with a pattern formed by in-game objects. For example, there are various patterns of objects gathered together in the rectangular regions observed by humans in Figure 3. Therefore, the area that the observer should look for is not each object, but the pattern that the objects are forming such as what objects (player's units and buildings) are doing, where objects move, how many objects are grouped, and where objects are located on the entire map (one player's units are in another player's base).

4.3.2. Finding Region of Interest with Mask R-CNN

As mentioned in the previous subsection 4.3.1, because our handling problem detects a set of objects, not each object, our method particularly adjusted target data that consist of the **masked data**, **labels**, and **bounding box coordinates** used in the Mask R-CNN such as Table 2.

For masked data, the human observational data in Figure 3 are masked as in Figure 4. In this figure, the observed area is marked with one, and the non-watched area is marked with zero. Additionally, for example, when there are data of five spectators, as shown in Figure 3, the masked data is processed by dividing it into five channels, as shown in Figure 4. The existence of masked data is advantageous because masked data allow the model to accurately express the area containing an object.

Labeling in the object detection algorithm refers to determining which object the masked images are. Furthermore, in our method, each masked data by human observation in Figure 4 was labeled as the same type of object, as presented in Table 2, such as (1, 1, 1, 1, 1). This labeling representation is called instance segmentation (Hafiz & Bhat (2020)).

There are two types of segmentation methods in computer vision: semantic segmentation and instance segmentation. Semantic segmentation is a technique for treating multiple objects of the same type with the same category as if they were one entity. When multiple objects of the same type are in an image, each object is masked differently as an individual object in instance segmentation.

On the other hand, in the case of semantic segmentation, multiple objects of the same type are all displayed on the same screen as one class. The problem of choosing an observing re-

gion is not to represent all the different regions of interest, but to adopt one ROI with the highest priority. Therefore, our proposed method employs instance segmentation and can output one partial screen for the entire game state by using instance segmentation.

Furthermore, when the trained model predicts multiple screens for a single partial screen, the screen with the highest-class score can be chosen. Because Mask R-CNN is one of the popular object detection algorithms among the instance segmentation algorithms, we used Mask R-CNN as an object detection algorithm for our method.

It is applied to box coordinates similarly as the existing Mask R-CNN. The box $(x_{min}, y_{min}, x_{max}, y_{max})$ representing the mask coordinates is also shown in Figure 4.

Loss Function The loss functions of our method with Mask R-CNN are as follows.

$$\mathcal{L}_{total} = \mathcal{L}_{mask} + \mathcal{L}_{cls} + \mathcal{L}_{bbox} + \mathcal{L}_{ROCI} \tag{1}$$

where the mask loss $\mathcal{L}_{mask}$, the classification loss $\mathcal{L}_{cls}$, and the box loss $\mathcal{L}_{box}$ are inherited from Mask R-CNN.

The first following loss function of the mask branch $\mathcal{L}_{mask}$ (2) applied average binary cross-entropy (Ruby & Yendapalli (2020)).

$$\mathcal{L}_{mask} = -\sum_{i=1}^{K} (y_i \log(p_i) + (1 - y_i) \log(1 - p_i)) \tag{2}$$

where y denotes the label for the masked data (in 128×128 size. The area observed by the participant is filled with 1, with the remaining 0, as shown in Figure 4). p denotes the predicted probability for the point for all K points. K denotes (128×128) the entire pixel points.

Second, the loss function of classification branch $\mathcal{L}_{cls}$ (3) is applied to cross-entropy. Mask R-CNN solves multi-class classification problems using cross-entropy by assigning class labels based on the types of objects defined in the image dataset. However, because the screen that the participants are watching is labeled as one in our setting for an automatic observer, as presented in Table 2, the multi-class loss function in our method modifies the binary classification problem that determines whether there is an area to watch in a given image.

$$\mathcal{L}_{cls} = -\sum_{i=1}^{N} \sum_{j=1}^{M} y_{i,j} \log(o_{i,j}) \tag{3}$$

N denotes the number of participants, and M denotes the number of class categories. For our setting, M is one. Additionally, o represents the probability that there is an area for the participants to watch in a given image.

The loss function of bounding-box $\mathcal{L}_{bbox}$ (4) applied smooth $L1$ loss[8]:

$$L_{bbox} = \begin{cases} 0.5(z - \hat{z})^2 & if\ |(z - \hat{z})| < 1 \\ |z - \hat{z}| - 0.5 & otherwise \end{cases} \tag{4}$$

[8]https://pytorch.org/docs/stable/generated/torch.nn.SmoothL1Loss.html

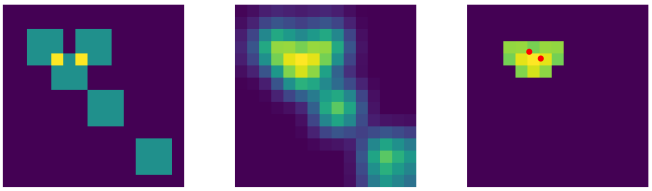


Figure 5: ROCI: applying Gaussian filter to masked image and finding local maxima from the processed image

The box loss represents the regression problem that attempts to fit four coordinates $(x_{min}, x_{max}, y_{min}, and y_{max})$. z denotes four x and y coordinate labels, and $\hat{z}$ denotes four x and y coordinates predicted by the model.

Input: Input masked image $\boldsymbol{I}$, Gaussian filter $\boldsymbol{F_W}$ of size $\boldsymbol{w} \times \boldsymbol{h}$

Output: Additional loss function $\mathcal{L}_{ROCI}$

if *overlapping area exists* **then**

  for ***all*** *pixel* $\boldsymbol{I}(x, y)$ **do**

    Define a window $\boldsymbol{W}$ of size $w \times h$ around $\boldsymbol{I}(x,y)$

    Compute the Gaussian filter $\boldsymbol{F}_{\boldsymbol{W}(x,y)}$ at the location (x,y)

    $\boldsymbol{Z}(x,y) = \sum \sum (\boldsymbol{F}_{\boldsymbol{W}(x,y)} * \boldsymbol{W}(x,y))$

  end

  Remove all elements inside the $\boldsymbol{Z}(x,y)$ that are less than a threshold δ

  $\boldsymbol{Z}_{max}(x,y)$ = maximum filter applied to $\boldsymbol{Z}(x,y)$

  Peak candidates $\boldsymbol{C}$ = nonzero elements in $\boldsymbol{Z}_{max}(x,y)$

  Sort $\boldsymbol{C}$ by intensities

  Define maximum distance between peaks, D

  for $i = 0, ..., len(\boldsymbol{C})$ **do**

    for $j = 0, ..., len(\boldsymbol{C})$ **do**

      if Minkowski distance between candidates $c_i, c_j < D$ **then** Remove c_j from $\boldsymbol{C}$

    end

  end

  Peaks = $\boldsymbol{C}$ with size of number of observers, n

  Consider each peak as the viewport of human

  Calculate mask, labels, and boxes using the found peaks

  $\mathcal{L}_{ROCI} = \sum_{i=0}^{n} (\mathcal{L}_{mask_i} + \mathcal{L}_{cls_i} + \mathcal{L}_{bbox_i})$

else

  $\mathcal{L}_{ROCI} = 0$

end

$\mathcal{L}_{total} = \mathcal{L}_{mask} + \mathcal{L}_{cls} + \mathcal{L}_{bbox} + \mathcal{L}_{ROCI}$

Algorithm 1: Loss Function for Mask R-CNN with ROCI term

ROCI-Mask R-CNN The goal of the observer is to view events that satisfy as many spectators as possible, so viewing the area where spectators are commonly interested is critical. To emphasize human common interests, we first defined the region of common interest (ROCI), which refers to the area that overlaps with each human observer's viewport.

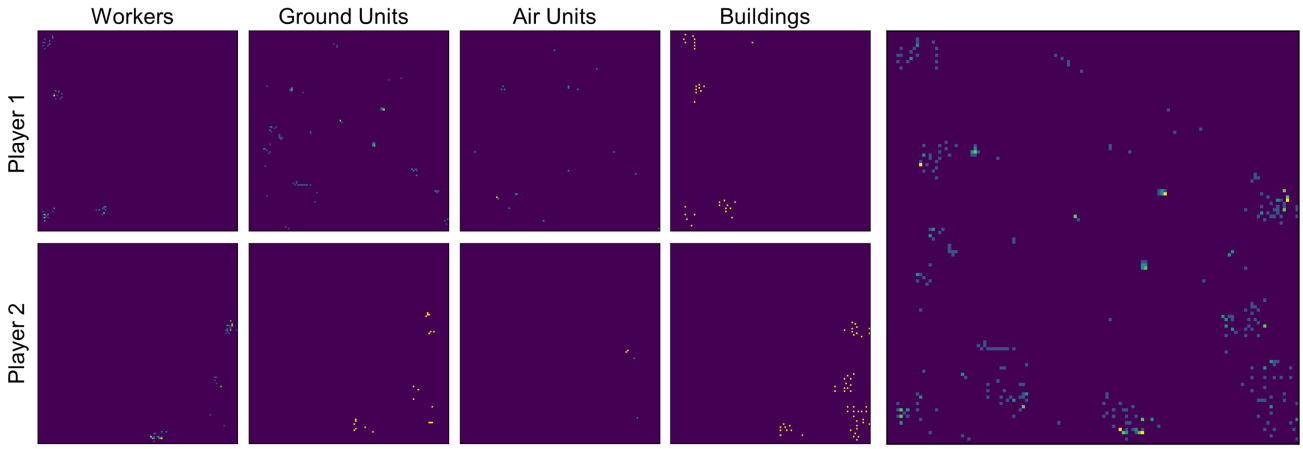


Figure 6: Example for model input. The left four columns describe player-side information. Each player's information consists of workers, ground units, air units, and buildings on that frame. The last figure shows an overlapped channel for all information in that frame, an aspect similar to the minimap information.

The first image of Figure 5 is the masked human observational data, and the overlapped area between masked viewports of humans is highlighted in yellow. To find the point where the views of humans overlap, we applied a Gaussian filter to the masked image, which can be seen in the second image of Figure 5. This is done to smooth the abrupt increase in value so that local maxima are found in the centers of overlapping areas rather than on the edge. Next, we calculated the local maxima of the image to find peaks of the processed masked image. In the last image of Figure 5, the dots are the found local maxima. We also created artificial viewports by using the peaks as center points. The number of these additional viewports changes proportionally to the number of human viewport intersections or ROCI. Next, we used these new viewports as an additional object to add additional loss, $\mathcal{L}_{ROCI}$, to the original Mask R-CNN loss, as shown in Algorithm 1. Therefore, loss increased when the model predicted viewport farther from ROCI, which emphasizes the most interesting region to multiple humans.

4.4. Evaluation Metric

Our evaluation is divided into quantitative and qualitative components. Quantitative evaluation determines how similar our findings are to the general human data. When applied to the case of game observing, one of the evaluation metrics in object detection is the intersection of union (IoU), which is expressed as the following equation. Here, VP is a predicted viewport by model, and VH is a viewport of humans.

$$IoU = \frac{Area\ of\ overlap}{Area\ of\ union} = \frac{|VP \cap VH|}{|VP \cup VH|} \tag{5}$$

However, in the case of observing, this evaluation metric should be modified. As it has been discussed in section 4.3, the matching ground truth for a predicted viewport is not a single viewport of a specific human but viewports of all human observers. Therefore, in the quantitative evaluation, the performance of the observer can be measured using the following equation.

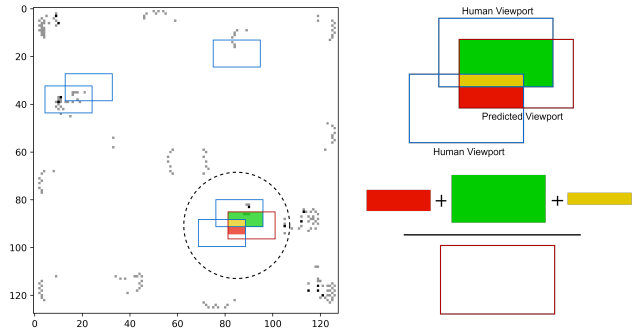


Figure 7: Quantitative Evaluation

$$\begin{aligned} Performance &= \frac{Areas\ of\ overlap}{Area\ of\ a\ viewport} \\ &= \frac{|VP \cap (VH_0 \cup \ ... \ \cup VH_{n-1} \cup VH_n)|}{|VP|} \end{aligned} \tag{6}$$

Here, n is the number of human observers, and VH_n is a viewport of the nth human. If the overlapping area between predicted and human viewports is large, it indicates that the prediction model's observing behavior is similar to general human behaviors. The overlapping area of viewports was summed and divided by the size of a single viewport, as shown in Figure 7. It is then averaged across entire frames of testing replay.

Qualitative evaluation is the process of actually running the StarCraft game to check where it is observing. The qualitative evaluation process confirms how close the proposed method is to human observation in the game. Qualitative evaluation was performed by selecting and comparing specific frames that only humans could observe, which event prediction could not do.

5. Experiments

In this section, we explain the experimental setup in detail, including data collection, data preprocessing, and benchmarking methods.

Table 3: Replay information for training and evaluation. In "Matchup Races", Z, P, and T denote Zerg, Protoss, and Terran, respectively.

Map Name	Fighting Spirit										Jade		Othello	
Matchup Races	Z vs. P						T vs. P		Z vs. T		Z vs. P		Z vs. P	
Starting Location	11	7	1	7	11	1	11	7	11	5	11	5	11	5
# of Replays	15		1		1		1		1		1		1	
Difference	Default (Table 5)		Start Location (Table 6b)				Race (Table 6a)				Map (Table 6c)			
Experiments Evaluation	3-Fold Cross Validation		Generalization Test											

5.1. Input Data from Replay Files

5.1.1. Collecting Replay Files

Among several online sites, game replays are collected. We collected StarCraft replay files from two well-known game communities where various players upload game replays they played: Ygosu[9] and BWReplays[10]. We choose replay files under constraints:

- **APM (Actions Per Minute)** is one of the most commonly used measures in StarCraft to represent a player's level. This measurement can be quantified by collecting player command actions. We gathered replay files from each player whose APM is greater than 250.
- **Race**, StarCraft has three playable races: Protoss, Zerg, and Terran. We used Protoss vs. Zerg, Terran vs. Protoss, and Zerg vs. Terran replay files for training and evaluation.
- **Playing map**, even though there are many existing StarCraft maps, we collected games played in representative competitive map "Fighting Spirit," "Jade," and "Othello." Each map layout is shown in Figure 8.
- **Starting Location** Each player starts at an arbitrary place among the preset places upon maps whenever the game begins. For example, "Fighting Spirit" and "Jade" have four preset start locations on 2, 5, 8, and 11 clock positions on the map; and also "Othello" has four preset start locations but start locations on 1, 4, 7, and 10 clock positions. We filtered replay files with each player's start location. We trained our model with Zerg (11) vs. Protoss (4) (number indicates the start location in the clock position on the map)replay files. The model is then evaluated on other cases for generalization tests.

We collected 15 replay files that met these criteria for the training dataset. We collect additional 6 replay files to consider various cases, such as different starting locations, different race combinations, and different playing maps, for evaluating and testing. Each replay contains about 4,000 frames, and the total amount of collected data is approximately 420,000 (= 21 replays × 4,000 frames × 5 times per replay). Table 3 presents replay information about matchup races, playing maps, and starting locations for training and evaluation steps.

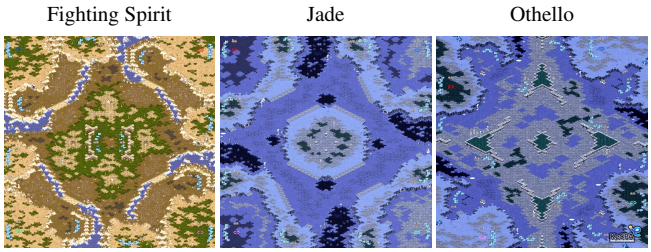


Figure 8: Actual images for used map

5.1.2. Data Extraction

We extract raw data from replay files using BWAPI for every 8 frames (one second ≃ 24 frames.) Raw data include below:

- **Unit Information** involves unit id, type, owner, and coordinate for every unit and building.
- **Vision Information** includes players' union visible regions for each frame. Because many popular RTS games use the fog of war concept, a visible region for each player changes over game time as a result of unit position and movement. We hypothesized that the actual observational screen would be counted as one of the visible regions at the time. Consequently, we expected this information to aid model selection by narrowing the candidate region across the entire map.

5.1.3. Data Preprocessing

In handling StarCraft data via BWAPI, there are two types of coordinate systems: **Position** denotes the distance from the left-top of the map in pixel unit values; and **Tile Position** denotes the divided position value by tile size (default: 32). We chose the tile position coordinate system because computing with pixel unitized values requires a significant amount of computational resources and effort. Furthermore, we classified unit information based on the features of abstracted units rather than all of the detailed unit types. The following are the characteristics of the abstracted units:

[9] https://ygosu.com/
[10] http://bwreplays.com/

- **Worker** involves worker units, which can gather resources and construct buildings. Probe (Protoss), SCV (Terran), and Drone (Zerg) are included. Although worker units can be classified as ground (they can also attack other units), there are no significant differences between them except under certain conditions such as in the early stages of the game or when employing some strategies in which workers make up a large part of the force.

- **Trivial** involves trivial units such as critters (neutral organisms) and larva (Zerg unit that only has the ability to morph into other Zerg units.)

- **Ground** involves all the ground units except worker units and trivial units.

- **Air** involves all the flyable units, including not only having the capability of attack ability but also only transport or support ability.

- **Building** involves all the constructions. Each construction has different abilities, such as training units, increasing the number of units that players can have, improving unit performance, and unlocking abilities. Furthermore, some of them are capable of attacking other units.

We transformed raw data into (T×C×W×H) shaped data. T denotes the number of frames over the replay. C denotes the number of channels. W and H denote map width and height unitized by tile size.

Figure 6 shows the example of 8 input channels used in our proposed framework. The input data include each player's workers, ground units, air units, and buildings.

We trimmed the data every four frames with these processes so that models can process temporal information as model inputs. Hence, models receive (4×32×128×128) shaped data as input (32 is calculated as 4 frames × 8 channels).

5.2. Collecting Human Observational Data

We gathered human observational data to train the automatic observer model. Approximately 40 hours of observation data from 21 games attended by 25 participants are collected. Each replay is viewed by five different people. Fifteen participants watched five replays each, while ten participants watched three replays each. Participants were people between the ages of 20 and 35 who watched or played StarCraft for at least two hours per week. Each participant used a keyboard and mouse to spectate the game replays freely from start to end. While the participants spectate the replay file, the (x, y) coordinates of the location they currently watched are recorded at each frame. Following the completion of data collection, participants were asked to complete questionnaires about the parts they were most interested in.

Even though the five participants watched the same replay, the observing patterns of each participant were different. On average, the viewport of a participant overlapped the mean of 52.3 percent (SD = 4.36) with the viewport of the other four participants. The participants focused on different aspects of the game. While some participants said they focused on building upgrades, others said they focused on expanding resources or fighting. However, based on the percentage of overlapped viewports, while the participants' priorities in observing differ, there are some events that they are all interested in and observe.

Table 4: Target data and prediction data shape

	Target Data	Prediction Data
Masked Data	$[-1, 5 + n, 128, 128]$	[-1, over 10, 128, 128]
Bboxes	$[-1, 5 + n, 4, 1]$	[-1, over 10, 4, 1]
Labels	$[-1, 5 + n, 1]$	[-1, over 10, 1]
Class Scores		[-1, over 10, 1]

5.3. Benchmarks with Our Methods

As presented in Table 3, we first collected 15 replay files and performed K-folds cross-validation ($k = 3$) based on these data. 15 replays were sampled into three sets, two of which were used as training data and the third as testing data. The total number of training epochs was 20. The procedure was repeated three times, and the test set was rotated between them.

5.3.1. Our Method using Object Detection Mechanism

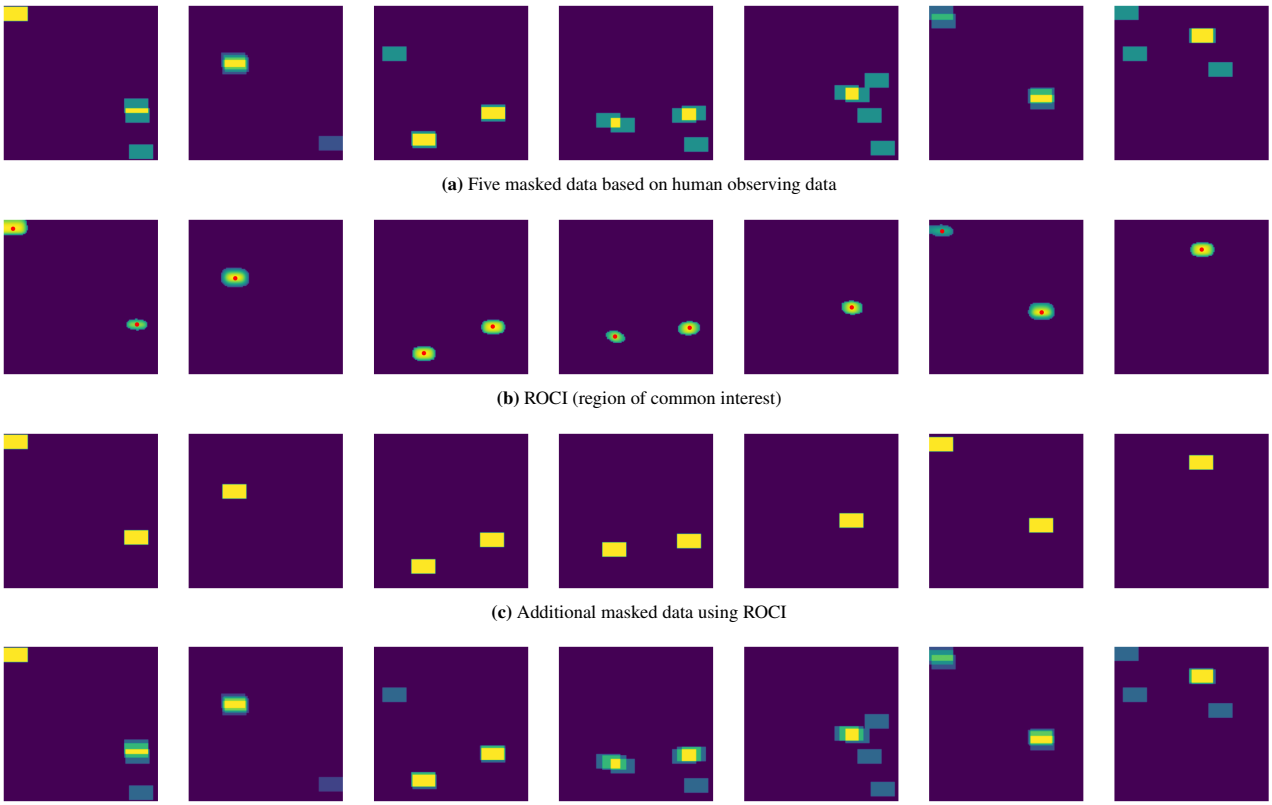
In the previous section, we collected human observational data from five participants per replay file, which is recorded as (x, y) left-top coordinates on the map for each frame. We converted these data into target data that consist of **masked data**, **bounding box coordinates** (bboxes), and **labels**, as presented in Table 4.

Masked data are one observing screen corresponding to (20×12) size in the total map size of (128×128). (128×128) is filled with zeros, without the area corresponding to (20×12) that is filled with one. There are five participants per replay file, so there are five masked data per frame. Therefore, the total size of masked data is $(-1\times (5 + \boldsymbol{n}) \times 128\times 128)$. Here, "-1" means batch size, "$\boldsymbol{n}$" means additional masked data using ROCI, as shown in Figure 9c. Additionally, Figure 9d shows the total masked data with ROCI ("***5+n***"). These masked data indicate the ROI on the entire map by participants.

Bounding box coordinates are tangentially related to the masked data such as Figure 4. Bounding box coordinates indicate $(x_{min}, y_{min}, x_{max}, y_{max})$ coordinate values corresponding to ROI on the map. The "x" coordinates of these boxes range from 0 to 108 (128-20), "y" coordinates of these boxes range from 0 to 116 (128-12); this is because the size of the masked data is (20×12). We experiment with our method using Mask R-CNN under three conditions as follows.

- **Target data with a single human data** We used training data that only included observations from one person per game. One of the five participants was chosen at random. Labels are fixed in size (1).

- **Target data with five human data** We used training data that contains five different human observations in the same game. Labels are fixed in size $(1, 1, 1, 1, 1)$.



(a) Five masked data based on human observing data

(b) ROCI (region of common interest)

(c) Additional masked data using ROCI

(d) Total masked data with ROCI

Figure 9: Masked Data used in Our Method

- **ROCI-Mask R-CNN** At each frame, we calculated the location of a viewport-sized region where human viewports intersected with other humans using Algorithm 1. The threshold value δ used is 1.1, which was determined experimentally. We selected the threshold value between 1, which was the value of the area where the single human observer was watching in a masked image, and 2, which was the value of the area two human observers were concurrently watching. The final threshold value, 1.1, was experimentally determined by testing the value from 1 to 2 with intervals of 0.1. As shown in Figure 9b, the red dots indicate the detected center of overlapping points between humans.

  Then, we created masked data with that viewport-sized region, as shown in Figure 9c. Finally, we added additional masked data on Figure 9b, as shown in Figure 9d. In this case, the size of the labels changes depending on the number of detected ROCI. If the number of detected ROCI increases, the size of the label proportional is greater.

After experimenting with our framework, our framework predicts four values: "masked data," "bounding boxes," "labels," and "class scores," as presented in Table 4.

Masked data display an area of approximately (20×12) in size of (128×128). **Bounding boxes** predict values for $(x_{min}, y_{min}, x_{max}, and y_{max})$, and **labels** output a matrix of ones similar to (1, 1, 1, 1, 1). The number of labels indicates the number of ROI predicted by the model. Additionally, **class scores** show the probability that labels are predicted correctly.

Usually, the trained model by our framework predicts that there are more than 10 objects (ROIs) in one image frame, as presented in Table 4. Because the trained model predicted that there are more than ten ROIs on average in a single frame, the most accurate region should be chosen. Among the predicted ROIs, our framework selects a region with the most probability of class scores. The pairs of "masked data," "bounding boxes," "labels," and "label scores" are used for evaluation in the following section 6.

5.3.2. Behavior Cloning for Automatic Observer

To imitate human observational data, one method we used is **behavior cloning (BC)**, which is one of the primary methods for imitation learning (Torabi et al. (2018)). First, the agent receives training data, which consists of expert-performed states and actions. The expert policy is then replicated by training classifiers or regressors. (Ross & Bagnell (2010)) In our case, a regressor is used for training (Bain & Sammut (1999)). The Mean Squared Error (MSE) loss function is used, as shown in the following equation.

$$\sum_{i=1}^{D}(x_i - y_i)^2 \tag{7}$$

Table 5: Evaluation: Percentage of the overlapped viewport with fixed races, starting location, and map.

Game Progression	BC	SSCAIT	AIIDE	Human	Mask R-CNN w/ one human	Mask R-CNN w/ five humans	Mask R-CNN w/ ROCI
1/4	0.075	0.485	0.557	0.549	0.593	0.624	**0.641**
2/4	0.059	0.503	0.513	0.516	0.550	0.584	**0.595**
3/4	0.058	0.511	0.501	0.509	0.535	0.574	**0.586**
4/4	0.061	0.524	0.491	0.516	0.530	0.569	**0.588**

Here, *i* denotes frame. *x* denotes predicted camera location, and *y* denotes human camera location.

5.3.3. Rule-based Observer in StarCraft

We used modified observer modules that were used in StarCraft AI competitions, SSCAIT[11][12] and AIIDE StarCraft AI Competition[13], to compare our learning-based methods. AIIDE and SSCAIT observers are both rule-based automatic StarCraft observers. SSCAIT was created to control the in-game camera using game events, priorities, and timers. The camera focuses on predefined events such as an attack on a unit with the highest predefined priority score. The timers are used to determine whether the camera should proceed to the next event, such as a more prioritized event or a new event, or whether it should remain stationary. Furthermore, rather than teleporting, SSCAIT smoothly moves the camera to the desired position. More details for each observer module can be found in (Mattsson et al. (2015)).

6. Result Analysis and Evaluation

In this section, we present and analyze the results of the experiment conducted in the previous section. The result and analysis are divided primarily into quantitative analysis, which shows how much each method is similar to general human data quantitatively, and qualitative analysis, which compares the in-game viewport of each method.

6.1. Quantitative Analysis

We compared seven different observing methods presented in Table 5: Human observer, SSCAIT (rule-based observer), AIIDE (rule-based observer), behavior cloning observer model, Mask R-CNN observer with the single human label, Mask R-CNN observer with multiple human labels, and ROCI-Mask R-CNN. We used an overlapping area between the predicted viewpoint by each method and the viewport of multiple human observers to evaluate how similar the behavior of each method is to the behavior of multiple humans, as shown in Figure 7. A detailed explanation of the evaluation method can be found in section 4.4.

We did not choose a single human viewport to calculate the overlapping area. Instead, we used all five human viewports. This is because each human has different priorities and interests and views different scenes, as it was presented in section 5.2. According to our questionnaires, some human observers valued building upgrades more than others, while others valued battles and unit movements more. However, because humans share approximately 52% of their viewport with other humans on average, there is still significant overlapping of the viewport between humans, implying that humans agree on the importance of certain situations despite having different perspectives on the game. Thus, focusing on these common interests while mimicking human behavior is important because it satisfies more viewers. In this metric, if the resulting viewport predicted such a common area of interest, the viewport is more likely to overlap with humans, resulting in a higher score.

We evaluated each model in four types of test sets. **The first test set** is to evaluate the overall performance of each model based on 3-fold cross-validation using 15 replay files, as presented in Table 3. The first test set is set on the same starting location, matchup races, and map of the training set (Table 5). **The second to last test set (Table 6a, 6b, 6c)** is to evaluate how well it performs in unseen situations (Table 6c), specifically, how well the model generalizes. We conduct a **generalization test** with three models generated based on the first test set through 3-fold cross-validation. The second test set is set in a different starting location from the training set (Table 6a). The third type of test set is set on different matchup races from the training set (Table 6b). Additionally, the last test set is set on a different map from the training set (Table 6c).

The overall result of the first type of test set is presented in Table 5. Because game strategy evolves over time, we evaluated each model based on game progression time: the first quarter of the game, the half of the game, the third quarter of the game, and the entire game. From the start to the end of the game, the overlapping area of ROCI-Mask R-CNN was the largest, followed by the order of five humans Mask R-CNN, one human Mask R-CNN, human, SSCAIT, AIIDE, and BC.

Consequently, the ROCI-Mask R-CNN best observed the areas where it was viewed by most humans with interests. Mask R-CNN with one human label performed worse than Mask R-CNN with five human labels. As each human values the interesting parts of the game state differently, a single human observational data is insufficient to represent the ROI of the public in general. Therefore, Mask R-CNN with multiple human labels was able to find a pattern that is more similar to the general human observation pattern. The ROCI-Mask R-CNN performed the best out of the seven. This is because ROCI-Mask R-CNN additionally considers regions where objects, which are human

[11] https://github.com/Plankton555/SSCAIT-ObserverModule
[12] https://sscaitournament.com/
[13] https://github.com/davechurchill/StarcraftAITournamentManager

Table 6: Generalization Test

(a) Generalization Test: percentage of overlapped viewport with **different races**.

Game Progression	BC	SSCAIT	AIIDE	Human	Mask R-CNN w/ one human	Mask R-CNN w/ five humans	Mask R-CNN w/ ROCI
1/4	0.020	0.562	**0.621**	0.607	0.597	0.612	0.610
2/4	0.015	0.547	0.495	0.559	0.551	**0.584**	0.578
3/4	0.008	0.589	0.490	0.576	0.563	0.600	**0.611**
4/4	0.004	0.581	0.482	0.572	0.569	0.593	**0.615**

(b) Generalization Test: percentage of overlapped viewport with **different starting locations**.

Game Progression	BC	SSCAIT	AIIDE	Human	Mask R-CNN w/ one human	Mask R-CNN w/ five humans	Mask R-CNN w/ ROCI
1/4	0.028	0.526	0.558	0.545	0.585	0.567	**0.609**
2/4	0.031	0.487	0.477	0.493	0.481	0.500	**0.538**
3/4	0.038	0.482	0.459	0.496	0.483	0.498	**0.529**
4/4	0.042	0.508	0.476	0.512	0.508	0.529	**0.559**

(c) Generalization Test: percentage of overlapped viewport with **different map**.

Game Progression	BC	SSCAIT	AIIDE	Human	Mask R-CNN w/ one human	Mask R-CNN w/ five humans	Mask R-CNN w/ ROCI
1/4	0.088	0.510	**0.649**	0.556	0.557	0.628	0.598
2/4	0.111	0.470	0.544	0.515	0.521	**0.565**	0.563
3/4	0.101	0.490	0.514	0.519	0.516	0.545	**0.55**
4/4	0.109	0.502	0.509	0.514	0.519	0.556	**0.581**

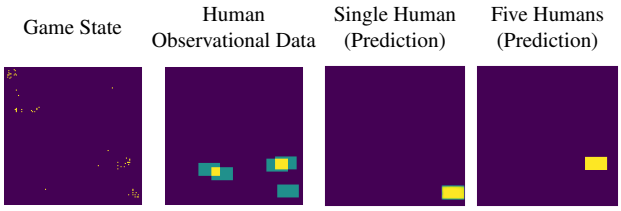


Figure 10: The result of comparing a single human and five humans

observational data, intersect. Therefore, when ROCI is emphasized, it is more suitable for locating the most general human observer.

While both SSCAIT and AIIDE are rule-based methods, they demonstrated significant performance differences. This demonstrates that rule-based methods are not consistent, and their performance is heavily dependent on the person specifying the rules, making them less suitable for automatic observation.

It is worth noting that, despite the fact that the Mask R-CNN with five humans model was trained using human observational data, the overlapping area of the Mask R-CNN with five humans was larger than the human average. This is because Mask R-CNN with five humans was trained with multiple human observers rather than just one. Because humans have different preferences and perspectives and they can make mistakes occasionally, such as not observing important scenes, humans do not watch the same scene as other humans do constantly. In Figure 10, each human views different scenes except for a few overlaps. If trained with a single human data, Mask R-CNN only learns from the view and perspective of a single human. As shown in Figure 10, in the testset, the Mask R-CNN with one human predicts the viewport where only a single human is watching, rather than the area where the humans are generally watching. On the other hand, the Mask R-CNN with five humans learns from multiple views and perspectives of humans. Additionally, its prediction is more likely general to multiple humans, such as in Figure 10, resulting in higher performance than both a human average and Mask R-CNN with a single human. More examples about Figure 10 are in the appendix section. More examples are Figure 16.

Additionally, Mask R-CNN has access to more detailed information about the entire map, such as the layout of air or ground units, whereas humans only have access to more general information such as the minimap and current viewport.

Generalization Test As mentioned in section 5.1.1, we collected 15 replay files with fixed race, location, and map with Zerg (11), Protoss (4), and map (Fighting Spirit). Additionally, our proposed method in the same race, location, and map showed superior performance to the previous study, as presented in Table 4. Moreover, we additionally collected the six replay files in three unseen situations, which are with two unseen maps, two unseen races, and two unseen starting locations, to measure the generalization performance of our model. Additionally, we evaluated our model with new replay files. Tables 6a, 6b, and 6c present the generalization performance results for unseen races, unseen starting locations, and unseen maps, respectively.

In the first quarter of the game, when testing the model on the test set with different races and maps from the training set, the rule-based observer, AIIDE, outperformed the Mask R-CNN with ROCI. Mask R-CNN with five humans performed better in

Figure 11: Result of comparing five humans and ROCI at the beginning of the game

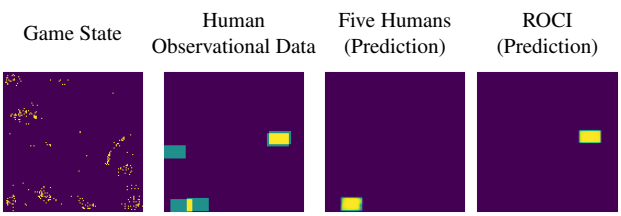


Figure 12: The result of comparing five humans and ROCI at the end of the game

the first half of the game. However, the performance of both AIIDE and Mask R-CNN with five humans dropped dramatically as the game progressed, with AIIDE dropping more dramatically than Mask R-CNN and Mask R-CNN with ROCI showing the best overall performance. The number of players and units is small at the start of the game, and the game is relatively simple. As shown in Figure 11, in the early stage of the game, the human observers mostly view a small number of the same scenes because significant events do not occur much. However, as the game progresses, the game and strategies become more complex, and significant events, such as battles, can occur in multiple locations at the same time. However, in the middle or end stage of the game, in Figure 12, the human observers view different scenes, and only a few human viewports overlap with each other. Therefore, as for the Mask R-CNN with five humans, in the early stage of the game, the performance does not affect much even without additional ROCI masked data because the human viewports already mostly overlap with each other, and the area of human viewports is mostly the same as ROCI. However, as the game progress, the viewports of humans are divided. Additionally, the areas of human viewports are no longer mostly the same as ROCI. Thus, the Mask R-CNN with ROCI shows better performance in the entire game because it highlights commonly interesting areas when the viewports of humans are divided. As for the AIIDE, which has to manually define each event and the importance of each event, in the early stage of the game, it is easier to show high performance because events are few and simple, unlike in the later stage of the game. More examples about Figures 11 and 12 can be found in the appendix section. More examples are Figures 14 and 15.

The result of the generalization test of Mask R-CNN with ROCI shows that our model is flexible and can adapt to different types of situations from the training data.

6.2. Qualitative Analysis

In this section, we compare how each method is applied qualitatively. Although our method performs well and is better deduced in quantitative evaluation when compared to other methods, it needs to be demonstrated through practical proofs in various cases. Therefore, we examine the same test replay file with all methods to see where each method focuses on specific situations such as events or unit tracking. By doing this, it demonstrates how each method performs in practice.

Figure 13 shows how each method practically works out in event cases. Columns indicate each method: Behavior-Cloning; Rule-based methods results that AIIDE and SSCAIT observer module; human's observational results that replay data from a person's log, which is arbitrarily chosen, among test datasets; and our method result.

Figure 13a and Figure 13b show when players first encounter each other using a worker unit and a ground unit, respectively. Both scenarios are critical for each player, and revealing veiled information and specifying the enemy's location as soon as possible can help them plan their build order and playing style. In the first encounter, humans and ours usually focused/tracked where the appropriate position was, whereas behavior-cloning and rule-based did not. While the first encounter between two players is an important situation in the game, the rule-based methods watch the base of the player rather than focusing on the first encounter. In the rule-based method, to detect an event that a unit of each player encounters, the distance between units should be calculated. Because a single player can control up to 200 units, calculating the distance between each unit is inefficient, more so if the number of players increases. Thus, in event-based methods, the event encounter between two players is not defined and fails to observe the important scene, such as the first encounter, unlike our method.

In StarCraft, gathering more abundant resources than the enemy can make advantageous situations by pressing the enemy using plenty of units; thus, interrupting gathering resources for the enemy is an important strategic action. Figure 13c shows the situation where Zealots (Protoss ground units) intrude on Zerg's expanded base and prevent gathering resources. This situation lasts about 30 seconds as Zealots travel around narrow aisles between mineral fields inflicting drones. Except for behavior-cloning, most methods focus on this situation. Rule-based methods, on the other hand, concentrate on other locations where other events occurred recently.

Figure 13d shows that Hydralisks (Zerg ground unit) attack Protoss's defense line. In this situation, rule-based methods did not observe before combat began and only observed when combat started. Humans and our method, on the other hand, predict that the battle will happen and are focused on the battlefield from the moment units are deployed to the frontlines, even when no combat occurs.

Figure 13e shows that Protoss assembles units on the outpost to break down Zerg's pressure. There are no notable events or battles in this situation. Therefore, rule-based methods shift their focus to another location where an event occurred recently. However, humans and ours are focused on the scene where units are assembling to watch a possible future battle.

(a) Probe (Protoss worker unit) first encounters Zerg camp

(b) Zergling (Zerg melee ground unit) found Protoss camp

(c) Zealots (Protoss melee ground units) inhibit Zerg gather resources behind Minerals

(d) Hydralisks (Zerg ranged ground units) press Protoss defense-line

(e) Protoss assemble units to breakdown Zerg's pressure

Figure 13: Example for comparing methods qualitative evaluation

Overall, the **behavior-cloning** tries to catch our situations but failed to find our exact positions or timings, also it gets dramatically worse on the test dataset. The **rule-based methods** try covering predefined events on each script. Its performance depends on how many rules are predefined and how fine they are. If the rule did not cover the specific situation, it fails to observe properly. Furthermore, the rule-based method did not predict future events. Although **humans'** performance relies on their personal interests and preferences, it usually converges in important situations. However, some humans focus on trivial scenes sometimes, even though the majority of human are focusing on other places the interesting events are occurring. **Our method** is overall successful in detecting important situations using human observational data. Some human observers frequently fail to focus on appropriate scenes; however, our method learns from the observational data of multiple human observers, thus observing the scenes that humans will commonly be interested in.

7. Conclusion

We proposed a novel framework of automatically providing an observing screen to the audience for Esports using the object detection mechanism based on human observational data.

The role of the observer in Esports is to provide a screen that the public will find most interesting. Therefore, the proposed framework collects several human observational data representing the public and creates an automatic observer using an object

detection mechanism based on this data. The object detection mechanism is to classify what objects are in the image and discover where they are. Our method treats the observation area of the participants as an object of the object detection mechanism.

The participant's observation area contains many in-game objects. Therefore, treating the participant's observation area as an object is to objectify the various patterns that these in-game objects are making. Thus, our method also includes selecting in-game features to create patterns for these in-game objects.

Moreover, we evaluated our proposed methods, IL, Mask R-CNN, rule-based methods, and the average value of collected human observational data using two evaluation metrics. The first is quantitative evaluation. Using quantitative evaluation, the performance of methods is assessed by noting how similar viewports are to the collected human observational data. The second is qualitative evaluation, which involves entering the predicted coordinates into the StarCraft game and evaluating which screens are displayed under what conditions.

Our proposed method outperformed other methods up to the total playtime in the quantitative evaluation. Moreover, the qualitative evaluation shows that the automatic observer by our method was able to capture scenes where humans are interested, which could not be done by event prediction.

Our main contribution is to propose a method for automatically providing the most exciting scenes, even in complex games similar to StarCraft. Furthermore, we proposed several methods for using human observational data and an evaluation metric based on human observational data. Our framework can also be applied to other games that can represent some of the overall game state, not only StarCraft.

There are several limitations of this study. First, current Esports observers must interact with commentators to provide a more satisfying experience to the spectators. However, the proposed model cannot yet interact with the commentators. The second limitation is that to achieve real-time automatic observation, the help of the development company is needed to obtain the real-time in-game state.

Several future works exist. The first one is to improve the performance of the proposed automatic observer. For example, simple data augmentation techniques, such as translating/rotating, can increase the generalization performance. The second one is to extend our method to other games. This paper only applies the automatic observer to the StarCraft platform. We plan to expand our work to other commercial games, such as Dota 2 or LoL. Moreover, while only the Mask R-CNN was applied in this paper, we plan to use different object detection algorithms and compare their performances.

Acknowledgements

This research was supported by the National Research Foundation of Korea (NRF) funded by the MSIT (2021R1A4A1030075).

References

Bain, M., & Sammut, C. (1999). A framework for behavioural cloning. *Machine Intelligence*, *15*, 103–129.

Cho, H. C., Kim, K. J., & Cho, S. B. (2013). Replay-based strategy prediction and build order adaptation for starcraft ai bots. In *2013 IEEE Conference on Computational Intelligence and Games, CIG 2013*. doi:10.1109/CIG.2013.6633666.

Girshick, R. (2015). Fast r-cnn. In *inproceedings of the IEEE International Conference on Computer Vision* (pp. 1440–1448). doi:10.1109/ICCV.2015.169.

Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 580–587).

Hafiz, A. M., & Bhat, G. M. (2020). A survey on instance segmentation: state of the art. *International Journal of Multimedia Information Retrieval*, *9*, 171–189. doi:10.1007/s13735-020-00195-x.

He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2020). Mask r-cnn. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *42*, 2961–2969. doi:10.1109/TPAMI.2018.2844175.

Hostetler, J., Dereszynski, E., Dietterich, T., & Fern, A. (2012). Inferring strategies from limited reconnaissance in real-time strategy games. In *Uncertainty in Artificial Intelligence - inproceedings of the 28th Conference, UAI 2012* (pp. 367–376).

Justesen, N., & Risi, S. (2017). Learning macromanagement in starcraft from replays using deep learning. doi:10.1109/CIG.2017.8080430.

Kirillov, A., Wu, Y., He, K., & Girshick, R. (2020). Pointrend: Image segmentation as rendering. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 9799–9808).

Lie, H., Lukas, D., Liebig, J., & Nayak, R. (2019). A novel learning-to-rank method for automated camera movement control in e-sports spectating. *Communications in Computer and Information Science*, *996*, 149–160. URL: https://link.springer.com/chapter/10.1007/978-981-13-6661-1_12. doi:10.1007/978-981-13-6661-1_12/COVER/.

Mattsson, B. P., Vajda, T., & Čertickỳ, M. (2015). Automatic observer script for starcraft: Brood war bot games (technical report). *arXiv preprint arXiv:1505.00278*, . URL: https://arxiv.org/abs/1505.00278v1. doi:10.48550/arxiv.1505.00278.

Oh, I.-S., Cho, H., & Kim, K.-J. (2017). Playing real-time strategy games by imitating human players' micromanagement skills based on spatial analysis. *Expert Systems with Applications*, *71*, 192–205. URL: https://www.sciencedirect.com/science/article/pii/S0957417416306613. doi:https://doi.org/10.1016/j.eswa.2016.11.026.

Phang, D. W.-S. (2014). *Intelligent Camera Control in Game Replays*. Ms thesis.

Ren, S., He, K., Girshick, R., & Sun, J. (2017). Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *39*. doi:10.1109/TPAMI.2016.2577031.

Ross, S., & Bagnell, D. (2010). Efficient reductions for imitation learning. In Y. W. Teh, & M. Titterington (Eds.), *inproceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics* (pp. 661–668). PMLR volume 9. URL: https://inproceedings.mlr.press/v9/ross10a.html.

Ruby, U., & Yendapalli, V. (2020). Binary cross entropy with deep learning technique for image classification. *Int. J. Adv. Trends Comput. Sci. Eng*, *9*.

Synnaeve, G., & Bessière, P. (2012). Special tactics: a bayesian approach to tactical decision-making. In *IEEE Conference on Computational Intelligence and Games (CIG)* (pp. 978–979). URL: https://hal.archives-ouvertes.fr/hal-00752841.

Torabi, F., Warnell, G., & Stone, P. (2018). Behavioral cloning from observation. In *IJCAI International Joint Conference on Artificial Intelligence*. volume 2018-July. doi:10.24963/ijcai.2018/687.

Tot, M., Conserva, M., Chitayat, A. P., Kokkinakis, A., Patra, S., Demediuk, S., Munoz, A. C., Olarewaju, O., Ursu, M., Kirmann, B., Hook, J., Block, F., Drachen, A., & Perez-Liebana, D. (2021). What are you looking at? team fight prediction through player camera. In *IEEE Conference on Computatonal Intelligence and Games, CIG*. volume 2021-August. doi:10.1109/CoG52621.2021.9619038.

Uijlings, J. R., Sande, K. E. V. D., Gevers, T., & Smeulders, A. W. (2013). Selective search for object recognition. *International Journal of Computer Vision*, *104*. doi:10.1007/s11263-013-0620-5.

Wang, X., Song, J., Qi, P., Peng, P., Tang, Z., Zhang, W., Li, W., Pi, X., He, J., Gao, C. et al. (2021). Scc: an efficient deep reinforcement learning agent mastering the game of starcraft ii. In *International Conference on Machine Learning* (pp. 10905–10915).

Wender, S., & Watson, I. (2012). Applying reinforcement learning to small scale combat in the real-time strategy game starcraft:broodwar. In *IEEE Conference on Computational Intelligence and Games, CIG 2012*. doi:10.1109/CIG.2012.6374183.

Zhao, Z. Q., Zheng, P., Xu, S. T., & Wu, X. (2019). Object detection with deep learning: A review. *IEEE Transactions on Neural Networks and Learning Systems*, *30*. doi:10.1109/TNNLS.2018.2876865.

Čertický, M., Churchill, D., Kim, K.-J., Čertický, M., & Kelly, R. (2019). Starcraft ai competitions, bots, and tournament manager software. *IEEE Transactions on Games*, *11*, 227–237. doi:10.1109/TG.2018.2883499.

Appendix A.

The hyper-parameters used in the proposed method are as follows in Table 7. Our basic parameter settings follow the site[14]. We conducted additional experiments according to the window size, epochs, and learning rate in hyper-parameters. The experimental results are presented in Appendix D.

Table 7: Hyper-parameters setup for Mask R-CNN

Hyper-parameters	Value
Backbone	Resnet50
Input image size	800×800
Window size	[1, 2, 4, 8]
Batch size	4
Epochs	[5, 10, 20, 30]
Network optimizer	SGD
Learning momentum	0.8
Learning rate	[0.00025 ,0.005, 0.01]
Weight decay	0.0005
Scale of anchor	(32, 64, 128, 256, 512)
Aspect ratio of anchor	(0.5, 1, 2)
RPN NMS threshold	0.7

Appendix B.

In our paper, we used Algorithm 1 to find multiple points where humans are commonly interested. We have experimented and included the result of when only a single intersection that has the largest value is used as ROCI. As shown in Table 10, the results show lower performance than the proposed method.

Appendix C.

Rather than using the percentage of overlapped viewport area, the other evaluation metrics exist. We included an alternative quantitative evaluation to further validate our result. One feasible evaluation is using a specific threshold value to determine whether each method predicted the object, in our case the

[14] https://pytorch.org/tutorials/intermediate/torchvision_tutorial.html

Table 8: The percentage of correct prediction with intersection threshold=0.3

Game Progression	BC	SSCAIT	AIIDE	Human	Mask R-CNN w/ one human	Mask R-CNN w/ five humans	Mask R-CNN w/ ROCI
1/4	0.104	0.596	0.664	0.665	0.686	0.725	**0.743**
2/4	0.081	0.630	0.628	0.634	0.651	0.695	**0.702**
3/4	0.080	0.644	0.618	0.633	0.641	0.689	**0.701**
4/4	0.084	0.660	0.607	0.639	0.638	0.685	**0.706**

Table 9: The percentage of correct prediction with intersection threshold=0.5

Game Progression	BC	SSCAIT	AIIDE	Human	Mask R-CNN w/ one human	Mask R-CNN w/ five humans	Mask R-CNN w/ ROCI
1/4	0.059	0.538	0.631	0.619	0.658	0.695	**0.711**
2/4	0.044	0.564	0.588	0.581	0.612	0.656	**0.661**
3/4	0.044	0.575	0.573	0.576	0.599	0.645	**0.653**
4/4	0.046	0.590	0.561	0.580	0.594	0.640	**0.657**

Table 10: The comparison of the percentage of the overlapped viewports between Mask R-CNN with ROCI and Mask R-CNN with single ROCI that had the largest value.

Game Progression	Mask R-CNN /w ROCI	Mask R-CNN /w single ROCI
1/4	**0.641**	0.639
2/4	**0.595**	0.588
3/4	**0.586**	0.581
4/4	**0.588**	0.575

human viewport, correctly or not to calculate the accuracy. For example, in the case when the threshold is 0.5, if the percentage of the overlapped area between the human viewport and the predicted viewport exceeds 0.5, the prediction is correct.

We included Table 8 and 9, which show the results of the alternative evaluation method when the threshold is 0.3 and 0.5. As shown in the table, similar to the original evaluation in our paper, Mask R-CNN with ROCI shows the highest performance in both thresholds 0.3 and 0.5.

Appendix D.

As mentioned Appendix A, we have trained models using different epochs [5, 10, 20, 30], different window sizes [1, 2, 4], and different learning rates [0.00025, 0.005, 0.01]. The results are presented in Tables 11, 12, and 13. Here, the models are not trained 3-fold and are only trained once.

Table 11: The results of Mask R-CNN with ROCI at learning rate 0.01, 0.005, and 0.00025.

	Learning Rate		
Game Progression	0.01	0.005	0.00025
1/4	0.629	0.635	0.714
2/4	0.565	0.564	0.631
3/4	0.547	0.562	0.617
4/4	0.552	0.569	0.644

Table 12: The results of Mask R-CNN with ROCI at epoch 5, 10, 20, and 30.

	Epoch			
Game Progression	5	10	20	30
1/4	0.662	0.648	0.648	0.648
2/4	0.583	0.572	0.571	0.571
3/4	0.581	0.554	0.554	0.554
4/4	0.587	0.550	0.550	0.550

Table 13: The results of Mask R-CNN with ROCI at windowsize 1, 2, 4, and 8.

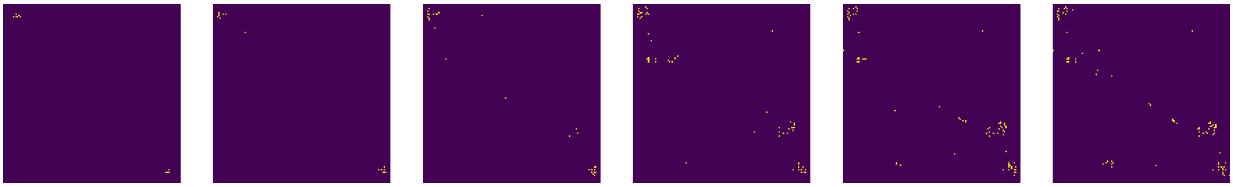
	Windowsize			
Game Progression	1	2	4	8
1/4	0.610	0.621	0.647	**0.651**
2/4	0.583	0.614	0.618	**0.621**
3/4	0.570	0.588	**0.603**	0.597
4/4	0.560	0.593	**0.600**	0.598

Appendix E.

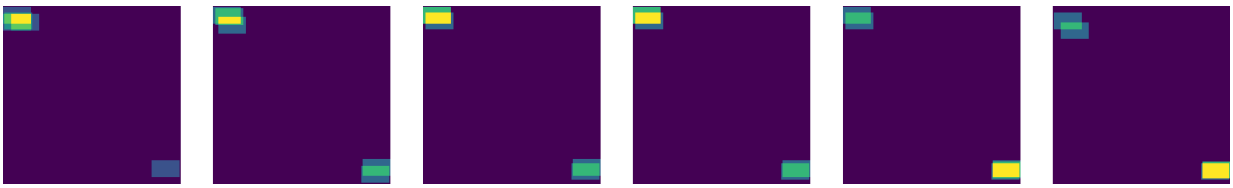
As mentioned in section 6.1, we additionally show the experimental results at the beginning of the game and end of the game, as shown in Figures 14 and 15. At the beginning of the game, the predictions of human observation data (Figure 14b) and the trained model with five human data (Figure 14c) and ROCI (Figure 14d) are almost similar. On the other hand, at the end of the game, predictions are almost different.

Appendix F.

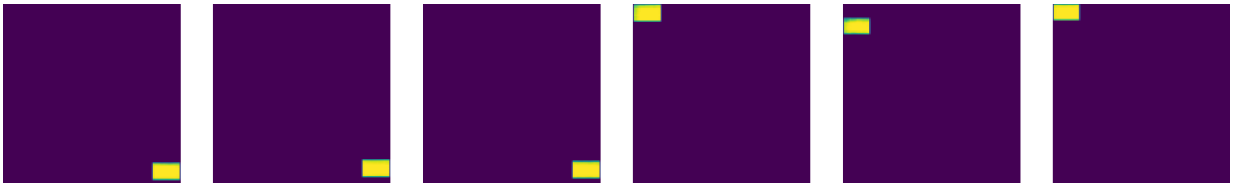
As mentioned in section 6.1, we additionally show the experimental results for various states, demonstrating that the use of multiple human observational data is effective, as shown in Figure 16. Figure 16a shows the game state, and Figure 16b shows the target data for five human observational data. Figures 16c, 16d, and 16e show the prediction of the model trained with single human data, the prediction of the training model with five



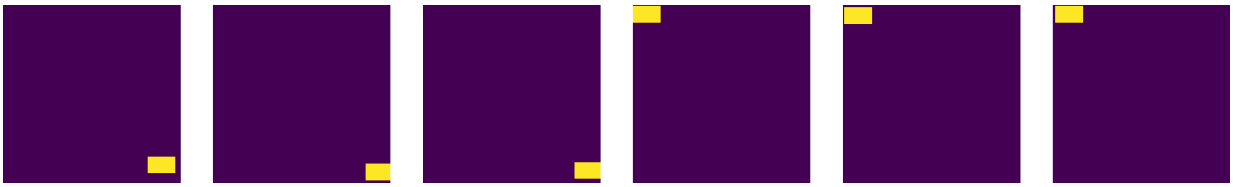
(a) Game state feature



(b) Five masked data based on human observational data

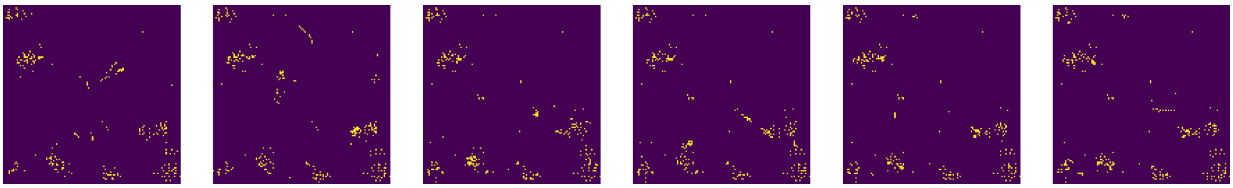


(c) Results for Mask R-CNN with five human data



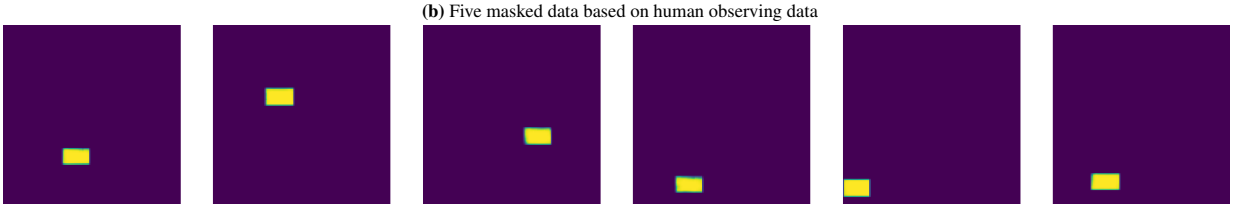
(d) Results for Mask R-CNN with ROCI

Figure 14: Experimental results at the beginning of the game



(a) Game state feature

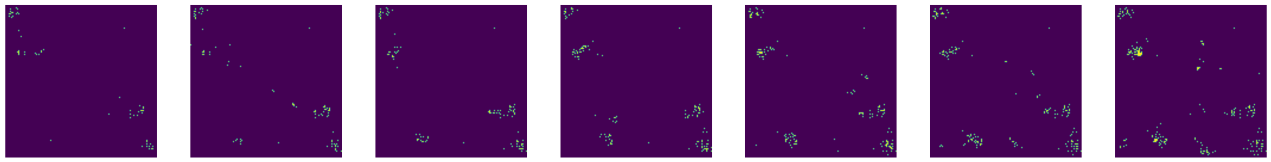
(b) Five masked data based on human observing data



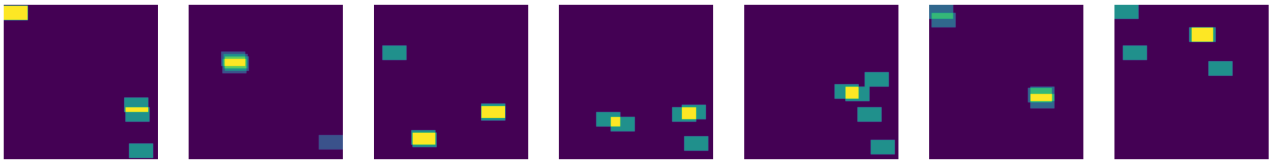
(c) Results for Mask R-CNN with five human data

(d) Results for Mask R-CNN with ROCI

Figure 15: Experimental results at the end of the game



(a) Game state feature



(b) Five masked data based on human observational data

(c) Results for Mask R-CNN with a single human data

(d) Results for Mask R-CNN with five human data

(e) Results for Mask R-CNN with ROCI

Figure 16: Experimental results for our method

human data, and the prediction of the learning model that combines five human data and ROCI, respectively. Comparing the results of Figure 16b and Figures 16c, 16d, and 16e, we can see that Figure 16e predicts the overlapping part of the human observational data best.