scientific reports

Check for updates

OPEN

Improving esports viewing experience through hierarchical scene detection and tracking

Ho-Taek Joo[1,3], Sung-Ha Lee[2,3], Insik Chung[1] & Kyung-Joong Kim[1,2]✉

The role of an observer in esports is to provide spectators with the most engaging scenes in real time. To automate this process, various research has been conducted. In this study, we utilize Vision Transformer (ViT)-based object detection to enhance the accuracy of automatic observers. However, while ViT-based detection more accurately identifies engaging game scenes, it often leads to frequent and abrupt scene changes, reducing viewer comfort. To address this issue, we propose a novel hierarchical structure that combines scene detection with scene tracking, maintaining high accuracy while ensuring smoother transitions between scenes. This approach also improves inference speed, as the tracking model is faster than the detection model. We computationally evaluated six observer models in terms of accuracy and camera stability, with our method demonstrating significantly more stable camera control. Additionally, user testing indicated a strong preference for our model over those without tracking. A video comparing our method to the state-of-the-art can be viewed at https://youtu.be/gWiU4GACZEg.

Keywords StarCraft, Esports, Game observers, Spectators

Esports, a term that refers to organized competitive video gaming, is an industry that has recently been in the spotlight. The Esports industry is growing at an incredibly fast rate[1–3], and does not present signs of declining in the foreseeable future[4].

One of the essential elements in the esports industry is the human observer. In esports, human observers are responsible for creating an engaging viewing experience for spectators by controlling the in-game camera and continuously presenting scenes that are critical to the game or likely to be of interest to the spectators.

However, such human observers are costly to train and hire, as they must possess high game expertise and a deep understanding of spectators' interests, making them unsuitable for small game competitions. Furthermore, since multiple events often occur simultaneously across the game map, human observers are at risk of missing important events.

Therefore, various studies have focused on developing automated observers to replace human observers in esports. The methods being researched for the automated observer can be broadly classified into rule-based and learning-based approaches. These methods have been applied to various games, including Defense of the Ancients 2 (Dota 2), League of Legends (LoL), and StarCraft[5]. In particular, StarCraft features significantly more units compared to Dota 2 or LoL, creating complex and simultaneous events, which pose a challenging scenario for observing.

One of the learning-based studies[6] on automatic observers proposed a method for collecting human observational data and learning human observation styles from this data. The study treated human observational data as objects within two-dimensional spatial regions. To learn these two-dimensional spatial scenes from the human observational data, the researchers applied object detection algorithms, demonstrating significantly improved performance compared to previous studies. We have also partly adopted this approach in our work.

In our study, we employed Vision Transformer (ViT)-based object detection in place of the CNN-based detection used in previous studies[6] to enhance accuracy. Compared to Convolutional Neural Networks (CNNs), which were utilized in earlier learning-based observer research, Vision Transformers (ViTs) possess larger receptive fields. This attribute grants them a superior capability in comprehending the broader context during the feature extraction process[7].

[1]School of Integrated Technology, Gwangju Institute of Science and Technology, 123 Cheomdangwagi-ro, Buk-gu, Gwangju 61005, South Korea. [2]AI Graduate School, Gwangju Institute of Science and Technology, 123 Cheomdangwagi-ro, Buk-gu, Gwangju 61005, South Korea. [3]Ho-Taek Joo, Sung-Ha Lee have contributed equally to this work. ✉email: kjkim@gist.ac.kr

However, while the ViT-based object detection mechanism demonstrated higher accuracy in detecting game scenes, it exhibited notable fluctuations in continuous detection, frequently and abruptly changing scenes, unlike the behavior of human observers. While this does not impact the computational accuracy, it can affect the viewing experience of human spectators greatly.

To address this challenge, we propose a method for creating a stable automatic observer while still exhibiting high accuracy. This method integrates scene detection with scene tracking mechanism. The scene detection model (ViT-based object detection) is designed to identify the most engaging scenes within the game. Concurrently, the scene tracking model focuses on following the scenes pinpointed by the scene detection model. This approach maintains the continuity and coherence of the viewing experience, effectively reducing abrupt scene shifts, and providing a more natural and human-like observation flow. With this integrated system, our study aims to enhance camera control and the overall viewing experience while still achieving high computational accuracy.

More specifically, the scene detection model is trained using human observational data to learn human-like observation patterns. Meanwhile, the scene tracking model continually searches for scenes in subsequent frames that closely resemble those identified by the scene detection model. Our proposed method is briefly shown in Fig. 1. In Fig. 1a shows the representation of the entire map of StarCraft. The map includes StarCraft's in-game units, resources, buildings, etc.

The scene detection algorithm finds the red border (*b*), given (*a*) in Fig. 1. The red border in the Fig. 1 represents the scene that the spectators in Esports will be most interested in on the entire screen of StarCraft, and (*c*) shows the actual screen of StarCraft in the red border of (*b*). The scene (*c*) found by the scene detection algorithm is given to the scene tracking method, and the scene tracking algorithm finds the scenes (*d*) most similar to the found scene (*c*) in subsequent frames. In (*e*), the red border represents the scene found by the scene detection method, and the green borders represent scenes found by the scene tracking method in consecutive frames.

Another major advantage of this method is that it not only stabilizes the outputs without significantly compromising accuracy for spectators but also greatly reduces inference time, a critical metric for automatic observers. The primary role of an observer in Esports is to deliver real-time observation scenes to the spectators. Most games in Esports can display smooth on-screen motion when provided with inference times reaching at least 30 frames per second (FPS) [8]. If the inference time falls below 30 FPS, real-time broadcasting becomes challenging. Therefore, the automatic observer should achieve more than 30 FPS. However, without the integration of scene tracking, the observer relying solely on scene detection mechanisms using object detection algorithms, such as Mask R-CNN or MViTv2, falls below the 30 FPS threshold. It is with the incorporation of scene tracking that real-time broadcasting becomes feasible.

For the computational evaluation, We compared six different models, employing MViTv2[9] for ViT-based object detection. These models include: MViTv2 with tracking, Mask-RCNN with tracking, MViTv2, Mask-RCNN, sscait, and a human observer. We computationally evaluated both accuracy and camera stability. The results showed that the ViT-based object detection method achieved the highest accuracy, but it demonstrated significantly poorer camera stability. However, the inclusion of a tracking method in the learning-based models notably improved camera stability.

In addition to the computational accuracy result, we conducted a user test to access the actual viewing experience of the spectators. We compared the ViT-based object detection model with and without tracking mechanism to evaluate the impact of integrating tracking. The result indicated a higher preference for our model that included the tracking mechanism compared to those without it.

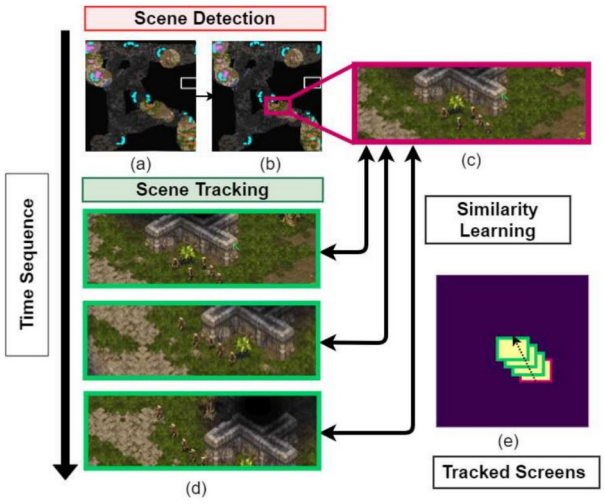

Fig. 1. Automatic Observer Model using Scene Detection and Tracking Algorithms: Given the entire image (**a**) of the game, the scene detection model finds a scene (**b**, **c**). When the scene detection model informs the scene to be tracked, the scene tracking model continuously tracks it (**d**, **e**) using similarity learning. Figure was created using draw.io (https://app.diagrams.net/).

In summary, the main contributions of this work are as follows:

- We propose a novel hierarchical structure, integrating scene detection and scene tracking methods, to stabilize camera control and create automatic observers that have behavior similar to that of human observers.
- We utilized the ViT-based object detection mechanism for scene detection to increase the accuracy of the observer, which is novel in the context of the game observer.
- By employing similarity-based tracking, our method significantly improves inference time, which is an important factor in real-time observing.
- Our proposed observer model achieves higher accuracy with significantly stable camera control, and user tests demonstrated a preference for our method compared to the object detection-based observer without tracking.

Related work

Automatic observer

An observer shows the most interesting in-game scene for Esports spectators, and an automatic observer replaces the human observer. Methods used for automatic observers can be divided into rule-based and learning-based methods. First, rule-based automatic observers predefine interesting events according to a priority system and display a defined event on the screen whenever it occurs. Rule-based automatic observers for StarCraft have been developed, such as AIIDE (Artificial Intelligence and Interactive Digital Entertainment, https://www.cs.mun.ca/dchurchill/starcraftaicomp/) for the StarCraft AI Competition and SSCAIT (Student StarCraft AI Tournament, https://sscaitournament.com/) observers As a result, the MViTv2[5]. Second, learning-based automatic observers have been studied using deep learning or machine learning. These studies also define each event and its corresponding hierarchical importance before using deep learning or machine learning methods to predict or classify a few predefined events, such as team fights, farming, and ambushing[10,11]. Both the rule-based observer and learning-based observer predefine events and follow them.

The limitations of previous studies are that they rely on the manual definition of events and their hierarchical importance prior to applying deep learning or machine learning techniques. These approaches are typically used to predict or categorize a limited set of predefined events, such as team battles, resource gathering, or ambushes. However, despite their widespread use, these event-based learning methods have notable drawbacks. They are incapable of identifying events that have not been explicitly defined, and the contextual nature of games means that even the priorities of predefined events can change dynamically during gameplay. Furthermore, performing event inference at every frame can lead to inconsistencies in detected events across consecutive frames. If similar frames are classified as different events, abrupt scene transitions may cause confusion for spectators. Even when the same event is detected, the lack of temporal coherence may result in unstable camera movements, leading to inconsistent scene presentation rather than a smooth and natural viewing experience.

Object detection for automatic observer

Object detection algorithms classify what objects are in an image and predict where they are in the image. Object detection research has grown rapidly and has begun to be applied in various fields such as face detection, pedestrian detection, and medical image detection[12–15]. Recently, an automatic observer study also applied an object detection mechanism[6] to train the model to identify exciting scenes for spectators. In that study, researchers collected human observational data and designated the two-dimensional scene observed by humans as the Region of Interest (ROI). The scene observed by the human was masked with values of one, and the remaining area was masked with values of zero and used as target data for the object detection algorithm. The study demonstrated substantially improved performance compared to previous works, but the proposed method takes a long time because it finds areas of interest by performing object detection algorithms in every frame.

Visual object tracking with Siamese network

Visual object tracking (VOT) automatically tracks specific objects in a given video where the object is in the first frame. That is, given the initial state of a target in the first frame of a video sequence, VOT aims to automatically obtain the states of the object in the subsequent video frames. Recent methods using Siamese networks for VOT have been developed, such as naive cross-correlation[16] and depth-wise cross-correlation[17,18]. Recent Siamese network-based object tracking algorithms use a target image of the first frame in a video image and a search image that becomes an area to be found in the next frame as input. Because the similarity between two input images is learned offline, additional online learning and parameter fine-tuning are not required, so these algorithms have the significant advantages of fast speed and high accuracy.

Vision transformer

The Vision Transformers (ViT)[19] have emerged by adapting the highly successful transformer architecture[20] from natural language processing to visual processing. Their approach is to divide the image into small patches and then use a self-attention mechanism to learn the relationships between these image patches. The initial ViT started with image classification tasks and has demonstrated impressive performance across a variety of domains, including object detection[21,22] and object tracking[23,24]. However, transformers inherently require longer computational times than CNN due to the quadratic complexity of self-attention mechanisms[19]. This issue becomes particularly pronounced in tasks requiring high-resolution images or where speed is crucial, such as object detection and tracking, leading to a preference for CNNs[25,26] in these areas. Despite these issues, recent advancements like multiscale[27] and window attention[28] significantly reduced computational times. These developments have enhanced the speed of vision transformers to a level suitable for real-time applications[9,29,30].

Utilizing these advancements, we have developed a transformer-based observer that surpasses prior CNN-based models in speed and accuracy.

StarCraft dataset

The proposed framework is divided into two main modules: scene detection and scene tracking models. These models are trained separately, with datasets specifically designed for their individual requirements.

Dataset for scene detection

We used a human StarCraft observational dataset from the work[6]. The dataset consists of 21 StarCraft game datasets, called replay files, that contain in-game states and the (x, y) coordinates of where humans watched at each frame. Each game replay is watched by five different people, aiming to capture collective human viewing tendencies, especially in complex situations where multiple events occur simultaneously, and each person may focus on different aspects. In such cases, data from a single individual may not be reliable. The total amount of data is approximately 420,000 frames (= about 4,000 frames per game $\times$ five times per game $\times$ 21 game replay files).

The primary purpose of this dataset is to train a model that can predict areas of human interest within StarCraft replays based on the in-game state. This dataset is expected to train an AI model that can present the most engaging scenes for viewers. The AI model will detect multiple simultaneous events and select a single viewport based on the event with the highest predicted probability.

Dataset for scene tracking

We adapted a human StarCraft observational dataset from the work of Joo et al.[6] for scene tracking. Observational patterns in game viewing typically involve a cyclical process of event discovery followed by the tracking of these events. Accordingly, the dataset was segmented into two distinct parts: one for the initial identification of events (scene detection) and the other for the subsequent tracking of these events (scene tracking). In the process of selecting appropriate data for scene tracking, the focus was on the spatial continuity between consecutive frames that were observed by human viewers. Specifically, when the area of overlap between consecutive frames exceeded 50%, it was inferred that the observer was engaged in a tracking action. Additionally, given that the tracking dataset essentially constitutes trajectory data across consecutive frames, a subset of data was extracted where a minimum of 50% overlap was consistently maintained across a sequence of 12 frames.

Method

In this research, we present a novel hierarchical architecture for an automatic observer model in Esports. This model is specifically designed to identify and continuously monitor the most engaging scenes for viewers. The architecture consists of two main components: the scene detection network and the scene tracking network.

The scene detection network's role is to identify the most exciting moments within a StarCraft game. Once an engaging scene is identified by the scene detection network, the scene tracking network steps in. Its function is to keep track of and follow scenes that are similar to the initially identified one, doing so over subsequent frames. This process creates a sequence of data that continuously focuses on the initially selected scene, ensuring a consistent and engaging viewing experience.

In the lower part of Fig. 2, various StarCraft scenes are illustrated. Those identified by the scene detection network are marked with red borders, while those followed by the scene tracking network are enclosed in blue

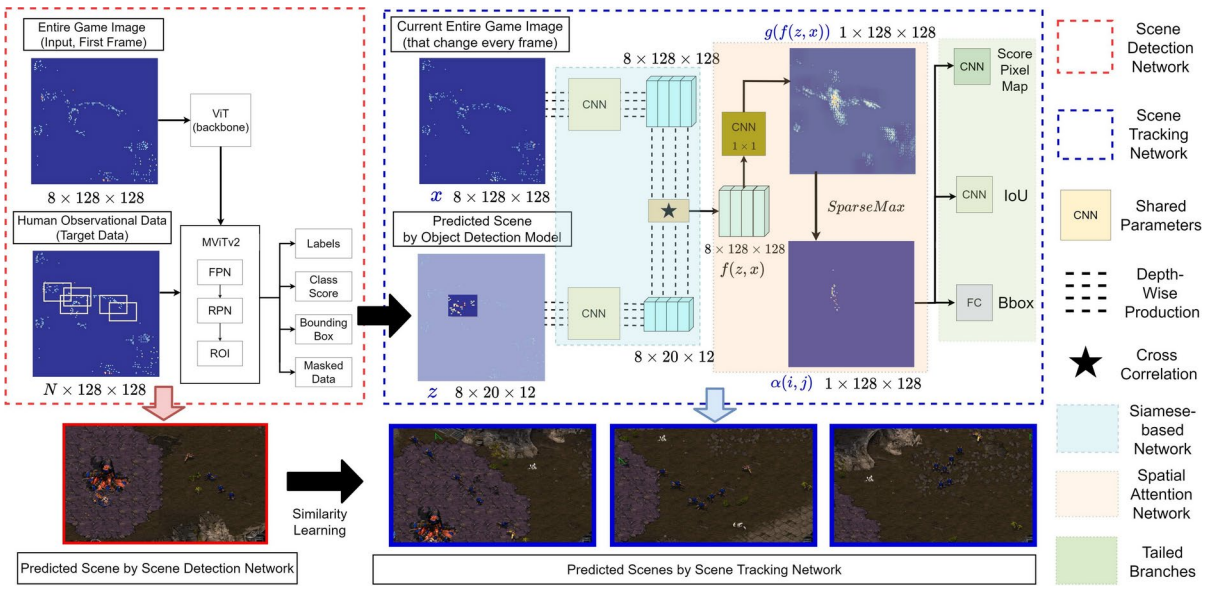

Fig. 2. The overall architecture of our method. Figure was created using draw.io (https://app.diagrams.net/).

borders. The image within the red border depicts the initial scene selected for tracking. On the other hand, the images within the blue borders show subsequent frames where the scene tracking network has successfully located scenes similar to the initial scene identified in red. For example, an image within a red border might show a blue attack unit (Zergling) pursuing a red unit (Probe). The subsequent images in blue borders, identified by the scene tracking network, would then display the continued tracking of this interaction.

Scene detection network with ViT (MViTv2)

The role of the scene detection network in the overall structure is to search for the most exciting scene in the entire StarCraft map. The scene detection network[9] for an automatic observer is designed to learn the style of human observers based on human observational data. The scene detection network utilizes MViTv2, a ViT-based object detection model that specializes in capturing both fine details and broad contextual information within images. This proves to be a significant advantage in the Starcraft Observer, as it necessitates the find scene based not only on the interest of the local scene but also their contextual relevance within the entire game screen.

As shown in the red dotted box in Fig. 2, the scene detection network receives the entire StarCraft game map image, whose size is $(8\times128\times128)$, and finds the most exciting scene, whose size is $(8\times20\times12)$. The model's target data are two-dimensional spatial scenes observed by N humans while watching StarCraft replays. The size of the target data is $(N\times128\times128)$ and is the same as the map size of the input data.

The loss functions consist of the mask loss $\mathscr{L}_{mask}$, the classification loss $\mathscr{L}_{cls}$, and the box loss $\mathscr{L}_{box}$ as follows.

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

The mask loss $\mathscr{L}_{mask}$ is to determine the extent to which the human observational data and the predicted scene overlap. The mask loss is the average binary cross-entropy [31]. The classification loss function $\mathscr{L}_{cls}$ determines whether the predicted scene is a target data (1) or not (0). The loss function is also binary cross-entropy. Finally, The bounding box loss function is similar to regression loss. The model predicts four coordinates $(x_{min}, x_{max}, y_{min}, y_{max})$ in the target data. The bounding box loss function $\mathscr{L}_{bbox}$ applies smooth $L1$ loss[32].

Output The scene detection network finally predicts "masked data," "bounding boxes," "labels," and "class scores." Among the four predicted values, "masked data," and "bounding boxes" are also used in the scene tracking network. Masked data refers to the most exciting scene $(8\times20\times12)$ on the entire map $(8\times128\times128)$. Bounding boxes is predicted coordinates for $(x_{min}, y_{min}, x_{max}, y_{max})$ for target data.

Scene tracking network

The role of the scene tracking network is to keep track of the scene that is most similar to the scene the detection network finds each time the image in the current frame is updated. The scene tracking network is surrounded by a blue dotted line in Fig. 2. The scene tracking network is divided into **Input data, Siamese-based Network, Spatial Attention Network,** and **Tailed Branches**, which are described in detail below.

Input data The input data for the scene tracking network are a pair of image patches (denoted as z, found by the scene detection network) cropped from the first target frame and the current frame image (denoted as x) for tracking in Fig. 2. Here, the z image is fixed until another event occurs, and the x image is the entire game image that changes over the frame, which is the smallest unit of time in the game.

Siamese-based network for similarity learning Siamese neural network[33] is the method for employing a structure to learn the similarity between inputs. A siamese network is commonly trained based on an image dataset with RGB (Red, Green, Blue) channels, but the dataset's channels for automatic observers are divided according to the characteristics of in-game data (e.g., ground units, air units, buildings, etc.) Therefore, we used a modified version of the existing siamese network.

$$f(z,x) = \psi(\phi(z) \star \phi(x)) \tag{2}$$

where the input z is the StarCraft scene inferred by the scene detection network, and the input x is the current frame image in Fig. 2. And where $\phi(.)$ denotes a channel-wise Convolutional Neural Network (CNN) layers, $\star$ denotes the cross-correlation operation, and $\psi(.)$ denotes the depth-wise CNN layers[34].

The depth-wise CNN layers $\psi(.)$ are used in siamese networks, but the **channel-wise CNN layers** $\phi(.)$ are newly applied in our method. The reason for only applying channel-wise CNN layers $\phi(.)$ is that the StarCraft data have unique properties for each channel. Channel-wise CNN layers mean that each channel of input data is passed through convolution layers, but the results of each channel are not summed, unlike regular CNN layers. The final cross-correlation operation[16] is based on the obtained maps by channel-wise CNN layers. Cross-correlation usually locates a smaller image z within a larger image x by learning similarity. As a result, the output $f(z, x)$ represents the similarity at all translated sub-windows on a dense grid between inputs z and x, and the goal of the network is to obtain the maximum value of similarity on $f(z, x)$.

Spatial Attention Network We proposed the spatial attention network that helps to find the most similar location on the $f(z, x)$ map by filtering out dis-similar scenes. The $f(z, x)$ map shows which part of x is similar to image z for each channel that contains in-game features. To calculate the importance of each channel, which represents in-game features, the spatial attention network goes through a 1×1 CNN layer. In other words, this CNN layer is summed by applying only the weight for each channel, and it determines which channel is used with the most importance. As a result, the spatial attention network first obtains a convolutional feature map $g(f(z, x))$ and then computes an attention mask α_t emphasizing the most similar scene. The attention mask represents the probability distribution for all pixels (i, j) on the feature map $g(f(z, x))$ using the softmax or sparsemax operators[35], emphasizing pixels with high similarity.

$$\alpha_t = \text{sparsemax}(g(f(z,x))), \qquad \text{so that} \sum_{i,j} \alpha_t(i,j) = 1 \tag{3}$$

$$\alpha_t = \begin{cases} \alpha_t, & \text{if } \alpha_t > threshold \\ 0, & otherwise \end{cases} \tag{4}$$

Finally, the obtained map $\alpha_t(i,j)$, which represents high similarity, is followed by three additional predictors.

Three-tailed branches for finding the most similar scene The loss functions for three branches are designed based on SiamMask[36]. The bounding box (Bbox) loss is cloned from SiamMask, the score pixel map (SPM) loss was additionally proposed, and the IoU loss is modified from SiamMask.

- **Bbox branch**: The branch predicts the central coordinates of the square box of the predicted scene. This branch computes the smooth L1 loss[32] between the scene tracking network's predicted Bbox and the scene detection network's Bbox output $(x_{min}, y_{min}, x_{max}, y_{max})$.

$$L_{bbox} = \begin{cases} 0.5(\mathrm{z}-\hat{\mathrm{z}})^2 & \text{if } |\mathrm{z}-\hat{\mathrm{z}}| < 1 \\ |\mathrm{z}-\hat{\mathrm{z}}| - 0.5 & \text{otherwise} \end{cases} \qquad \frac{(x_{min}+x_{max})}{2}, \frac{(y_{min}+y_{max})}{2} \in \hat{z} \tag{5}$$

- **SPM branch**: The SPM branch predicts the largest value of $\alpha_t(i,j)$ corresponding to each grid cell. The ground truth is also the scene detection network's bounding box output, like the Bbox loss.

$$L_{SM} = \begin{cases} 0.5(\mathrm{z}-\hat{\mathrm{z}})^2 & \text{if } |\mathrm{z}-\hat{\mathrm{z}}| < 1 \\ |\mathrm{z}-\hat{\mathrm{z}}| - 0.5 & \text{otherwise} \end{cases} \qquad z' = \arg\max_a \alpha_t(i,j) \tag{6}$$

- **IoU branch**: This branch measures the extent to which the predicted scene overlaps with the target data. The predicted scene is created through box coordinates, and the target data are human observational data. Here, if the overlapping area is 70% or more, it is 1. Otherwise, it is 0[37].

$$\mathrm{IoU}^* = \frac{\text{Intersection}(B, B^*)}{\text{Union}(B, B^*)} \tag{7}$$

where B is the predicted bounding box and B^* is its corresponding ground-truth target data.

$$L_{IoU} = \begin{cases} 1, & \text{if } IoU > threshold(0.7) \\ 0, & otherwise \end{cases} \tag{8}$$

Training Objective We optimize a training objective in the scene tracking network as follows:

$$L_{total} = \lambda_1 L_{\text{bbox}} + \lambda_2 L_{\text{SM}} + \lambda_3 L_{\text{IoU}} \tag{9}$$

where $\lambda_1, \lambda_2, \lambda_3$ are hyperparameters.

Experiments

Implementation details

For the **dataset for evaluation**, As mentioned in section 3, we used 21 StarCraft game datasets, which can be divided into two types. The 15 game dataset with fixed starting locations, maps, and species, and the 6 game datasets with different starting locations, maps, and species. Here, 15 game datasets are evaluated using 3-fold cross-validation, and 6 game datasets are only evaluated for generalization performance. For **hyperparameters**, we set $\lambda_1 = 0.01$, $\lambda_2 = 1$, $\lambda_3 = 0.2$. And for **implementation setup**, we used the AdamW optimizer to train our method for 20 epochs. The batch size was set to 32, and the learning rate and weight decay were both set to 1×10^{-4}. After training for 10 epochs and 15 epochs, the learning rate decreased to 1×10^{-5} and 1×10^{-6}, respectively. The entire training process took approximately 10 hours on an NVIDIA RTX 2080 Ti GPU.

Baselines

We evaluated our method and compared it to the previous rule-based and learning-based automatic observer methods in StarCraft. The baselines are as follows.

SSCAIT observer The SSCAIT Observer[5], represents a rule-based system designed to control the in-game camera by following predefined events, which are prioritized based on a set scoring system. Developed specifically for the automatic observation of StarCraft AI competitions, it exemplifies a traditional approach in game observation automation. **Mask R-CNN observer** The Mask R-CNN Observer employs deep learning techniques for automatic observation in StarCraft. This method leverages object detection algorithms to anticipate scenes likely to be focused on by human observers. Selected as a baseline for its recency and superior performance, the implementation adheres to the hyperparameters and dataset specifications detailed in[6]. **Mask R-CNN observer with scene tracking (Ours)** Building upon the Mask R-CNN framework, this method introduces scene tracking into the scene detection process. Targeting StarCraft's typical 24 frames per second gameplay, it applies scene tracking every (24N) frame, ensuring continuous observation of the same scene. This dual-stage

process first identifies specific events using Mask R-CNN in the initial frame of each (24*N*) sequence, followed by sustained tracking of these events throughout the subsequent $(24N - 1)$ frames. **MViTv2 observer (Ours)** This observer is based on the MViTv2 model, a Vision Transformer (ViT)-rooted object detection paradigm. It is designed to offer an advanced interpretation and analysis of game scenes, leveraging sophisticated object detection methodologies. **MViTv2 observer with scene tracking (Ours)** This approach parallels the Mask R-CNN Observer with Tracking but utilizes the MViTv2 model. It identifies specific events in the first frame of every (24*N*) frame sequence, employing scene tracking techniques to maintain focus on these events throughout the remaining $(24N - 1)$ frames.

Evaluation and results

We conducted a comprehensive assessment to evaluate our model's suitability and performance within the context of Esports. This evaluation addressed various aspects such as scene selection accuracy (regarding the engagement of scenes for spectators), inference time (to ensure real-time application feasibility in Esports), and the naturalness of screen transitions (aiming to enhance the viewer experience).

1 . Quantitative evaluation

Evaluation for selecting engaging scenes

In esports, the primary task of the observer is to select interesting game scenes for the general audience. To evaluate how well the automatic observer's view aligns with spectator interests, we used observational data from multiple human viewers per game-five in our case-, as the ground truth. We then assessed the overlap between the predicted scene by the observer and the scenes observed by the human viewers. The evaluation focuses on how closely our findings align with general human spectating data. We assessed the model's ability to identify scenes that capture spectator interest using standardized metrics.

As previously discussed in Section 3 on data collection, the ground truth data for our model is derived from the viewing patterns of human spectators. For each StarCraft match replay, five spectators were assigned to observe the game, and their observational data was used as the ground truth. When our AI model predicted a viewport, we evaluated its accuracy using the Intersection over Union (IoU) metric, a standard measure in object detection. IoU is calculated as the ratio of the overlapping area between the predicted viewport (VP) and the human observational viewports (VH) to the total area of their union, as defined by the following equation:

$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}} = \frac{|VP \cap VH|}{|VP \cup VH|} \tag{10}$$

However, this metric was adapted for our evaluation, considering that the ground truth should reflect the combined viewports of all human observers, rather than a single observer's viewport. The model's performance is better evaluated by measuring how well the predicted viewport aligns with the combined viewports of all spectators. This can be quantified using the following equation:

$$\text{Performance} = \frac{\text{Area of Overlap with Human Viewports}}{\text{Area of Predicted Viewport}} = \frac{|VP \cap (VH_0 \cup, ..., \cup VH_{n-1} \cup VH_n)|}{|VP|} \tag{11}$$

Here, *n* is the number of human observers, and VH_n is a viewport of the *n*th observer. A large overlapping area between the predicted and human viewports indicates that the model's predictions closely resemble general human observational behavior. The overlapping area was calculated for each frame and averaged across the entire testing replay, providing a robust quantitative evaluation of the model's performance.

Table 1 presents the performance in terms of Average Overlap (AO) for six baseline models across different game progression phases, focusing on accuracy. An in-depth examination reveals that the MViTv2 model without tracking exhibits the highest accuracy with an AO of 0.6561, closely followed by its counterpart with tracking at an AO of 0.6294. The Mask R-CNN without tracking ranks third with an AO of 0.5880, slightly ahead of the Mask R-CNN with tracking, which has an AO of 0.5819. The human observers are next with an AO of 0.5079, and the SSCAIT system records the lowest accuracy at an AO of 0.4490.

When examining the impact of tracking on algorithm performance, models without tracking tend to achieve higher accuracy. This superior accuracy is attributed to the models' strategy of basing predictions solely on closeness to the ground truth data for each frame. While effective in maximizing accuracy, this approach leads to frequent scene changes and potentially jittery camera movements, which could distract the viewer. In contrast, models that incorporate these features create a more stable and consistent viewing experience, as demonstrated

Progression	SSCAIT	Human	Mask R-CNN	Mask R-CNN w/ scene tracking (Ours)	MViTv2 (Ours)	MViTv2 w/ scene tracking (Ours)
Start ∼ 1/4	0.3874	0.5227	0.6716	**0.6661**	**0.6974**	**0.6695**
1/4 ∼ 2/4	0.4562	0.5030	0.5893	**0.5769**	**0.6445**	**0.6137**
2/4 ∼ 3/4	0.4519	0.4994	0.5548	**0.5469**	**0.6393**	**0.6105**
3/4 ∼ End	0.4609	0.5064	0.5364	**0.5376**	**0.6434**	**0.6239**
AO (average)	0.4490	0.5079	0.5880	**0.5819**	**0.6561**	**0.6294**

Table 1. The AO benchmark.

in the evaluation of camera control and user testing. They achieve this by continuously identifying and following similar areas from frame to frame once a region close to the ground truth has been pinpointed. This method enhances viewer comfort by reducing abrupt scene transitions, although it may miss rapid and transient changes in the scene content, slightly compromising accuracy. Despite this, the benefits of tracking-significantly faster inference speeds, smoother transitions, and greater stability-are crucial for live broadcasting and spectator experience.

Evalution for inference time (FPS)
Considering the real-time nature of Esports, the speed at which our model processes outputs was evaluated. The aim was to ensure that the model operates within the time constraints necessary for live Esports broadcasting without compromising on the quality of the output. We conducted an assessment of the inference time for four leading learning-based observer models, as identified in Table 1.

This evaluation was measured in frames per second (FPS), a crucial metric for determining an observer's ability in the context of real-time Esports broadcasting. For effective live Esports broadcasting, a minimum performance threshold of 24 frames per second is required.

As represented in Table 2, our proposed tracking-based observer models demonstrate significant processing speeds of 93 and 85 FPS in the StarCraft game environment, which is surpassing the commonly accepted threshold of 60 FPS. These results indicate a notable improvement over existing models, addressing one of the key challenges Esports observation: maintaining real-time inference speeds for broadcasting while ensuring reliable tracking accuracy. Notably, this represents approximately a sevenfold increase in tracking speed compared to observer models without tracking, since the tracking model is much lighter than the object detection model. Consequently, our models have successfully achieved the requisite inference speed for seamless integration into live Esports broadcasting environments.

Evaluation of camera control
We have developed an evaluation method for stable camera control to quantitatively assess whether the observer has stable camera control. For the camera control evaluation, we hypothesized that stability is mainly affected by two factors: screen fluctuation and screen transition. Screen fluctuation refers to the shakiness of the camera, which results from changes in the camera's direction in a continuous manner. Conversely, screen transition denotes the camera shifting from one scene to a different scene in a discontinuous manner. These two factors can negatively impact the experience of spectators if they occur frequently.

Firstly, to quantify screen fluctuation F, we employed the camera change direction angle, denoted as θ. This angle is defined as the angle between the camera's orientation in the previous frame and the current frame, and then between the current frame and the next frame. If $\theta = 0$, which indicates the camera moves without changing direction, or the camera remains still, there is no camera shake. Camera instability is most severe when the camera changes its moving angle to drastically different directions. Camera movement oscillation increases if an angle change approaches $theta = \pi$, representing a complete reversal of the camera's direction. Therefore, this can be calculated as the following equation, which has a maximum value when $\theta = \pi$ and a minimum at $\theta = 0$, summed and averaged over the total number of game frames:

$$F = \frac{1}{\text{total frames}} \sum_{i=0}^{\text{total frames}} |\cos(\theta_i) - 1| \tag{12}$$

Secondly, to quantify the camera transition score T, we determined whether a transition has occurred by calculating the overlap percentage between consecutive frames. If the overlap between the current frame and the next frame does not exceed the overlap threshold of t percent, we consider that the screen has transitioned. To obtain the total camera transition of the game, we added the number of transitions in the game and averaged over the total number of frames. Therefore, the camera transition score be calculated as the following equation:

$$T = \frac{1}{\text{total frames}} \sum_{i=0}^{\text{total frames}} \begin{cases} 1, & \text{if overlap} < t \\ 0, & \text{otherwise} \end{cases} \tag{13}$$

Finally, we define the camera instability score, which is proportional to camera fluctuation and camera transition score:

$$Camera\ instability = F \times T \tag{14}$$

We calculated the screen fluctuation, screen transition, and resulting camera instability for six different observers. In this paper, we used $t = 70$ to determine whether the screen has been transitioned or not.

Method	Mask R-CNN	Mask R-CNN w/ scene tracking (Ours)	MViTv2 (Ours)	MViTv2 w/ scene tracking (Ours)
FPS	15	**93**	**12**	**85**

Table 2. Evaluation on four observers for inference time.

Progression	SSCAIT	Human	Mask R-CNN	Mask R-CNN w/ scene tracking (Ours)	MViTv2 (Ours)	MViTv2 w/ scene tracking (Ours)
Screen fluctuation	0.0195	0.0206	0.0946	**0.0198**	**0.441**	**0.0203**
Screen transition	0.0222	0.0129	0.0349	**0.0110**	**0.0326**	**0.00846**
Camera instability	0.00044	0.00027	0.0033	**0.00022**	**0.0144**	**0.00017**

Table 3. The camera control scores of the observing models.

(a) Example of the actual game screen where the model tracks a group of moving ground units.

(b) Example of the actual game screen where the model tracks a large group of units in combat

Fig. 3. Qualitative evaluation: In-game screenshots of MViTv2 w/ tracking and MViTv2.

As represented in Table 3, the MViTv2 with tracking demonstrated the lowest score in camera instability. This performance was followed closely by Mask-RCNN with tracking, which also benefited from the integration of tracking mechanisms, highlighting the importance of combining detection and tracking for effective camera control. The human observer exhibited the second-lowest score, followed by SSCAIT. Learning-based models without tracking showed significantly higher scores, with the ViT-based object detection method scoring higher than Mask-RCNN. This indicates that our tracking method significantly stabilized the camera control of the observer. To summarize, by integrating tracking into object detection-based models such as Mask R-CNN and MViTv2, we developed a model that excels in camera stability and surpasses other automated observers in maintaining a balance between event detection and viewer comfort.

2. Qualitative evaluation

A qualitative analysis was conducted using actual in-game scenes to evaluate the effectiveness of the observer models. Figure 3 displays the in-game visuals as captured by the MViTv2 and its advanced variant, MViTv2 with tracking. Notably, Fig. 3a illustrates the scene where a group of ground units is moving, while Fig. 3b depicts a scene of intense combat involving numerous units.

In the analysis of Fig. 3a, the MViTv2 model alternates its focus among different groups of units across the first, second, and fourth screens, differing from the units highlighted in the third and fifth screens. This leads to multiple screen transitions in a short time, potentially complicating the viewer's understanding of the action. Conversely, the MViTv2 with tracking model consistently follows a specific group of ground units across all screens, keeping the focus centralized. This uniform tracking approach ensures a more seamless and intelligible viewing experience.

Upon examining Fig. 3b, the MViTv2 model, in its third and fifth screens, captures only a segment of the combat units. On the other hand, the MViTv2 with tracking model consistently covers all units engaged in the combat throughout all screens, providing an all-encompassing view of the action. This method ensures that spectators receive a more complete and fluid representation of the in-game events.

This qualitative evaluation clearly demonstrates that the MViTv2 with tracking model substantially improves the spectator's experience by offering a more stable and continuous portrayal of in-game dynamics. Also, the corresponding video examples can be found in the supplementary files for further reference.

Method	SSCAIT	Human	Mask R-CNN	Mask R-CNN w/ unit tracking	Mask R-CNN w/ scene tracking (Ours)	MViTv2 (Ours)	MViTv2 w/ unit tracking	MViTv2 w/ scene tracking (Ours)
AO (average)	0.4490	0.5079	0.5880	0.5624	**0.5819**	**0.6561**	0.6100	**0.6294**

Table 4. Ablation Study: Comparison of AO Scores.

Method	SSCAIT	Human	Mask R-CNN	Mask R-CNN w/ unit tracking	Mask R-CNN w/ scene tracking (Ours)	MViTv2 (Ours)	MViTv2 w/ unit tracking	MViTv2 w/ scene tracking (Ours)
Screen fluctuation	0.0195	0.0206	0.0946	0.0288	**0.0198**	**0.0441**	0.0360	**0.0203**
Screen transition	0.0222	0.0129	0.0349	0.0151	**0.0110**	**0.0326**	0.0103	**0.00846**
Camera instability	0.00044	0.00027	0.0033	0.00043	**0.00022**	**0.0144**	0.00037	**0.00017**

Table 5. Ablation Study: Camera Control Scores.

Questions
Q 1. The observer captured important in-game events (event).
Q 2. The camera movement was natural (camera movement).
Q 3. I was immersed in the game (immersion).
Q 4. Observing helped me understand in-game situations clearly (understanding).
Q 5. Observing was satisfactory (satisfaction).

Table 6. Survey to evaluate the experiences of users. Users rated their responses on a 5 point Likert scale.

Comparisons to other tracking method
For this ablation study, we experimented with two different tracking methods: Mask R-CNN observer with unit tracking and MViTv2 observer with unit tracking.

- **Mask R-CNN observer with unit tracking** This approach was adopted for an ablation study to compare whether the proposed scene tracking method is optimal compared to other tracking methods. This observer is based on the Mask R-CNN model. Unlike the scene tracking method, which considers the entire feature set of the current viewport and identifies the most similar viewport in the subsequent frame, unit tracking follows a more conventional approach by tracking individual units. Specifically, this method identifies the unit closest to the center of the detected viewport and tracks the nearest corresponding unit in the next frame. This tracking technique is maintained on events throughout the remaining $(24N - 1)$ frames.
- **MViTv2 observer with unit tracking** This approach follows the same methodology as the Mask R-CNN observer with unit tracking, but instead of using Mask R-CNN as the base model, it employs MViTv2.

Overall, in both Mask R-CNN-based and MViTv2-based tracking models, all tracking methods, as expected, yielded lower performance compared to the original detection model, but showed higher performance compared to the SSCAIT and human, in Table 4. However, among the tracking methods, unit tracking scored lower than scene tracking, demonstrating the advantage of our proposed scene tracking approach in maintaining better performance.

Table 5 shows that in both Mask R-CNN and MViTv2 models, incorporating tracking methods significantly improves camera instability scores. While both tracking methods contributed to better camera control, unit tracking exhibited a slightly higher instability score compared to scene tracking, indicating that scene tracking results in more stable camera movements.

User test

We conducted a user study to evaluate the impact of integrating a tracking method into object detection. Participants were shown to two variations of the MViTv2-based automatic observation model-one with the tracking module and one without-for comparative analysis. To assess the user experience, we used a custom questionnaire 6 that addresses five critical aspects of spectating esports: the accuracy of event detection, the naturalness of camera movement, the game immersion, the clarity of game understanding, and overall user satisfaction. To ensure consistency in user evaluation across various game situations, we used a total of 12 different game matches. Each participant watched a unique game match.

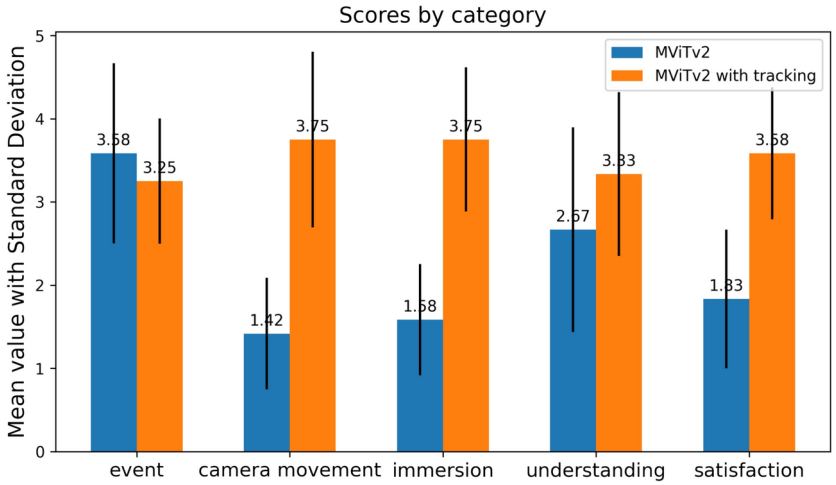

Fig. 4. User scores of post-survey in Table 6 for the MViTv2 and MViTv2 with tracking. The graph depicts the mean and standard deviation for each category.

Participants

We recruited 12 participants, all of whom had experience in either spectating or playing StarCraft. All participants were male, and their ages ranged from 24 to 32 years (Mean = 27.7, SD = 2.67). Participants were surveyed regarding the frequency of their StarCraft engagement over the past month. Among them, 5 reported spectating or playing StarCraft for less than one hour, 2 for one to five hours, 4 for five to ten hours, and 1 had not spectated or played recently. Additionally, we asked participants about the last time they spectated or played StarCraft: 2 participants reported that it was a day ago, 9 reported it was a month ago, and 1 reported it was a year ago.

Procedure

Upon arrival, participants were briefed on the overall study procedure and the objectives of the experiment. After the briefing, participants were presented with two screens simultaneously, each displaying the same game but observed through different models: The MViTv2 (ViT-based object detection) and MViTv2 with the tracking method. The sequence of model presentations and the game replays observed by the participants were randomized. Following each session, participants were requested to complete a survey, which is presented in Table 6, to evaluate the observing models. This process was repeated for four different StarCraft games. After the experiment, participants were asked to provide feedback on each model after completing the user test.

Results

Figure 4 presents the user test results for MViTv2 and MViTv2 with tracking. The mean scores for MViTv2 with tracking were higher than those for MViTv2 in four categories, except in event capturing. We assessed the statistical significance of these results. Due to the small sample size and the lack of normal distribution in the data, we employed the Mann-Whitney U test[38] for our statistical analysis. The results indicated a statistically significant difference between the two models in terms of camera movement ($p < 0.01$), immersion ($p < 0.01$) , and satisfaction ($p < 0.01$).

We also collected qualitative feedback from users by interviewing them about each observer for further analysis of the viewing experience. Participants were asked to provide overall feedback about each observer freely. The most common observation, mentioned by 9 out of 12 participants, was that the model without tracking made it harder to immerse in the game (e.g., *"The left one (model without tracking) had quick screen transitions, which made it a bit uncomfortable to watch.", (P1)*)

The interview results also provided insights into how the model affects viewer engagement over time. One participant noted that *"the early game showed no significant difference, but as the game progressed and events occurred more frequently (combat, unit movement), the left one (model without tracking) showed more frequent screen transitions and became shaky and chaotic", (P12)*. This interview, combined with the questionnaire results, suggests that the proposed tracking observer model maintains stable camera control even as the game becomes more complex, thus significantly improves scores for immersion and overall satisfaction compared to the model without tracking.

This result aligns with our quantitative evaluation, showing that our tracking method significantly enhances natural camera movement, with slightly lower event detection, in other words, accuracy. This is because the model without tracking exhibited more rapid scene transitions, making it less likely to miss important scenes when various events occurred almost simultaneously across the map, such as when a player splits their units to attack multiple bases at once. However, this decrease in accuracy did not notably affect the user, showing no significant difference in event detection score. Additionally, multiple participants mentioned during interviews that such rapid scene transitions negatively affected their spectating experience (e.g., *"The left one (model without tracking) was good at quickly switching screens to detect events, but there was too much unnecessary movement even within the same screen, making it hard to follow the observing", (P8)*). Furthermore, the MViTv2 with tracking

showed significantly higher results in camera movement, immersion, and satisfaction than the MViTv2 without tracking.

Limitation and future works

The limitation of our work is that the timing of switching between the detection model and the tracking model was not optimized. Currently, the tracking model runs for a fixed duration instead of being switched into the detection model flexibly. This rigidity may lead to spectator dissatisfaction: prolonged tracking of less significant scenes can induce boredom, while brief tracking of crucial scenes might miss key moments. To address this limitation, in future work, we plan to develop a reinforcement learning model that optimizes the switching timing between detection and tracking. The action of the agent will be two: selecting detection or selecting tracking at each timestep. The reward function can be designed as a combination of the camera instability score (negative reward) and the area of overlap score (positive reward). Furthermore, to further stabilize the transition between scenes, we plan to interpolate camera movement. This can be achieved by integrating rule-based smoothing into the post-processing stage.

Conclusion

In this work, we utilized a ViT-based object detection method to accurately detect engaging scenes. To solve the instability of camera control in learning-based observers, we present a hierarchical structure consisting of scene detection and tracking models to create a stable and fast automatic observer for Esports. This structure is inspired by the analysis of how human observers continuously provide engaging scenes to spectators.

Human observers identify the most interesting events in the overall state of the game and track these events, thereby providing engaging scenes to the spectators. Our proposed method employs a scene-detection technique to find interesting events throughout the game and a scene-tracking method to follow these events. This dual approach not only captures the engaging scenes but also ensures stable camera movement, closely mimicking the behavior of human observers.

The proposed method not only significantly enhances camera movement stability, but it also operates at a much faster speed compared to previous works that did not use a scene-tracking method. In the user test conducted to verify our computational evaluation in a real-world context, the model using our proposed tracking method received the highest overall user score compared to that without tracking.

Data availability

The datasets generated and analysed during the current study are available in the Starcraft_data repository, https://github.com/superfeyn/Starcraft_data/tree/main.

Received: 10 October 2024; Accepted: 10 March 2025
Published online: 21 March 2025

References

1. Gawrysiak, J., Burton, R., Jenny, S. & Williams, D. Using esports efficiently to enhance and extend brand perceptions-a literature review. *Phys. Culture Sport Stud. Res.* **86**, 1–14. https://doi.org/10.2478/pcssr-2020-0008 (2020).
2. Lokhman, N., Karashchuk, O. & Kornilova, O. Analysis of esports as a commercial activity. *Problems Perspectives Manag.* **16**, 207–213 (2018).
3. Cranmer, E. E., Han, D.-I.D., van Gisbergen, M. & Jung, T. Esports matrix: Structuring the esports research agenda. *Comput. Hum. Behav.* **117**, 106671. https://doi.org/10.1016/j.chb.2020.106671 (2021).
4. Marques, N. The role of breakthrough technologies in the growth of esports. *IEEE Potentials* **38**, 24–26. https://doi.org/10.1109/MPOT.2019.2893754 (2019).
5. Mattsson, B. P., Vajda, T. & Čertický, M. Automatic observer script for starcraft: Brood war bot games (technical report). arXiv preprint arXiv:1505.00278https://doi.org/10.48550/arxiv.1505.00278 (2015).
6. Joo, H.-T., Lee, S.-H., Bae, C.-M. & Kim, K.-J. Learning to automatically spectate games for esports using object detection mechanism. *Expert Syst. Appl.* **213**, 118979 (2023).
7. Han, Y. *et al.* Learning generalizable vision-tactile robotic grasping strategy for deformable objects via transformer. arXiv preprint arXiv:2112.06374 (2021).
8. Berner, C. *et al.* Dota 2 with large scale deep reinforcement learning. arXiv preprint arXiv:1912.06680 (2019).
9. Li, Y. *et al.* Mvitv2: Improved multiscale vision transformers for classification and detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 4804–4814 (2022).
10. Phang, D. W.-S. *Intelligent Camera Control in Game Replays*. Ms thesis, Lehigh University (2014).
11. Tot, M. *et al.* What are you looking at? team fight prediction through player camera. In *IEEE Conference on Computatonal Intelligence and Games, CIG*, vol. 2021-August, https://doi.org/10.1109/CoG52621.2021.9619038 (2021).
12. Zhang, S. *et al.* Faceboxes: A cpu real-time face detector with high accuracy. In *2017 IEEE International Joint Conference on Biometrics (IJCB)*, 1–9 (IEEE, 2017).
13. Tian, Y., Luo, P., Wang, X. & Tang, X. Deep learning strong parts for pedestrian detection. In *Proceedings of the IEEE International Conference on Computer Vision*, 1904–1912 (2015).
14. Kong, B., Zhan, Y., Shin, M., Denny, T. & Zhang, S. Recognizing end-diastole and end-systole frames via deep temporal regression network. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, 264–272 (Springer, 2016).
15. Li, L., Xu, M., Wang, X., Jiang, L. & Liu, H. Attention based glaucoma detection: a large-scale database and cnn model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 10571–10580 (2019).
16. Bertinetto, L., Valmadre, J., Henriques, J. F., Vedaldi, A. & Torr, P. H. Fully-convolutional siamese networks for object tracking. In *ECCV*, 850–865 (2016).
17. Li, B. *et al.* Siamrpn++: Evolution of siamese visual tracking with very deep networks. In *CVPR*, 4282–4291 (2019).
18. Xu, Y., Wang, Z., Li, Z., Yuan, Y. & Yu, G. Siamfc++: Towards robust and accurate visual tracking with target estimation guidelines. In *AAAI*, 12549–12556 (2020).
19. Dosovitskiy, A. *et al.* An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR* (2021).
20. Vaswani, A. *et al.* Attention is all you need. In *NIPS*, 5998–6008 (2017).
21. Carion, N. et al. End-to-end object detection with transformers. *Computer Vision-ECCV* **2020**, 213–229 (2020).

22. Zhang, H. *et al.* Dino: Detr with improved denoising anchor boxes for end-to-end object detection. arXiv preprint arXiv:2203.03605 (2022).
23. Meinhardt, T., Kirillov, A., Leal-Taixe, L. & Feichtenhofer, C. Trackformer: Multi-object tracking with transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 8844–8854 (2022).
24. Sun, P. *et al.* Dancetrack: Multi-object tracking in uniform appearance and diverse motion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 20993–21002 (2022).
25. Redmon, J., Divvala, S., Girshick, R. & Farhadi, A. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 779–788 (2016).
26. Jiang, P., Ergu, D., Liu, F., Cai, Y. & Ma, B. A review of yolo algorithm developments. *Procedia Comput. Sci.* **199**, 1066–1073 (2022).
27. Fan, H. *et al.* Multiscale vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 6824–6835 (2021).
28. Liu, Z. *et al.* Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 10012–10022 (2021).
29. Lin, L., Fan, H., Zhang, Z., Xu, Y. & Ling, H. Swintrack: A simple and strong baseline for transformer tracking. *Adv. Neural. Inf. Process. Syst.* **35**, 16743–16754 (2022).
30. Lv, W. *et al.* Detrs beat yolos on real-time object detection. arXiv preprint arXiv:2304.08069 (2023).
31. Ruby, U. & Yendapalli, V. Binary cross entropy with deep learning technique for image classification. *Int. J. Adv. Trends Comput. Sci. Eng***9** (2020).
32. Girshick, R. Fast r-cnn. In *inproceedings of the IEEE International Conference on Computer Vision*, 1440–1448, https://doi.org/10.1109/ICCV.2015.169 (2015).
33. Koch, G., Zemel, R., Salakhutdinov, R. *et al.* Siamese neural networks for one-shot image recognition. In *ICML deep learning workshop*, vol. 2, 0 (Lille, 2015).
34. Sifre, L. & Mallat, S. Rigid-motion scattering for texture classification. arXiv preprint arXiv:1403.1687 (2014).
35. Martins, A. & Astudillo, R. From softmax to sparsemax: A sparse model of attention and multi-label classification. In *International Conference on Machine Learning*, 1614–1623 (PMLR, 2016).
36. Wang, Q., Zhang, L., Bertinetto, L., Hu, W. & Torr, P. H. Fast online object tracking and segmentation: A unifying approach. In *CVPR*, 1328–1338 (2019).
37. Jiang, B., Luo, R., Mao, J., Xiao, T. & Jiang, Y. Acquisition of localization confidence for accurate object detection. In *ECCV*, 784–799 (2018).
38. Mann, H. B. & Whitney, D. R. On a test of whether one of two random variables is stochastically larger than the other. *Ann. Math. Stat.* **18**, 50–60 (1947).

Acknowledgements
This research was supported by Culture, Sports and Tourism R&D Program through the Korea Creative Content Agency grant funded by the Ministry of Culture, Sports and Tourism in 2024 (Project Number: RS-2024-00441523). This work was supported by Institute of Information and Communications Technology Planning and Evaluation (IITP) grant funded by the Korea government (MSIT) (2019-0-01842, Artificial Intelligence Graduate School Program (GIST)).

Author contributions
Ho-Taek Joo. wrote the main sections of the manuscript, including 2. Related Work, 3. StarCraft Dataset, 4. Method, and 5. Experiments. Sung-Ha Lee. contributed by writing 1. Introduction, 6. User Test, and 7. Conclusion. Insik Chung . contributed by writing 3. StarCraft Dataset, and 5. Experiments. Kyung-Joong Kim, as the corresponding author, reviewed and provided feedback on the entire manuscript. All authors reviewed and approved the final version of the manuscript.

Declarations

Competing interests
The authors declare no competing interests.

Additional information
Supplementary Information The online version contains supplementary material available at https://doi.org/10.1038/s41598-025-93692-0.

Correspondence and requests for materials should be addressed to K.-J.K.

Reprints and permissions information is available at www.nature.com/reprints.

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.