A Novel Learning-to-Rank Method for Automated Camera Movement Control in E-Sports Spectating

Hendi Lie[1], Darren Lukas[1], Richi Nayak[1], Jonathan Liebig[2]

[1] Queensland University of Technology, Brisbane, Australia
[2] Liebig Productions, Aachen, Germany
{h2.lie, lukasd, r.nayak}@qut.edu.au, jj@liebig.gg

Abstract. The popularity of modern competitive gaming (or e-sports) has skyrocketed in the past decade. A key part of e-sports is the spectating experience where fans watch tournament games through a camera of the observer. Bigger tournaments hire professional human observers with high-end tools to monitor important events in the game map for broadcasting the game. This setup is prone to errors. It results in missing important events within the game and lowers the spectating experience overall. It is also not sustainable in long-term and not affordable for the small-scale tournaments. This paper proposes a novel method of automated camera movement control using the AdaRank learning-to-rank algorithm to find and predict important events so the camera can be focused on time. The *Dota 2* game setup and its replay data are used in extensive experimental testing. The proposed method has shown to outperform the accuracy of both a past machine learning approach and a professional team of human observers.

Keywords: E-sports, Learning-to-rank, Automated Camera Control, *Dota 2*.

1 Introduction

Competitive gaming also called as e-sports, is a fast-growing industry with almost a billion dollar in revenue that attracts millions of dedicated players and fans world-wide [1–3]. Many tournaments covering different games, ranging from small online competitions to large international ones, are organised every day. The increasing interest in e-sports and exponential growth in fan-base have resulted in some of these tournaments to offer a gigantic prize pool. For example, Dota 2's International offers a total prize pool of US$24.6 million, exceeding the financial prestige of many traditional sports events such as The Masters in golf [1].

With more fans following the games of their favourite e-sports teams, broadcasting of competitive matches has become a key business of the industry. Large studios bid for tournament broadcasting rights and employ many casters (i.e., the game commentators) and observers (i.e., the camera operators tasked for observing and capturing important events in the game). In comparison to traditional sports like football where important events tend to occur around a single focal point (e.g. the ball), observing an e-sports game is more challenging as multiple important events can occur simultaneously on different parts of the game [4]. To solve this problem, studios commonly invest in

multiple dedicated observers, large screens, and the state-of-the-art visual and recording tools for the large tournaments. However, this setup is not affordable for smaller, online tournaments is not sustainable in long-term. Moreover, it is prone to missing important events, thus limiting the spectating experience for many fans.

As an alternative to human observers, automated methods have been proposed. Past approaches include using a machine learning model to predict the occurrence of important events, then apply a heuristic camera controller to focus on certain entities in the game [5]. These approaches have been reported to be inaccurate, missing many important events and reducing overall spectating experience [5, 6].

To provide a more accurate and affordable camera operation, this paper proposes a novel method based on learning-to-rank. Instead of predicting the occurrence of important events explicitly, we propose to rank entities based on their respective importance values. An entity's importance is calculated as the sum of related events values. We train an AdaRank [7] model to produce this ranking ahead of time and use a simple camera controller to focus on the top entities. To the best of our knowledge, this is the first work where the camera control in e-sports is predicted using a learning-to-rank model.

The proposed method is evaluated using a past matches replay data of a popular game, *Defense of the Ancients 2 (Dota 2)* [8]. We identified entities to be ranked as player-controlled *heroes* and produced a dataset annotated with important events retrieved from the game. Empirical analysis reveals that the proposed method outperforms a past automated approach and a professional team of human observers in accurately predicting important events and focusing the camera on top important entities ahead of time. This approach is a step forward in providing quality spectating experience for an audience of this growing industry.

2 Background and Related Work

E-sports is a term commonly used to refer to competitive video gaming coordinated by different leagues, ladders, and tournaments and watched by fans over internet broadcasting platforms [3]. There are many popular titles/games in e-sports, such as Dota 2, League of Legends, Fortnite, Counter Strike and Overwatch, bringing diverse games play and significant audience. Computer games and e-sports have been a subject of past machine learning research, focusing efforts in building game platforms [9], predicting winners of games [10] and building intelligent bots [11, 12].

While significant efforts have been devoted to building game platforms, and training players and bots in making intelligent moves, the research problem of spectating experience has not received much attention. Due to the overall increased interest in e-sports games, the exponential growth of fan base and wide availability of high-speed Internet access for video streaming [3], this problem has become increasingly important. Application of automated camera system is not new to traditional sports. For instance, Pixellot [13] is an automated camera system used in the broadcasting of a diverse range of sports including football, basketball, and rugby. It utilises multiple sensors and cameras which are connected to an autonomous server that controls the point of focus.

Specific to e-sports, many games have a built-in automated camera in their spectating system. For example, in *Dota 2* and *League of Legends*, there is a camera system available in spectator mode called directed camera [14, 15]. Although no official documentation regarding details of their approach can be found, we believe they implement a mixture of heuristics and machine learning algorithms to follow events in a game automatically. These built-in systems, however, are known to regularly miss important events and produce unsatisfactory spectating experience [6].

The following work comes closest to the proposed method. A K-Nearest Neighbour (KNN) model was used to perform event classification on game entities [5]. The predicted events are then used as an input to a heuristic-based camera controller. At a given timeframe, each game entity is labelled with up to 6 different game events. The dataset included the entity features computed based on movements, combat activities and damage received.

Based on visual empirical analysis with human viewers, the resulting camera movements in [5] were found to miss important events and jumped too much between entities in the game. Many viewers thought the default directed camera performs better. The model has shown overfitting and poor performance on test data as it was trained on the data of a single match only. In addition, the camera movement is highly dependent on a complex, game specific heuristic and not extendable to any other e-sports.

To overcome problems with past approaches, this paper offers a different perspective. Instead of explicitly finding and focusing the camera on important events, we focus on important entities involved in these events. We hypothesize that predicting the actual importance of entities is not important, instead, we use a learning-to-rank model to produce optimal ordering. The model is trained on large amount of past replay data to prevent overfitting. To ensure this method is extendable to other e-sports games, we implement a simple heuristic that rely on the top-ranked entities output by the model. This approach results in a camera that exceed accuracy of human observers and still produce smooth spectating experience.

3 Proposed Method

To enable accurate and smooth automated camera movements, we propose a method based on a learning-to-rank model. The proposed method (shown in Fig. 1) contains 3 components: 1) a game data parser; 2) an Ada-rank model, and 3) a camera controller.

In maximising spectating experience, the camera needs to show important events and related entities as they take place. We conjecture that the model should be trained to rank entities based on future importance. We define a parameter δ, time duration when future importance values of entities are computed. For each e-sport match, starting at timestamp $t = 0$, we first apply the game data parser to extract game logs, perform data pre-processing and produce features based on condition and interactions of entities at t. During the training process, we label each entity in the dataset with importance value based on events taking place between t and $t + \delta$. This allows the model to predict entity importance ahead of time. The most important entities output by the

model is then followed by the camera controller until $t + \delta$ and this process is repeated until the match ends. Different values of δ were tested to find the optimal value.

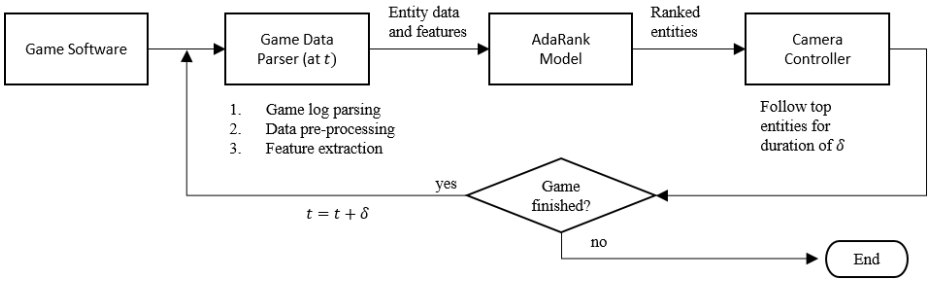

Fig. 1. Components in the proposed method.

3.1 The Learning-to-rank model

In observing an e-sports match, the camera is expected to follow entities involved in important events, such as scoring, killing opposition team or achieving a specific goal. Assuming importance values of all events can be correctly quantified, the importance of an entity can be determined by the sum of all events' importance the entity has been involved in. In predicting these important entities, we conjecture that the actual importance value is not significant, instead, we need to optimise the ordering of the top entities. Therefore, we model the problem of finding entity importance as the entity ranking problem and propose to solve it using a learning-to-rank model.

Learning-to-rank is a class of machine learning techniques to solve the ranking problem [16]. In Information Retrieval, it is commonly used to order a set of documents $\boldsymbol{D}$ optimally based on a query q by utilizing a ranking model $f(q, d), d \in \boldsymbol{D}$ [17]. Learning-to-rank is considered as supervised learning which requires document-query pairs labelled with relevance judgement. There are two types of data labelling in learning-to-rank: absolute judgment and relative judgement. Absolute judgment compares the document directly to the query independently to other documents, while in relative judgement, all relevant documents are ranked together based on preference [18]. We employed relative judgement in this work because the relevance of a game entity is relative to other entities.

There are three major approaches to learning-to-rank: 1) pointwise which aims to assign each document-query pair with a relevancy score; 2) pairwise that investigates ranking as a comparison over pairs of documents; and 3) listwise that tries to produce optimal ranking in the entire list of documents. We choose listwise approach as it has been shown to generally outperform the other methods [16] and it fits to the ranking problem of entities in an online game more naturally.

To formally define the problem, let $\boldsymbol{T}$ be the set of all timestamps sampled every time interval δ from an e-sport match. For each $t \in \boldsymbol{T}$, let $\boldsymbol{E_t} = \{e_{t,1}, e_{t,2}, \dots, e_{t,n}\}$ be set of n game entities to be ranked, $\boldsymbol{Y_t} = \{y_{t,1}, y_{t,2}, \dots, y_{t,n}\}$ be the set of label importance values and $y_{t,i} \in \boldsymbol{Y_t}$ be the importance label of $e_{t,i} \in \boldsymbol{E_t}$. Depending on the context of the game (e.g. games where entities could be summoned or killed), the value

of n could increase or decrease over different ts. In computing $y_{t,i}$, we consider all important events $\boldsymbol{v}$ occurring between t to $t+\delta$ that $e_{t,i}$ is involved in. This way of labeling allows the model to predict important entities before they are involved in future important events.

Suppose a feature vector $\mathbf{X_t} = \{x_{t,1}, x_{t,2}, \dots, x_{t,n}\}$ is created at every t using feature function ϕ, with $x_{t,i} = \phi(t, e_{t,i})$. We present a training example as $\boldsymbol{S_t} = \{(x_{t,1}, y_{t,1}), (x_{t,2}, y_{t,2}), \dots, (x_{t,n}, y_{t,n})\}$. The proposed listwise learning-to-rank model learns a ranking model $f(\boldsymbol{x_{t,i}})$ such that $\boldsymbol{E_t}$ is sorted by $\boldsymbol{Y_t}$ in descending order.

Objective Function. To produce an accurate entity ranking, we employed Normalised Discounted Cumulative Gain (NDCG) as the listwise objective to optimise. NDCG uses a graded relevance and discount function which allows a listwise algorithm to prioritise top-ranked entities in the optimisation process [19].

Let f be a ranking model trained on a dataset $\boldsymbol{S_t}$ as described before. Suppose $|\boldsymbol{E_t}|$ be the list of entities sorted according to their respective $\boldsymbol{Y_t}$. NDCG of f on $\boldsymbol{S_t}$ for top k ($k \leq n$) entities is computed as follows [12]:

$$NDCG_k(f, \boldsymbol{S_t}) = \frac{DCG_k(f, \boldsymbol{S_t})}{IDCG_k(\boldsymbol{S_t})}$$

Where DCG (Discounted Cumulative Gain) can be computed by:

$$DCG(f, \boldsymbol{S_t})_k = \sum_{i=1}^{k} \frac{2^{y_{t,i}} - 1}{log_2(i+1)}$$

And IDCG (Ideal Discounted Cumulative Gain):

$$IDCG_k = \sum_{i=1}^{|\boldsymbol{E_t}|} \frac{2^{y_{t,i}} - 1}{log_2(i+1)}$$

AdaRank. We implemented AdaRank [7] as the learning-to-rank model. Compared to other learning-to-rank algorithms, AdaRank can optimise any query-based performance measures with values in the range of [-1,+1] efficiently, by fitting the chosen NDCG optimisation function [7]. It also provides a practical framework for ranking which assumes that the document pairs for the same query are independently distributed. AdaRank focuses on training the top of document list and it treats queries, not document pairs. Thus, the algorithm is not biased toward queries with a higher number of document pairs, which is important in games where a number of entities n could change as the game progresses (such as first-person shooting games, where entities/players could die and get eliminated from the game).

3.2 Camera Controller

The proposed camera controller component follows a simple heuristic and takes the top-2 ranked entities as input. As per the heuristic, the top-ranked entity should always be on camera between two consecutive time frames, t and $t + \delta$. For the second top ranked entity, the camera controller follows how close it is to the first entity. If the two entities are within the reach of camera frame, the controller will focus on the midpoint of these two entities. An illustration of this heuristic is presented in Figure 2.

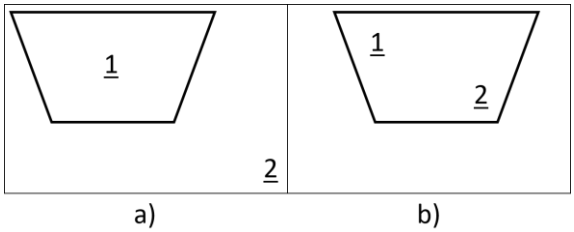

Fig. 2. A heuristic algorithm for setting camera focus given top-ranked entity (1) and second-ranked entity (2). At a), the distance between (1) and (2) is too large, therefore the camera only focuses on (1). At b), (1) and (2) are close, thus the camera captures both.

4 Empirical Analysis

4.1 Selection of E-sports Game

In this paper, we train the learning-to-rank model and build the camera controller based on *Dota 2 (Defense of the Ancients 2)*. *Dota 2* is a free to play MOBA (Multiplayer Online Battle Arena), released in 2013 by Valve Corporation. It is one of the most popular games in the world with a large and mature e-sports scene [1, 8]. This game is chosen due to its wide popularity as well the authors' sound knowledge of the game and its mechanics.

A *Dota 2* match has 10 players divided into 2 teams, the *Radiant* (green) and the *Dire* (red), each with bases located at different ends of the map. At the start of a match, each player chooses to play a *hero*. There are 115 heroes available to play, each with different abilities, attributes, and roles in the team. Each player can only choose one hero at the start of the game. The game is played on a large map with buildings (*towers*, *barracks*), *creeps*/*minions*, neutral *creeps* and the largest neutral creep called *Roshan*. The objective of the game is to destroy the opposing team's largest building called *Ancient* located in their base while keeping your team's *Ancient* alive. A *Dota 2* match usually lasts between 30-50 minutes.

Like traditional sport, fans can *spectate* live matches in *Dota 2* game client. At any given time, only a small portion of the map is visible to the camera, as illustrated in Figure 3. Everyone can control their own camera and focus on different parts of the map through mouse movement. However, with the vast size of the Dota 2 game map

and many events occurring simultaneously at the same time, it is difficult to watch relevant events. In a common tournament spectating setting, fans follow the movement of the tournament official observer camera.

Fig. 3. Dota *2* game map, showing bases of Radiant (green) and Dire (red) teams. The markers on the map represent towers (T), barracks (B), ancients (A), Roshan (R), arrows (5 Dire heroes) and teardrops (5 Radiant heroes). The green trapezium tagged by the blue oval illustrates a portion of the map available to see for a spectating camera.

4.2 Dataset

The dataset is constructed from the game logs of *Dota 2* matches taken place between October-November 2017 in *Dreamleague*. To acquire the relevant features, we downloaded replay files through Valve's API and OpenDota API [9]. Using these files, we extracted the following three main log files using the Clarity file parser [20]:

1. **Combat logs:** a detailed damage/heal information for every entity in the game including tick (server time), inflictor, receiver, the value of damage/heal, health value before and after the event.
2. **Life state logs:** a death/spawn information for heroes, buildings, and Roshan.
3. **Property change logs:** a history of property (e.g. health point, mana point, level, location and net worth) of heroes throughout the game.

We process log entries from combat and property change logs to generate features for each player's hero. Over every time interval δ, we produced critical features for each hero including health, mana, alive status, modifier status, distance from other important entities and damage/heal received and given. A total of 1502 numerical and binary features were generated. *Dota 2* server could produce log entries every tick (1 tick = 1/30 second), thus time interval value δ onward will be in tick unit.

We used these three logs to identify important events and assign a value/weight to each event and subsequently to each hero. For instance, the kill events are labelled by utilising hero spawn/death recorded in life state log and hero to *Roshan* damage are identified through significant damage Roshan received from that hero in the combat

log. There are many events considered as important in a *Dota 2* match, including hero kills/team fights, tower, and barrack destructions, and Roshan kills.

To quantify the importance of different events, we constructed an arbitrary importance structure in Table 2. The value of each type of event is set based on impact these events make to the game. For example, killing a hero eliminates it temporarily from the game and weakens the opponent team, while destroying a tower or barracks allow you to advance further to the *Ancient*. The significant combat damages indicate an attempt to fight and should be shown to the spectators. Different e-sports will require different events importance structure, but it is easily extendable.

Table 1. Dataset statistics

Statistics	Training + validation	Test
Matches	14	7
Avg. game length (mins:secs)	46:15	46:25
Dimension (rows x features)	1,000,000 x 1502	366,700 x 1502
Size (GB)	7.8	4.51

Lastly, the dataset was transformed to Microsoft LETOR (**Learning-to-rank**) format. Each row contains a tick as ID, hero features, and relevance labelling. As there are maximum 10 heroes in any given time t, we assigned a degree of relevance of 10 to hero $e_{t,i}$ with highest $y_{t,i}$, 9 to the next highest, subsequently until 1 degree of relevance the lowest. If there is more than one hero with same y_t, they are assigned same degree of relevance. Table 3 shows a sample transformed data of 10 heroes in a tick.

Table 2. Event importance hierarchy for *Dota 2*

Event type	Event name	Value
Death	Roshan	75.0
	Hero	60.0
	Tower/barracks	60.0
	Neutral	3.0
	Creep	0.5
Damage	Hero to tower/barracks	4.0
	Hero to hero	2.0
	Hero to Roshan	2.0
	Roshan to hero	2.0
	Tower to hero	1.5
	Hero to neutral	1.2
	Creep to hero	0.5
	Hero to creep	0.4
Other	Smoke usage	5.0

Table 3. A sample LETOR formatted row

hero	Importance	Query ID	F#1: Health	F#2: Max health		F#1501: Stunned	F#1502: Disabled
#0	1.0	qid:52190	1:800.0	2:800.0	...	1501:0.0	1502:0.0
#1	10.0	qid:52190	1:1061.0	2:1240.0	...	1501:0.0	1502:0.0
#2	9.0	qid:52190	1:276.0	2:1080.0	...	1501:0.0	1502:0.0
#3	8.0	qid:52190	1:1326.0	2:1340.0	...	1501:0.0	1502:0.0
#4	7.0	qid:52190	1:1001.0	2:1540.0	...	1501:0.0	1502:0.0
#5	1.0	qid:52190	1:880.0	2:880.0	...	1501:0.0	1502:0.0
#6	1.0	qid:52190	1:1415.0	2:1620.0	...	1501:0.0	1502:0.0
#7	1.0	qid:52190	1:1180.0	2:1180.0	...	1501:0.0	1502:0.0
#8	6.0	qid:52190	1:454.0	2:1080.0	...	1501:1.0	1502:0.0
#9	5.0	qid:52190	1:991.0	2:1100.0	...	1501:0.0	1502:0.0

4.3 Implementation

We used the AdaRank implementation from RankLib library [21]. All other components from the data parser to camera controller are implemented in Python 3.6 using standard libraries (e.g. numpy, pandas and scikit-learn) running on Windows OS.

4.4 Evaluation

In this work, we performed the two-stage empirical analysis. Firstly, we test different values of time interval δ. Lower δ allows the camera to change entities ranking more often and be more reactive to catch important events. However, very frequent camera movements could result in dizziness and unpleasant spectating experience, hence finding optimal δ is important.

Table 4. Hyperparameters settings

Hyperparameter	Values	Details
Time interval δ	{15, 30, 60 and 120 ticks} (0.5, 1.0, 2.0 and 4.0 seconds respectively)	Lower δ result in more responsive camera, at risk of dizziness and unpleasant spectating experience
For each δ		
Z-score Normalisation	{yes, no}	Features in the dataset are on different scale and variance.
Tolerance	{0.001, 0.002, 0.004}	Tolerance between two consecutive rounds of learning in AdaRank. Higher tolerance should result in less epoch/training iteration, preventing overfitting at risk of underfitting.

For each δ tested, we performed a grid search on a set of hyperparameters (detailed on Table 4) and evaluate each combination using NDCG@2 on 5-fold cross validation. The optimal set is used to produce the best performing model for that specific δ, then this model is evaluated on the test dataset. NDCG@2 is chosen as the implemented camera controller for *Dota 2* takes top 2 entities as input.

Next, to ensure the model and camera controller works well in capturing important events, we benchmarked the accuracy of the proposed method against 1) human observers and 2) KNN-based event classification method [5]. For human observers, we captured the *Dreamleague* 2017 observer team's camera movement data from the replay files. For the KNN-based method, as described by the author, we used K=5 to yield the best result.

5 Results

5.1 NDCG

Table 5 summarises the NDCG@2 evaluation values for various experiment settings. Our results show a gradual drop in NDCG@2 values from the best 15 ticks model to the 60 ticks model, followed by a significant drop for the 120 ticks model. Low δ value (15 ticks) allows the model to only predict the occurrence of events in short immediate future, resulting in the better ranking model. However, based on this result, using slightly higher δ (at 60 ticks) does not significantly reduce the performance and is preferable to produce reduce dizziness and produce a more pleasant spectating experience.

Table 5. NDCG@2 evaluation of best hyperparameter set for each δ

Time interval δ	Dataset	
	5-fold cross training and validation	Test
15 ticks (0.5 second)	0.7147	0.7222
30 ticks (1.0 second)	0.7095	0.7099
60 ticks (2.0 seconds)	0.6911	0.6892
120 ticks (4.0 seconds)	0.6354	0.6295

Table 6. Accuracy testing result on test data

Time interval δ	Total miss; % miss		
	Our method	KNN-based	Human observers
15 ticks (0.5 second)	118; 24.1%	179; 36.5%	85; 17.3%
30 ticks (1.0 second)	71; 14.5%	191; 38.9%	
60 ticks (2.0 seconds)	78; 15.9%	171; 34.8%	
120 ticks (4.0 seconds)	100; 20.4%	164; 33.4 %	
Total number of events			490

5.2 Accuracy

Table 6 shows the accuracy score evaluation of the proposed learning-to-rank model, KNN-based classifier (K=5) and professional human observers. Overall, the proposed method performs at a similar accuracy with human observers, with the 30-ticks and 60-ticks models even outperforming them. By following the top-2 ranked entities output by the models, the camera can focus on important events as they occur in the game.

The 15-ticks model missed more events than all other learning-to-rank models. With such small δ, the model could only capture limited number of events per time interval, resulting in sparse importance values and high variance error. Setting higher δ value could allow the model to capture more events in per time interval and generalise better.

The KNN-based classifiers do not perform well and are consistently less accurate than both learning-to-rank models and human observers. Other than accuracy, during the experiments, we found that the KNN approach is significantly slower in making predictions. As it needs to store all training examples, KNN produces a very memory consumptive model despite having significantly less features. Our method is faster, lighter and more suitable for producing predictions in real-time situation.

6 Discussion and Future Work

We have presented a novel method of the automated camera movement control for e-sports spectating. We proposed an innovative approach based on learning-to-rank, specifically using an AdaRank model trained on replay data, complemented by a simple camera controller heuristic. This method was tested using the *Dota 2* replay game data. We explored the effect of different time intervals δ and found that 60 ticks model produce good result while still maintaining reasonable reactiveness. The whole system (model + camera controller) was compared against a past machine learning approach and human observers and was found to be more accurate.

In future, we would like to expand the work. Firstly, since the *Dota 2* camera is embedded inside the game client software without documentation or API, we could only follow entities by mimicking keyboard presses to focus on certain heroes. With a free camera control which could move to any coordinate in the game, the camera can capture more non-hero related events, such as tower destroyed by creeps/illusions, resulting in a more extensive spectating experience.

Lastly, the produced automated camera system should be viewed from the spectators' perspective and therefore, it is advised to conduct surveys on the camera system to evaluate the spectating satisfaction. In addition, as our camera controller is implemented with *Dota 2* in perspective, we would like to expand the algorithm and test our method on games from a different genre, such as first-person shooters. Another possible expansion is to incorporate external features such as player's popularity or spectator's preference in prioritising camera focus.

References

1. Khan, I.: Dota 2's The International 7 breaks esports prize pool record, http://www.espn.com.au/esports/story/_/id/19861533/dota-2-international-7-breaks-esports-prize-pool-record, last accessed 2018/08/15.
2. Perez, M.: Report: Esports To Grow Substantially And Near Billion-Dollar Revenues In 2018, https://www.forbes.com/sites/mattperez/2018/02/21/report-esports-to-grow-substantially-and-near-a-billion-dollar-revenues-in-2018, last accessed 2018/08/15.
3. Hamari, J., Sjöblom, M.: What is eSports and why do people watch it? Internet Research. 27, 211–232 (2017).
4. Cook, M., Summerville, A., Colton, S.: Off The Beaten Lane: AI Challenges In MOBAs Beyond Player Control. arXiv Artificial Intelligence (cs.AI). (2017).
5. Phang, D.W.: Intelligent Camera Control in Game Replays, (2014).
6. Did you know that the directed camera is broken? : DotA2, https://www.reddit.com/r/DotA2/comments/3yt2yc/did_you_know_that_the_directed_camera_is_broken/, last accessed 2018/08/15.
7. Xu, J., Li, H.: AdaRank: a boosting algorithm for information retrieval. In: Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. pp. 391–398. ACM Press, New York, New York, USA (2007).
8. Valve Inc.: Dota 2, http://blog.dota2.com/, (2018).
9. OpenDota: OpenDota API, https://docs.opendota.com/, last accessed 2018/08/15.
10. Semenov, A., Romov, P., Neklyudov, K., Yashkov, D., Kireev, D.: Applications of machine learning in Dota 2: Literature review and practical knowledge sharing. In: CEUR Workshop Proceedings. pp. 1–5 (2016).
11. OpenAI: OpenAI Five, https://blog.openai.com/openai-five/.
12. Vinyals, O., Gaffney, S., Ewalds, T.: DeepMind and Blizzard open StarCraft II as an AI research environment, https://deepmind.com/blog/deepmind-and-blizzard-open-starcraft-ii-ai-research-environment/.
13. Tamir, M., Oz, G., Ridnik, T.: Method and system for automatic television production, (2017).
14. Fandom: Spectator Mode - League of Legends Wiki.
15. Dota 2 Gamepedia: Spectating - Dota 2 Wiki.
16. Liu, T.-Y.: Learning to Rank for Information Retrieval. Foundations and Trends in Information Retrieval. 3, 225–331 (2007).
17. Li, H.: A short introduction to learning to rank. IEICE Transactions on Information and Systems. E94–D, 1854–1862 (2011).
18. Niu, S., Guo, J., Lan, Y., Cheng, X.: Top-k Learning to Rank: Labeling, Ranking and Evaluation. Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval. 751–760 (2012).
19. Wang, Y., Wang, L., Li, Y., He, D., Liu, T.-Y., Chen, W.: A Theoretical Analysis of NDCG Type Ranking Measures. In: Conference on Learning Theory. pp. 1–26 (2013).
20. Schrodt, M.: Clarity - Comically fast Dota 2 and CSGO replay parser, https://github.com/skadistats/clarity, (2018).
21. Dang, V.: RankLib, https://sourceforge.net/p/lemur/wiki/RankLib/, (2013).