Extracting Gameplay Information from a Projection of 3D Modelled Objects using OpenCV

David Kang
M.S. Computer Science and Software Engineering
University of Washington Bothell
Seattle, Washington
dkang94@uw.edu

Matthew Munson
M.S. Computer Science and Software Engineering
University of Washington Bothell
Seattle, Washington
mmunson2@uw.edu

Sana Suse
M.S. Computer Science and Software Engineering
University of Washington Bothell
Seattle, Washington
sana96@uw.edu

Lalin Wachirawutthichai
M.S. Computer Science and Software Engineering
University of Washington Bothell
Seattle, Washington
lalinw@uw.edu

I. Introduction

The video game industry has increased in popularity in recent years: its market value growing from 93.1 billion USD in 2015 to 159.3 billion USD in 2020[9]. Not only has video game popularity increased, but the amount of people interested in watching a video stream of someone playing a game has also grown. This is known as streaming, where one person provides a live video feed of their gameplay for numerous others to watch. Streaming is novel in its facilitation of interaction between the audience and the streamer, allowing the person playing the game to react to the audience in real time [8].

Some of the most popular streaming platforms include Twitch, YouTube Gaming, and Facebook Gaming. These platforms continue to grow, with Twitch accumulating "1.49 billion gaming hours watched in April 2020" which was 50% more than the previous month [9].

Due to this popularity, it can be extremely useful to extract data from video game streams, where the obtained metrics can be used for a variety of applications. To prevent security vulnerabilities and cheating behavior most video games do not have an API available for third parties to access game data. This means that the most accessible way to obtain gameplay metrics is through the audiovisual stream outputted to the player.

Metrics extracted from a video stream can provide viewers with additional information about what they are watching, even allowing them to request certain information such as the number of times a streamer has won a game.

Currently, streaming platforms allow streamers to manually input the category that their stream fits into, including the specific video game or a descriptor of what they are doing. This allows the platforms to optimize and categorize search results and recommendations depending on the streamer's content. We propose that the audiovisual stream provided by the streamer could be analyzed to extract metrics that could automatically determine the category, specific video game, and descriptor of what the player is doing. This could potentially save time when creating a stream and provide streaming platforms with additional information to aid in the streamer's engagement as well as their gameplay performance.

The increasing popularity and competitiveness in online gaming has made cheating a serious problem. Most online games have built-in cheat detection using game variables, but analysis of a video stream could provide developers with an additional avenue for recognizing cheaters. Game developers currently play a cat and mouse game with dishonest players, as cheating tools are updated to work around the variables being tracked by the developers when detecting cheating. Analyzing a video stream provides additional metrics for cheat detection and may be more difficult to circumvent than other methods, such as measuring the speed at which a player aims.

Unlike multiplayer games where cheating affects large numbers of players and is therefore considered an important problem, cheat detection in single-player games is normally a very low priority for developers. However, cheating becomes a major problem in single-player games when they are played competitively such as in a speedrunning competition. In these cases, access to game data is not available to the speedrunning community moderators, so analyzing a video feed becomes the only method of cheat detection.

Outside of streaming services, this detection program could be used to assess a player's gameplay performance. Esports are becoming increasingly popular and bring a level of seriousness to video games that is akin to major league sports. Prospective Esports players spend countless hours training and extracting metrics from their play can be essential to gauging performance. An entire industry of video game coaching has begun to form, with personal one-on-one coaching as well as reviews of gameplay recordings to provide feedback on gameplay and decision making. Extracting metrics from video gameplay could allow a player or coach to see gameplay trends and could even allow for automatic gameplay evaluation.

For the scope of this project, we will focus on the team-based, multiplayer first-person shooter Overwatch, published by Blizzard Entertainment. Despite being released in 2016, the game remains popular at 60 million players as of 2021 - a ten million player increase since late 2019 [5]. There are 32 playable characters (called "heroes") available with each having a unique set of weapons and abilities. Counting the number of times that an ability is used provides a metric to measure a player's performance.

Overwatch features two-dimensional user interface (UI) elements that include a picture of the hero being played. This would be the easiest method for determining the hero being played, as image detection is highly efficient at recognizing static logos and text. To create a more novel and challenging use of image detection, it was decided to ignore the UI and instead attempt to identify 3D geometry. This consists of a 3D modelled weapon that translates and rotates slightly as the player moves and is subject to lighting effects created by the surrounding world. Each weapon is unique, correlating with a single specific hero. Identifying these weapons thus provides an avenue for identifying the hero currently being played. Another machine vision challenge is identifying when a weapon performs a certain action, such as firing. This involves a short animation, usually less than a second long. Our project is defined by the detection of a hero as well as the action that they are performing in Overwatch.

II. Related Work

Kozlowski, Korytkowski and Szajerman

A similar study analyzing gameplay video streams was written by Kozlowski, Korytkowski and Szajerman in 2020. The study used multiple algorithms to detect visual elements on the screen [11]. The authors first detected a 2D UI element which indicated that the player's weapon was currently displayed. This served as a signal to begin their 3D object detection algorithm, which would identify the 3D modelled weapon based on features similar to those in Overwatch. In this process, the authors took an average of 30 frames to eliminate minor movements in the video stream which could interfere with object detection.

This averaging of frames was found to be effective in reducing the effect of weapon movement but would also eliminate the possibility of identifying weapon actions, which often occur in a very small number of frames. The proposed project in this paper attempts to achieve the same accuracy in detection as previous work while analyzing each frame individually without blurring.

Mahendran, Bilen, Henriques, and Vedaldi

Another study looks at gameplay streams to retrieve metadata from the game by using annotated data to train models for object recognition, detection, and segmentation [6]. Unlike the proposed project, the authors modify an existing engine that replicates gameplay so that accurate gameplay metrics can be accessed to annotate the visual data and provide accurate ground truths. Since we are working with a video game that does not have such an engine already created, we will not be able to get perfectly accurate annotations.

Viola and Jones

A key reference for our project was the work done to establish object detection via a cascade of classifiers. Proposed by Viola and Jones in 2001, a framework for object detection was created with the intention of being applied to real-time applications [7].

At its core, detection in this framework was based on simple Haar-like features within an image. To improve the efficiency of calculating these images, the authors utilized the concept of an integral image. This is an alternative image representation which replaces the value of each pixel with the sum of all previous pixels from above and to the left. Using the integral image, the Haar-like features could be calculated in constant time, drastically increasing the speed of object detection overall [8].

The adaboost machine learning algorithm was used to identify the most selective features that had been created, and then form object classifiers from this input. The final aspect of the Viola & Jones methodology was to apply these classifiers to a target image in cascading fashion in order of complexity. This allows for the majority of negative target images to be eliminated through detection via simple classifiers, greatly improving the time complexity of the framework.

Their work was implemented in the form of facial recognition, and it returned very positive results. After training classifiers from over 6000 features and using a target image set containing over 500 faces, the authors found that their algorithm performed almost 15 times faster than other existing face detectors at the time, while maintaining similar accuracy results [7].

While the methodology was presented in the context of facial recognition, it can be easily adapted to any form of object recognition, with success depending on the strength of features identified in the image. In this project, these methods for detecting weapons and weapon actions.

III. Methodology

The proposed project consisted of a series of different components, each performing distinct tasks. These components would work together to provide all functionality necessary in detecting heroes from an input video stream. At its core, two different object detection algorithms were used: cascade classifiers and template matching. Separate tools were created to allow for compiling input, preprocessing target images, and evaluating system accuracy and performance. Fig. 1 showcases the connections between all of these components.

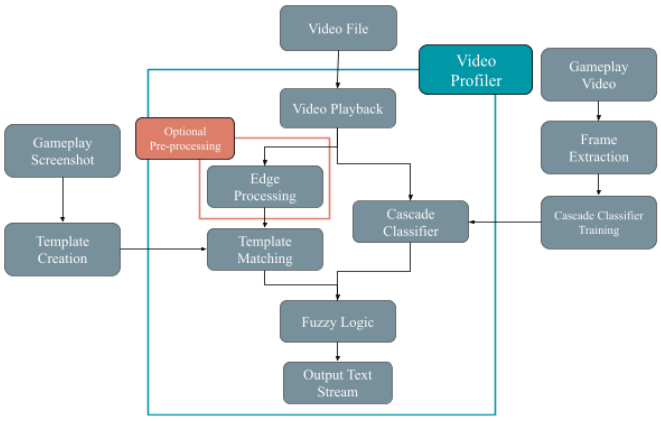

Fig. 1 System Overview

A. *Cascade Classifiers*

Using a cascade of classifiers for object detection was introduced by Viola & Jones in 2001, and it was an attractive option for this project due to its computationally efficient nature. Classifiers for both hero and weapon action detection were based on Haar-like features and were created via the Gentle AdaBoost algorithm. Classifier input was sourced from recordings of Overwatch gameplay with the user interface disabled.

Positive Samples:

Positive samples for each classifier were taken from input videos which attempted to consider the hero or weapon action across many different backgrounds. These videos were then passed as input to the Video Frame Extractor Tool, which split them into individual frames that could be used for classifier training. For each frame, manual identification of positive instances of the desired object was performed by drawing bounding boxes using the opencv_annotation tool. Once bound, all positive instances of the objects were compiled and used as input for the opencv_traincascade tool.

Negative samples:

In order to increase the variety of data in negative samples, input was sourced from videos of each different playable hero in the game. Frames were once again isolated using the Video Frame Extractor Tool, and the resulting data was used in classifier training. Whenever possible, positive images for heroes or weapon actions that did not correlate to the classifier being trained were used as additional negative input (i.e. the positive input for a mercy classifier was also used as negative input for the Lucio classifier).

Detected Hero	Positive Samples Count	Negative Samples Count	Stages Trained
Mercy	701	1980	20
Lucio	499	2136	20

Fig. 2: Input data used in classifier training for hero detection. Two classifiers were trained, one for the hero Mercy and another for the hero Lucio.

Detected Weapon Action	Positive Samples Count	Negative Samples Count	Stages Trained
Holding Staff	653	2423	20
Holding Pistol	378	2681	20
Healing	62	2550	15
Damage Boosting	127	2485	15

Fig. 3: Input data used in classifier training for weapon action detection. Four different classifiers were trained, corresponding to four actions that the hero mercy can take.

As seen in Fig. 3, positive data for the staff and pistol classifiers was much more plentiful than for healing and damage boosting. To effectively compare template matching and cascade classifier detection, this project was designed with only the first two actions in mind. The healing and damage boosting classifiers were trained after comparison data had been recorded. As a result, input data for healing and damage boosting was taken from previously recorded videos, limiting the total positive samples available. Due to this lack of data, evaluation results were not taken for either classifier.

The staff and pistol classifiers represented a more static form of classifier, as both actions only involve minimal movement throughout any given gameplay sample. The healing and damage boosting classifiers were created to explore more challenging detection classifiers - each of these actions involves many moving parts.

B. *Edge Matching*

Edge matching is an approach that utilizes points along an edge to form contours of an object. By comparing the contours of the object to the contours in the input image, the detection method is able to determine if the object exists in the input image. The edge-based approach is straightforward when detecting simple geometric shapes. However, it can be exponentially more difficult to match an object from a more complex image using its contours.

The intuition behind selecting edge matching as another methodology to identify a hero is the restrictive nature of the location and movement of the object we want to detect. Each hero in Overwatch can be identified by their equipped weapon(s). Each weapon has a fixed general location, viewing angle, minimal vertical and horizontal translation, and minute to no rotation. This combination of conditions eliminates a lot of parameters that make edge matching prone to detection errors or false negatives.

Hausdorff Distance is a well-known measurement used in edge matching. The intention was to compare the distances between all the points along the contours of the weapon in both images. If the distance comparison is low, it would imply that the two weapons are more similar than different [4]. The distance score would be able to determine which weapon is on the input edge map if a calibrated detection threshold is exceeded. Our implementation uses a one-sided calculation of the distance. This means that it is only necessary to calculate the sum of the minimum distances from all the points in the base image edge map to the nearest point on the input image edge map, and not vice versa. However, it was discovered that the distance between a point on the base edge map to its nearest point on the input edge map is not guaranteed to be a point that exists along the contours of the weapon when overlaid, even if the general location is the same. The point on the edge being compared may be a point along the edge of the scenery instead, which causes random errors in the distance comparison scores when calibrating a working threshold.

Adjustments to the application were investigated and the problem was transformed into a shape-matching problem using

the edges of the objects. It was possible to generate clean shape templates with the addition of the GrabCut tool and masks. However, applying the same treatment to the input image required discretionary tuning, which may vary greatly from image to image.

Edge matching was intended to be a strong contender for object detection in video games. As the game environment and player models are 3D rendered, little to no complications were expected when detecting edges for comparison. Nevertheless, that was not the case with 2D images of 3D rendered scenes. It is believed that the hyper-realistic characteristics in 3D render scenes, such as overly high illumination, clarity of all objects (both nearby and distant), and limited textures may have contributed to difficulties in accurately perceiving field of depth.

The matching results could not produce a stable and reliable threshold. Thus, it was deemed insufficient for use in our implementation. However, as the 3D rendered shots can capture contours with high clarity, the edge mapping process was incorporated into other detection pipelines as optional pre-processing.

C. *Template Matching*

Template matching is a methodology of finding the most probable location of a template in a larger source image. The method does so by scanning each row and column and seeing how each similarly sized region compares with the template. Due to the scanning of every row and column, the method itself can be computationally expensive, however it has been seen to produce great results. In addition, the search area can be limited to a specific boundary to reduce the volume of comparisons made.

Template matching is implemented in OpenCV and allows for six different methods of matching[1]. Of the six methods, two of them permit an optional mask parameter which identifies which regions of the template should be focused upon. This allows for focusing on specific sections of the template while ignoring others in comparison to the input image. To determine which matching methods work best, eight possible configurations were investigated with the goal of understanding which are best suited to identify weapons used in gameplay samples.

In addition to the matching methods, different preprocessing tools were investigated in an attempt to improve the detection method's accuracy. This included converting the input image to grayscale and detecting the edges in an image to be used for template matching. Converting the input image to grayscale is expected to improve the accuracy of object detection because it allows the program to ignore color-based matching and instead focus on the contours present within the input image. Likewise, isolating the detected edges within an image improves matching accuracy because the detected edges are assumed to be the major features of the image that we want to have detected.

The inputs for the template matching method provided by OpenCV are the input image, template image, matching method, and the optional mask image. In the proposed implementation, the input image is a single frame from a gameplay recording. Although previous approaches chose to average multiple frames from the video, it was decided to use single frames, as they would allow for more responsive detection during short weapon actions. The template images were cropped images of the frames identified as the best representations of an average weapon state when held by the hero. The mask used was created by obtaining the GrabCut of the template image. The GrabCut processing helped to identify the bounding perimeters of the weapon that was being detected.

D. *Evaluation Methods*

In order to evaluate the accuracy of the created object detection methods, the ground truth of each frame had to be determined. A tool was created to allow a user to label the hero and weapon action for each frame of a video. The labels were arranged into a CSV file where each row represented a frame and contained the hero and weapon action. This file could then be accessed at a later time, and the detected result of each frame compared against the ground truth. The number of correct and incorrect identifications could then be tallied and compared to determine the accuracy of the algorithm. This process was automated to compare against 30 videos, allowing for different object detection settings to be quickly evaluated and for the most effective settings to be identified.

Fig. 4: A screenshot of the Ground Truth Assessment Tool

E. *Performance Evaluation*

In order for a system to extract metrics from a video game in real time, it must be able to keep up with the frame rate of the video game. Most modern video games play at 60 frames per second, posing a considerable performance challenge to analyze at full speed. OpenCV's built-in video processing algorithms struggle to play 60 FPS videos without any additional frame processing, and average around 30 FPS

[1]https://docs.opencv.org/master/df/dfb/group__imgproc__object.html#ga3a7850640f1fe1f58fe91a2d7583695d

playback. Template matching was found to have the largest performance requirements and averaged around 1 frame per second during playback. Cascade classifier performed slightly better, averaging 4 FPS. Without modification, these values are too slow for real-time object detection.

IV. Results

A. Cascade Classifier Configurations

Detection via a cascade of classifiers was performed using the detectMultiScale method. Different levels of preprocessing were experimented with, and it was found that converting target images to grayscale and performing equalization each improved overall accuracy, with specific results shown in Fig. 5. It should be noted, however, that grayscale conversion may hinder detection when attempting to differentiate between two objects that are similar in shape but vary in color.

While other configuration options showcased occasional accuracy improvements, these results were not consistent. One notable example was limiting the minimum and maximum sizes of identified objects. Though this prevented potential false-positive results, false-negatives became more prevalent. A consistent threshold at which false negatives did not increase could not be determined, and the limitations were thus removed.

Hero Detection Average Accuracy (%)		
No Preprocessing	Grayscale	Grayscale + Equalization
48	55	74

Fig. 5: Hero Detection based on Cascade Classifiers

B. Template Matching Configurations

To evaluate the configurations to be used for template matching, the accuracy of each matching method was measured against multiple evaluation videos. An additional dimension was included in this investigation to understand what types of preprocessing methods improved the accuracy of the object detection. The preprocessing methods consisted of converting the parameter images to grayscale and detecting the edges in the parameter images. Fig. 6 demonstrates the accuracy of detecting heroes with template matching and the respective preprocessing methods.

Similar to the results with the cascade classifier configurations, it was expected that most of the accuracy measures would improve with any additional preprocessing methods, since they are conducted to simplify the detection process. However, this was not the case for all matching methods. This seems to be because of the specific ways that each matching method calculates their match scores.

One significant finding was that the use of mask images consistently improved the accuracy of the matching method. This is expected since the purpose of the mask images is to instruct the program which regions of the template image to focus on or not. There was one anomaly to this finding which was when the TM_SQDIFF matching method was used with edge detection; even when a mask image was used, the accuracy measure did not change.

From these results, it was deduced that the TM_CCORR_NORMED matching method with a mask image worked best for this use case, displaying 80.37% accuracy. Therefore TM_CCORR_NORMED with a mask image is used as the default configurations for any further analysis when template matching is used.

	Hero Detection Average Accuracy (%)		
Matching Method	No pre-processing	Grayscale	Edge Detection
SQDIFF	45.75	47.14	57.14
SQDIFF_NORMED	59.16	50.38	57.14
CCORR	57.14	55.76	62.91
CCORR_NORMED	65.75	49.48	72.20
TM_CCOEFF	57.40	55.25	76.80
CCOEFF_NORMED	55.66	51.45	77.69
SQDIFF*	64.67	67.75	57.14
CCORR_NORMED*	75.33	62.56	80.37

*With mask image.

Fig. 6: Hero Detection Accuracy with Template Matching Configurations

C. Hero Detection and Weapon Action Detection

In order to evaluate the overall accuracy of the system, the accuracy of each object detection method was measured against set evaluation videos that were not used to create any templates or to train the cascade classifier models. Fig. 7 describes the average accuracy of each detection method against specific video sets. The first two sets of videos are of single heroes, Lucio and Mercy, while the Switch video set is a spliced video of the two aforementioned heroes with random timing. The second and third column display the accuracy when detecting heroes, while the fourth and fifth column show the accuracy when detecting weapon actions for the hero Mercy.

The hero detection methods worked very well, at about 80% accuracy. This was consistent between template matching and cascade classifiers. The accuracy decreases by 10% for the next hero when using template matching. However, this decline is even more pronounced when using cascade classifiers, showcasing a loss of nearly 50% accuracy. It is unclear why this hero has a consistently lower accuracy when compared to the first hero.

With the switch video set, while cascade classifiers performed moderately well at 75% accuracy, it can be seen that template matching performed exceptionally well, at 97% accuracy. However, it should not be concluded that template matching can be used in the final product with this accuracy because of the contents of the switch videos. The videos in this set included mostly static views of the hero's weapon, where they may have been idle or walking, therefore they do not show

large changes to the weapon, allowing for template matching to easily detect a match in the frames of the video.

For weapon action detection, the detection methods were evaluated against videos of Mercy conducting two different actions - holding the staff or a pistol. The hero is able to do many other detailed actions such as firing the pistol and healing with the staff, but this system does not attempt to positively identify these actions.

It was found that neither detection methods performed well, with cascade classifiers showing 55% accuracy while template matching had 62% accuracy. This suggests that the very minute visual differences when the weapon action changes are hard to detect as compared to the entire weapon changing when the hero is changed.

	Average Accuracy (%)			
	Hero Detection		Weapon Action Detection	
Hero	Template Matching	Cascade Classifiers	Template Matching	Cascade Classifiers
Mercy	84	86	62	55
Lucio	75	50		
Switch*	97	75		

Fig. 7: Hero Detection and Weapon Action Detection Average Accuracy

Fig. 8: A screenshot of the final system working on gameplay footage

V. Conclusion

This project presents a network of systems and tools that can be used to identify heroes played and actions conducted in the video game Overwatch from visual information alone. Object detection is performed through two different pathways: via template matching and via a cascade of classifiers. Aside from these two systems, many different components have been created to support, enhance, and compile results for the project. These components have been integrated into this project to perform specific tasks but can also exist separately and may be used for other purposes.

Detection results were both positive and negative. When attempting to detect the hero Mercy, both template matching and cascade classifiers performed admirably, reaching detection rates of approximately 85%. However, both methods performed significantly worse when attempting to detect the hero Lucio, achieving results below any threshold that could be considered sufficient for consistent hero detection. When comparing the ways in which both methods were set up to detect Lucio, there does not appear to be any clear reason for this disparity in results. Because the difference is so stark and consistent across both detection methods, it presents a primary avenue for future work on this project. One potential consideration is that the two heroes selected (Mercy, Lucio) may be inherently good or bad use cases for the object detection methods used. If these methods were further applied to other Overwatch heroes, patterns may emerge in the evaluation data that lead to further understanding of what causes positive or negative results.

Computational performance of the project system was lackluster and well below an acceptable level for real-time application. This project was designed with the goal of evaluating object detection on every frame of a video, and thus considerations for performance improvements were not yet taken into account. Going forward, however, there are several different options that may speed up the system. First and foremost, frames can be skipped, allowing for uninterrupted playback with the drawbacks of delaying hero detection and potentially missing short weapon actions. Processing could be made asynchronous, allowing the video to continue playing. This would add some latency between display and detection. More efficient video access techniques could be used to reduce the amount of time accessing the video file. The GPU can be utilized to significantly speed up video decoding.

It may also be possible to update the methodology used for object detection to consider the result of detection from previous frames. Currently, detectors for both template matching and cascade classifiers iterate through all possible options to determine which hero or weapon action is present. However, most frames feature the same hero as the frame before them, which could be used to drastically cut down on overall computation time. Instead of checking against every classifier or template, the detectors could first check against the specific template or classifier that correlates to the detection result from the previous frame. Only if this detector returns false for cascade classifiers or returns a value beneath a calibrated threshold for template matching, would the detector iterate through the remaining options.

Despite the performance limitations and inconsistent accuracy, both object detection strategies presented in this paper provide pathways for understanding gameplay via a strictly visual data source. Template matching proved to be a more accurate method of detecting both the heroes and weapon actions tested in this project but was markedly slower than using a cascade of classifiers. Looking forward, there are many opportunities for growth and expansion on this project. The existing detection algorithms would require considerable optimization to be used in real-time processing environments. Nonetheless, the relatively high accuracy of individual frame analysis without any blurring means that it is possible for object

detection methods to reliably identify short actions such as a video game animation.

REFERENCES

[1] M. Consalvo, Cheating: Gaining Advantage in Videogames. The MIT Press, 2007. doi: 10.7551/mitpress/1802.001.0001.

[2] M. A. Pluss, K. J. M. Bennett, A. R. Novak, D. Panchuk, A. J. Coutts, and J. Fransen, "Esports: The Chess of the 21st Century," Frontiers in Psychology, vol. 10, p. 156, 2019, doi: 10.3389/fpsyg.2019.00156.

[3] X.-Y. Xu, X. (Robert) Luo, K. Wu, and W. Zhao, "Exploring viewer participation in online video game streaming: A mixed-methods approach," International Journal of Information Management, vol. 58, p. 102297, Jun. 2021, doi: 10.1016/j.ijinfomgt.2020.102297.

[4] "Hausdorff distance." http://cgm.cs.mcgill.ca/~godfried/teaching/cg-projects/98/normand/main.html (accessed Jun. 09, 2021).

[5] M. Park, "Overwatch gained 10 million players last year despite no new heroes or modes," PC Gamer, Apr. 30, 2021. https://www.pcgamer.com/overwatch-gained-10-million-players-last-year-despite-no-new-heroes-or-modes/ (accessed May 31, 2021).

[6] A. Mahendran, H. Bilen, J. F. Henriques, and A. Vedaldi, "ResearchDoom and CocoDoom: Learning Computer Vision with Games," arXiv:1610.02431 [cs], Oct. 2016, Accessed: Jun. 05, 2021. [Online]. Available: http://arxiv.org/abs/1610.02431

[7] P. Viola and M. J. Jones, "Robust Real-Time Face Detection," International Journal of Computer Vision, vol. 57, no. 2, pp. 137–154, May 2004, doi: 10.1023/B:VISI.0000013087.49260.fb.

[8] W. A. Hamilton, O. Garretson, and A. Kerne, "Streaming on twitch: fostering participatory communities of play within live mixed media," in Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, New York, NY, USA, Apr. 2014, pp. 1315–1324. doi: 10.1145/2556288.2557048.

[9] "The giants of the video game industry have thrived in the pandemic. Can the success continue?," Washington Post. Accessed: May 31, 2021. [Online]. Available: https://www.washingtonpost.com/video-games/2020/05/12/video-game-industry-coronavirus/

[10] "Video game market value worldwide 2023," Statista. https://www.statista.com/statistics/292056/video-game-market-value-worldwide/ (accessed May 31, 2021).

[11] K. Kozłowski, M. Korytkowski, and D. Szajerman, "Visual Analysis of Computer Game Output Video Stream for Gameplay Metrics," in Computational Science – ICCS 2020, Cham, 2020, pp. 538–552. doi: 10.1007/978-3-030-50426-7_40.