Towards automatic placement of media objects in a personalised TV experience

Towards Automatic Placement of Media Objects in a Personalised TV Experience

Brahim Allan, Ian Kegel, Sri Harish Kalidass

British Telecommunications plc, Ipswich, UK

Andriy Kharechko, Michael Milliken, Sally McClean, Bryan Scotney, Shuai Zhang

Ulster University, Belfast, Northern Ireland

Abstract.

In this paper we describe a Proof-of-Concept implementation of an automated system to drive one aspect of a personalised, object-based TV experience, particularly sports content, example football and rugby. Our Proof-of-Conceptuses a sequence of analytics processes, which can be performed offline or in real time, to allow a new media object to be automatically positioned on a multi-angle broadcast video sequence without occluding any key action, thus enabling additional graphic or video content to be used to personalise the broadcast for individual viewers. First, an object detection algorithm using a deep neural network model detects the players and ball, and its filtered output defines the region of interest within each frame of the video sequence. To avoid occlusion of key action by the media object, the remaining space within the sequence of frames is analysed to propose suitable locations. Our algorithm applies layout rules to ensure the object is placed in the best position based on broadcaster-defined templates (e.g. top-left, top-right, bottom-right and bottom-left). The output shows that our Proof-of-Concepts capable of processing video sequences with multiple camera angles and proposing the start time and duration of the media object without occluding any key action.

Keywords. Object Based Broadcasting, Deep Learning, Artificial Intelligence, Object Detection, User Experience.

1. Introduction

Traditionally, television and other audio-visual content has been delivered to viewers as a finished composition, including audio and video streams and, for example, the channel logo, subtitles and graphics – all delivered as a single stream. Object-Based Broadcasting – sometimes known as Object-Based Media – allows the content of TV programmes to be customisable for each individual. Figure 1, prepared by the BBC [1], illustrates the difference between the object-based and traditional approaches to TV production.

The objects relate to different elements that make up the media content, and thus, the overall television experience. Objects can be video and/or audio streams, and in the case of sports, for example, might also include individual camera feeds of crowds, players, managers, or referees, and may also include items such as graphic overlays (sports statistics, leaderboards, etc.). Decomposing the media content into multiple individually separate objects and providing these independently enables the viewers to effectively act as their own broadcast directors and to determine their own viewing experience whilst maintaining the producers' creative intent.

In 2021, UK media regulator Ofcom published a comprehensive review of trends in object-based media [2] which classified objects into three different kinds: Layers, which are consumed in parallel at particular moments on a timeline; Chunks, which are sequential segments of media consumed consecutively; and Transmedia objects, which can be consumed separately to add to an overall media experience. Transmedia objects, which include trailers, reviews, out-takes and social media posts, are widely available but rarely organised in a single database. Layers and Chunks, however, are the building blocks of the personalised viewing experience and their presentation is governed by timeline and layout rules. Historically, it can be considered that synchronised subtitles overlaid on broadcast TV programmes were the first Layer objects, and accessibility features remain a key use case for object-based media today, involving audio, video and graphic layers. Chunk objects are the enablers of highly personalised programme content such as interactive narratives and individually-tailored news and documentary shows. However, for live broadcast content such as live sports, personalisation is achieved

predominantly through the use of Layer objects while viewers simultaneously share the same match narrative as it is played out for them. An important design consideration in any object-based media system is how the control of personalisation is shared between the programme's director and the individual viewer. Unless frequent interaction is expected which results in an immediate effect (for example in a chunk-based Netflix interactive documentary [3]), personalisation features would usually be expected to require only very occasional interaction, such as to configure indirect preferences, in order to avoid distraction from the crafted narrative or live event. For the director and content rights holder, this increases confidence that their professional standards and house style will be preserved. However, it is important to note that this new field is developing rapidly as more content is produced for AR and VR platforms in which participants inherently have considerably greater control over their viewing experiences.

Data analysis and Machine Learning techniques could enable new object-based personalisation features by augmenting the production process using knowledge models and real-time analytics. For example, by applying object detection to the live broadcast TV stream, the resulting metadata could be used to automate the presentation of additional supplementary graphics with content based on viewers' personal preferences. Our goal is to assess and validate whether dynamic graphic placement is feasible on live content while maintaining acceptable user experience. The aim is to take advantage of the inherent delay in adaptive video streaming to 'look ahead' in the live stream by 30-60 seconds and make graphic placement decisions accordingly

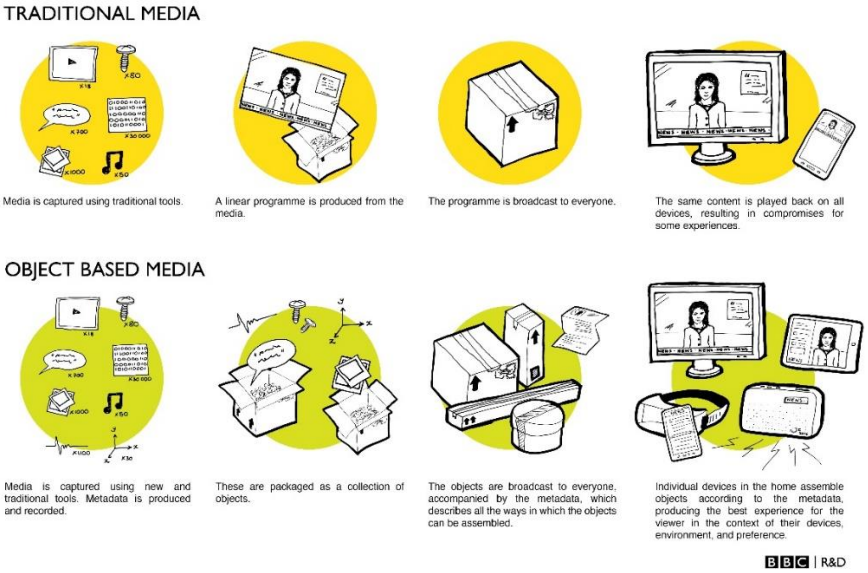

Figure 1. Traditional vs Object-based Media **[1]**.

Applying Artificial Intelligence to object-based personalisation could allow us to create a more effective user experience using emerging fast-developing technologies such as deep learning to enable many different personalised versions to be created, which could never be directed manually by the broadcaster. Combined with the power of cloud services, these could deliver the best possible TV experience while remaining cost-efficient to broadcasters and content service providers.

This paper describes the initial results of a project being carried out by BT Applied Research in collaboration with Ulster University (as part of the BT Ireland Innovation Centre). The project seeks to identify data analysis and Machine Learning techniques, including object detection, which could support the delivery and quality assurance of advanced personalisation features within future object-based TV experiences. If successful, new AI-supported software would be incorporated within the broadcast production chain, client devices or TV platforms in line with the deployment of new Object Based Broadcasting (OBB) features.

In this paper we describe the design of this Proof-of-Concept and the results that we obtained. Firstly, we describe related work and explain the benefit of object detection models and their different architectures. We then describe the use case and the objectives of dynamic graphic placement for live football, and explain in detail the method we have developed. We then present the results of the Proof-of-Concept applied to a recorded replay video sequence, and describe a Proof-of-Concept implementation which performs dynamic placement on

a live video feed when requested by a viewer. Finally, we provide our conclusions and identify the next steps for this research.

2. Related Work

2.1 Object-Based Broadcasting

Having been the subject of research by BT, the BBC, and others for several years, Object-Based Broadcasting is starting to be adopted as a means of differentiating mainstream TV production, made possible by the widespread availability of devices such as Smart TVs, media streamers, game consoles, and Set-Top Boxes which are capable of processing and rendering media objects to create a broadcast-quality experience.

OBB provides viewers with a more personalised relationship (including the opportunity for direct interaction) with the content which facilitates tackling issues of accessibility and device heterogeneity, curation of programmes and creation of potentially new forms of content and experiences as well as delivering efficiencies in the production process. However, OBB also requires changes to the production workflow as well as client devices and the platforms which manage them. [4] describes a platform for the production, delivery and orchestration of object-based experiences developed by the EU-funded H2020 ICT project 2-IMMERSE, as well as a case study of a prototype multi-screen experience for MotoGP which included a wide range of personalisation features, including the placement of supplementary content over the race video feed. In contrast, [5] describes a prototype object-based learning experience developed by the BBC based entirely on recorded media, which illustrates the importance of new data models and production tools which support the temporal arrangement of media objects to form a personalised narrative. In [6], the authors explain the challenges of delivering the end-to-end production of a personalised object-based broadcast of a live football match from Wembley Stadium. The work reported in this paper builds on the object-based graphics features developed for this live football production. In [7], Ericsson Research describe an alternative technique for triggering and controlling the rendering of personalised graphics overlays on a client device by using timed events embedded in the MPEG-DASH stream which is being used to deliver live or recorded video.

2.2 Artificial Intelligence and Machine Learning in Broadcasting: Object Detection

AI and ML techniques are playing increasingly crucial roles in broadcast production workflows in order to reduce cost and increase personalisation. A recent ITU report [8] highlighted the wide range of applications:

- Workflow and bandwidth optimisation,
- Automated content creation,
- Content selection for targeting audience demographics,
- Optimization of asset selection – metadata creation,
- Dynamic product placement and content personalisation,
- Advertising for broadcast.

The automated detection and recognition of objects within audio and video content features among many of the use cases cited within these categories.

Object detection models are primarily based on supervised learning using deep neural network algorithms that take in an RGB image as an input and output the predicted labels and bounding boxes for a set of detected objects. Initially, we considered other approaches to people detection [9] [10], like the support-vector machine [11] with a linear kernel applied to a grid of local histograms of oriented gradients [12] or detection using aggregate channel features [13], but on our tasks their accuracy was greatly outperformed by deep neural networks.

Object detection models are primarily based on supervised learning using deep neural network algorithms that take in an RGB image as an input and output the predicted labels and bounding boxes for a set of detected objects. Object detection models generally require higher resolution input images compared to image classification models, to perform the recognition and localisation of the detected object. Research [14] conducted on single-stage and two-stage detectors indicates that the input image resolution does have an impact on the model's performance. It must also be noted that the input image will be resized to match the input layer dimension, irrespective of the image size supplied for inference. The findings indicate that the single-stage detector such as You Only Look Once (YOLO) version 3 performs better on low resolution detection of small

objects, whereas the Faster Region-Based Convolution Neural Network (Faster R-CNN) two-stage detector performs better on both high resolution and low-resolution images in detection of medium and large-sized objects. It was also found that the object detection models containing the ResNet backbone were severely affected with higher resolution images, where the computation time drastically increased. In a summary, the higher resolution images achieved higher accuracy in detection with a comparable increase in computational time, if not drastic, and vice versa.

There are two main types of object detection model architecture [15]**,** as shown in Figure 2.

a) *Single-stage detectors.* In single-stage models, the image classification and bounding box generation for identifying objects are performed in a single step which allows them to demonstrate higher inference speed but lower accuracy as compared to two-stage detectors. The examples include SSD [16] and YOLO [17] object detection models.

b) *Two-stage detectors*. Performing the object detection process in two stages offers high localisation and recognition accuracy. In the first step, the image goes through a Convolutional Neural Network (CNN) where the regions of interest - also known as regional proposals - are generated. The regional proposals use a selective search algorithm to identify objects in the image, yet they remain unclassified (i.e. unlabelled). In the second stage, the regional proposals are entered into a pipeline for object classification and bounding-box regression. The Faster R-CNN object detection model is an example of a two-stage detector.

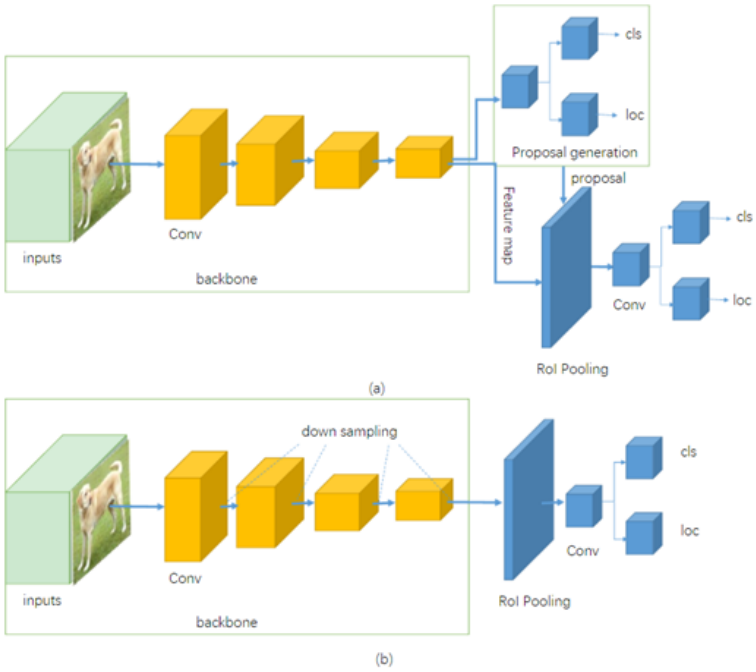

Figure 2. One-stage and two-stage Object Detection model architecture **[18]**

The above Figure 2 illustrates a single-stage object detector (a) and two-stage object detectors (b). The Input layer is the first layer in any deep learning model and it will be followed by a backbone layer that computes features from the input image. VGC and ResNet are widely adopted backbone architectures also known as feature extractors.

According to the research conducted [19] by a medical imaging team in China for pill detection using Single-stage (YOLO) and two-stage detectors (Faster R-CNN), the accuracy of two-stage detectors is found to be higher than single-shot detectors, however, the single-stage detectors are faster and therefore, there is a trade-off between inference speed and performance. Research conducted in 2021 by the University of Sevilla on real-time single-stage and two-stage detectors [14] illustrates similar findings, i.e., the single-stage detectors such as FCOS and YOLOv3 outperform two-stage detectors such as Faster R-CNN in inference timing (per second). Although, the accuracy of two-stage detectors is higher in general, more specifically the Faster R-

CNN combined with Res2Net-101 backbone, offers enhanced accuracy and real-time inference, thus providing a good trade-off.

To evaluate the different types of object detection models, it is important to understand the key object detection model metrics. Mean Average Precision (mAP) is a widely-adopted metric which compares the actual ground truth bounding box with the generated bounding box for the detected object. The mAP is generated by calculating the Average Precision (AP) for each class and then evaluating the mean of all APs, which gives the mAP of the object detection model. The AP is calculated from the precision-recall curve as the weighted mean of precisions achieved at each threshold, with the increase in recall from the previous threshold used as the weight. The precision and recall metrics need to be calculated to generate the precision-recall curve, which produces the AP values at different thresholds. Precision measures how accurate the predictions are, i.e., out of the total predictions made, it gives a percentage of correct predictions. Similarly, recall measures how good the model is in detecting all positives, i.e., it gives a percentage of positive predictions out of the total top K predictions made. For example, if a given frame contains 10 football players and the model only detects and labels one football player then its precision is high, however, very low recall as only one out of ten football players was detected.

The University of Sevilla's research [14] benchmarks different object detection models using mAP and indicates that there is a trade-off between single-stage and two-stage detectors' model performance and inference speed and this is primarily affected by the model architecture, backbone and input image resolution. The distinction between model performance (mAP) of the single-stage and two-stage detectors on small, medium, and large objects is interesting. The Faster R-CNN model combined with state-of-the-art backbones has greater mAP on small, medium, and large objects detected and similarly, while YOLOv3 tends to have the highest mAP on smaller and medium objects on single-stage type detectors.

We are comparing the overall performance of our dynamic graphic placement Proof of Concept by experimenting with one single-stage detector object detection model (YOLOv3) and a two-stage detector object detection model (Faster R-CNN). We are considering the YOLOv3 object detection model because, according to [20], it should perform significantly better in the detection of True Positives (TP) compared to SSD, even though SSD produces a much-reduced number of False Positive (FP). It was also found that SSD had higher precision and lower recall and YOLOv3 produced comparably balanced precision and higher recall metrics.

As part of our experimentation we will undertake fine-tuning, which includes not only the refinement of model parameters for enhancement of the model performance but also experimenting with different combinations of backbones that can benefit our Proof of Concept.

The Faster R-CNN models offer inferencing speeds up to 7fps, whereas the YOLOv3-based single-shot detection models can reach up to 45fps on the same data set [21]. The other key factors governing the inference speed include access to GPU-based inferencing, model architecture, and input image resolution.

The NVIDIA's GPU-accelerated libraries offer enhanced performance and during the deployment phase, considering the scalability of the solution, GPU-powered infrastructure would be preferred to harness accelerated performance. Currently, our Proof of Concept is developed and tested using NVIDIA's GPU development kit, the Jetson XAVIER NX, which offers powerful GPU computation with CUDA-accelerated libraries.

The computational power offered by CUDA cores and NVIDIA's libraries such as TensorRT accelerates the inference speed of state-of-the-art object detection models including Faster R-CNN and YOLOv3. The table below illustrates the inference speed achieved by employing NVIDIA's GPU and deep learning toolkits:

			Jetson Nano	Jetson Xavier NX			Jetson AGX Xavier			T4
Model Arch	Inference resolution	Precision	GPU (FPS)	GPU (FPS)	DLA1 (FPS)	DLA2 (FPS)	GPU (FPS)	DLA1 (FPS)	DLA2 (FPS)	GPU (FPS)
YoloV3 - ResNet18	960x544	INT8	11*	78	55	55	223	84	84	620
FasterRCNN - ResNet10	480x272	INT8	16*	127	N/A	N/A	281	N/A	N/A	932
SSD - ResNet18	960x544	INT8	10.6*	124	56	56	216	77	77	760
DSSD - ResNet18	960x544	INT8	9*	66	45	45	189	67	67	586
RetinaNet - ResNet18	960x544	INT8	8.5*	60	45	45	147	41	41	296
MaskRCNN - ResNet50	1344x832	INT8	0.6*	5.4	3.2	3.2	9.2	4.5	4.5	24
Note: * - Inference using FP16 on Jetson Nano.										

Table 1 – Object detection model performance on NVIDIA GPU's using DeepStream SDK [22]

Object detection tends to be easier for detecting single objects in an image frame, however it becomes challenging for detecting multiple objects in a frame where the detected objects can overlap with successive detections and cause similar challenges for deep learning model to detect and identify multiple objects. To facilitate multiple successive detections, state-of-the-art single-stage object detection models generally employ the *Anchor box method* for generating bounding boxes for predicting the object label and location. Anchor boxes are custom-sized boxes of different ratios of height and width designed to match the relative ratios of the object classes being detected. The anchor boxes are of varying sizes and fine-tuned to match the size of objects of different classes being trained [23].

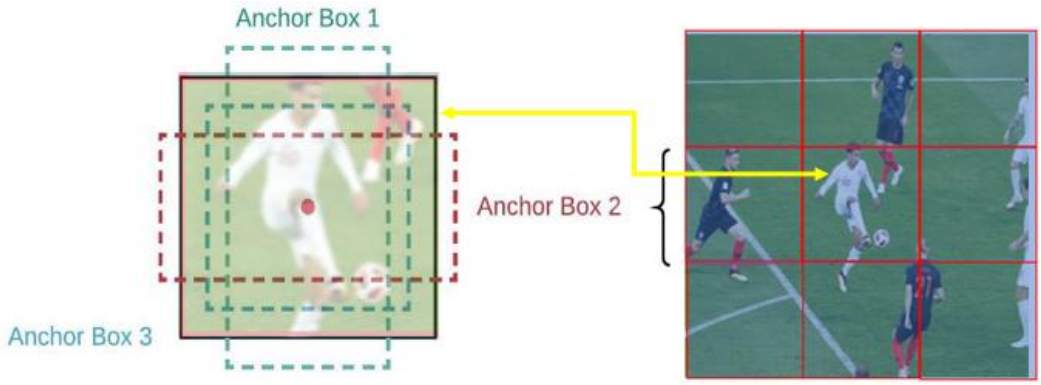

Figure 3. Example illustrating Anchor box method.

When the input image is passed through the object detection model, a grid of a fixed size is applied to the image, and the anchor boxes of different sizes are applied on the grid to detect and identify the objects. The number of anchor boxes varies from 2 to 5 and usually depends primarily on the type of data.

3. Use Case: Dynamic Graphic Placement for Live Football

The introduction of features which enable the appearance of broadcast presentations on individual screens to differ significantly from each other presents a challenge for a broadcaster seeking to guarantee a high production quality. Personalised media objects which are requested by customers must be consistent with the broadcaster's house style and, crucially, their timing and screen location must not obscure key action in the video beneath.

The use case addressed in this paper is to automate the choice of timing and screen location for the insertion of a personalised media object – a graphic overlay – within a football match broadcast. The graphic overlay might, for example, show personalised player or match statistics at appropriate times during the broadcast, such as after key events such as goals or near misses, or in response to events taking place in other simultaneous matches.

Detailed data for a wide range of sports is available to broadcasters and content providers on a near-live basis from suppliers such as Stats Perform (Opta) [24], and personalised media objects can subscribe to subsets of this data as required and defined by the broadcaster. Furthermore, should sufficient data not be available from an external supplier, a broadcaster may choose to author it themselves as part of their production workflow.

Earlier, we explained how personalisation features such as this would usually be expected to require minimal interaction to avoid distracting the viewer from the live broadcast. For this use case, a viewer could be expected to set up a simple profile in advance, specifying, for example, their favourite players and teams. This would preferably be done using apps on personal devices (such as tablets and smartphones), while a simpler option might be provided on their TV device to enable or disable additional personalised graphics.

The decision-making of the graphics placement is a complex process which could further be improved by taking into consideration of the analytics of the game, for example, the area of interest of the play and the direction of the play in the game. The proposed approach in this paper is a fundamental building step to firstly ensure that no key action involving players and the ball is occluded by the choice of the placement of graphics overlay. The current framework of the decision-making process is flexible to be extended to apply varying weightings to objects (players/ball) depending on their involvement in the area(s) of interest. This can further provide the current algorithm the possibilities to find additional suitable locations for graphics placement if the relaxed criteria to avoid occlusion on key actions. In contrast to the player and match data used within a personalised graphic overlay, it cannot be assumed that technical data from parts of the broadcast production process can be easily made available to support this decision-making process. A wide variety of different systems are used across the industry to process and switch video, and sometimes to track the positions of players and the ball. Our approach in this use case is to develop a general technique which is widely applicable across televised field-based team sports.

Adaptive bitrate streaming is widely used by broadcasters to deliver live and on-demand content as it makes use of established web standards and caters for the widest range of devices and usage contexts through its ability to dynamically switch between different versions of a stream, varying by resolution and bitrate, for example [25]. To achieve reliable and smooth playback, most broadcasters and content providers configure their players to use buffers of significant size, resulting in latency of 30 seconds or more. Benchmarks are not usually shared publicly, but the existence of streaming delay is widely reported in consumer surveys [26].

To address our use case, we can take advantage of the delays introduced by the widespread use of adaptive streaming, hence enabling an appropriately-located analysis system to ‘look ahead’ in the live stream by 30-60 seconds and make graphic placement decisions accordingly. If these delays are reduced as greater network reliability removes the need for some buffering, the use case can still be applied to non-live elements of a match broadcast, for example placing the graphic overlay during a replay sequence in which recordings of multiple camera angles of a key event, such as a goal being scored, are shown consecutively.

The player detection and ball detection are both of interest to this research, however, it should be noted that the automated proposition of location for placement of graphics is based on the detection of players. The ball detection and tracking are considered in future scope for the overall optimisation of the system. Further, there are interesting challenges to the detection of the ball, compared to the detection of players and this is primarily caused by the size of the object of interest (football) that needs to be detected on the live stream. According to our initial research on the accuracy-based performance of object detection algorithms, it was found that these models have different AP (Average Precision) for detection of the same object of different sizes such as; small, medium and large [27]. Therefore, these initial findings indicate that an optimal object detection model with higher AP on different sized objects must be selected and further fine-tuned with multi-scale training to produce a robust model that relatively has less impact due to varying object sizes.

3.1 Objectives

Our goal is to provide a personalised graphic overlay to be positioned in a multi-angle video sequence without occluding any key action. This implies several objectives:

1. To apply video analysis to ensure that the graphic is placed in an area where there is least occlusion of the key action.
2. To automate the detection of boundaries between parts of the sequence shot from different camera angles, to enable the shot type to be taken into account when making the placement decision.

3. To demonstrate insertion of the graphic for a fixed period of time and on particular shot types only (such as wide-angle shots).
4. To demonstrate a system that places the graphic in the best location based on templates defined by the broadcaster e.g. top-left, top-right, bottom-right and bottom-left (Figure 4). This is essential to ensure compliance with their production style and to avoid clashes with non-object-based graphics which may be present.

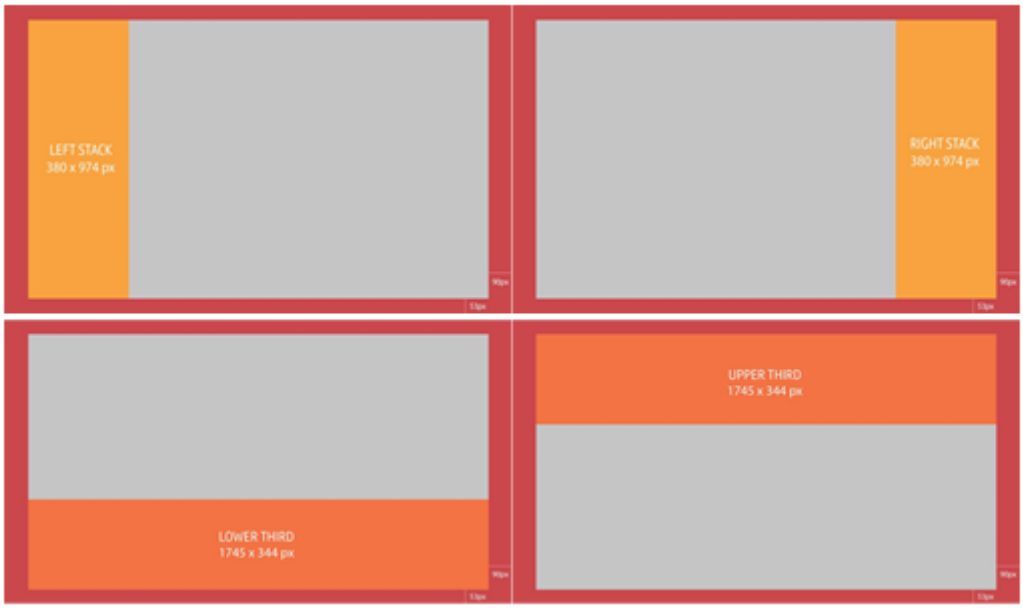

Figure 4. Standard frame areas to display graphic overlay: left stack (top left), right stack (top right), lower third (bottom left), upper third (bottom right).

3.2 Method

The object detection model deployed for our use case is a single-stage pre-trained model, implemented using the Nvidia DeepStream framework [28]. The model is constructed with the ResNet10 backbone (i.e. feature-extracting network). It is a 4-class object detection model capable of detecting the following types of objects: *Vehicle*, *RoadSign*, *TwoWheeler* and *Person*. The model uses the Anchor Box method to generate two tensors, *cov* and *bbox*. The image is divided into 16x16 grid cells [29]. The *cov* tensor (shortened for "coverage" tensor) defines the number of grid cells that are covered by an object. The *bbox* tensor defines the normalized image coordinates of the object top-left and bottom-right coordinates with respect to the grid. Each class has its own *cov* and *bbox* tensors.

It is assumed that in the football video sequence, the region of interest (ROI) within a video frame is defined by positions of the players and the ball provided by the object detection model. As shown in Figure 5, for every frame the model generates bounding boxes around detected objects, together with their class labels and probabilities.

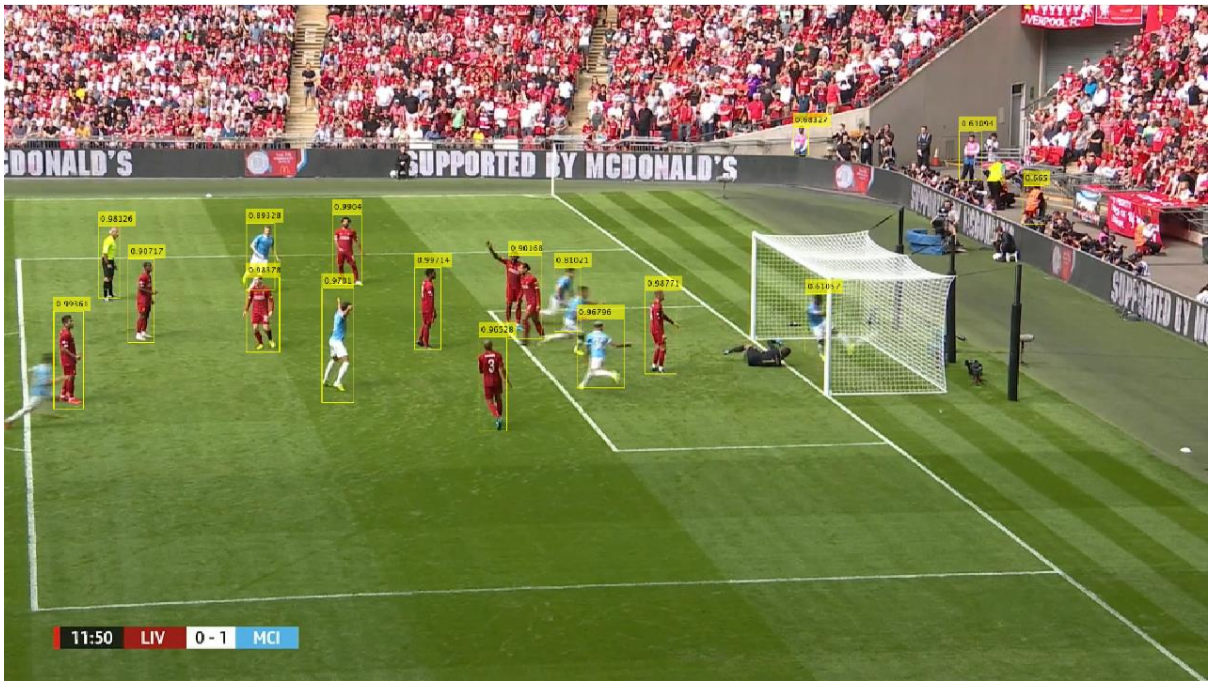

Figure 5. Players detected in a frame of a football video.

Our method for dynamic placement of the graphic comprises the following steps:

1. **Searching for valid locations**:
    - Defining a map of available locations for the graphic within a frame.
    - Updating the map of locations with every new frame depending on the detected objects in that frame by excluding every location which may overlap with any of them and incrementing the durations of availability of the others.
    - Producing a list of locations which are continuously available for at least the minimal required display time together with the start times and durations of their availability.
2. **Optimisation**:
    - Finding locations from the list which satisfy target criteria (e.g. have the longest availability, are vacant from the first frame of the buffer, belong to a certain pre-defined region etc.).
3. **Decision-making**:
    - Selecting a location subject to production practice, artistic intent and viewers' preferences.
    - Suggesting the start time and duration of the graphic based on the timeline of detected occlusions and other criteria (e.g. production considerations).

The first stage of the dynamic placement process is to extract a description (e.g. a set of bounding boxes) of the ROI within each frame of the video sequence in order to estimate and to minimise the risk (or the size) of occlusions at every possible location of the graphic.

The ROI is mapped onto a binary image where all pixels which belong to relevant detected areas are marked (i.e. a heatmap), and our task is to find the best position (which minimises occlusion of the game) within the unmarked (or, in other words, free) space to place the graphic. A sequence of such heatmaps produced for a video clip (Figure 6) demonstrates free space available within every frame together with duration of its availability. However, generating binary heatmaps and storing them as bitmaps is not efficient in terms of memory use and may become computationally prohibitive. Instead, bounding box data are stored for each frame in a fixed-length queue (of the same size as the video buffer) while the algorithm (Algorithm 1) uses them as raw input to search for eligible locations within each frame to display the graphic.

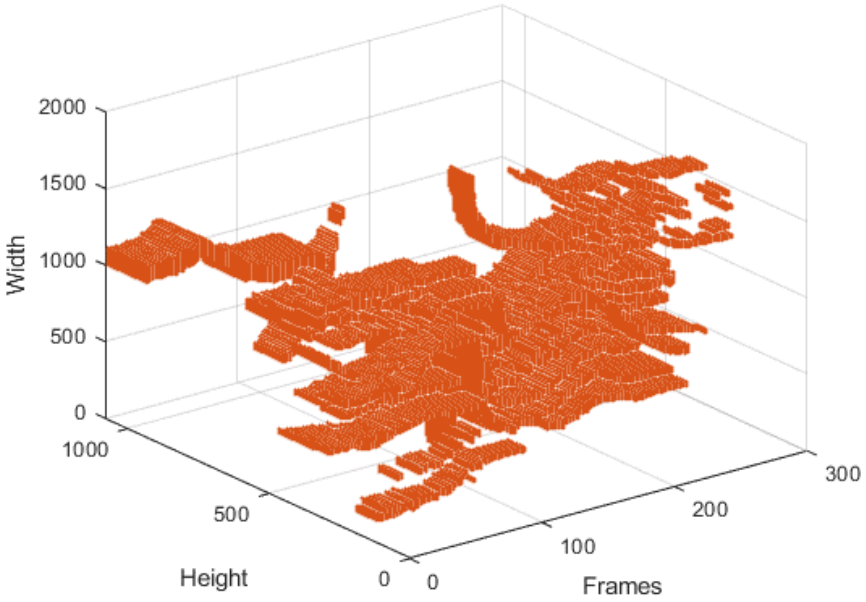

Figure 6. A set of heatmaps corresponding to each consecutive frame of shot 1 of the replay video.

Algorithm 1: Searching for locations to display the graphic

Input: frame_h ≥ gfx_h > 0, frame_w ≥ gfx_w > 0, (*sizes of the TV frame and the graphic)*
min_time > 0, *(minimal duration to display the graphic)*
B ∈ ($\mathbf{B}_1$,..., $\mathbf{B}_n$), n > 0, (*buffer of detected objects for the n frames ahead)*
$\mathbf{B}_t$ ∈ {(x_i, y_i, h_i, w_i): 0 < x_i ≤ frame_h, 0 < y_i ≤ frame_w, h_i > 0, w_i > 0, i ≥ 0} , 1 ≤ t ≤ n
Ouput: L ∈ {(x, y, s, d): 1 ≤ x ≤ frame_h, 1 ≤ y ≤ frame_w, 1 ≤ s ≤ n, 1 ≤ d ≤ n} (*list of valid locations)*

Set A ← $\mathbb{0}^{(\text{frame\_h} - \text{gfx\_h} + 1) \times (\text{frame\_w} - \text{gfx\_w} + 1)}$ (*intialisation of availability map of locations with zeros)*
Set L ← {} *(initialisation of list of locations with an empty set)*
Foreach t ∈ (1, ..., n)
Set V ← $\mathbb{1}^{(\text{frame\_h} - \text{gfx\_h} + 1) \times (\text{frame\_w} - \text{gfx\_w} + 1)}$ (*intialisation of map of vacant locations (the current frame) with ones)*
Foreach (x, y, h, w) ∈ $\mathbf{B}_t$ *(removing all locations which overlap with any detected object)*
Foreach i ∈ (x – gfx_h + 1, ..., x + h – 1) **and** j ∈ (y – gfx_w + 1, ..., y + w – 1)
Set $\mathbf{V}_{ij}$ ← 0
Foreach i ∈ (1, ..., frame_h – gfx_h + 1) **and** j ∈ (1, ..., frame_w – gfx_w + 1)
If $\mathbf{V}_{ij}$ = 0 (*processing locations causing occlusions)*
If A_{ij} ≥ min_time
Set L ← L ∪ (i, j, t - A_{ij}, A_{ij})
Set A_{ij} ← 0
Else *(incrementing availability of vacant locations)*
Set A_{ij} ← A_{ij} + 1
Foreach i ∈ (1, ..., frame_h – gfx_h + 1) **and** j ∈ (1, ..., frame_w – gfx_w + 1)
If A_{ij} ≥ min_time (*processing locations which are still vacant)*
Set L ← L ∪ {(i, j, n - A_{ij} + 1, A_{ij})}
Return L

---

4. Results

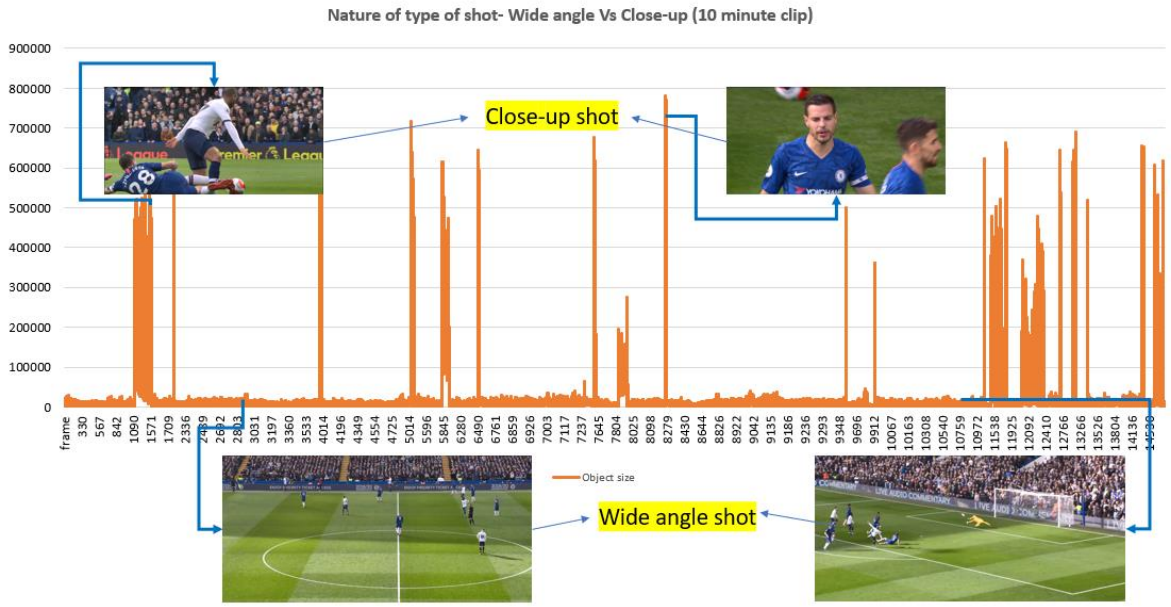

Figure 7: Analysis of 10-minute test clip to benchmark object detection

Figure-7 illustrates the results of detected objects in a 10-minute duration test clip. The x-axis represents the frame ID/frame number and the y-axis represents the size (pixel area) of the detected objects. The increase in the number of pixels indicates the large object detected due to a close angle shot. Similarly, the smaller the number of pixels on detected objects, the wider the field of view of the input stream and the smaller the size of detected objects. It must be noted that this graph doesn't directly co-relate to the number of objects detected, although multiple objects were detected throughout the input test clip video.

A test video clip of approximately 10 minutes was used to benchmark the object detection model in the absence of reliable ground-truth data. To the best of our knowledge, there are no publicly available labelled data sets to train object detection models using content of this type, and significant resources would be required to manually create them. Figure 7 indicates the individual object sizes of detected objects for 15,000 frames in a 25 frames per second (fps) video test clip. The peaks and troughs in the above graph clearly evidence the difference between wide-angle and close-up shots respectively. This indicates that the model's performance is consistent, and it is able to detect and identify objects of different sizes.

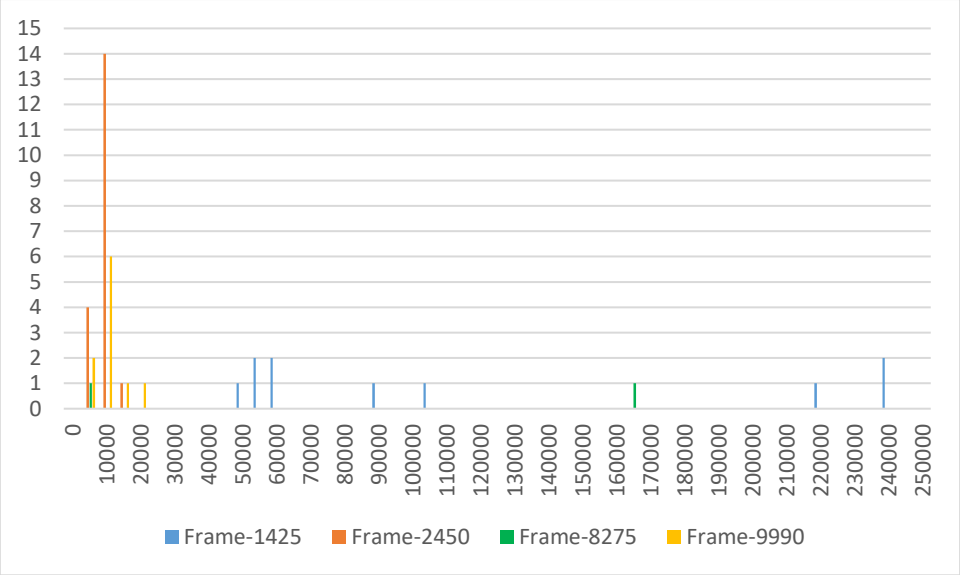

Figure 8: Histogram of object sizes across selected frames

Figure-8- represents the variation of sizes of objects detected in randomly selected 4 different frames of the 10-minute video test clip. The x-axis denotes the size of the objects corresponding to the pixel area occupied by the objects and the y-axis represents the number of objects detected on a particular frame. The results indicate that objects of different sizes and numbers across the test video stream were detected by the object detection model.It can also be seen that wide-angle shots (frames 1425 and 8275) exhibit distinctly different histograms to close-up shots, as expected.

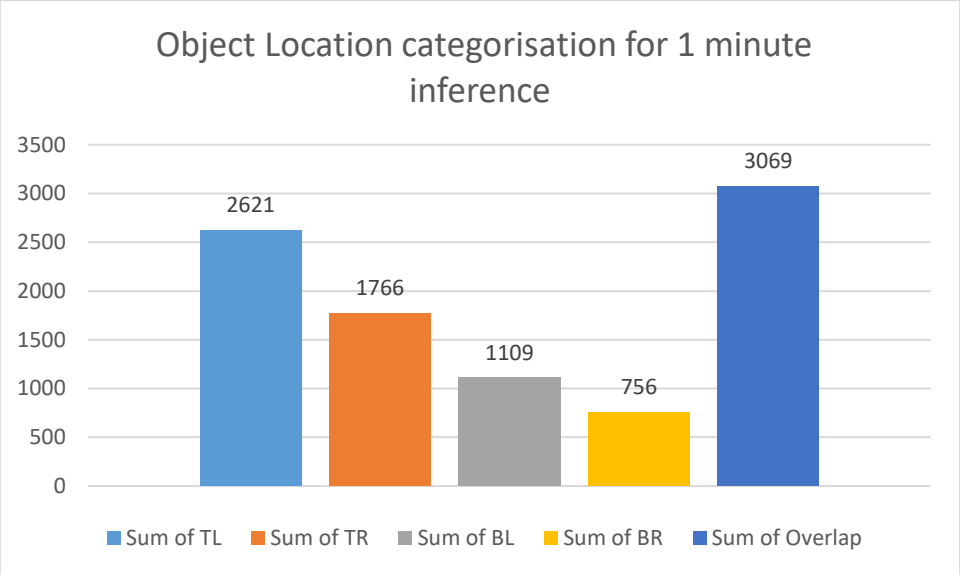

Figure 9: Example of object location distribution during a 1-minute wide-shot sequence

A list of objects and their respective locations within the frame was extracted for a 1-minute wide-shot sequence within our 10-minute test clip. This was used to identify the approximate distribution of detected objects over 1,500 frames (at 25fps). It can be noted that there are a total of 9,321 objects (football players) detected and the majority of them are on TL (Top-Left) quadrant of the frame, and the least-detected area for players is the BR (Bottom Right) quadrant of the frame, with one third of all objects being located in more than one quadrant of the frame.

The one-minute subset of the test video was chosen to derive and understand the distribution of the detected objects of interest in an individual frame. The camera angle, the field of view, and the area of focus are consistently changing; therefore, it is important to analyse the generic distribution of detected objects to facilitate the design

improvements of the graphic placement algorithm. The one-minute subset is a random selection, and multiple such subsets are selected to analyse the distribution of the detected objects. The above-mentioned graph is an example of one of the subsets that were selected to elucidate the analysis. The 10-minute test clip is sampled from the 2019 Community Shield match between Manchester City and Liverpool games, this could be found online [30].

Without the ability to dynamically determine placement of the media object, a broadcaster would need to specify in advance a fixed default location (such as the top-left corner of the screen) for any personalised graphic overlays chosen by the viewer. This simple analysis illustrates the important point that a fixed location for a supplementary graphic over football match content has a high probability of occluding key action.

The dynamic placement process was tested on a video sequence which contains a replay of the first goal scored during the 2019 Community Shield match between Manchester City and Liverpool. The video comprises four main shots from different angles, plus a fifth crowd reaction shot. The boundaries between the shots were automatically determined based on analysis of rapid changes of the average sizes of the detected objects within every frame (Figure 10). Analysis of the Kullback–Leibler divergence (i.e. the relative entropy) between normalised histograms of every two consecutive frames (Figure 11) not only confirms these shot boundaries (as its rapid rise usually indicates a change of shot type), but also suggests an approach for classification of shots (for targeting relevant auxiliary content) based on histogram analysis, which is one of the subjects of ongoing research.

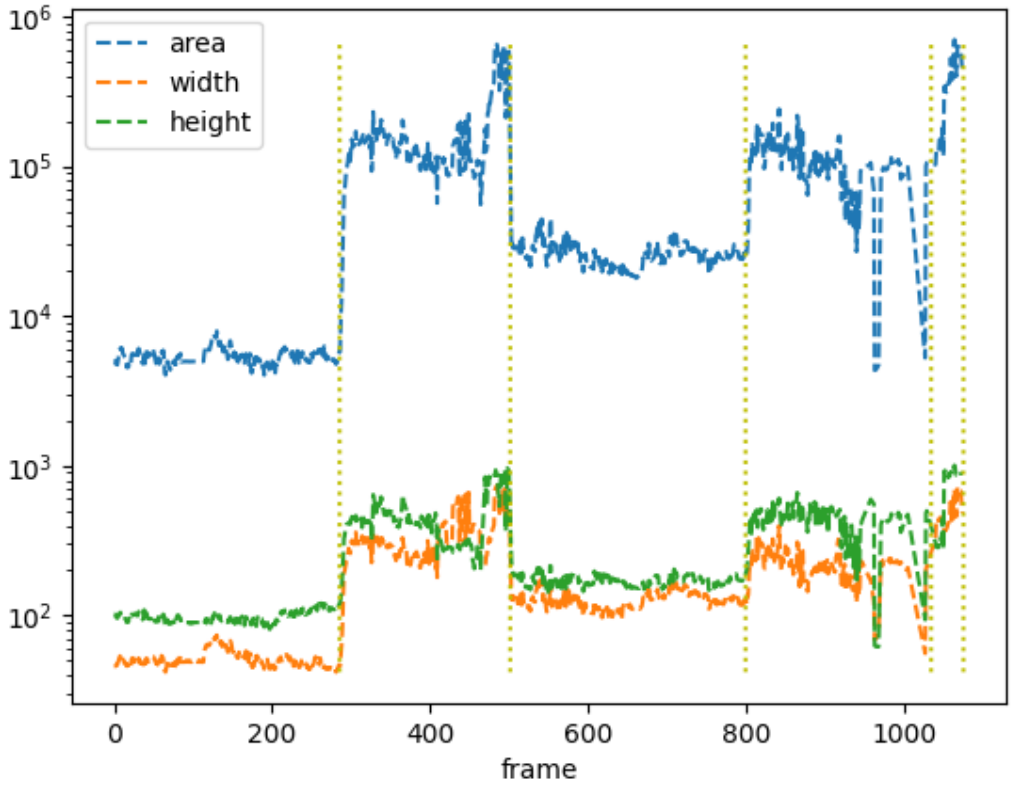

Figure 10. Average height, width, and area of detected objects in each frame (vertical dotted bars show boundaries between clips shot from different camera angles) plotted on a logarithmic scale.

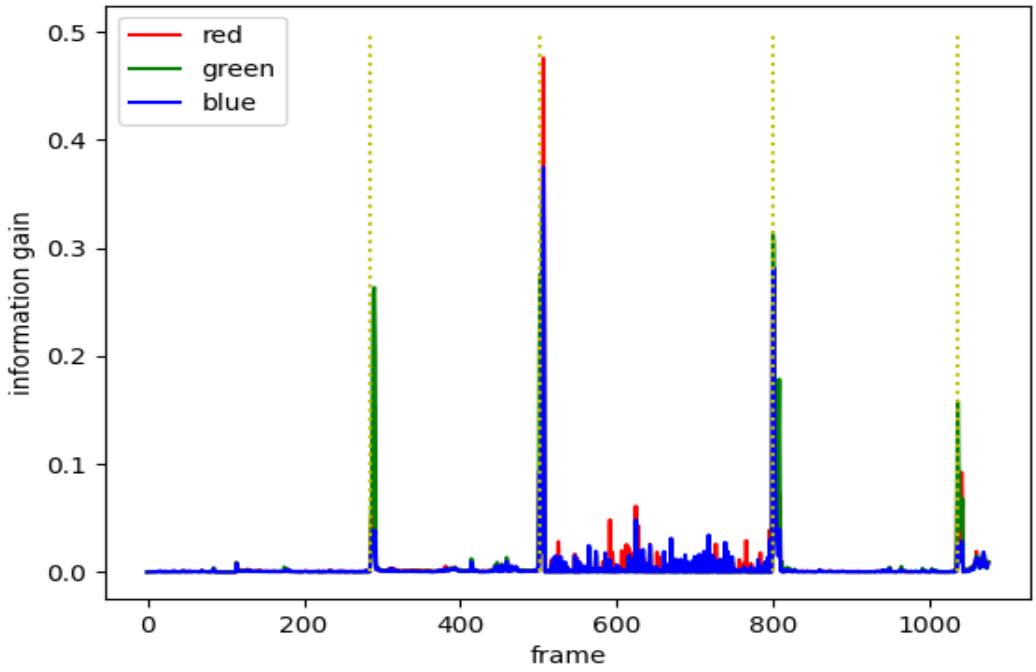

Figure 11. Kullback–Leibler divergence between normalised histograms from three colour channels of consecutive frames (vertical dotted bars show boundaries between clips shot from different camera angles).

The co-ordinates of the bounding boxes of objects of interest are shown as a sequence of heatmaps (one for each consecutive frame) for every shot (Figure 6). The lengths of those shots as well as the output of the graphic placement algorithm (the longest durations of continuous display of the graphic in fixed positions without causing any occlusions of key action in each shot) are listed in Table 1. Unlike wide-angle shots I and III, shots II and IV show several close, narrow-angle views and shot V records reactions of the fans.

		Duration and frame range for graphic placement		
Shot	**Length**	**Fixed location**	**Left stack**	**Safe frame**
I	285 (1 - 285)	171 (115 – 285)	171 (115 – 285)	**285** (1 - 285)
II	216 (286 - 501)	115 (387 – 501)	153 (344 - 496)	153 (344 - 496)
III	298 (502 - 799)	**298** (502 - 799)	**298** (502 - 799)	**298** (502 - 799)
IV	235 (800 - 1034)	171 (864 - 1034)	202 (833 - 1034)	202 (833 - 1034)
V	41 (1035- 1075)	11 (1035 - 1045)	25 (1035 - 1059)	**41** (1035 - 1075)
I-V	1075 (1 - 1075)	470 (387 - 856)	470 (387 - 856)	470 (387 - 856)

Table 1: The first and last frames as well as duration of every shot within the replay video sequence together with the proposed start time and duration to display the graphic without causing any occlusions of the game in a fixed (default) location, the left stack and the whole frame excluding the safe area (i.e. dark red area in **Figure 4**)

An example of dynamic graphic placement during shot I of the replay video sequence is visualised in Figure 12. As one can notice from Table 1, the algorithm computes the best locations of the graphic within each of the shots and suggests the best rigid location for the whole video clip. First we run the graphic placement algorithm to check availability of the default location - the top-left corner of the left stack (as shown in the top-right image of Figure 12): in shot I it does not overlap with any detected objects of interest for its last 171 frames (column 'Fixed location' and the top-left image). Searching throughout the whole frame without safe zones marked dark red in Figure 4 (column 'Safe frame'), we discover locations available for the whole duration of shot I, 285

frames (the bottom images). Therefore, the graphic placement algorithm allows us to display an overlay graphic without any occlusion of any key action for longer period of time and in a desired area of the frame.

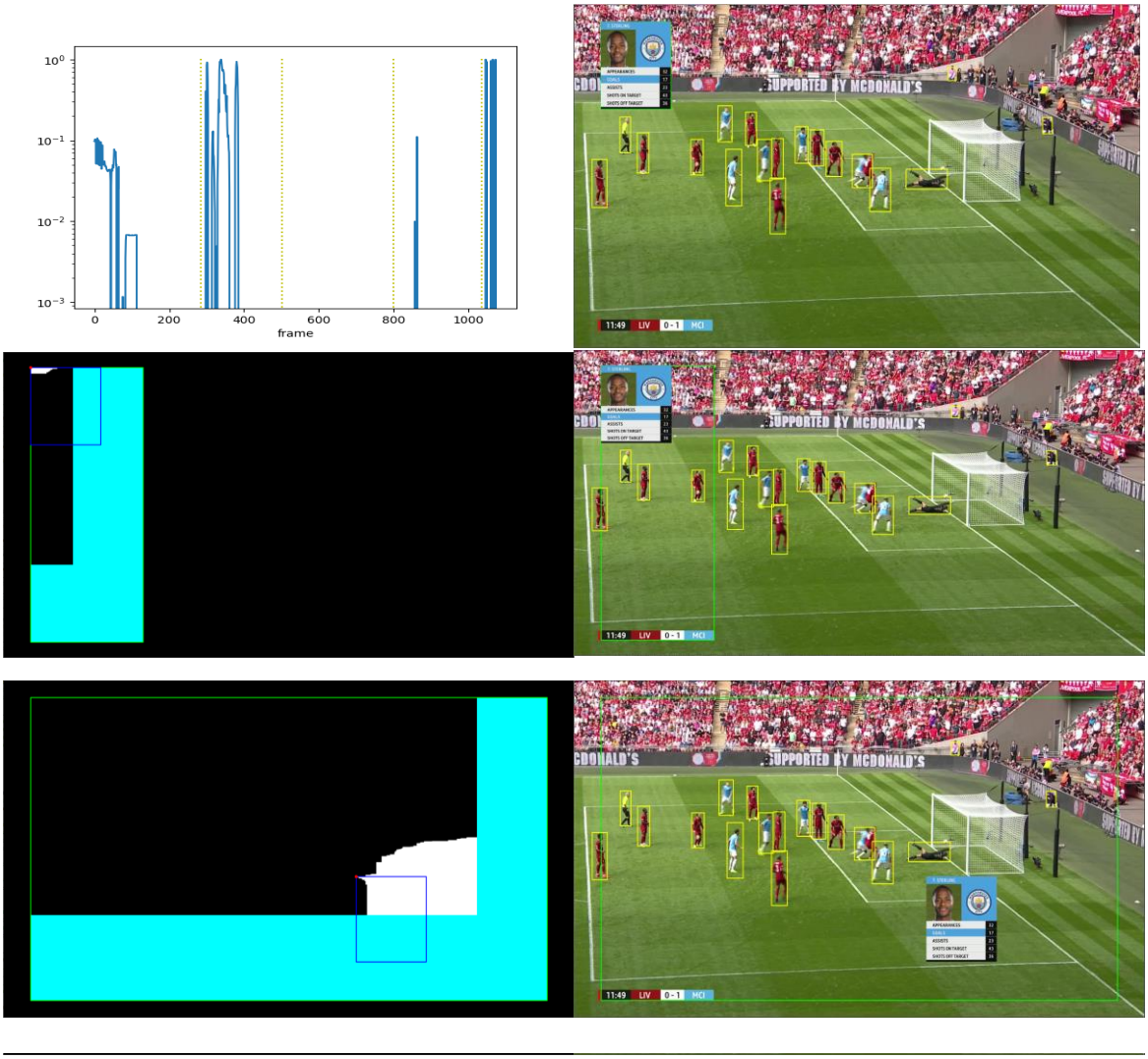

Figure 12. The demo of graphic placement on shot I in three pre-defined templates: the default location (top right), the left stack (middle right) and the whole frame excluding safe zones (bottom right). The information graphic is placed by the decision-making algorithm in such a way that it fits in the display area (the green rectangle) aiming at its top left corner and avoids occluding any objects of interest (in the yellow boxes). The output of the algorithm is shown for the left stack (middle left) and the whole frame without safe zones (bottom left): locations of the graphic are defined by positions of its top left corners (the interior of the green rectangles without cyan areas), those continuously available for the longest time are shown in white, the selected location is marked in red, the respective position of the graphic is denoted by the blue rectangle. The relative overlap of the graphic in the default location with the detected objects of interest is plotted per frame on a logarithmic scale (top left).

5. Live Proof of Concept

Having demonstrated that the dynamic graphic placement technique described above provided good results when applied to a recorded replay video sequence, a live Proof of Concept was developed in order to:

- Facilitate experimentation and refinement of the dynamic placement algorithm on a wider range of content.

- Determine whether dynamic graphic placement could be carried out on live content, taking advantage of the inherent delay in adaptive video streaming to 'look ahead' in the live stream by 30-60 seconds and make graphic placement decisions accordingly.

Figure 13 shows the high-level end-to-end architecture of Object-Based Broadcasting for live sports production, demonstrating how existing broadcast functions are combined with new components to enable object-based personalisation. The left-hand column indicates that both recorded media assets (graphics, title sequences, transitions etc) and app software are prepared in advance of live production. It should be noted that existing TV player apps will need to be updated to provide object-based personalisation, potentially across multiple screens. The next column summarises some of the key processes involved in live production, including the capture, mixing and encoding of the live match programme, the graphics overlaid on it and the data associated with the progress of the match and its players (as well historical data and data related to concurrent matches). Unlike a conventional broadcast production process, these video, audio, graphics and data components remain as separate objects delivered independently to the viewer (customer) via the Content Delivery Network, and indeed the live match programme is provided as a 'clean feed', with no graphics overlaid on it.

The OBB platform includes of a set of app management web services with which the viewer's TV apps communicate while the personalised experience is delivered. The Orchestration component is a back-end web service which distributes instructions which control the timing and properties of media objects to be rendered. The Orchestration component can receive an instruction from a live graphics production tool controlled by a human operator, which will include a real-time (NTP-synchronised) timestamp representing the time the requested graphic overlay object should be displayed. Real-time timestamps are also inserted during encoding of the live match 'clean feed' for adaptive bitrate streaming, so when a TV app decodes a buffered segment of this stream, it can read the corresponding timestamp for each frame showing when it was encoded live. This means that the TV app can synchronise an Orchestration instruction to display the graphic overlay object with the delayed live video stream.

Figure 13 also indicates how Orchestration instructions could be supplied by an 'AI Engine'. This AI Engine, another back-end web service, can be triggered directly by Orchestration, or following an interaction by a viewer's app.

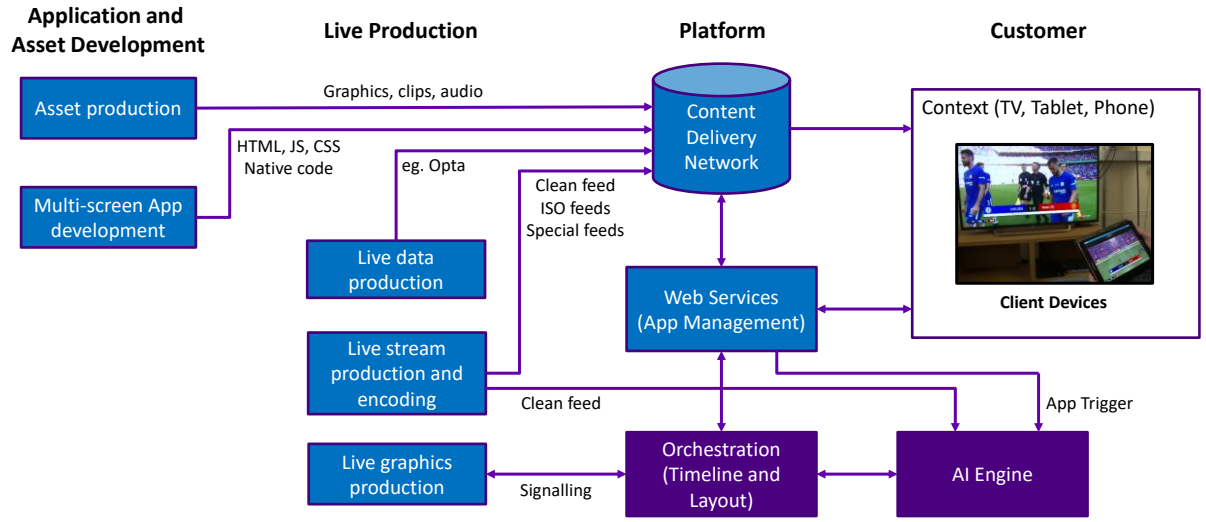

Figure 13. Architecture of an Object-Based Broadcasting system incorporating an AI Engine.

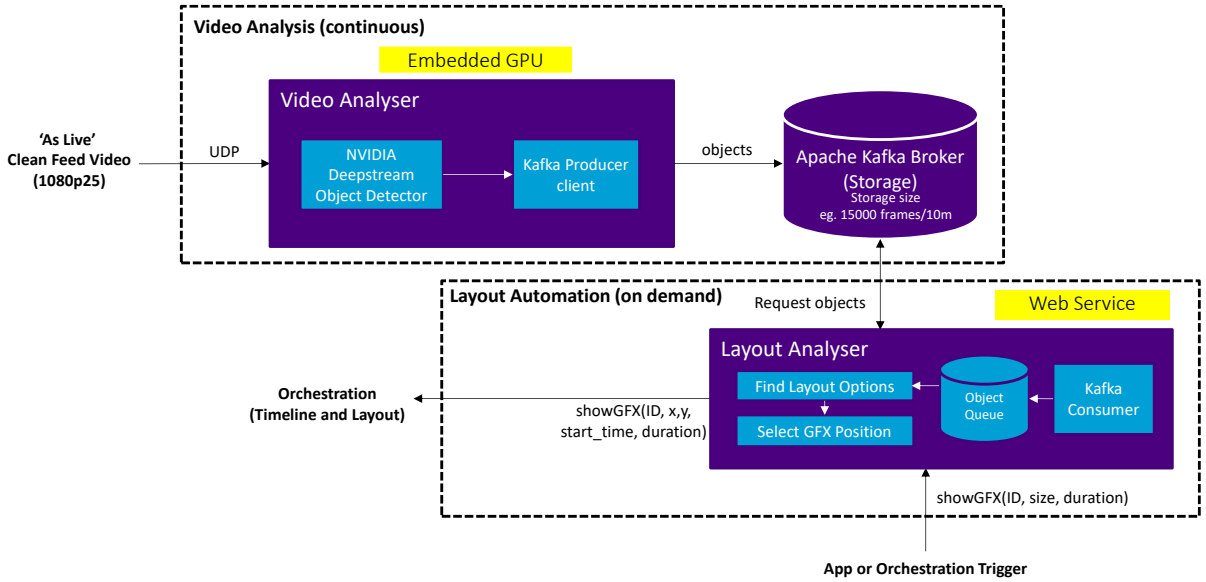

Figure 14. Internal architecture of the **Proof-of-Concept** AI Engine.

Figure 14 shows the internal architecture of our Proof-of-Concept AI Engine, which comprises three components:

The Video Analyser is a GPU-accelerated video processor which detects the bounding boxes of key objects in every frame of a live video stream, using the NVIDIA DeepStream framework as described above [28]. For the purpose of our experiments, it is provided with a recorded 'clean feed' video stream, played back as if the match were live, and sent over UDP for minimal delay from real-time. The same clean feed is also encoded, packaged and delivered to client devices using adaptive streaming, in the same way as described above for a real broadcast event.

An Apache Kafka [31] Broker is used to temporarily store a queue of object bounding boxes which are generated by the Video Analyser for every frame of the incoming video stream. A real-time timestamp is also attached to each frame. Apache Kafka provides a reliable solution for managing this data feed with low latency, and its ability to store the queue makes it possible to use this architecture for iterative testing of the dynamic placement algorithm by running it many times over the same object queue.

The Layout Automation component is a web service which is called on demand whenever a personalised overlay graphic is required. This request could be made directly from a viewer's client application or, more likely for our use case, will be triggered by the Orchestration component. Layout Automation incorporates a Kafka Consumer which, when triggered, requests a list of object bounding boxes for every frame in the queue corresponding to the estimated length of the delay between the client video playback and real time (for example, 30s). It then performs the dynamic placement algorithm as described in the Method section above, taking its starting point in the queue from the timestamp provided in the web request, which must represent playback time at the client, hence looking forward in time from the client's perspective. In order to achieve this, the Orchestration component maintains a regularly-update record of current playback time at each client TV app.

If the viewer were to be given the opportunity to interactively request a personalised graphic overlay, an optimisation which could be performed at this stage would be to restrict placement to locations which are vacant from the starting point chosen in the queue, so that a graphic could be placed instantaneously in response to their request.

Once the location and start time of the dynamic placement have been determined by Layout Automation, the component sends an instruction to the Orchestration component in the OBB platform to display the graphic object. Iterative development and testing have been carried out with this Proof-of-Concept to optimise the efficiency and accuracy of the placement algorithm and to ensure that the placement location can be calculated with a latency of less than 1s.

For the purposes of our Proof of Concept, our OBB platform and AI Engine components were hosted within a lab environment, most running as containerised services on a private cloud platform. In an operational setting, we would expect all of these components, including GPU-accelerated video processing, to be hosted on a public cloud platform which would provide cost-effective on-demand access to a dynamic graphic placement capability.

6. Conclusions and Further Work

In this paper we have described the development of a Proof-of-Concept automated system to drive one aspect of a personalised, object-based TV experience.

We have developed an AI-based approach to the automated insertion of personalised graphics. The detection of players and the ball in a football replay clip is performed by a deep neural network algorithm whose output is used to define the region of interest within each frame of the clip. Data analysis of the remaining space in a sequence of frames suggests suitable locations for the graphic to avoid occlusions of key action of the game, and furthermore, our system places the graphic at the best position based on templates pre-defined in accordance with the broadcaster's style guide (e.g. top-left, top-right, bottom-right and bottom-left). Our solution also automatically detects boundaries between parts of the clip shot from different camera angles so that graphics are inserted on wide shots only.

We have found that the ability to dynamically determine object placement is vital for a broadcaster to avoid the need to specify in advance a fixed location (such as the top left corner of the screen) for any personalised graphic overlays chosen by the viewer. Fixing a location in advance to allow a new media object to be inserted over football match content can lead to a high probability of occluding key action.

We have found that during a replay video sequence the algorithm is able to calculate the best locations (which minimise occlusion of the game) to insert the graphic within each of the shots and propose the best location for the entire video clip.

The architecture of an AI engine for placement of graphic objects within a live video broadcast has been described. This takes advantage of the inherent delay in adaptive video streaming to 'look ahead' in the live stream and make placement decisions accordingly. For our live Proof of Concept, we have optimised the algorithm to determine a placement location with a latency of less than 1s.

Based on our experimentation with single-stage and two-stage object detection architectures, we believe that the Faster R-CNN model (with ResNet-18 or ResNet-50 backbone) and YOLOv3 (with DarkNet-53 or ResNet-18) could provide more accurate object detection of football players within a live video stream. We will be performing further experiments to validate the selection of this model. We also plan to explore how the addition

of object tracking could enable us to automatically determine additional relevant information about sports match content, such as states of play. This could provide further guidance for the temporal placement of personalised graphics.

We hope that the results that we obtained in this Proof-of-Concept will help to influence future design of Object Based Media experiences, so that if personalised media objects are to be inserted over video content, this can be achieved while minimising the occlusion of key action in the scene. We believe that our approach could applied to a range of field-based sports, but further work is required to assess its performance and suitability. Having demonstrated that dynamic placement of media objects can be achieved in a live broadcast environment, we will now seek to test different use cases with real viewers.

We are planning to carry out an evaluation in a form of subjective testing by expert viewers as one of the directions of our future work. Our test design has been informed by a previous evaluation of dynamic subtitles [32]. We have selected 10 clips of 30-60 seconds duration based around the key events within an English Premier League match and will use our Proof of Concept to identify a graphic placement immediately after each event has happened. At the beginning of the test, and to provide a basis for comparison, we will ask each expert to suggest where a broadcaster would typically place an information graphic. We will then play each clip showing our suggested graphic placement to each expert and, after each clip, ask them to complete a questionnaire measuring their user experience [33] [34] on a Likert scale [35]. We will include additional qualitative questions to explore their opinion of the placement in terms of location on screen and timing. We may potentially extend the evaluation to include measurement of cognitive load using other modalities, like eye-tracking and electroencephalography [36].

References

[1] M. Armstrong, "Object-Based Media: A Toolkit for Building Responsive Content," in *Proceedings of the 32nd International BCS Human Computer Interaction Conference (HCI)*, Belfast, UK, 2018.

[2] E. Howells and D. Jackson, "Object-based media report," Ofcom, London, 2021.

[3] Netflix, "Black Mirror: Bandersnatch," Netflix, 2018. [Online]. Available: https://www.netflix.com/gb/title/80988062.

[4] J. Walker, D. Williams, I. Kegel, A. Gower, J. Jansen, M. Lomas and S. Fjellsten, "2-IMMERSE: A Platform for Production, Delivery, and Orchestration of Distributed Media Applications," *SMPTE Motion Imaging Journal,* vol. 128, no. 7, 2019.

[5] J. Cox, M. Brooks, I. Forrester and M. Armstrong, "Moving Object-Based Media Production from One-Off Examples to Scalable Workflows," *SMPTE Motion Imaging Journal,* pp. 127(4): 32-37, 2018.

[6] T. Röggla, J. Li, J. Jansen, S. Fjellsten, I. Kegel, L. Pilgrim, M. Trimby, D. Williams and P. Cesar, "From the Lab to the OB Truck: Object-Based Broadcasting at the FA Cup in Wembley Stadium," *Proceedings of CHI Conference on Human Factors in Computing Systems Extended Abstracts (CHI'19 Extended Abstracts),* 2019.

[7] M. Ibrahim, T. Lohmar, A. El-Essaili and A. d'Allonnes, "TV graphics personalization using in-band events," in *Proceedings of the In-Programme Personalization for Broadcast (IPP4B) Workshop, ACM TVX2017*, New York, 2017.

[8] ITU-R, "Report BT.2447-0: Artificial intelligence systems for programme production and exchange," BT Series, 2019.

[9] P. Dollar, C. Wojek, B. Schiele and P. Perona, "Pedestrian Detection: A Benchmark," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, Miami, USA, 2009.

[10] P. Dollar, C. Wojek, B. Schiele and P. Perona, "Pedestrian Detection: An Evaluation of the State of the Art," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 34, no. 4, pp. 743-761, 2012.

[11] C. Cortes and V. Vapnik, "Support-Vector Networks," *Machine Learning,* vol. 20, no. 3, pp. 273-297, 1995.

[12] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, San Diego, USA, 2005.

[13] P. Dollar, R. Appel, S. Belongie and P. Perona, "Fast Feature Pyramids for Object Detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 36, no. 8, pp. 1532-1545, 2014.

[14] M. Carranza-García, J. Torres-Mateo, P. Lara-Benítez and J. García-Gutiérrez, "On the Performance of One-Stage and Two-Stage Object Detectors in Autonomous Vehicles Using Camera Data," *Remote Sensing -MDPI,* vol. 13, no. 89, 2021.

[15] C. Andreu, "Semantic Image Cropping," Queen Mary University of London, 2018.

[16] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu and A. C. Berg, "SSD: Single Shot MultiBox Detector," in *Computer Vision – ECCV 2016: 14th European Conference*, 2016.

[17] J. Redmon and A. Farhadi, "YOLO9000: Better, Faster, Stronger," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, HI, USA, 2017.

[18] L. Jiao, F. Zhang, F. Liu, S. Yang, L. Li, Z. Feng and R. Qu, " A Survey of Deep Learning-based Object Detection," *NEURAL NETWORKS AND LEARNING SYSTEMS-IEEE,* 2019.

[19] T. Lu, H. Tianran and W. e. a. Liyao, "Comparison of YOLO v3, Faster R-CNN, and SSD for Real-Time Pill Identification," *BMC Medical Informatics and Decision Making,* no. Currently-Under peer review, 2021.

[20] Morera and Ángel, "SSD vs. YOLO for Detection of Outdoor Urban Advertising Panels under Multiple Variabilities," *Sensors (Basel, Switzerland),* vol. 20, 2020.

[21] S. Alkentar, B. Alsahwa, A. Assalem and D. Karakolla, "Practical comparation of the accuracy and speed of YOLO, SSD and Faster RCNN for drone detection," *University of Baghdad - Journal of Engineering,* vol. 27, no. 8, 2021.

[22] NVIDIA, "NVIDIA DeepStream SDK Developer Guide," 08 March 2020. [Online]. Available: https://docs.nvidia.com/metropolis/deepstream/5.0/dev-guide/index.html#page/DeepStream_Development_Guide/deepstream_performance.html#wwpID0E0YD0HA.

[23] Y. Zhong, J. Wang, J. Peng and L. Zhang, "Anchor Box Optimization for Object Detection," in *IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2020.

[24] Stats Perform, "Stats Perform - World Leaders in Sport Data," 2022. [Online]. Available: https://www.statsperform.com/opta/.

[25] Y. Sani, A. Mauthe and C. Edwards, "Adaptive Bitrate Selection: A Survey," *IEEE Communications Surveys and Tutorials,* vol. 19, no. 4, 2017.

[26] M. Jackson, "ISP Review - Broadcast Lag in Live Online TV Sport Streaming Frustrates Fans," ISP Review, 10 June 2021. [Online]. Available: https://www.ispreview.co.uk/index.php/2021/06/broadcast-lag-in-live-online-tv-sport-streaming-frustrates-fans.html.

[27] Lin, Tsung-Yi, Goyal, Priya, R. H. Girshick, Kaiming, Dollar and Piotr., "Focal Loss for Dense Object Detection," in *International Conference on Computer Vision (ICCV)-IEEE*, 2017.

[28] NVIDIA, "DeepStream SDK," 5 11 2021. [Online]. Available: https://developer.nvidia.com/deepstream-sdk.

[29] NVIDIA, "DetectNet_v2 TAO Toolkit 3.0 documentation," 25 August 2021. [Online]. Available: https://docs.nvidia.com/tao/tao-toolkit/text/object_detection/detectnet_v2.html. [Accessed 12 November 2021].

[30] BT, "Man City vs Liverpool (1-1, 5-4 on pens) | 2019 Community Shield highlights," BT Sport, 4 August 2019. [Online]. Available: https://www.youtube.com/watch?v=k9_tz9bi3rs.

[31] "Kafka 3.0 Documentation," [Online]. Available: https://kafka.apache.org/documentation/.

[32] M. Armstrong, A. Brown, M. Crabb, C. J. Hughes, R. Jones and J. Sandford, "Understanding the Diverse Needs of Subtitle Users in a Rapidly Evolving Media Landscape," *SMPTE Motion Imaging Journal,* vol. 125, no. 9, pp. 33-41, 2016.

[33] E. L.-C. Law and P. van Schaik, "Modelling user experience - An agenda for research and practice," *Interacting with Computers,* vol. 22, no. 5, pp. 313-322, 2010.

[34] E. L.-C. Law, V. Roto, M. Hassenzahl, A. P. Vermeeren and J. Kort, "Understanding, Scoping and Defining User Experience: A Survey Approach," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'09)*, Boston, USA, 2009.

[35] R. Likert, "A technique for the measurement of attitudes," *Archives of Psychology,* vol. 22, no. 140, pp. 5-55, 1932.

[36] J.-L. Kruger, S. Doherty, W. Fox and P. de Lissa, "Multimodal measurement of cognitive load during subtitle processing: Same-language subtitles for foreign-language viewers," in *Innovation and Expansion in Translation Process Research*, Amsterdam, Netherlands, John Benjamins Publishing Company, 2018, pp. 267-294.