Real-Time Occlusion-Aware Mixed Reality through Motion Capture Reconstruction and Instance Segmentation

Utrecht University (with Beyond Sports)
Department of Information and Computing Sciences
Master of Science: Game and Media Technology

Martin Rønning (2331136)
First supervisor: Prof. dr. Remco Veltkamp
Daily supervisor: dr. Daniel Karavolos
Second supervisor: dr. ir. Ronald Poppe

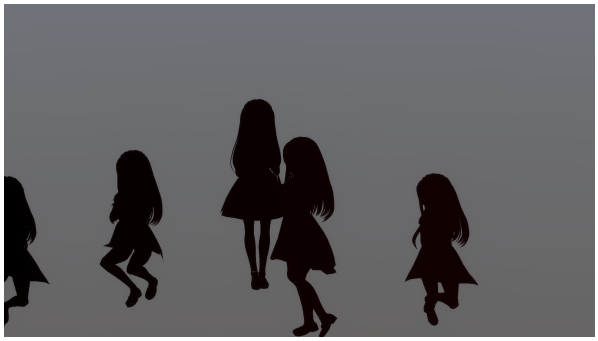


Figure 1: Motion capture reconstruction in 2D perspective, functioning as our main approach's source of depth, and where the 2D outlines of these meshes correspond to what we refer to as synthetic masks

Figure 2: Broadcast footage with intended real-time occlusion of an inserted virtual object composited using live segmentation masks and depths from the motion capture reconstruction

8

Abstract

Realistic occlusion in mixed reality (MR) broadcast applications requires per-pixel depth ordering between real performers and inserted virtual objects, information that is unavailable from standard broadcast video alone. This work proposes a real-time MR pipeline that addresses this problem by combining motion capture reconstruction (MCR) as an explicit depth source with frame-wise instance segmentation for per-pixel masking, implemented within a Unity environment. A spatially-sorted projection mapping algorithm associates AI-generated segmentation masks with their corresponding reconstructed performers, enabling depth-ordered composition via a custom shader. A synthetic fallback mechanism based on reconstructed mesh outlines handles frames where live segmentation fails. The pipeline is evaluated across five experimental runs and two sequences of varying scene complexity, using a three-phase benchmarking framework covering intersection detection, depth classification, and visual fidelity. MCR-based depth achieved near-perfect depth classification accuracy (>99%) compared to 68–80% for monocular depth estimation, confirming its feasibility for broadcast MR occlusion. The primary real-time configuration (*Live+MCR*) achieved a median pipeline latency of ∼35 ms, narrowly missing the $\tau_{30} \approx 33.3$ ms threshold for 30 FPS, with optimization pathways identified. Also, a comparative evaluation of RF-DETR and YOLOv11 for real-time instance segmentation demonstrates that transformer-based architectures are viable for live MR inference, with RF-DETR-M achieving 8.94 ms latency at competitive mask fidelity.

Abbrevations

- **MR** Mixed Reality
- **VR** Virtual Reality
- **AR** Augmented Reality
- **AI** Artificial Intelligence
- **MCR** Motion Capture Reconstruction
- **VIS** Video Instance Segmentation
- **FPS** Frames Per Second
- **YOLO** You Only Look Once
- **RF-DETR** Roboflow Detection Transformer
- **NMS** Non-Maximum Suppression
- **CNN** Convolutional Neural Network
- **ViT** Vision Transformer
- **SAM** Segment Anything Model
- **VDA** Video Depth Anything
- **VOB** (Inserted) Virtual Object
- **BVH** Biovision Hierarchy
- **IDH** Intermediate Data Format
- **ROI** Region-of-Interest
- **CLAHE** Contrast Limited Adaptive Histogram Equalization
- **GUI** Graphical User Interface
- **CSV** Comma-Separated Values
- **I/O** Input/output
- **GPU** Graphical Processing Unit
- **RAM** Random Access Memory
- **ONNX** Open Neural Network Exchange

Notations

- **R** Real input broadcast video
- **V** Virtual reconstruction scene
- **M** Predicted segmentation mask
- **P** Reconstructed performer
- τ_{30} Maximum per-frame processing latency allowed for 30 FPS ($\approx$ 33.3 ms)
- τ_{60} Maximum per-frame processing latency allowed for 60 FPS ($\approx$ 16.7 ms)
- z Depth w.r.t the camera in 3D space
- I Gamma correction
- γ Gamma factor
- d Euclidean distance
- μ Binary thresholding factor
- $\mathcal{P}$ 3D point (World Space)
- $\mathcal{R}$ Rotation matrix
- t Translation vector
- u X-coordinate (screen space)
- v Y-coordinate (screen space)
- f Focal length
- c Prinicipal point
- $mIoU$ Mean-Intersection-over-Union
- mAP Mean-Average-Precision
- $\mathbf{D}$ Depth map values vector
- $\Omega_{\mathbf{P}}$ Depth map of the region of some **P**
- $\omega_{\mathbf{P}}$ Scalar depth value of the region of some **P**
- s Depth map scale
- o Depth map offset

1 Introduction

Mixed reality (MR) is a hybrid technology where the real world merges with virtual stimuli [1]. It has numerous applications, such as to assist surgeons, educational platforms as well as in the entertainment industry. In order to realistically blend realities in the case of MR media, some information on the depth of the scene is crucial [2]. Occlusion functions as a perceptual depth cue in such MR scenes, where objects occluded by others in the right order helps to understand how close or far away these objects lie within the scene. However, accomplishing accurate occlusion of objects originally belonging to one reality - with respect to the other reality - is challenging without precise depth information for both realities. In many cases, such as when the only input is video footage of the real world, there is no present depth information at all.

One promising approach to recovering this missing depth information is to leverage the output of a motion capture (MC) system. While MC systems themselves rely on initial depth sensing (e.g., via multiple cameras) to generate 3D skeletal data, this data can subsequently serve as a precise source of depth for MR objects interacting with the real objects in the form of a *reconstruction*. MC in general has important applications in biomechanics studies, such as for studying muscoskeletal diseases, but also in the entertainment industry, for example through creating virtual avatars of sports players. Marker-based MC refers to the technique of capturing movement data through the use of physical markers on the moving object and cameras to record them [3]. On the other hand, markerless MC implies that these physical markers are not necessary, which is typically preferred due to ease-of-use. After capturing the MC data, a 3D skeletal reconstruction of the moving objects can be obtained. This reconstruction effectively encodes the depth of each joint relative to the camera and to other performers, providing the spatial information required to handle and resolve occlusion in a MR setting.

A key requirement for implementing realistic occlusion in MR is the ability to precisely determine, at the pixel level, which parts of a scene belong to each performer. This process, referred to as composition, involves classifying individual pixels based on their depth ordering to correctly overlay virtual objects *behind* real performers and *in front* of others. To achieve this, we need a method that not only identifies all performers in a video frame but also distinguishes between them as unique entities, ensuring that each player can be associated with the correct depth information from the MCR.

One such method is instance segmentation, which involves segmenting a pixel-precise mask around each object instance in an image while simultaneously assigning it a unique identifier. This differs from semantic segmentation, which merely labels all pixels belonging to a class (e.g. "all players") without differentiating between individual instances of that class. The ability to separate a specific player from others in the same frame is essential for our occlusion task, as multiple performers may interact, overlap, or occlude one another, requiring distinct depth ordering per individual. Ideally, this problem would be addressed by video instance segmentation (VIS) [4], which extends instance segmentation across time by explicitly tracking object identities throughout a sequence. VIS would provide both spatial mask precision and temporal identity persistence which is exactly what our occlusion task demands. However, for real-time applications, the computational overhead of VIS methods often proves prohibitive, and dedicated tracking components can introduce latency that significantly degrades performance.

This project therefore adopts a hybrid approach: we employ per-frame instance segmentation for spatial accuracy, while leveraging the MC data itself to maintain temporal consistency. By aligning the 3D skeletal reconstruction with the 2D video feed, player identities can be derived through time without requiring explicit visual tracking.

1.1 Background

This research project is being conducted at Beyond Sports, a sports technology company which itself is part of Sony within their sports department branch. They specialize in sports visualization technology, and thus graphics and enhancements on top of existing broadcast footage, often live. They have coined the term *alternative broadcast* or *altcast* as the concept of turning real broadcast footage into virtual broadcasts, and they offer this in both traditional screen-based representation as well as virtual reality (VR) representation. For example, for a live football match they would have access to the necessary data which allows them to manipulate the footage and replace the players with characters from cartoons. As the football players move, shoot and celebrate, the altcast cartoon character would very closely mimic the same movements as the footballer. In addition to this, they are also able to replace the stadium, add effects for entertainment purposes, and allow unique camera angles. In our project the domain is shifted from live sports to live music, but the most significant aspects of the solution and further considerations remain mostly the same.

1.2 Objective

Our goal is to handle occlusions in real time for MR scenes, specifically targeting broadcast sports footage at 30, and ideally 60 frames per second (FPS). Unlike VR, where all objects are virtual and occlusion is therefore trivial, MR requires seamless integration of virtual content with real-world footage. The primary limitation in current broadcast MR applications is the lack of depth information. Without knowing whether a real player is in front of or behind a virtual element, realistic occlusion cannot be achieved. For instance, when a live football broadcast cuts to a close-up shot of a player with the audience in the background, current technology cannot reliably replace the audience with

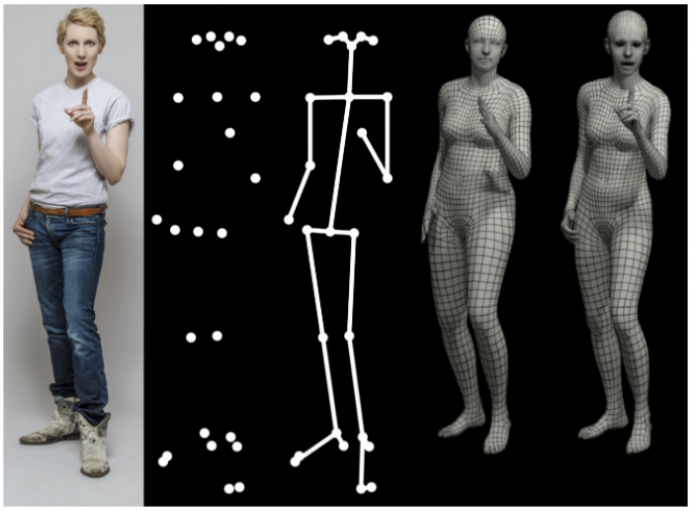


Figure 3: Illustration of the work of Pavlakos et al. and SMPL-X, which reconstructs a real object with significant detail and expressiveness of the resulting mesh. Such functionality would perfectly complement our approach, but it is computationally infeasible in the context of real-time MR. Our approach essentially uses only the first step of this figure; the pose estimation to then derive the 3D skeletal data of our performers, intended to reduce latency.

virtual characters because it cannot determine their correct depth ordering relative to the player.

To address this, we leverage the unique capture infrastructure provided by our industry partner. Their multi-camera setup simultaneously records broadcast footage and generates MC data as a structured byproduct of the acquisition process. This MC data, consisting of 3D skeletal reconstructions of the performers, serves as a critical input to our pipeline. In tandem with the frame-wise camera parameter data provided for the broadcast, it allows precise depth understanding for each player relative to the camera, which would otherwise be unavailable from the broadcast video alone.

Our pipeline combines this MC-derived depth with instance segmentation masks extracted from the broadcast video feed. By aligning the 3D skeletal data with the 2D masks, we can determine the per-pixel depth ordering required for correct occlusion. The output of our system is an occlusion-aware composite frame, where **VOBs** are rendered either in front of or behind real performers based on their true depth relationship. While our current implementation focuses on occluding inserted **VOBs**, the same principles can be extended to replaced objects, such as virtual advertising boards, with some modifications to the main pipeline required.

2 Related Work

The relevant fields of work for our approach consist of MC reconstruction and instance segmentation, as well as their application within AR and MR for handling occlusions.

2.1 Motion Capture Reconstruction

MCR techniques can be broadly categorized by their underlying sensing technology and whether they require physical markers. Marker-based MC, which uses reflective markers tracked by multiple cameras, remains the most widely adopted approach due to its high precision and reliability [5]. However, the requirement for attached markers limits its applicability in unconstrained environments such as live sports broadcasts, where performers cannot be instrumented.

Markerless MC has therefore emerged as an active research area, with various approaches balancing accuracy against practical constraints. Biplanar videoradiography (XROMM) offers high precision for capturing in vivo bone motion directly through X-ray imaging, but requires specialized equipment and controlled laboratory settings [6]. Between these extremes lie multi-view keypoint-based methods that provide three-dimensional tracking from conventional cameras, though typically with offline processing requirements [5, 7].

More recent advances in markerless 3D pose estimation have leveraged parametric body models such as SMPL [8] by Loper et al. and SMPL-X [9] by Pavlakos et al., which fit a statistical body mesh to multi-view image evidence. These approaches incorporate both pose estimation and image-derived body shape information, making them more robust to occlusions and varying body dimensions than methods relying solely on keypoint detections. However, the computational complexity of mesh fitting has largely restricted these techniques to offline processing.

Despite significant progress in markerless 3D pose estimation, a gap remains for methods that combine three-dimensional accuracy with real-time performance in uncon-

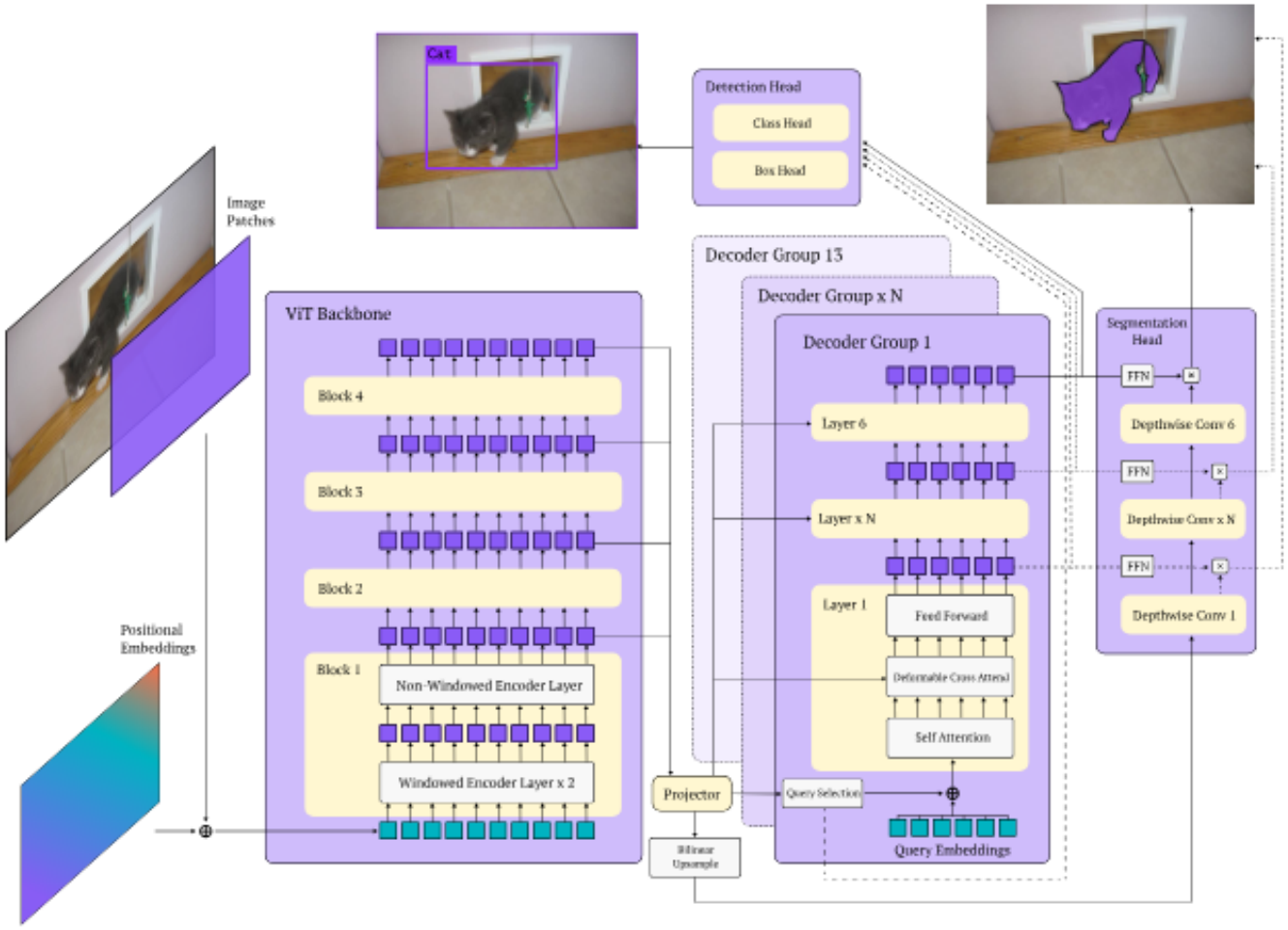


Figure 4: System diagram of methodology for RF-DETR, utilizing the pre-trained DINOv2 ViT backbone, deformable cross-attention in place of NMS and with the recently added (at the time of writing) segmentation head, which is used in our pipeline for real-time instance segmentation.

strained environments. Applications such as live broadcast MR require both the depth precision of multi-view reconstruction and the low latency necessary for real-time compositing, which is a combination not adequately addressed by existing literature.

2.2 Real-Time Instance Segmentation

The occlusion pipeline requires two distinct capabilities from segmentation models: real-time inference on video frames to drive the live compositing, and high-quality offline annotation to generate ground-truth masks for evaluation. These roles impose fundamentally different constraints, with the former demanding low latency and the latter prioritising mask precision. The models discussed here are selected accordingly.

For real-time instance segmentation, the two candidate architectures evaluated in this work are YOLOv11 and RF-DETR. Both support instance segmentation and can run at interactive frame rates, but they differ substantially in architecture, which has implications for accuracy and robustness in the context of this task. YOLOv11 was selected as one of the established YOLO models that supported instance segmentation, while RF-DETR was selected as a novel state-of-the-art real-time instance segmentation alternative. Furthermore, the two models were selected to each represent and consequently compare the two distinct main architectures within instance segmentation research; convolutional neural network (CNN) and vision transformer (ViT).

YOLOv11 is a single-stage convolutional detector that processes the entire image in a single forward pass [10]. Its architecture incorporates the C3k2 block and Spatial Pyramid Pooling Fast (SPPF) for multi-scale feature extraction [10]. The single-stage design is the primary reason YOLO-family models have been favoured for real-time applications: by avoiding the two-stage region proposal step, they achieve significantly lower latency than two-stage de-

tectors [10]. Instance segmentation support was added in later versions of the YOLO family, extending its applicability beyond bounding-box detection.

RF-DETR, by contrast, is a transformer-based detector built on the DINOv2 vision transformer backbone [11]. It replaces anchor boxes and Non-Maximum Suppression (NMS) with deformable cross-attention, giving the model richer spatial awareness over features. Sapkota et al. [11] found that RF-DETR reached stable performance in fewer training epochs than YOLOv12, and demonstrated stronger results in cluttered and occluded scenarios, which are properties directly relevant to multi-person scenes where performers frequently overlap. In our experiments RF-DETR met real-time performance requirements, demonstrating that transformer-based architectures are viable for live inference in this context. Instance segmentation support in RF-DETR was introduced only as of October 2025, meaning virtually no independent benchmarks exist for this capability beyond those published by Roboflow themselves [12, 13], at the time of writing. Evaluating it in an applied instance segmentation setting is therefore a direct contribution of this work.

Both models are evaluated here specifically on instance segmentation of video frames, which distinguishes this work from Sapkota et al. [11], who compared RF-DETR and YOLOv12 on object detection rather than instance segmentation. Furthermore, we evaluated YOLOv11 as opposed to YOLOv12.

2.3 Benchmarking and Annotation

Two additional models are used outside of the real-time inference pipeline, serving as reference points for evaluation. Segment Anything Model 3 (SAM3) is a promptable foundational segmentation model capable of producing highly precise instance masks [14]. Its computational cost makes it unsuitable for real-time deployment in complex multi-instance scenes, but this precision serves two purposes in this work: it is used to generate ground-truth segmentation masks, with a human-in-the-loop verification process, against which the real-time models are evaluated. Furthermore, its output also functions as an alternative benchmark representing the upper bound of segmentation quality achievable without real-time constraints.

Video Depth Anything is a monocular depth estimation model designed specifically for video input, providing temporally coherent and fine-grained depth maps [15]. While too slow for real-time use, it serves as an alternative benchmark to the MCR approach discussed in Section 2.1, offering a purely vision-based source of depth information that does not require a multi-camera setup.

2.4 Occlusion-Handling

The preceding sections have established the main building blocks of our pipeline: MC reconstruction for depth, and instance segmentation for per-pixel masking. This section situates our overall approach within the landscape of AR/MR occlusion strategies and explains how these components are combined to address the core problem.

Handling occlusion correctly in AR/MR fundamentally requires knowing the depth ordering of real and virtual elements at the pixel level. The established literature suggests two broad strategies for obtaining this information [16].

The first, which our approach follows but in real time, is a model-based approach which generally relies on a pre-existing 3D model of the environment which can be either immediately available or derivable through some set of computations [17, 18]. The general issue with the model-based approach is expensive computation and hence slow runtime, as well as data that is often unavailable. This approach can also be seen as a reconstruction-based approach.

Our strategy decouples occlusion composition from depth entirely by introducing instance segmentation, which is used to extract per-pixel masks for each performer, while depth is obtained explicitly from MC reconstruction, or more precisely, pose estimation from 3D skeletal data. Since this reconstruction is entirely built on pose estimation and without volumetric data, the resulting meshes are approximate, but the real value in this reconstruction is the hierarchical positioning of the meshes relative to each other. Aligning these two sources of information yields the depth ordering required for correct occlusion without relying on depth inference from the image itself. In other words, we compromise our reconstruction quality so that implicit occlusion compositing is no longer possible and instead perform both steps explicitly, and in return we intend to gain real-time reconstruction capacity.

The second general occlusion-handling approach relies on monocular depth estimation, using models such as Video Depth Anything or real-time sensor data to infer a per-pixel depth map from a single camera feed, which is then used directly for compositing. This requires little to no additional capture infrastructure, but introduces dependency on the accuracy of depth estimation models, which can be unreliable in complex or cluttered scenes. This can also be referred to as an explicit depth-based approach, and is an established and flexible method in AR and MR occlusion-handling research [2, 19, 20].

A third non-traditional strategy, proposed by Watson et al. [21], avoids explicit depth entirely. Rather than obtaining a depth map through estimation or reconstruction, they train a VIS architecture to intrinsically encode depth ordering. The underlying assumption is that a sufficiently capable segmentation model, trained on data annotated with occluded instances, can produce masks that implicitly reflect correct occlusion ordering. This is referred to as implicit depth: the model learns which parts of which objects are occluded and handles compositing accordingly, without a separate depth signal.

While an empirical comparison against the implicit depth approach of Watson et al. was initially proposed, it was ultimately excluded from the scope of this evaluation due to

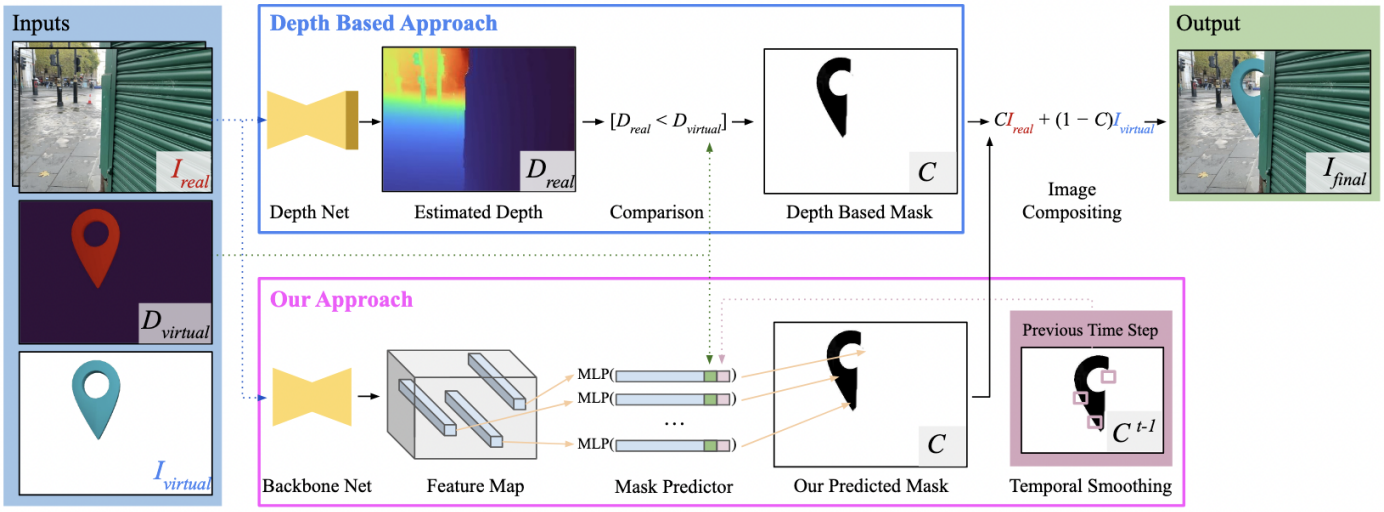


Figure 5: System diagram of methodology for the implicit depth approach of Watson et al., where depth, and consequently also occlusion, is implicitly inferred with AI. Our approach also suggests an alternative approach to occlusion-handling with the use of deep learning, but keeps the depth estimation and occlusion composition steps segregated.

fundamental architectural and environmental incompatibilities. The implicit pipeline relies on specific input modalities, most notably requiring pre-rendered 2D depth maps of the virtual object and a distinctly formatted set of camera parameters. This fundamentally conflicted with our methodology, which natively embeds and renders 3D virtual objects directly within a real-time Unity environment. Adapting our pipeline to continuously extract and format isolated **VOB** depth maps and translate the broadcast camera metadata into their specific framework presented a severe architectural mismatch.

2.5 Research Questions

Clear advantages of our approach include depth that is natively handled within the Unity ecosystem, which allows to seamlessly perform calculations in the context of an MR application, which is priceless as the application grows in complexity. Compared with other approaches, such as by Watson et al. [21], the pipeline is not intrinsically designed for MR but rather AR, where MR assumes that you may interact with virtual, and even real, objects. That is not to say that such an approach cannot be adapted for the Unity ecosystem, but rather highlighting the native MR aspect of our approach.

Furthermore, as we elaborate upon in Section 8, our pipeline combines several components in a novel manner to handle occlusions within MR, but the magnitude of the project is not extensive enough to go in-depth in our implementation within these various components. An example of this is that we use the pre-trained RF-DETR as opposed to further optimizations such as training on an alternative dataset specific to our domain or even model distillation. Thus proving the feasibility of our approach, particularly for our depth classification, would present a promising solution to the depth estimation bottleneck (as proposed in Watson et al.), specifically in the context of real-time broadcast MR footage.

Particularly compared with Watson et al., another aspect of our research was to investigate whether an alternative deep learning approach may be effective. They place this deep learning component in their methodology pipeline such that it performs both implicit depth estimation *and* implicit occlusion compositing. Therefore, we split the occlusion-handling process into two phases - depth estimation (by reconstruction) and occlusion compositing - differing from traditional approaches where *only* depth estimation is done (and occlusion compositing is implicit) *and* the Watson et al. approach where you essentially do both depth estimation nor occlusion compositing implicitly.

This leaves the main research question as: can the combination of MCR and real-time instance segmentation produce precise, real-time explicit depth classification and consequent explicit occlusion compositing for occlusion-aware MR broadcast footage?

Furthermore:

- How do CNN-based (YOLOv11) and transformer-based (RF-DETR) architectures compare in instance segmentation quality and real-time performance for multi-person broadcast footage?
- How does MC-derived explicit depth compare to monocular depth estimation (Video Depth Anything) as a source of depth information for occlusion compositing in MR applications?
- How close does real-time instance segmentation approach the segmentation quality achievable by a high-precision offline model (SAM3)?

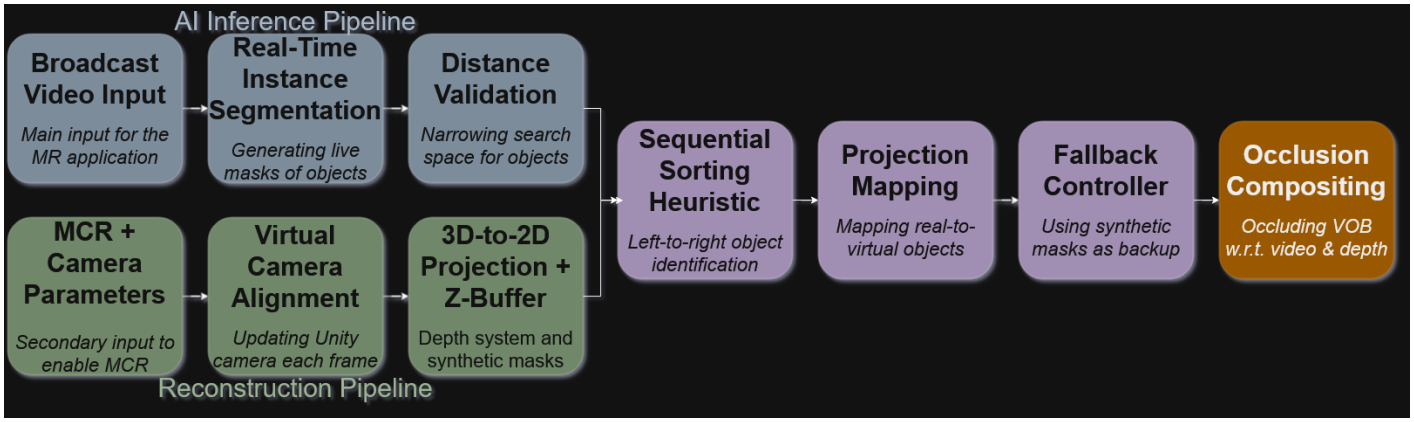


Figure 6: System diagram of methodology for our insertion MR application, combining MCR and AI in real time for composition, including the fallback controller when inference misses masks for a frame

3 Methodology

We consider two alternative forms of MR for this methodology in general: MR where objects from the real video footage **R** are replaced (replacement MR) and MR where **VOBs** are inserted into **R** (insertion MR). However, we only implement the insertion approach in this project. Figure 6 represents an overview of the project pipeline for our goal of insertion MR, which includes an AI inference pipeline as well as a reconstruction pipeline, which gives us the reconstruction **V**, before giving unique IDs to each performer and mapping the real mask **M** to their reconstructed counterpart **P**. Finally, this allows occlusion-aware composition for our MR application.

3.1 Reconstruction

The input to the reconstruction pipeline consists of multi-camera video footage which we capture simultaneously with the broadcast feed and that we process with Sony's markerless multi-view MC system to produce skeletal motion data in the Biovision Hierarchy (BVH) file format [22]. The MC system derives 3D joint positions by triangulating keypoints across multiple synchronised camera views, yielding a frame-aligned skeletal reconstruction of each performer. Crucially, this reconstruction shares a temporal reference with the broadcast footage, which is what allows us to meaningfully project the 3D skeletal data onto the 2D broadcast frame. We make the spatial alignment between the two possible by the frame-wise camera parameter data which our industry partner provides us, which encodes the intrinsic and extrinsic properties of the broadcast camera at each frame. This is elaborated upon in Section 4 and Section 4.3.

It is worth clarifying the role of multi-view data in our pipeline, as this is a potential source of confusion. Multi-view reconstruction methods offer high geometric precision but require synchronised camera arrays that are generally unavailable in standard broadcast settings. Our pipeline does rely on multi-view data, but only indirectly: the MC capture infrastructure provided by our industry partner uses a multi-camera setup to generate 3D skeletal reconstructions as a byproduct of their existing acquisition process. This data is a fixed external input to our system. Our pipeline itself operates entirely on single-view broadcast footage **R**, using the MC data only as a source of depth information rather than performing any multi-view reconstruction of its own.

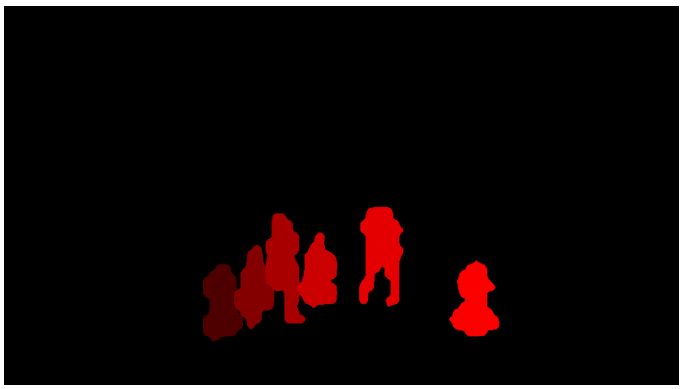
After obtaining the BVH file, we convert it to the Intermediate Data Format (IDF) [23] , which we stream directly into a Unity scene. We dynamically load the IDF data and render it as animating object meshes in real time. The result is that our real-world 3D spatial data is brought into the Unity environment, defined as the reconstructed performers $\{\mathbf{P}\}$ in the MC reconstruction **V** of our real input video **R**.

3.2 Segmentation

Our MR pipeline for handling occlusion requires instance segmentation in order to specify precisely which pixels belong to which instance such that our composition step can distinguish between instances based on depth. VIS would be ideal to accomplish this task throughout a video but a frame-wise instance segmentation model was selected instead to keep in line with the goal of real-time processing. SAMv2 [24] was tested on **R** but it quickly became clear that the real-time performance of a class-agnostic segmentation model such as SAMv2 would not be anywhere close to our goal of 30, never mind 60 FPS. Therefore we tested a couple class-specific models instead, and we evaluated YOLOv11 and RF-DETR as the two primary candidates, representing the dominant CNN and transformer-based architectures for real-time instance segmentation respectively.

For our initial segmentation experiments frames are pre-processed through a Region-of-Interest (ROI) masking step that constrains the input to the region of the frame occupied by $\{\mathbf{P}\}$, reducing the model's search space. The masked

(a) Weak signal



(b) Strong signal

Figure 7: Segmentation mask output $\{\mathbf{M}\}$ visualized for two frames only a few frames apart, where the weak signal occurred due to an unhandled occlusion from the segmentation model.

frame is converted to grayscale and gamma-corrected before being letterboxed to the model's native input resolution. The non-linear gamma correction is applied as:

$$I_{\text{out}} = I_{\text{in}}^{\gamma} \tag{1}$$

where I_{in} is the pixel intensity normalized to the range $[0, 1]$, and we empirically set $\gamma = 2.5$.

Additional contrast enhancement via Contrast Limited Adaptive Histogram Equalization (CLAHE) and other techniques such as image sharpening and contrast filtering were evaluated but excluded from the final pipeline, as they introduced noise without measurable improvement. Note that this pre-processing was explicitly skipped for RF-DETR due to results discussed in Section 6.1.

Post-inference, the raw mask logits from each architecture must be converted into discrete binary segmentation masks. For our primary model, RF-DETR, this is achieved directly through a strict step-function threshold μ. Given a raw predicted pixel intensity p, the finalized binary pixel value p_{binary} is calculated as:

$$p_{\text{binary}} = \begin{cases} 255, & \text{if } p \geq \mu \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

where μ is an empirically tuned confidence threshold.

It is worth noting that YOLO11 required an extensive post-processing pipeline prior to this thresholding—including Gaussian blurring, morphological dilation, and median filtering—to handle spatial noise. However, these processing operations were found to actively degrade the quality of RF-DETR's transformer-based outputs through qualitative inspection. Consequently, RF-DETR masks are not smoothed and are thresholded directly. The specific threshold values are detailed in Section 4.7.

To filter out background noise or falsely segmented non-performers (e.g., stagehands or camera operators), a spatial distance validation is applied. The geometric center of each $\mathbf{M}$ is calculated and compared against the projected centers of the known $\{\mathbf{P}\}$ (Section 3.3. If the center of some $\mathbf{M}$ exceeds a predefined safe distance threshold from $\{\mathbf{P}\}$, it is rejected. However, a dynamic bypass mechanism is built into this validation: during sequences of intense zoom (defined by a focal length threshold), this distance threshold is explicitly ignored. Intense optical zooming exponentially magnifies minor physical camera misalignments, which would otherwise trigger false rejections of valid masks.

With our models selected and processed, we performed a comparative experiment evaluating both models on a one-minute segment of $\mathbf{R}$. A Pareto analysis of mAP against inference time showed RF-DETR to be the superior candidate (see Section 5.1 and Section 5.2), and it was therefore selected for the final occlusion experiments.

Another attempt to further optimize the segmentation was to perform inference strictly within the bounds of the bounding boxes (with some generous padding) of $\{\mathbf{P}\}$ in an attempt to limit the search space of the segmentation model. While this might have yielded some precision improvement especially for our CNN-based YOLOv11 which improves its performance with narrowed search space (see Section 6.1), it predictably took a heavy hit on latency as the inference overhead was multiplied by the number of visible performers in the frame. This attempt was therefore scrapped.

Figure 7 highlights the segmentation differences between a few frames where the weak signal determines the frame to be missing $\{\mathbf{M}\}$. For the strong signal you can see six distinct masks which will clearly surpass the threshold and hence the frame *does* contain masks. The differences in intensity of the distinct masks seen in the figures is an implementation technique to give a unique identity to each mask in the composition step which is done at the shader level.

3.3 Projection Mapping

Projecting $\{\mathbf{P}\}$ onto $\mathbf{R}$ follows the standard pinhole camera model [25]. A 3D world point $\mathcal{P}_w$ is first transformed to camera space via the extrinsic parameters:

$$\mathcal{P}_{\text{cam}} = \mathcal{R}\mathcal{P}_w + t \tag{3}$$

where $\mathcal{R}$ is the 3×3 rotation matrix encoding the camera orientation and $\mathbf{t}$ is the translation vector encoding the camera position. This step expresses the world-space point in the camera's own coordinate frame, accounting for where the camera is physically located and which direction it is pointing. In our setup, $\mathcal{R}$ and $\mathbf{t}$ are updated per frame from the pan, tilt, and tripod parameters described in Section 4.4.

The resulting $\mathcal{P}_c$ is then projected onto $\mathbf{R}$ via the intrinsic parameters:

$$u = f \cdot \frac{X_w}{Z_w} + c_x, \qquad v = f \cdot \frac{Y_w}{Z_w} + c_y \tag{4}$$

where (u, v) are the resulting 2D pixel coordinates, f is the focal length, and (c_x, c_y) is the principal point. The division by Z_w implements perspective projection: points further from the camera have a larger Z_w, which shrinks their projected coordinates toward the principal point, producing the natural effect of distant objects appearing smaller. Multiplying by f scales the result to pixel units, and adding (c_x, c_y) shifts the origin from the camera's optical axis to the image coordinate frame.

Crucially, Z_w serves a dual purpose in our pipeline. It is both the quantity that drives the perspective projection and the depth value used directly in the z-comparisons defined in Section 3.6. This means that depth information is obtained as a direct byproduct of the projection itself, without requiring any separate depth estimation step.

In our implementation this projection is handled per-frame through Unity's `Camera.WorldToViewportPoint()` function, which returns both the normalised 2D viewport coordinate and the depth of the point relative to the camera. The per-frame intrinsic and extrinsic parameters defined in Section 4.5 are applied to the Unity camera before each call, ensuring $\mathbf{V}$ accurately mimics that of $\mathbf{R}$ at each time step.

3.4 Sequential Sorting

Because the real-time inference models perform frame-by-frame instance segmentation, they do not inherently track object identities over time. To assign the correct depth value from the Z-buffer to each $\mathbf{M}$, we must deterministically map $\{\mathbf{M}\}$ to their corresponding $\mathbf{P}$. For this we introduce a spatially-sorted projection mapping algorithm.

The mapping process begins by projecting the 3D world coordinates of $\{\mathbf{P}\}$ onto the 2D camera plane using the real-time $\mathbf{R}$ camera metadata (Section 3.3). Any $\mathbf{P}$ whose projected coordinates fall outside the visible screen bounds (or behind the camera) is culled. The remaining visible $\{\mathbf{P}\}$ are then sorted strictly along the horizontal axis (X-axis) of the screen from left to right. A unique identifier is assigned sequentially based on this physical ordering, such that the leftmost $\mathbf{P}$ becomes ID 1.

Finally, the remaining validated $\{\mathbf{M}\}$ are also sorted horizontally from left to right. Because both the projected $\{\mathbf{P}\}$ and $\{\mathbf{M}\}$ share the exact same spatial sorting logic, they are naturally paired. The leftmost $\mathbf{M}$ is rendered into the composition buffer with a pixel value corresponding to ID 1, seamlessly linking it to the depth data of the leftmost $\mathbf{P}$.

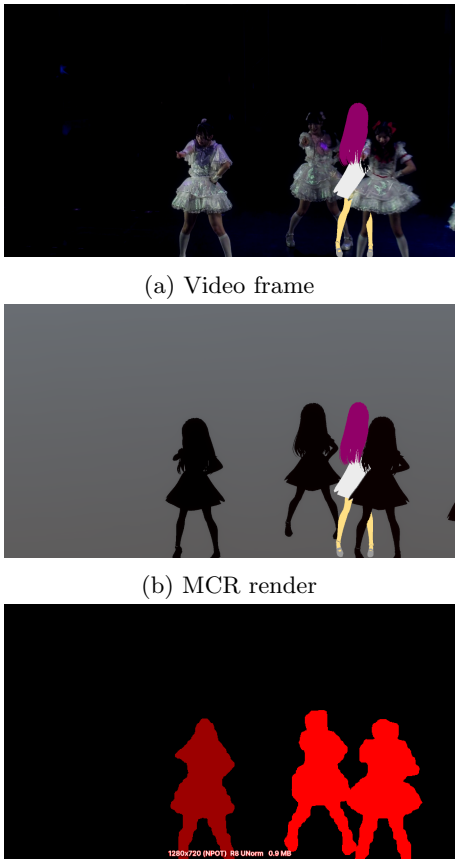
3.5 Fallback

When one or more $\mathbf{M}$ are missing from the inference step, the pipeline falls back to using the *synthetic masks*, defined as the mesh outlines of $\{\mathbf{P}\}$. However, for some frame i when $\mathbf{M_i}$ are only missing for that frame and not multiple consecutive frames, $\{\mathbf{M_{i-1}}\}$ from the previous frame are re-used instead. Since it is quite common for the segmentation model to drop its segmentation for a single frame, this ensures less flickering from live to synthetic masks during runtime. An example of a frame missing $\mathbf{M}$ is illustrated in Figure 7, and the fallback itself in Figure 8.

It should be mentioned that the criteria for $\mathbf{M}$ missing is not entirely trivial; for some frames there could indeed be zero pixels classified as a mask, but other frames could have a very weak mask signal. Therefore an empirically determined threshold was introduced to account for this. Lastly, for a frame when this threshold is met for some $\mathbf{M}$ of some $\mathbf{P}$ but not for all, this is also considered as missing $\mathbf{M}$ for this frame and the fallback commences.

Considering this fallback approach, padding is added to the meshes of $\{\mathbf{P}\}$. The exact padding magnitude is manually determined through qualitative inspections of the synthetic masks. Due to misalignment (as explained in Section 7) as well as lack of volumetric information of the real performers, $\{\mathbf{P}\}$ do not correspond perfectly with the real performers, and this is particularly visible during zooming sequences of $\mathbf{R}$, where some $\mathbf{R}$ may be clearly larger than its corresponding real performer. Therefore, padding that may qualitatively appear to reduce the general proportion difference during a non-zoomed sequence may adversely affect zoomed sequences, and vice versa. We prove through an ablation study that a chosen padding improved performance over non-padded runs in Section 6.6.

Attempts to further improve this fallback technique included testing various parameters for allowed number of frames dropped before synthetic masks are used, but having more than a single frame allowed resulted in severe spatial offset especially during sequences where the performers rapidly moved horizontally or vertically. We also attempted to manually translate $\{\mathbf{M_{i-1}}\}$ with respect to the centroids of the corresponding $\{\mathbf{P}\}$, but this was not feasible in terms of computation speed.



(a) Video frame

(b) MCR render

(c) Segmentation mask

Figure 8: Fallback occlusion handling: live video frame (left), MCR character render (center), and the corresponding RF-DETR segmentation mask (right). The produced segmentation masks do not match the number of detected performers (4) with the MCR outlines as reference, thus the MCR outlines (synthetic masks) are themselves used for the occlusion as an example of the fallback mechanism in use.

3.6 Z-comparisons

To determine the correct depth ordering between real performers and inserted **VOBs**, we perform per-frame z-comparisons using the MC-derived skeletal data.

For some reconstructed performer **P** in **V**, we compare its z-value with that of the inserted **VOB** relative to the camera, producing a z-buffer that is applied to the segmentation masks in the composition step.

We consider **P** to be in-front of **VOB** if, and only if, it satisfies:

$$z_{\mathbf{P}} \leq z_{\mathbf{VOB}} \tag{5}$$

Consequently, **VOB** is in-front of **P** if, and only if:

$$z_{\mathbf{VOB}} > z_{\mathbf{P}} \tag{6}$$

The pipeline utilizes binary Z-buffer thresholding in virtual world space. Consequently, partial volumetric occlusions are resolved by forcing a strict front-to-back ordering, meaning an object cannot be both behind and in-front of another.

This does not apply for "see-through" cases, such as for a virtual donut, where the open space inside the donut would still render objects behind it, assuming the donut is placed in front of those objects. Partial occlusions, however, refer to cases where two objects are almost completely on the same z-index, and here we make the simplification of selecting one to be in-front of the other.

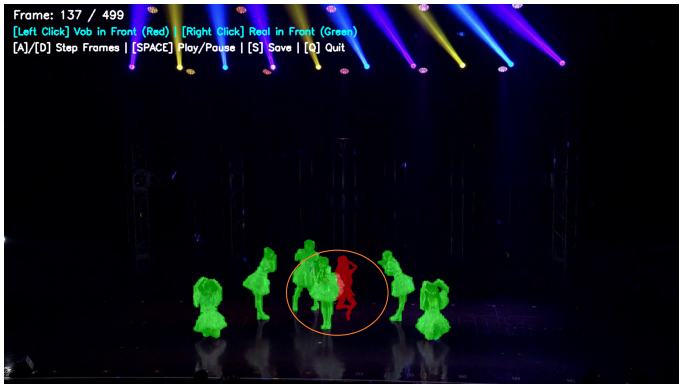
A further consequence of binary thresholding arises from the segmentation model itself, which is the tendency to occasionally merge overlapping objects into a single segmentation mask, particularly during close-proximity choreography. To handle these scenarios, our pipeline operates under a 'closest-takes-all' depth heuristic. The assumption is that the closest object is physically occluding the majority of the entities behind it, allowing its depth value to safely represent the entire merged mask without severely degrading the MR realism. This is particularly quantified through our Visual Fidelity metric from Section 3.11.

Lastly, it is worth to mention that a *thickness factor* is added to the inserted **VOB** to account for the hard thresholding, intended to give a more natural cut-off point along the z-axis and to improve the final occlusion.

3.7 Monocular Depth Estimation

As an alternative and comparison to the MCR-based depth we also implement monocular depth estimation. Video Depth Anything (VDA) is used to produce a metric-based depth map, which is done during the offline stage and is not intended as part of the real-time pipeline. However, in order to align this depth system with our existing reconstruction and camera alignment, several steps are required.

We load the depth map values **D** corresponding to a vector of pixel intensities equal in size to that of the resolution of **R**. For the region of the two performers most near and most distant relative to the camera $\Omega_{\text{near}} \subset \mathbf{D}$ and



(a) **VOB** incorrectly determined to be occluding the real performer

(b) Occlusion corrected by determining the real performer to be occluding **VOB**

Figure 9: Manual ground-truth annotator GUI tool that detects intersections between choreographed **VOB** and existing ground-truth SAM3 masks and allows user to determine which object occludes which.

$\Omega_{\text{far}} \subset \mathbf{D}$, we aim to calculate the difference in distance between them, which requires two representative anchoring scalar values which we take as the geometric centers ω_{near} and ω_{far} respectively.

We compare the difference of distance between $\mathbf{P}_{\text{near}}$ and $\mathbf{P}_{\text{far}}$ where z_{near} and z_{far} are the corresponding depth values in 3D world space, and we can calculate the depth scale s between $\mathbf{D}$ and $\mathbf{R}$ and the depth offset o between z_{near} and the scaled ω_{near}:

$$s = \frac{z_{\text{far}} - z_{\text{near}}}{\omega_{\text{far}} - \omega_{\text{near}}} \qquad o = z_{\text{near}} - (\omega_{\text{near}} \cdot s) \tag{7}$$

Finally, we sample the 10th percentile [26, Section 5.3.2] of $\mathbf{D}$ within Ω_{near} and within Ω_{far}, respectively, as an attempt to ensure the depth calculation is less influenced by outliers, and then apply s and o to receive the sampled 10th percentile pixel:

$$\hat{D} = (P_{10}(\Omega) \cdot s) + o \tag{8}$$

where $P_x(\Omega)$ denotes the x-th percentile of distribution Ω.

3.8 Composition

With projection mapping, segmentation masks and z-buffer all in place, we perform the final MR occlusion composition. The composition step is implemented through a custom shader attached to the **VOB**. The shader receives three inputs per frame: $\mathbf{R}$, $\mathbf{M}$, and the depth values of $\{\mathbf{P}\}$ derived from the z-buffer. For each pixel of **VOB**, the shader evaluates the depth comparison established in Section 3.6 and applies binary alpha clipping accordingly. If Equation (5) holds true, the alpha is set to zero, causing the corresponding video pixel to be revealed instead. Otherwise the alpha remains at one and the **VOB** pixel is rendered. $\mathbf{R}$ is rendered as a background canvas, with **VOB** composited over it in an overlay pass, ensuring correct draw order independent of scene geometry.

3.9 Annotation

Ground-truth annotations were constructed as follows. The **VOB** outline mask was rendered for each frame across two predefined sequences of $\mathbf{R}$, with the **VOB** following a fixed choreography so that its projected outline is identical across all experimental runs for a given sequence. These **VOB** outline masks were then combined with the offline SAM3 ground-truth segmentation masks described in Section 2.3, which were verified through a human-in-the-loop annotation process. For frames in which the **VOB** outline and some $\mathbf{M}$ intersect, the correct depth ordering was determined manually using a dedicated annotation tool: for each intersection in each frame, an annotator recorded whether the **VOB** should appear in front of or behind the intersecting $\mathbf{M}$. An illustration of this annotation process can be seen in Figure 9.

The result is a per-frame, per-intersection ground-truth CSV defining the correct occlusion decision for each observable occlusion event across both sequences. The two sequences differ in scene complexity: frames 0–500 feature greater camera movement and more challenging lighting conditions, while frames 500–1000 present less complex scenes with fewer occlusion events, providing a basis for evaluating performance across varying difficulty levels.

3.10 Temporal Requirements

In this project we define a system as real-time for MR if the total per-frame processing overhead τ satisfies:

$$\tau_{30} \leq \frac{1}{30} \approx 33.3 \text{ ms}$$

with an ideal target of:

$$\tau_{60} \leq \frac{1}{60} \approx 16.7 \text{ ms}$$

Both thresholds are grounded in established conventions for real-time and MR applications. 60 FPS is widely re-

garded as the standard for high frame rate interactive systems [27], and platforms such as Microsoft HoloLens explicitly recommend a target of 60 FPS to avoid perceptual degradation in MR experiences [28]. 30 FPS represents a broadly accepted lower bound for real-time processing in the MR community [29] [30], below which temporal artifacts become perceptually disruptive. Since **R** is downsampled to 30 FPS, the τ_{30} threshold is the operative requirement for this work. However, a pipeline meeting the τ_{60} threshold would retain sufficient headroom to generalise to a native 60 FPS broadcast stream without modification, which remains the aspirational target.

3.11 Benchmarking

We evaluate the occlusion pipeline across three phases, each targeting a distinct aspect of the compositing result. All phases operate on collected data consisting of per-frame depth values per **P**, {**M**}, **VOB** outline masks, and runtime metrics captured across the full pipeline. Phase 1 and 2 are evaluated through a confusion matrix of correctly and incorrectly classified frames, from which accuracy, precision, recall, and F1-score can be derived.

Phase 1: Detection evaluates whether the pipeline correctly identifies frames in which a real performer intersects with the **VOB**. A true positive is defined as a frame where both the ground-truth and the predicted result agree that an intersection exists. This phase is sensitive to camera alignment quality, as the projected **VOB** position directly determines whether an occlusion event is registered.

Phase 2: Occlusion evaluates the correctness of the depth ordering decision. A true positive is defined as a frame where both the ground-truth and the predicted result agree that the **VOB** should be rendered in front of a given performer, and vice versa for true negatives. For MCR-based runs this phase depends on the accuracy of the z-buffer derived by {**P**}, while for VDA-based runs it depends on the reliability of the monocular depth estimate.

Phase 3: Region evaluates the visual fidelity of the composited result through the IoU between the ground-truth intersection region and the predicted intersection region, i.e. the visible area of the **VOB** after occlusion is applied. This phase captures the quality of {**M**} as they are utilized for the final composition.

The recall of our Phase 1 confusion matrix essentially says: out of all the frames where the **VOB** actually overlapped with a dancer, what percentage did the pipeline successfully notice? This is crucial to evaluate because undetected occlusions (False Negatives) lead to unexpected behavior of the **VOB**, often resulting in unnatural rendering which breaks the MR immersion. Although the precision is less important in that sense, since hallucinated occlusions (False Positives) are less likely to break the immersion, simply detecting excess occlusions would lead to a perfect recall, thus we also provide the F1-score to ensure this is not the case for our system.

Furthermore, for Phase 2 we evaluate the accuracy since this is a strict binary classification test. We are interested in the system's performance in correctly classifying the **VOB**'s position (True Positives and True Negatives).

For Phase 3, we evaluate the visual fidelity of {**M**} using two distinct metrics: Standard IoU and Boundary IoU [31]. For two binary masks **M1** (ground truth) and **M2** (predicted), the standard Intersection over Union (IoU) is defined as:

$$\mathrm{IoU}(\mathbf{M1}, \mathbf{M2}) = \frac{|\mathbf{M1} \cap \mathbf{M2}|}{|\mathbf{M1} \cup \mathbf{M2}|} \tag{9}$$

where $|\cdot|$ denotes the number of pixels (or area).

Boundary IoU restricts evaluation to a narrow band of pixels around the contour of each mask [31]. Let $\mathbf{M}_d$ denote the boundary region of a mask **M**, defined as the set of pixels inside **M** that are within a Euclidean distance d of its exterior.

Given two masks **M1** and **M2**, alongside their respective boundary regions $\mathbf{M1}_d$ and $\mathbf{M2}_d$, the Boundary IoU is defined as:

$$\mathrm{IoU}_{\mathrm{boundary}}(\mathbf{M1}, \mathbf{M2}) = \frac{|(\mathbf{M1}_d \cap \mathbf{M1}) \cap (\mathbf{M2}_d \cap \mathbf{M2})|}{|(\mathbf{M1}_d \cap \mathbf{M1}) \cup (\mathbf{M2}_d \cap \mathbf{M2})|} \tag{10}$$

In this work, we set $d = 10$ pixels.

Standard IoU measures the total overlapping area between our **M** and the ground truth. This essentially tells us how well the pipeline captured the mass of the performer. While a high Standard IoU confirms that the system correctly localized the performer, it is a highly forgiving metric; some **M** that perfectly covers a large torso but completely misses the fingers will still score mathematically high. Therefore, while necessary to prove general spatial tracking, Standard IoU alone is insufficient for evaluating the perceptual quality of an MR broadcast.

To address this, we also report Boundary IoU, which strictly limits the evaluation to a narrow band of pixels along the contour of some **M**. In MR, the illusion of realistic occlusion is entirely dependent on these edges. If some **M** bleeds into the background or has a jagged perimeter, the composited **VOB** will exhibit clipping, instantly breaking the immersion like a poorly keyed green screen. By reporting Boundary IoU alongside Standard IoU, we can explicitly quantify the crispness of the silhouette. Furthermore, analyzing the discrepancy between these two metrics allows us to diagnose specific failure cases—such as determining whether dynamic camera motion confused the system about *where* the performer was (low Standard IoU), or if it simply blurred the exact pixel-edges of their clothing (low Boundary IoU).

3.12 Runs

We evaluate the following experimental runs:

- **Live + MCR**: real-time RF-DETR segmentation with MC-reconstruction-derived depth

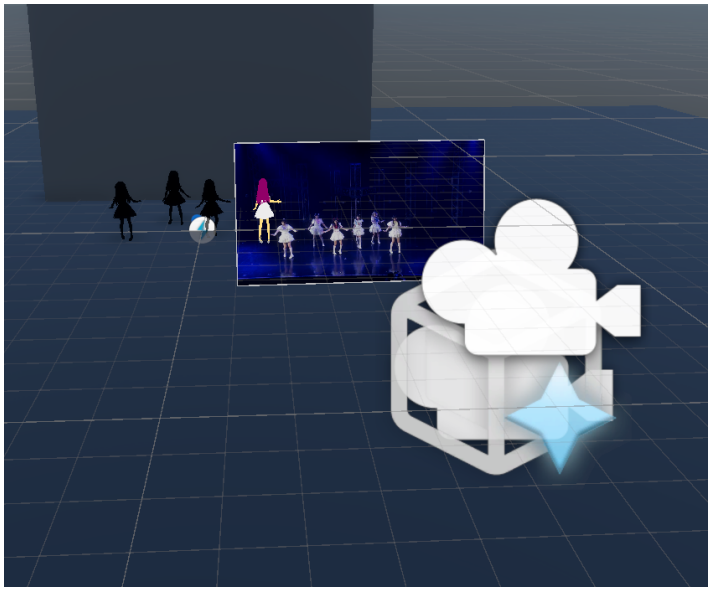


Figure 10: Scene setup for occlusion-handling within Unity, which shows the reconstructed performers $\{\mathbf{P}\}$ in the background, the video canvas for $\mathbf{R}$ with the **VOB** overlay, and the camera which transforms in real time

- **Offline + MCR**: high-precision offline segmentation with MC-reconstruction-derived depth
- **Offline + VDA**: high-precision offline segmentation with Video Depth Anything-derived depth
- **Synthetic + MCR**: only synthetic fallback masks with MC-reconstruction-derived depth
- **Live + VDA**: real-time RF-DETR segmentation with Video Depth Anything-derived depth

The Offline + MCR run serves as the primary upper-bound reference for segmentation quality within our depth framework, isolating the contribution of the segmentation model from that of the depth source. Offline + VDA provides an alternative upper bound that replaces both components with offline alternatives, allowing a direct comparison between MCR and VDA as depth sources independently of real-time constraints. The Synthetic + MCR run isolates the contribution of the synthetic fallback mechanism and forms the basis of an ablation study examining the effect of the silhouette scaling factor, which accounts for the geometric discrepancy between some $\mathbf{P}$ and the corresponding real performer silhouette.

4 Experimental Setup

To validate the real-time occlusion pipeline detailed in Section 3, a comprehensive offline evaluation framework was constructed. While the target application is a live broadcast environment where the video feed $\mathbf{R}$ and its accompanying camera metadata are streamed continuously, our experimental setup utilizes pre-recorded data to ensure deterministic and reproducible benchmarking. Specifically, $\mathbf{R}$ is processed as a standard video file, and its corresponding per-frame intrinsic and extrinsic camera parameters are read synchronously from a locally stored CSV file. Transitioning this setup to a live data stream in a production environment is presumed to be a straightforward engineering adaptation. The following subsections detail the hardware environment, dataset characteristics and the Unity environment setup.

4.1 Experiment Hardware

All experiments were conducted on a system equipped with an NVIDIA GeForce RTX 5080 GPU with 16 GB of graphics memory, an Intel Core Ultra 7 265K processor clocked at 3.90 GHz, and 32 GB of system RAM.

4.2 Dataset

The input video was recorded in a broadcast studio setting featuring six performers forming a pop group executing a choreographed dance routine. The video was captured using a Sony HDC-5500 camera with active panning, tilting, and zooming throughout the sequence, introducing dynamic camera motion representative of live broadcast conditions. $\mathbf{R}$ was recorded at 59.94 FPS and 1920×1080 resolution

Table 1: Camera Parameters

Parameter	Description
Focal Length	The lens's focal length in pixels
Principal Point	The optical center of the image
Focus Point	Range of focus
F-number	The iris of the camera
Radial Distortion	Coefficients for radial distortion
Nodal Offset	Nodal offset from stand to sensor
Encoded Pan	The registered pan of the camera
Encoded Tilt	The registered tilt of the camera

with a total duration of approximately five minutes, and was downsampled to 30 FPS and 1280×720 resolution for the purposes of this project. This was done to reduce the annotation overhead for ground-truth segmentation as well as to reduce runtime during experimentation. A one-minute subset was used for all experiments.

The one-minute subset is divided into two evaluation sequences of 500 frames each. Sequence 1 (frames 0–500) is characterised by greater camera movement and more challenging lighting conditions, making it the more demanding sequence for segmentation. Sequence 2 (frames 500–1000) presents a less complex scene with more stable camera behaviour, though occlusion events remain present. Sequence 1 contains **417** annotated occlusion events and Sequence 2 contains **706**, providing a basis for evaluating performance across varying scene complexity.

Ground-truth segmentation masks were generated using a human-in-the-loop pipeline. Initial masks for all six performers were proposed by SAM3 and subsequently verified and refined frame-by-frame by human annotators to ensure pixel-level accuracy and temporal consistency across both sequences.

4.3 Unity Pipeline

The core of the project is implemented within a Unity application that handles reconstruction, segmentation, and composition in a single integrated pipeline. The Unity application simultaneously renders the $\mathbf{V}$ and runs segmentation inference on $\mathbf{R}$, with both processes running in parallel to produce the final occlusion-aware composite. The MC data and broadcast footage used in this project were collected through the industry partner and are confidential, and are therefore not openly available, with only selected previews being presented.

$\mathbf{R}$ is loaded directly into Unity and rendered as a background canvas. $\mathbf{V}$ is streamed into the scene via the IDF format as described in Section 3.1, with the Unity camera updated per-frame according to the intrinsic and extrinsic parameters of Table 1, ensuring the virtual camera accurately mimics the perspective of the real broadcast camera at each time step. It is worth mentioning an initial temporal offset ($\lambda = 200$ ms) at the beginning of $\mathbf{V}$. We account for this by manually finding the alignment point, which would have to be done differently during an actual live broadcast.

Translating the physical camera properties of the Sony HDC-5500 into Unity's camera model required compromise, as Unity's built-in camera does not expose a complete equivalent of a production broadcast camera system. The properties successfully mapped into Unity are focal length, focus distance, aperture, radial distortion, position, and orientation. Achieving accurate alignment between the real and virtual camera perspectives is critical to the quality of the projection mapping and consequently the occlusion result, and residual misalignment is reflected in the spatial artifacts discussed in Section 7.

Instance segmentation is performed directly within Unity using RF-DETR, loaded as an ONNX model and executed via the ONNX Runtime library in C#. The native Unity Inference Engine was found to be incompatible with the model architecture and was therefore replaced with ONNX Runtime for all inference. The exported `.onnx` model receives each frame as input and produces the predicted segmentation masks $\{\mathbf{M}\}$, which are passed directly into the composition shader described in Section 3.8.

The software environment was standardized across all experiments to ensure reproducibility. The pipeline was developed in **Unity 6.000.51f1** using the **High Definition Render Pipeline (HDRP)** for high-fidelity composition. Neural network inference was handled via **ONNX Runtime 1.23.2**. The model specifications used for the benchmarks are detailed in Table 9.

4.4 Camera Setup

The camera used for the main example is Sony HDC-5500 with an image resolution of 1920x1080 and 59.94 FPS. As part of the scene reconstruction we track MC data of the 6 performers part of the act. This is also done at 59.94 FPS, and its coordinate system is placed approximately 3.5 meters behind the center of the stage, i.e. in the positive z-direction. Note that this specific camera setup operates with a z-up coordinate system, i.e. x-forward, but we chose to translate this into a y-up equivalent for consistency with Unity.

Furthermore, the initial location for the Shotoku SH120VR tripod of the camera is defined as (in centime-

ters):

$$\mathbf{tripod} = \begin{bmatrix} 205.05 \\ 509.1 \\ -1567.1 \end{bmatrix}$$

For this coordinate system, in contrast with the MC system, the origin is defined as the approximate center of the stage of the act. It is also then translated to a y-up system.

There is a sensor position offset defined as (in centimeters):

$$\mathbf{offset_{sensor}} = \begin{bmatrix} 1.5 \\ 19.3 \\ 15.3 \end{bmatrix}$$

Finally, the sensor size for HDC-5500 is 9.6mm for the width and 5.4mm for the height.

4.5 Camera Parameters

The intrinsic parameters are specific to a certain camera setup, while the extrinsic parameters change as the camera moves [25]. All the relevant parameters are listed in Table 1.

4.6 Pipeline Parameters

To ensure reproducibility across all experimental runs, specific empirical thresholds were established for the projection mapping and fallback mechanisms. For the spatial distance validation described in Section 3.4, the safe distance threshold was set to 0.075 of the normalized screen dimensions. The zoom bypass mechanism was configured to trigger when the broadcast camera's focal length reached or exceeded 25 units. For the synthetic fallback ablation study (Section 3.5), the padded $\{\mathbf{P}\}$ were scaled by a factor of 1.2 in width and 1.1 in height relative to their default geometric bounds. Furthermore, we use a thickness factor of 0.25 to account for the hard thresholding.

Finally, the missing masks threshold was set to 300 pixels per visible performer out of the 1280x720 possible pixels, and was found to work well, but should be selected with care with respect to the resolution of $\mathbf{R}$ and should be tinkered with prior to live runs to ensure $\{\mathbf{M}\}$ are consistently natural. If the value is too low, noisy segmentation signals may occur, for example through multiple random submasks within the region of one performer.

4.7 Segmentation Experiments

To evaluate segmentation performance and justify the model selection described in Section 3.2, we conduct a comparative experiment between RF-DETR and YOLOv11 on R. Four model variants are evaluated: RF-DETR-M, RF-DETR-L, YOLO11-M, and YOLO11-X, representing both smaller and larger variants of each architecture. RF-DETR is evaluated without pre-processing, as transformer-based architectures were found through qualitative inspection to be adversely affected by the applied pre-processing pipeline, while YOLO11 variants are evaluated both with and without pre-processing. RF-DETR does not employ NMS by design due to its deformable attention mechanism, while YOLO11 variants use the default NMS configuration provided by the model implementation.

Each model variant is evaluated at three confidence thresholds: 0.35, 0.45, and 0.55. These values were selected based on qualitative inspection of preliminary inference outputs: thresholds below 0.35 produced excessive low-quality masks, while thresholds above 0.55 led to a notable reduction in detected instances. The range 0.35–0.55 was therefore selected to systematically identify the optimal operating point for each model.

For pre-processing, YOLO11 inputs were smoothed with a Gaussian blur using a 5×5 kernel, while RF-DETR inputs received no additional filtering beyond ROI masking and letterboxing, as qualitative inspection showed that further processing degraded transformer-based inference quality. For post-processing, YOLO11 mask outputs were thresholded at 0.60, followed by morphological dilation with a 7×7 elliptic kernel and a median blur pass to reduce boundary noise. RF-DETR mask outputs were thresholded directly at 0.30.

Performance is evaluated using mean average precision (mAP) at an IoU threshold of 0.50, computed against the SAM3 ground-truth masks. Inference time is measured as median per-frame latency and evaluated against the thresholds defined in Section 3.10. A Pareto analysis of mAP against inference time is used to identify the optimal model variant, which is subsequently used for all occlusion experiments.

5 Results

The results are presented in two parts. The first covers the segmentation experiments, comparing RF-DETR and YOLOv11 across latency and mask fidelity to establish the model selection for the pipeline. The second covers the occlusion experiments, evaluating the full pipeline across the three phases defined in Section 3.11 and across the five experimental runs defined in Section 3.12. System specifications can be found in Section 4.1 and Table 9.

5.1 Segmentation Latency

Inference times were recorded by aggregating the raw neural network inference duration and the subsequent NMS step. Note that the first 10 frames of each sequence were excluded from the final averages to account for TensorRT warmup times.

As detailed in Table 2, three of the four evaluated model architectures comfortably operated within the real-time threshold. The transformer-based RF-DETR-M achieved the lowest overall latency, averaging just 8.94 ms per frame at a 0.35 confidence threshold. YOLO11-M and the larger RF-DETR-L architectures yielded comparable latencies, averaging 12.36 ms and 12.45 ms respectively. Conversely, the heaviest CNN architecture, YOLO11-X, averaged 17.52 ms per frame, effectively failing the τ_{60} real-time ceiling.

Table 2: Mean inference latency (Inference + NMS) across tested segmentation models. Values represent the 0.35 confidence threshold configurations, and are tested against the ground-truth SAM3 masks.

Model	Architecture	Latency (ms)	FPS
RF-DETR-M	Transformer	8.94	111.8
YOLO11-M	CNN	12.36	80.9
RF-DETR-L	Transformer	12.45	80.3
YOLO11-X	CNN	17.52	57.1

5.2 Segmentation Accuracy

Spatial accuracy was quantified using mean Intersection-over-Union (mIoU) against offline SAM3-generated ground truth masks, directly evaluating pixel-level mask precision. The results, summarized in Table 3, demonstrate that RF-DETR-L achieved the highest absolute spatial fidelity, yielding an mIoU of 0.731 at the 0.35 confidence threshold. The heaviest CNN model, YOLO11-X, performed second best (mIoU = 0.701), while the lighter, highly-performant RF-DETR-M maintained a competitive mIoU of 0.691. As expected, increasing the confidence thresholds from 0.35 to 0.55 resulted in a steady degradation of spatial accuracy across all models, as the strict confidence requirements limited the number of predicted masks.

Table 3: Mean Intersection-over-Union (mIoU) scores evaluating pixel-level mask fidelity across varying confidence thresholds.

Model	Conf=0.35	Conf=0.45	Conf=0.55
RF-DETR-L	**0.731**	0.715	0.691
YOLO11-X	0.701	0.690	0.671
RF-DETR-M	0.691	0.659	0.604
YOLO11-M	0.632	0.628	0.614

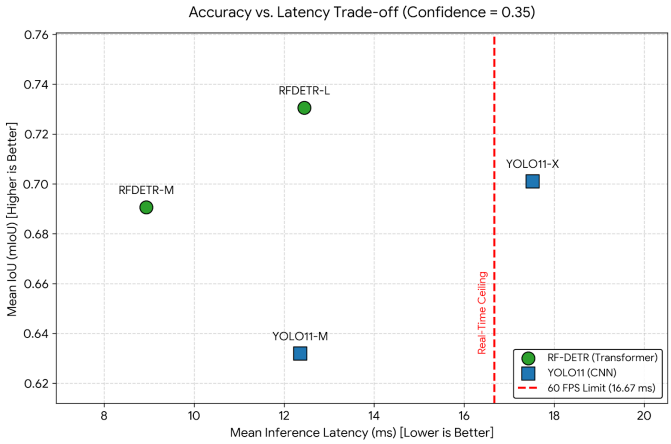


Figure 11: Mean IoU (mIoU) versus Inference Latency at a 0.35 confidence threshold. RF-DETR-L provides the optimal upper bound of spatial accuracy while operating strictly below the τ_{60} (60 FPS) threshold.

5.3 Phase 1: Intersection Detection

With the segmentation experiments completed and for the first phase of the MR occlusion pipeline we evaluate the system's ability to successfully detect a physical intersection between the inserted **VOB** and the real performers in the 2D broadcast frame. To evaluate robustness, this was cross-evaluated across Sequence 1 (more aggressive panning and zooming) and Sequence 2 (more stable camera but more occlusions). For Sequence 1 and 2, there were **417** and **706** occlusions, respectively. Table 4 shows the results of Phase 1.

Though not the sole reason for undetected intersections, the drop rate of the live methods significantly influence this metric. These can be seen in Table 5.

Table 4: Phase 1 Detection Metrics. Compares Recall and F1-Scores across dynamic (Seq 1) and stable (Seq 2) camera conditions.

Method	Sequence 1 (417 Occlusions)		Sequence 2 (706 Occlusions)	
	Recall	F1-Score	Recall	F1-Score
Offline (Baseline)	1.000	1.000	1.000	1.000
LiveMCR	0.767	0.835	**0.891**	**0.908**
LiveVDA	**0.772**	**0.836**	**0.891**	**0.908**
Synth. (Pad.)	0.601	0.732	0.766	0.847
Synth. (Unpad.)	0.565	0.705	0.679	0.789

Table 5: AI Fallback and Drop Rates for Live Methods. Evaluates the frequency of AI segmentation failure requiring synthetic or smoothed fallbacks.

Method	Sequence	Total	Fallback Frames		Drop Rate	
			Synth.	Smooth	Full	Smoothed
LiveMCR	1 (Dynamic)	500	89	24	22.60%	17.80%
LiveVDA	1 (Dynamic)	500	89	25	22.80%	17.80%
LiveMCR	2 (Stable)	500	142	43	37.00%	28.40%
LiveVDA	2 (Stable)	500	142	43	37.00%	28.40%

5.4 Phase 2: Depth Classification

Phase 2 constitutes the core occlusion decision: determining whether the **VOB** should be rendered in front of or behind the intersecting performer. This classification was calculated exclusively on correctly detected intersections (True Positives) from Phase 1 to isolate depth accuracy from segmentation failures. Table 6 shows the results of Phase 2.

5.5 Phase 3: Visual Fidelity

The final phase evaluates the visual fidelity of the composited occlusion. For all correctly ordered depth interactions from Phase 2, the final rendered mask was compared against the offline SAM3 ground truth. Because MR occlusions are highly sensitive to boundary artifacts, performance was measured using both Standard IoU and Boundary IoU ($d = 10$ pixels). Table 7 shows the results of Phase 3.

Table 6: Phase 2 Depth Classification Accuracy. Evaluates the correct Z-buffer ordering between the **VOB** and performers.

Method	Seq 1 (Dynamic)	Seq 2 (Stable)
Offline + MCR	94.96%	**99.43%**
Offline + VDA	90.17%	96.46%
LiveMCR	**99.69%**	99.21%
LiveVDA	80.43%	68.52%
Synth. (Pad.)	98.40%	99.08%
Synth. (Unpad.)	98.72%	98.96%

Table 7: Phase 3 Visual Fidelity. Standard IoU measures general area overlap, while Boundary IoU ($d = 10$) measures precision specifically at the mask edges.

Method	Sequence 1 (Dynamic)		Sequence 2 (Stable)	
	Std. IoU	Bnd. IoU	Std. IoU	Bnd. IoU
Offline (Baseline)	1.000	1.000	1.000	1.000
LiveMCR	**0.712**	0.298	**0.662**	**0.404**
LiveVDA	0.706	**0.318**	0.636	0.391
Synth. (Pad.)	0.609	0.268	0.522	0.360
Synth. (Unpad.)	0.524	0.200	0.454	0.286

5.6 Pipeline Execution Latency

The final evaluation metric is the end-to-end execution latency of the integrated Unity MR pipeline. As defined in Section 3.10, the temporal threshold for 30 FPS real-time processing is $\tau_{30} \approx 33.3$ ms. Runtime was calculated as the median per-frame latency, explicitly excluding read/write I/O overhead to simulate a live broadcast stream.

As detailed in Table 8, the baseline offline method (*Offline + MCR*) required a median net latency of just 9.26 ms per frame on Sequence 2. The primary real-time configuration, *LiveMCR*, achieved a median net latency of 34.96 ms (Seq 1) and 35.04 ms (Seq 2), remaining highly consistent regardless of camera motion. However, with the AI execution step accounting for over 27.7 ms of that total, it narrowly missed the τ_{30} ceiling. The *SyntheticPadded* fallback method recorded a slightly slower latency than OfflineMCR because of the processing required for the Mask Processing step.

6 Discussion

6.1 Segmentation Experiment

Initial experiments evaluated the impact of image pre-processing on real-time inference. The pre-processing consisted of limiting frame search space through using the spatial data of the reconstructed performers as reference to derive a region-of-interest bounding-box, as well as gamma correction. The findings indicated an architectural divide: pre-processed inputs improved the mIoU for the CNN-based YOLO11 models, but degraded performance for the

(a) Live signal showing 5 masks in total with one being a combined mask of two performers, leading to incorrect occlusion-handling in this region

(b) The video frame for which segmentation was inferred, note that in this scenario the fallback mechanism would commence as only 5 masks out of 6 visible performers are found

Figure 12: The combined mask case illustrated.

Table 8: Median Pipeline Latency. Values represent the net execution time per frame in milliseconds, isolating AI and Mask Processing overhead. The remaining latency difference Net - (AI + Mask) is simply the video runtime and synchronizing overhead latency within the Unity environment ($\sim$ 2-4 ms).

Method	Seq 1 Net	Seq 2 Net	AI (ms)	Mask (ms)
Offline + MCR*	10.96	9.26	0.00	7.34
LiveMCR	34.96	35.04	27.78	4.14
LiveVDA	36.79	37.27	28.02	6.63
Synthetic*	11.88	13.63	0.00	11.56

*Offline and Synthetic methods skip live AI inference.

transformer-based RF-DETR architecture.

When evaluating the pipeline requirements for MR compositing, execution speed and mask fidelity must be balanced. While YOLO11-X achieved high accuracy, its average inference time of 17.52 ms exceeded the τ_{60} ceiling required for 60 FPS output. Consequently, it was excluded from consideration for the final MR pipeline. Ultimately, the *RF-DETR-M (Conf=0.35, No Pre-processing)* configuration was selected as the optimal baseline engine. It provided the most performant inference latency (8.94 ms) while maintaining a competitive mIoU score. Meanwhile, the *RF-DETR-L (Conf=0.35, No Pre-processing)* was significantly more precise, thus an optimized version of this pipeline might benefit from this configuration as an alternative if latency was less of an issue.

Thus for our research question regarding CNN-based versus transformer-based deep learning architectures for instance segmentation in the context of multi-person broadcast footage, we found that RF-DETR, both the M and L models, were preferable to the CNN-based YOLOv11 models tested. A larger set of models would have to be tested to generalize this finding, however, but the deformable cross-attention feature of RF-DETR is indeed meant to improve spatial context awareness across features. In our specific experiments —which included complex occlusion scenarios as often is expected in multi-person footage— our results supported this claim.

Ultimately, we also found that our strongest real-time instance segmentation model managed a 0.731 mIoU score when compared with the ground-truth high-precision model SAM3, meaning a significant downgrade but still competitive. It certainly cannot be relied on as a ground-truth on its own, but with its very strong inference time (12.45 ms on average) this is an acceptable trade-off for the reduced latency.

6.2 Detecting Occlusions

Because the offline baselines utilize the exact human-verified ground truth masks used to define the occlusion events, they serve as a perfect theoretical baseline (F1-Score = 1.000). As shown in Table 4, the dynamic camera motion of Sequence 1 visibly impacted the detection stage across all real-time methods.

The F1-score serves as a good indicator of whether we detect the existing occlusions present in a frame. Interestingly, the F1-score increases from the more dynamic Sequence 1 to the more stable Sequence 2, despite an increased drop rate. This is due to the nature of the dropped frames; where in Sequence 1 the movement conditions of the scene cause missing intersections particularly during zoomed frames, Sequence 2 is less impacted by this phenomenon and therefore the dropped frames have less impact on the overall detection system.

The reason for the increased drop rate is again a scene characteristic, but a different one. For Sequence 2 there are nearly twice as many occlusions as in Sequence 1, leading to the segmentation model often not being able to produce coherent masks per performer, leading to combined masks of multiple performers or other abnormalities. There are also more cases of multi-occlusion, where more than two performers occlude each other, which is a particularly

complex occlusion scenario. Therefore, this comparison between the two sequences provide promising results, as the scene with more occlusions and higher drop rate ultimately detects more correct occlusions.

Simultaneously, it highlights a deficiency in the system in particular when dealing with camera zooming. This is further elaborated upon in Section 7.

6.3 Depth System

The results, presented in Table 6, conclusively validate the use of the MCR-based depth over monocular VDA depth inference, and highlight the resilience of the MCR Z-buffer to camera motion. The offline baseline (*Offline + MCR*) achieved near-perfect accuracy (99.43%) on Sequence 2, with a slight drop to 94.96% during the aggressive movement of Sequence 1 due to minor spatial misalignments between the real camera and the reconstructed mesh.

Interestingly, *LiveMCR* maintained a flawless depth accuracy of 99.69% on Sequence 1, marginally outperforming the ground-truth SAM3 masks.

This is likely because of *LiveMCR*'s performance in Phase 1, meaning undetected occlusions were challenging (as is expected) and since these are all detected for the ground-truth masks, a few frames might have had misidentification events after the projection mapping step (which is also applied to the offline method), leading to poor occlusion for the affected mask regions.

Additionally, slight mask differences may lead to extremities not intersecting when expected which admittedly might have too much of a weight on the result, since if the depth classification order between two objects is wrong but the thinner/misplaced mask does not intersect with its finger, the misclassification is not detected.

This is proven through a significant drop in recall (0.767 from baseline 1.000) from Phase 1 for *LiveMCR*, signaling the inability to detect many occlusion events.

Conversely, the monocular depth estimation model, VDA, struggled significantly. While it performed reasonably well with offline masks on the stable sequence (96.46%), its accuracy fell to 68.52% when deployed with live masks (*LiveVDA*). Even on Sequence 1, where *LiveVDA* improved to 80.43%, an 80% success rate still means that the **VOB** is flickering between the foreground and background several times per second. This is essentially due to the sampling process described in Section 3.7 which is reliant on a consistent and reliable mask region to sample its depth values from. This works well for the offline masks, but falls significantly short with the more unstable live segmentation masks.

This confirms that our chosen explicit depth estimation model cannot reliably classify occlusion scenarios to the level required for broadcast-quality MR occlusion when paired with live segmentation. However, the MCR-based system performed near-perfect, and is not reliant on depth sampling that the monocular depth estimation method requires.

6.4 Visual Fidelity

As outlined in Table 7, *LiveMCR* provided the highest visual fidelity among the real-time capable methods. On the stable sequence (Seq 2), it achieved a Standard IoU of 0.662 and a Boundary IoU of 0.404. However, the data reveals a crucial insight when transitioning to the dynamic sequence (Seq 1): while the Standard IoU for *LiveMCR* actually increased to 0.712, its Boundary IoU plummeted to 0.298. This discrepancy proves that dynamic camera motion and motion blur do not confuse the model about *where* the performer is (the mass area remains accurate), but severely degrades the exact pixel-edges of their clothing, which is critical for preventing MR clipping artifacts.

Crucially, while the synthetic meshes proved highly reliable for depth ordering (Phase 2), their visual fidelity was substantially lower than the segmented masks. *SyntheticPadded* achieved a Standard IoU of only 0.522 on Sequence 2, confirming that while padding helps detect intersections, the rigid geometric boundaries of the skeletal mesh cannot emulate the pixel-perfect contours of human clothing as effectively as live segmentation.

6.5 Research Finding

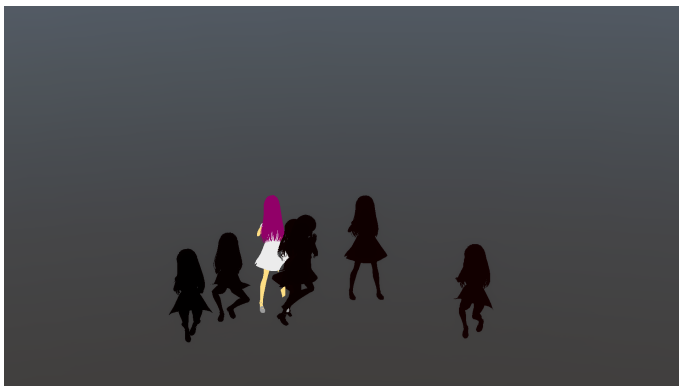
As elaborated upon in Section 6.1, the RF-DETR models outperformed the YOLOv11 models, and had a very respectable mIoU score considering the low latency when compared to the slower offline model - although these offline inference times were not explicitly measured.

Furthermore we found in Section 6.3 that the live segmentation masks revealed volatility in the depth sampling method, leading to significantly worse depth classification for this combination.

Finally, to answer our main research question, the combination of MCR and real-time instance segmentation could indeed produce accurate explicit depth classification, but fell short of the τ_{30} goal by the narrowest of margins in terms of run-time, thus by our own definition not performing in real time. Furthermore, we were not able to produce consistently precise occlusion compositing —although we did not set any explicit mIoU goal— and this presents one of our main targets for future improvement.

6.6 Padding Ablation

The synthetic fallback methods demonstrated the necessity of spatial padding. The unpadded synthetic mesh missed over 32% of all actual occlusion events in Sequence 2 due to the skeletal boundaries being too thin compared to the performers' physical clothing. Applying a spatial padding bias (*SyntheticPadded*) significantly recovered these missed detections in both sequences.



(a) Synthetic masks of the reconstructed meshes clearly occlude the **VOB**

(b) However, the dissimilarity between the mesh and the real performer causes an imperfect occlusion

Figure 13: Synthetic masks and their fidelity flaws illustrated, required in cases of combined or missing masks.

7 Limitations

It is worth noting that our pipeline does not explicitly model lighting. In the context of this work, occlusion compositing is concerned with correct depth ordering at the pixel level rather than photorealistic rendering, so lighting consistency between virtual and real elements is treated as out of scope. This is a known limitation: in a production setting, mismatched lighting between inserted **VOBs** and the real scene would be perceptually noticeable, and addressing it would require additional computation and implementation which are not prioritized for this project.

Furthermore, another limitation is that our pipeline does not explicitly handle multiple **VOBs**, and consequently we also did not evaluate the latency performance for this. However, the necessary data to handle multiple **VOBs** is definitely present and it is likely possible to retain approximately the same latency, but is unlikely to scale well with a large number of **VOBs**, both in terms of latency and occlusion quality (though of course the **VOBs** in isolation would occlude each other perfectly well).

There are some spatial and temporal artifacts. Spatial artifacts mostly refer to the existence of an offset between $\mathbf{tripod}_{real}$ and $\mathbf{tripod}_{virtual}$, i.e. the virtual tripod is not placed in a position so that its perspective perfectly matches that of its real counterpart. Finding this offset is now mostly handled through iterative and manual tweaking. There is also a processing-related temporal offset between the movement of the performers in the virtual representation and the real performers as described in Section 4.3. This is problematic since we need to perform instance segmentation in real time and if the temporal offset cannot also be corrected live, then the reconstruction will have a discrepancy in its depth information as well as its synthetic masks.

The issue of misidentification as described in Section 3 is a recurring issue through the project pipeline. In our reconstruction, all our reconstructed performers are always perfectly present and available, and we always have their depth, but during segmentation this is not always the case. Objects may have their segmentation mask missing for one or multiple frames, and because we are still expecting those masks for their corresponding reconstructed performers, pixels in the region of those missing masks will be occluded incorrectly, since depth information is then temporarily missing.

Even hypothetically with a real-time temporal tracking model that also segments the object's mask, misidentification can still occur during imperfectly segmented frames. An example is when two real dancers are segmented as one mask (Figure 12), which would mean that the segmentation model believes that one object from the previous frame disappeared, although it still clearly is visible in the scene. This is particularly relevant because we are focusing specifically on occlusion, where such scenarios happen frequently as objects are closely positioned next to and around each other.

Another issue lies with the offline monocular depth estimation described in Section 3.7. When this is done together with the offline ground-truth masks expected to align with the same regions within the produced depth map, it works as expected. However, when utilizing this depth estimation together with the live segmentation masks, this assumption often breaks and the sampling quality suffers. Therefore, the results accomplished with the LiveVDA method is not a particularly reliable indicator as to the quality of the produced depth map itself, however it gives a good picture of its actual performance as a depth system in our real-time pipeline as we implemented it.

8 Future Work

In order to account for the main bottlenecks of the pipeline, there are several options. One solution would be to separate the computations, as both the reconstruction and the segmentation step on their own run at a reasonable speed, but combined they struggle. Thus a system pipeline with segregated hardware could handle either one of the computations and stream the relevant data. For instance, this

could be materialized through streaming the segmentation masks to the separate computer, allowing it to handle occlusion and keeping the entire system real-time. Assuming AI inference latency segregated from the occlusion-pipeline, we could then use an even larger and more precise model —in our case RF-DETR-L immediately comes to mind— to perform the inference as the computations would be in parallel. Furthermore, building a custom model architecture or doing model distillation for the particular target of segmenting music or sports performers within concert venues or sport arenas would be another promising avenue to explore in order to improve the segmentation signal output.

A more fundamental improvement to the pipeline would be the replacement of the frame-wise segmentation model with an integrated video segmentation architecture. The projection mapping step, and its associated misidentification failure cases, exists precisely because our segmentation model lacks temporal identity persistence. Pairing a real-time instance segmentation model with a dedicated tracker, such as through Roboflow's tracker module [32], represents one practical path toward this. A more architecturally elegant solution is offered by recent work such as VidEoMT [33], which unifies segmentation and temporal association within a single ViT encoder through a lightweight query propagation mechanism, eliminating the need for specialized tracking components entirely while achieving real-time performance. Integrating such an approach would allow identity persistence to emerge natively from the segmentation model, potentially making the projection mapping step redundant and significantly reducing the misidentification failure rate.

Another exciting possibility is to utilize the available data of the reconstructed meshes within the Unity pipeline to extrapolate an approximate mask during dropped frames. This could be through shifting the centroid of the live mask with respect to the synthetic mask, as a simple example. This was attempted in our project but found to be computationally unfeasible, but with more time perhaps a clever solution can be found. On the other side, an equally exciting possibility is to let the reconstructed meshes learn from the segmentation masks. This could be realized by initially building a more sophisticated mesh, such as with SMPL-X [9], based on one or more masks (if a time delay is allowed, this can be through a powerful model like SAMv3), which then is animated as usual but more closely mimics the real performers. This latter example would then likely improve the visual fidelity of the synthetic masks during dropped frames, possibly improving occlusion quality.

9 Conclusion

This work has demonstrated the feasibility of combining MC reconstruction and real-time instance segmentation for occlusion-aware MR compositing in broadcast footage. The core finding is that MCR-derived explicit depth is substantially more reliable than monocular depth estimation for this task, achieving near-perfect depth classification accuracy that monocular approaches could not approach under live inference conditions. The pipeline's primary limitation is runtime: at $\sim$35 ms median latency it narrowly misses the 30 FPS real-time threshold, though the individual components are each capable of meeting it in isolation, suggesting that architectural separation across hardware would resolve this.

The segmentation experiments established RF-DETR as the superior candidate for real-time instance segmentation in this domain, confirming that transformer-based architectures are viable for live broadcast inference and providing one of the first applied evaluations of RF-DETR's instance segmentation capability. The three-phase evaluation framework introduced in this work provides a reusable structure for assessing occlusion pipelines across detection, depth classification, and visual fidelity independently, which may be of value for future work in this area.

The main remaining bottlenecks include the projection mapping step, which introduces misidentification failures due to the lack of temporal identity persistence in the segmentation model, as well as the dissimilarity between real and reconstructed masks which leads to poor visual fidelity and qualitative occlusion-quality. Integrating a real-time tracking architecture such as VidEoMT represents the most promising path toward eliminating the former limitation, while the latter could be optimized through a more sophisticated utilization of the available data in both the real and virtual realms, in order to achieve a fully robust real-time MR occlusion system.

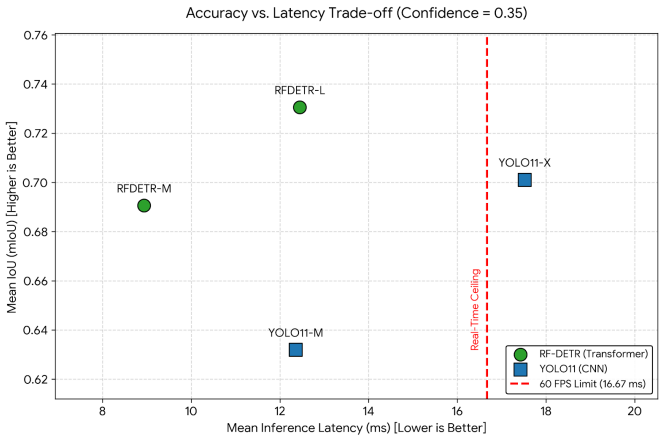
AI Disclaimer

This thesis project was conducted with the assistance of AI tools for brainstorming, code generation, LaTeX formatting, and grammar correction. All AI-generated outputs have been thoroughly reviewed and validated by the author, who assumes full responsibility for the final content and methodology presented in this work.

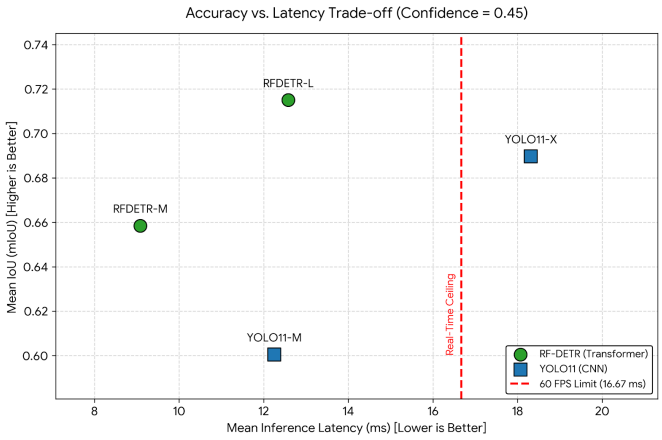
10 Appendix

Table 9: Software and Model Environment Specifications.

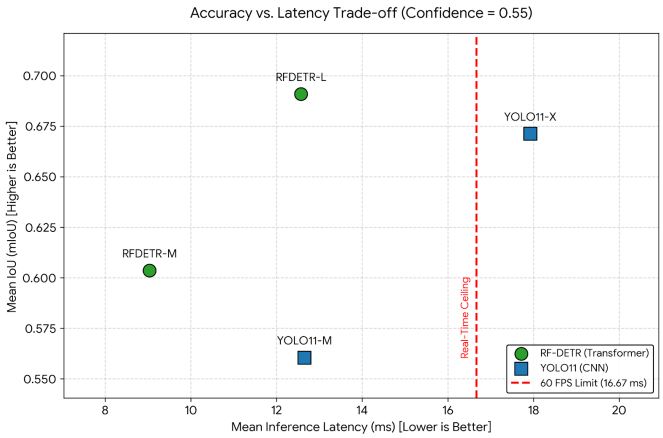
Component	**Specification**
Unity Version	6.000.51f1 (HDRP)
ONNX Runtime	1.23.2
RF-DETR Opset	16
YOLO11 Opset	12
Input Res. (RF-DETR-M)	432×432
Input Res. (RF-DETR-L)	504×504
Input Res. (YOLO11)	640×640
Batch Size	1
Execution Provider	TensorRT / CUDA



(a) Confidence Threshold = 0.35



(b) Confidence Threshold = 0.45



(c) Confidence Threshold = 0.55

Figure 14: Mean IoU (mIoU) versus Inference Latency across three confidence thresholds. Across all configurations, RF-DETR-L provides the optimal upper bound of spatial accuracy while operating strictly below the τ_{60} (60 FPS) threshold.

References

[1] Dipesh Gyawali. *Mixed Reality: The Interface of the Future*. 2023. arXiv: 2309.00819 [cs.HC]. URL: https://arxiv.org/abs/2309.00819.

[2] David Walton and Anthony Steed. "Accurate Real-Time Occlusion for Mixed Reality". In: Nov. 2017, pp. 1–10. DOI: 10.1145/3139131.3139153.

[3] Lars Mündermann, Stefano Corazza, and Thomas P. Andriacchi. "The Evolution of Methods for the Capture of Human Movement Leading to Markerless Motion Capture for Biomechanical Applications". In: *Journal of NeuroEngineering and Rehabilitation* 3.6 (2006). DOI: 10.1186/1743-0003-3-6.

[4] Linjie Yang, Yuchen Fan, and Ning Xu. *Video Instance Segmentation*. 2019. arXiv: 1905.04804 [cs.CV]. URL: https://arxiv.org/abs/1905.04804.

[5] Laurie Needham et al. "The development and evaluation of a fully automated markerless motion capture workflow". In: *Journal of Biomechanics* 144 (2022), p. 111338. ISSN: 0021-9290. DOI: https://doi.org/10.1016/j.jbiomech.2022.111338. URL: https://www.sciencedirect.com/science/article/pii/S0021929022003797.

[6] Elizabeth L. Brainerd et al. "X-ray reconstruction of moving morphology (XROMM): precision, accuracy and applications in comparative biomechanics research". In: *Journal of Experimental Zoology Part A: Ecological Genetics and Physiology* 313.5 (2010), pp. 262–279. DOI: 10.1002/jez.589.

[7] Urbano Lugrís et al. "Human motion capture, reconstruction, and musculoskeletal analysis in real time". In: *Multibody System Dynamics* 60 (2024), pp. 3–25. DOI: 10.1007/s11044-023-09938-0.

[8] Matthew Loper et al. "SMPL: A Skinned Multi-Person Linear Model". In: *ACM Transactions on Graphics* 34.6 (2015), 248:1–248:16. DOI: 10.1145/2816795.2818013.

[9] Georgios Pavlakos et al. "Expressive Body Capture: 3D Hands, Face, and Body from a Single Image". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 10975–10985.

[10] Rahima Khanam and Muhammad Hussain. *YOLOv11: An Overview of the Key Architectural Enhancements*. 2024. arXiv: 2410.17725 [cs.CV]. URL: https://arxiv.org/abs/2410.17725.

[11] Ranjan Sapkota et al. *RF-DETR Object Detection vs YOLOv12 : A Study of Transformer-based and CNN-based Architectures for Single-Class and Multi-Class Greenfruit Detection in Complex Orchard Environments Under Label Ambiguity*. 2025. arXiv: 2504.13099 [cs.CV]. URL: https://arxiv.org/abs/2504.13099.

[12] Roboflow. *RF-DETR: Roboflow Detection Transformer*. https://github.com/roboflow/rf-detr. Accessed: 2025-11-08. 2024.

[13] Isaac Robinson et al. *RF-DETR: Neural Architecture Search for Real-Time Detection Transformers*. 2025. DOI: 10.48550/arXiv.2511.09554. arXiv: 2511.09554 [cs.CV]. URL: https://arxiv.org/abs/2511.09554.

[14] Nicolas Carion et al. *SAM 3: Segment Anything with Concepts*. 2025. arXiv: 2511.16719 [cs.CV]. URL: https://arxiv.org/abs/2511.16719.

[15] Lihe Yang et al. *Depth Anything V2*. 2024. arXiv: 2406.09414 [cs.CV]. URL: https://arxiv.org/abs/2406.09414.

[16] Márcio C. F. Macedo and Antônio L. Apolinário. "Occlusion Handling in Augmented Reality: Past, Present and Future". In: *IEEE Transactions on Visualization and Computer Graphics* 29.2 (2023), pp. 1590–1609. DOI: 10.1109/TVCG.2021.3117866. URL: https://pubmed.ncbi.nlm.nih.gov/34613916/.

[17] David E. Breen et al. "Interactive Occlusion and Automatic Object Placement for Augmented Reality". In: *Computer Graphics Forum* 15.3 (1996), pp. 11–22. DOI: 10.1111/1467-8659.1530011.

[18] Anton Fuhrmann et al. "Occlusion in Collaborative Augmented Environments". In: *Computers & Graphics* 23.6 (1999), pp. 809–819. DOI: 10.1016/S0097-8493(99)00107-7.

[19] Yuan Tian et al. "Handling occlusions in augmented reality based on 3D reconstruction method". In: *Neurocomputing* 156 (2015), pp. 96–104. DOI: 10.1016/j.neucom.2015.01.006.

[20] Aleksander Holynski and Johannes Kopf. "Fast Depth Densification for Occlusion-aware Augmented Reality". In: *ACM Transactions on Graphics (TOG)* 37.6 (2018), 194:1–194:11. DOI: 10.1145/3272127.3275083.

[21] Jamie Watson et al. *Virtual Occlusions Through Implicit Depth*. 2023. arXiv: 2305.07014 [cs.CV]. URL: https://arxiv.org/abs/2305.07014.

[22] Jeff Lander. *BVH File Specification, Character Studio Motion Capture File Format*. https://research.cs.wisc.edu/graphics/Courses/cs-838-1999/Jeff/BVH.html. Accessed: 2025-11-12. 1998.

[23] Simplified Solutions Inc. *IDF v3.0 Specification*. http://www.simplifiedsolutionsinc.com/images/idf_v30_spec.pdf. Accessed: 2025-11-12. 2002.

[24] Nikhila Ravi et al. *SAM 2: Segment Anything in Images and Videos*. 2024. arXiv: 2408.00714 [cs.CV]. URL: https://arxiv.org/abs/2408.00714.

[25] OpenCV. *Camera Calibration with OpenCV*. URL: https://docs.opencv.org/4.x/dc/dbb/tutorial_py_calibration.html (visited on 10/28/2025).

[26] Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing*. 4th. Pearson, 2018.

[27] Yutao Liu et al. "Frame Rate and Perceptual Quality for HD Video". In: Sept. 2015, pp. 497–505. ISBN: 978-3-319-24077-0. DOI: 10.1007/978-3-319-24078-7_50.

[28] Microsoft. *Understanding Performance for Mixed Reality*. https://learn.microsoft.com/en-us/windows/mixed-reality/develop/advanced-concepts/understanding-performance-for-mixed-reality. Accessed: 2025-11-03. 2024.

[29] Taehee Lee and Tobias Höllerer. "Hybrid Feature Tracking and User Interaction for Markerless Augmented Reality". In: *Proceedings of the IEEE Virtual Reality Conference (VR)*. IEEE. 2008, pp. 145–152.

[30] Daniel Wagner et al. "Real-Time Detection and Tracking for Augmented Reality on Mobile Phones". In: *IEEE Transactions on Visualization and Computer Graphics* 16.3 (2010), pp. 355–368. DOI: 10.1109/TVCG.2009.81.

[31] Bowen Cheng et al. "Boundary IoU: Improving Object-Centric Image Segmentation Evaluation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021, pp. 15334–15342.

[32] Roboflow. *Trackers: Modular Multi-Object Tracking Algorithms*. https://github.com/roboflow/trackers. Apache 2.0 License. 2025.

[33] Narges Norouzi et al. *VidEoMT: Your ViT is Secretly Also a Video Segmentation Model*. 2026. arXiv: 2602.17807 [cs.CV]. URL: https://arxiv.org/abs/2602.17807.