FRAME-BY-FRAME OBJECT RECOGNITION AND STORAGE FOR VIDEO ANALYTICS

Carl Zhang, Eric Jia, Freya Zhang, Jiamin Wang, Ruoqi Yang, Xinshu Zhao
Team Object-Oriented Pathfinders
Data Science 435/535 Computer Science 449/549
{cz81, ejj3, flz1, jw233, ry30, xz120}@rice.edu

Partha Dutta (Sponsor)
TechnipFMC
partha.dutta@technipfmc.com

Jonathan Garcia (Sponsor)
TechnipFMC
jonathan.garcia@technipfmc.com

Jordan Clark (Sponsor)
TechnipFMC
jordan.clark@technipfmc.com

Arko Barman (Professor)
Rice University
arko.barman@rice.edu

Nhi Le (PhD Mentor)
Rice University
nyl1@rice.edu

CONTENTS

1 Introduction

1.1 Background

TechnipFMC is a leading technology provider to the traditional and new energies industry, delivering fully integrated projects, products, and services across subsea and surface operations. With proprietary technologies and comprehensive solutions, the company helps clients unlock new possibilities to develop energy resources more efficiently while supporting the global transition toward lower-carbon operations. A critical part of these industrial activities involves continuous video monitoring to ensure safety and operational integrity.

For instance, a TechnipFMC center in Houston that handles refurbishments may have dozens of engineers working on a large number of multimillion dollar equipment at a given point in time. A key part of employee safety and awareness involves personal protective equipment, such as helmets, goggles, and steel-toed boots. To ensure a healthy working environment, TechnipFMC monitors their work stations with security cameras and verifies compliance with safety protocols, especially involving heavy and dangerous machinery. Doing so means efficiently identifying workers and objects within the environment.

However, much of this monitoring still depends on manual human review, which is time-consuming, inconsistent, and difficult to scale. As the volume of visual data increases, there is a growing need for automated systems that can detect and record key objects and events in real time. Such automation would improve situational awareness, reduce human workload, and enhance both safety and operational decision-making.

Object detection is a key computer vision task that identifies objects within images or video frames. Unlike image classification, which applies one label to an entire image, object detection outputs both class labels and bounding boxes, showing what objects are present and where they appear (Zhao et al., 2019). This capability is the basis for many real-world applications, such as autonomous driving and medical imaging.

This project focuses on developing a data-driven pipeline that integrates object detection with structured data management to support automated video monitoring. The goal is to design a system capable of processing raw video footage, identifying objects of interest, and storing the detection results in a database that enables users to retrieve and review corresponding video segments efficiently. As such, a user can send their question as a query to the database system and view a collection of clips that match the query as a result. For instance, a reviewer may ask ”when does the hammer appear?” and pass this request to the system. The system will then communicate the video segments involving hammers to the user.

The solution integrates two complementary components: a computer vision task that focuses on accurately detecting and localizing objects in video frames, and a software engineering task that designs the database and retrieval mechanisms to manage large-scale detection outputs. Together, these components lay the foundation for a scalable monitoring framework that can strengthen safety oversight, streamline operations, and advance TechnipFMC's broader goals of digital transformation and operational excellence.

1.2 Objectives

The objective of this project is to design and implement a proof-of-concept system that automates object detection and data retrieval within video monitoring workflows. The system aims to replace manual video review with a structured, data-driven process that captures object occurrences and allows direct access to relevant video segments through a queryable database.

This project encompasses two primary goals:

- **Object Detection:** Develop a pipeline capable of identifying and localizing objects within video frames while recording their categories, confidence scores, and timestamps. This involves identifying models with a balance of accuracy and speed, fine-tuning select pretrained models, and evaluating their performance. Due to privacy restrictions on TechnipFMC's proprietary footage, the pipeline is trained and evaluated using publicly available datasets for proof-of-concept validation.

- **Database and Retrieval System:** Design a structured database to store detection results efficiently and implement a retrieval mechanism that supports flexible querying. The system enables users to search for specific objects and directly link query results back to the relevant video frames or playable segments. Thus, the user workflow will be simple: input the target video name and optionally an object name, and the user will see the video with a collection of timestamps that involve all identified objects or only the provided object of interest. This integration demonstrates the feasibility of a scalable end-to-end framework that can later be adapted for deployment in TechnipFMC's internal systems.

Our final deliverables to TechnipFMC will be two items, each corresponding to one of the aforementioned goals. First, we will provide a model training suite that contains sample models trained on the public dataset and a workflow that allows TechnipFMC engineers to seamlessly replace our proof-of-concept training data with their proprietary data to train models for their work environment. Second, we will provide a software system that includes a backend database and a frontend user interface with the following features:

1. Integrates an object detection model
2. Allows a video to be uploaded and evaluated with the model, with results stored in the database
3. Allows users to send queries with a video identifier and object name and view relevant video clips

1.3 Project Impact

The proposed system has the potential to significantly enhance the efficiency, reliability, and safety of TechnipFMC's industrial monitoring operations. By transitioning from manual inspection to automated object-based retrieval, the project demonstrates how intelligent data management can strengthen situational awareness and operational decision-making. The system is expected to generate the following impacts:

- **Operational efficiency:** Automating object detection and retrieval is expected to significantly reduce the manual workload associated with reviewing video footage. By enabling fast, targeted searches through a structured database, the system allows personnel to concentrate on higher-level tasks such as incident analysis and decision-making.
- **Enhanced safety monitoring:** The system improves response times by recording detected objects and providing direct access to their corresponding video segments. This capability allows supervisors to quickly identify abnormal or unsafe activity without having to proceed through an onerous manual process, accelerating review processes and improving employee morale.
- **Data accessibility and organizational insight:** Beyond simple retrieval, the centralized database provides a searchable historical record of detections that can be analyzed for trends and recurring workplace hazards. This supports long-term safety audits, staff training, and operational optimization, which will foster a culture of transparency and continuous improvement.

2 Literature Review

2.1 Object Detection

Object Detection is a crucial core computer vision task that locates and classifies all instances of objects in a class within images or video frames. Model output for each object instance typically includes a box surrounding the instance (bounding box), a class label, and a confidence score. The most significant metrics are accuracy, both classification and localization accuracy, as well as speed. Object detection serves as the foundation for numerous other computer vision tasks, including image captioning and object tracking. These have extensive real-world applications, ranging from robot vision to video surveillance (Zou et al., 2023). As such, object detection has garnered attention and undergone rapid evolution, as evidenced by the increasing number of publications mentioning ”object detection” over the past two decades.

Key milestones in object detection following the introduction of deep learning-based methods include the development of one-stage and two-stage detectors. Two-stage detectors began with CNN Regions with CNN features (RCNN), which was introduced in 2014. While these models had high precision, they were not often employed due to poor speed and complexity. On the other hand, one-stage detectors are able to retrieve all objects in one-step inference and are well liked for their speed, but usually suffer when detecting small objects. You Only Look Once (YOLO) was introduced in 2015 as the first one-stage detector in the deep learning era (Redmon et al., 2016). The most recent developments are found in transformers, which do not use the traditional convolution operator. In 2020, DETR was introduced, which enables the detection of objects without anchor points or boxes, as it views object detection as a set prediction problem.

Looking at video surveillance, object detection is essential as it monitors the movement of objects, a prerequisite for event tracking and target tracking. The introduction of deep learning techniques has improved the efficiency and accuracy of security systems, as the drawbacks of conventional algorithms include limited data size and high time complexity (Rao & Kumar, 2025). YOLO has been employed for crowd density analysis, where it identified and tracked individuals in real-time video streams. The authors found YOLO's real-time processing capabilities and high accuracy to be useful for crowd management and emergency planning. A study conducted by Abbas et al. presented a real-time framework for object detection and tracking for security surveillance systems. Their final approach involved a YOLOv4 network, stating ”this allows the technique to detect individuals in an indoor environment with greater precision and a reasonable inference time” (Abba et al., 2024). However, due to its computational intensity, YOLO's inference speed is slower than other object detection algorithms. This can be a critical problem, as quick processing is often necessary in video surveillance and security contexts.

Convolutional Neural Networks (CNNs) have also been widely adopted, due to their ability to learn intricate features from input data. The CNN is especially suited for cluttered and busy settings, as it can handle rotation and scale variations, which are common in videos where objects can move drastically. The main drawback is that CNNs require large amounts of labeled data and can be computationally expensive, similar to YOLO (Rao & Kumar, 2025).

Baed on prior studies and research, we will focus on the following three approaches: region-based detectors, YOLO, and transformer-based detectors.

2.2 The Era of Region-Based Detector

Region-based object detectors are a class of models designed to first generate candidate regions where objects might appear and then classify and refine these regions to detect objects accurately. This paradigm began with R-CNN (Girshick et al., 2014), which combined Selective Search (Uijlings et al., 2013) region proposals with CNN feature extraction and SVM classification, followed by bounding-box regression for improved localization. While highly accurate, R-CNN was computationally slow due to redundant per-region processing. Fast R-CNN addressed this by sharing convolutional feature maps across all proposals and introducing (RegionofInterest) RoI pooling, significantly improving efficiency. Faster R-CNN (Ren et al., 2015) further advanced the paradigm with the Region Proposal Network (RPN), enabling nearly cost-free, end-to-end proposal generation and achieving near real-time performance. The aim of region-based detectors is to achieve precise object localization and classification, particularly for small or complex objects, by leveraging focused region proposals. Their advantages include high accuracy and flexibility for related tasks, such as instance segmentation, while drawbacks include higher computational cost and added complexity compared with single-stage detectors. Key studies in this line of research include R-CNN, Fast R-CNN, and Faster R-CNN, which collectively illustrate the evolution of region-based approaches and their ongoing impact on modern object detection research.

2.3 The YOLO Paradigm: Unified Real-Time Detection

The YOLO framework (Redmon et al., 2016) revolutionized object detection by reformulating it as a single-stage regression problem, predicting bounding boxes and class probabilities directly in a single forward pass. Unlike two-stage detectors that first generate region proposals, YOLO divides an image into a grid, enabling real-time performance while maintaining strong accuracy. Early versions, inspired by GoogLeNet (Szegedy et al., 2014), progressively evolved by integrating new architectural and training improvements. YOLOv2 introduced anchor boxes, k-means optimization of anchor dimensions, batch normalization, and multi-scale training to improve localization, stability, and robustness. YOLOv4 enhanced multi-scale context understanding via spatial pyramid pooling and path aggregation networks, along with an efficient backbone and advanced training strategies. Later iterations, including YOLOv7, refined gradient propagation, feature aggregation, and label assignment for denser supervision (Wang et al., 2022). YOLOv10–v12 incorporate architectural innovations inspired by Transformers, enhanced attention mechanisms, and memory-efficient computation (e.g., R-ELAN, FlashAttention), improving small-object detection, spatial reasoning, and speed (Wang et al., 2024a). YOLO's paradigm aims to unify detection in a single pass, trading minimal accuracy loss for high efficiency. Its main advantages are real-time performance and simplicity, while limitations include potential struggles with very small or overlapping objects without advanced attention mechanisms. The YOLO family represents a continuous evolution in single-stage detection, with each version building upon prior work and taking inspiration from both CNN- and Transformer-based methods.

2.4 The End-to-End Revolution: Transformer-Based Detectors

Transformer-based object detectors adapt the self-attention mechanism, originally developed for natural language processing (NLP), to vision tasks. Self-attention allows the model to weigh relationships among all positions in an image simultaneously, supporting holistic scene reasoning rather than localized convolutional processing. DEtection TRansformer (DETR) (Carion et al., 2020) reframed detection as a set prediction problem, using a Transformer encoder-decoder to predict a fixed number of latent vectors called object queries, each representing a potential object. Training uses bipartite (Hungarian) matching to pair predictions with ground-truth objects. While elegant and fully end-to-end, DETR suffers from slow convergence and high computational cost due to dense self-attention. Deformable DETR (Zhu et al., 2020) addresses this by sparsely attending to a small set of key points around each reference location, improving efficiency and small-object performance. Real-time variants such as RT-DETR (Zhao et al., 2023) and RT-DETRv3 (Wang et al., 2024b) further accelerate convergence and improve training stability through hybrid encoders, selective query initialization, and hierarchical dense supervision. Most recently, RF-DETR (Robinson et al., 2025a) combines these innovations to achieve state-of-the-art real-time detection performance, surpassing 60 AP on MS COCO while being production-ready with optimized inference and deployment tools. Transformer-based detectors aim to balance accuracy and efficiency by capturing global context, with the main trade-offs being higher memory and compute requirements compared to traditional CNN-based detectors.

3 Data Description

3.1 Overview

In this project, we primarily use the YouTube-VIS 2022 dataset (Yang et al., 2022), a large-scale public benchmark designed for video instance segmentation and related video understanding tasks. The dataset is built from short, real-world YouTube video clips and focuses on objects appearing in their natural environments, captured under unconstrained conditions such as camera motion, occlusion, scale variation, and complex backgrounds.

YouTube-VIS 2022 covers 40 object categories, spanning people, animals, vehicles, and common everyday objects. Annotations are provided at the video level and performed by human annotators on a subset of frames sampled from each video. As a result, while the dataset is organized around videos, annotations are temporally sparse and are not available for every frame.

Annotations in YouTube-VIS 2022 are performed at the instance level across time. Each object instance is assigned:

- a semantic category label,
- a unique instance ID that remains consistent throughout the video,
- a per-frame segmentation mask, from which bounding boxes can be derived and used for object detection.

This annotation scheme enables both spatial object localization and temporal reasoning. We adopt YouTube-VIS 2022 as a public proxy dataset, as the sponsors' internal monitoring footage is private and cannot be shared.

3.2 Examples

Each sample in YouTube-VIS 2022 is organized at the video level, where a single video consists of a sequence of RGB frames and a set of annotated object instances that may appear, disappear, or undergo changes in pose and scale over time. At the frame level, an entry includes an image frame along with multiple object annotations, where each object is associated with a category label, an instance ID, and a spatial annotation in the form of a bounding box.

Although the original dataset provides segmentation masks, bounding boxes derived from these masks are used as the primary form of supervision in this project. By overlaying annotations onto video frames, we observe that objects are captured from a wide range of viewpoints and distances, often under motion blur or partial occlusion. Figure 1 illustrates a representative annotated frame from the dataset.

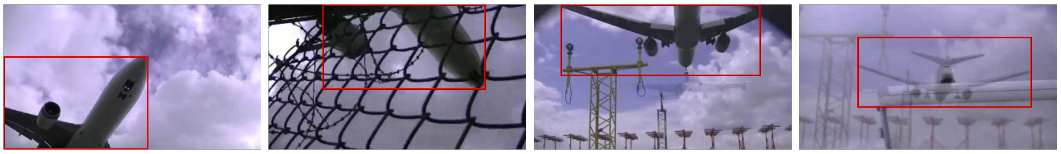

Figure 1: An example annotated images.

4 Data Exploration

This section explores the structure and characteristics of the YouTube-VIS 2022 dataset from the perspective of how the data is actually used in our project. The goal is to make the dataset understandable to readers without prior familiarity with video instance segmentation benchmarks.

Dataset contents. Although YouTube-VIS 2022 is released as a video dataset, annotations are not provided for every frame. Instead, the dataset consists of videos where only a subset of frames are labeled. In total, the raw dataset contains 4,096 videos, of which 3,477 videos include annotations. For each annotated video, labels are provided at fixed temporal intervals, with approximately one annotated frame every five frames. As a result, the effective data consists of temporally ordered images sampled from the original videos.

From a modeling perspective, the dataset therefore functions as a large collection of labeled images extracted from videos, where each image retains an implicit temporal relationship to neighboring frames. Each annotated frame may contain multiple objects, and the dataset spans 40 object categories, including people, transportation-related objects, animals, and sports equipment. The frequency of these categories varies substantially, motivating further analysis through visualizations such as category distributions and per-video annotation statistics.

Annotation format. Annotations in YouTube-VIS 2022 are stored at the video level and organized by object instance. Each annotation corresponds to a single object instance within a video and contains a list of per-frame entries. Specifically, an annotation specifies the video identifier, the object category, and a temporally ordered sequence of bounding boxes aligned with annotated frames.

Each element in the bounding box list represents the spatial extent of the object in a specific frame, encoded as $[x, y, width, height]$. Frames in which the object does not appear are represented by `null` entries. As a result, a single annotation implicitly encodes the temporal trajectory of an object across the video. Although annotations are stored in this video-centric format, the effective dataset used in this project consists of image frames with non-null bounding boxes, which are treated as individual labeled images during model training.

Strengths and shortcomings. The dataset exhibits several strengths that make it suitable for exploratory model development:

- The data is collected from real-world YouTube videos, capturing natural variation in viewpoint, motion, and background complexity.
- Annotated frames are nearly fully labeled, providing high-quality supervision for object detection.
- Temporal ordering of frames is preserved, allowing the dataset to retain contextual information from the original videos.

However, the exploratory analysis also reveals several limitations:

- **Class imbalance**: animal-related categories are overrepresented relative to other object types.
- **Category mismatch**: the object categories primarily reflect everyday scenes and wildlife, which do not align well with the sponsor's target application in industrial and production environments. Due to privacy constraints, such environments are not represented in the public dataset.
- **Temporal redundancy**: adjacent annotated frames sampled at fixed intervals are often visually similar, reducing the effective diversity of training images.
- **Incomplete annotation coverage**: several hundred videos contain no annotated frames and therefore cannot be directly used for supervised learning.

Addressing dataset limitations. These observations motivate a series of data preparation steps prior to model training. In particular, it is necessary to reduce category mismatch and imbalance, mitigate redundancy caused by temporally adjacent frames, and restructure the data into a format suitable for image-based learning while preserving video-level grouping. In this project, we treat the YouTube-VIS dataset as a public exploratory proxy, while TechnipFMC plans to continue this line of research using internal industrial video data in future work. The specific data processing strategies adopted in this study are described in Section 5.

5 Data Wrangling

This section describes the data wrangling pipeline used to transform the raw YouTube-VIS 2022 dataset into a training-ready dataset for image-based object detection. Each step directly addresses the limitations identified during data exploration.

Class selection and category restructuring. The original YouTube-VIS 2022 dataset contains 40 object categories spanning people, animals, vehicles, and sports-related objects. To better align the dataset with the project goals and reduce excessive class imbalance, we retain a subset of categories that are more relevant to monitoring and human–object interaction scenarios.

The following 19 semantic object classes are retained:

- person
- airplane
- boat
- car
- motorbike
- train
- skateboard
- snowboard
- surfboard
- tennis_racket
- dog
- rabbit
- bear
- tiger
- bird
- fish
- monkey
- snake
- frog

All annotations belonging to categories outside this list are removed. After filtering, category identifiers are reindexed to form a compact and contiguous label space. An implicit background category is introduced during reindexing to support object detector training.

Train, validation, and test split. To prevent information leakage, the dataset is split at the *video level* rather than the image level. All images originating from the same video are assigned to the same split. The processed dataset is divided into training, validation, and test sets using fixed proportions of 70%, 15%, and 15%, respectively, with a fixed random seed to ensure reproducibility.

Rather than using a naive random split, we adopt a category-aware video-level splitting strategy. For each video, the set of object categories appearing in its annotated frames is identified, and videos are assigned to splits in a way that preserves similar category presence across all subsets. This approach reduces the risk that rare categories appear only in a single split and yields more reliable evaluation results.

Annotation format transformation. The original YouTube-VIS annotations are video-centric, where each object instance is represented by a list of per-frame annotations across time. To support image-based object detection, the annotation format is transformed into a COCO-style, image-centric representation.

In the transformed dataset, each annotated frame is treated as an independent image. For every image, one or more object annotations are stored, each consisting of an image identifier, a category identifier, a bounding box encoded as $[x, y, width, height]$, and the corresponding bounding box area. Frames without any object annotations are discarded. Crowd annotations are not retained, and segmentation masks are not used in the transformed format. After splitting, image identifiers are unique within each dataset split.

Additional preprocessing steps. Several additional preprocessing steps are applied to improve data quality and balance. First, frames are uniformly subsampled within each video to reduce temporal redundancy caused by visually similar adjacent frames. Second, frames that contain only the *person* category are further downsampled on a per-video basis to mitigate residual class imbalance without removing entire videos.

Finally, the processed images and annotations are reorganized into a consistent directory structure. For each dataset split, a standalone COCO-style annotation file is generated, and all required images are copied into split-specific directories. In addition, the COCO-style annotations are converted into YOLO format to support training YOLO-based object detection models. This conversion preserves the same images, category definitions, and bounding boxes, while adapting the annotation representation to the input requirements of different model architectures. The resulting dataset is compact, balanced, and suitable for training and evaluating multiple object detection frameworks.

6 Modeling

6.1 Model Selection

For this project, we decided to move forward with two object detection paradigms into modeling:

1. YOLO
2. Transformer-based detector (RF-DETR)

These two models represent different strengths: YOLO prioritizes speed and deployment efficiency, while Transformer-based detectors emphasize global context, stability, and high performance in complex scenes.

We ultimately decided to move away from region-based detectors (R-CNN) because their latency and complexity do not align with TechnipFMC's real-time requirements, and comparable or higher accuracy can now be achieved with modern YOLO and Transformer models.

6.2 YOLO

The first YOLO model treated object detection as a single regression problem (Redmon et al., 2016). As such, an input image is divided into a grid, where each grid cell is responsible for predicting bounding boxes and confidence scores for any objects. Each grid cell also predicts conditional class probabilities.

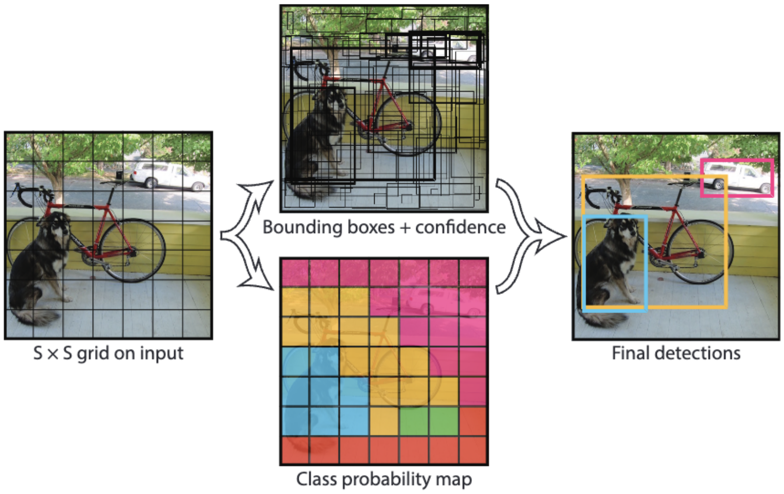

Figure 2: YOLO model breakdown from (Redmon et al., 2016)

This was implemented through a CNN, where the first convolutional layers extracted features, and the fully connected layers ”predicted output probabilities and coordinates” (Redmon et al., 2016). By performing detection in a single forward pass, YOLO achieved substantially faster inference than region-proposal-based methods.

Subsequent YOLO versions replaced fully connected layers with fully convolutional detection heads and introduced multi-scale feature aggregation to improve performance on objects of varying sizes (Bochkovskiy et al., 2020) These architectural refinements preserved the model’s real-time speed while improving localization accuracy and robustness.

During inference in our system, YOLO processes each video frame independently and produces frame-level detections that can be efficiently linked across time. Its low latency and high throughput make it well-suited for continuous monitoring and real-time alerting across multiple video streams.

The main limitation of YOLO is reduced robustness in visually complex scenes. Because predictions are made using predominantly local convolutional features and limited global context, YOLO can struggle with heavy occlusion, dense object layouts, or small objects at a distance (Carion et al., 2020). As a result, it may produce less stable detections across frames compared to Transformer-based detectors.

In TechnipFMC’s monitoring environments, where real-time responsiveness and scalability are critical, YOLO provides an efficient and practical detection backbone. While it may sacrifice some accuracy in complex scenes, its speed and reliability make it a strong baseline model and a useful complement to higher-capacity architectures such as RF-DETR.

6.3 RF-DETR

RF-DETR is a Transformer-based object detection model that builds on the DETR family introduced in Section 2.4 (Robinson et al., 2025b). It represents a newer generation of end-to-end detectors that can reason over the entire image at once and is designed to strike a balance between accuracy and practical runtime.

Internally, RF-DETR uses a pre-trained Vision Transformer (ViT) network to turn the input image into a set of visual features at different scales. On top of this backbone, the model alternates between layers that focus on local neighbourhoods (for fine details) and layers that can see the whole image at once (for global context). These features are then passed into a compact detection head that predicts bounding boxes, object categories, and confidence scores for a fixed number of potential objects.

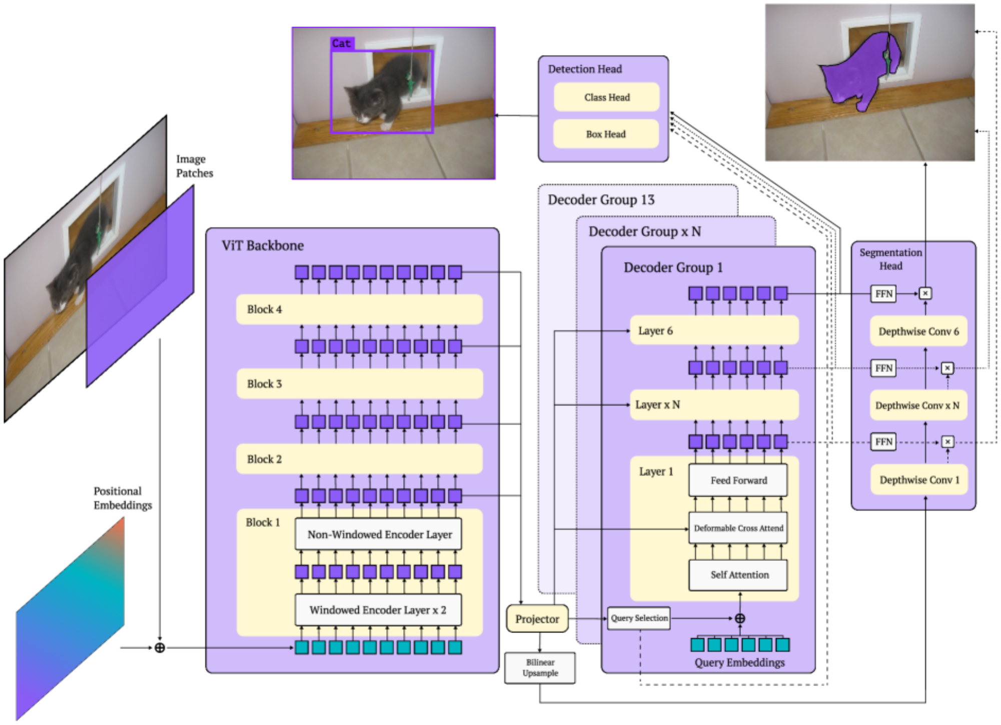

Figure 3: High-level architecture of RF-DETR, illustrating how image features are processed through Transformer layers and used to produce object detections (Robinson et al., 2025b).

The model is trained so that several intermediate stages can already make reasonable predictions. At deployment time, we can optionally shorten the network by skipping some of these later stages to reduce processing time, with only a modest impact on accuracy. During inference in our system, RF-DETR analyzes each video frame once and directly produces frame-level detections, without additional filtering steps. These detections are then grouped into continuous time segments.

Because RF-DETR uses global context, it is well-suited to scenes with multiple objects, cluttered backgrounds, or partial occlusions. It tends to produce stable and consistent detections across frames, which is important for safety review and auditing workflows.

The main trade-off is computational cost. Compared to lightweight models such as YOLO, RF-DETR generally requires more processing power and runs more slowly on constrained hardware. This makes it better suited for settings where detection quality is more important than maximum throughput.

TechnipFMC's monitoring environments often involve multiple workers, large equipment, and visually complex scenes. In these situations, missed or inconsistent detections could undermine safety reviews. RF-DETR's ability to leverage full-scene context makes it a strong candidate for these conditions and a useful complement to faster, lighter models in our overall system design.

7 Experiments

7.1 Model Tuning

The two models that we are considering are the RF-DETR and YOLO. Beginning with the medium variants of pre-trained RF-DETR and YOLOv12 (Tian et al., 2025) models, we performed fine-tuning using the training and validation data subsets described in section 5.

7.1.1 RF-DETR Strategy

For RF-DETR, we fine-tune the medium variant of the pre-trained RF-DETR model using the training and validation subsets described in Section 5. Our tuning strategy focuses on adjusting learning rates, learning rate scheduling, and the number of training epochs.

We initially experimented with multiple learning rate settings and found that decoupling the learning rates for different model components led to more stable training. In particular, setting a lower learning rate for the encoder (10^{-5}) and a higher learning rate for the detection head (10^{-4}) produced improved convergence and validation performance.

After establishing this learning rate configuration, we experimented with different learning rate schedulers, including step-based and cosine scheduling. Cosine scheduling resulted in smoother loss curves and more stable validation behavior, allowing us to extend the number of training epochs during fine-tuning without overfitting. All model components were fine-tuned jointly, without freezing the backbone. Based on observed training and validation loss trends, hyperparameters were refined iteratively to select stable and well-performing configurations.

7.1.2 YOLO Strategy

When training YOLO, our strategy focused on tuning the number of epochs, initial learning rate, and the learning rate factor. The learning rate factor is a configuration specific to YOLO's training mode, defined as $\frac{final\_learning\_rate}{initial\_learning\_rate}$. Other configurations are left default, for instance an image size of 640x640 and optimizer as AdamW (more configurations detailed in Appendix A.1).

To find the optimal hyperparameters, we initially attempted incrementing epochs by 20 in the range of 20 to 60, alternating the initial learning rate between 0.001 and 0.0001, and alternating the learning rate factor between 0.01 and 0.001. Because our goal is fine-tuning, the selection of learning rates is lower to prevent forgetting learned features and the number of epochs is not too high to prevent overfitting. After training a few model weights and analyzing the loss curves, we made finer adjustments to the training configurations.

7.2 Evaluation Metrics

To evaluate the performance of our models, we focus on the tradeoff between speed and accuracy and use the testing data subset described in Section 5. For speed, we measure the inference time, and for accuracy, we measure the mean average precision (mAP), precision, and recall.

The inference speed evaluation targets the Python implementation of model prediction because this reflects the method with which the retrieval system integrates the model. Thus, calculating inference time is direct and intuitive. Regardless of the model in question, the timer will begin upon invoking the model prediction task and will end upon receiving the resulting prediction.

The accuracy evaluation centers around mAP@0.5 and mAP@[0.5:0.95], which are metrics commonly used in the object detection space. mAP@0.5 is mean average precision calculated with bounding boxes marked as correct when the intersection-over-union (IoU) threshold is set to 0.5; mAP@[0.5:0.95] is the same but averaging IoU at 10 thresholds in the range of 0.5 to 0.95 (inclusive), in increments of 0.05. In other words, mAP@[0.5:0.95] is calculated by averaging the average precision values with the IoU threshold at 0.5, at 0.55, etc. The use of these accuracy metrics aims to achieve a score that is representative of detection of objects across different classes as well as the exactness of the bounding boxes.

In addition to mAP, we also report precision and recall to give a more interpretable view of model performance. Precision measures the fraction of predicted boxes that correspond to true objects, so high precision means the model produces few false alarms. Recall measures the fraction of true objects that the model successfully detects, so a high recall means the model misses fewer objects. We compute precision and recall at a fixed IoU threshold, consistent with the settings used for mAP@0.5.

For our sponsors, the model needs to be able to handle video processing quickly and with high accuracy. The project aims to streamline a manual review workflow, so a slow model may be frustrating to use. On the flip side, a fast model with poor accuracy renders the retrieval system unreliable as

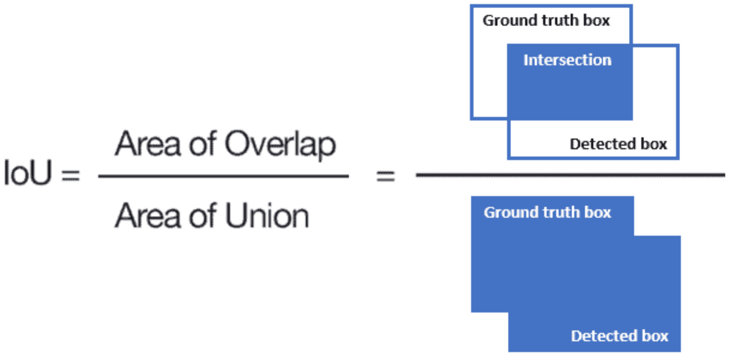

Figure 4: Intersection-over-union (IoU) is compares the predicted bounding box against the ground truth bounding box by finding the ratio between their intersection and their union.

human reviewers cannot be certain that no objects are missed. Therefore, our model needs to be fast and accurate.

7.3 Results

We evaluated four object detection configurations on the YouTube-VIS test split: pretrained RF-DETR, finetuned RF-DETR, pretrained YOLO12-m, and finetuned YOLO12-m. The finetuned RF-DETR model was trained for 20 epochs, with a learning rate of 1×10^{-4} and encoder learning rate of 1×10^{-5}. The finetuned YOLO model was trained for 40 epochs, with an initial learning of 1×10^{-3} and a learning rate factor of 1×10^{-4}. Performance was measured using standard detection metrics (mAP, precision, recall) as well as average inference time per image. All evaluations were conducted on the same NVIDIA RTX 2060 GPU, ensuring that runtime comparisons are fair and consistent.

Table 1: Performance of RF-DETR and YOLO12-m on the YouTube-VIS test split (frame-level evaluation).

Model	mAP@0.5	mAP@[0.5:0.95]	Precision	Recall	Time (ms/img)
RF-DETR (pretrained)	0.549	0.459	0.695	0.612	50.13
RF-DETR (finetuned)	**0.762**	**0.645**	**0.895**	**0.781**	55.96
YOLO12-m (pretrained)	0.478	0.400	0.746	0.540	23.08
YOLO12-m (finetuned)	0.584	0.452	0.850	0.615	**21.29**

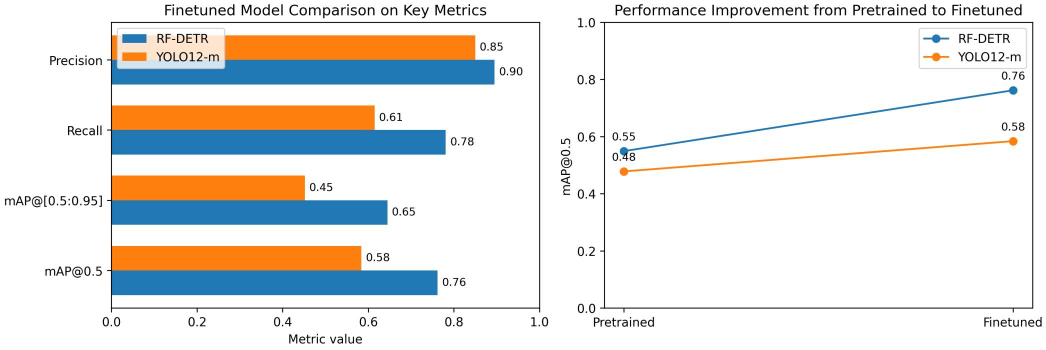

Figure 5: Left: comparison of finetuned RF-DETR and YOLO12-m across key metrics. Right: improvement in mAP@0.5 from pretrained to finetuned models.

Finetuning leads to clear improvements for both architectures. Compared to their pretrained counterparts, finetuned models detect more relevant objects, achieve higher accuracy, and better adapt to the specific categories present in YouTube-VIS. Importantly, these gains come without a meaningful increase in inference time, demonstrating that domain adaptation is a key factor in improving real-world performance.

Among all models, the finetuned RF-DETR achieves the strongest detection performance. It consistently scores highest on accuracy-related metrics, meaning it finds more objects and places bounding boxes more precisely. The finetuned YOLO12-m model is less accurate overall but processes images much more quickly, which makes it better suited for handling large amounts of video data efficiently.

Failure cases and challenging conditions. Despite improvements from finetuning, both models struggle in difficult visual conditions. In very dense scenes with many closely packed objects, YOLO12-m is more likely to miss small objects or merge nearby instances into a single detection. Both models also face challenges when objects are heavily occluded, visually camouflaged, or only partially visible. While RF-DETR handles clutter more reliably, neither model is fully robust under extreme visual ambiguity.

7.4 Discussion

Meaning of the Results for the Target Use-Case. The evaluation results confirm that automated object detection is a practical solution for analyzing video. Both model families are capable of reliably identifying objects of interest, especially after finetuning to the target dataset. This supports the project's goal of building a system that can automatically extract meaningful information from large volumes of video data.

A key takeaway is the balance between accuracy and speed. Models with stronger global reasoning capabilities perform better in visually complex scenes, while faster models enable efficient processing of large video archives. Since all experiments were conducted under realistic hardware constraints, the results reflect performance that can reasonably be expected in an early-stage deployment.

Overall, these findings show that object detection is a strong foundation for the broader vision of the project. By detecting and indexing objects at scale, the system can significantly reduce the effort required for safety reviews and prepare the ground for future extensions such as event detection and automated alerts.

Model Choice for the Sponsor's Vision. From the sponsor's perspective, the most suitable model is one that balances reliability, efficiency, and ease of integration. While RF-DETR offers the highest detection accuracy, its higher computational cost makes it better suited for detailed analysis or selective review tasks. In contrast, finetuned YOLO12-m provides faster and more predictable performance, making it a better fit for continuous video processing and large-scale deployment.

These results suggest a hybrid approach moving forward. YOLO12-m can serve as the primary model for scanning and indexing large volumes of video, while RF-DETR can be applied in situations where accuracy is especially critical. This strategy aligns well with the sponsor's long-term vision of building a scalable, extensible video analytics pipeline that can evolve toward higher-level understanding of events and actions.

8 System Design for Video Object Retrieval

8.1 Overview

Integrating a video object detection model, a retrieval system was designed to enable practical application in real-world scenarios. The core objective of this system is to transform model outputs into a structured and queryable format, meaning that users can issue searches—such as entering a video ID or selecting an object name—and the system can return the exact time intervals where those objects appear. This allows users to efficiently locate when specific objects appear within a video and visualize these results interactively.

The system automatically analyzes new videos using the trained detection model, records the detected objects and their corresponding timestamps, and stores these results in a centralized database. Through the retrieval interface, users can select a video, choose an object, and obtain a list of all timestamps where the object appears. The interface further allows users to click a timestamp and directly jump to that specific segment of the video. This design bridges the gap between model inference and end-user interaction, achieving a complete workflow from detection to visualization.

8.2 System Architecture

The overall architecture of the video object retrieval system consists of four layers: detection, storage, retrieval, and visualization. Each layer performs a distinct role within the data flow pipeline, as illustrated below.

The data flow of the system can be summarized as: *Video Input → Model Detection → Structured Storage → Query Retrieval → Interactive Visualization.*

8.3 Database Schema Design

The database schema is organized around two main tables: *videos* and *objects*. The *videos* table stores metadata for each processed video, including the file name, S3 storage path, duration, upload time, and the set of detected object categories. Each entry receives a *video_id*, which is an auto-generated unique identifier created by the PostgreSQL database. This ensures that every video can be referenced consistently, independent of naming conventions or storage changes.

The *objects* table stores detection results in a segment-based format rather than on a frame-by-frame basis. When the same object category appears across consecutive frames, the system merges these frames into a single detection segment and stores only the first frame and its corresponding timestamp. Each detection segment is assigned a unique *object_id* by the database, which serves as an identifier for that specific occurrence rather than representing an object category. The actual category label is stored separately as *object_name*, which corresponds to the semantic class predicted by the model (e.g., “person”, “car”). This representation avoids unnecessary duplication and prevents short videos from producing excessively large numbers of records.

During inference, the model processes the video frame by frame, but the resulting detections are consolidated before insertion. High-level video metadata is written to the *videos* table, while each merged detection segment is written to the *objects* table and linked to its parent video through *video_id*, forming a clear one-to-many relationship. This structure enables efficient queries of detected objects within a video while keeping the database compact and manageable.

When users issue queries—such as entering a video ID or selecting an object name—the retrieval module accesses the stored start timestamps, segment boundaries, and other related metadata to return the relevant intervals. The front-end interface then uses this information to jump directly to the appropriate moments within the video. Overall, the schema balances data granularity, storage efficiency, and retrieval performance, forming the foundational storage layer for the video object retrieval system.

8.4 Advantages and Discussion

- **Structured and Searchable Storage**: Detection results are stored in a relational database instead of log files, allowing for systematic indexing and future data analysis.
- **Flexible Object and Time Retrieval**: The schema design supports multi-dimensional queries by object category, video, and temporal range, providing efficient access to relevant segments.
- **Interactive Visualization**: Users can verify detection results intuitively by selecting objects and navigating to specific moments in the video interface.
- **High Extensibility**: The modular design allows the system to scale with additional videos, models, or metadata fields without significant structural changes.
- **Practical Integration of Model Output**: The system closes the loop between model inference and application, enabling direct use of detection models in downstream video analysis or deployment environments.

Overall, this design provides a comprehensive and scalable framework for transforming video detection results into an interactive and interpretable retrieval system, thereby enhancing the usability and applicability of computer vision models.

9 Conclusion

This project aims to build an efficient and scalable video recognition and retrieval system that enables users to quickly locate key segments in videos and improve analytical workflows. We implemented a complete process covering object detection, video parsing, metadata storage, and an interactive retrieval interface. The selected model demonstrates reliable accuracy on core object categories, and the system supports fast querying via video ID and object ID, allowing users to intuitively visualize and explore relevant video segments.

9.1 Future Work

Looking ahead, the system can be further developed to better support TechnipFMC's operational needs. Training and fine-tuning the models on internal video footage would improve performance in real production environments. Supporting both large-scale batch video ingestion and real-time streaming detection would enable faster safety reviews and continuous monitoring workflows. The system can also be extended beyond object detection to include object tracking and action recognition, allowing the system to capture how objects and people move and interact over time. Finally, refining the user interface would make the system easier to use for engineers and reviewers, improving adoption and day-to-day efficiency.

A Appendix

A.1 Ultralytics YOLO Training Parameters

Ultralytics provides a training mode for the YOLO models, with various configurations. Below is a list of some adjustable hyperparameters, based on the Ultralytics documentation.

1. Epochs: number of full passes over the training dataset
2. Patience: number of epochs to wait without improvement in validation metrics before stopping training early
3. Image size: target image size for training
4. Optimizer: choice of optimizer during training, affects convergence speed and stability
5. Fraction: specifies the fraction of the dataset used for training
6. Initial Learning Rate: influences how fast the model weights will be updated
7. Learning Rate Fraction: the ratio between the final learning rate and the initial learning rate, e.g. $learning\_rate\_fraction = \frac{final\_learning\_rate}{initial\_learning\_rate}$

References

Sani Abba, Ali Mohammed Bizi, Jeong-A Lee, Souley Bakouri, and Maria Liz Crespo. Real-time object detection, tracking, and monitoring framework for security surveillance systems. *Heliyon*, 10(3), 2024. doi: 10.1016/j.heliyon.2024.e34922.

Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*, 2020.

Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. *arXiv preprint arXiv:2005.12872*, 2020.

Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *arXiv preprint arXiv:1311.2524*, 2014.

M. Koteswara Rao and P. M. Ashok Kumar. Exploring the advancements and challenges of object detection in video surveillance through deep learning: A systematic literature review and outlook. *Journal of Theoretical and Applied Information Technology*, 103(6), March 2025. doi: 10.5281/zenodo.17178412.

Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. 2016. URL `https://arxiv.org/abs/1506.02640`.

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *arXiv preprint arXiv:1506.01497*, 2015.

Isaac Robinson, Peter Robicheaux, Matvei Popov, Deva Ramanan, and Neehar Peri. Rf-detr: Real-time transformer-based object detection and instance segmentation. `https://github.com/roboflow/rf-detr`, 2025a. SOTA Real-Time Object Detection and Segmentation Model.

Isaac Robinson, Peter Robicheaux, Matvei Popov, Deva Ramanan, and Neehar Peri. Rf-detr: Real-time transformer-based object detection and instance segmentation, 2025b. URL `https://github.com/roboflow/rf-detr`. GitHub repository.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. *arXiv preprint arXiv:1409.4842*, 2014.

Yunjie Tian, Qixiang Ye, and David Doermann. Yolo12: Attention-centric real-time object detectors, 2025. URL `https://github.com/sunsmarterjie/yolov12`.

Jasper R.R. Uijlings, Koen E.A. van de Sande, Theo Gevers, and Arnold W.M. Smeulders. Selective search for object recognition. *International Journal of Computer Vision*, 104(2):154–171, 2013.

Ao Wang, Hui Chen, Lihao Liu, Kai Chen, Zijia Lin, Jungong Han, and Guiguang Ding. Yolov10: Real-time end-to-end object detection. *arXiv preprint arXiv:2405.14458*, 2024a. URL `https://arxiv.org/abs/2405.14458`.

Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. *arXiv preprint arXiv:2207.02696*, 2022.

Shuo Wang, Chunlong Xia, Feng Lv, and Yifeng Shi. Rt-detrv3: Real-time end-to-end object detection with hierarchical dense positive supervision. 2024b. URL `https://arxiv.org/abs/2409.08475`.

Linjie Yang, Yuchen Fan, and Ning Xu. The 4th large-scale video object segmentation challenge - video instance segmentation track, June 2022.

Yongjian Zhao, Jian Zhang, Liang Chen, et al. Rt-detr: Detrs beat yolos on real-time object detection. *arXiv preprint arXiv:2304.08069*, 2023.

Z. Q. Zhao, P. Zheng, S. T. Xu, and X. Wu. Object detection with deep learning: A review. *IEEE Transactions on Neural Networks and Learning Systems*, 30(11):3212–3232, 2019. doi: 10.1109/TNNLS.2018.2876865.

Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable detr: Deformable transformers for end-to-end object detection. *arXiv preprint arXiv:2010.04159*, 2020.

Zhengxia Zou, Keyan Chen, Zhenwei Shi, Yuhong Guo, and Jieping Ye. Object detection in 20 years: A survey, 2023. URL `https://arxiv.org/abs/1905.05055`.