Eyes Up: Interpretation of Live Video for Characters and Digits

James Utley, David Mok, and Dhruvil Borda
{utleyj,dmok2,dhruvilb}@bu.edu

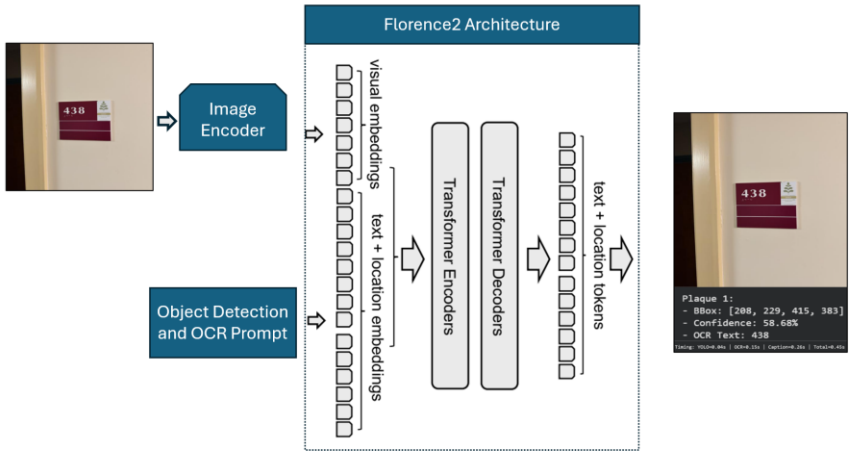

Figure 1. Sample diagram of purely Florence2 OD and OCR output with architecture from [3]

1. Task

The effective operation of autonomous systems in human-centric environments hinges on their ability to perceive and interpret the world in real time. Our project seeks to answer the question of how effective a VLM would detect and interpret live video for the purposes of robotic navigation. In order to make our approach feasible for the time and resources available to us, we narrowed our scope to focus primarily on room number panels such as those found in hospitals, schools, and airports. From these we investigated how to increase the speed of object detection and character recognition of bulky VLM architectures. In this project we compare two VLMs and their performance: Florence2 as the lighter model and Qwen2-VL as the heavier model control group.

2. Related Work

The idea to include VLMs in autonomous navigation is not new and current research is exploring how to deploy VLM architectures for real-time visual analysis. One such example comes from [8], which sought to mitigate VLM hallucinations in autonomous vehicles. [8] utilized a low-rank adaptation to an existing model which had the express purpose of identifying potential hallucinations in detection and reduced inference time of the VLM by 51-67%. However, what [8] did not discuss was the potential for errors in a long running, near infinite live stream, the primary focus of [9].

In [9], Xu et al. propose a unified framework and cache management to decrease inference-time with minimal training. While [9] reaches a frame rate of 8 frames per second, this speed was only achieved on a high end NVIDIA H100. Therefore the size of StreamingVLM is still too large to be reasonably deployed in real-time environments.

Lighter VLM frameworks, like the one described in [5], are showing promising real-time results. In [5], Robinson et al. propose RF-DETR which, they argue, show significant improvement on COCO and Roboflow100-VL, both of which are considered the current benchmarks for real-time detection. The results, while promising, require further verification, and [5] does not disclose its training methods or the exact frameworks for their models. A second model proposed in [2] offers similar results, with high performance on zero-shot inference, but does not provide clear results on the speed of its real-time detection.

While these lighter VLM frameworks offer promising results, we chose to use Florence2, proposed in [3]. One of the lightest weight open-source models with only 0.23 billion parameters, Florence2 offers the possibility of high performance alongside low-rank adaptability.

3. Datasets

This project utilized two distinct, custom-curated datasets, each created with a specific strategic purpose in the training and evaluation pipeline.

• Real-World Dataset: This dataset comprises 813 images of door panels collected across the Boston University campus and some universities in Michigan, capturing a variety of real-world conditions including different lighting, angles,

and sign styles. The images were manually annotated using Roboflow to create ground-truth labels for training and evaluation. The dataset was split into 652 images for training, 143 for validation, and 20 for testing. This dataset served as the ultimate benchmark for model performance on the target task.

• Synthetic Dataset: To increase the speed of VLM detection, we designed custom structured output grammar and generated a synthetic dataset of approximately 5,000 images. This dataset consisted of a scattering between three and eight random digit images from the MNIST dataset onto a blank canvas. The corresponding ground-truth labels in our target structured format were auto-generated, eliminating the time-consuming process of manual annotation.

This two-dataset strategy showed promising results. It allowed for efficient acquisition of the fundamental output grammar on a large, perfectly-labeled synthetic set, followed by domain-specific fine-tuning and generalization on a smaller, more targeted real-world set.

4. Approach

4A. *Yolov7 + Florence2*

To achieve the stated task, there are two major components that are required. The first is the ability to detect a door plaque and the second is to interpret the information on the door plaque. The model chosen for detection was YOLOv7. The dataset of 813 images was used to fine-tune the YOLOv7 with pre-trained weights. It was trained using an input resolution of 640 x 640 pixels, 75 epochs, batch size of 16, and stochastic gradient descent (SGD) with momentum on Google Colab with a T4 GPU. The loss was calculated using box loss and objectness loss. Box loss measures the accuracy of the predicted bounding box coordinates relative to the ground truth using intersection over union (IoU) and objectness loss uses binary cross-entropy to train the model to distinguish between regions containing objects and background regions. For post-processing, non-maximum suppression (NMS) was used to filter redundant detections through sorting predictions by confidence score, selecting the highest confidence prediction, removing predictions with IoU exceeding the threshold of 0.45, and looping this process until all predictions are processed. The model chosen for interpretation was the Florence2 model. This model uses a sequence-to-sequence architecture with a vision encoder (DaViT) and a multi-modal encoder-decoder transformer. Based on the output bounding boxes from the YOLOv7 model, the images were cropped before being interpreted by the Florence2 model. Through this model, the object character recognition (OCR) and caption tasks were used to interpret the information on the door plaques.

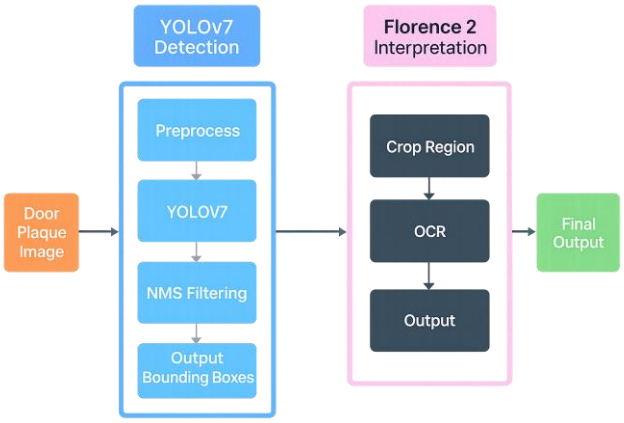

Figure 2. YOLOv7 + Florence2 block diagram

4B. *Florence2*

As described in 4A, Florence2 offers diverse multimodal capabilities. Since we got 4A to work, the question became, can we improve the performance by eliminating Yolov7 and doing both object detection and optical character recognition (OCR) on Florence2. To utilize Florence's capabilities, we used Low-Rank Adaptation (LoRA) to train additional weights while avoiding the compute required to retrain the entire model. We also intended to use Quantized Low-Rank Adaptation (QLoRA) to improve on the speed and to compare accuracy with the LoRA model. Ultimately the QLoRA proved too difficult to deploy, so we only trained a LoRA model.

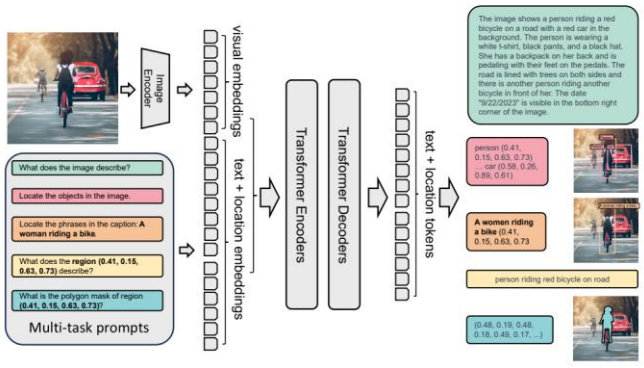

Figure 3. Florence2 Block Diagram from [3]

Our implementation of Florence2 relied heavily on HuggingFace's transformers library, bitsandbytes, accelerate, peft, datasets, safetensors, roboflow, and supervision. Furthermore, we used the blogpost by Piotr Skalski about how to finetune Florence2 as a base model for how to configure LoRA (our version is in our GitHub but the blogpost is available here: https://blog.roboflow.com/fine-tune-florence-2-object-detection/).

Finally, we ran into difficulty downloading the HuggingFace Florence2 model, since the current version of Florence

requires the use of Flash Attention which is outlined in [3]. Flash Attention is a type of attention that requires a powerful gpu to download and run it, and will only work on Python 3.10. Since Google Colab only runs Python 3.11 and 3.12, and the Shared Computing Cluster (SCC) at BU did not seem to be able to change Python versions, we were forced to work around Flash Attention, using the workaround described by James Braza published in a HuggingFace discussion group (found here: https://huggingface.co/microsoft/phi-1_5/discussions/72). I believe that later problems that we had with Florence2 stemmed from this adaptation of the model's built in attention module.

4C: *End-to-End Fine-Tuning of Qwen2-VL with QLoRA*
The second end-to-end approach was more ambitious, exploring the capabilities of a much larger VLM. The rationale was that a larger model like Qwen2-VL, with 2 billion parameters, would possess a greater capacity to learn the required output grammar with high fidelity and handle more complex visual scenes. To make training such a large model feasible, we employed QLoRA (Quantized Low-Rank Adaptation). This critical optimization technique combines 4-bit quantization of the model's weights with the addition of LoRA adapters, dramatically reducing memory requirements for fine-tuning from approximately 16GB to just 6GB of VRAM and making it accessible on consumer-grade GPUs like the NVIDIA RTX 3090. The number of trainable parameters was reduced to just 33 million, or 1.6% of the total model size. A two-phase training strategy was implemented:
1. Phase 1 (Grammar Learning): The model was first trained on a large synthetic dataset of ~5,000 images generated from MNIST digits to efficiently teach it the precise structure of our output format.
2. Phase 2 (Generalization): The model was then fine-tuned on our smaller, real-world dataset of 813 door panels to adapt the learned grammar to the specific visual characteristics of the target domain.
The following section details the datasets used to train and evaluate these approaches.

5. Evaluation Metrics

To conduct a comprehensive and multi-faceted evaluation of each approach, we employed a suite of metrics designed to assess performance across detection accuracy, text recognition correctness, localization precision, and processing speed.
• Precision: The percentage of positive detections that are correct. This metric measures the model's reliability in avoiding false positives.
• Recall: The percentage of all ground-truth objects that were correctly detected. This metric measures the model's ability to find all relevant objects.
• F1-Score: The harmonic mean of Precision and Recall, providing a single, balanced measure of a model's overall detection performance.
• Mean Intersection over Union (IoU): A measure of the localization quality of the predicted bounding boxes, calculated as the average area of overlap between predicted boxes and their corresponding ground-truth boxes.
• Text Accuracy: The correctness of the transcribed text label within a successfully detected bounding box, measuring the OCR capability of the model.
• Inference Speed: The time taken to process a single image, typically expressed in frames per second (FPS), which is a critical measure for real-time applicability.
• Parse Success Rate: The percentage of model outputs that successfully conform to the predefined structured grammar, indicating the model's reliability in generating valid, machine-readable data for downstream systems.
These metrics provide a holistic view of each model's strengths and weaknesses, which are detailed in the following results section.

6. Results

6.1 Yolo + FLorence2

From the completed training and validation, the YoloV7+Florence2 model achieved a precision of 95%, a recall of 95%, a mAP@0.5 of 98%, and a mAP@0.5:0.95 of 85%. The mAP@0.5 metric measures the mean average precision at an IoU threshold of 0.5 which means that a detection is correct if the bounding box overlap with the ground truth exceeds 50%. The mAP@05.5:0.95 metric measures the mean average precision averaged across IoU thresholds from 0.5 to 0.95 in increments of 0.05.

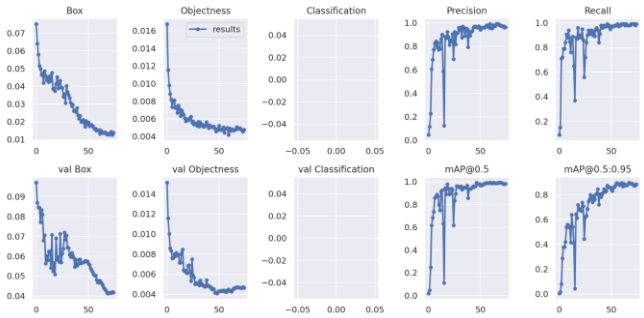

Figure 4. Training and validation results of approach A.

Figure 4 shows the output from the pipeline of approach A. The output includes the bounding box coordinates, the confidence of the bounding box, and the OCR text. The compute time is also calculated for each step of the pipeline including the YOLOv7 detection and the Florence2 interpretation.

Figure 5: Output of pipeline

6.2 Florence2

While the LoRA training of Florence2 was successful, attempting to reload the low rank trained model repeatedly failed (we theorize potentially because of the absence of Flash Attention) and was therefore unable to provide evaluation metrics. What we do have is the training and validation data results. While initially trained on a dataset of 2113 images (the original 813 images including several augmentations to the images), we only discovered after training that the augmentations had been applied only to the training data and not the validation data, skewing our results. When we re-trained Florence2 without the un-augmented 813 image dataset, we used four different learning rates:

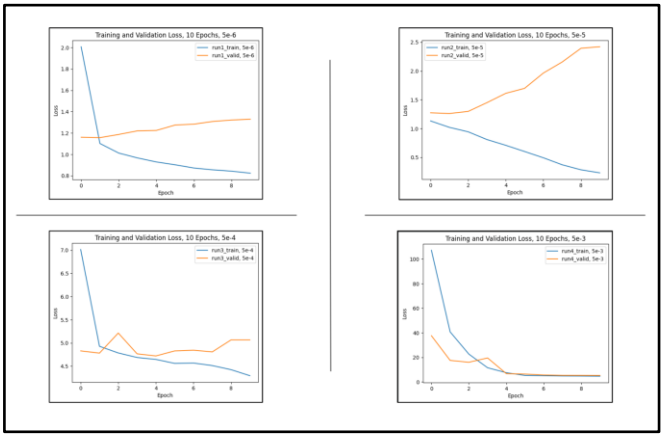

Figure 6: Florence2 Training and Validation Loss with four different learning rates

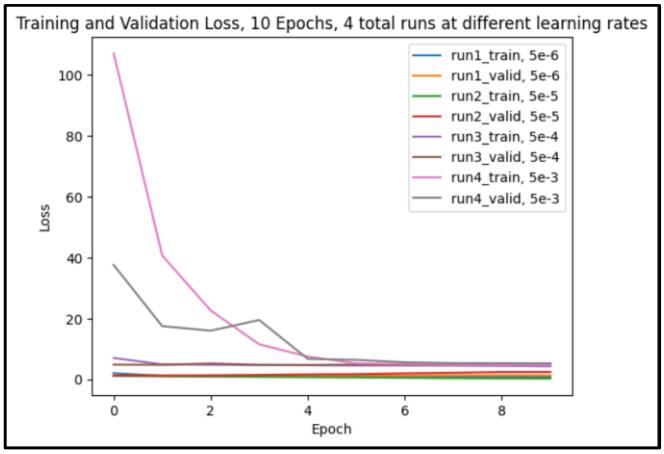

Figure 7: All Florence2 training and validation losses superimposed onto one graph

As you can see, the low rank adaptation of Florence2 did better on our dataset and improved through the epochs. We had the most success training with the learning rate 5e-4 and were we to continue with the project, we would likely use an adaptive learning rate and a larger dataset to ensure that the results behave more normally. I believe the aberrations that we are seeing in the training vs validation losses is due to the relatively small 813 image dataset. Given more time, we would expand our dataset and we believe the results would stabilize.

Given more time, we would further pursue Florence2 with a dedicated, local GPU able to download the Flash Attention and thereby have a model that will reliably run after LoRA and QLoRA training.

6.3 Qwen2-VL

The end-to-end approaches were evaluated to determine if a unified architecture could provide a better balance. The table below presents the performance of the Qwen2-VL model, benchmarked on a representative 50-image subset of the validation data. The following analysis focuses on the validated performance of the Qwen2-VL approach against the established baseline.

Metric	Qwen2-VL (QLoRA)
Precision	0.909
Recall	0.837
F1-Score	0.872
Mean IoU	0.837
Text Accuracy	0.882
Inference Speed	3.22 sec/img (0.31 FPS)
Parse Success Rate	~98%

The Qwen2-VL (QLoRA) model emerged as the clear 'accuracy champion' of our experiments, achieving high scores across precision (91%), recall (84%), mean IoU (84%), and text accuracy (88%). This demonstrates the capacity of a large VLM to effectively learn both localization and recognition tasks simultaneously. However, this high accuracy came at a significant cost to performance. With an inference speed of just 0.31 FPS, the model is far too slow for real-time navigation. Finally, the model demonstrated an exceptional ability to adhere to the specified output format, achieving a ~98% parse success rate. This makes its output highly reliable and easily consumable by downstream programmatic systems.

7. Conclusion

While we were not successful in proving the efficacy of VLM design for the purpose of real-time detection, we successfully demonstrated that VLMs have the capability to replace more traditional detection heads. Ultimately, for a VLM to be competitive in real-time detection, it would require more time and resources than was available to our team. Like [2] and [5], we would likely need to develop our own VLM architecture designed for efficiency in one particular task. Both Florence2 and QwenV2 are bulky multimodal systems that have a lot of features that would be out of the scope of our project and needs. Trimming away some of these unnecessary features would increase the efficiency of the model and would thereby make it faster.

We were able to accomplish a proof of concept for these ideas, reducing the number of outputs through QwenV2 to increase efficiency of the system, and achieving Low Rank Adaptation of Florence2. Furthermore, YoloV7 + Florence2 provided the best results and would be another direction we would investigate in the future, namely how we can make Florence2 more efficient and faster at Optical Character Recognition.

Although traditional detection pipelines remain the superior choice for applications where speed and efficiency are paramount, it is the belief of the authors of this project that our work is a step in the direction of one day using transformer architectures for superior efficiency for object detection and optical character recognition.

Appendix A. Code repository

GitHub Repository:
https://github.com/jam-utley/EC523_Project-Room_Number_Navigator

References

1) A. Jagan Sathyamoorthy, K. Weerakoon, M. Elnoor, A. Zore, B. Ichter, F. Xia, J. Tan, W. Yu, and D. Manochia. CoNVOI: Context-aware Navigation using Vision Language Models in Outdoor and Indoor Environments., arXiv.org, 2024
2) B. Kim. Ultra-Light Test-Time Adaptation for Vision--Language Models. https://arxiv.org/abs/2511.09101, 2025.
3) Bin Xiao, Haiping Wu, Weijian Xu, Xiyang Dai, Houdong Hu, Yumao Lu, Michael Zeng, Ce Liu, Lu Yuan: Florence-2: Advancing a Unified Representation for a Variety of Vision Tasks. https://arxiv.org/abs/2311.06242, 2025
4) Chien-Yao Wang, Alexey Bochkovskiy, Hong-Yuan Mark Liao: YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. https://arxiv.org/abs/2207.02696, 2025
5) I. Robinson, P. Robicheaux, M. Popov, D. Ramanan, N. Peri. RF-DETR: Neural Architecture Search for Real-Time Detection Transformers. https://arxiv.org/abs/2511.09554, 2025
6) J. Redmon, S. Divvala, R. Girshick, A. Farhadi. You Only Look Once: Unified Real-Time Object Detection. https://arxiv.org/abs/1506.02640, 2015
7) K. Garg, X. Tang, J. Heo, D.R. Morgan, D. Gergle, E. B. Sudderth, A.M. Piper. "It's trained by non-disabled people": Evaluating How Image Quality Affects Product Captioning with VLMs. https://arxiv.org/abs/2511.08917, 2025
8) K. Long, J. Guo, T. Zhang, H. Yu, X. Li.A Low-Rank Method for Vision Language Model Hallucination Mitigation in Autonomous Driving. https://arxiv.org/abs/2511.06496, 2025
9) R. Xu, G. Xiao, Y. Chen, L. He, K. Peng, Y. Lu, and S. Han., StreamingVLM: Real-Time Understanding for Infinite Video Streams. https://arxiv.org/abs/2510.09608, 2025
10) Y. Li, S. Qian, H. Liang, L. Zheng, R. An, Y. Guo, and W. Zhang. CapGeo: A Caption-Assisted Approach to Geometric Reasoning. arXiv.org, 2025