VMART: Rethinking Art Gallery Experience through Painting Detection, Classification and Retrieval

Davide Bilardello
“Enzo Ferrari” Department of Engineering
Modena, Italy
285039@studenti.unimore.it

Federico Melis
“Enzo Ferrari” Department of Engineering
Modena, Italy
287301@studenti.unimore.it

Emanuele Prato
“Enzo Ferrari” Department of Engineering
Modena, Italy
284215@studenti.unimore.it

Abstract

In this paper, we propose a novel way to explore art galleries, leveraging on various computer vision and AI-based techniques. In the following sections, we show how a structured elaboration pipeline combined with a well-integrated mixed reality scene can achieve very good results and provide a new way of exploring museums. We employed plenty different techniques and strategies, exploring their advantages and their drawback. Our Visual Museum Augmented Reality Tour, VMART, uses YOLO object detection to localize paintings and adjust perceptive distortion with classical computer vision methods. We then use DINOv2 to create embeddings used to help the painting construction and consent the retrieval. Finally, we use a modified EfficientNet finetuned for style and genre classification. We designed a new way of exploring a museum, by consenting a direct interaction between the user and the paintings, with information and similar artwork displayed on-demand.

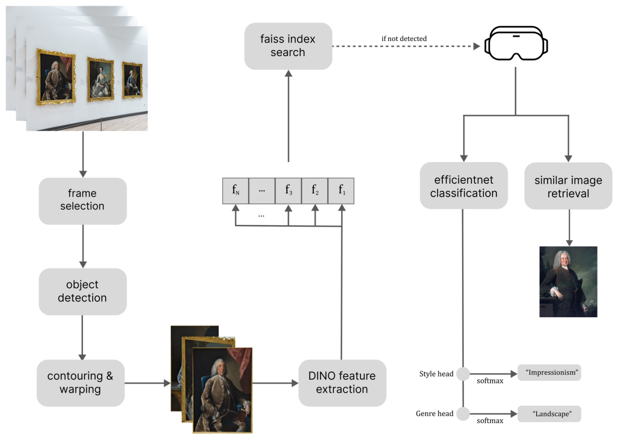


Figure 1. The figure shows the full pipeline. On the left part we see the detection and transformation process. In the middle, the feature extraction. In the right, we see the classification and the retrieval

.

1. Introduction

In recent times, deep learning methods demonstrated interesting results in various fields. In particular CNN-based architectures [14, 33, 38] have made success and pushed the research on this topic. Task such as object detection and classification achieved big improvement making them easier, lighter and reliable. Tackling these tasks with supervised learning is not the only way; in fact, there are multiple learning strategies including self-supervised learning that almost reached weakly-supervised learning strategy performance.

In our project we have to properly detect paintings in museums, classify them and perform retrieval. While exploring the space of solutions we encountered some tricky problem to deal with. For instance, generalizing well on paintings detection task is not trivial, given the fact that exists different border’s shape (e.g. circular, rectangular, etc.).

In the past, attention-centric [43] approaches was mainly employed in Natural Language Processing (NLP) tasks. Nowadays, thanks to the big improvement given by the research, this approach is shifting also in computer vision, creating Image Encoder [8] and variety of models that relies on them [4, 5, 29].

Our researches led us to exploit different kinds of model for each task. In the pre-processing task we evaluate and select the best frame coming from the headset, i.e. META Quest 3, that will be the more proper image in which can be done the detection. In object detection task we locate paintings with YOLO [30] and warped them to correct the perspective distortion. Once the paintings are detected, they are sent to DINO to verify if they are already been projected as an object in the Unity scene, with which we can interact. The interaction provides two responses; the first one is the classification of the interested painting, exploiting the ResNet [14, 39, 40]; the second one is the retrieval of similar paintings extracted from the WikiArt dataset, exploiting the DINOv2 [29]. The whole pipeline is shown in 1.

The complete codebase is publicly available at: `https://github.com/cvcs-vmart/vmart`.

2. Related Work

Object Detection. In the last years, object detection has became one of the most investigated computer vision field [46]. Nowadays there are plenty works and studies on this specific sector and the task is almost solved. The techniques used are various, including CNN-based methods [12, 13, 19, 30, 31] and attention-based ones [4]. This approaches have demonstrated to be well designed and capable of achieving a good level of generalization even on a big number of possible object classes [6, 23]. However, there are some open challenges, like how to making faster inference and reducing the number of needed parameters. Further research investigates how to perform object detection in self-supervised or even unsupervised manner [44].

Transfer Learning and Finetuning. A key idea in modern computer vision is to first train deep neural networks on huge datasets and then adjust them for specific tasks. ImageNet [6] is a celebrity in the dataset field, and is used to provide an effective pre-training for plenty different tasks. Usually when talking about ImageNet, we are referring ImageNet ILSCVR used in the famous ImageNet competition; this version of the ImageNet dataset only contains 1.000 classes while the complete one have like 20.000 classes. Later, "Big Transfer (BiT)" [21] [45] showed that training even larger models on even bigger datasets led to better performance and adaptability across many visual tasks. BiT also stressed the importance of good "fine-tuning recipes", recommending simpler optimizers and proper learning rate settings. Our project uses an EfficientNetV2-L model [39] [40].

Adjusting these pre-trained models for new datasets and tasks, called fine-tuning, is very important. While older methods often just trained the last part of the network, it's now common to fine-tune the whole network, especially if the new dataset is large enough. How you change the learning rate during this process is crucial. Many strategies exist, like fixed rates, cosine annealing [26], and cyclical learning rates [34]. Our fine-tuning method uses a special learning rate schedule: it starts with a "warm-up" period where the learning rate is zero for t_1 epochs, then it goes up steadily for t_2 epochs, and then it goes up and down in cycles with a period of t_3. This helps the model learn effectively.

Categorizing artistic images by style and genre is hard because "style" and "genre" can be abstract concepts. However, with deep learning, convolutional neural networks have become much better at recognizing the complex visual patterns in art [15, 45]. Datasets like WikiArt [41]. Our work directly contributes to this area by showing that advanced pre-trained models and our specific fine-tuning methods can accurately classify artistic style and genre using the WikiArt dataset.

3. Approach

Our approach consists in detecting paintings within a museum environment, process it to adjust perspective and project an interactive layer within the scene if the painting result a new one after the re-identification phase. We then let the user choose when to obtain similar images leveraging on DINO and Faiss and obtaining information with our classification on genre and style.

3.1. Headset and server communication loop

The communication between the computer vision pipeline and the headset application is crucial and important as it is the starting and ending part of the loop. The entire process begins with the headset transmitting a video stream to a Python server via a WebSocket connection. The server communicates newly detected paintings back to the headset using an HTTP request.

Before detailing approaches used to solve these tasks, we will describe the mixed reality headset technologies employed.

Interacting with the environment with Meta SDK for Unity. The Passthrough Camera is Meta's most recent API release, designed to facilitate interaction with the native cameras of their headsets. This API allow capturing video stram from the frontal cameras, applying on device computer vision algorithms, placing 3D elements in the world space, and more. Depth API is a support API that enables the possibility to raycast into the user's field of view (FOV) enabling mixed reality interaction. Furthermore Scene API allow scene-aware experiences by mapping the rooms before its utilization.

Real-time pipeline. The Figure 2 illustrates our real-time pipeline. A video stream from the headset is taken and sent to a python server that acquires the data. We use a WebSocket for this communication to ensure a lossless byte stream. This is important for maintaining lossless frame quality and for potentially sending additional coupled data. After the complete computer vision pipeline, the final stage sends the bounding boxes of the detected paintings back to

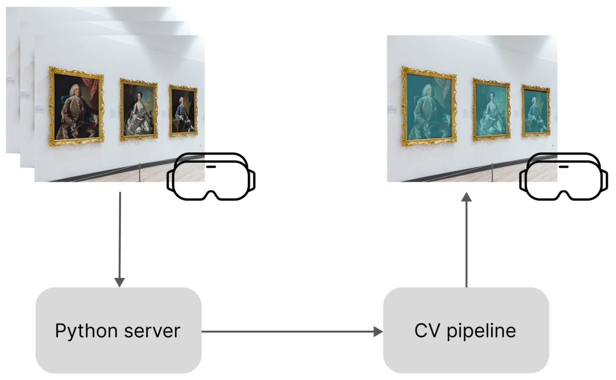


Figure 2. Headset loop

the headset, including their unique IDs. This communication is done via an HTTP request from the last python service to the headset. Finally, on the headset side, the Depth API projects these bounding boxes into 3D space.

This approach works very well but it has a critical constraint: the entire loop must operate very quickly otherwise the headset movement can cause the bounding boxes to fall outside the user's field of view.

Non Real-time pipeline. The next two approaches are based on the idea of using the headset's historical pose (position and rotation) to project and place bounding boxes at a later time. However, knowing only the headset's historical pose is insufficient for accurate ray casting. We used the left eye camera and this are its parameters:

- $(x_c, y_c) = (639.7092, 480.8897)$: This is the principal point, which represents the exact center of the image sensor where the camera's optical axis ideally intersects.
- $(f_x, f_y) = (866.6758, 866.6758)$: These are the focal lengths, expressed in pixels.
- $s = 0$: This is our skew coefficient. A value of zero indicates that the pixel axes of the camera sensor are perfectly perpendicular.
- $(w, h) = (1280, 960)$: These are the dimensions of our acquired image. This is not an intrinsic parameter.

Depth API can still be used to estimate painting placement by simply raycasting the FOV with a matrix of rays. Without prior knowledge of a painting's location, the intersection points with the real world are used to determine the wall's orientation and position during data acquisition. However, this method has a significant drawback: it's extremely computational intensive, causing visible delays.

Scene API offers an alternative, allowing us to hit with our rays a 3D model of the room from any direction. This solution requires a room scan, but it provides a zero lag experience.

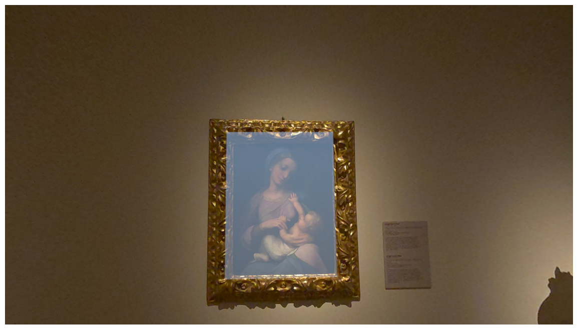


Figure 3. A debug layer is placed to show the detected painting

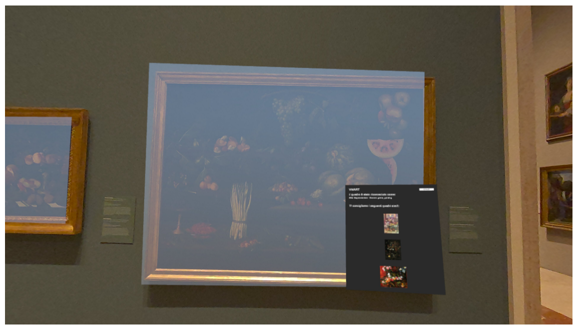


Figure 4. A detected painting with its information panel

Painting interaction. Once a layer is placed over a detected painting in the real world, the user can interact with it using the controller's trigger. This action brings up a floating panel right in front of him. This panel displays the genre and style of the painting, as given by our ResNet model, along with the top 3 most similar paintings from our retrieval model.

3.2. Painting Detection

Providing a good painting detection is an hard challenge to face with. Given an image our goal is identify all the instances of painting class. The result of this step will affect the entire pipeline, so we spent time in developing a reliable and efficient way to perform the task. The real-time constraint and the complexity of the environment have created the needs of a model able to perform real-time object detection on detailed image acquired from an unreliable source also affected by egocentric motion that can completely destroy the image quality.

Painting detection plays a key role in our context and doesn't have an absolute solution; since a lot of environment variables and domain-specific needs can affect some solution effectiveness making certain solution more suitable then others we explored different possible solution before making a choice. In particular, we explored both classical computer vision approach based on elementary operation (e.g. gradients and morphological operator) and more mod-

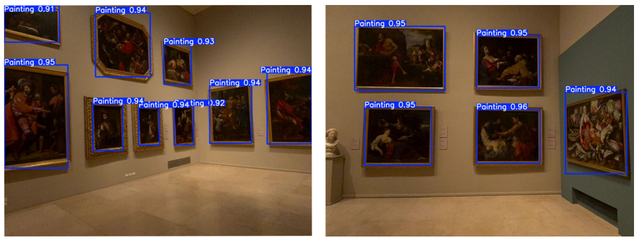


Figure 5. This figure shows some qualitative results of YOLO detection. Note that our system discard those detection that are close to borders, avoiding the projection of slice of paintings.

ern approach, based on deep learning techniques.

3.2.1. Classical Computer Vision Object Detection

In this section, we discuss our work on thinking and developing a deep learning-free pipeline for process images and detect paintings.

Hough-based Detection. The research stared from here, investigating how we can identify an arbitrary number of paintings inside a single image. This first solution relies on Hough transformation [10], and in particular on one of its variant. Hough Transformation [10, 17] is a very common technique used to identify straight lines in images and we used it for extracting the border of paintings.

We begin the Hough-based detection pipeline by resizing the input image to a standard dimension while keeping the ratio between height and width. We also set a maximum dimension value that guides the resizing process. We then apply a filter on the resized image, usually a Gaussian blur or a Bilateral Filter [42], and then applying on this output Canny operator [3] to produce edges. Now we used Hough to identify and collect straight edges lines in polar coordinates and then project those lines into our coordinates system. Unfortunately this approach provides not precise lines, because we have to manually extend it, and without a prior knowledge on the paintings dimension it is almost impossible to guess a correct dimension that results suitable for both big and small paintings. We then moved our attention to a slightly different method, Hough Lines Probabilistic [28]. This Hough algorithm variant provides lines that are already in euclidean form and, assuming well defined edges, it provides lines with a reasonable length. Before continuing, we filter out oblique lines using a threshold on their slope. At this point we created two cluster based on angular coefficient, to discriminate horizontal and vertical lines. This was necessary because we want the interception points that should be placed around painting corners. Regretfully, Hough provides very noisy results, containing several redundant lines, and, despite the effort spent in develop non-maximum suppression [16] algorithm for both lines and interception points (e.g. a non-maximum suppression based on custom convolution operator), we doesn't manage to get clean results suitable for bounding box extraction.

Another weakness of this approach consist lack of reliability; this model was susceptible to all those object that exhibit some kind of straight lines producing instable results across different scenarios. It is also totally ineffective while detecting circular paintings.

Contours-based Detection. The contours-based paintings detection shares some pre-processing seen in the previous detection, but fully relies on a specific OpenCV operation, "findContours". This method is an enriched version of the Suzuki-Abe algorithm for finding contours that distinguish inner border from outer border.

We start by applying the same ratio-preserving resizing technique seen previously, and then proceed with some smoothing filter (e.g. Gaussian blur or Bilateral filter). Next, we apply Canny to compute edges and then apply in these edges the morphological closing operator i.e. a dilation followed by an erosion. This little adjustment results in minimal visual change, but it significantly affects the result of the next operations. At this point we apply Suzuki-Abe algorithms obtaining collections of points that represent some polygons. Now we filter out the noisy contours by thresholding on polygons area. At this point, we reach a critical stage of the process; in fact, we need to discard noisy contours or fit them to a regular size. We explored two different strategies, one that tries to fit those contours to a rectangular dimension, and the other one, made from scratch that tries to approximate an arbitrary polygon and then evaluate how rectangular it is.

The first attempt uses a well-established OpenCV function, boundingRect, a straightforward operator that extracts the smallest axis-aligned rectangle containing all the contour points. Although naïve, it is still effective.

The other approach consist in computing the convex hull of the contour and then approximate this polygon with another polygon with less vertex that falls in an approximation range based on the convex hull perimeter. If the produced approximation is composed by four vertexes, it continues in the pipeline, where we evaluate how much it is rectangular. At this point we select the triplets (p_1, p_2, p_3) of connected vertexes and define:

$$\begin{aligned} \overrightarrow{v_1} &= p_1 - p_2 \\ \overrightarrow{v_2} &= p_3 - p_2 \end{aligned} \tag{1}$$

This two vectors starts from p_2 and goes towards p_1 and p_3. We now compute the cosine and use it to evaluate the regularity of the rectangle. The evaluation process consists of quantifying the deviation of the polygon's angles from 90 degrees. We repeat this process for each contours. Each rectangles is saved with a score associated, that indicates how regular was the polygon.

In order to be resistant to domain shift and to handle images of different sizes, including fine-grained or low-resolution images, we managed to repeat the whole detection process for a set of Gaussian blur and Bilateral filter hyperparameters. For each set of hyperparameters, we produce the bounding box and save it. We then sort by the number of bounding boxes found in descending order, and subsequently by the score, also in descending order; by doing this, we should obtain the best output based on heuristic. This multi-scale approach allow us to be less dependent on painting composition.

This methods works relatively well, but it present some critical issue due to the complexity of the task; even with this method we are not able to catch non-rectangular paintings, and we are susceptible on the environment construction (i.e. ropes that hold up painting are detected as contours).

3.2.2. Deep learning-based Detection

After accurate researches on classical computer vision approach, we decided to migrate through a more robust object detection algorithm. The advent of deep learning-based object detection systems, based on CNN [12, 13, 24, 30, 31] and also on Transformer [4], has made possible providing a generalized and domain-agnostic understanding of all the characteristics of known objects, leveraging on both receptive field and attention mechanism [43]. For our purposes, we found a good tradeoff between architectural complexity, computation required and data need. In particular, we chose a CNN-based model, YOLO [30].

Painting Detection with YOLO. We chose the nano-sized model of YOLO11 models family [19]. Since we doesn't had so much time and data, we leveraged on already existing datasets; in particular we trained and tested the detection in three different datasets [1, 2, 37] found on Roboflow (see experiments section for further details); all those dataset contains around 1k images. Our train configuration includes batch size of 32 images of size 640, and it does 100 epochs at most, with a patience of 20 epochs. This results in around 32 steps per epoch. Then, we use 10 epochs of warmup, and then, a cosine annealing schedule [25]. We also used dropout [35] of 0.4. After several optimizer changes [20, 27, 32], we chose to let YOLO decides the best optimizer. During the dataset creation we applied some data augmentation technique such as random blurring, auto orient, random saturation and brightness variation. In addition we let YOLO apply the usual online augmentation technique modifying the impact of some of them (hsv_S, mosaic, scale, perspective). In particular, we slightly increased the frequency of scale and perspective distortions. This adjustment is expected to be impactful in our domain, where egocentric motion can significantly affect the perceived scale and orientation of paintings. By training in this way, the model exhibit a good generalization and is able to catch almost all the paintings, even the smaller one.

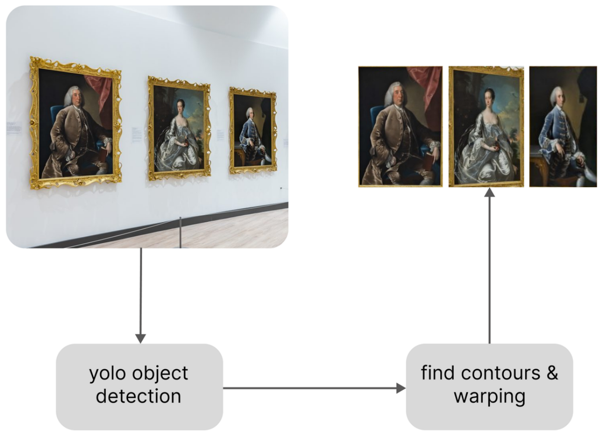


Figure 6. The figure shows an example of input that go through the detection system and is then processed by the transformation pipeline, producing the images on the right.

However, this high level of generalization and robustness against very unlikely paintings may leads to errors; for instance, detecting just a slice of a painting or a very distant one can be a problem.

In our specific case, we don't need to recognize either the furthest or the incomplete paintings. We handled this problem by limiting a bit the detection. At first, we rise the confidence required to output a prediction, avoiding the majority of noisy recognition including the smaller paintings that usually comes out with around 0.85 confidence.

The partially visible paintings instead, has been removed by filtering out the detections that results too close to the image borders.

3.3. Transformations

In the previous section, we faced the challenge of identify and provide contours or bounding boxes for paintings. As mentioned above, we chose to employ the proposed deep learning-based approach to perform this detection. Unfortunately, YOLO and most of the existing method for object detection provides bounding boxes. As we now, bounding boxes are not the best way to express a precise contour; the problem is that a bounding box remains rectangular regardless of the actual shape of the object and the orientation of the acquisition point wrt the object itself. This limitation can be misleading for further tasks (similarity, classification and retrieval). Despite those task relies on deep learning approach that absolutely overtake this kind of problem, we want to produce a standard and reliable representation for the images, in order to be even more resilient. We finally addressed this requirements combining a contour detection algorithm with an homography-based image transformation.

Contour Extraction and Homography Warping. This approach consist in applying the already seen method for contour extraction based on Suzuki-Abe algorithm, but within a extremely simplified scenario. The subsequent step, is warping the image leveraging on perspective transformation. Due to the existing complexity and considering those scenarios with small detected paintings, we opt for best effort operations.

We begin this step with a set of bounding boxes given from the detection module. For each of those bounding boxes, we apply a padded crop on the complete image to extract the painting. This cropping strategy provides some leeway for the contours extraction phase. After the crop has been made, we perform the usual ratio-preserving resize. Now, starts the contours extraction process; briefly we smooth the painting, preparing it for Canny. After this, we close some little gaps with morphological closing operator and give it to "findContours" that extract the contours. We now apply our convex hull-based polygon approximation. At this point we sort the contours by area in descending order and keep just the largest. This should be the painting contour or in worst case scenario the contour of the cropped image itself. Then, if the area is at least half of the image area, we proceed by sorting the contour points (top-left, top-right, bottom-right, bottom-left) and estimating the optimal homography matrix with "findHomography" that compute the transformation matrix for mapping the four source points to the four angles of a new image. Finally, we compute the perspective transformation using "warpPerspective" [11]. To be able of computing the correct homographic transformation we need at least those four points; recalling we want to estimate this matrix:

$$\begin{bmatrix} X' \\ Y' \\ 1 \end{bmatrix} = \begin{bmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} \tag{2}$$

where (X', Y') indicates the destination and (X, Y) the source (in homogeneous coordinates). A destination point can be expanded by doing the matrix multiplication and doing the conversion from homogeneous to inhomogeneous coordinates:

$$\begin{cases} x'_1 = \dfrac{h_{00}X_1 + h_{01}Y_1 + h_{02}}{h_{20}X_1 + h_{21}Y_1 + 1} \\ y'_1 = \dfrac{h_{10}X_1 + h_{11}Y_1 + h_{12}}{h_{20}X_1 + h_{21}Y_1 + 1} \end{cases} \tag{3}$$

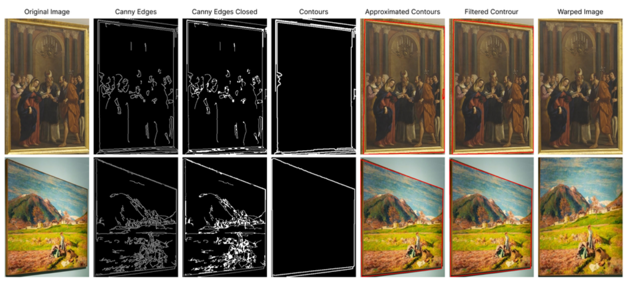


Figure 7. This figure illustrates the entire transformation pipeline from extraction of edges to image warping.

This is the expression of the inhomogeneous destination coordinates for the first point. Having four of this points, results in a system of eight equations in eight unknowns (the h_{ij} values), thus forming a solvable system.

With this transformation pipeline we provide better inputs for re-identification phase, avoiding all the problems related to perspective and rotation. As shown in Figure 6 the perspective correction is done only when the produced contour is clear and allow to correctly identify the painting corners. A qualitative result is provided in 7.

3.4. Embedding Construction and Storing

Once the paintings are detected and warped, are sent to DINOv2 [29] employed to extract high-performance visual features useful, in our case, to verify if a painting is already detected previously and to do retrieval. To facilitate the achieving of these tasks' objectives, Faiss [9, 18] proves valuable, enabling efficient similarity searches between vectors in high-dimensional spaces, i.e. what DINO extracts.

DINO feature extraction. There are eight models available; in this project is used the smaller one (ViT-S).

The approach is very simple:

1. A painting image that is detected and warped is processed and is given as input to DINOv2;
2. As output it returns a vector of dimension (1, T, E), where T is the number of patch tokens + 1 which is the CLS token, and E is the embedding dimension, i. e. 384 in ViT-S. Subsequently, the mean is calculated along the first dimension, resulting in a vector of size (1, 384);
3. The vector is normalized with the L2-Normalization:

$$v_{norm} = \frac{v}{||v||_2} \tag{4}$$

where

$$||v||_2 = \sqrt{\sum_{i=1}^{d} v_i^2} \tag{5}$$

FAISS search.

1. The index of the paintings already detected is loaded, it is computed the Euclidean distance between the normalized vector and all vectors of the index;
2. If the distances are bigger than the threshold (hyperparameter), or if the index is empty, i.e. no paintings are detected previously, then the painting is constructed in the Unity scene and the index is updated;
3. Else the painting is discarded.

The normalization part is not mandatory due the fact that the Euclidean distance [7] doesn't require normalization to compute the distance between the painting and the vectors in the faiss index, but with normalization the distances are more readable and the chose of a good threshold becomes easier. Furthermore, when the faiss index is an instance of IndexFlatL2 (i.e. this case), the Euclidean distance is employed to determine similarity. However, if IndexFlatIP is used, the calculation involves the inner product, that is like to perform cosine similarity [22, 36] if the vectors are normalized.

Threshold Selection. In the 5-th step of the method described before, there is the employ of a threshold to decide what paintings are already detected. The selection of a threshold is challenging because setting it too low might lead to misidentifying a previously observed painting if viewed from a different angle, whereas setting it too high could result in omitting paintings that aren't detected but are sufficiently similar to those detected to fall within the threshold. Our method involves analyzing the WikiArt dataset, where paintings are categorized by style, to determine in-class similarity for each group. Given the fact that could be some duplications of the same paintings, and that we don't know how many paintings there are in each class, we considered the 25th percentile distance for each element inside the class and compute the mean between these distances. This is done for each class and then is computed the mean between the results. Next, we examined five paintings from various angles, perform similarity across these perspectives, and calculated the average similarity for each painting. Finally, we combined these results by computing an overall average, aiming to define a robust threshold suitable for the task.

3.5. Style and Genre classification with a ResNet

Task and Dataset. An important step of our pipeline is the classification of genre and style of paintings, providing useful information to users. For our purposes, we utilized the WikiArt dataset [41], a collection of 80,000 paintings. The WikiArt dataset consists of 27 style classes and 11 genre classes. Examples of styles include Impressionism, Cubism, Baroque, and Romanticism, while genre includes categories such as illustration, abstract painting, landscape, and portrait. The genre classification also includes an "Unknown Genre" label.

The Net. We started our work with a ResNet [14], a milestone in the story of CNNs. In fact, its residual block are still a widely diffuse practice to work with. Initially, we experimented with standard ResNet architectures; however, their performance resulted suboptimal for our specific task. We then transitioned to EfficientNetV2-L, pre-trained on ImageNet. We observed that this new employed architecture outperform the previous ResNet configurations.

EfficientNets were developed to challenge the common practice of randomly increasing model's depth, number of channels per layer, and input resolution hoping in improvements. Instead, they proposed a balanced scaling approach across these three dimensions. This process is known as "Compound Scaling". ϕ is the compound coefficient that uniformly scales the model's width, depth, and resolution. Meanwhile, α, β and γ are constants that are determined to find the most optimal scaling. These exponent values are usually found through practical methods like grid search or an optimization process.

$$\begin{aligned}
\text{depth: } d &= \alpha^{\phi} \\
\text{width: } w &= \beta^{\phi} \\
\text{resolution: } r &= \gamma^{\phi} \\
\text{s.t. } \alpha \cdot \beta^2 \cdot \gamma^2 &\approx 2 \\
\alpha &\geq 1 \\
\beta &\geq 1 \\
\gamma &\geq 1
\end{aligned}$$

EfficientNetV2 was developed to speed up the training process by considering advances in new GPUs and recent research discoveries, while also incorporating effective regularization.

Classification Approach. As mentioned, we need to discriminate paintings on two classes, style and genre. To achieve this, we removed the default final MLP layer of the EfficientNetV2-L by replacing it with two MLP heads, one dedicated to genre classification and the other to style classification. Each of these heads is made of three hidden layers, each containing 512 neurons, followed by an output layer corresponding to the respective number of classes. For regularization, we applied a dropout [35] rate of 40% before the input to each head layer and incorporated a weight decay of 10^{-5}. To enhance performance, the entire EfficientNetV2-L model was fine-tuned with a learning rate of 10^{-5}. Conversely, the learning rate for each individual head was fine-tuned as detailed in the "Related Works"

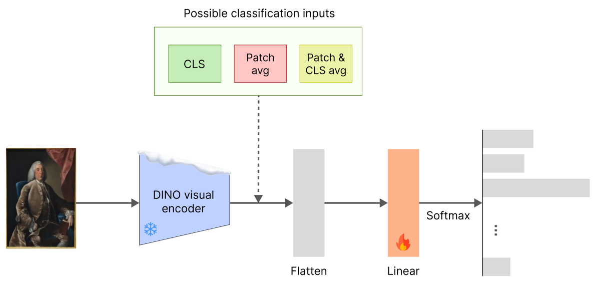


Figure 8. Architecture for classification with DINOv2. It uses the frozen DINO image encoder and a linear layer that projects the representation into the class space, using either the CLS token, the average of patch tokens, or a combination of both.

section, with a maximum learning rate of 10^{-3}. Prior to training, we removed the "Unknown Genre" label from the dataset and implemented a sampler to assign weights to each image based on the frequency of its associated style within the entire dataset. We will refer at this implementation as EfficientNetV2-L.

3.6. Retrieval

It is decided to adopt the WikiArt dataset to do retrieval with the paintings we encounter in the museum. Firstly all images of the dataset are sent to DINOv2, for the feature extraction and embedding creation, then they are added in a IndexFlatL2 index saved as *all_paintings.index*. For the retrieval part the approach is very similar to what described in the **3.4 Embedding Construction and Storing**. It is considered the embedding of the painting of interest captured in the museum and is computed the similarity with the paintings inside *all_paintings.index*. There will given the five (what we requested, but any number is fine, e.g. the first more similar element, the top ten, etc.) more similar paintings that lives inside the index. These paintings are sent to the VR headset.

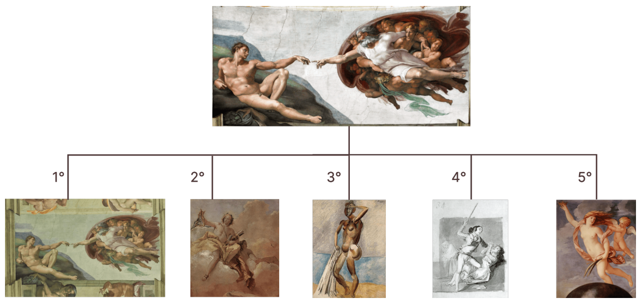


Figure 9. The figure shows a qualitative results for retrieval. The retrieved images are displayed ordered by their similarity wrt the input one.

4. Experiments

DINO as Classificator. In the beginning we decided to rely on a ResNet to do our classification. However a ResNet, since it's trained with cross entropy, tends to learn transformed features that results helpful in distinguish between genres and styles. So, it is possible that a CNN tries to extract not only the semantical features but also some low level features. We asked ourself what happens if we use a self-supervised feature extractor like DINOv2, that has not been trained on that classification task. This was an interesting experiment because we are exploring how much effective can be a zero-shot classification based just on semantical features.

During our exploration, we built a model that uses a frozen image encoder and then a linear layer to project on the classification dimension. This is shown in 8. But what's the best features to do this classification? In fact our DINOv2 variant produces 257 enriched outputs, obtained using the attention [43]. The first is the CLS token and the other 256 are the transformed input patches. We conducted 3 different classification tests, using different feature vectors:

1. Just the CLS token.
2. The average of the transformed patches.
3. The average of both transformed patches and CLS.

The results shown in 3 does not shows a clear winner, since the three feature vectors seems to have a similar impact. However taking the average of both patches and CLS ends up in a slightly high accuracy on the style while losing some points in genre.

Note that we trained each model separately; for instance, considering the CLS version, trained and tested a model for the style and one for the genre.

YOLO Ablation Study. In this section, we provides our ablation study on our YOLO11n model. As shown in 1 we removed or substituted each of the component of chosen baseline. We reported mAP50, mAP50-95 and F1 score. The mean average precision metrics reports the level of IoU in the validation set while F1 gives an indication of recall/precision tradeoff. Our baseline achieves 91.1% in mAP50-95, 98.1% in mAP50 and 95.5% of F1 score. In fact, turning off dropout lies to a slight improvement in mAP50 and F1, but we decided to stick to our baseline since we give more importance to mAP50-90 that measures in a more severe way the IoU; it produces a mean of the mAP over 10 thresholds of IoU, providing a more complete evaluation, rewarding the more accurate detections.

Object Detection Dataset. The dataset choice and its quality its clearly a crucial point. For our purposes we needed a dataset containing a decent amount of paintings

Variant	mAP50-95 (%)	mAP50 (%)	F1
Baseline	90.1	**98.6**	**95.9**
No Warmup	89.0	97.3	94.7
Multi Scale	88.9	96.4	94.5
No Augment	88.5	97.8	95.7
No Cosine	89.5	97.7	94.8
Adam	88.3	98.4	95.3
SGD	90.0	97.9	94.6
Our proposal	**91.1**	98.1	95.5

Table 1. YOLO Ablation Study.

located in museums or art galleries. Finding a good dataset that fit the task, it's not so trivial; in fact, we spent several hours in doing this. Due to the lack of time and resources we abandoned quickly the idea of building our dataset, moving our attention to Roboflow, that offers countless datasets constructed by community. For our specific case, we found 3 datasets [1, 2, 37]. Those 3 dataset contains around 1k images each. We first trained leveraging on the well looking one [1]. After several test, once found a good hyperparameters configuration for the model, we started some tests exploiting the other two datasets. The results are shown in 10 and they underline the difference in training with different dataset. The mAP50-95 on the validation set is much more higher with our chosen dataset [1] wrt the discarded datasets [2, 37].

5. Results

5.1. Models results

Style and Genre recognition. We present two experiments focused on style and genre recognition. The first aims to identify which experimental setups lead to our performance. The second compares various models and their variations on this task.

- Tab. 2 aim to show how different processing steps affect performance. The baseline is a vanilla EfficientNetV2-L.
- Tab. 3 shows different classification performance while changing the backbone. We tested with both a semantical feature extraction (DINO) and a more class-driven feature extraction (EfficientNet). The baseline is a Multi-head ResNet-18.

5.2. Case Study Evaluation

We express our sincere gratitude to the Galleria Estense in Modena for their support in testing our pipeline within their rooms. We had a chance of testing the whole project inside a real scenario and we tested on a large variety of paintings of various shapes, dimensions, lighting conditions, and represented figures. Below we present the results obtained during the test through pictures. In addition, there is a video

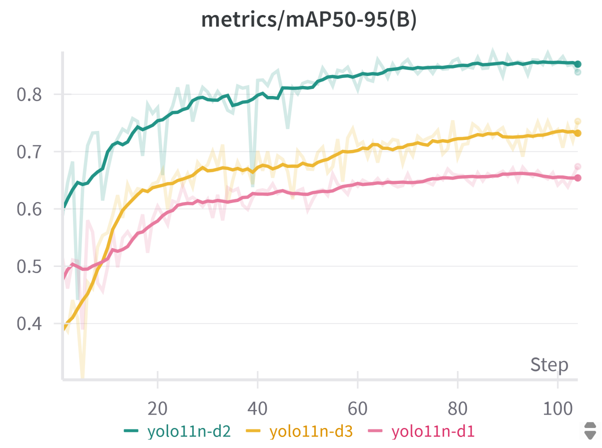


Figure 10. This figure shows the mAP50-90 across training on different datasets.

where you can better see the results, and the 1 section for the observations.

6. Conclusion

In this paper, we introduced VMART (Visual Museum Augmented Reality Tour), a new mixed reality application designed to improve how people explore art galleries using computer vision. Our work shows that by combining object detection, accurate image adjustments, feature recognition, and classification, we can create an engaging and informative museum visit.

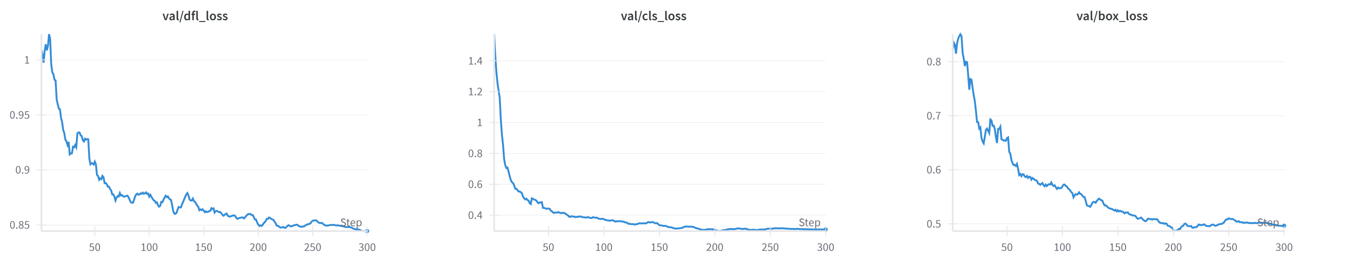
Our main achievements include an optimal YOLO detection, followed by a process that adjusts perspective. We also used DINOv2 to recognize paintings users have seen before and to find similar artworks from a large dataset. A key part of our project also involved training an EfficientNetV2-L model to discriminate paintings by style and genre. Testing at Galleria Estense in Modena was very helpful. It confirmed that our system works well in real museum settings, even with different painting shapes, sizes, and lighting.

While VMART shows good results, we note some limitations. The pipeline results generally fast; it can be affected by quick headset movements, sometimes causing the detected boxes to shift slightly. The method that uses pre-scanned room data, result in a smooth experience but requires an initial setup.

For future work, we plan to tackle these limitations. We also aim to consent the user to have a direct role within the classification and the retrieval by adding direct interaction through natural language instructions.

Variant	Style			Genre		
	α(%)	P (%)	R (%)	α(%)	P (%)	R (%)
Baseline	57.4	-	-	66.3	-	-
Dropout	60.5	-	-	67.2	-	-
Dropout, weight decay	64.2	64.2	64.2	70.1	69.6	70.0
Dropout, weight decay, finetune	61.2	-	-	70	-	-
Dropout, weight decay, finetune, lr scheduler	64.1	65.2	64.1	71.3	71.5	71.3
EfficientNetV2-L (Our)	**65.6**	**66.3**	**65.6**	**80.5**	**80.9**	**80.5**

Table 2. Different test of 2 head EfficientNet training



(a) Distributed Focal Loss

(b) Classification Loss

(c) Box Loss

Figure 11. YOLO validation results.

Variant	Style α (%)	Genre α (%)
Baseline	59.9	69.0
x2 EfficientNetV2-L	**66.4**	**81.9**
x2 DINOv2 cls	55.4	68.4
x2 DINOv2 patch	56.1	68.8
x2 DINOv2 patch + cls	57.3	67.9
EfficientNetV2-L (Our)	65.6	80.5

Table 3. Genre and Style accuracy comparison between different models. Our proposal is an EffienctNetV2-L with two heads.

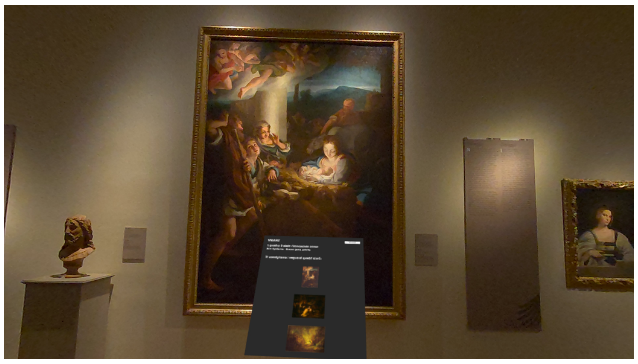


Figure 13. Photo taken at Galleria Estense

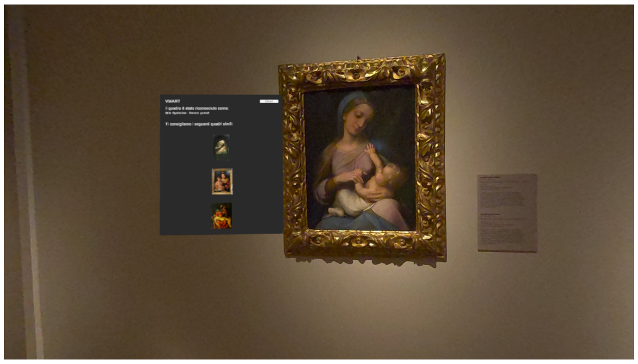


Figure 12. Photo taken at Galleria Estense

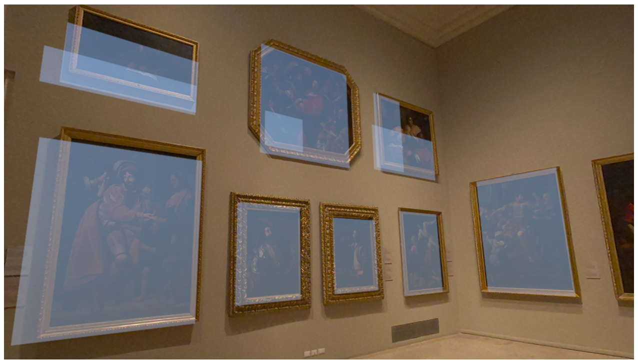


Figure 14. Photo taken at Galleria Estense

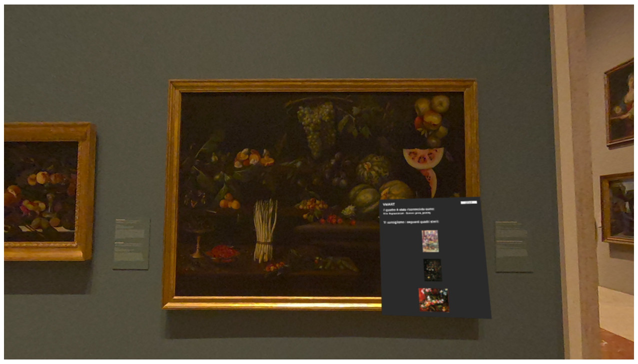


Figure 15. Photo taken at Galleria Estense

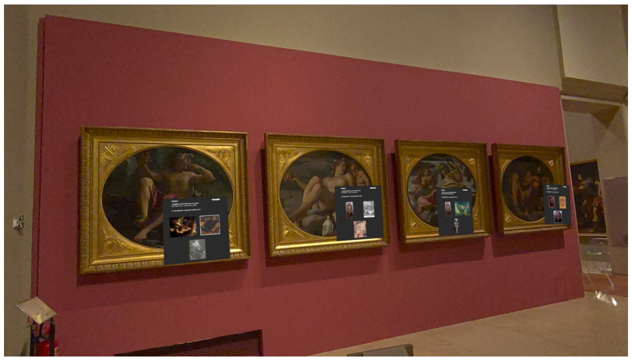


Figure 19. Photo taken at Galleria Estense

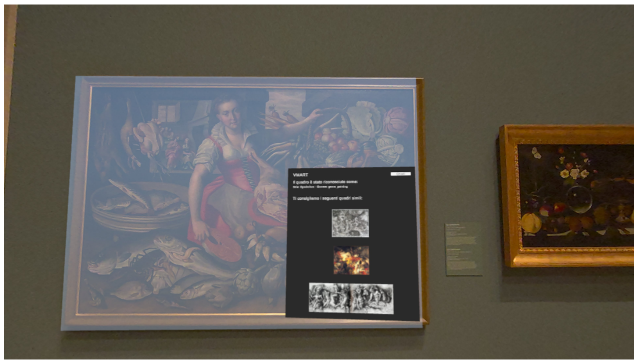


Figure 16. Photo taken at Galleria Estense

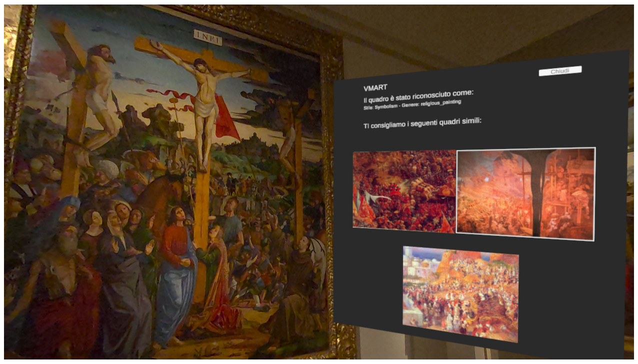

Chiudi
VMART
Il quadro è stato riconosciuto come:
Stile: Symbolism - Genere: religious_painting
Ti consigliamo i seguenti quadri simili:

Figure 20. Photo taken at Galleria Estense

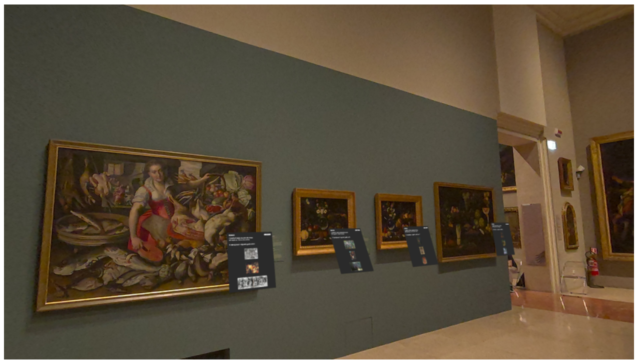


Figure 17. Photo taken at Galleria Estense

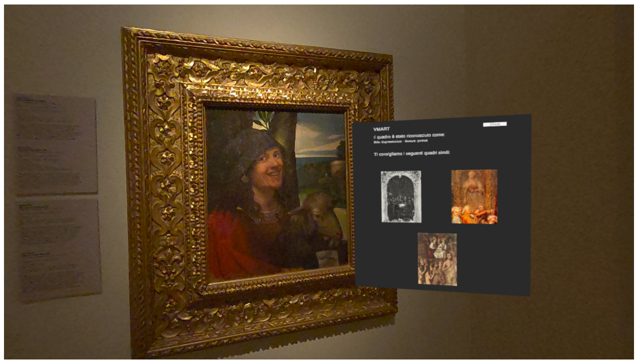


Figure 21. Photo taken at Galleria Estense

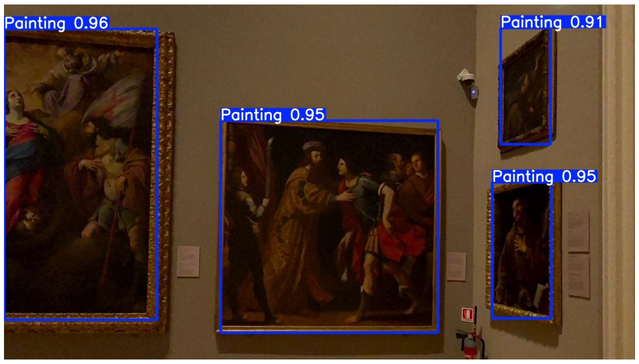


Figure 18. Photo taken at Galleria Estense

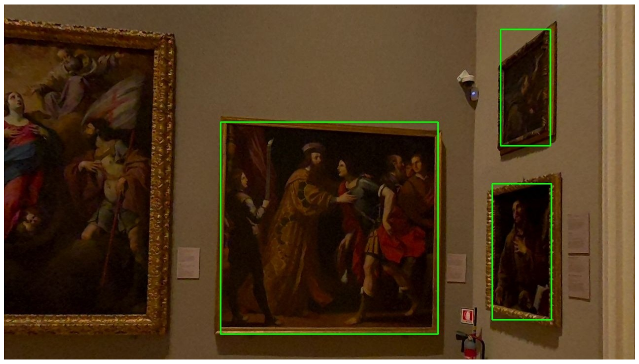


Figure 22. Photo taken at Galleria Estense

References

[1] Artera. Painting border detection dataset, 2024. 5, 9

[2] ArtSee. Paintingdetection dataset, 2024. 5, 9

[3] John Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(6):679–698, 1986. 4

[4] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers, 2020. 1, 2, 5

[5] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers, 2021. 1

[6] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009. 2

[7] Ivan Dokmanic, Reza Parhizkar, Juri Ranieri, and Martin Vetterli. Euclidean distance matrices: Essential theory, algorithms, and applications. *IEEE Signal Processing Magazine*, 32(6):12–30, 2015. 7

[8] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2021. 1

[9] Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. The faiss library. 2024. 6

[10] Richard O. Duda and Peter E. Hart. Use of the hough transformation to detect lines and curves in pictures. *Commun. ACM*, 15(1):11–15, 1972. 4

[11] Martin A. Fischler and Robert C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, 1981. 6

[12] Ross Girshick. Fast r-cnn, 2015. 2, 5

[13] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation, 2014. 2, 5

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015. 1, 2, 7

[15] Yiyu Hong and Jong-Weon Kim. Art painting detection and identification based on deep learning and image local features. *Multimedia Tools and Applications*, 78, 2019. 2

[16] Jan Hosang, Rodrigo Benenson, and Bernt Schiele. Learning non-maximum suppression, 2017. 4

[17] John Illingworth and Josef Kittler. A survey of the hough transform. *Computer vision, graphics, and image processing*, 44(1):87–116, 1988. 4

[18] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Faiss-billion-scale similarity search with GPUs. *IEEE Trans. on Big Data*, 7(3):535–547, 2019. 6

[19] Rahima Khanam and Muhammad Hussain. Yolov11: An overview of the key architectural enhancements, 2024. 2, 5

[20] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017. 5

[21] Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Joan Puigcerver, Jessica Yung, Sylvain Gelly, and Neil Houlsby. Big transfer (bit): General visual representation learning, 2020. 2

[22] Alfirna Rizqi Lahitani, Adhistya Erna Permanasari, and Noor Akhmad Setiawan. Cosine similarity to determine similarity measure: Study case in online essay assessment. In *2016 4th International Conference on Cyber and IT Service Management*, pages 1–6, 2016. 7

[23] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft coco: Common objects in context, 2015. 2

[24] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. *SSD: Single Shot MultiBox Detector*, page 21–37. Springer International Publishing, 2016. 5

[25] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016. 5

[26] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts, 2017. 2

[27] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019. 5

[28] J. Matas, C. Galambos, and J. Kittler. Robust detection of lines using the progressive probabilistic hough transform. *Computer Vision and Image Understanding*, 78(1):119–137, 2000. 4

[29] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023. 1, 2, 6

[30] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection, 2016. 1, 2, 5

[31] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks, 2016. 2, 5

[32] Sebastian Ruder. An overview of gradient descent optimization algorithms, 2017. 5

[33] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2015. 1

[34] Leslie N Smith. Cyclical learning rates for training neural networks. In *2017 IEEE winter conference on applications of computer vision (WACV)*, pages 464–472. IEEE, 2017. 2

[35] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014. 5, 7

[36] Harald Steck, Chaitanya Ekanadham, and Nathan Kallus. Is cosine-similarity of embeddings really about similarity? In *Companion Proceedings of the ACM Web Conference 2024*, page 887–890. ACM, 2024. 7

[37] Vlad Stoenescu. Paintings dataset, 2024. 5, 9

[38] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions, 2014. 1

[39] Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks, 2020. 2

[40] Mingxing Tan and Quoc V. Le. Efficientnetv2: Smaller models and faster training, 2021. 2

[41] Wei Ren Tan, Chee Seng Chan, Hernan Aguirre, and Kiyoshi Tanaka. Improved artgan for conditional synthesis of natural image and artwork. *IEEE Transactions on Image Processing*, 28(1):394–409, 2019. 2, 7

[42] C. Tomasi and R. Manduchi. Bilateral filtering for gray and color images. In *Sixth International Conference on Computer Vision (IEEE Cat. No.98CH36271)*, pages 839–846, 1998. 4

[43] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023. 1, 5, 8

[44] Enze Xie, Jian Ding, Wenhai Wang, Xiaohang Zhan, Hang Xu, Peize Sun, Zhenguo Li, and Ping Luo. Detco: Unsupervised contrastive learning for object detection, 2021. 2

[45] Wentao Zhao, Wei Jiang, and Xinguo Qiu. Big transfer learning for fine art classification. *Computational Intelligence and Neuroscience*, 2022:1–19, 2022. 2

[46] Zhengxia Zou, Keyan Chen, Zhenwei Shi, Yuhong Guo, and Jieping Ye. Object detection in 20 years: A survey. *Proceedings of the IEEE*, 111(3):257–276, 2023. 2

VMART: Rethinking Art Gallery Experience through Painting Detection, Classification and Retrieval

Supplementary Material

7. Additional Observations on Galleria Estense

Following extensive testing at the Galleria Estense, we present our observations and encountered issues.

The application on the viewer device is designed to continuously transmit data streams to a server, which then processes the data and responds with detected paintings. While this functionality generally performs well, the object detection system occasionally misidentifies other elements, such as screens, posters, or windows, as paintings. Consequently, most tests were conducted by toggling the connection and sending segment data in a controlled environment.

The Scene API was employed as a supplementary tool to enable asynchronous placement of paintings; however, it exhibited limitations. The primary challenge stems from the Scene API's design, which is optimized for small, domestic environments. The scale of the gallery pushed its capabilities to their limits:

- The gallery walls are notably high, and during room mapping, the VR system consistently truncated the mapped area at a maximum height from the floor.
- The room mapping system is intended to map an entire floor, allowing for room additions once doors are recognized. However, the large doors within the gallery were not consistently recognized, preventing us from mapping the entire floor plan.
- In larger rooms, the system would occasionally cease mapping after covering only a portion of the room. Subsequent manual adjustments were required to complete the mapping process.

Data acquisition is performed through the left eye's perspective, which does not precisely correspond to the actual view through the headset. Furthermore, the acquired images possess a narrower Field of View (FOV) compared to the real cameras.

Irregular shapes or highly detailed frames may not be consistently recognized.

For distant paintings, the placement of the recognition layer might be slightly shifted relative to the painting's true center.

DINO rarely exhibit not correct behaviors in re-identification phase; in 24 DINO doesn't recognize the already identified paintings and tells the visor that he found a new painting, or in 25 DINO doesn't discriminate the two painting that are almost identical in semantic.

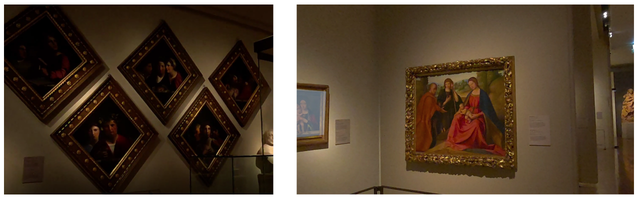


Figure 23. Paintings not detected

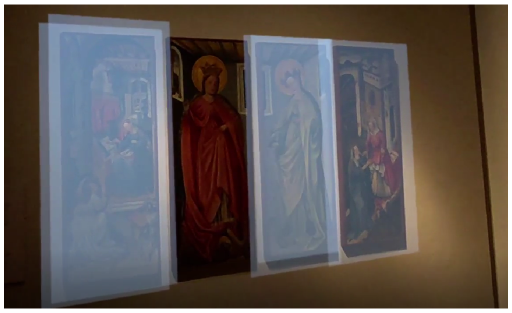


Figure 24. Paintings detected multiple times

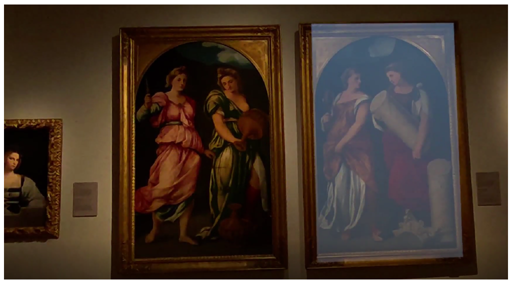


Figure 25. Case of not correct re-identification

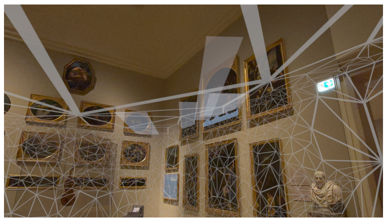


Figure 26. Ceiling problem