

Università degli Studi di Salerno

Corso di Laurea Triennale in Informatica

Image Recognition and Augmented Reality for Painting Narration

Relator: prof. Andrea F. Abate

Candidate: Michelangelo Esposito – 0512104784

Academic year 2019/2020

**Index**

[1. Introduction 3](#_Toc43936494)

[2. Image Recognition Using Convolutional Neural Networks 4](#_Toc43936495)

[2.1 A brief overview of Computer Vision 4](#_Toc43936496)

[2.2 Deep Learning and Convolutional Neural Networks 5](#_Toc43936497)

[3. Augmented Reality 8](#_Toc43936498)

[3.1 Overview 8](#_Toc43936499)

[3.2 AR Hardware 9](#_Toc43936500)

[3.3 Tracking approaches and algorithms 9](#_Toc43936501)

[4. Application Overview 12](#_Toc43936502)

[4.1 Requirements 13](#_Toc43936503)

[4.1.1 Use case model 15](#_Toc43936504)

[4.1.2 User interface mock-ups 17](#_Toc43936505)

[5. Application design 18](#_Toc43936506)

[5.1 Architecutre overview 18](#_Toc43936507)

[5.2 Android client 20](#_Toc43936508)

[5.3 Node.js Server 23](#_Toc43936509)

[5.4 Django Server 23](#_Toc43936510)

[5.5 Testing and Case study 24](#_Toc43936511)

[5.5.1 Painting analysis 24](#_Toc43936512)

[6. Bibliography 25](#_Toc43936513)

# Introduction

When visiting a museum or an art gallery, inspecting a painting is an activity limited to a quick visual analysis and, in some cases, to a small description found on a plate; for the most part, the history behind a piece and some of its details get lost and forgotten. A guide can help enhancing the discovering experience, however guided tours are not always available, as in the case of small private galleries, or they often provide long routes to which the visitor may not be interested in.

The purpose of this thesis work is to identify alternative support tools for the analysis and the fruition of art pieces, focusing on paintings. The main question that arises is the following: What technologies fit this purpose and are capable of delivering a smart and flexible system that can be used by a wide range of users?

In order to answer this question, we realized a mobile application which magnifies the educational experience provided by a painting, using Artificial Intelligence and Augmented Reality techniques. Through the usage of a convolutional neural network, the implemented system can recognize a piece in a scene and then generate a virtual augmented guide, which will start narrating the painting to the user, while projecting its details in the virtual environment. The application is accompanied by a visual interface … ; besides, the employment of Text-To-Speech technology in the narration strengthens the usability of the software for visually impaired or disabled users.

“The Birth of Venus”, by Italian renaissance artist Sandro Botticelli was chosen as a case study for the experimentation of the application; such a piece lends itself perfectly to the purpose, given the rich history that characterizes it and the clear spatial separations of characters and elements in the scene depicted. It was therefore possible to verify the applicability of the used technologies, in particular relatively to the training mechanisms of the neural network, for which a limited number of samples produced a satisfactory result.

Although this work has been focused on the recognition of two-dimensional pictorial pieces, with few modifications, the system is able to operate on three-dimensional artefacts.

The realized software, following an appropriate reengineering, could also be integrated as a service into other systems: a museum application, for example, could provide the use of the system following the purchase of a ticket or as a reference for specific targeted tours.

In the first chapter, an overview of computer vision and its numerous branches is provided. The focus will then shift towards Image Recognition, the technique used to identify the presence of a painting in a scene.

In the second chapter we

In the third and final chapter, the design of the application along with its core functionalities will be discusses. We will be ending with a cue on the possible evolutions and integrations of the developed system.

# Image Recognition Using Convolutional Neural Networks

Image Recognition is a computer vision technique that allows machines to interpret and categorize what they “see” in images or videos. Recognizing image patterns and extracting features is often the initial step of more complex computer vision techniques, like object detection or image segmentation. There are, however, various standalone applications that make the technique an essential machine learning task and the employment of neural networks has become the state-of-the-art approach for image recognition.

A model is trained to take an image as the input and output a target class, a set of labels describing the image. Along with a predicted class, a model may also output a confidence score related to how certain it is that an image belongs to a class. The technique can be broken into two separate branches: single and multiclass recognition; in single class image recognition, a model, or binary classifier, predicts only one label per image. On the other hand, multiclass models can assign several labels to a single image, outputting a confidence score for each one.

Nearly all image recognition models begin with an encoder, which is made up of blocks of layers that learn statistical patterns in the pixels of images that correspond to the labels they are attempting to predict. The encoder is then typically connected to a fully connected or dense later, that outputs confidence scores for each possible label.

## A brief overview of Computer Vision

As humans, the perception of what surrounds us happens daily and with relative ease: recognizing the physical properties of our world, such as the color of an object, the texture of a surface or the translucency of a container, becomes an unchallenging activity quickly.

An indisputably harder ability, such as deriving context information from an environment, is still a capacity we develop early on in our life.

Computer Vision is a scientific field that deals with how computers can gain high-level understanding from digital images or videos.

In the last decades, impressive milestones have been reached in various fields: in medicine, segmentation of brain tumor has high clinical relevance for the estimation of the volume and spread of a tumor and skeleton segmentation techniques have been able to provide a fast and reliable 3D observation of fractured bones; in the security industry, CV techniques such as real-time face recognition or object detection, combined with biometry are able to provide an easier control over entire buildings.

“However, despite all of these advances, the dream of having a computer interpret an image at the same level as a two-year old remains elusive.”[1]

Why must an “intelligent” machine resort to physics-based and probabilistic models to disambiguate between different possible solutions, when describing the world and reconstructing its properties is such an effortless task for humans?

The problem is based both on the still limited understanding of biological vision and on the complexity of vision perception in a dynamic and nearly infinite varying physical world.

“Perceptual psychologists have spent decades trying to understand how the visual system works and, even though they can devise optical illusions to tease apart some of its principles, a complete solution to this puzzle remains elusive.”

…

Computer Vision can be decomposed in a set of branches,

## Deep Learning and Convolutional Neural Networks

Generally speaking, an Artificial Neural Network is an algorithm designed to recognize patterns in

data and group them together; it is based on a collection of connected units or nodes, called artificial neurons, which can receive signals, process them and then signal the other neurons connected to them through connections called edges. Each neuron and edge can have an adjustable weight, which increases or decreases the strength of the signal received;

The original goal of the ANN approach was to solve problems in the same way that a human brain would, however over time the focus shifted to performing specific tasks in various fields, such as computer vision, speech recognition, machine translation or medical diagnosis.

Formally, “A neural network is a sorted triple *(N, V, w)* with two sets *N*, *V* and a function *w*, where *N* is the set of neurons and *V* a set *{(i, j)* | *i, j ∈ N}* whose elements are called connections between neuron i and neuron j. The function *w: V → R* defines the weights, where *w((i, j))*, the weight of the connection between neuron i and neuron j, is shortened to *wi,j*. Depending on the point of view it is either undefined or 0 for connections that do not exist in the network.”[2]

Looking at a neuron *j* and at its connected neurons, the propagation function of *j* is defined as the function used to transport values through *j* …; it usually receives the outputs *oi1*, …, *oin* of other neurons *i1*, ..., *in* and transforms them in consideration of the connecting weights *wi,j*into the network input net*j* that can be further processed by the activation function. The network input is thus the result of the propagation function.

“Let I = {*i1*, *i2*, …, *in*} be the set of neurons, such that … . Then the network input of j, called net*j* is calculated by the propagation function *fprop* as follows:"

net*j* = *fprop* (*oi1*, …, *oin, wi1, j*,…, *win, j*) = ∑ i ∈I (*oi . wi,j*)

The activation state of a neuron indicates the extent of a neuron’s activation and is often referred as just activation. Its formal definition is as follows: “Let *j* be a neuron. The activation state *aj*, in short activation, is explicitly assigned to *j*, indicates the extent of the neuron’s activity and results from the activation function”.

Each neuron may also have a threshold such that a signal is sent only if the aggregate signal crosses the threshold: “Let j be a neuron. The threshold value Θj is uniquely assigned to j and marks the position of the maximum gradient value of the activation function.”

“Let *j* be a neuron. The activation function is defined as:

*aj*(*t*) = *fact* (net*j*(*t*), *aj*(*t – 1*), Θj)

It transforms the network input netj, as well as the previous activation state aj(t – 1) into a new activation state aj(t), with the threshold value Θ playing an important role, as already mentioned.”

The activation function is usually defined globally, for all neurons, and only the threshold values are different for each neuron and can be adjusted by a learning procedure

…

Usually neurons are grouped into layers, each one performing a different transformation on their input; signals travel from the first layer (input layer) to the last (output layer), with the possibility of traversing the layers multiple times, depending on the classification of the network.

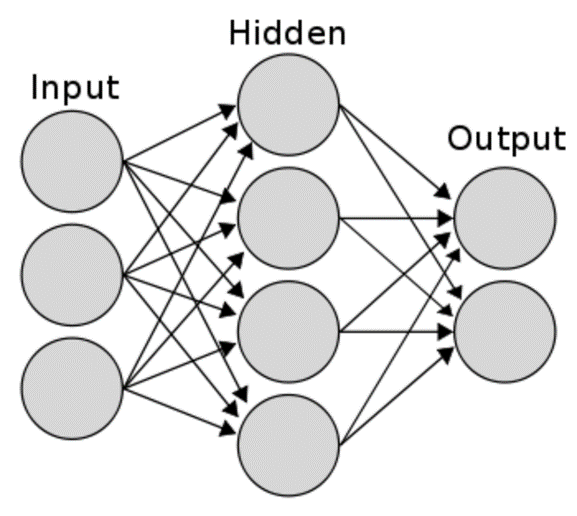


Figure 1 A typical two-layer neural network. Input layer does not count as the number of layers of a network

In a feedforward network, the signal can only travel forward and after a transformation is performed, the new values become the input values of the next layer; they are often used in data mining problems. A feedback network, on the other hand, has feedback paths, which allow the signals to travel in both directions using loops; they are often used in optimization problems, where the network looks for the best arrangement of interconnected factors.

What happens inside a neuron? The input node receives a number representing the information, which is presented as an activation value, where each node is given a number; higher the number, the greater the activation. Based on the connection strength and transfer function, the activation value passes to the next node. Each of the nodes sums the activation values that it receives, by calculating a weighted sum, and modifies that sum based on its transfer function. Then an activation function is applied, used by the neuron to understand if it needs to pass along a signal or not. The activation runs through the network until it reaches the output nodes, which return the information in a way a human interpreter can understand. The network will then use a cost function to compare the output with the expected result

(cfr. A. Bonner, The Guide to Deep Learning: Artificial Neural Networks).

Deep Learning architectures suitable for image recognition are based on variations of Convolutional Neural Networks (CNNs). A CNN is a Deep Learning algorithm which can take an input image, assign importance to various aspects/objects in it (learnable weights and biases) and be able to differentiate one from the other. While in primitive methods filters are hand-engineered, with enough training, CNNs have the ability to learn these filters/characteristics.

The architecture of such a network is analogous to the connectivity patterns of human neurons and was inspired by the organization of the visual cortex, where individual neurons respond to stimuli only in a restricted region of the visual field, known as receptive field; a collection of such fields overlap to cover the entire visual area.

The first part of the CNN consists of convolutional and max-pooling features extractor layers, while the second part consists of the fully connected layer which performs non-linear transformations of the extracted features and acts as the classifier. If the neurons in the convolutional layer find the features they are looking for they produce a high activation.

In image processing, to calculate convolution at a particular location (x, y), a k x k sized chunk, the kernel, is extracted from the image, centred at location (x ,y); the values in this chunk are then multiplied element-by-element with the convolution filter, also sized k x k, and then they are added together to obtain a single output.



Figure 2 Convolution operation example (V. Gupta, Image Classification using Convolutional Neural Networks in Keras)

Storing an image means keeping track of the color information associated to each individual pixel in a color matrix; the size of each pixel depends on the color depth (8-16-24 bit). Once images reach a notable dimension, calculations can get very intensive, so the role of the CNN is to reduce the images into a form easier to process, without losing features which are critical for prediction sake.

This is achieved by a max pooling layer, which is responsible of reducing the spatial size of the image (not the depth); this reduces the number of parameters, avoiding overfitting, the condition when a trained model learns too much out of the training data and loses the ability to generalize.

A common form of pooling is max pooling where a filter of size p is taken and the maximum operation over the sized part of the image is applied.



Figure 3 Max pooling example (V., Image Classification using Convolutional Neural Networks in Keras)

The fully connected layer is made up of an ANN, which purpose is to combine the detected features into more attributes, in order to predict the classes with greater accuracy.

Many neural network architectures[[1]](#footnote-1) exist for image recognition, including:

* AlexNet: deep neural network winner of the ImageNet classification in 2012; it’s widely credited with sparking a resurgence of interest in using deep convolutional neural networks to solve computer vision problems. The network is relatively large, with over 60 million parameters and many internal connections, thanks to dense layers that make the network quite slow to run in practice.
* VGGNet: network developed by researchers from the Visual Geometry Group (VGG) at Oxford University. VGGNet has more convolution blocks than AlexNet, making it “deeper”, and it comes in 16- and 19-layer varieties, referred to as VGG16 and VGG19, respectively.

# Augmented Reality

## Overview

Augmented Reality is a relatively new technology which is focused on the blending of digital elements, often interactive, such as visual overlays or 3D models and animations, into real-world environments. An AR experience

… and it can involve multiple sensory modalities, including visual, auditory, haptic, somatosensory and olfactory. With the help of advanced AR technologies (e.g. adding computer vision, incorporating AR cameras into smartphone applications and object recognition) the information about the surrounding real world of the user becomes interactive and digitally manipulated; this information can be virtual or real, e.g. seeing other real sensed or measured information.

Compared to Virtual Reality, where the user’s perception of reality is completely based on virtual information, Augmented Reality is considered an example of “Mixed Reality”, since it incorporates both real world and fictional elements. For example, in architecture, VR can be used to create a walk-through simulation of the inside of a new building, while AR can be employed to show a building’s structures and systems super-imposed on a real-life view.

The term “Mixed Reality” was introduced by Paul Milgram in a 1994 publication called “Augmented Reality: A class of displays on the reality-virtuality continuum”; this paper presented the idea that between virtual and real there is a spectrum of different mixtures of both.



Figure 4 Milgram et al., The Reality-Virtuality Continuum

The term Mixed Reality is introduced as follows:

“Within this framework it is straightforward to define a generic Mixed Reality (MR) environment as one in which real world and virtual world objects are presented together within a single display, that is, anywhere between the extrema of the RV continuum”.

Today there are applications available for or being researched for AR in nearly every field, including archaeology, art, medicine, military industry and entertainment; it is expected that even more potential areas of application are soon to be rising.

In military training, the personnel is often equipped with HMDs (head-mounted displays)

… removing real objects from a scene.

Although some impressive milestones and goals have been reached in recent years, due tue the technology still being young, a lot of areas still require further research before the employment of highly-reliable AR systems.

## AR Hardware

In order to take advantage of AR technology, terminal devices with a complex composition of hardware modules are necessary.

This was first achieved, to some extent, by a cinematographer called Morton Heilig in 1957. He invented the Sensorama, a machine capable to deliver visuals, sounds vibration and smell to the viewer; obviously it wasn’t computer controlled, however it was the first example of an attempt at adding additional depth to a visual experience.

The first actual AR system ever built was developed by Louis Rosenberg in 1992 at the USAF Armstrong Labs; it was called Virtual Fixtures and was a complex robotic system, designed to compensate for the lack of high-speed 3D graphics processing power in the early 90s. In order to create the augmented experience, a unique optics configuration was employed that involved a pair of binocular magnifiers aligned so that the user’s view of the robot arms were brought forward so as to appear registered in the same location of the user’s real arms.

Generally speaking, the hardware components that make up … are: a processor, a display, sensors and input devices. Modern mobile computing devices, such as smartphones and tablets contain these elements, which almost always include one or multiple cameras and Microelectromechanical systems (MEMS) sensors such as an accelerometer, GPS and solid-state compass, and are suitable AR platforms.

Motion tracking devices can be classified based on the technologies used for the measurements: mechanicals, electromagnetic, optic, acoustic and inertial; for each one there is a list of reference parameters:

* Work volume
* Sampling rate: the rate at which the device detects and updates variables;
* Resolution
* Latency: the time slot between an event and its recognition;
* Precision

## Tracking approaches and algorithms

When discussing the requirements of realistic and “plausible” tracking systems for AR applications, Fitzmaurice and Feiner focus on

“In Augmented Reality, virtual objects supplement rather than supplant the real world. Preserving the illusion that the two coexist requires proper alignment and registration of the virtual objects to the real world. Even tiny errors in registration are easily detectable by the human visual system”[3].

This contributes to the identification of a key measurement for AR systems: how realistically they integrate augmentations with the real world: the software must derive real world coordinates, independently of the camera used to move through the scene, and interpret the semantic context of the environment. This process is called image registration and is often performed in real time; it uses different methods of computer vision, mostly related to video tracking. Many computer vision methods of augmented reality are made up of two parts: in the first stage, interest points are detected in the camera images, by using feature detection methods such as corner detection, blob detection or edge detection, and other image processing methods. In the second stage the AR application generates a real-world coordinate system from the data obtained in the first stage.

When it comes to object tracking in AR, the distinction is usually made between two modes of tracking, known as marker and markerless.

When using the marker approach, the tracking is based on the recognition of specific figures (markers); from these, a reference point for the scene can be extracted and used to generate augmented content.

Markers must follow specific constraints:

* A marker must be perfectly squared, with well-defined proportions.
* The external borders must be well defined and continuous;
* The inner image must be completely asymmetric, so that different orientations cannot cause any ambiguity.
* Markers must be as loosely coupled as possible; this way, when the camera is far away from the image, which is represented by fewer pixels the greater the distance, it is still possible to distinguish it.



Figure 5 Comparison of some augmented reality fiducial markers for computer vision, ARTag, Wikipedia

With this technique, tracking becomes possible as long as the marker are in the camera’s field of view; additionally, any eventual noise in the camera stream can impact on the detection of the image.

Some workarounds exist for these limitations such as the multi-marker approach, where a cluster of markers is used to allow for a freer camera movement in the scene. With this method, the detection is more fault tolerant, since a detection error on one of the markers, caused by camera noise or lighting conditions, can be corrected by using the other markers

Since the usage of markers is not always possible due to physical limitations, markerless AR solutions have become an industry preference

The tracking becomes much more complex though: a combination of camera systems, dedicated sensors and math is employed to detect and map an environment in a very accurate way; by using a virtual map of the area, the AR application can then place and track objects in the scene.

, the markerless approach has recently become a preference for the industry: with this technique, in the development environment, each virtual product is assigned a coordinate value of longitude and latitude: when the system is installed on a mobile device, it can get the current location using the integrated GPS functionalities; the gyroscope is then used to determine the relative angle relationship and the camera is used to turn the display background into the real environment.

Due to the lack of the necessary hardware components, the markerless method is not supported in typical desktop environments,

Some hybrid approaches also exist: the combined usage of multi-marker systems along with gyroscopes and accelerometers dealing with camera movement can help … tracking errors when detection fails.

# Application Overview

The application aims to deliver an enhanced educational experience through the usage of Augmented Reality and Artificial Intelligence technology, in order to provide the user with additional information about a painting or help visually impaired people receive auditive aid. Anyone with a supported device can come across new ways to appreciate art and even discover new details in an evocative piece.

After launching the application, a user will have the ability to scan a painting by using their smartphone’s integrated camera; if the image is recognized, the system will then lookup its associated information and generate a virtual guide using AR. The guide will begin interacting with the user via Text-To-Speech, by narrating the details of the painting and projecting them in the augmented environment.

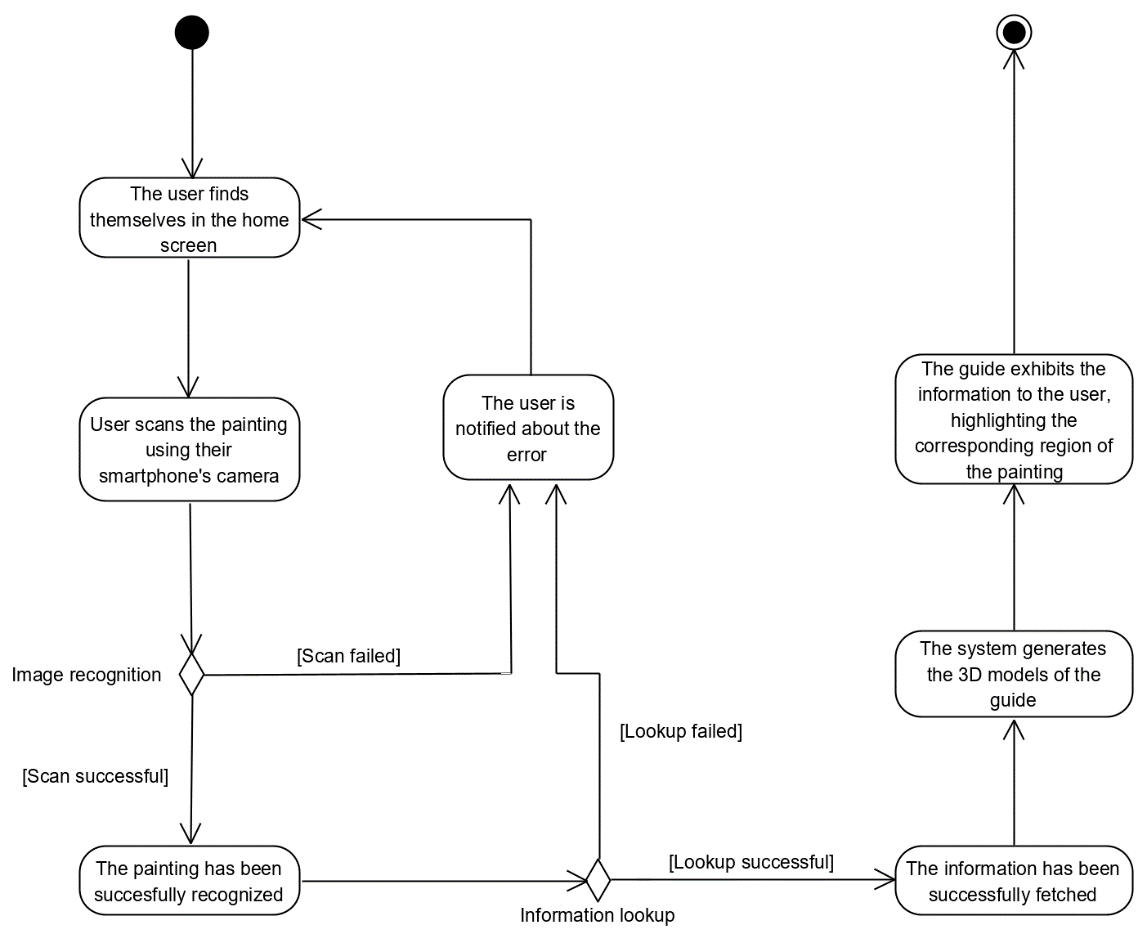


Figure 1 UML activity diagram depicting the interaction between a user and the system.

## Requirements

The system aims to achieve the following goals:

* Enhance the educational experience for students and art enthusiasts.
* Create a more interactive alternative to traditional museum guides.
* Aid visually impaired people experience art.

**Functional requirements**:

* **FR\_1**: The system must allow the user to scan a painting with the smartphone’s integrated camera and identify it.
* **FR\_2**: The system must be able to retrieve the information associated to a scanned painting.
* **FR\_3**: The system must be able to project an augmented reality guide in the environment.
* **FR\_4**: The system must be able to highlight the details on the painting by using Augmented Reality.
* **FR\_5**: The system must utilize Text-To-Speech technology when providing the user with the requested information.
* **FR\_7**: The system must allow an administrator to add a new painting and its related information to the archive.

**Non-functional requirements**:

**Usability**:

* **NFR\_U1**: The system’s interface must be easy to use and not ambiguous.
* **NFR\_U2**: The system must ensure operations are performed in the most direct way available.
* **NFR\_U3**: The system must include a user manual.

**Reliability**:

* **NFR\_R1**: The information provided by the system must always be reliable and consistent.
* **NFR\_R2**: The system must ensure any error message is delivered to the user in less than 3 seconds.

**Performance**:

* **NFR\_P1**: System response time must not exceed 300ms when performing lookup operations.
* **NFR\_P2**: Errors related to … must not exceed the threshold of x occurrences per month.

**Supportability**:

* **NFR\_S1**: The system must include an exhaustive documentation, in order to more easily support maintenance.

In order to illustrate the usage of the application at a very high abstraction level, a series of scenarios have been crafted; these highlight the sequence of action performed by different kinds of users in different environments, when the system is operational.

Each scenario is characterized by a name, a unique identifier, its participants and the flow of actions that make up the usage example.

|  |  |
| --- | --- |
| **Scenario name** | MuseumUsage |
| **Scenario ID** | SC\_1 |
| **Participants** | Ann: art student visiting the Salvador Dalí Museum in Figueres with her class. |
| **Flow of events** | 1. Ann is wandering into the halls of the museum when a particular piece catches her eye, Palladio’s Corridor of Thalia, so she gets closer and starts examinating it. 2. The girl isn’t satisfied with the little information provided by the plate next to the painting and wants to know more. 3. Ellie, one of Ann’s classmates suggests her …, to quench her thirst for more details. 4. Ann decides to give it a try, so she downloads the app and, after reading the manual, tries to scan the painting using her smartphone. 5. The system recognizes the painting and proceeds to generate an interactive talking head of the artist, in the space in front of Ann, using AR. 6. The artist then begins to narrate the story of the painting to Ann, in particular he tells Ann how the piece is heavily influenced by Italian Renaissance art. 7. Moving to the painting itself, the guide starts to speak about the strongly lit figure of a girl playing with a skipping rope in the top left, while the corresponding region is highlighted on the canvas. |

|  |  |
| --- | --- |
| **Scenario name** | HomeUsage |
| **Scenario ID** | SC\_2 |
| **Participants** | Frank: art enthusiast working from home during quarantine. |
| **Flow of events** | 1. After being stuck at home for more than three weeks, Frank decides to   ... his old photo books.   1. While browsing the pages, the man notices a picture of him next to “The Bedroom” by Vincent Van Gogh, snapped during a trip some years ago. 2. Frank decides to use … to try and get more info on the painting in the picture, so using his phone he scans it. 3. The system correctly recognizes it and proceeds to generate the guide, which begins narrating the details of the painting to Frank. |

|  |  |
| --- | --- |
| **Scenario name** | AidedUsage |
| **Scenario ID** | SC\_3 |
| **Participants** |  |
| **Flow of events** |  |

## Use case model

From the identified scenarios a more formal use case model has been constructed: this provides a series of use cases performed by the different users of the system.

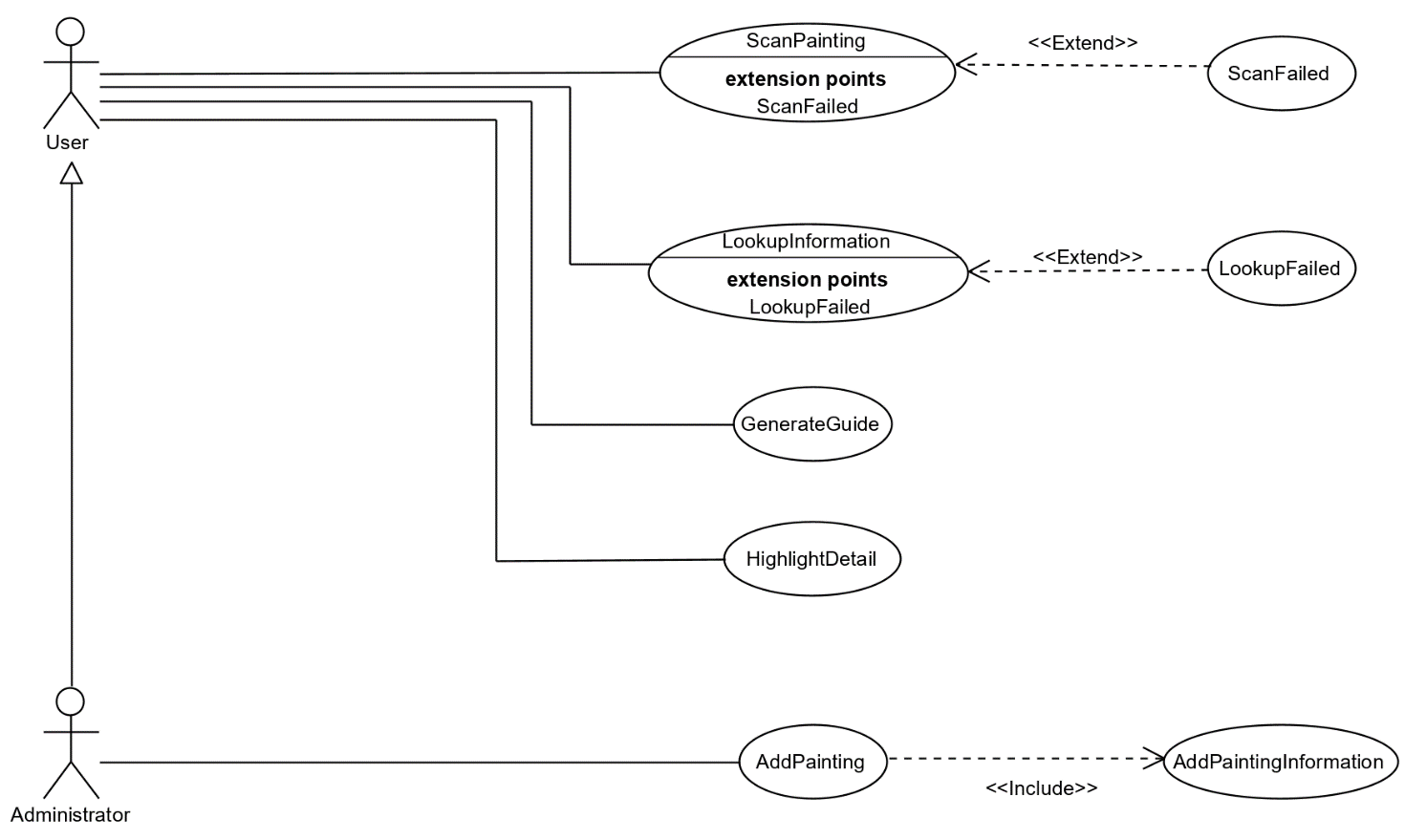


Figure 2 UML Use Case Diagram of the system

|  |  |
| --- | --- |
| **Use case name** | ScanPainting |
| **Use case ID** | UC\_1 |
| **Participating actors** | User |
| **Preconditions** | 1. The user finds themselves in the home screen |
| **Flow of events** | 1. The user points the camera towards the painting. 2. The system begins looking for a match of the painting. |
| **Postconditions** | The scanning process was successful and the user is waiting for the system to lookup the associated information. |
| **Exceptions** | * 1. The system cannot find any match for the painting   (Use case “ScanFailed” – UC\_1.1). |

|  |  |
| --- | --- |
| **Use case name** | ScanFailed |
| **Use case ID** | UC\_1.1 |
| **Participating actors** | User |
| **Preconditions** | The system cannot find any match for a painting. |
| **Flow of events** | 1. The system notifies the user about the error. |
| **Postconditions** | The user is redirect in the home screen of the application. |

|  |  |
| --- | --- |
| **Use case name** | LookupInformation |
| **Use case ID** | UC\_2 |
| **Participating actors** | User |
| **Preconditions** | A painting has been successfully scanned and the user is waiting for the system to lookup the associated information. |
| **Flow of events** | 1. The system retrieves the painting information in its internal archive. |
| **Postconditions** | The system is able to retrieve the requested information. |
| **Exceptions** | * 1. The system cannot find any information for the matched painting   (Use case “LookupFailed” – UC\_2.1). |

|  |  |
| --- | --- |
| **Use case name** | LookupFailed |
| **Use case ID** | UC\_2.1 |
| **Participating actors** | User |
| **Preconditions** | The system cannot find the information associated to a scanned painting. |
| **Flow of events** | 1. The system notifies the user about the error. |
| **Postconditions** | The user is redirect in the home screen of the application. |

|  |  |
| --- | --- |
| **Use case name** | GenerateGuide |
| **Use case ID** | UC\_3 |
| **Participating actors** | User |
| **Preconditions** | User has scanned a painting and the system has correctly recognized it and has retrieved its associated information. |
| **Flow of events** | 1. Based on the scanned painting’s author, the system retrieves the 3D model of the guide to display and projects it in the space pointed by the user. |
| **Postconditions** | The 3D guide has been successfully generated in the application. |

|  |  |
| --- | --- |
| **Use case name** | HighlightDetail |
| **Use case ID** | UC\_4 |
| **Participating actors** | User |
| **Preconditions** | The system has generated the virtual guide for a painting. |
| **Flow of events** | 1. The guide mentions a region of the painting. 2. The system highlights that region. |
| **Postconditions** | The virtual copy of the painting has been successfully generated in the application. |

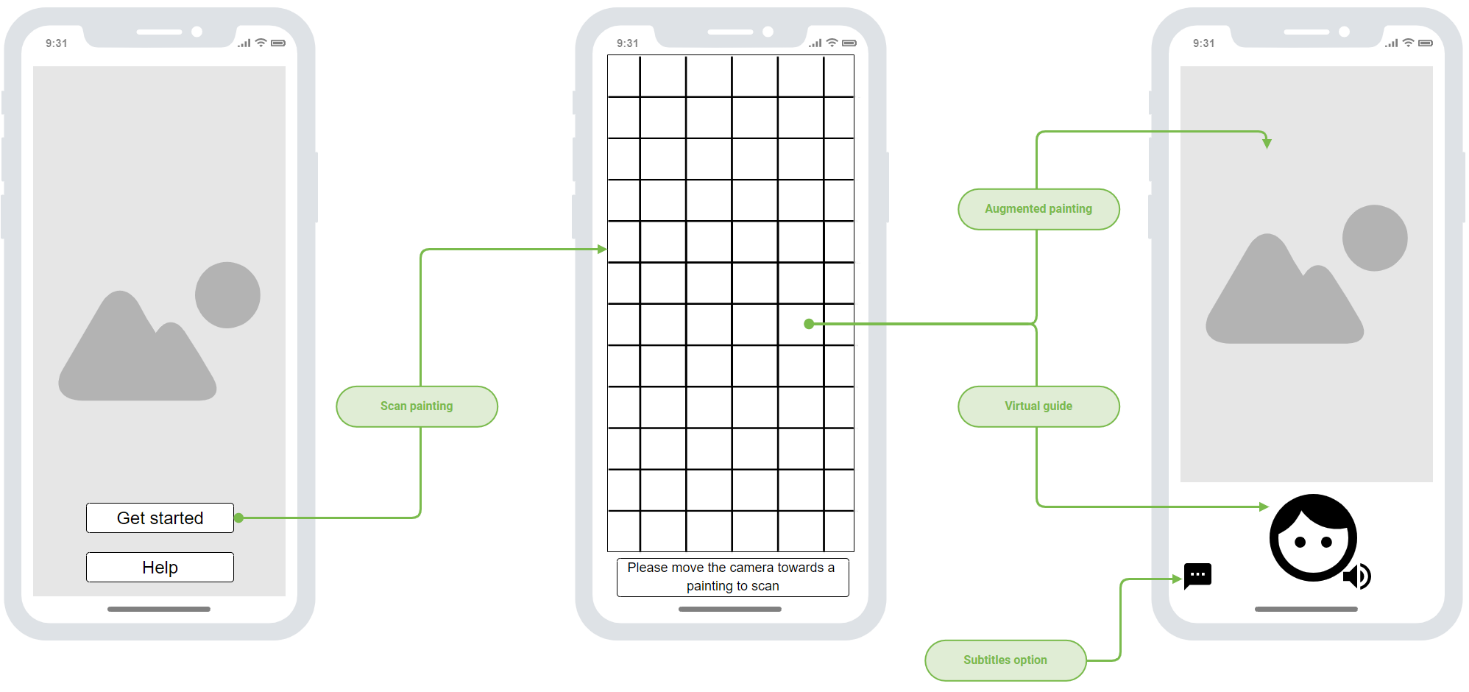
|  |  |
| --- | --- |
| **Use case name** | AddPainting |
| **Use case ID** | UC\_5 |
| **Participating actors** | Administrator |
| **Preconditions** | The administrator requests the form to add a new painting. |
| **Flow of events** | 1. The admininistrator compiles the form with the painting image and the information associated to the it (Use Case “AddPaintingInformation” – UC\_6) |
| **Postconditions** | The new painting has benn successfully added to the archive and will now be recognized when scanned. |

|  |  |
| --- | --- |
| **Use case name** | AddPaintingInformation |
| **Use case ID** | UC\_6 |
| **Participating actors** | Administrator |
| **Preconditions** | An administrator is adding a new painting to the archive |
| **Flow of events** |  |
| **Postconditions** | The |

## User interface mock-ups

According to

First UI idea:



# Application design

## Architecutre overview

Following the initial requirements analysis, a set of design goals for the application has been identified; these will act as a guide during the development, in order to ensure the implementation will not drift away from the original design views.

* **DG\_01: Accessibility**: Ensuring the application is targeted towards a wide spectrum of users is a high priority;
* **DG\_02: Reliability**: The information provided by the system must always be reliable.
* **DG\_03: Usability**: The application’s user interface must be simple and friendly to newcomers.

The system is based on the Model-View-Controller (MVC) architectural design pattern, where the functionalities are distributed into three components:

The Model is the central component of the pattern; it handles the data, logic and rules of the application, while being independent from the user interface. The View .Finally, the Controller component handles the user’s requestsm and redirects the control to the view

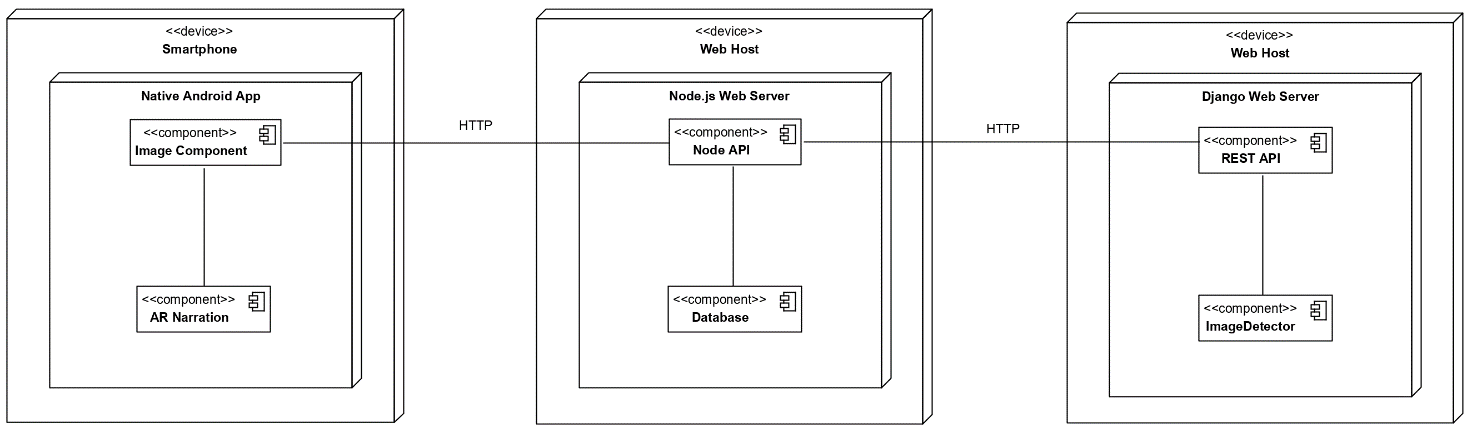
At the hardware level, a distinction between three devices is made: the user’s smartphones acts as the client side of the application: the Image component encapsulates the camera control and the network functionalities, while the AR Narration component is where the actual narration of the painting takes place.

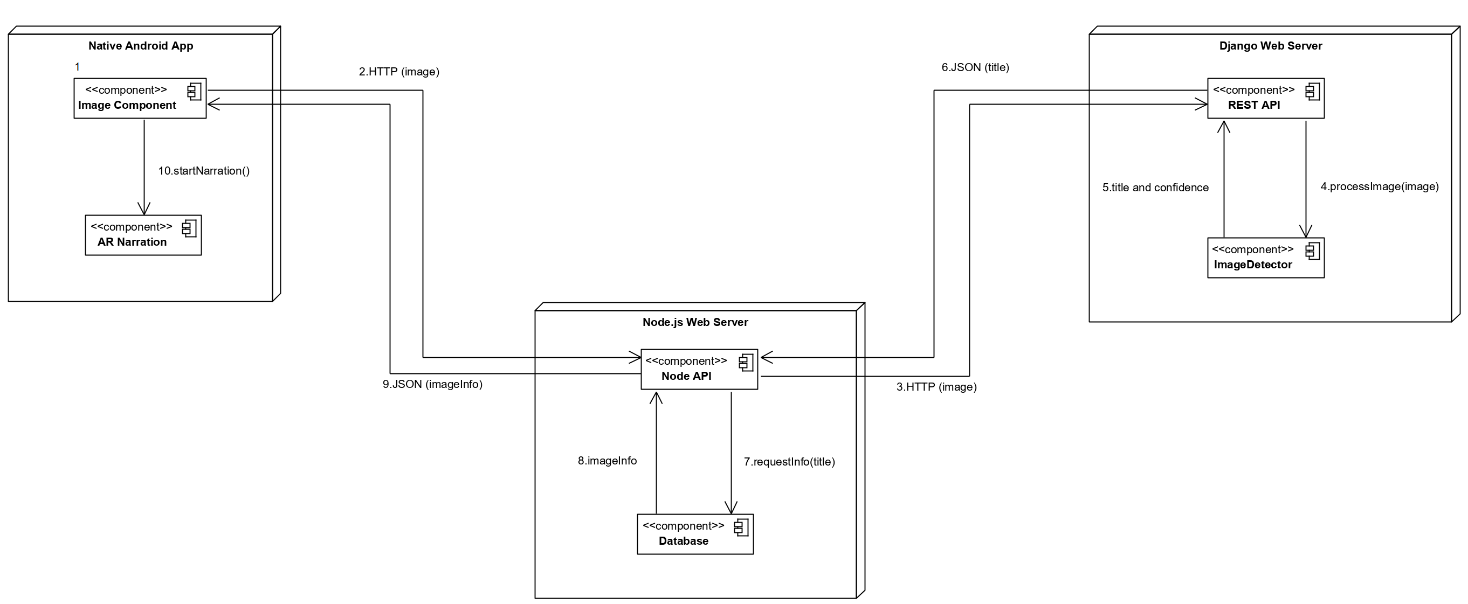
On one of the web hosts, the Node.js server acts as a middle-ground between the client and the image recognition component. The Node API component manages the network requests, whil the Database component handles the persistent information associated to each painting.

On the second web host, a REST API, realized via the djangoRestFramework module, is used to interface the Django AI server with the Node server. Here the neural network is encapsulated in the ImageDetector component, in its own class which provides the methods necessary to run a pre-trained inference graph on the received image and output a title and a confidence score. If this confidence score, expressed as the probability of a specific painting being present in the scene, is greater than a pre-set threshold, a positive feedback is sent back to the Node server.

The communication between the different devices happens via HyperText Transfer Protocol (HTTP) …

The two web servers could potentially be deployed onto the same machine; the reason they have been implemented as two distinct entities is just to keep a logic separation between the components.





Here a list of steps illustrating the processing of the image is presented:

1. User starts up the application and captures and image with the smartphone’s integrated camera.
2. The captured image is sent to the Node.js server, using an HTTP multipart form-data request.
3. Once received, the image is forwarded from the Node API to Django server, via the exposed REST API.
4. The Django server receives the image and processes it via the underlying neural network, in the ImageDetector component.
5. A title and a confidence score are output for the received image by the neural network. It then passes it to the REST API component.
6. Response is then sent back to the Node.js server as a JSON object containing the two fields.
7. The Node.js server retrieves the information associated to the identified painting from a MongoDB relational database.
8. Any retrieved information is sent to the Node API component.
9. The information is sent back to the Android client.
10. The control is finally sent to the AR Component, which proceeds with the narration of the painting.

## Android client

This section is dedicated to the client side of the system, a native Android application written in Java;

The AR functionalities are provided by the ARCore SDK, Google’s AR solution. By using various APIs, it allows smartphones to detect the surrounding environment, understand the world and interact with its information; ARCore is based on three key technologies to integrate the virtual content into the real world, as seen through the smartphone’s camera:

* Six degrees of freedom allows the device to understand and track its position relative to the world;
* With enviromental understanding, the divece can detect the size and location of surfaces;
* Light estimation allows the device to estimate the environment’s current lighting conditions.

When it comes to the handling of the 3D environment, the choice was between a fully-fledged game engine, like Unity or Unreal Engine, and an external Android library, such as Google’s Sceneform. We decided for the latter, since this allowed us to keep a higher control over the whole application development; furthermore, this would make future maintenance or additions easier. Sceneform . It makes it straightforward to render 3D scenes into AR and non-AR applications, without having to learn OpenGL; it includes:

* A high-level scene graph API;
* A realistic physically based rendered provided by Filament;
* An Android Studio plugin for importing, viewing and building 3D assets.

The painting information received from the Node.js server is stored in a Painting type object …

For handling the Text-To-Speech narration, the integrated android.speech.tts module has been used; since each TTS utterance is executed asynchronously, the narration works by setting up a custom UtteranceProgressListener for the TTS object and using two handlers to receive messages from the its thread during the utterance life cycle. The reason why the messaging is necessary in the first place, is because all operations influencing the augmented elements of the scene must be performed on the main thread on which the application is running.

The first handler, onTtsStartHandler, receives a message as soon as the speak() method on the TextToSpeech object has been called: it proceeds to generate the image corresponding to the segment which is being narrated and animates it forward in the environment.The second handler, onTtsDoneHandler, receives a message when an utterance has been successfully completed, without any errors, and it destroys the previously generated image node in the scene.

In order to provide a better user experience, we decided to implement user interface functionalities to navigate through the narration; these include a pause/resume button and two buttons to move to the previous and next narration segments. Since the native TTS module doesn’t provide any pause functionalities we found ourselves with two main alternatives:

1. Make the TTS object record the narration to an audio file, using the native synthesizeToFile() method, and then use a MediaPlayer object to control the narration;
2. On resume, make the narration start from the beginning of the current segment.

By using the first approach any future implementation of any kind of interaction mechanism would require a reengineering of the speech context. We decided to proceed with the second approach, by carefully dividing the segments into smaller, independent parts.

**Package structure**:

Bho

│

├───activities

│ ├───MainActivity

│ ├───ImagePreviewActivity

│ └───ArActivity

│

├───painting

│ ├───Painting

│ └───PaintingDetail

│

├───texttospeech

│ ├───TextToSpeechManager

│ └───CustomUtteranceProgressListener

│

├───network

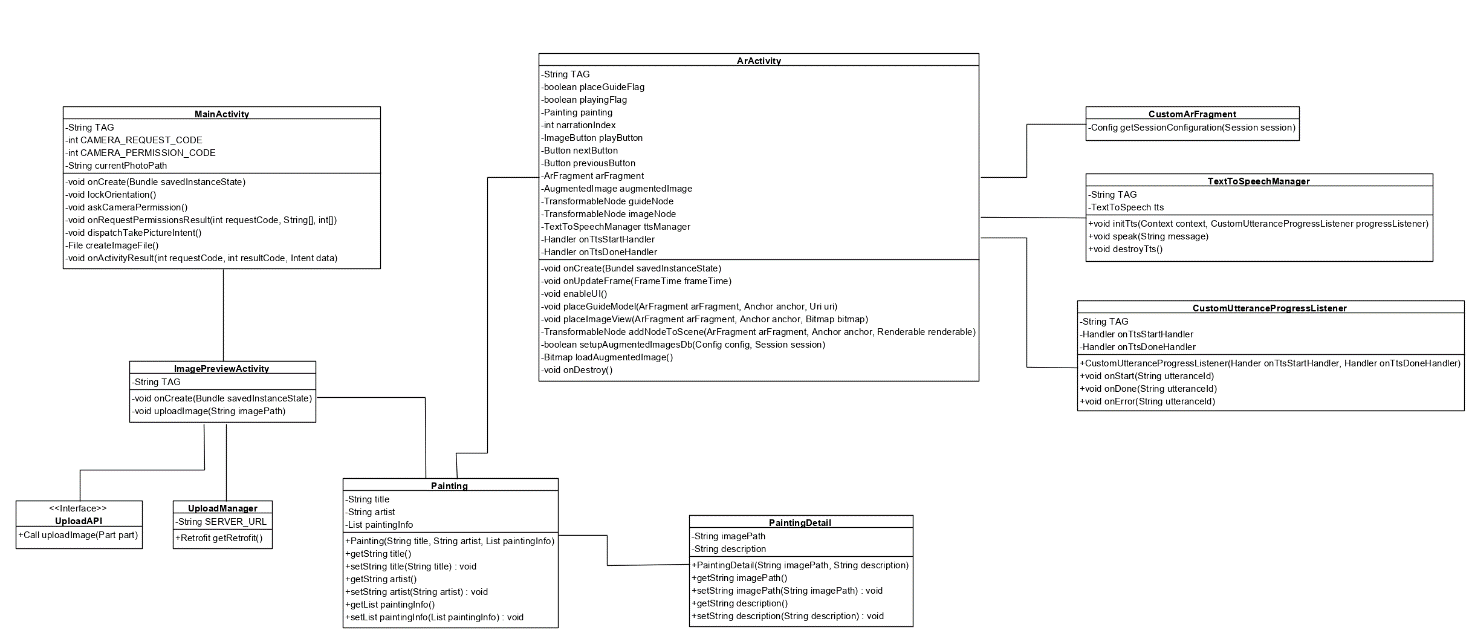
│ ├───UploadAPI

│ └───UploadHandler

│

└───CustomArFragmen

**Class diagram**:



**activities package**:

Package containg the Android activities of the application.

|  |  |
| --- | --- |
| Class | MainActivity |
| Description | The activity containing the home screen functionalities. It provides an onClickListener for the start button, which starts up device’s camera and, after a picture is captured, redirects the control to the ImagePreviewActivity. |
| Instance variables | private static final String TAG: String used for debug and alerting purposes.  private static final int CAMERA\_REQUEST\_CODE: integer code used when requesting the camera permission.  private static final int CAMERA\_PERMISSION\_CODE: integer code used as a comparison when requesting the camera permission. |
| Methods | void onCreate (Bundle savedInstanceState): |

|  |  |
| --- | --- |
| Class | ImagePreviewActivity |
| Description |  |
| Instance variables | String TAG: String used for debug and alerting purposes. |
| Methods | void onCreate (Bundle savedInstanceState): |

|  |  |
| --- | --- |
| Class | ArActivity |
| Description | String TAG: String used for debug and alerting purposes. |
| Instance variables |  |
| Methods | void onCreate (Bundle savedInstanceState): |

**network package**:

|  |  |
| --- | --- |
| Class | UploadHandler |
| Description |  |
| Instance variables |  |
| Methods |  |

|  |  |
| --- | --- |
| Class | UploadAPI |
| Description |  |
| Instance variables |  |
| Methods |  |

**painting package**:

|  |  |
| --- | --- |
| Class | Painting |
| Description |  |
| Instance variables | String title: the title of the painting  String author: the author of the painting  List<PaintingDetail> paintingInfo: a List object containing the image-description couples making up the narration |
| Methods |  |

|  |  |
| --- | --- |
| Class | PaintingDetail |
| Description |  |
| Instance variables |  |
| Methods |  |

**texttospeech package**:

|  |  |
| --- | --- |
| Class | TextToSpeechManager |
| Description | Wrapper class for the android.speech.tts object |
| Instance variables | String TAG: String used for debug and alerting purposes. |
| Methods |  |

|  |  |
| --- | --- |
| Class | CustomUtteranceProgressListener |
| Description | Progress listener for the narrated utterances. Its purpose is to notify the main thread during key moments of the utterance life cyle; |
| Instance variables | String TAG: String used for debug and alerting purposes. |
| Methods | onStart(String utteranceID):  onDone(String utteranceID):  onError(String utteranceID): |

## Node.js Server

External modules used:

* **cors:** Cross-Origin Resource Sharing authorization;
* **express:**
* **multer:** storage;
* **fs:** access to the file system;
* **request:** forwarding of the request to the Django server.

## Django Server

TensorFlow is an open source platform for machine learning. It provides a comprehensive, flexible ecosystem of tools, libraries and community resources tha lets researchers push the state-of-the-art in ML

## Testing and Case study

In this last paragraph we will talk about the case study used to test the realized application.

“The Birth of Venus”, by Italian renaissance artist Sandro Botticelli was chosen as a case study for the experimentation of the application; such a piece lends itself perfectly to the purpose, given the rich history that characterizes it and the clear spatial separations of characters and elements in the scene depicted.

### Painting analysis



Figure 3 The birth of Venus - Sandro Botticelli

The main focus of the composition is the goddess of love and beauty, Venus, born by the sea spray and blown on the island of Cyprus by the winds, Zephyr and, perhaps, Aura. She is met by a young woman, sometimes identified as the Hora of Spring, who holds a cloak covered in flowers, ready to cover her. A detail often overlooked is the lack of shadows in the scene; according to some interpretations, the … is set in an alternative reality, still very similar to our own.

The goddess is standing on a giant scallop shell, as pure and perfect as a pearl. She covers her nakedness with long, blond hair, which has reflections of light from the fact it has been gilded. The fine modelling and white flesh colour give her the appearance of a statue, an impression fortified by her stance, which is very similar to the Venus Pudica, an ancient statue of the greek-roman period.

You may wonder why Venus is standing on a shell; the story goes that the God Uranus had a son named Chronus, who overthrew his father and threw his genitals into the sea; this caused the water to be fertilised, and thus the goddess was born.

In the top left of the piece we can notice Zephyrus, god of the winds; he is holding Aura, personification of a light breeze. The two are highlighting the pale face of the goddess, while blowing the shell towards the coast.

Regarding Aura, some scholars are in doubt about her identity; she may in fact be Chloris a nymph which married Zephyrus in an alternative story.

The Hora herself may be a complementary version of the nymph Chloris. Are they two versions of the same person then? It might be; the story of this woman is narrated in “I Fasti” by latin author Ovidio and the painted in “The Spring”, by Botticelli himself, where the woman gets kidnapped by Zephyrus to become a mystical figure. The theory is quite farfetched, however there’s a detail in its favour: the roses falling around her and Zephyrus.

# Bibliography

[1] R. Szeliski, Computer Vision: Algorithms and Applications, Springer, 2010

[2] D. Kriesel, A Brief Introduction to Neural Networks, 2005

[3] R. Azuma. Tracking requirements for augmented reality. Communications of the ACM, 36(7):50-51, 1993

1. Fritz AI, Image Recognition Guide, <https://www.fritz.ai/image-recognition/> [↑](#footnote-ref-1)