DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Bachelor's Thesis in Informatics

Development of an Augmented Reality App for Stolperstein Memorials

Nicolas Lenßen

DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Bachelor's Thesis in Informatics

Development of an Augmented Reality App for Stolperstein Memorials

Entwicklung einer Augmented Reality App für Stolperstein-Denkmäler

Author:	Nicolas Lenßen
Supervisor:	Prof. Gudrun Klinker, Ph.D.
Advisor:	Dr. rer. nat. David A. Plecher, M.A.
Submission Date:	09.08.2021

I confirm that this bachelor's thesis in informatics is my own work and I have documented all sources and material used.

Munich, 09.08.2021 Nicolas Lenßen

Acknowledgments

I would like to thank David for our regular meetings, the interest and feedback shown and the opportunity to implement my own ideas.

I would also like to thank my friends who took the time to test the application and supported and motivated me with their feedback and interest. I would especially like to mention Lukas, who took the time to review large parts of the work and make suggestions for improvement.

Abstract

With over 80,000 memorial plaques, the *Stolpersteine* are the world's largest decentralized memorial. However, no more information than the names of the Holocaust victims is accessible on the memorial slabs embedded in the ground. To address this shortcoming, we developed a mobile application that displays more detailed information about the victims using Augmented Reality. In order to ensure the accurate location of this information, we developed a method that uses object detection to recognize the plaques and localize them precisely in three-dimensional space. Furthermore, the application is accompanied by a web interface, allowing authorized users to enter the information to be displayed without much effort. Due to circumstances, it was difficult to formally evaluate the application, but feedback from an informal evaluation indicates that the application is generally well received.

Contents

1 Introduction

"A person is only forgotten when their name is forgotten" - This quote from the Talmud captures the central idea of the *Stolperstein Project*[1], which commemorates the victims of National Socialism. It commemorates them individually by means of small memorial plaques, so-called *Stolpersteine* (engl. Stumbling Stones), which are set in the sidewalk in front of the victim's last voluntary place of residence.

Each *Stolperstein* bears the name of the victim, the date of birth, and in some cases the place, and date of death. Their purpose is to keep alive the memory of these people and at the same time to show how they lived in the midst of our society before they were persecuted and forced to move. This makes them a memorial, but at the same time a reminder and warning sign that lets the viewer mentally "stumble" when he sees them while walking, as the founder of the project, Gunter Demnig, puts it [Dem21]. An example of a *Stolperstein* installation can be seen in Fig. 1.1.

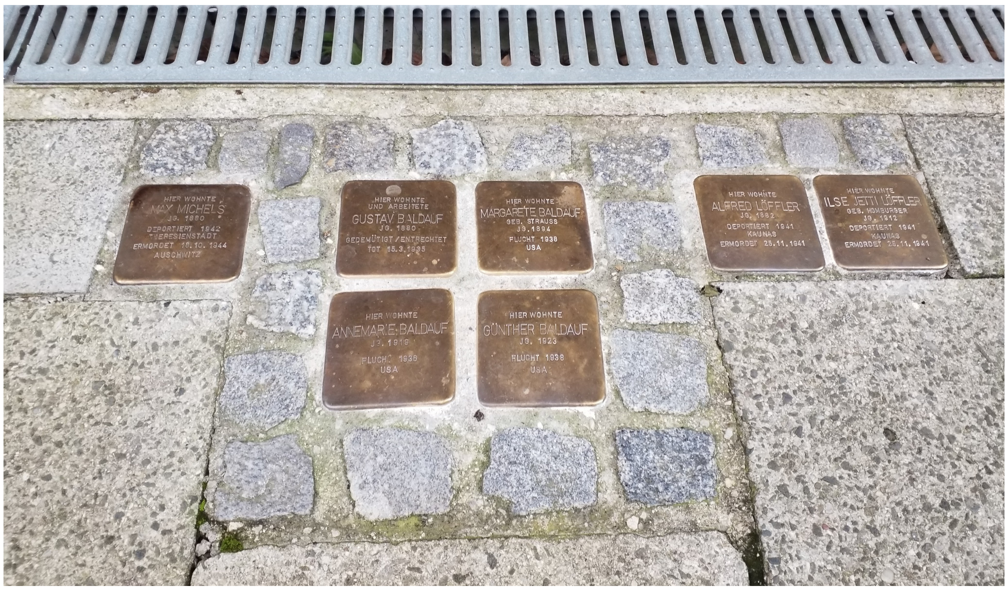


Figure 1.1: Stolpersteine in the Seestrasse 8 in Munich

[1]`http://www.stolpersteine.eu` [accessed 2021-08-03]

The majority of the *Stolpersteine* are dedicated to Jewish victims but can be commissioned by anyone for 120 euros for anyone persecuted by the National Socialists, such as Jews, Sinti or Roma, homosexuals or victims of National Socialist euthanasia programs. Since its initiation in 1992, the project has gained such great popularity, that nowadays over 80,000 *Stolpersteine* in 1,600 cities in 26 countries [JEW19] (as of 2019) are installed. This makes it the largest decentralized memorial in the world [Www18].

Besides the great popularity of the project, however, there is also criticism from some quarters. The possibility to be overlooked and the lack of interaction with the monument, due to its placement on the ground, is a point that is often raised. Furthermore, it is criticized that the *Stolpersteine* barely offer any information about the victims.

Our work addresses these issues by developing an Augmented Reality mobile application that presents more information about the victims behind the *Stolpersteine*, which the user can access interactively.

In the following chapters, we explain the necessary technical background, discuss the requirements for the application, and describe the technical realization of the application. Finally, we present the results of an informal evaluation, and mention improvement possibilities for the future.

2 Technical Background

This chapter gives an overview over Augmented Reality as the technology chosen for the implementation of the application and provides the necessary technical background to understand the used method of object detection.

2.1 Augmented Reality

Intuitively, Augmented Reality (AR) can be described as "the act of superimposing digital artifacts onto real environments" [Pan+18]. A more precise and also widely accepted definition of AR comes from Azuma, who defines AR as an experience that combines **real** and **virtual** content, that is **interactive in real-time** and **registered in 3 dimensions** [Azu97].

Another way of defining AR, is through the Reality-Virtuality (RV) Continuum by Milgram [Mil+95], illustrated in Fig. 2.1. It sees Augmented Reality as situated in a continuum between the real environment, aka the real physical world and a completely virtual world. Together with Augmented Virtuality (AV), which is closer to a completely virtual environment, AR is part of the greater Mixed Reality (MR), which compromises everything between the extrema of the RV Continuum.

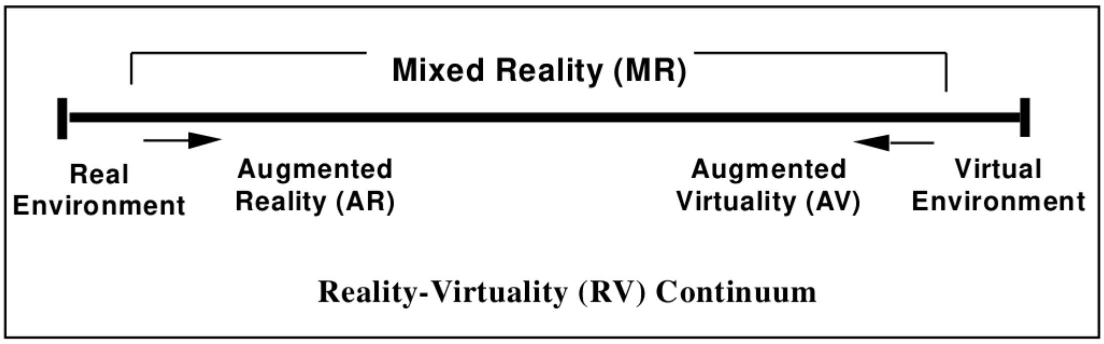


Figure 2.1: Simplified representation of the Reality-Virtuality Continuum [Mil+95]

2.1.1 Applications

According to Carmigniani et al. [Car+11], the four types of applications that are most often used for AR research are advertising and commercial, entertainment and education, medical applications and the mobile sector. However, as the technology has matured in the last years, many of these applications are not only subject to research, but already being used in the real world. In the following sections, some examples of various applications from both research and production in the above areas are briefly presented (without explicitly mentioning the mobile sector).

Advertising and Commercial

Many industries have already recognized the potential of AR for realistic three-dimensional presentation of information and are using it as a technology to present and promote their products. Swedish retailer *IKEA*, for example, has developed its smartphone application *IKEA Place*[1], which allows users to virtually place models of their furniture products in their own space, as shown in Fig. 2.2. This allows users to test whether the furniture will fit in their space before making a purchase, and of course helps *IKEA* promote their products.

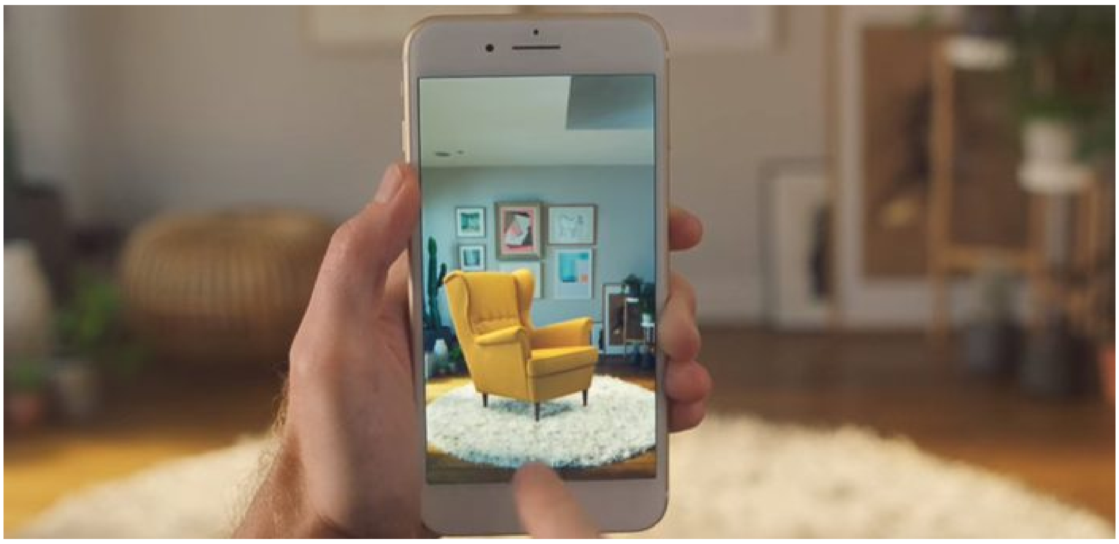


Figure 2.2: The *IKEA Place* AR app lets the user virtually position furniture [Gue21]

[1]`https://apps.apple.com/us/app/ikea-place/id1279244498` [accessed 2021-08-03]

Entertainment and Education

The concept of displaying information in 3D space has many applications in the education/tourism sector: For research purposes, many museum guides have been developed as in the works [Miy+08; SW05; Dam+08]. These guide the user through the exhibition and offer additional information concerning the exhibits at the proper positions.

In the natural history museum of the Smithsonian Institution visitors can use the AR application *Skin and Bone*[2] to see the reconstructed animals imposed over their exhibited bones (See Fig. 2.3). Other museums such as the National Museum of Singapore[3] use AR to create immersive experiences, e.g., by transforming drawings into three-dimensional animations. Besides in museums, AR can also help to convey cultural heritage by virtually reconstructing former structures in real-life size. A very early example of this is *Archeoguide* [Vla+02], a personalized tour guide that takes the user around the ruins of ancient Olympia and uses AR to show what these buildings (might) have looked like in the past.

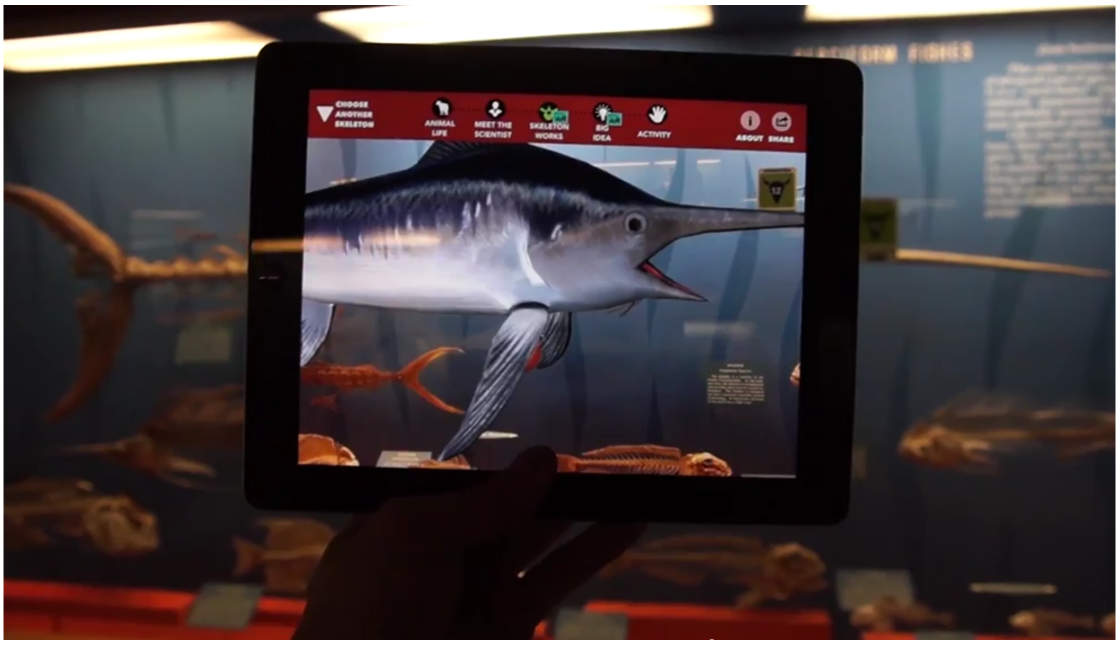


Figure 2.3: Skin and Bones AR application. Screenshot from the museum's demo video.

[2]`https://naturalhistory.si.edu/exhibits/bone-hall` [accessed 2021-08-03]

[3]`https://www.nhb.gov.sg/nationalmuseum/our-exhibitions/exhibition-list/story-of-the-forest` [accessed 2021-08-03]

Medicine

In the medical sector, AR could become an invaluable tool to support the medical personal, for example, in computer-aided surgery, where efforts are made to provide the surgeon with X-ray vision of the relevant parts of the patient during surgery using a head-mounted display [Bir+02].

In addition to its use in the operating room, AR also finds application as a novel treatment method in rehabilitation or therapy [EVF19]. For example, Escobedo et al. have shown that their mobile application increases sustained and selective attention in children with autism and elicits positive emotions during therapies.

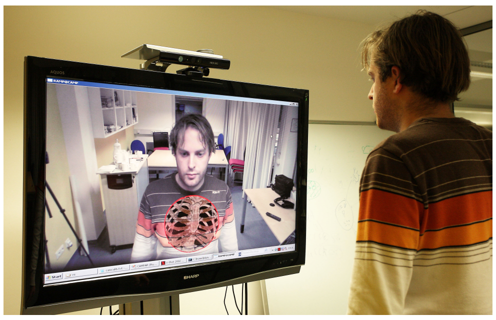
2.1.2 Augmented Reality Devices

Most AR devices fall either into the category of **Head Mounted Display (HMD)**, **Handheld Augmented Reality (HAR)** or **Spatial Augmented Reality (SAR)** [Car+11].

SAR devices are separated from the user and usually stationary installed in the environment. The content is displayed via a video screen as in Fig. 2.4a, through video projectors against a surface, or through other technologies such as holograms.

In contrast, **HMD**s are worn on the head and place the AR content directly over the user's view. While they used to be extremely heavy and bulky devices , today, devices like the Hololens in Fig. 2.4b exist, which are almost the size of eyeglasses.

HAR devices are small computing devices—nowadays usually smartphones or tablets—that show imposed AR content using video-see-through techniques. Since current smartphones and tablets are equipped with sophisticated sensors, have sufficient computing capacity and are widely in use, they have become a very advantageous platform for developing AR applications [WSG14]. The application developed in this work also falls under the category of Handheld Augmented Reality.


(a) SAR in form of an AR mirror [Blu+12]

(b) The Hololens HMD from Microsoft [Mar21]

Figure 2.4: AR devices

2.2 Object Detection

Object Detection is the task of detecting and locating instances of visual objects in an image [Zou+19]. This is done by predicting a rectangular bounding box for the location as well as a label for the class with a confidence score (see Fig. 2.5).

In the last ten years, the field has been revolutionized, as Deep Neural Networks (DNNs) replaced the traditional methods that relied on carefully designing feature extractors by hand [Zou+19]. In contrast to these, a DNN is capable of learning feature extractors from annotated examples alone and far surpass traditional methods today.

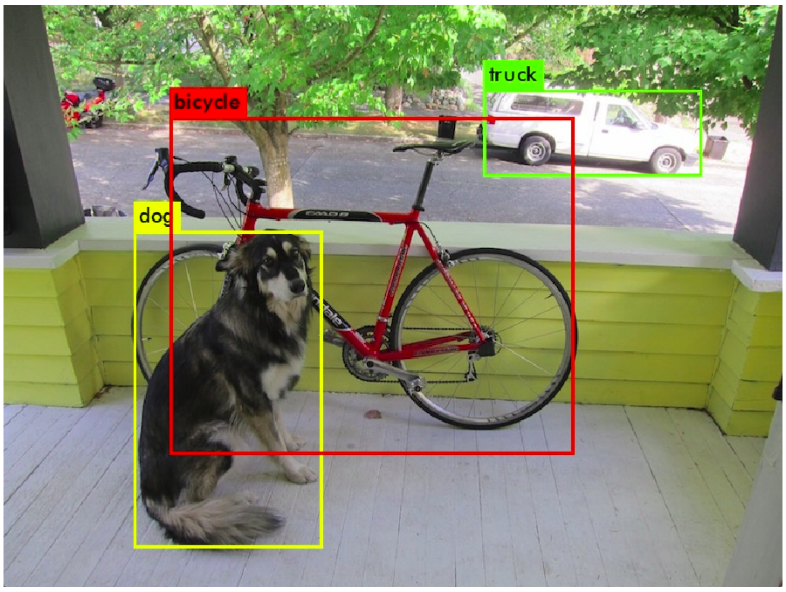


Figure 2.5: An example of applied object detection [Red18]

2.2.1 Benchmarks

At this time, the de-facto standard for object detection in the community is the Microsoft Common Objects in Context (COCO) dataset[4] [Lin+15] benchmark. It consists of 164k images with 897k annotated everyday objects from 80 different classes. It is considered more difficult and more realistic than popular previous datasets, such as the Pascal VOC [Eve+10] or the ILSVRC [KSH12] datasets [Zou+19].

[4]`http://cocodataset.org` [accessed 2021-08-03]

2.2.2 Deep Neural Networks

Abstractly, DNNs are universal function approximators [HSW89]. In this sense, the task of object detection can also be viewed as the approximation of a very high-dimensional function.

The function to be approximated is the function that assigns every image the correct bounding box coordinates and confidence values. The DNN then essentially is a function with many parameters (called **weights**), which are adjusted in the training process, such that it constitutes a good approximation of the function to be approximated. Inputs to this function are the images usually in the form of three-dimensional tensors representing their RGB values. After the training, if done right, the DNN can be used to make predictions on new images, which we call **inference**.

Layers

DNNs consist of multiple layers, which are relatively simple functions, that can be chained to approximate more complex functions. The input and outputs of each layer are tensors. In the context of object detection the first input tensor corresponds to the input image. The tensors usually are of shape $NxCxHxW$ or $NxHxWxC$, where N stands for the **batch** dimension, C for the channels, H for the height and W for the width. For the input image, C corresponds to the color channels, and H and W to the image height and width, but in between layers, these dimensions usually change.

The single elements of these tensors are in some scenarios referred to as "neurons," resulting in the name "Deep Neural Network," where the term "Deep" comes from the fact that many of the above-mentioned layers are stacked.

In the following, the layers relevant and used in this work will be briefly explained:

Convolutional Layer The convolutional layer applies a filter kernel to extract features from the input. This works like traditional methods such as the Sobel operator [Z+98] that is used to detect edges in an image. The kernel, in the case of 2D convolutions is a $h \times w \times c$ matrix, where h and w stand for height and width respectively, and c for the number of filters, together referred to as **kernel size**.

The output values are calculated by sliding this filter kernel over the input by a number of steps, called **strides** and adding the input elements together, weighted by the kernel. However, in contrast to the traditional methods, the values (weights) that constitute the filter kernel are not defined in advance, but the network learns those values during the training process.

Leaky ReLU Layer The leaky ReLU layer consists of an activation function that is applied to each element of its input tensor. These non-linear functions are needed since the other layers usually represent linear functions. The combination of linear functions is again a linear function, which drastically limits the network's capabilities to approximate an arbitrary function. The activation function of the leaky ReLU layer is defined as follows:

$$f(x) = \begin{cases} x & \text{for } x \geq 0, \\ a \cdot x & \text{for } x \leq 0 \end{cases}$$

where a is a scalar where $0 < a < 1$, often with $a = 0.1$.

Max Pool Layer Pooling layers, in general can be used to reduce the dimensionality of the input. A max pool layer achieves this, by performing a convolution with a kernel that outputs the maximum of its input values. Like in a convolutional layer, this kernel is slid over the input by some number of strides and has a kernel size.

For example, a 2D max pool layer with kernel size 2×2 and strides of 2 would halve the size of its input in both width and height dimension.

Upsampling Layer The upsampling layer is used to scale up the input tensor. Therefore, it usually serves the contrary purpose of a max pool layer. Methods to achieve this include repeating the tensor elements or setting new values by interpolation between the original elements, where the **scale factor** defines by which factor the input is scaled up in the W and H dimension.

Route Layer With these layers, features from earlier in the network are preserved.

There exist two versions of this layer: The first concatenates the output from a layer earlier in the network with the output from the layer directly before this layer.

The second version reorganizes the input tensor, such that every alternate pixel is put in another channel. This results in an output of dimension $N \times \frac{H}{2} \times \frac{W}{2} \times C \cdot 4$ for input of size $N \times H \times W \times C$.

Training

In training, the DNN is shown annotated examples (training data) to "learn" to make good predictions, i.e. find a better approximation of the desired function.

Training Loop The training process takes place in the training loop (Algorithm 1):

```
while not done do
    (0) Sample from the training data.
    (1) Make prediction for the sample.
    (2) Compute prediction loss.
    (3) Compute gradient of weights with respect to loss.
    (4) Update network weights.
end
```

Algorithm 1: The DNN training loop

The loop is run either for a fixed number of iterations or can be prematurely stopped, when the predictions do not improve. During a step in the loop, the following is performed:

(0) First a certain number of data points from the training data, a batch, is sampled. (1) Predictions with the current network for this batch are made. (2) The output of the predictions is compared with the ground truth, which are the labels for the respective input. The loss—some distance function specifying how far the predictions deviate from the ground truth in this step— is computed. (3) Next, the gradient of the weights with respect to the loss is computed. For each weight the gradient corresponds to the contribution to the loss that this weight had. (4) With the gradient, the DNNs weights are updated such that the prediction would have more closely matched the ground truth data. The factor determining how strong this update is, is called **learning rate**.

Data Augmentation

In order to train a DNN successfully, a high quantity and quality of training data are required. Here quality means that the training data should come from the same source as the data that will be later used in inference. Further, it should be diverse and balanced, such that the network is able to generalize from it.

Because collecting training data is usually quite time and/or cost-consuming, the method of *Data Augmentation* is often used to enlarge the set of training data. It artificially extends the training data by generating new training data by modifying existing training data. This also has the benefit that it can make the trained DNN more robust since it will be presented with input variations that might otherwise not be included in the training data, e.g., darker images.

Examples for data augmentation for image data are:

Rotation: Rotate the image by a random angle.
Saturation: Randomly change the saturation of the image.
Hue: Randomly change the hue (color) of the image.
Exposure: Randomly change exposure (brightness) of the image.
Noise: Add noise to the image, e.g. random white and black dots.

3 Related Work

3.1 Location-Based Mobile Augmented Reality Apps for Cultural Heritage in the Context of Commemoration

Several other works that also use location-based augmented reality using mobile devices to preserve the memory of specific people or events. This chapter will introduce some of them and how they are related to and differ from our work.

3.1.1 Border Memorial: Frontera de los Muertos

The Border Memorial: "Frontera de los Muertos" [FA15] is an augmented reality public art project and memorial. It commemorates the thousands of Mexican migrant workers who died while trying to cross the U.S./Mexico border.

At its core, it consists of a smartphone application that allows the user to visualize where human remains have been found in the area around the border and its surrounding desert. This is achieved by leveraging geolocation software to superimpose a skeleton in the traditional Mexican "Dia de Los Muertos" style at the GPS locations of each recorded death, as can be seen in 3.1. By using only the GPS to localize the positions where the objects are to be placed in AR, the application can only reach a precision of a few meters. However, in this case, this is not a problem since the placed content does not interact with the environment, and the user does not know the exact location where it would be supposed to be.

Similar to the *Stolpersteine*, the memorial published by Freeman and Auchter "focuses on individual-level mourning and memorialization" [FA15] and makes this commemoration happen in a place strongly connected with the victim, here through their place of death, in our application through the last freely chosen place of residence. However, the authors of this work do not identify the individual nor provide any further information about it. This focus on the holistic problem rather than individuals distinguishes it from our application, whose explicit goal is to provide more information about the victim. In our work, the additional information ensures that the observer does not perceive the victim as one of the thousands but rather understands his or her story in more detail. What further sets our work apart from this one is that we do not create a new monument but expand an already existing one in the virtual space.

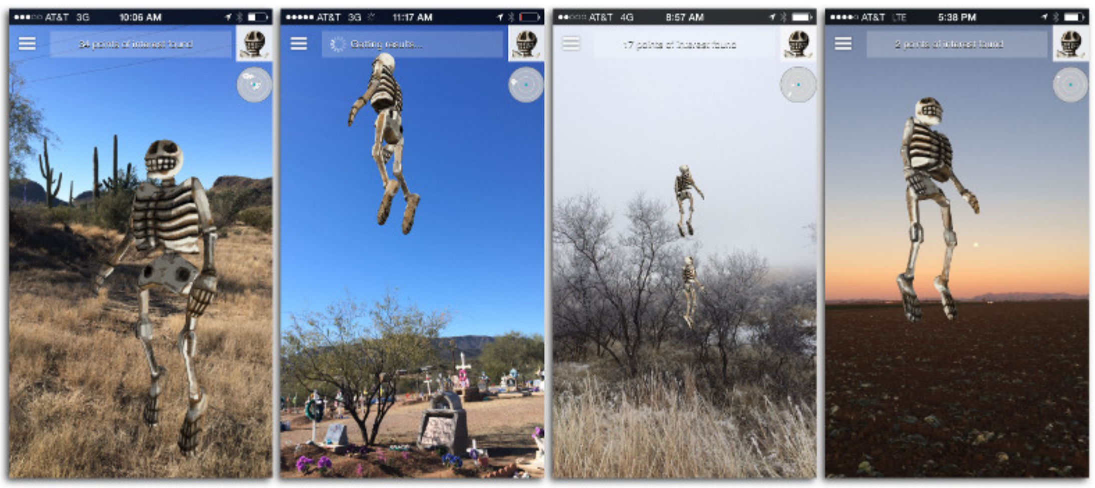


Figure 3.1: Screenshots of the Border Memorial application, published by the author. Skeletons are superimposed at the locations of recorded human deaths.

3.1.2 Bomb Sight

Bomb Sight [JAS13] is an academic project mapping the locations of the bombs that fell on London during the Blitz from 07/10/1940 to 06/06/1941 to a publicly accessible map on their website[1]. Accompanying the website, an app was released [Bom] that allows users to discover these locations through Augmented Reality.

When using the app in London, red bomb symbols are positioned in world space at the places where a bomb fell during the second world war in a 300-meter radius around the user, as can be seen in a screenshot of the application in Figure 3.2. By tapping on the bomb symbol, additional information is displayed on the screen, such as the current distance to the bomb and images from that period.

The application relates to our work in the way that it uses Augmented Reality as a way to present location-related information with a focus on a city-wide scale. It is noteworthy, however, that the user does not necessarily visit the precise location since the information can be revealed from far away by clicking the corresponding icon or through selection from the overview map. This is in contrast to our application where the user has to be on site to obtain the information. Furthermore, here the information itself is not presented in AR but exclusively in 2D in the form of text and images. Only the spatial point that this information relates to, the location where the bomb fell, is shown in AR. In this application, GPS is also sufficient to achieve sufficient localization accuracy, since the point only needs to be determined to within a few meters.

[1]`www.bombsight.org` [accessed 2021-08-03]

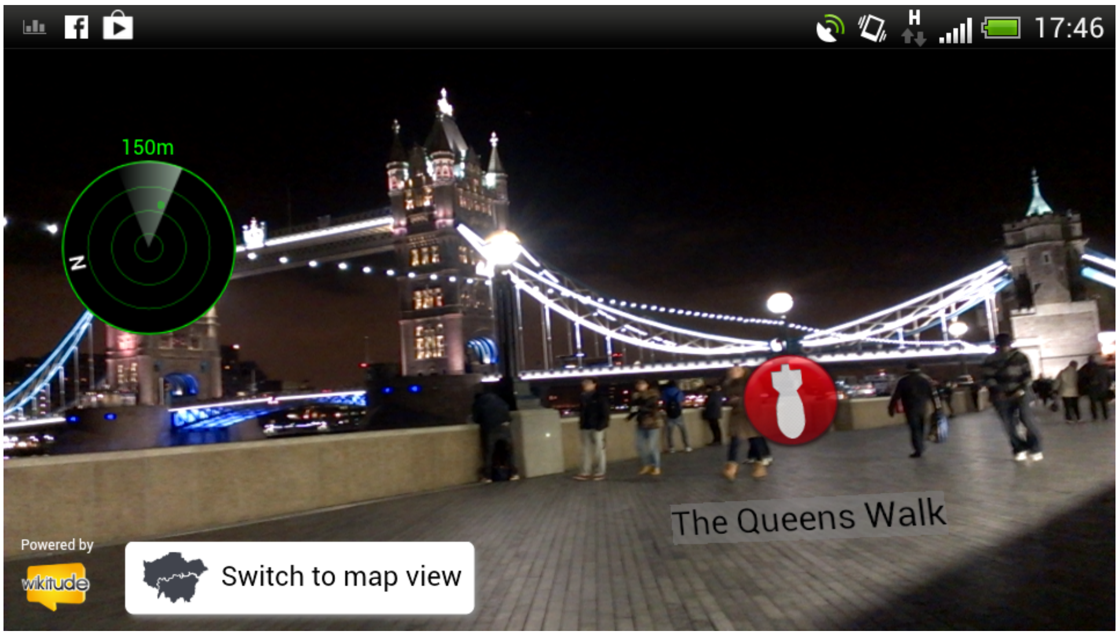


Figure 3.2: Screenshots of the Bomb Sight application [Bom]. The red bomb icon marks the position where a bomb fell during the Second World War.

3.1.3 Bergen-Belsen AR App

Through the integration of historical database content with AR, Pacheco, Wierenga, Omedas, et al. ([Pac+14], [Pac+15]) aim at enhancing cultural heritage content. To this end, they deployed the Bergen-Belsen Augmented Reality app, "a mobile, on-site, guide/companion tool for the geolocalization, visualization[,] and exploration of historical data" [Pac+14]. It has been developed for Bergen-Belsen, the location of a former concentration camp that was burned down shortly after its liberation, in fear of the spread of a typhus epidemic.

The application allows the users to walk through the site, now a nature park, in a guided or unguided manner and experience 3D reconstructions of the former structures in Augmented Reality at chosen points of interest (see Fig. 3.3). In this way, AR is used to bring the complex historical data about the concentration camp back into 3D space to "facilitate the understanding of complex historical datasets and the generation of more meaningful experiences in cultural heritage contexts" [Pac+14].

A close connection to our work is obviously its focus on commemoration and information about the Holocaust through the help of AR. The work further also uses the concept of "points of interest." Users obtain the information only when visiting these locations, encouraging an active engagement with the place, which is similar to the role that the *Stolpersteine* play in our application. Similar to our application, this work also integrates a database interaction and offers a web application to facilitate the addition and editing of the information content by other users. The authors mention that for the placement, device GPS, gyroscope and compass are used with an GPS accuracy of around 1.5 meters. We assume, that this comparatively good GPS accuracy results from the use in a low-interference outdoor environment and should be sufficient to place the content approximately in its right place.

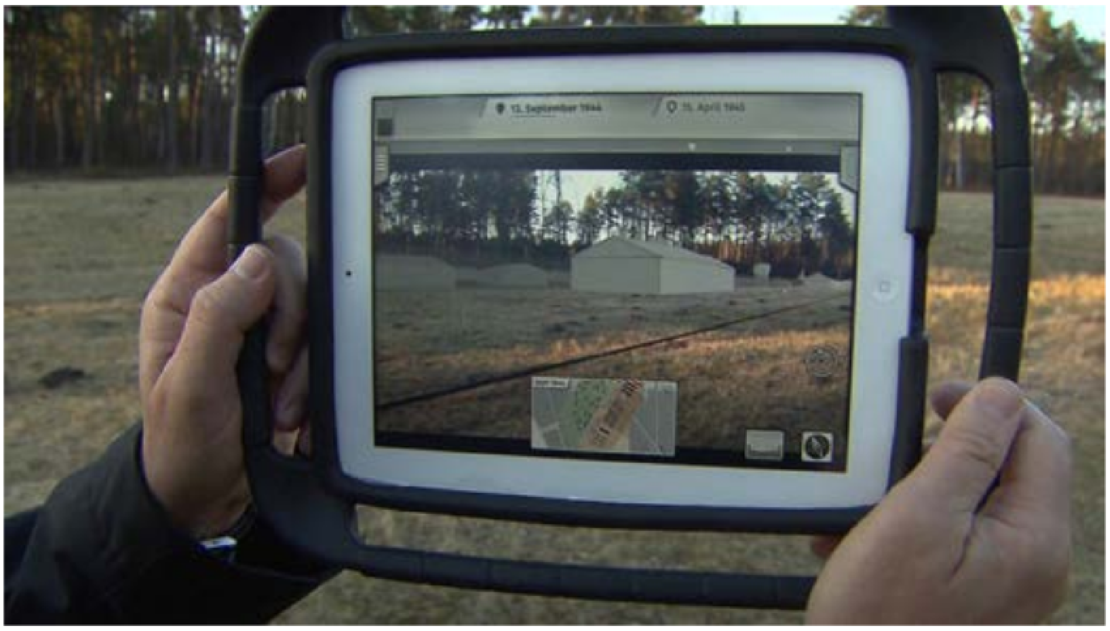


Figure 3.3: Using the Bergen-Belsen app, the user can see the virtual reconstructions at points of interest. Demonstration image from [Pac+15].

3.2 Combining Augmented Reality and Object Detection

Similar to our application, other research works also combine AR with object detection in order to detect and localize objects, that do not have the properties of traditional markers.

Upadhyay, Aggarwal, Bansal, and Bhola [Upa+20] for example aim to build an interactive shopping experience using AR, that displays information about the products and lets the user add them to their shopping cart. In order to classify the products and get their position for the placement of the AR information, they leverage object detection. To this end, an SSD [Liu+16] object detector based on the mobilenets [How+17] architecture is used and trained on a dataset of shopping items.

However, as described in the work, the application is run on a laptop with a webcam, instead of a mobile device, which would be the expected device choice for such an application. We suspect the realization on a SAR instead of HAR device here is due to the inference time of the DNN used for object detection. Due to their comparatively weaker processors and graphic cards, inference times are significantly higher on mobile devices and seem to be an obstacle to the use of object recognition in mobile AR applications in many cases.

In "Edge assisted real-time object detection for mobile augmented reality" [LLG19] the inference time problem is tackled by designing a method to efficiently off load the object detection task to the cloud. This way most compute resources stay available for the rendering of the AR content and with a powerful GPU in the cloud, the detection and return of the result can be completed in real-time. Especially for applications that require a continuous real-time detection, this method looks very promising.

However, the object detection task, does not necessarily have to be transferred to the cloud. The authors of [Rao+17] chose another path of combating the inference time problem, by implementing their own lightwight version of the SSD detector for mobile devices, based on a truncated squeeznet [Ian+16] architecture. This way they obtain a object detector that achieves around 2 FPS on a Qualcomm Snapdragon 821 mobile CPU. The detector is used together with GPS, IMU and Magnetometer, to identify and localize objects from a database, such that information can be added to them in AR.

This combination of geospatial data and object detection for the recognition and differentiation of the objects is similar to our application. The setting however is somewhat different, as the objects in the described work are mostly buildings and thus farther away. This means that in this case the GPS inaccuracy is negligible and the GPS position can be used to determine the distance to the object to a sufficient degree. With this distance it is then possible to project the coordinates of the object detected in the 2D screen to its approximate 3D coordinates. This detection method differentiates it from our work, as well as their approach to perform the detection continuously, which—given the 2 FPS inference time their method has—leads to an application which only achieves "near-real-time performance" as the authors state.

4 Concept

In the following, we state the motivation for our work and indicate the identified requirements that our application should meet to achieve the mentioned goals.

4.1 Motivation and Goal

Although they are generally very popular and receive a lot of praise, the *Stolperstein* memorials have also been criticized time and again. For one thing, some people find the installation on the ground inappropriate because it does not allow for interaction at eye level with the memorial, and the *Stolpersteine* could be stepped on when walking over them. Another point that has been criticized is that by only displaying the names and birth/death dates (and sometimes location) of the victims (as can be seen in Fig. 1.1), the memorials do not convey enough information and therefore are not able to awaken empathy with the victims [Wet20]. Typically, such concerns could be addressed by placing an information panel providing further information regarding the memorial. However, in the case of the *Stolpersteine*, this is not a feasible option since it would require severe constructional interventions, bureaucratic burdens, and greater costs, all factors that go against the idea of the project to make the creation of a memorial as uncomplicated and affordable as possible.

With our work, we want to address this problem by developing an application that enhances the monument by presenting additional information about the victims and make them interactively accessible. Thereby we hope to improve the commemoration of the individual persons, as well as educate viewers about the circumstances and history that these persons had to live through.

Because the placement of the *Stolperstein* at a specific location is essential to the memorial, Augmented Reality was chosen as a technology since it allows to connect the information to the physical location without requiring to alter the physical environment. Further, it enables a way of commemoration which literally takes place "on eye level," e.g. through the placement of photos of the person in the physical world in front of the user. Additionally, studies such as [KJO19], and [DIK13] and [Ibá+14] have shown that Augmented Reality as an immersive, new and exciting way of interacting can increase the learning motivation of users. We also hope to benefit from these effects to increase attention and receptivity to the stories and facts presented.

4.2 Mobile Application Requirements

In order to achieve the goals stated in 4.1, the following requirements for the mobile AR application have been identified:

- The application should be able to recognize and localize *Stolpersteine*.
- The application should allow the user to obtain more information through interaction with recognized *Stolpersteine*.
- The application should display the information for a selected *Stolperstein* in AR at the precise location of the *Stolperstein*. The user should not be overwhelmed with information but be able to access it piece-wise through interaction with the memorial.
- The information should be conveyed through different media, such as text on the screen, images, or virtual maps placed into the physical world.
- The application should generate the content programmatically for every *Stolperstein* based on data coming from an external database, such that only textual/image data has to entered into the database.
- When not close to a *Stolperstein*, the user should be able to access an overview of the *Stolpersteine* in her/his surrounding.

4.3 Backend Requirements

Further, to present the content for as many *Stolpersteine* as possible and let responsible persons edit and add the presented information, a way to input information into the application is needed. The responsible persons are researchers or relatives of persons a *Stolperstein* has been dedicated to and will be refered to as data administrators.

To this end, a backend with web interface offering the functionality below is created:

- The web interface should allow data administrators to add, edit and delete information about the person a *Stolperstein* is dedicated to.
- The modification of the information should only be possible for authorized data administrators.
- The web interface should be easy to use for data administrators without a technical background.
- The backend should store the information in a database and offer an interface through which the mobile application can access this information.

5 Technical Realization

At its core, our system consists of two parts: The mobile application, which presents the content to the user in AR, and the backend with a web interface, through which data administrators can input information about the *Stolpersteine*.

5.1 Architecture Overview

This section provides an overview of the components of the system and how they interact with each other. A graphical representation can be found in Figure 5.1.

The main part of the system, the mobile application, was built using *Unity*[1]. This C# game engine is very well suited for the development of AR applications due to its advanced AR framework, *ARFoundation*[2], also used in this work for displaying AR content. The **Backend Interface** (5.6) allows the components of the mobile application to access data from the Backend, such as information on individual *Stolpersteine* or the coordinates of all *Stolperstein* locations. The latter are required by the **Location Handler** (5.2) to check whether a *Stolperstein* is nearby. If this component detects that the area of a *Stolperstein* is entered, the **Stolperstein Area Handler** (5.3) is notified, which handles the logic while the user is inside the *Stolperstein* area. It calls the **Stolperstein Detector** (5.4) in order to determine the position of the *Stolperstein(e)* that are present and orders them based on the order information given by the Backend Interface. Finally the **Scene Constructor** (5.5) is called, in order to create the AR content for the selected *Stolperstein* from data provided by the Backend Interface.

The Backend offers a **Webinterface** (5.7.2) through which data about the *Stolpersteine* can be added or modified. The data is subsequently stored in the **Database** (5.7.3) and can be accessed over the **Backend API** (5.7.1) by the mobile application.

The individual components are explained in more detail in the following chapters roughly in the order they would normally be called when using the application, followed by the Backend Interface and the Backend.

[1]`https://unity.com/` [accessed 2021-08-04]

[2]`https://docs.unity3d.com/Packages/com.unity.xr.arfoundation@4.1/manual/index.html` [accessed 2021-08-04]

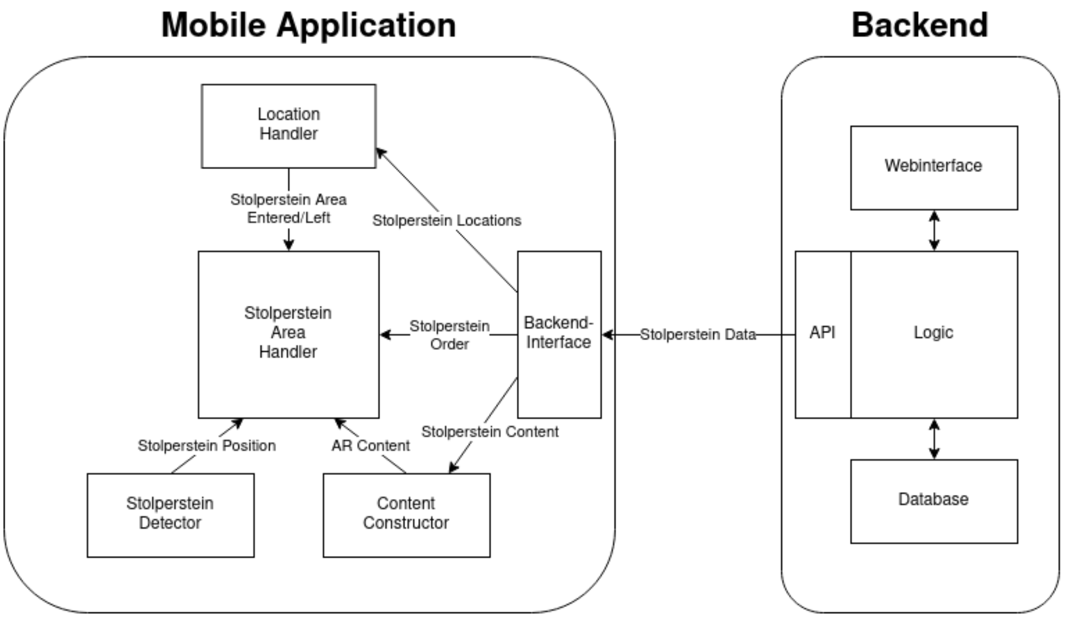


Figure 5.1: Overview of the system architecture. Rectangular boxes define the different components. Arrows depict the messages/data passed between components.

5.2 Location Handler

The Location Handler component serves two purposes: first, it has to check whether the area of a *Stolperstein* location is entered or left. In the case of entering an area, it also has to determine which *Stolpersteine* are located there. We will refer to this area, which is the area in a certain radius around the location where *Stolpersteine* are installed as the *Stolperstein* area.

To achieve the goals stated above, the component continuously accesses the user's location and compares it against a list of *Stolperstein* locations. To get the current user location in GPS coordinates, the *MapBox*[3] Location Service is used, which relies on the device's GPS and network location. It periodically updates the user location based on the GPS data from the device; however, if the device network location is fresher and more accurate, the network location is preferred over the GPS data

For every location update, the Location Handler compares the user coordinates against a list of all coordinates of *Stolperstein* locations, initially retrieved from the

[3]`https://www.mapbox.com/` [accessed 2021-08-04]

Backend Interface. The correct measure of distance between two coordinates in a geographic coordinate system using longitude and latitude can be calculated by the Haversine Formula [Van13] or with Vincenty's Formula [Vin75]. However since the user is located quite close to the relevant *Stolperstein(e)*—at least in the same city—we can get a sufficient approximation of the distance by not taking the spherical nature of the earth into account, using a function developed for this purpose by Mapbox [Aga21].

The distance to the closest *Stolperstein* location is used to determine whether the radius of a *Stolperstein* is entered or left. If this is the case, the corresponding event is fired, containing the entered *Stolperstein* location, if applicable.

A value of 15 meters was chosen for the radius, to compensate for GPS inaccuracy, which varies between 7-13 m for a modern smartphone in an urban outdoor environment [MB19]. During testing, we could confirm that entering this radius usually also means that the *Stolperstein(e)* are already visible for the eye.

While the user is outside a *Stolperstein* area, the Location Handler displays the current position and the locations of the *Stolpersteine* nearby on a map that can be enlarged to fill the whole screen by clicking on it.

5.3 Stolperstein Area Handler

Once inside a *Stolperstein* area, the Stolperstein Area Handler coordinates the logic and different components to initiate the detection of the *Stolpersteine* and handle the user selection of a certain *Stolperstein*.

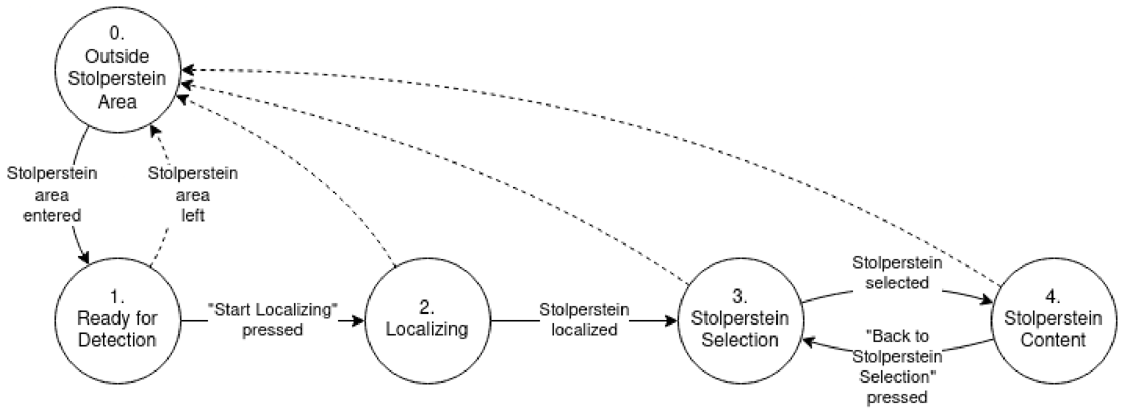

0. Outside Stolperstein Area
Stolperstein area entered
Stolperstein area left
1. Ready for Detection
"Start Localizing" pressed
2. Localizing
Stolperstein localized
3. Stolperstein Selection
Stolperstein selected
"Back to Stolperstein Selection" pressed
4. Stolperstein Content

Figure 5.2: State diagram of the application. Circles denote states that the application can be in, arrows the transitions. Dotted arrows all represent the transition "leave stolperstein area."

This logic is implemented by the following states, also visualized as a state diagram in Fig. 5.2:

0. Outside the area, the component is in a passive state and does nothing.

1. Once the area is entered, the app prompts the user to locate himself/herself in front of the *Stolerstein(e)* and press a button to start the localization. The ground plane detection is also started, which will be later needed by the Stolperstein Detector component.

2. When the user has pressed the localization button, the **Stolperstein Detector** is invoked, which is described in detail in Section 5.4.

3. Once the detected positions of the *Stolpersteine* are returned, it is inferred which position corresponds to which *Stolperstein*, by sorting them based on their position relative to the user on the ground plane and comparing them to the order information made available by the **Backend Interface** (see 5.6).

   For each detected *Stolperstein*, an anchor is created, which is a pose in the physical environment that is tracked using additional computational effort. In the following, the anchors will be used as reference points for the placement of the AR content for the respective *Stolpersteine*.

   An object is augmented on top of each detected *Stolperstein* to provide more information after clicking on it. Which object is augmented depends on the reason for the persecution of the victim, e.g., for Jewish Holocaust victims, a Star of David is augmented on the *Stolperstein*, as can be seen in Fig. 5.3. The person's name is also displayed on the ground above the *Stolperstein* when the device is focused on it, to make it easy for the user to select the desired *Stolperstein*. However, if there is only one *Stolperstein* at the location, the whole selection step is skipped.

4. If a *Stolperstein* is selected, the previous AR content is removed, and the **Content Constructor** is called, which creates the AR content for the selected *Stolperstein* as described in 5.5. The content is then placed at the position of the *Stolperstein*, and the user can interact with it, or through the press of a button, can choose to get back to select another *Stolperstein*.

Should the user leave the *Stolperstein* area at any time, all AR the content is removed, and the component returns to the passive state 0. When the same or another *Stolperstein* area is entered the next time, the process is started again from point 1.

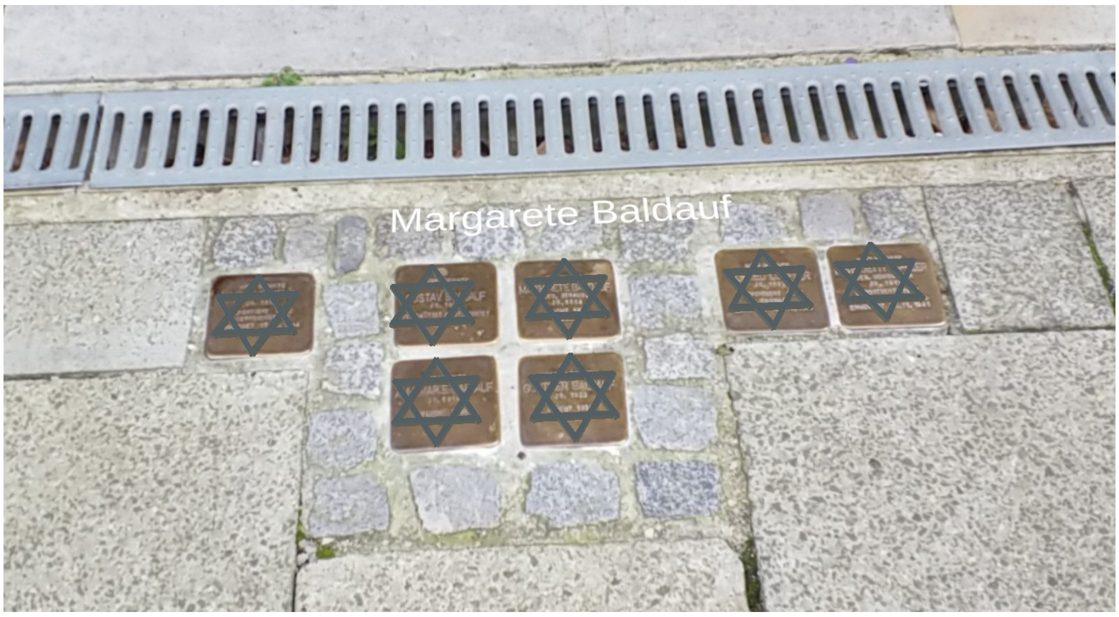


Figure 5.3: Selection State: The detected *Stolpersteine* are marked by augmenting an AR object onto them, in this case, the Star of David, since all victims were persecuted due to their Jewish religion. By touching it, the user can select the *Stolperstein*. Pointing the device towards one of the Stolpersteine, the person's name is displayed over it in AR.

5.4 Stolperstein Detection

In order to place the virtual content correctly, our application has to be able to detect the presence and exact 3D location of each *Stolperstein*. The Stolperstein Detector realizes this task. Because it posed a major technical challenge, we first describe the problem and how we approached it. We subsequently explain our solution and its realization more closely and finally show how it was used to form the Stolperstein Detector component.

5.4.1 The Detection Problem

Problems with traditional methods Localizing an object only via GPS might be an option when the AR content does not have to be placed at the exact location, as is the case in the works presented in the Sections 3.1.2 and 3.1.1. Our application, on the other hand, is intended to display augmentations at the precise location of the *Stolperstein*, which requires centimeter-level accuracy that GPS does not provide (GPS accuracy is discussed in Section 5.2).

A common approach in Augmented Reality to such a problem would be to use a traditional computer vision (CV) approach to detect a marker, which could either be attached to the *Stolperstein* or be the *Stolperstein* itself. However, applying a marker in the real world to each *Stolperstein* is infeasible for apparent reasons, and using the *Stolperstein* as a marker with traditional CV methods also harbors significant problems. Since the application is supposed to work for different *Stolperstein* instances, the markers will vary in their appearance due to different engravings and differences in manufacturing (each *Stolperstein* is a unique handcrafted piece). While all *Stolpersteine* have the same general shape and size, they also share these with cobblestone, as is often be found in the locations they are installed in, thus making them unsuitable for detection with traditional CV methods. Moreover, since they are installed outdoors, they are perceived under very different lighting conditions, which makes the detection of a conventional marker even more difficult. It should also be noted that out-of-the-box object recognition methods exist for augmented reality, most notably those developed by *vuforia*[4]. However, these are only suited for the detection of one particular object as well and face the same problems as described above. Ultimately, we need a method that can generalize across different *Stolperstein* instances perceived in different environments.

Solving the Detection Problem using Object Detection Since the introduction of DNNs, object detection methods have excelled at detecting generalized object classes on 2D images in different environments. The latest advances in the field make it even possible to perform such a task in real-time. Therefore our solution to the problem is to leverage object detection to recognize and locate the *Stolpersteine*.

Finding the bounding boxes of a *Stolperstein* in the 2D image is not enough to determine its 3D world position, because we lack the information in the depth dimension. However, we can take advantage of the fact that the *Stolpersteine* are installed in the pavement in order to get their 3D position. Being embedded in the pavement means that they are positioned on the ground plane, which in turn we can determine.

If we project the point in the center of the bounding box from the 2D image back into 3D space—reversing the camera projection—we get a line of possible positions for the *Stolperstein* in 3D space. Finding the intersection of this line with the 3D ground plane, we obtain a single point that corresponds to the assumed position of the *Stolperstein* in the world. Fig. 5.4 shows the detected ground plane and bounding box in order to visualize the concept.

This approach also means that, in theory, we only need to detect the coordinates of the *Stolperstein* center in the camera image, rather than its full bounding box coordinates.

[4]`https://library.vuforia.com/features/objects/object-reco.html` [accessed 2021-08-04]

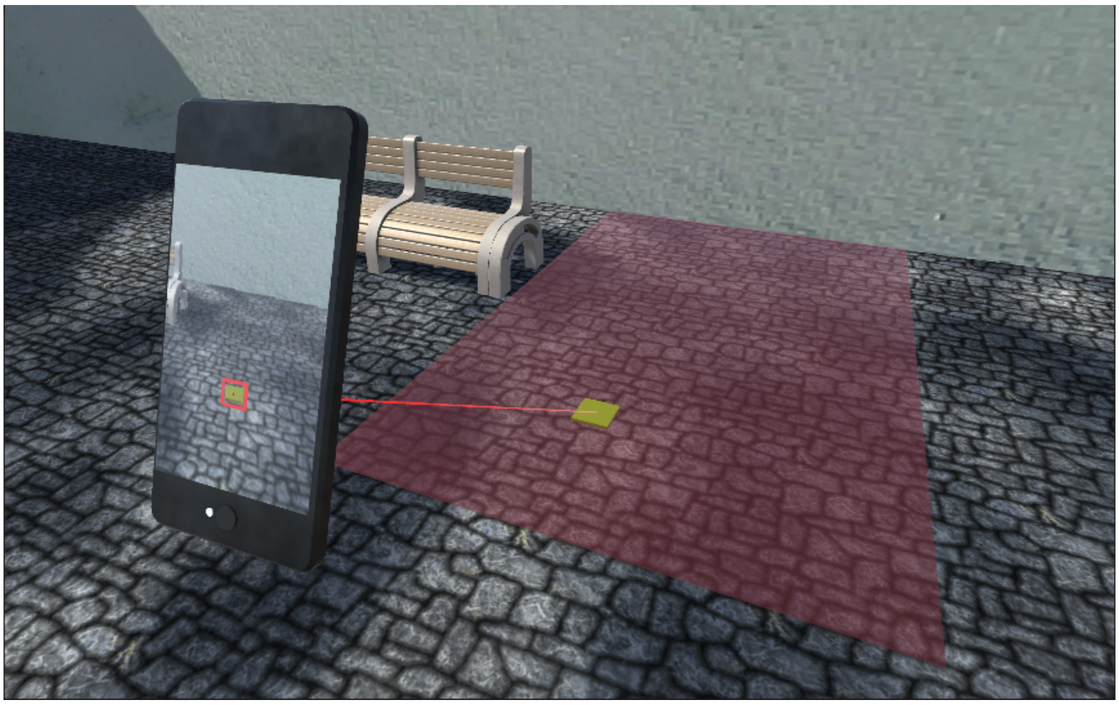


Figure 5.4: Locating a *Stolperstein* in 3D. The red square on the mobile device depicts the found bounding box for the *Stolperstein*, the red line the resulting possible positions in 3D. Through the intersection of the line with the ground plane (red plane), the position of the *Stolperstein* in 3D is determined

In the latter case, however, much more research has been done, leading to a better theoretical foundation and far more efficient algorithms. For this reason, we stick with the more traditional object detection approach of locating the whole bounding box for each *Stolperstein*.

5.4.2 Choice of Object Detection Algorithm

For the COCO benchmark alone, 164 different object detection algorithms have been submitted in the last three years[21], so the choice of the algorithm for our application not trivial.

We decided not to simply use the algorithm that scored highest on this benchmark, since most algorithms are very different in complexity and speed of inference and may be very good just for this particular problem, but cannot be readily adapted for our task. Therefore, we decided to focus only on theoretically grounded algorithms that have already proven themselves in many scenarios.

To this end, we pre-selected object detection algorithms that are further examined based on their appearance in recent academic studies on object detection such as [PNS20] or [Kur20]. We also paid special attention to the performance of the method in terms of inference speed, so that the user experience does not suffer from long waiting times for the next image or the predicted positions no longer match the real positions because the camera has moved too much in the meantime.

These criteria led us to closer examination of the following three object detection algorithms:

Faster R-CNN

Faster Region based Convolutional Neural Network (Faster R-CNN) [Ren+15], is a two-stage detector, evolved from its predecessors R-CNN [Gir+14] and Fast R-CNN [Gir15] with major improvements in inference time. It consist of a region proposal network, which is responsible for identifying potential objects and their bounding boxes, and a classification network, which decides whether and which object is in the proposed regions based on the features found in them. A visualization from the R-CNN paper [Ren+15], which works according to the same principle, is shown in Figure 5.5. Being a two-stage detector, the algorithm is still relatively slow compared to the other methods studied, despite its improvements in inference time.

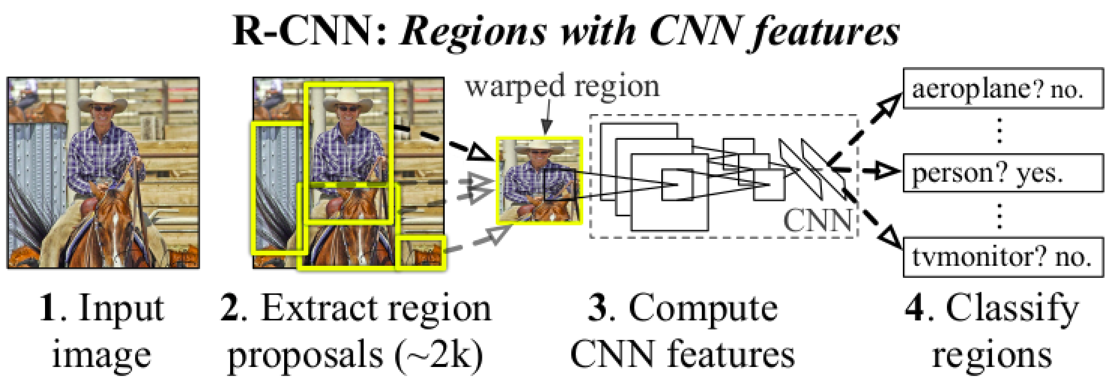


Figure 5.5: Object Detection with R-CNN, which functions according the same principles as Faster R-CNN

SSD

The Single Shot Detector (SSD) [Liu+16] is a one stage detector: Bounding boxes with corresponding labels are determined in a single pass through a Deep Neural Network (DNN) which produces scores and adjustments for default bounding boxes in the image.

This makes it very fast and simpler than conventional two-stage detectors such as Faster R-CNN. According to the author, it also outperforms Faster R-CNN in accuracy, but on the COCO benchmark, it actually performs worse than Faster R-CNN. For usage in resource-constraint environments, it is often combined with the mobilenetv2 [San+18] backbone network, a lightweight architecture suitable for mobile devices. The algorithm was the state of the art in object detection for a while, but has since been surpassed by other methods in terms of both speed and accuracy.

YOLOv4

The YOLOv4 [BWL20] object detector is the fourth generation of the YOLO [Red+16] detector family, which stands for You Only Look Once, since like SSD it is also a one-stage detector architecture.

The main focus of the YOLO architecture models lies on their speed, a tradeoff for the accuracy that other methods can achieve. According to the author of YOLOv3, [RF18] has already achieved an inference time three times faster than SSD, and the author of YOLOv4 reports a further 10% increase in FPS compared to YOLOv3. With reported 65 FPS on Tesla V100 GPU, this makes YOLOv4 undoubtedly the fastest algorithm among those examined. In terms of accuracy, YOLOv3 still was comparable with SSD and in some tasks clearly beaten by Faster R-CNN (such as in [Sin+21]). YOLOv4 manages to outperform both in accuracy, as can be seen, among other things, from work that compares these methods [KSP20], [FHL20].

Given this superiority in accuracy and especially speed over the other algorithms examined, we chose YOLOv4 as the detection algorithm for our application.

5.4.3 YOLOv4 Architecture

The YOLO architecture consists at its core of two parts: A backbone network, which performs feature extraction using convolutional layers, and a head, which parses and filters these features to arrive at the final bounding box predictions.

Backbone Network

The backone network is the resource consuming part of the object detector where the "learning" process happens. Since our application runs on mobile devices whose memory and computational capacity are limited, we decided to use a modification of YOLOv4 for devices with limited computational capacity, called YOLOv4-tiny. This architecture was released by the authors of the original YOLOv4 paper [Boc20] and only differs in the backbone network. It uses a compressed backbone network of only 20 instead of the original 110, which significantly reduces the inference time compared to the original YOLOv4 model, at the expense of some of its accuracy. The used backbone network architecture is shown in Fig. 5.6.

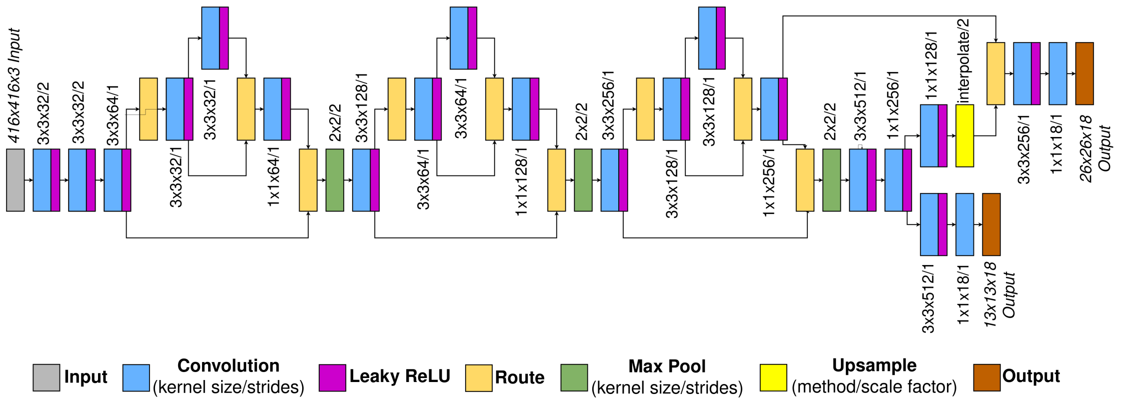


Figure 5.6: Architecture of the YOLOv4-tiny backbone network

The YOLOv4-tiny backbone outputs 2 tensors, each corresponding to a $S \times S$ grid of feature vectors, where each feature vector contains information about 3 possible bounding boxes at the region in the input image that corresponds to the grid cell. Fig. 5.7 illustrates the layout of such an output tensor. However, the first four elements of each box do not yet represent the box coordinates, and will be denoted as t_x, t_y, t_w, and t_h. The mapping to the box coordinates will be discussed in the next section. Our architecture uses a $S = 26$ grid for detecting smaller objects and a $S = 13$ grid responsible for detecting larger objects.

Prediction Head

The head is responsible for converting the grids of feature vectors produced by the backbone network into the final bounding box predictions. This is achieved in two steps:

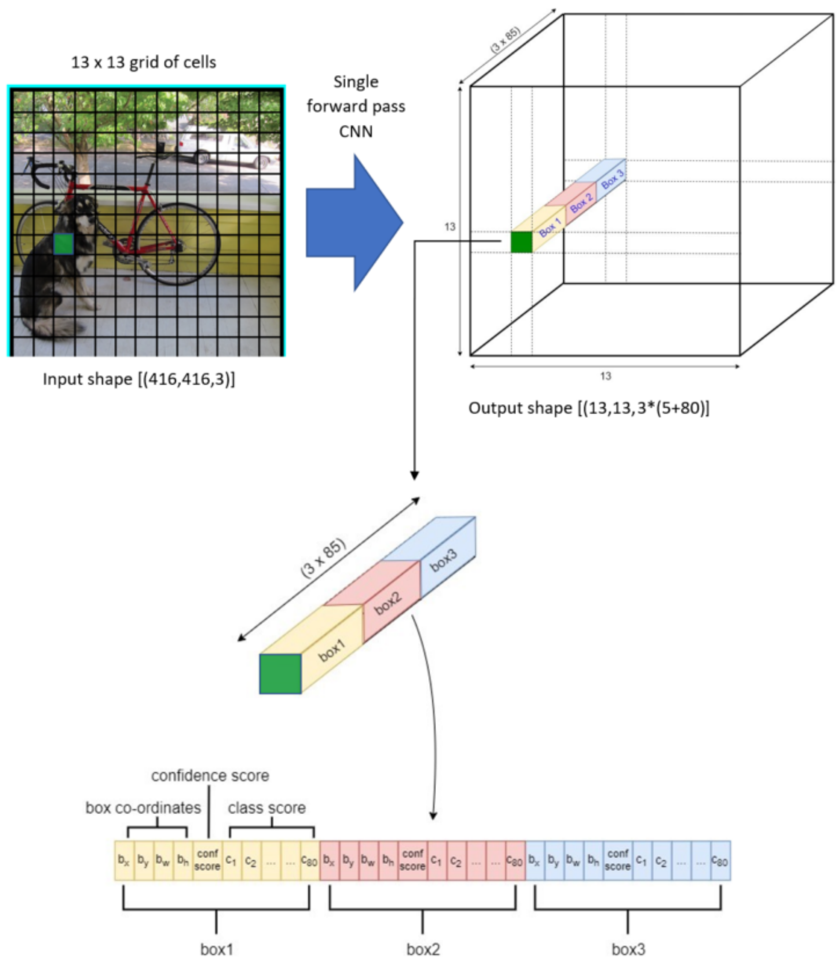


Figure 5.7: The layout of the backbone output [Sad21]. The elements in the bottom feature vector have already been mapped to box coordinates. In this example the model was trained on the 80 different classes of the COCO dataset, therefore 80 class scores exist.

Mapping to Box Coordinates In the first step, the feature vectors are mapped to the bounding box coordinates. As in the YOLOv3 architecture, a feature vector of one grid cell relates to 3 bounding box predictions for that cell. Each bounding box prediction is specified by its central x and y coordinates b_x, b_y with respect to the cell, its height and width b_w, b_h, a confidence score, and a class score for each class. The confidence score can be interpreted as the confidence of the model that there actually is an object at the position specified by the bounding box. The class score is a probability distribution over all classes that defines the network's prediction of which class the detected object belongs to. The layout of the vectors after mapping to the box coordinates can be seen in Fig. 5.7.

The way the box coordinates are determined for each cell is visualized in 5.8: To obtain the x and y coordinates b_x, b_y, the t_x and t_y output values of the feature vector are fit through a sigmoid function, resulting in their relative position in the cell. Multiplying these values with the cell width and height (omitted in the figure) and adding the cell offset c_y, c_y gives the absolute x and y center coordinates b_x, b_y of the bounding box in the image.

To arrive at the width and height b_w, b_h, t_w, t_h are exponentiated and multiplied by their respective anchors p_w, p_h. The anchors are precomputed priors for common bounding box sizes first introduced by YOLOv2 [RF17] and are determined by performing K-means clustering on the dataset. For this work, the anchors were not determined specifically for the used training dataset, but anchors obtained from the COCO dataset were reused.

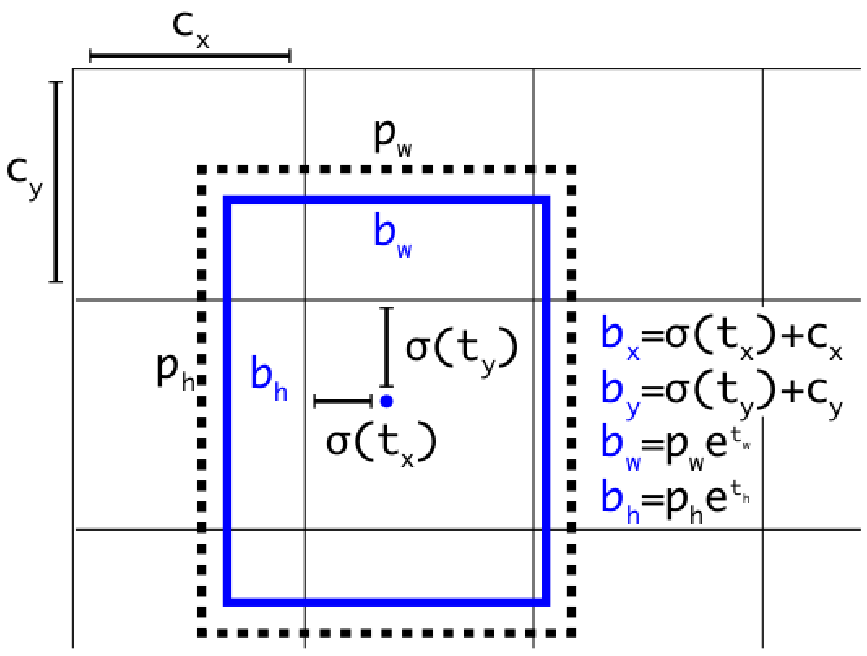


Figure 5.8: Calculation of the box coordinates for a single cell [RF17]

Filtering by Non-Maximum Suppression In the second step, irrelevant and overlapping bounding boxes are removed. To this end, the computer vision method of non-maximum suppression is used. Fig. 5.9 shows and example of bounding box predictions before and after applying non-maximum suppression.

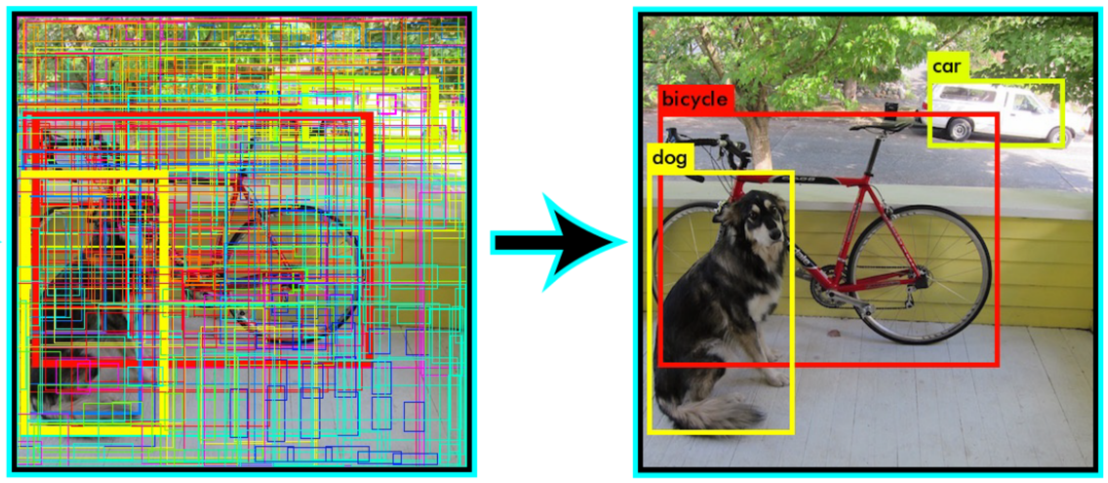


Figure 5.9: Results of filtering bounding boxes by non-maximum suppression [Pos19]. Low confidence and overlapping bounding boxes are filtered out.

To filter out irrelevant bounding boxes, the confidence score is multiplied by the class scores to yield a prediction score for each class per bounding box. If the highest prediction score does not exceed the selected prediction threshold, the prediction is discarded, otherwise the class with the highest score is chosen as the bounding box label. In our application, since we are only predicting one class, the label is always "Stolperstein". Through experimenting with different values, we found a prediction threshold of 0.4 to be yielding good results.

In order to remove overlapping bounding boxes for the same object, the Intersection over Union (IoU) $\frac{A \cap B}{A \cup B}$ for the areas A and B of two bounding boxes is used on the remaining bounding boxes. The boxes are removed using Algorithm 2 (with an IoU threshold of 0.5). The remaining bounding boxes are the final predictions of the model.

```
while unprocessed boxes exist do
    select the unprocessed box with highest prediction score.
    remove all the boxes whose IoU with the selected box exceeds the IoU threshold.
    mark the selected box as "processed"
end
```

Algorithm 2: Non-maximum suppression filtering by IoU [Zam19].

5.4.4 Acquisition of Training Data

In order to train a robust object detector for the *Stolpersteine*, labeled images from different *Stolpersteine* in different environmental conditions from different angles had to be collected. The usual approach for collecting such training data is to first gather the images and then label the ground truth bounding boxes manually for each image. However, to save us this time-consuming and tedious process, we created a tool for the recording of training data.

This tool is a separate Unity application that allows us to place markers at the positions of the *Stolpersteine* in AR by hand, which are subsequently tracked (See Fig. 5.10). The corner coordinates of these markers correspond to the coordinates of the *Stolpersteine* in AR space. We can then project these coordinates into Screen space to derive the coordinates of our bounding boxes in screen space. Once the *Stolperstein* markers are placed, the recording can be started, which accesses the camera frame twice per second and saves the frame along with the calculated bounding boxes at the moment of capture. This way, by moving around the *Stolpersteine*, a series of labeled images from different positions and view angles can generated with little effort. Using this method, we recorded a total of 3768 images from 55 different *Stolperstein* locations for training.

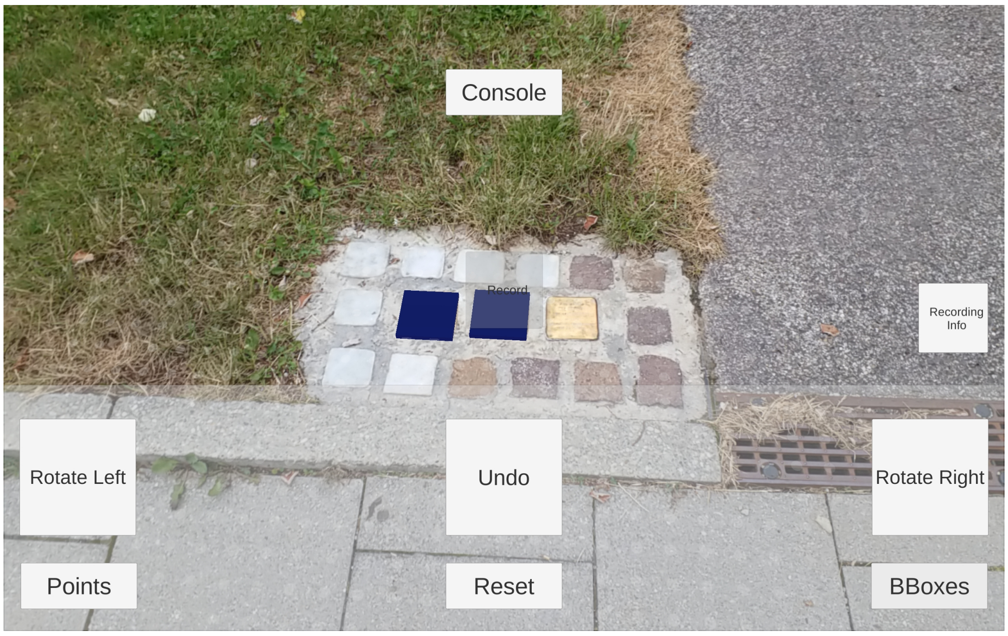


Figure 5.10: Placing *Stolperstein* markers with the training data recording tool

5.4.5 Training

We trained the YOLOv4 network using the Darknet [Red16] deep learning framework, the framework in which the YOLO models were originally developed. There are also inofficial implementations in more popular frameworks such as PyTorch and Tensorflow, but these usually do not reach the same level of accuracy as the official implementation in Darknet. In addition, the ease of use of the Darknet framework as well as the existence of detailed tutorials on how to train a custom YOLO object detector in this framework [Boc21], [Tec21], led us to the decision to use it for our purposes.

Before training, we initialized our DNN with the weights of a network that has already been successfully trained on the COCO dataset. This method is called pre-training and is based on the assumption that some parts of another object detection task are similar to ours (e.g., low-level feature detection), which speeds up the training significantly. For training, the dataset of collected and labeled *Stolperstein* images was split into 90% training and 10% validation data. Finally, rotation, saturation, exposure and hue image augmentation were performed to enlarge the training data.

Fig. 5.11 shows the learning curve obtained for training the final network with a batch size of 64 and a learning rate of 0.0001. The decreasing blue loss curve shows that the model makes fewer predictions for the training data over time, up to a point where no more improvement can be observed. How the networks capabilities to make predictions on new data is shown by the red mean Average Precision (mAP) curve. The increasing values indicate that the DNN is getting better at making predictions for the unseen validation data, showing that it is clearly learning to detect the *Stolpersteine*.

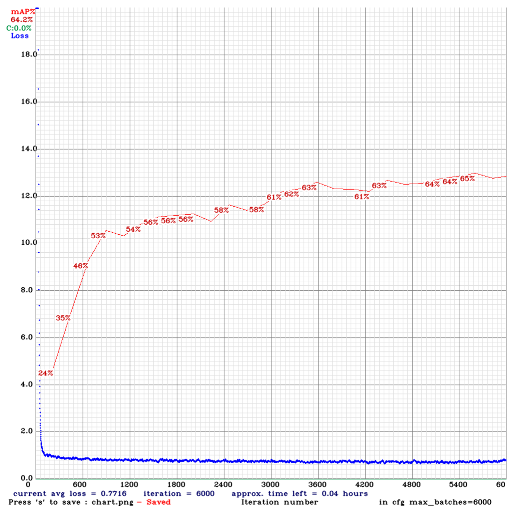


Figure 5.11: Learning curve for the trained DNN

5.4.6 Integrating the 2D Detection into Unity

In order to use the object detector for the mobile application, it has to be integrated into Unity to be used for inference. For this purpose, *Barracuda*[5], a "lightweight cross-platform neural network inference library for Unity" was used. Barracuda requires the DNNs in the *Open Neural Network Exchange (ONNX)*[6] format, a common file format that allows interchangeability of Deep Neural Network between different deep learning frameworks.

Therefore, the weights of the network trained in Darknet, as described in 5.4.5, were exported and converted to ONNX using the pytorch-YOLOv4[7] git repository. Since the resulting ONNX model also implements the YOLO head (described in Section 5.4.3), which contains operators not supported by the Barracuda version used, the model head was removed using a custom onnx-graphsurgeon[8] script. The resulting backbone ONNX model was then imported into Unity.

The YOLO head was implemented directly in C# in Unity, as described in 5.4.3, based on a tutorial for YOLOv2[9] to obtain the predicted bounding boxes from the feature tensors provided by inference of the ONNX model in Barracuda.

Unfortunately, when the object detector is run in the Unity mobile application, real-time inference is no longer possible, unlike running it on a PC. This is due to the overhead caused by Unity, which is not originally designed to run Deep Neural Networks, and the limited capacities of the mobile device. Nevertheless, we observed inference times between 0.3 and 0.5 seconds on our test device (*Samsung Galaxy Tab S4* from 2018), which is quite acceptable for our application since the detection only has to be executed once in the best case.

5.4.7 The Stolperstein Detector Component

The preceding sections have explained how the 2D object detector was trained and integrated into Unity. In the following the final Stolperstein Detector component of the mobile application is described, which uses this object detector to localize the *Stolpersteine* in 3D space.

[5]`https://docs.unity3d.com/Packages/com.unity.barracuda@2.0/manual/index.html` [accessed 2021-08-05]

[6]`https://onnx.ai/` [accessed 2021-08-05]

[7]`https://github.com/Tianxiaomo/pytorch-YOLOv4` [accessed 2021-08-05]

[8]`https://github.com/NVIDIA/TensorRT/tree/master/tools/onnx-graphsurgeon` [accessed 2021-08-05]

[9]`https://docs.microsoft.com/en-us/dotnet/machine-learning/tutorials/object-detection-onnx` [accessed 2021-08-05]

The Stolperstein Detector, when triggered, executes the object detector with the latest camera image cropped and scaled to a resolution of 416×416. The object detector returns the coordinates of the bounding boxes of the *Stolperseine* in screen space. Raycasting for the center of each detected bounding box against the detected ground plane determines the assumed position of the *Stolperstein* on the pavement, as explained in Section 5.4.1. For the ground plane detection, an out-of-the-box method offered by the ARFoundation framework is used, which is already started in the Stolperstein Area Handler as soon the *Stolperstein* area is entered. This guarantees a bit of device movement—which is necessary for determining the ground plane—when the user walks the last few meters to the *Stolperstein*.

It can happen, though rarely in practice, that an object which is not a *Stolperstein* is detected as such. To catch these cases, plausibility checks are performed on the predictions. First, it is ensured that exactly as many *Stolpersteine* are detected as are expected for this location. Furthermore, it is ensured that they are all on the same ground plane and none is located unrealistically far from the others.

Should no *Stolpersteine* be detected or the detection refused by the plausibility checks, a new detection is performed at 3-second intervals until a detection was successful. This interval was chosen because it still allows for a smooth user experience with only occasional short delays when performing the detection. As soon as the detection was successful, the found positions of the *Stolpersteine* are returned to the Stolperstein Area Handler.

It should be noted that the center of the bounding box for a *Stolperstein* on the screen does not exactly match the center of the *Stolperstein* in 3D when viewed at an angle due to the distortion caused by the perspective projection of the camera. However, testing revealed that this deviation is so small in our use cases that it is not really necessary to eliminate this effect by calculations.

5.5 Content Constructor

The content constructor creates the AR content for a *Stolperstein*, based on the text, coordinate and image data retrieved from the Backend Interface. This way, the AR

The complete content is constructed once, but only a subset is visible at any one time to avoid overwhelming the user with information or cluttering the screen. We will refer to such as subset of content that is shown at a time as “scene”. Which scenes are available for a *Stolperstein* depends on the data that is provided for this *Stolperstein*. By interacting with the 3D objects in a scene, the user can switch to another scene and always return to the previous scene via a button.

If the corresponding information is available, the following scenes are supported:

Base Scene

The Base Scene remains visible when other 3D scenes are selected. It consists of a photo from the person (if available) displayed AR at head height over the *Stolperstein* so that has the opportunity to "look the victim in the eye". Below that, the victim's name and date of birth and death are displayed as 3D text. A screenshot of the application with the base scene is shown in Fig. 5.12.

Selection Scene

The Selection Scene, also seen in 5.12, is the first scene the user is presented with once he or she has chosen a *Stolperstein*. For each of the following scenes, if available for this *Stolperstein*, it augments a 3D icon in a circle around the *Stolperstein*. These 3D icons represent the content that can be accessed through them, e.g. touching a three-dimensional "i" takes the user to the Information Scene. When you point the device at one of the icons, they are enlarged and accompanied by text indicating what information is displayed when you touch the icon.

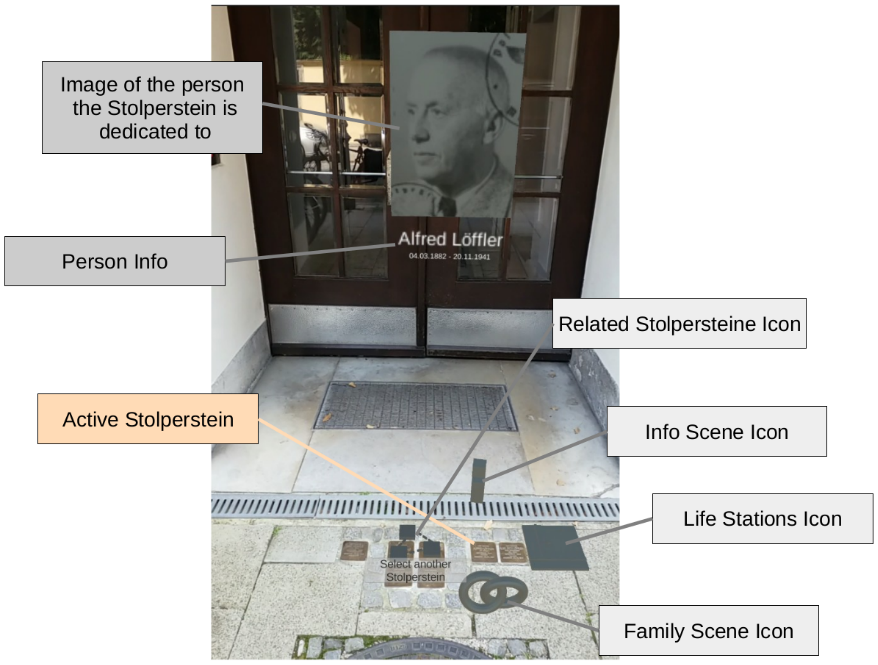


Figure 5.12: Base and Selection Scene. Augmented elements from the Base Scene are indicated through darker grey boxes, augmented elements from the Selection Scene are indicated by lighter grey boxes.

Info Scene

The Info Scene consists of a text overlay on the screen that conveys general information about the person that the *Stolperstein* is dedicated to. So far only text is supported, but could be extended in the future to also feature images and touchable cross-references that lead to another scene or a glossary.

Life Stations Scene

Selecting the Life Station Scene augments a 3D world map on the ground behind the *Stolperstein*. Markers are placed on the map at the coordinates of important stations in the victim's life. Clicking on such a marker opens a text window where the life station is elaborated. The map is presented in two styles, which are determined after the required zoom level has been calculated: For a highly zoomed-out map (e.g. with different cites or even countries), the map is displayed with satellite imagery and elevated terrain. A screenshot from the application with the Life Station Scene on a zoomed-out map is shown in Fig. 5.13. For zoomed-in maps (at city level), street data imagery on a flat terrain with 3D buildings is used.

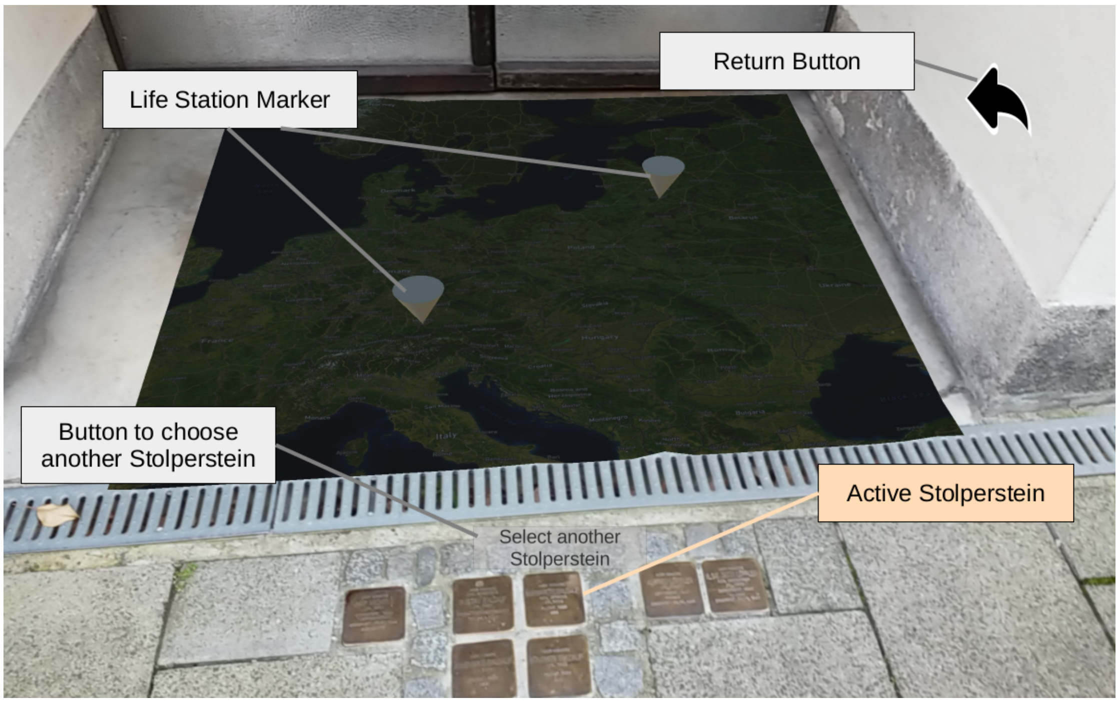

Return Button
Life Station Marker
Button to choose another Stolperstein
Active Stolperstein
Select another Stolperstein

Figure 5.13: Life Stations Scene

Family Scene

The Family Scene is a text scene similar to the Info Scene. It contains textual information about family members of the victim and could in the future be extended through images, cross references and possibly a family tree displayable in 3D.

Related Stolpersteine Scene

This scene contains information about other *Stolpersteine* related to the currently visited one. To display this information, a map is augmented on the ground, featuring the related *Stolpersteine* at their positions as well as the current position of the user. This map is usually at the city level, displaying buildings in 3D as well as markers representing the *Stolpersteine* as in Fig. 5.14. However, if the related *Stolpersteine* are far away a map with elevated terrain and satellite imagery is used as well. Again, touching the *Stolperstein* marker reveals more about the relation to the other *Stolperstein* in text, such as those dedicated to a family member, a friend, a work colleague, or a person deported to the same destination.

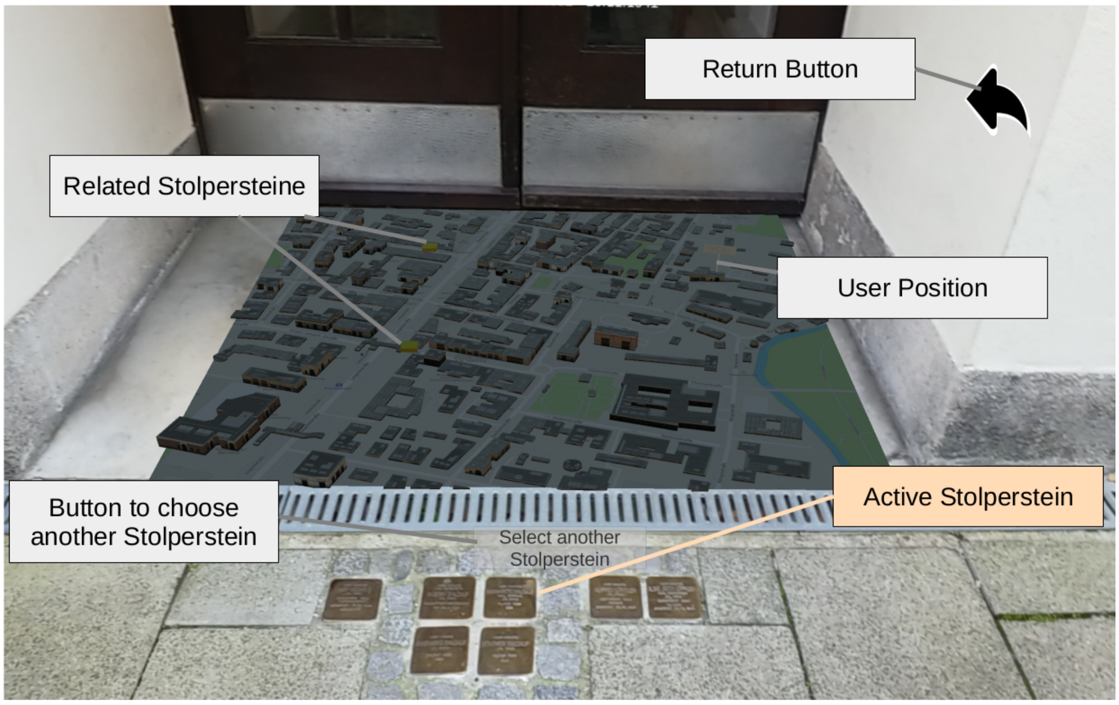


Figure 5.14: Related Stolpersteine Scene

To get a better idea of how the scene with the Augmented Reality compares to the situation in the physical world, we have included a third-person image of the application in action in Fig. 5.15.

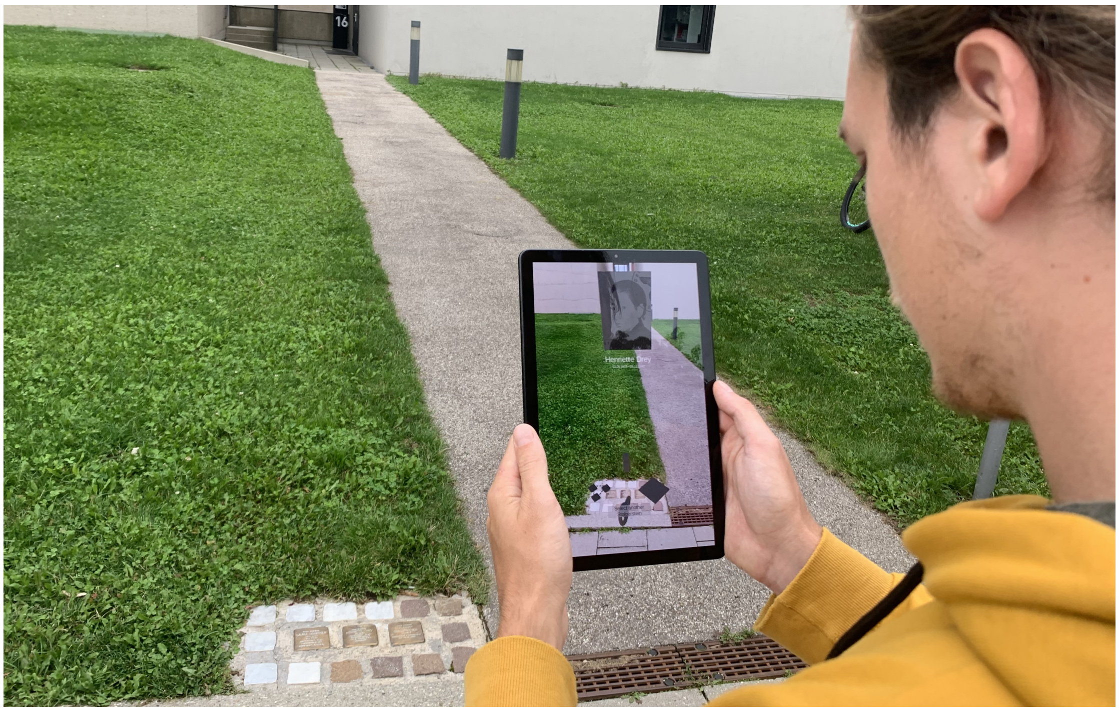


Figure 5.15: Third-person view of the mobile application in use

5.6 Backend Interface

The Backend Interface allows the other components to access the data stored in the Backend. This is achieved by performing HTTP request to the Backend, which returns a HTTP response containing a string in JSON format with the requested information that then can be parsed by the application. The Backend Interface initially requests and stores all *Stolperstein* locations such that these can be processed by the Location Handler. Furthermore, once informed that a *Stolperstein* area is entered, the information regarding the *Stolpersteine* at this location is requested, as well as the ordering information for these. To ensure a smooth data exchange, a stable internet connection is required for this component.

5.7 Backend

The Backend is implemented in Python using the *Django* framework[10] and is supposed to run on a server, independently from the application. It consists of a Web Interface that allows authorized data administrators to create and manage entries for *Stolpersteine*, a database that stores this information, and an API contact point for the mobile application to retrieve this information.

5.7.1 Backend Logic and API

The backend logic processes and validates the input of new information coming from the Web Interface and enters it into the Database. It also retrieves information requested by calls to the API.

Authorization Django's authentication and authorization system was used, in order to ensure that only authorized persons create and edit *Stolpersteine*. This means, that logging in with an account that has been assigned a group with the required access permissions, in order to create or modify a *Stolperstein* is required.

Two different access authorization groups have been implemented:

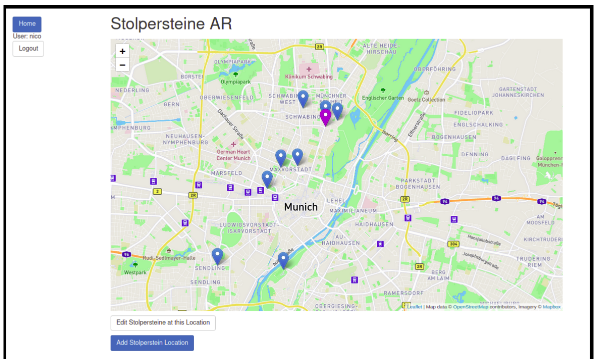
- **Editors:** Data administrators who have this group can create new *Stolperstein* locations and add or edit *Stolpersteine* at these locations. However, they do not have access to *Stolpersteine* at locations created by others.
- **Global Editors:** Holders of this group can create, edit and remove any *Stolperstein*. This group is intended for administrative persons.

API The API was implemented using the Django REST framework[11] and returns the data for HTTP requests in form of JSON strings. It answers to requests regarding the coordinates of all *Stolperstein* locations as well as requests concerning the information associated to the *Stolpersteine* at a location. For imagery data, only a reference is sent so that it can be requested by a seperate query if needed. Using the API requires no authorization, as it only enables the retrieval of data, that is publicly available anyway through use of the application.

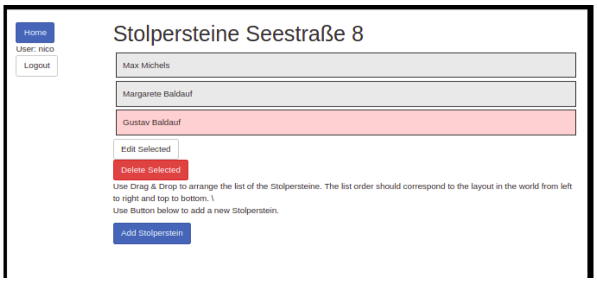
[10]`https://www.djangoproject.com/` [accessed 2021-08-05]
[11]`https://www.django-rest-framework.org/` [accessed 2021-8-05]

5.7.2 Webinterface

On the main page of the website, the data administrator can decide whether to create a new *Stolperstein* location by specifying the exact location on the map, or to select an already registered *Stolperstein* location (see Fig. 5.16a). Both actions bring him/her to a page offering an overview over the *Stolpersteine* at that location, as shown in Fig. 5.16b. Here the data administrator can add and remove *Stolpersteine*. By rearranging the list of *Stolpersteine*, the order of their placement in the ground can be specified, which is needed by the Stolperstein Detector (see 5.4) to distinguish between the *Stolpersteine*. Adding a *Stolperstein* is implemented by a form through which data can be added by entering text, uploading images, or selecting a point on a map for coordinates, as shown in Fig. 5.16c. Here only the general information field and the name of the associated person are required, all other fields are optional and only need to be filled in if the appropriate information is available.


Home
User: nico
Logout
Stolpersteine AR
Munich
Edit Stolpersteine at this Location
Add Stolperstein Location

(a) Main Page



Stolpersteine Seestraße 8

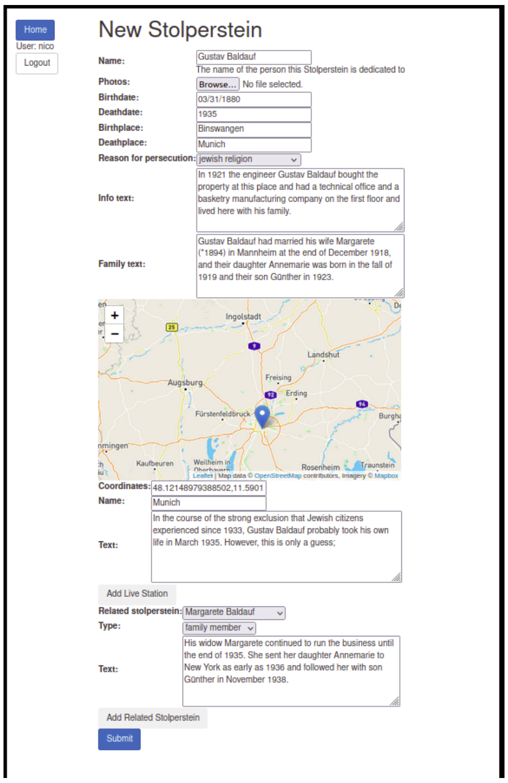
Max Michels

Margarete Baldauf

Gustav Baldauf

Use Drag & Drop to arrange the list of the Stolpersteine. The list order should correspond to the layout in the world from left to right and top to bottom. \
Use Button below to add a new Stolperstein.

(b) Stolperstein Location Overview Page



New Stolperstein

Name:	Gustav Baldauf The name of the person this Stolperstein is dedicated to
Photos:	Browse... No file selected.
Birthdate:	03/31/1880
Deathdate:	1935
Birthplace:	Binswangen
Deathplace:	Munich
Reason for persecution:	jewish religion
Info text:	In 1921 the engineer Gustav Baldauf bought the property at this place and had a technical office and a basketry manufacturing company on the first floor and lived here with his family.
Family text:	Gustav Baldauf had married his wife Margarete (*1894) in Mannheim at the end of December 1918, and their daughter Annemarie was born in the fall of 1919 and their son Günther in 1923.

Coordinates:	48.12148979388502,11.5901
Name:	Munich
Text:	In the course of the strong exclusion that Jewish citizens experienced since 1933, Gustav Baldauf probably took his own life in March 1935. However, this is only a guess;

Related stolperstein:	Margarete Baldauf
Type:	family member
Text:	His widow Margarete continued to run the business until the end of 1935. She sent her daughter Annemarie to New York as early as 1936 and followed her with son Günther in November 1938.

(c) Adding a new Stolperstein

Figure 5.16: Screenshots of the Web Interface.

5.7.3 Database

For storing of the data concerning the *Stolpersteine*, a SQLite3[12] database was chosen, since it is free, open-source, lightweight and easy to set up. At the current amount of data stored per *Stolperstein* (which is mostly text), it would theoretically have enough capacity to store information about every of the over 80,000 *Stolpersteine*. Interaction with the database, in the form of entering and retrieving data, is handled by the backend logic (see Section 5.7.1.

An overview over the *Stolperstein* data stored in the Database is provided by the relationship diagram created with DBVisualizer[13] in Fig. 5.17.

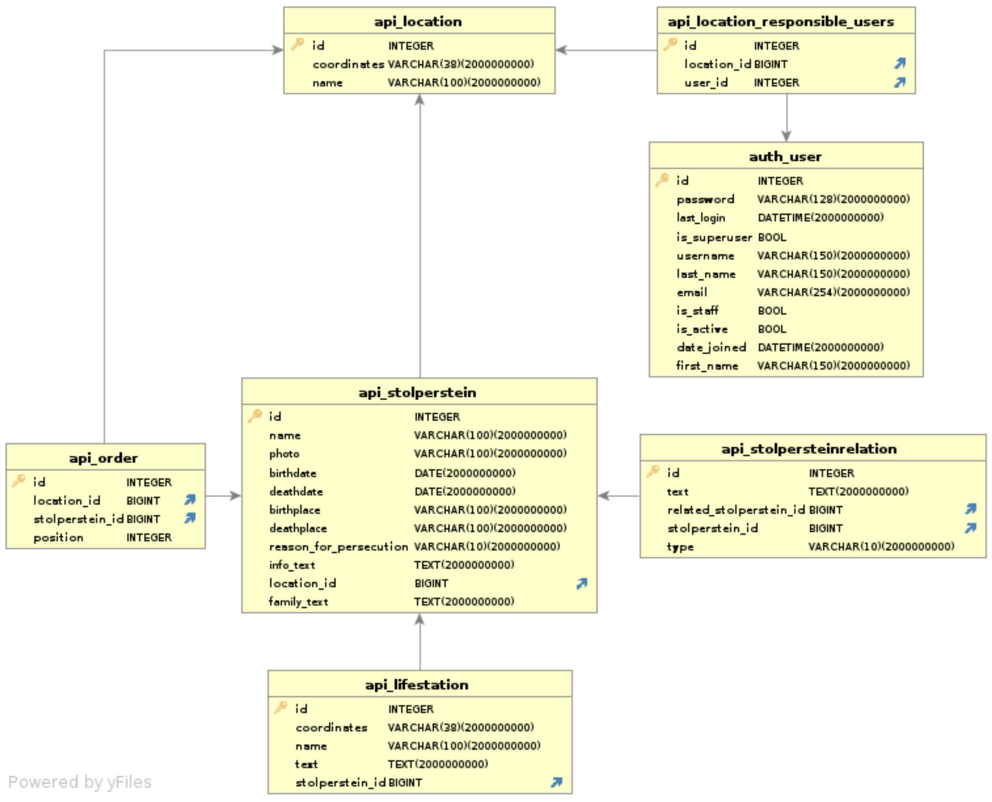

api_location
id INTEGER
coordinates VARCHAR(38)(2000000000)
name VARCHAR(100)(2000000000)
api_location_responsible_users
id INTEGER
location_id BIGINT
user_id INTEGER
auth_user
id INTEGER
password VARCHAR(128)(2000000000)
last_login DATETIME(2000000000)
is_superuser BOOL
username VARCHAR(150)(2000000000)
last_name VARCHAR(150)(2000000000)
email VARCHAR(254)(2000000000)
is_staff BOOL
is_active BOOL
date_joined DATETIME(2000000000)
first_name VARCHAR(150)(2000000000)
api_order
id INTEGER
location_id BIGINT
stolperstein_id BIGINT
position INTEGER
api_stolperstein
id INTEGER
name VARCHAR(100)(2000000000)
photo VARCHAR(100)(2000000000)
birthdate DATE(2000000000)
deathdate DATE(2000000000)
birthplace VARCHAR(100)(2000000000)
deathplace VARCHAR(100)(2000000000)
reason_for_persecution VARCHAR(10)(2000000000)
info_text TEXT(2000000000)
location_id BIGINT
family_text TEXT(2000000000)
api_stolpersteinrelation
id INTEGER
text TEXT(2000000000)
related_stolperstein_id BIGINT
stolperstein_id BIGINT
type VARCHAR(10)(2000000000)
api_lifestation
id INTEGER
coordinates VARCHAR(38)(2000000000)
name VARCHAR(100)(2000000000)
text TEXT(2000000000)
stolperstein_id BIGINT
Powered by yFiles

Figure 5.17: Relationship diagram of the Database. Tables automatically generated by Django have been excluded for visibility (with the exception of the auth_user table).

[12]`https://sqlite.org/index.html` [accessed 2021-8-05]
[13]`https://www.dbvis.com/` [accessed 2021-08-05

6 Evaluation

In order to get an assessment of the quality of the user experience of our application and obtain directions for the further improvement of the application, a user study with 30 test users is to be conducted.

However, due to the circumstances described in 6.3 the study was not carried out and only informal feedback was gathered. Sections 6.1 and 6.2 therefore describe how the evaluation would have been conducted under normal circumstances.

6.1 Method

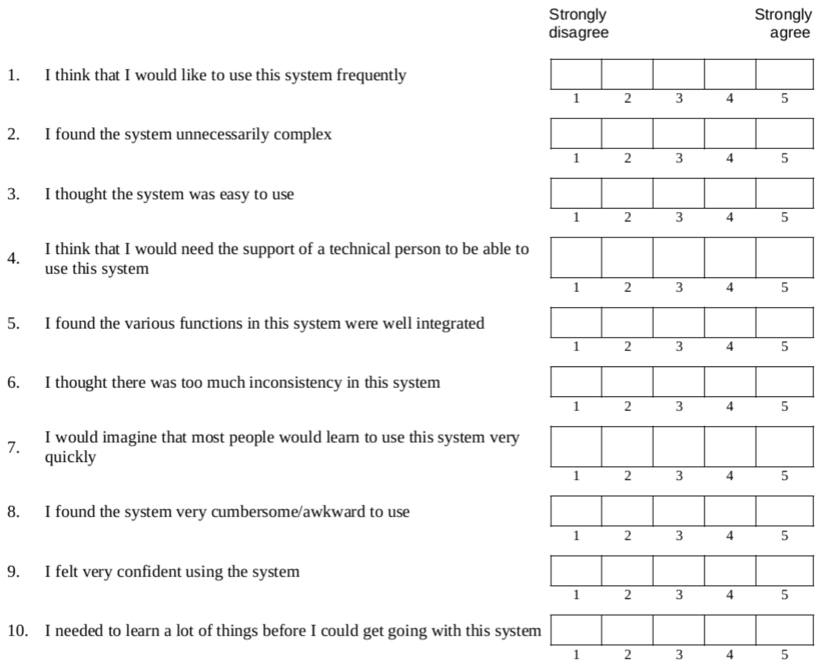
There is no general agreement for the best evaluation method for AR systems [DB11]. However, well established scales for measuring the usability of a system exist, such as the NASA Task Load Index (NASA-TLX) [HS88] and the System Usability Scale (SUS) [Bro96].

Specifically for Handheld Augmented Reality applications, there is the Handheld Augmented Reality Usability Scale (HARUS) [San+14], which consists of 8 questions regarding manipulative measures (the ease of handling the system) and 8 questions regarding comprehensibility measures (the ease of understanding the presented information). This scale was created based on user-reported issues frequently encountered in HAR systems. In these HAR systems, the issues often were related to the device itself, leading to questions in the HARUS , such as "I felt that using the application was comfortable for my arms and hands" or "I found the device difficult to hold while operating the system". However, as we are developing our application as a smartphone application, these questions are of no great use to us, as we have no direct influence on the device the user will ultimately use for the application (apart from during the user study). This renders most of the questions in the "manipulative measures" category useless and we therefore decided against using the HARUS scale.

In order to decide between the NASA-TLX and the SUS , the questions asked were more closely examined. The NASA-TLX contains questions on mental, physical and temporal demand as well as user performance, effort and frustration, whereas the SUS uses a subjective measurement of the perceived effectiveness, efficiency and satisfaction of the system. We found the latter to provide more valuable insights for our application, since factors like user performance and physical or temporal demand do not play a significant role in our application. Therefore, we opted for using the System Usability Scale. Since problems for non-native speakers with the word "cumbersome" have been reported, we chose the adapted version for non-native speakers [Fin06]. A complete SUS questionare as it is used is depicted in Fig. 6.1. It contains a total of ten questions, which can be marked from 1 "strongly disagree" to 5 "strongly agree". From the questions a single usability score between 0-100 (where higher is better) can be obtained, which allows comparison between applications.

Since the SUS is quite general and some more application specific feedback is desirable, the following four custom questions were added to the questionnaire (not contributing to the SUS score):

- "I think that the application helps in the commemoration of the victims"
- "I consider the application appropriate for the serious topic"
- "I think the realization as Augmented Reality App adds value to the application"
- "I found the placement and tracking of virtual objects convincing"



		Strongly disagree				Strongly agree
1.	I think that I would like to use this system frequently	1	2	3	4	5
2.	I found the system unnecessarily complex	1	2	3	4	5
3.	I thought the system was easy to use	1	2	3	4	5
4.	I think that I would need the support of a technical person to be able to use this system	1	2	3	4	5
5.	I found the various functions in this system were well integrated	1	2	3	4	5
6.	I thought there was too much inconsistency in this system	1	2	3	4	5
7.	I would imagine that most people would learn to use this system very quickly	1	2	3	4	5
8.	I found the system very cumbersome/awkward to use	1	2	3	4	5
9.	I felt very confident using the system	1	2	3	4	5
10.	I needed to learn a lot of things before I could get going with this system	1	2	3	4	5

Figure 6.1: System Usability Scale (SUS) for non-native speakers

6.2 Study Procedure

The study is to be carried out for one Stolperstein location, at the Egyptian Museum in Munich where the *Stolpersteine* for Laura Dobriner, Henriette Drey and Dr. phil. Ernst Darmstaedter are installed. Since collecting information for one site already requires a considerable amount of effort and is not the focus of this work, only one site was selected for the user study.

The survey is conducted in the following procedure, for one test user at a time:

1. The user fills out the demographic questionnaire.
2. The user is informed about the Stolpersteine project.
3. The purpose of the application is briefly explained to the user.
4. The Application is opened on the test device and handed over to the user outside the radius of the *Stolpersteine*.
5. The User uses the application as long as he/she wishes.
6. The User fills out the SUS questionare and and the additional questions.
7. Remaining open questions are answered and further feedback is written down.

All questions that occur during the study are written down. Questions concerning the usage of the application can be directly answered, otherwise they are answered after filling out the survey.

The needed time for each test subject is estimated at around 10-15 minutes for the whole process. As test device a *Samsung Galaxy Tab S4* is used.

6.3 Special Circumstances

On March 11, 2020 the coronavirus (COVID-19) outbreak was declared a global pandemic by the World Health Organisation [CV20]. This status still holds as of July 2021 and the public is urged to avoid direct contact as much as possible.

However, being in an early stage of the development, where the first user tests are performed, the direct contact to the test persons, in order to closely help and monitor them is of great importance. This, and the increased difficulty of finding unknown volunteers willing to participate in a site-based user survey under these circumstances led us to the decision, to only perform an informal evaluation.

This means that the application is only tested by a small number of friends and we will not collect any statistical data via the questionnaire afterwards. Instead, only oral feedback will be collected.

6.4 Results and Feedback

Feedback on the application was collected from five friends on four different days. Three of the participants were male and two female. Two of the participants were already very familiar with AR. Due to the informal nature of the survey and the small number of participants, no statistical analysis of the questionnaire is carried out. The most common observations and feedback from the informal user evaluation are summarized below.

Usability Most participants found all features themselves and stated that the application is intuitive. However, it was also noted several times that a small help/info button explaining to the user how to interact with the application would be useful. One point that all participants commented on was that the AR map was too dark, some participants also found other user interface (UI) elements too dark. Besides that it could be observed and was mentioned, that some common UI elements were missed, like a symbol indicating the possibility to expand the map or the option to scroll down on a text field.

The amount of information displayed was very well received. Especially for the texts it was mentioned, that they conveyed sufficient information, while not being too long such that they would loose the users attention.

Detecting and Tracking The Detection of the *Stolpersteine* was reported to work very well. The markers placed on the *Stolpersteine* were mostly at the correct locations and only sometimes a few centimeters off, which generally was not noticed by the users. A wrong initial placement of the AR content was only encountered once, when the ground plane was detected too high, such that the content was floating in the air.

Regarding the tracking, once the *Stolperstein* positions were localized, two people experienced a sudden shift of the AR content by a some centimeters, which might be have been related to the lighting conditions. Apart from that the tracking was relatively stable. One participant with an background in AR also explicitly mentioned, that he was surprised about how good the detection and tracking worked.

Application Concept All participants found the application adequate for the topic and mentioned that the information is incorporated in a meaningful manner. However it was partly noted that the question about the adequateness of the application would best be discussed with the persons concerned, such as relatives of the victims that the memorials are dedicated to. The incorporation of AR for the application was seen as very positive, as participants felt this led to a more immersive and engaging experience with the information than just reading a text on an info board. Especially the possibility to see a picture of the victim in 3D at head height and visualization of the related *Stolpersteine* in the area on a 3D map were highlighted by the participants. Many also saw more possibilities for the application of AR within this application and positively reacted to presented feature ideas for future development described in Section 7.1.

Summary Overall, regarding the informal evaluation, the application was well received. The detection and tracking seem to function at a satisfactory level, but there is still room for improvement in the user interface (UI) design. The general concept and its implementation was approved and especially people who were not yet familiar with AR were surprised and pleased with this way of presenting the information.

However, this evaluation must be taken with caution due to its informal nature and the bias of the participants, which must be assumed. Therefore, to confirm or refute these results as a next step a formal evaluation with more and unbiased participants is needed.

7 Future Work

Although the application is fully functional, there are certainly elements that could be extended and improved. Apart from these, which are listed in the following sections, a full-scale evaluation as soon as circumstances allow would be of great importance to gain more certainty about which parts should be given the most attention.

7.1 Features

There are multiple features that could not be introduced into the application due to the limited time available.

7.1.1 Displaying Near Stolpersteine in AR

So far, if the user is not close to a *Stolperstein*, he or she can consult a map to view the location of other *Stolpersteine* nearby. An interesting feature would be to additionally show the *Stolpersteine* in the immediate vicinity on the display in AR, as is done in Bombsight (see 3.1.2) with the Bomb Locations. This would make the application more immersive, possibly make navigation easier, and also show the large extent of former residences of Holocaust victims that can be found in many cities. This feature might be extended further to allow users to select a *Stolperstein* on the map or in AR to augment footsteps on the ground leading to this *Stolperstein*.

7.1.2 Content Layout Based on Space Available

It is not unusual for *Stolpersteine* to be installed directly in front of a house entrance or a gateway. This means that the augmentations in the current implementation would be augmented partly at positions that in reality are covered by a wall or door, thus destroying the illusion. However the ground plane, which is detected anyway for the localization of the *Stolpersteine* (see 5.4.7), could be used to circumvent this problem. By dynamically arranging augmented objects so that they are always at positions on or above the ground plane, it is possible to ensure that augmentations remain believable. We consider this a high priority feature for the further development of the application

7.1.3 Additional Information Content

Neighbourhood Information Further types of Information about person and location could be offered to the user. This could include information about the neighbourhood and its relation to the person. An idea would be to enter locations close by through selection on a map, together with an information text and/or a picture through the web interface. In the application, clicking on the "Neighbourhood" Icon would then allow the user to see markers displayed in world space at these locations when looking around. Clicking on such a marker would then reveal the information text and/or picture associated with that location. This idea especially received positive feedback when explained during the evaluation.

Audio Content Further the integration of audio content could also be supported. This could include original audio recordings, such as witness accounts, as well as audio recorded by the person creating the content for the *Stolperstein*, e.g. to replace or support a text section.

Glossary In addition, to keep the information texts short, terms that are not familiar to all users, such as “diaspora”, could be linked to a glossary explaining them in more detail. This glossary would not have to be limited to textual references, but could also include additional historical background information, such as more detailed content about places or events mentioned.

7.1.4 Using Information from the Web

While in the vast majority of cases detailed information about individual stumbling stones is not publicly available, there are many websites that already contain some information about the *Stolpersteine* and the corresponding victims. Unfortunately, there is no general database with all *Stolpersteine*, but in many cities associations and organizations keep a list of *Stolpersteine* with some basic information about the persons on their websites.

Future work could explore the possibilities to systematically extract this information, so that without adding content by external persons, many *Stolpersteine* could already be covered by the application with some basic information. This information could then still be expanded and supplemented via the web interface by researchers or relatives.

7.2 Performance

While some aspects of performance such as ground level tracking, GPS accuracy, and object detection time will naturally improve as better hardware becomes available over time, there are other areas that need active improvement in the application.

7.2.1 Tracking

The biggest technical problems so far have been with tracking, as the tracked objects sometimes shifted unexpectedly. Since an out-of-the-box method provided by Unity was used and tracking in changing environments (e.g. due to shadows) is generally considered difficult, we cannot present any concrete improvement options so far.

However, the effect of suddenly moved objects could be mitigated by restarting the detection when the AR content is no longer in the right place. The renewed detection could either be requested by the user by touching a button or automatically determined by continuously performing detection at intervals and comparing its results with the actual position of the AR content.

7.2.2 Stolperstein Detection

The remaining inaccuracy in the *Stolperstein* detection stems from two factors: the inaccuracy of the Object Detection itself, and the inaccuracy that is introduced when the device is moved between the start of inference and the arrival of the results.

Based on the results of tests and evaluation, we consider the former to be so minor that it does not require special attention for the time being. The latter was not a problem in the evaluation, as the users used the application properly and without jerky movements, but the application should optimally also be robust against misuse.

Therefore, to reduce the inaccuracy caused by movement of the device, the Inertial Measurement Unit (IMU) data of the device could be read to determine the current acceleration of the device. This data could be used to ensure that detection is only performed when the device is held relatively still, and results are discarded if large movements occur during inference. In addition, the expected shorter inference time with newer hardware will also allow for less movement between the start and end of inference, contributing to higher stability of the detection results when the device is moved.

7.3 User Interface

A big part that contributes to the user experience is the user interface. In the following, future work for the user interface of the mobile application as well as the web interface for entering information is described.

7.3.1 Webinterface

The design of the web interface is very rudimentary, as it primarily fulfills the function of entering information and is not used by the normal user, but only by those who are responsible for adding information about a *Stolperstein*. Nevertheless, it should be made more visually appealing and also include some guidance for new users on how to orient themselves. Furthermore, functional improvements could be introduced, such as the search for specific locations via text input. In order to find out exactly what improvements should be made, a user survey should be conducted for the web interface in the future.

7.3.2 Mobile Application

The planned improvements for the user interface result largely from feedback from the informal evaluation. Here, an info or help button would be welcomed by users that operate the application for the first time. In addition, the 3D information icons could be improved and modeled in more detail. In particular, the screen overlay UI (e.g., information texts) could be made more visually appealing, and common icons and features that users are familiar with from various applications, such as the ability to scroll down in the text, should be supported. Finally, addressing concerns about the visibility of objects, especially the 3D map, is a high priority.

8 Conclusion

In this work, a mobile application was developed that enhances the *Stolperstein* memorials with information presented via Augmented Reality. Alongside the application, a web interface was also created, through which information can be easily entered into the application.

The realization of the application was achieved by combining GPS data and a method developed by us for the detection and localization of the memorials in 3D space using object detection. The implementation of the method yields positive and consistent results and is considered very satisfactory by us. However, in the following tracking of the localized memorial positions, based on out-of-the-box methods, there are still noticeable inaccuracies in some cases.

An evaluation of the application was performed, which, due to the circumstances of the Covid-19 crisis, was only conducted on a small, informal scale. From the feedback, it appears that the application was well received by users and seems to be fulfilling its purpose of providing information about memorial victims in an interactive and novel way. User feedback also suggests that implementing Augmented Reality proved to be a good idea, as users with little experience using AR in particular reported increased interest in the content when it was provided through these means.

Accordingly, we have developed an application with great potential appealing both to the community of users interested in more information about the *Stolpersteine* and to those who are in a position to provide that information. There are also already concrete ideas on how the application could be further improved. The next major step would be to conduct a larger-scale evaluation that would also involve the people who are contributing the information.

In the larger context, the results of our work make us optimistic about the potential of combining Augmented Reality and Object Detection. Furthermore, we think that the concept of our work can be applied to other memorial sites where there is little space for the placement of additional information (such as street signs or tombstones). In this way, it can be ensured that not only the name of the person, but also their story is not forgotten.

List of Figures

Acronyms

AR Augmented Reality. iv, 2–6, 11–18, 21, 22, 31, 34, 35, 38, 42, 45–47, 49, 51, 52

AV Augmented Virtuality. 3

COCO Common Objects in Context. 7, 24, 26, 28, 29, 32

CV computer vision. 23, 30

DNN Deep Neural Network. 7–10, 15, 23, 26, 32, 33

Fast R-CNN Fast Region based Convolutional Neural Network. 25

Faster R-CNN Faster Region based Convolutional Neural Network. 25, 26

HAR Handheld Augmented Reality. 6, 15, 42

HARUS Handheld Augmented Reality Usability Scale. 42

HMD Head Mounted Display. 6

IMU Inertial Measurement Unit. 49

IoU Intersection over Union. 30

mAP mean Average Precision. 32

MR Mixed Reality. 3

NASA-TLX NASA Task Load Index. 42

ONNX Open Neural Network Exchange. 33

R-CNN Region based Convolutional Neural Network. 25, 52

RV Reality-Virtuality. 3, 52

SAR Spatial Augmented Reality. 6, 15

SSD Single Shot Detector. 15, 26

SUS System Usability Scale. 42–44

UI user interface. 45, 46, 50

YOLO You Only Look Once. 26–29, 32, 33

Bibliography

[21] *COCO test-dev Benchmark (Object Detection) | Papers With Code*. 2021. URL: `https://paperswithcode.com/sota/object-detection-on-coco` (visited on 07/11/2021).

[Aga21] V. Agafonkin. *Fast geodesic approximations with Cheap Ruler | by Mapbox*. 2021. URL: `https://blog.mapbox.com/fast-geodesic-approximations-with-cheap-ruler-106f229ad016` (visited on 07/08/2021).

[Azu97] R. T. Azuma. "A Survey of Augmented Reality." In: *Presence: Teleoperators & Virtual Environments* 6.4 (1997), pp. 355–385.

[Bir+02] W. Birkfellner, M. Figl, K. Huber, F. Watzinger, F. Wanschitz, J. Hummel, R. Hanel, W. Greimel, P. Homolka, R. Ewers, et al. "A head-mounted operating binocular for augmented reality visualization in medicine-design and initial evaluation." In: *IEEE Transactions on Medical Imaging* 21.8 (2002), pp. 991–997.

[Blu+12] T. Blum, V. Kleeberger, C. Bichlmeier, and N. Navab. "mirracle: An augmented reality magic mirror system for anatomy education." In: *2012 IEEE Virtual Reality Workshops (VRW)*. IEEE. 2012, pp. 115–116.

[Boc20] A. Bochkovskiy. *YOLOv4-tiny released: 40.2% AP50, 371 FPS (GTX 1080 Ti), 1770 FPS tkDNN/TensorRT*. 2020. URL: `https://github.com/AlexeyAB/darknet/issues/6067` (visited on 07/03/2021).

[Boc21] A. Bochkovskiy. *AlexeyAB/darknet: YOLOv4 / scaled-YOLOv4 / YOLO - neural networks for object detection (Windows and Linux version of Darknet)*. 2021. URL: `https://github.com/AlexeyAB/darknet#how-to-train-to-detect-your-custom-objects` (visited on 06/02/2021).

[Bom] Bombsight. *Announcing the release of the android app, Bombsight*. URL: `https://blitzbombcensusmaps.wordpress.com/2012/12/20/announcing-the-release-of-the-android-app/` (visited on 04/08/2021).

[Bro96] J. Brooke. "Sus: a "quick and dirty'usability." In: *Usability evaluation in industry* 189 (1996).

[BWL20] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao. "Yolov4: Optimal speed and accuracy of object detection." In: *arXiv preprint arXiv:2004.10934* (2020).

[Car+11] J. Carmigniani, B. Furht, M. Anisetti, P. Ceravolo, E. Damiani, and M. Ivkovic. "Augmented reality technologies, systems and applications." In: *Multimedia tools and applications* 51.1 (2011), pp. 341–377.

[CV20] D. Cucinotta and M. Vanelli. "WHO declares COVID-19 a pandemic." In: *Acta Bio Medica: Atenei Parmensis* 91.1 (2020), p. 157.

[Dam+08] A. Damala, P. Cubaud, A. Bationo, P. Houlier, and I. Marchal. "Bridging the gap between the digital and the physical: design and evaluation of a mobile augmented reality guide for the museum visit." In: *Proceedings of the 3rd international conference on Digital Interactive Media in Entertainment and Arts*. 2008, pp. 120–127.

[DB11] A. Dünser and M. Billinghurst. "Evaluating augmented reality systems." In: *Handbook of augmented reality*. Springer, 2011, pp. 289–307.

[Dem21] G. Demnig. *Project Stolpersteine FAQ*. 2021. URL: `http://www.stolpersteine.eu/en/faq/` (visited on 07/22/2021).

[DIK13] Á. Di Serio, M. B. Ibáñez, and C. D. Kloos. "Impact of an augmented reality system on students' motivation for a visual art course." In: *Computers & Education* 68 (2013), pp. 586–596.

[Eve+10] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. "The pascal visual object classes (voc) challenge." In: *International journal of computer vision* 88.2 (2010), pp. 303–338.

[EVF19] M. Eckert, J. S. Volmerg, and C. M. Friedrich. "Augmented reality in medicine: systematic and bibliographic review." In: *JMIR mHealth and uHealth* 7.4 (2019), e10967.

[FA15] J. C. Freeman and J. Auchter. "Border Memorial: Frontera de los Muertos." In: *Hyperrhiz: New Media Cultures* 12 (June 2015), pp. 1–1. DOI: `10.20415/hyp/012.am01`.

[FHL20] J. Fan, T. Huo, and X. Li. "A review of one-stage detection algorithms in autonomous driving." In: *2020 4th CAA International Conference on Vehicular Control and Intelligence (CVCI)*. 2020, pp. 210–214. DOI: `10.1109/CVCI51460.2020.9338663`.

[Fin06] K. Finstad. "The system usability scale and non-native English speakers." In: *System* 1.4 (2006), pp. 185–188.

[Gir+14] R. Girshick, J. Donahue, T. Darrell, and J. Malik. "Rich feature hierarchies for accurate object detection and semantic segmentation." In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014, pp. 580–587.

[Gir15] R. Girshick. “Fast R-CNN.” In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 1440–1448.

[Gue21] A. Guerrero. *Descargar Ikea Place, la app para amueblar tu casa desde el móvil*. 2021. URL: `https://www.actualapp.com/descargar-ikea-place-app-realidad-aumentada-43404` (visited on 07/09/2021).

[How+17] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. “Mobilenets: Efficient convolutional neural networks for mobile vision applications.” In: *arXiv preprint arXiv:1704.04861* (2017).

[HS88] S. G. Hart and L. E. Staveland. “Development of NASA-TLX (Task Load Index): Results of empirical and theoretical research.” In: *Advances in psychology*. Vol. 52. Elsevier, 1988, pp. 139–183.

[HSW89] K. Hornik, M. Stinchcombe, and H. White. “Multilayer feedforward networks are universal approximators.” In: *Neural networks* 2.5 (1989), pp. 359–366.

[Ian+16] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer. “SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and < 0.5 MB model size.” In: *arXiv preprint arXiv:1602.07360* (2016).

[Ibá+14] M. B. Ibáñez, Á. Di Serio, D. Villarán, and C. D. Kloos. “Experimenting with electromagnetism using augmented reality: Impact on flow student experience and educational effectiveness.” In: *Computers & Education* 71 (2014), pp. 1–13.

[JAS13] K. Jones, P. Aucott, and H. Southall. “Bomb Sight–mapping WW2 bomb census.” In: (2013).

[JEW19] J.E.W.S.org. *Holocaust- Stolpersteine is the world’s largest project of commemoration*. 2019. URL: `https://j-e-w-s.org/holocaust-stolpersteine/` (visited on 07/22/2021).

[KJO19] T. Khan, K. Johnston, and J. Ophoff. “The impact of an augmented reality application on learning motivation of students.” In: *Advances in Human-Computer Interaction* 2019 (2019).

[KSH12] A. Krizhevsky, I. Sutskever, and G. E. Hinton. “Imagenet classification with deep convolutional neural networks.” In: *Advances in neural information processing systems* 25 (2012), pp. 1097–1105.

[KSP20] J.-a. Kim, J.-Y. Sung, and S.-h. Park. "Comparison of Faster-RCNN, YOLO, and SSD for real-time vehicle type recognition." In: *2020 IEEE International Conference on Consumer Electronics - Asia (ICCE-Asia)*. 2020, pp. 1–4. DOI: `10.1109/ICCE-Asia49877.2020.9277040`.

[Kur20] W. Kurdthongmee. "A comparative study of the effectiveness of using popular DNN object detection algorithms for pith detection in cross-sectional images of parawood." In: *Heliyon* 6.2 (2020), e03480.

[Lin+15] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. Dollár. *Microsoft COCO: common objects in context*. 2015. arXiv: `1405.0312`.

[Liu+16] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. "SSD: Single shot multibox detector." In: *European conference on computer vision*. Springer. 2016, pp. 21–37.

[LLG19] L. Liu, H. Li, and M. Gruteser. "Edge assisted real-time object detection for mobile augmented reality." In: *The 25th Annual International Conference on Mobile Computing and Networking*. 2019, pp. 1–16.

[Mar21] L. M. Martins. *Hololens 2 – What's been done? – Geospatial World*. 2021. URL: `https://www.geospatialworld.net/blogs/hololens-2-whats-been-done/` (visited on 07/09/2021).

[MB19] K. Merry and P. Bettinger. "Smartphone GPS accuracy study in an urban environment." In: *PloS one* 14.7 (2019), e0219890.

[Mil+95] P. Milgram, H. Takemura, A. Utsumi, and F. Kishino. "Augmented reality: A class of displays on the reality-virtuality continuum." In: *Telemanipulator and telepresence technologies*. Vol. 2351. International Society for Optics and Photonics. 1995, pp. 282–292.

[Miy+08] T. Miyashita, P. Meier, T. Tachikawa, S. Orlic, T. Eble, V. Scholz, A. Gapel, O. Gerl, S. Arnaudov, and S. Lieberknecht. "An augmented reality museum guide." In: *2008 7th IEEE/ACM International Symposium on Mixed and Augmented Reality*. IEEE. 2008, pp. 103–106.

[Pac+14] D. Pacheco, S. Wierenga, P. Omedas, S. Wilbricht, H. Knoch, and P. Verschure. "Spatializing experience: a framework for the geolocalization, visualization and exploration of historical data using VR/AR technologies." In: Apr. 2014. DOI: `10.13140/RG.2.1.2384.5843`.

[Pac+15] D. Pacheco, S. Wierenga, P. Omedas, L. S. Oliva, S. Wilbricht, S. Billib, H. Knoch, and P. F. Verschure. "A location-based augmented reality system for the spatial interaction with historical datasets." In: *2015 Digital Heritage*. Vol. 1. IEEE. 2015, pp. 393–396.

[Pan+18] C. Panou, L. Ragia, D. Dimelli, and K. Mania. "An architecture for mobile outdoors augmented reality for cultural heritage." In: *ISPRS International Journal of Geo-Information* 7.12 (2018), p. 463.

[PNS20] R. Padilla, S. L. Netto, and E. A. da Silva. "A survey on performance metrics for object-detection algorithms." In: *2020 International Conference on Systems, Signals and Image Processing (IWSSIP)*. IEEE. 2020, pp. 237–242.

[Pos19] E. Posner. *Detection free human instance segmentation using Pose2Seg and PyTorch*. 2019. URL: `https://towardsdatascience.com/detection-free-human-instance-segmentation-using-pose2seg-and-pytorch-72f48dc4d23e` (visited on 06/15/2021).

[Rao+17] J. Rao, Y. Qiao, F. Ren, J. Wang, and Q. Du. "A mobile outdoor augmented reality method combining deep learning object detection and spatial relationships for geovisualization." In: *Sensors* 17.9 (2017), p. 1951.

[Red+16] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. "You only look once: Unified, real-time object detection." In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 779–788.

[Red16] J. Redmon. *Darknet: Open source neural networks in C*. 2013–2016. URL: `http://pjreddie.com/darknet/` (visited on 06/02/2021).

[Red18] J. Redmon. *YOLO: Real-Time Object Detection*. 2018. URL: `https://pjreddie.com/darknet/yolo/` (visited on 08/08/2021).

[Ren+15] S. Ren, K. He, R. Girshick, and J. Sun. "Faster R-CNN: Towards real-time object detection with region proposal networks." In: *arXiv preprint arXiv:1506.01497* (2015).

[RF17] J. Redmon and A. Farhadi. "YOLO9000: better, faster, stronger." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. July 2017.

[RF18] J. Redmon and A. Farhadi. *YOLOv3: An incremental improvement*. 2018. arXiv: `1804.02767`.

[Sad21] R. Sadli. *The beginner's guide to implementing Yolov3 in TensorFlow 2.0*. 2021. URL: `https://machinelearningspace.com/yolov3-tensorflow-2-part-1/#nms-unique` (visited on 06/15/2021).

[San+14] M. E. C. Santos, T. Taketomi, C. Sandor, J. Polvi, G. Yamamoto, and H. Kato. "A usability scale for handheld augmented reality." In: *Proceedings of the 20th ACM Symposium on Virtual Reality Software and Technology*. 2014, pp. 167–176.

[San+18] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen. "Mobilenetv2: Inverted residuals and linear bottlenecks." In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 4510–4520.

[Sin+21] S. Singh, U. Ahuja, M. Kumar, K. Kumar, and M. Sachdeva. "Face mask detection using YOLOv3 and faster R-CNN models: COVID-19 environment." In: *Multimedia Tools and Applications* 80.13 (2021), pp. 19753–19768.

[SW05] D. Schmalstieg and D. Wagner. "A handheld augmented reality museum guide." In: *In Proc. IADIS International Conference on Mobile Learning*. Citeseer. 2005.

[Tec21] Techzizou. *Train a custom Yolov4-tiny object detector using google colab*. 2021. URL: `https://medium.com/analytics-vidhya/train-a-custom-yolov4-tiny-object-detector-using-google-colab-b58be08c9593` (visited on 06/02/2021).

[Upa+20] G. K. Upadhyay, D. Aggarwal, A. Bansal, and G. Bhola. "Augmented reality and machine learning based product identification in retail using vuforia and mobilenets." In: *2020 International Conference on Inventive Computation Technologies (ICICT)*. IEEE. 2020, pp. 479–485.

[Van13] G. Van Brummelen. *Heavenly mathematics: The forgotten art of spherical trigonometry*. Princeton University Press, 2013. ISBN: 9780691148922.

[Vin75] T. Vincenty. "Direct and inverse solutions of geodesics on the ellipsoid with application of nested equations." In: *Survey Review* 23.176 (1975), pp. 88–93. DOI: `10.1179/sre.1975.23.176.88`.

[Vla+02] V. Vlahakis, M. Ioannidis, J. Karigiannis, M. Tsotros, M. Gounaris, D. Stricker, T. Gleue, P. Daehne, and L. Almeida. "Archeoguide: an augmented reality guide for archaeological sites." In: *IEEE Computer Graphics and Applications* 22.5 (2002), pp. 52–60.

[Wet20] J. Wetzel. "Ägyptisches Museum München:"Stolpersteine" zum Gedenken." In: *Süddeutsche Zeitung* (Oct. 23, 2020).

[WSG14] J. White, D. C. Schmidt, and M. Golparvar-Fard. "Applications of augmented reality." In: *Proceedings of the IEEE* 102.2 (2014), pp. 120–123.

[Www18] D. W. (Www.Dw.Com. *NS-Gedenkprojekt zählt jetzt 70.000 Stolpersteine*. 2018. URL: `https://www.dw.com/de/ns-gedenkprojekt-z%5C%C3%5C%A4hlt-jetzt-70000-stolpersteine/a-45997346` (visited on 07/22/2021).

[Z+98] D. Ziou, S. Tabbone, et al. "Edge detection techniques-an overview." In: *Pattern Recognition and Image Analysis C/C of Raspoznavaniye Obrazov I Analiz Izobrazhenii* 8 (1998), pp. 537–559.

[Zam19] S. Zambare. *Object detection: using non-max supression over YOLOv2*. 2019. URL: `https://medium.com/@sarangzambare/object-detection-using-non-max-supression-over-yolov2-382a90212b51` (visited on 06/15/2021).

[Zou+19] Z. Zou, Z. Shi, Y. Guo, and J. Ye. "Object detection in 20 years: A survey." In: *arXiv preprint arXiv:1905.05055* (2019).