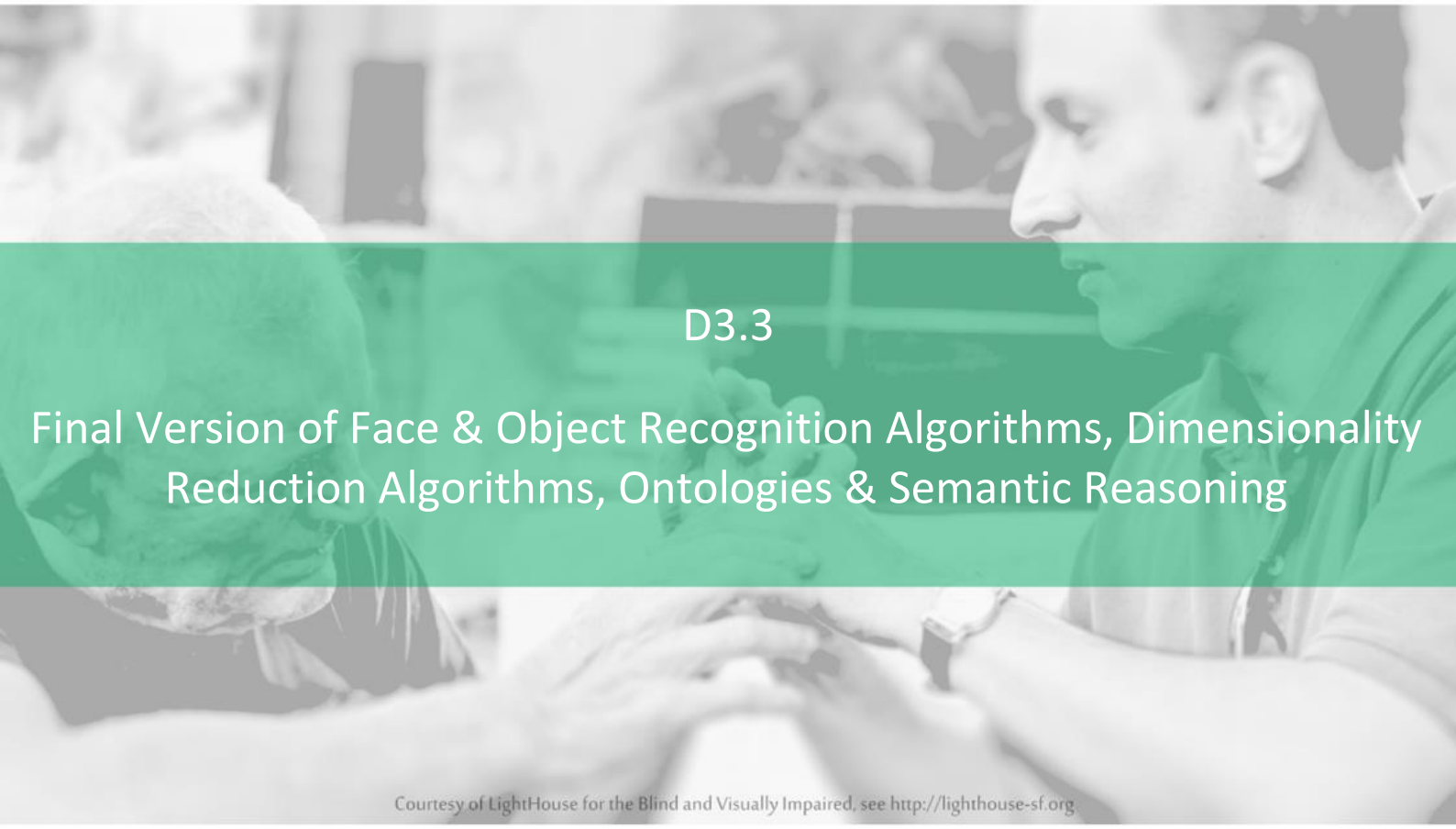
SUITCEYES

1 Jan 2018 - 31 Dec 2020

Smart, User-friendly, Interactive, Tactual, Cognition-Enhancer, that Yields Extended Sensosphere
Appropriating sensor technologies, machine learning, gamification and smart haptic interfaces



D3.3

Final Version of Face & Object Recognition Algorithms, Dimensionality Reduction Algorithms, Ontologies & Semantic Reasoning

Dissemination level		
PU	PUBLIC, fully open, e.g., web	X
CO	CONFIDENTIAL, restricted under conditions set out in Model Grant Agreement	
CI	CLASSIFIED, information as referred to in Commission Decision 2001/844/EC.	

Deliverable Type		
R	Document, report (excluding the periodic and final reports)	X
DEM	Demonstrator, pilot, prototype, plan designs	
DEC	Websites, patents filing, press & media actions, videos, etc.	
OTHER	Software, technical diagram, etc.	

Deliverable Details	
Deliverable number	D3.3
Part of WP	WP3
Lead organisation	CERTH
Lead member	P. Petrantonakis

Revision History			
V#	**Date**	**Description / Reason of change**	**Author / Org.**
v0.1	31-Mar-2021	Structure proposal	P. Petrantonakis / CERTH
v0.2	15-Apr-2021	1st draft for internal review	P. Petrantonakis / CERTH E. Kouslis / CERTH S. Darányi / HB N. Olson / HB
v0.3	20-Apr-2021	2nd draft addressing review comments	P. Petrantonakis / CERTH E. Kouslis / CERTH S. Darányi / HB N. Olson / HB
v0.4	23-Apr-2020	Pre-final draft after PC's comments	P. Petrantonakis / CERTH
v1.0	27-Apr-2020	Final draft submitted to the EU	T. Bebis / HB

Authors	
Partner	**Name(s)**
CERTH	P. Petrantonakis
CERTH	E. Kouslis
HB	S. Darányi
HB	N. Olson

Contributors		
Partner	**Contribution type**	**Name**
UNIVLEEDS	Internal review	R. Holt, M. Shaqura

Executive Summary

This document constitutes SUITCEYES deliverable D3.3 presenting the work conducted during the period M27-42 within WP3 and reports on the following:

(a) The final version of visual analysis algorithms evaluated in benchmark datasets and real – conditions testing for performing object detection and tracking, face detection and tracking, scene recognition and face-mask detection and recognition;

(b) Description of haptogram design for perceptible haptic information in low dimensional haptic grids;

(c) The final version of the semantic knowledge graphs for semantically integrating the multimodal outputs from the various heterogeneous SUITCEYES sensors and components.

Table of Contents

1 Introduction 3

2 Visual Analysis 4

2.1 Algorithms and Results 4
2.2 Server-based Visual Analysis 5
2.2.1 Object Detection 5
2.2.2 Scene Recognition for Situational Awareness 8
2.2.3 Communication Protocol 11
2.3 Facemask detection 12
2.4 Standalone version 15
2.4.1 Object and scene detection and recognition 16
2.5 Chapter Summary and Future Work 17

3 Haptic Communication Design 18

3.1 Progress roadmap 18
3.2 Constraints on haptogram design 19
3.3 New results 19
3.4 Participatory co-design efforts 23
3.5 Haptogram design toolkit 23
3.6 Summary of haptogram design results and future work 26

4 Semantic Knowledge Representation and Reasoning 27

4.1 Additions to the SUITCEYES Ontology 27
4.2 Knowledge Base Service (KBS) 28
4.2.1 KBS Communication Process with the HIPI system 28
4.2.2 Natural Language Output/Multiple Entity Reports 31
4.2.3 Efficient storing of incoming data 33
4.2.4 Active Object Search module 35
4.2.5 Knowledge Base Dashboard for HIPI system visualization 35
4.3 Chapter Summary and Future Work 38

5 Conclusions 39

References 40

Appendix A: The list of numbers between 0-19 and the Latin alphabet mapped to a 5x5 matrix 41

Appendix B: Ideation workshops and technological process of SHC adoption 42

1 Introduction

In this deliverable we present the final versions of the algorithms and methods that were integrated in the HIPI system.

In the second chapter, we present additions and modifications that have been made to the Visual Analysis (VA) tools with respect to: a) new standalone algorithms that run independently of a remote server (on-board run) b) new facemask detection algorithms, c) advanced implementations for scene recognition, d) communication protocol, and e) adjustments and optimization steps for object/face detection and recognition approaches.

In the third chapter, we describe the co-design of haptograms with the inventors of Social Haptic Communication (SHC). Moreover, the idea of haptogram design for multiple displays over several body parts with different grid sizes via a multipanel suit is presented, and a method of encoding distributed haptic messages is introduced. Finally, an open source tool for the community-based co-design of haptic communication elements is described.

The fourth chapter presents the third generation of the implemented SUITCEYES ontology and the semantic reasoning framework. The Knowledge Base Service (KBS) was further improved and extended to enable the communication between the components of the HIPI system and the ontology and to store the incoming data in a more efficient way. In addition, the Semantic Reasoning Mechanism (SRM) was further enriched to cover a wide range of queries to semantically improve the awareness of the user's surrounding environment, either in the form of structured content (JSON format) or as natural language phrases. The SRM was also extended to handle multiple entity output sentences. Finally, we present a web interface called Knowledge Base Dashboard (KBD) to visualize the processes executed within the HIPI system and more specifically the communication between the VA tool, the iBeacon sensors, the KBS and the user.

Finally, section 5 concludes the deliverable.

2 Visual Analysis

In this section, the algorithms and methodologies related to Visual Analysis, that were implemented, evaluated, and integrated to the platform are showcased and discussed. The code along with the instructions on how to set up the Visual Analysis platform is publicly available on Github[1].

2.1 Algorithms and Results

At this stage of development, in Visual Analysis (VA), our first priority was ensuring that the algorithms and modules that we have developed and utilized are the ones that fit our needs the most, since the list of available tools, methodologies and algorithms in the literature is ever growing. Towards that end, we compared the algorithms that we have already developed, with newly developed ones, in the two major subtasks of VA, which are object and scene detection and recognition. It should be stressed that due to the COVID-19 pandemic and the respective consequences of limited access to lab equipment and infrastructures, the testing and validation of the aforementioned algorithms in realistic scenarios was difficult and confined in indoor spaces. Nevertheless, from relatively extensive testing in realistic scenarios, we were able to identify their limitations and, in many cases, surpass them with the utilization of computer vision techniques and heuristic algorithms.

Another limitation or better yet dependency of the SUITCEYES VA module, that we realized throughout the testing process, is the capabilities of the network which is utilized in order to connect and transmit data to the remote VA Server on which the analysis part takes place. On average the upload time takes from 1 to 2 seconds for each image, on a network with a stable internet connection. As it has been noted, the capabilities of the network, in terms of upload speed, impact heavily the VA process introducing a lag in the real-time analysis, much greater than the time (1.3 sec on average for all functions, e.g. object detection, scene detection etc.) that the VA requires for processing of all the modules. In those cases, the delay can hamper the communication and the real-time aspect which is required. For that reason, a standalone version of the VA, which does not need the server is developed. This version aims to be an alternative to the standard one in cases with a poor network connection and not replace it, since the processing capabilities of the device, which is on the vest (prototype HIPI, Del. 4.3) are limited.

Lastly, the platform capabilities of the VA, enable the integration of new modules, that utilize the current ones on top of them. The new module that has been developed in this manner is the facemask detection module. With the current pandemic crisis of Covid-19 in mind, we developed a module that detects if a person wears a mask, with the utilization of the existing face detection module, a sub-part of the object recognition module.

[1] *https://github.com/Suitceyes-Project-Code/Suitceyes-Visual-Analysis*

The rest of the section is structured as follows: First, the server-based VA module is presented, showing the developments in object and scene detection and recognition modules. Next, the new facemask detection module is showcased. Finally, the standalone versions are presented.

2.2 Server-based Visual Analysis

In this section, the latest developments in the standard server-based version are showcased, which include the development of alternative modules that are aimed to be compared with the current ones and techniques and methods that improve their overall accuracy. The newest communication protocol is also presented.

2.2.1 Object Detection

In Object detection, the objective is to detect and recognize various everyday objects such as a Laptop or a Mug. This is achieved, when an object becomes visible to the camera. This is possible due to the RGB-D sensor, the RealSense R435i, that is attached to the HIPI and the Raspberry Pi 4 Model B, which read RGB-D images with depth information, encode the data, and upload it to the server for analysis. The depth information allows for distance estimation from objects, providing the user the ability to know the proximity of an object. In Table 2.1, a subset of everyday objects from the 600 available, that are tested with the VA, are showcased.

Table 2.1. Extended Object categories

table	chair	couch	mug
cell phone	book	laptop	monitor
bowl	microware	toothbrush	

The selected architecture for object detection, is the Faster R-CNN (FRCNN) architecture, pre-trained using the Open-Images V4 (Kuznetsova, Alina, et al., 2020) dataset. Other architectures were investigated to make sure the chosen architectures provides the best possible performance.

YOLO (Redmon, et al.,2016) architecture is popular for object detection. The way it works is by the application of a single neural network to the whole image, which in turn divides the images and then predicts potential objects with their corresponding probabilities. In contrast with FRCNN, it does not use regions to localize the object within the image. Instead, it focuses on areas with high probability of containing the object. The image is divided to an SxS grid, and for each cell in the grid, the potential bounding boxes are computed. The bounding boxes with a high probability contain the detected objects. This one step methodology is much faster than the standard two step methodology that FRCNN employs. Thus, potentially reducing the computational time required for each image. This reduction of time could be noticeable, however, only on a very fast internet connection, which requires less than 1.3 seconds for each image to be uploaded. Since, 1.3 seconds is the time needed to do the computations on the

server by the visual analysis tool, with the bigger portion of this time, 620 ms as shown in Table 2.2, being used for object detection. The particular version that we examined, for object detection, is YOLO version 3.

Another object detection architecture that we investigated is one that relies on the Single Shot Multibox Detector (Liu, Wei, et al.,2016) or SSD detector. In contrast with other object detection architectures, it does not have a region proposal step in the detection process but predicts the potential objects with classes in one single pass. Similar to the YOLO methodology and contrary to FRCNN, the Single Shot Multibox Detector is a one-step methodology. The multibox in the name means that the bounding boxes are predicted by taking anchors, which are predefined boxes, as a starting point and perform regression until object is located. The Multibox loss is calculated as:

$$L_{multibox} = L_{conf} + a * L_{loc}$$

Where L_{conf} is a cross entropy in the classification of the detected objects and L_{loc} is a regression loss on the coordinates of the detected bounding box. α is used to scale L_{loc}. This methodology is again faster than the FRCNN. The particular version, which we utilized, relies on the Resnet-v2 (Szegedy, et al., 2017) convolution network backbone.

We investigated the three architectures above to determine which one offers the best combination of speed and accuracy for our application. The tests were carried out with similar settings, which means aiming to detect the same objects in similar distance and lightning conditions. Additionally, for the same reason we applied these architectures on the same images in an offline manner and not on real-time, so we can understand their potential differences. Even though both the YOLO and the SSD architectures were slightly faster than the existing FRCNN architecture, there was a noticeable drop-off in object detection performance, when we tested them in realistic scenarios, so we opted to keep the current FRCNN architecture.

Table 2.2. Object detection comparison

Server based VA	***Object detection on Bottle***	***Object detection on face***	***Inference time (ms)***
Yolo	90%	100%	230
FRCNN	96%	100%	620
SSD	64%	100%	50

We tested them on images taken from footage while wearing the prototype HIPI (see Deliverable 4.3), where we measured how many times each methodology detected a specific object. With the indicative results presented in Table 2.2, it becomes clear that the FRCNN algorithm performs better than the other two architectures for detecting objects. On the other hand, utilizing the other versions is faster. In our case, however the inability to detect key objects efficiently deterred us from selecting them. In Figures 2.1 and 2.2 below, the FRCNN based object detection and depth estimation are presented.

Figure 2.1. Bottle detection and depth estimation

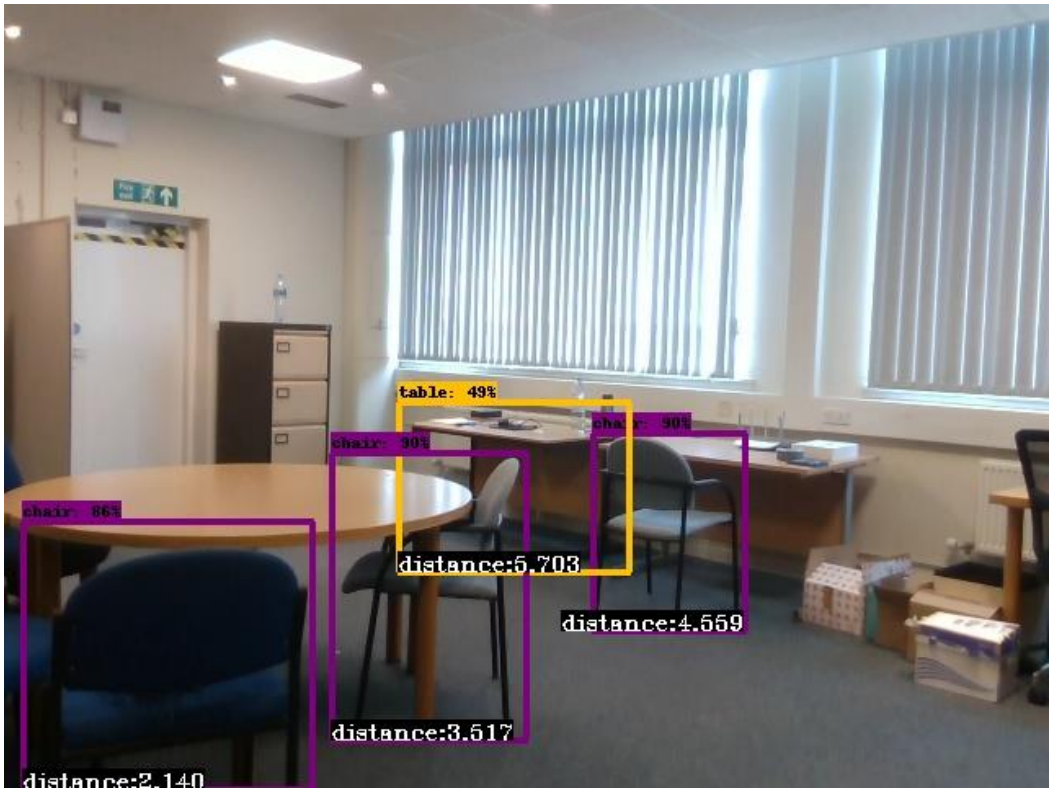


Figure 2.2. Chair and table detection and depth estimation

2.2.2 Scene Recognition for Situational Awareness

Apart from the detection and recognition of objects that are visible to the camera, another goal of the server-based Visual Analysis module is to recognize the environment that the user finds herself in. This means that the incoming images from the RGB camera are going to be categorized, in one of the 15 scene classes. This is achieved, with the utilization of a VGG16 network, trained on the Places[2] dataset. More information is available in deliverable 3.1, chapter 2.2.3.

The aforementioned network is a powerful network, which enables the VA module to classify the scene that the camera points at in most cases. However, in some fringe cases, where the camera points on a featureless environment such as a blank wall or extremely close to surfaces, it has been observed that the network struggles, which yields some stray misclassifications. Also, due to movement, of the user that wears the prototype HIPI, on which the camera is attached, some images may appear blurry, which also hinders the scene classification. To address these issues, various solutions based on computer vision techniques have been employed.

One integrated solution is the Laplace operator (Wang, 2007). With this operator, the target is to measure the amount of edges present on each image, through the second derivative of Laplacian. It highlights areas of the image which contain rapid intensity changes. This variation of intensity, in high levels indicates the existence of abundant edges and non-edges. In blurred images the Laplacian operator yields lower values, which consequently enables us to identify them. In combination with the performance of the scene recognition module when in synergy with the Laplacian operator, we have set up a threshold of the lowest possible variance that an image is

[2] http://places2.csail.mit.edu/

characterized as non-blurry, and the module will attempt to recognize the scene. The Laplacian operator L(f) is defined by:

$$L(f) = \frac{d^2 f}{dx^2} + \frac{d^{2f}}{dy^2}$$

which shows how the derivative is computed in both dimensions x and y. This method allows the VA module to identify the incoming images where the overall quality is lower due to blurriness. Additionally, images without meaningful content, such as extremely close images to a surface, will also be identified. Hence, these images are excluded from analysis as it is impossible to extract any information from them, which in turn improves the overall performance of the scene recognition sub-module.

Another solution that has been employed is the utilization of the depth information on each image to improve the confidence of the sub-module of a correct classification. It has been observed that the task of scene recognition becomes more accurate when the camera captures a whole area, whether it is a living room or a kitchen. When only a part of the overall scene is visible to the camera, it may lead to more ambiguous and confusing results. For instance, if the wearer of the vest finds herself present in a living room but the camera points on a table with a Laptop it becomes harder for the scene recognition to understand, whether it is an office or a living room scene. Using the depth information in such cases allows the scene recognition sub-module to identify if the camera points at a small part of an indoor location or a greater area with a bigger depth. This is achieved by considering all areas of the image and extracting the average depth, which in turn acts as an indicator for the confidence of the sub-module for a correct classification. This way when there is a large area in front of the camera, this average depth estimator value increases which in turn translates into a more descriptive image of the scene, suitable for the scene module to attempt to recognize. Smaller average depth estimator value means that the camera is near a large surface, such as a wall or most of its field of view is covered by objects which are extremely near. Consequently, those images do not offer clear view of the area and are not suitable for the scene recognition.

The indoor scenes that the scene recognition sub-module is able to classify are shown in the Table 2.3 below.

Table 2.3. Scene categories

basement	bathroom	bedroom	child's room
closet	corridor	dining room	elevator
kitchen	laundromat	living room	office

pantry	staircase		

In the Figures 2.3 and 2.4 below, visualized examples of the unified object and scene recognition and classification are shown, where in Figure 2.3 an office scene and in Figure 2.4 a living room scene are detected, respectively.

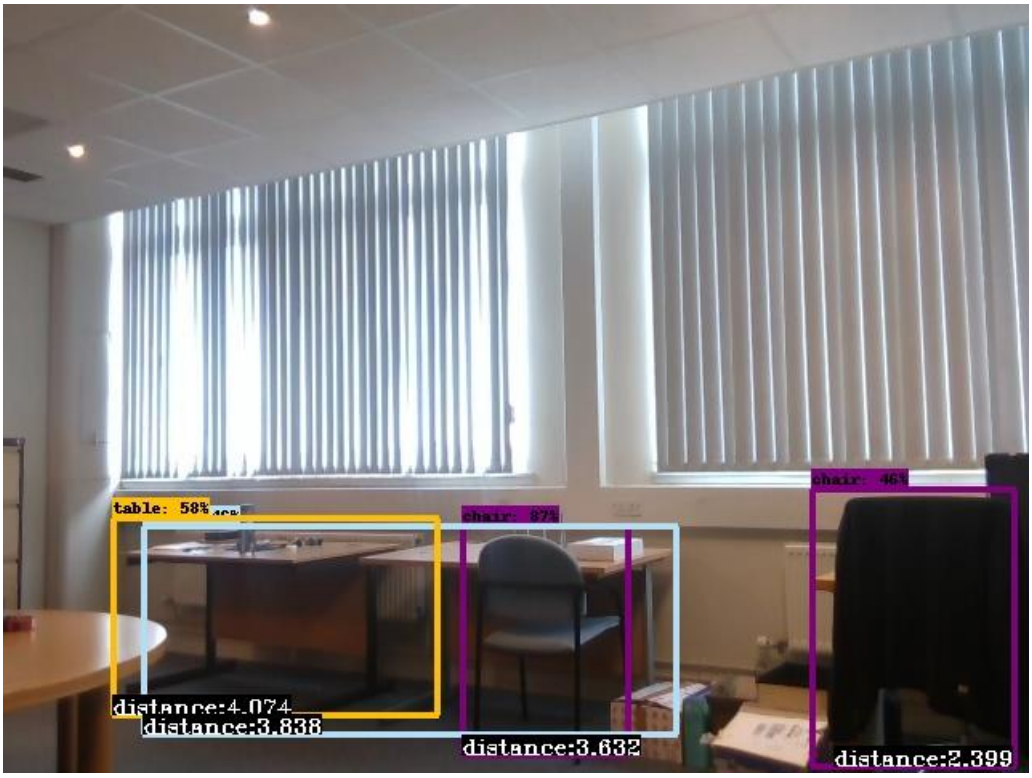


Figure 2.3. Office area scene recognition

Additionally, there are many instances where the user does not move while wearing the vest or her field of view does not change, while making minimal movements. In those cases, the incoming images will be very similar. In order to recognize the level of similarity between two consecutive incoming images, a measurement value utilizing image hashing (Gayozo, et al.,2014) is calculated on each image. The calculation is based on the luminance structure of images. Luminance (Lennie et al., 1993) is a measurement that shows the luminous intensity per unit area of light traveling in a given direction. The difference of the averaged calculated image hashing values between the two images is measured and represents the level of similarity between the images. More information on the usage of the aforementioned measure is available in chapter

4.2.3. For the measurement of the blurriness and similarity values the OpenCV[3] and imagehash[4] libraries were utilized, respectively.

Figure 2.4. Living room area scene recognition

2.2.3 Communication Protocol

As in the Figure 2.5 below, the overall process begins with the RGB-D sensor and the image acquisition and ends with the VA and the forwarding of information.

The RGB image and the corresponding depth image are directly uploaded to the server for visual analysis and simultaneously a json message is published. It is published on a shared channel, in order to notify the VA module that an image has arrived and is ready for analysis. In turn the VA module conducts the analysis, publishes its results and stores a visualized image with the detected objects and their corresponding bounding boxes on the server, as shown in Figures 2.1 and 2.2. This is a continuous process, which aims to provide information about the surrounding world in real time. More information is available in Deliverable 3.2, chapter 2.1.1.

With the RGB image, the depth information and the respective notification that is received through the MQTT protocol, already in server, all that is left is to conduct the Visual analysis. A

[3] *https://opencv.org/*
[4] *https://pypi.org/project/ImageHash/*

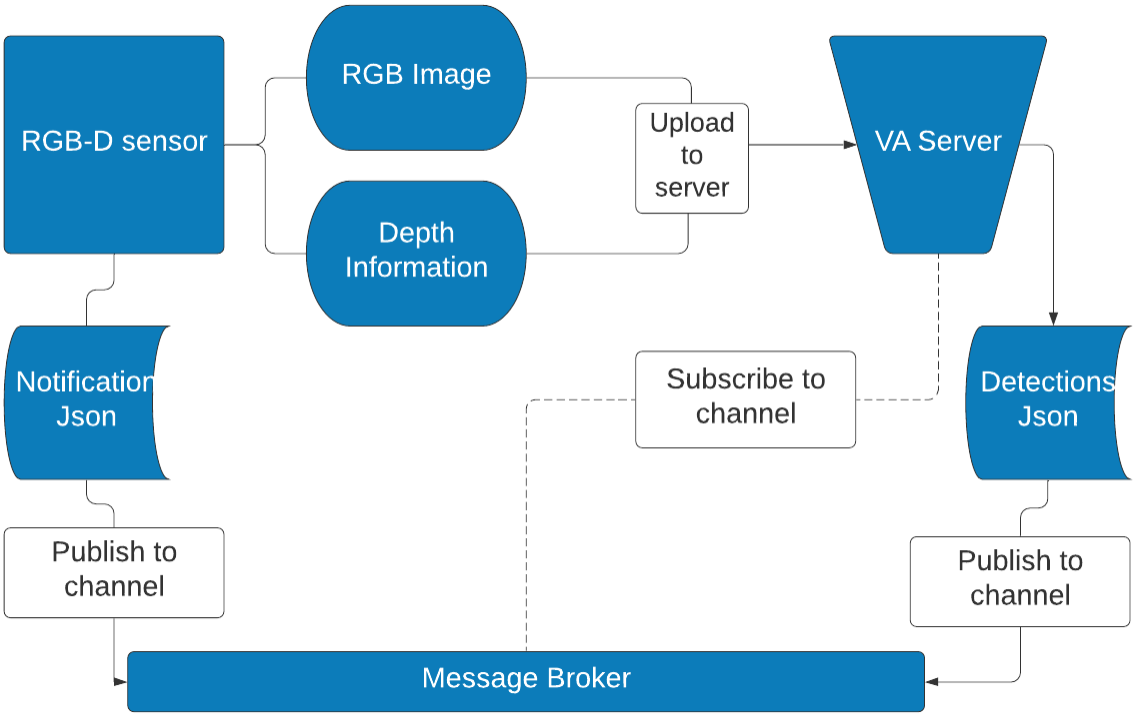


Figure 2.5. Communication Diagram

key aspect of the communication, between the RGB-D sensor and the VA tool, is the ability to notify in real-time the successful upload of the images to the server. The communication service that we utilized, the Realtime.co, discontinued their service, so we opted for a new communication protocol, which does not rely on a specific broker or service format to work. Instead, our goal was that once the new communication protocol is set up, the brokers and their providers could be easily interchangeable according to our needs. This is done with the utilization of the MQTT protocol. The MQTT or MQ Telemetry Transport, is a lightweight messaging protocol, which allows clients to send or listen for messages. This is done by one or more clients publishing the message to a channel, through a broker, which acts as a medium between clients, and one or more clients subscribing to the same channel to listen for incoming messages. In our case, we used the Ably[5] broker to act as a medium between clients. Furthermore, since MQTT is a well-established protocol, it is entirely straightforward to use our platform in the future with a different broker shall the need arise.

2.3 Facemask detection

A need that has become paramount in recent times, is the protection from viral diseases. Covid-19 safety measures require among other things a Facemask when close to another person. This led us to the development of a module that aims to assist people with deafblindness identify whether people in close proximity, wear a facemask. This is a module which utilizes the Visual Analysis platform, as it uses the faces of the people detected by the object detection module. In

[5] *ably.com*

the Figure 2.6 below, it is shown how the Facemask module is utilized based on the Visual Analysis Platform.

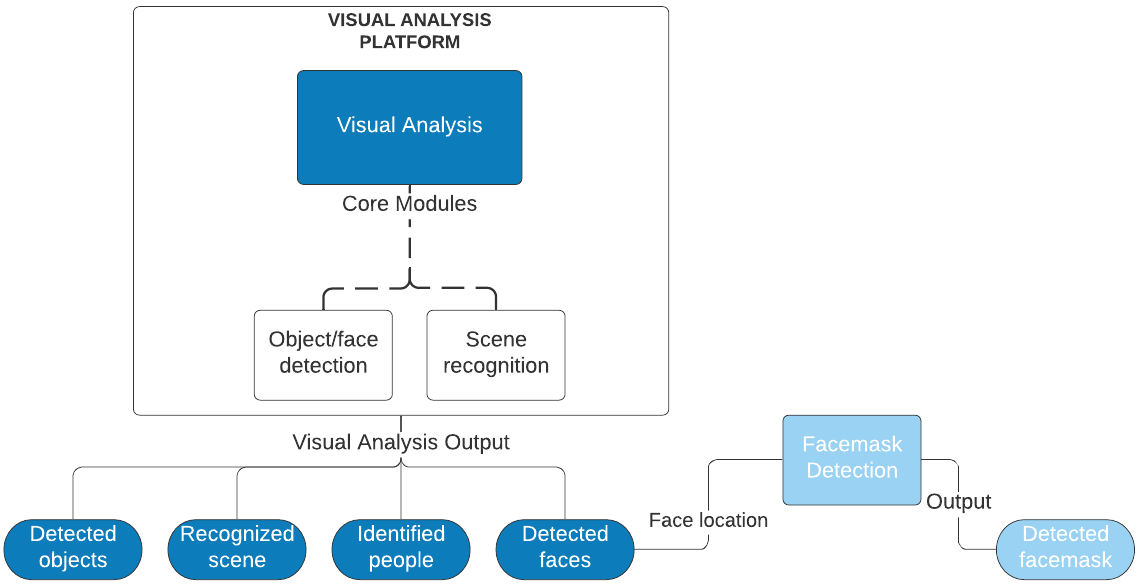


Figure 2.6. Visual Analysis Platform and Facemask module

For example, for the analysis in Figure 2.7 among other objects, the face of a person is detected along with the respective bounding box. The facemask module utilizes this detection and in turn detects whether this person wears a facemask or not. In this case a facemask was detected.

The Covid-19 pandemic led the community to a consistent effort to tackle the issue of face-mask detection. Loey et al., (2021) utilize an ensemble classification method, based on a Resnet50 feature extractor, using three datasets in the process, which are Real-World Masked Face Dataset[6], the Simulated Masked Face Dataset[7], and the Labeled Faces in the wild (Kawulok, Michal, Emre Celebi, and Bogdan Smolka 2016). Chowdary et al., (2020) employ the Inceptionv3 (Szegedy, Christian, et al.,2016) on the task of facemask detection and they use the Simulated Masked Face Dataset.

The model, that we developed, is based on ResNet50 (He, et al., 2016) network, on the task of detecting if people are wearing facemasks or not. Also, image augmentation has taken place, to improve the training process. The faces of people are already detected by the object detection module of the VA platform, which enables fast and accurate detection of facemasks. The datasets that we utilized are the Real-World Masked Face Dataset and Simulated Masked Face Dataset. In ideal conditions, this module is able to identify consistently if a person wears a facemask or not, however light conditions and various angles may affect the overall accuracy of the module. Thus, it is not possible to detect if a person wears a facemask, if she does not face the camera since in a sideways view the face detection module does not capture the area of the mask. The trained model is able to identify if someone is wearing a mask with 97.8% accuracy in ideal conditions. The facemask module is able to recognize every kind of mask that covers the mouth and the face

[6] *https://github.com/X-zhangyang/Real-World-Masked-Face-Dataset*
[7] *https://github.com/prajnasb/observations*

and is not limited to the standard surgical masks. However, the usage of masks makes the task of face recognition harder, since in order for a person to be identified characteristics from the whole face are required and a mask covers a large portion of the face. In Figures 2.7 and 2.8, a person with and without mask respectively is shown.

Figure 2.7. Person with facemask

Figure 2.8. Person without facemask

2.4 Standalone version

The standard VA methodology that we employ relies on a remote server as we have already mentioned in chapter 2.2.3. This way, with the utilization of the server capabilities, fast and accurate VA is possible. However, a stable network connection is paramount, in order for the stream of images to be constantly uploaded to the server, as shown in Figure 2.5. Even though the size of each image, which is less than 100 kilobytes, is not that large to warrant a fast network connection, there may be potential cases where the network upload speed is very poor, which would render the standard VA really slow in many situations.

For this reason, we developed a version, which does not rely on the Server for the VA but it conducts it directly on the wearable device, which in our case is the Raspberry Pi 4 Model B. This version is called standalone VA. This way, the server is not needed and only the bare minimum network speed is required for remote communication with the Ontology, relaying a Json message with the analysis results, as there is no need for an image upload. However, since the computational capabilities of the RPI, as expected, are limited compared to the server capabilities, the algorithms that are developed and utilized are less computationally heavy, which in turn results in a degraded performance in comparison to the standard Visual Analysis.

The standalone Visual analysis is not meant to replace the standard Visual Analysis. Lighter object and scene recognition and detection modules were developed and embedded. Also, the standard Face and Facemask recognition modules have been embedded in two separated, different standalone versions as due to the limited capabilities of the Raspberry Pi it is impossible for every module to run simultaneously.

2.4.1 Object and scene detection and recognition

For object detection, a network that relies on the convolutional network MobileNetV2 (Sandler, Mark, et al.,2018), has been utilized. This network, uses the Single Shot Multibox Detector methodology to predict the potential objects with classes. The MobileNetV2-based detector is able to recognize the same number of classes as the server version. Due to the restrictions that have been set from the limitations of the capabilities of the Raspberry Pi, the device which is on the prototype HIPI that executes the computations, usage of a more computationally heavy methodology is impossible.

Similar to object detection, for the scene recognition task, it is not possible to integrate the standard server-based algorithm, due to computational power constraints of the Raspberry Pi board. Thus, a transfer learning network based on the Resnet-v2 (Szegedy, et al., 2017) backbone network model was developed, with the objective to require less memory and computational power for inference. The aforementioned scene recognition model is trained on the combination of SUN (Xiao, et al.,2010) and the MIT Places (Zhou, et al., 2014) datasets. The number of detected classes are 15, the same with the standard scene recognition model (Table 2.2). From initial testing, it has been observed that this scene model is less susceptible to misclassifications when the camera is in peculiar angles, as in for example very close to surface or the camera is in an angle, but is able to classify in a lower rate, only when the incoming images show the whole scene and not part of it.

In order to understand the capabilities of the developed standalone version, as well as its limitations, extensive testing was required. To aid and speed up that process a version that emulates the standalone version was developed on the server. This version includes all the algorithms for object, scene and face detection and recognition that were used in the Standalone version, but this time they run on the Server side (just for testing purposes). The camera feed from the RGB-D sensor is uploaded to the server just like the standard server Visual Analysis module, but the analysis is done by the algorithms developed for the Standalone version. This allows for quick comparisons with the other versions, testing with depth information when the RGB-D sensor is not available locally and finally due to the lightweight nature of the algorithms, this version is much faster than any other developed one, even if it impacts the overall performance in terms of detection accuracy in some cases. On the Table 2.4 below, metrics that show the differences between the server-based version and the standalone version are listed.

Table 2.4. Standalone and server-based VA comparison

Versions	***Object detection on chair***	***Object detection on face***	***Scene recognition (office)***	***Total Inference time (sec)***
Standalone	92%	92%	73%	2

Server-based	100%	100%	85%	1.3

In the Table 2.4 above the server version is compared to the standalone one, on the same tasks. It shows how each object detection version fared with different objects on a realistic scenario, how many times the scene was recognized and how much time the analysis for every image took. As expected, the server-based version performs better on every task. The standalone version is much lighter so as to be able to run on the local device, which leads to poorer results, when compared with the Server-based version, however it can still reliably detect and recognize objects and scenes . The total inference time metric measures how fast the Server-based version runs on the Server and the Standalone on the Raspberry Pi, when processing an image.

2.5 Chapter Summary and Future Work

In this chapter, various corrections and improvements of the developed algorithms have been reported. Also, comparisons with alternative developed algorithms have been made in order to select the ones that fulfil our needs. Furthermore, a new module (face mask detection) has been developed, utilizing the platform's capabilities, as well as a standalone version that does not require a remote server to conduct the computations.

In future work, corrections and further changes to the algorithms, could be made from the results of tests that would be conducted with the assistance of deafblind people. Due to the pandemic crisis this has not been possible so far. Additionally, the effectiveness of the algorithms, in various adverse conditions, such as in poor light conditions could be further investigated, even though from initial testing it excelled in face recognition, where it was still able to recognize faces even in poorly lit rooms. With the continuous improvements of hardware devices, more computationally powerful devices than the Raspberry Pi could be chosen in the future, that would enable the Standalone Visual Analysis to utilize more computationally heavy algorithms, which in turn would result in better overall performance. Lastly, additional modules could be built on top of the Visual Analysis platform, one of which a Simultaneous localization and mapping module (Durrant-Whyte, Hugh, and Tim Bailey, 2016), that would allow better understanding of the surrounding environment.

3 Haptic Communication Design

In D3.2 we defined a haptogram as "a tactile symbol drawn over a touchscreen, its dynamic nature referring to the act of writing or drawing, where the touchscreen can take several forms, including a smart textile garment designated for specific areas on the body". We further elaborated on the concept of haptograms in Darányi et al., (2020) as "synthetically generated haptic patterns with a meaning, to be communicated as stimuli to the human body". Such haptograms can be mapped to one or more parts of the body, in single mode or multiple modes.

The purpose of haptograms in our framework is to implement an ontology-constrained messaging language for situation awareness information, leading to simple conversation. The idea presented here is to utilize vibrotactile actuators to display haptograms in a matrix of rows and columns, i.e., an *m* x *n* arrangement where *m* and *n* are not necessarily of equal numbers.

Following the work presented in D3.2 (which included communication design considerations, semantic theories underlying haptogram design, and haptogram examples), our efforts have evolved in a number of areas as described below.

3.1 Progress roadmap

Despite the problems caused by Covid-19, measures taken in the project enabled progress in several respects:

- We invested effort into the co-design of haptograms with the inventors of Social Haptic Communication (SHC). More details of this are presented below in the Participatory co-design effort section;
- We realized that a single 4x4 display on the back, as used in D3.2, does not allow for the copying of sophisticated SHC signs. In response to this finding, we explored the idea of a multipanel suit of multiple displays over several body parts with different grid sizes, and how distributed messaging could encode haptic messages. Using 4x4+2x1+1x5 = 23 distributed actuators provided by the HIPI, we worked out a new principle called *double contextualization* to construct and convey responses to precoded queries (PCQ) based on the ontology. PCQs and short messages as haptogram sequences were completed on a proof-of-concept level, integrated into the latest HIPI demonstrator. Appendix A shows how moving over from a 4x4 to a 5x5 grid would allow for more standard encoding opportunities;
- We managed to develop an open source tool for the community-based co-design of haptic communication elements, whose testing is still ongoing. Combined with the availability of the "chairable" and the Tactile Board prototypes, user communities thereby will have a sustainable solution to customize their own haptogram designs e.g., for and in a classroom setting.

3.2 Constraints on haptogram design

To integrate available project components from a communication perspective on a proof-of-concept level, we departed from the following constraints:

- The current vocabulary is limited to an indoor environment;
- We worked with a worst-case scenario, i.e., complete dual-sensory loss of the user as our starting point, where every element of communication must be predefined because none will be supported/replaced by visual or audible information;
- Haptograms were encoded as complexes of parametrized physical constituents, aka *haptemes* for components of touch, to symbolize “tangible ideas”;
- Based on psychophysical experiments, static patterns on the back were discarded in favour of dynamic haptograms, as these are better perceived by a human receiver. Other recommendations included interpolation of signals for improved perception of continuous lines, spacing between the actuators, and ways of stressing key elements.

3.3 New results

Progress in WP4 and WP5 enabled us to propose a more advanced variant of communication by haptograms. Due to an expansion of the vocabulary by combining concepts in the ontology with ones outside of its scope, the current haptogram kit has 155 concepts instead of 104 in D3.2, including short messages, contexts, visual-detection-based scenes as topics, objects, persons, and relations. Having sufficiently grounded the use of haptograms in linguistics in the previous report, this time we solely focused on the use of *distributional semantics*, where the meaning of a word follows from its context.

Currently, complex messages use three categories, namely purely syntactic haptograms such as *context indicators* to disambiguate topics in context (“kitchen in time” vs. “kitchen in space”); *topic indicators* to differentiate between content haptograms (“window in kitchen” vs. “window in living room”), and *topic fillers* (e.g., “mobile”, “chair”, “table”, “glass”, “stranger”, etc.). With six contexts (“space”, “time”, “person”, “emotion”, “activity”, “range”), and multiple topics which correspond to visually detected scenes labelled by the ontology (e.g., “bathroom”, “corridor”, “elevator”, etc.), a number of objects and persons can be distinguished according to spatial and temporal relations. Most of them are mapped to the back as display, augmented by tap sequences on the shoulders and in the belt area, respectively. Figure 3.1 illustrates the current distribution of incoming haptograms.

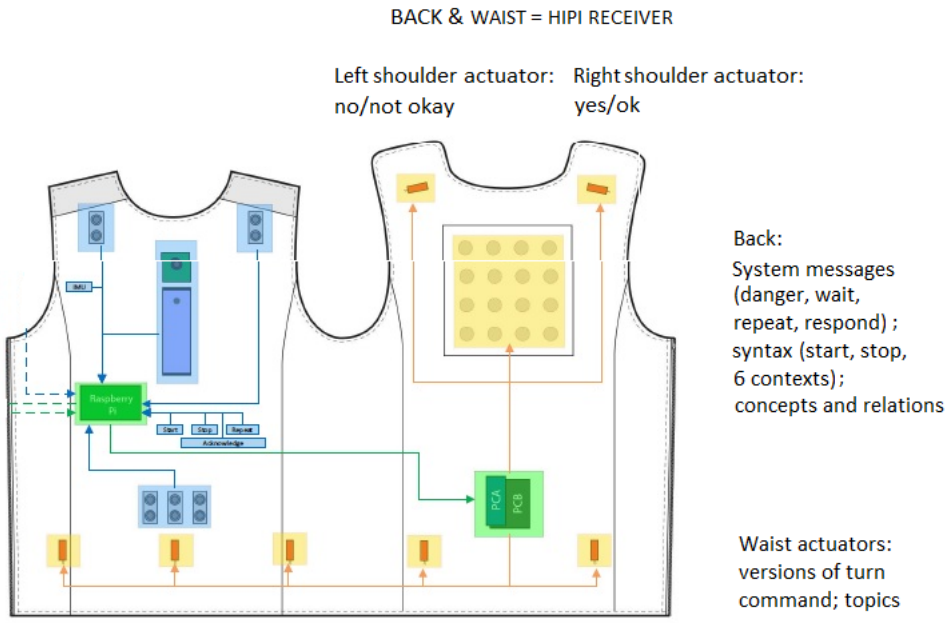


Figure 3.1: Distribution of haptograms according to their types on the multipanel suit.

This approach allows for the *double contextualization* of semantic content, i.e., framing concepts by topics, and in turn, topics by their contexts. The result is a simple haptic language whose grammatical structure goes beyond haptogram sequences as presented in D3.2, at the cost of repetitive -- and thereby hopefully more learnable -- syntactic elements. Figure 3.2 illustrates the schema of the communication process.

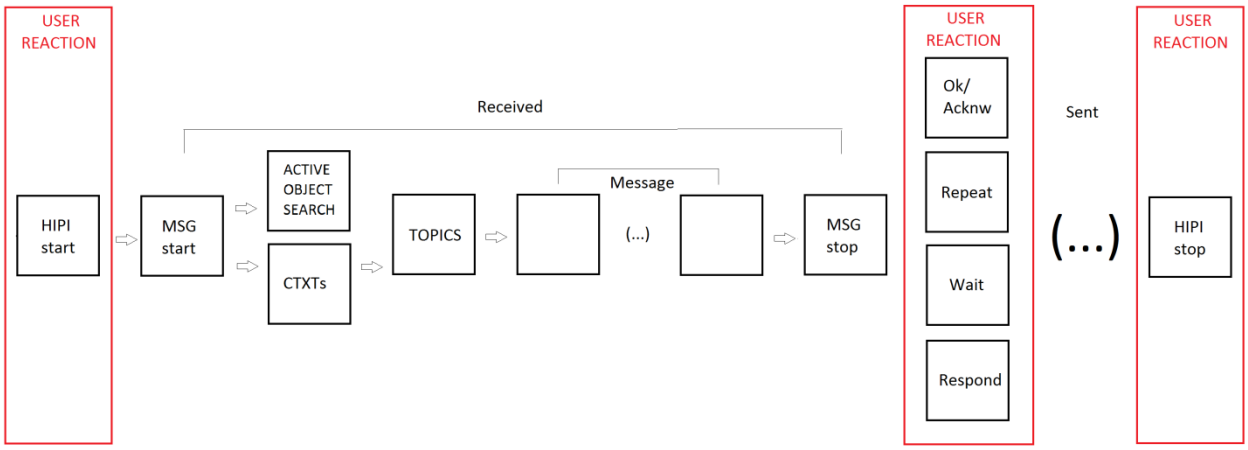


Figure 3.2: Message structure using different categories of haptograms mapped to one or more body parts.

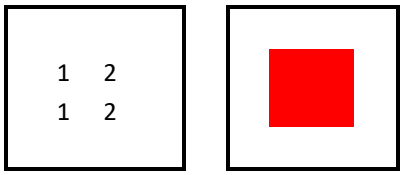
Mapped to the back in this example, the reply to a precoded question from the user looks like this:

PCQ1: "Where am I?"

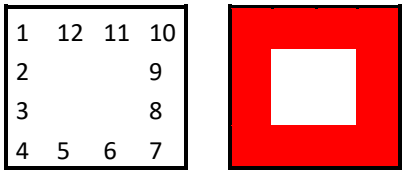
A1: “You are in the kitchen.”

Numbers in left-hand side squares denote order of vibration. Red patterns in right-hand side squares denote total vibration pattern.

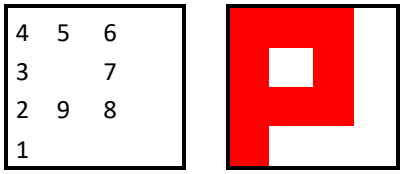
system:msg_start



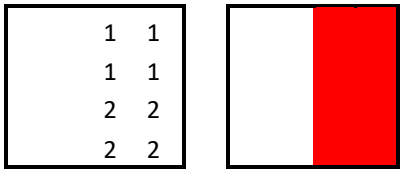
system:ctxt_start



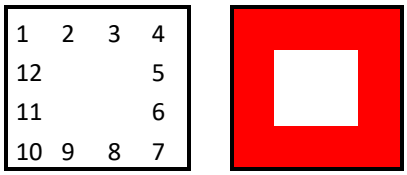
context:person



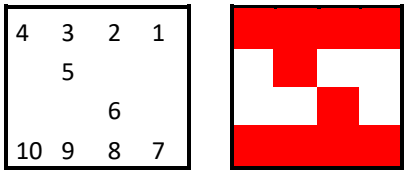
you



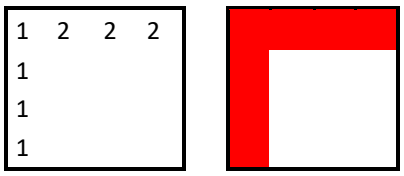
system:ctxt_stop/change



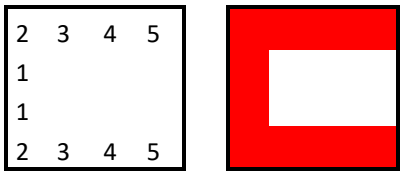
context:space



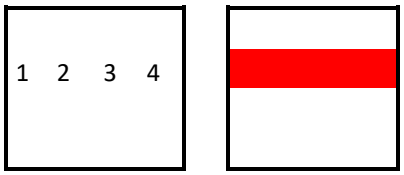
space_direction:in/in front/ahead



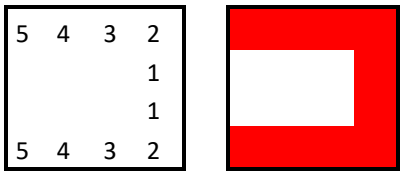
system:topic_start



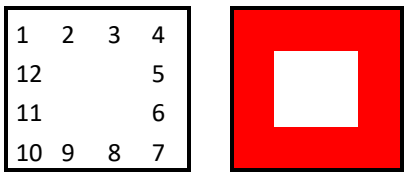
topic:kitchen



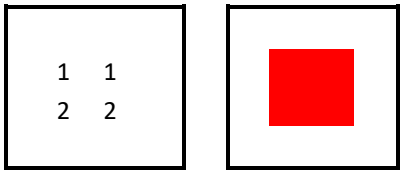
system:topic_stop/change



system:ctxt_stop/change



system:msg_stop



3.4 Participatory co-design efforts

For people with deafblindness, tactile sign language is a common mode of communication. More recently, Social Haptic Communication (SHC) (Lahtinen 2008, Lahtinen and Palmer 2010) also gained momentum. In SHC, a signer conveys messages to a receiver by hand movements and gestures on appropriate body parts of the receiver.

A couple of decisions were made early in the project to (a) develop haptogram patterns in a way to closely resemble SHC signs in order to facilitate recognition; and (b) involve potential users in participatory co-design as well as testing and improvements.

Based on these decisions a number of workshops were planned to take place in April 2020, but due to the situation caused by the pandemic, our access to, and interactions with the users were severely influenced, and as a result those plans had to be abandoned. Our haptogram co-design efforts thereby became limited to a number of online ideation workshops with the initiators of SHC, Riitta Lahtinen and Russ Palmer. An outline of the ideation process, combined with respective hardware solution development to increase the resolution of multipanel displays, is outlined in Appendix B. In turn, that hardware solution is underlying the development of a new haptogram design toolkit introduced in the next section.

3.5 Haptogram design toolkit

As the number of haptograms grows and since variations in design are needed, manual design of the actuation patterns soon becomes tedious and problematic. To facilitate our efforts in defining haptograms, we designed a toolkit that enables the user to engage in pattern definitions involving a set of actuator motors (for now, from just a few, up to 96 actuator motors) in a user friendly yet sophisticated fashion.

The haptogram designer tool provides a space for adding new haptograms. On pressing the “Add” option, one is prompted to choose a grid size, the current options being 3x3, 4x4, 5x5, 6x6 and 12x8 (Figure 3.3).

Each of the nodes in these grids corresponds with one actuator motor. The idea is not, however, to create a display with this number of nodes on one body part, rather, the designer will make the decision about which node is to be placed where. For example, if one wishes to create a grid of five-by-five actuator motors on the back, two actuators on each shoulder and seven around the waist, totalling 36 actuators, then one can choose to use a grid of 6x6 on the toolkit, while keeping the location of nodes noted for ease of use (Figure 3.4).

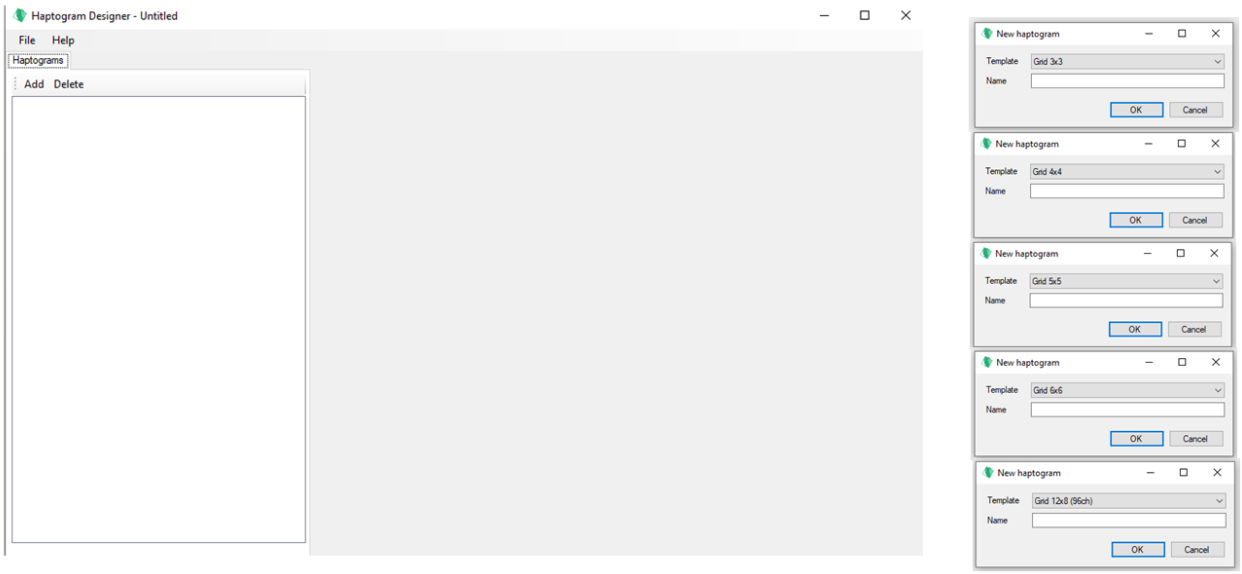


Figure 3.3: Haptogram designer start page and grid size options.

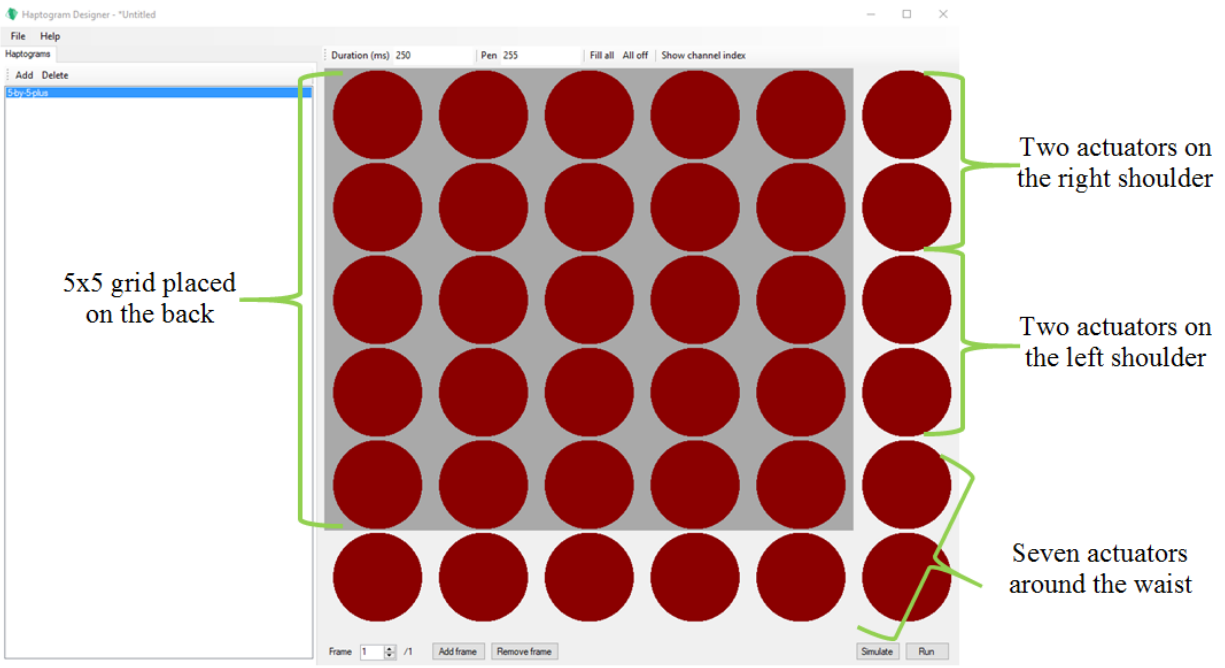


Figure 3.4: One can dedicate different nodes to actual vibrotactile motors on different parts of the body.

Once a grid size is selected and a name is given to the haptogram to be designed, one can easily define any desirable pattern of actuation for each haptogram. With this tool one can define (i) the nodes to be activated, (ii) their order of activation, (iii) the intensity of vibrotactile

stimulation, (iv) the duration of each activation, (v) the duration of pause between each activation, and (vi) whether the order of activation is sequential or simultaneous or both (Figure 3.5). Furthermore, it is possible to show the channel index to help the toolkit user to keep an overview of the associations between each node and their decided placement on different body parts.

If a haptogram is made up of many frames, one can choose to make adjustments in any of the frames directly, rather than having to restart and define all the details from the beginning. By thus defining each of the haptograms, the number of entries in the haptogram vocabulary grows. The defined haptograms can also be deleted if one so chooses (Figure 3.6).

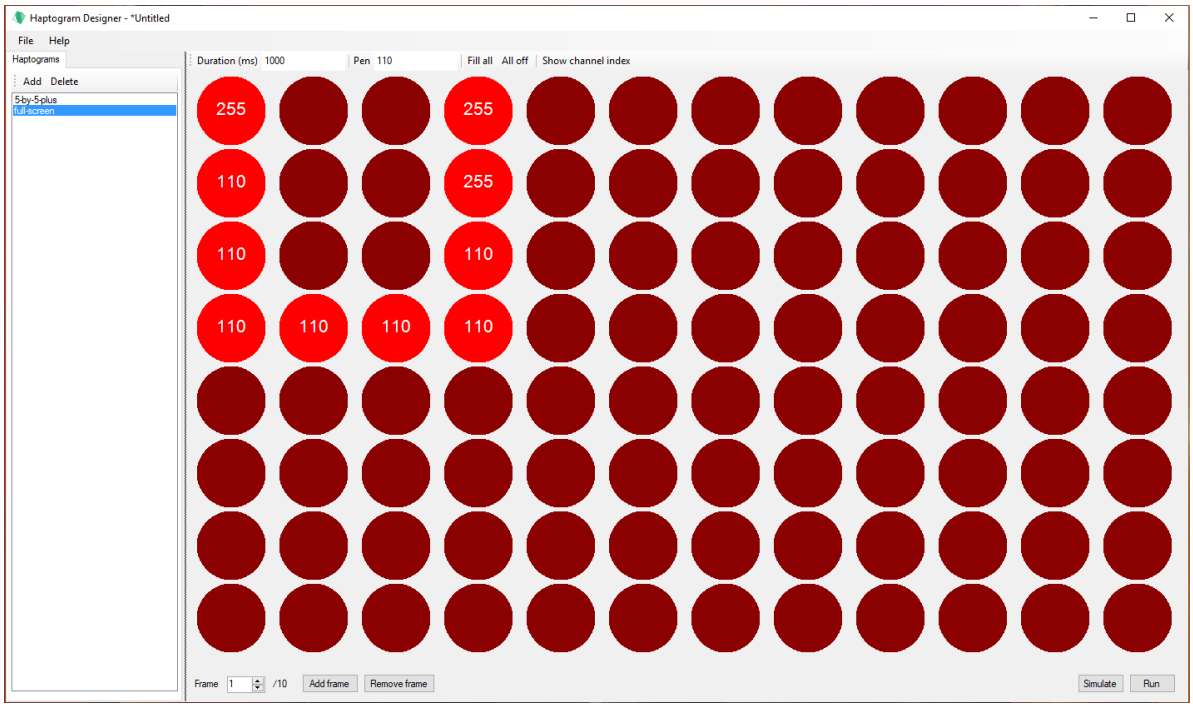


Figure 3.5: The intensity and duration of each activation can be defined.

Once the complete pattern for a haptogram is defined, one can choose to either simulate or run it, where the former option displays the pattern on the computer screen, and the latter will activate the actuator motors placed on the body.

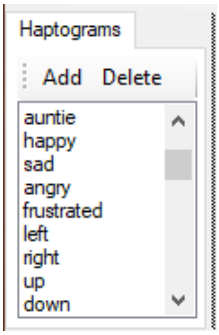


Figure 3.6: Adding to and deleting items from the haptogram vocabulary.

3.6 Summary of haptogram design results and future work

We demonstrated the potential of haptograms for future studies and applications. With the feasibility of its design principles illustrated, *in vivo* continuous 2-way communication about a set of topics between a user and the HIPI, or two or more users, is not the case yet, but *in silico* availability via the platform already is.

At this point in time, it was neither desirable nor realistic to plan with a canonical haptogram library for a proof-of-concept. With customized solutions to individual and/or community needs for developers now within reach, the sustainability of haptograms is provided. To this end, we focused on a solution which combines active object search with their labelling on word, phrase ("in the elevator", "next to you"), and sentence level by means of expanding vocabularies and emerging grammars. This latter component offers a range of possibilities by the use of multiple panels as displays over the body, allowing for localized vs. distributed patterns in one or more codes, so that these can combine both syntactic and disambiguating elements to represent semantic content.

As research on haptic communication technologies grows, and the need for definition of actuation patterns and pattern encoding becomes more wide-spread, it is very likely that our haptogram design toolkit will be of value even for others dealing with use of actuators. Although the system described here is already easy to use, we intend to develop this tool further (even beyond the end of SUITCEYES) for extended usability and functionality to meet further needs. We have sought and are likely to receive external funding to support our proposed improvements. Among others, an aim is to make this tool available online so that anyone can draw the patterns that they wish using our online interface and receive the encoded actuation instructions simply and without the need for coding skills or the need to refer to Gidhub for more technical instructions.

4 Semantic Knowledge Representation and Reasoning

In the current section, we aim to describe the 3rd (and final) version (v3) of the implemented SUITCEYES ontology and the semantic reasoning framework. The ontology (KB, Knowledge Base) was extended to fit the needs of the users'/project requirements and to the advanced functionality of the HIPI (further from D3.2). Moreover, the Knowledge Base Service (KBS) was improved and extended to enable the communication between the components of the HIPI system and the ontology and to store the incoming data in a more efficient way. In addition, the Semantic Reasoning Mechanism (SRM) was enriched to cover a wide range of queries to semantically improve the awareness of the user's surrounding environment, either in the form of structured content (JSON format) or as natural language phrases. Using these outputs, the HIPI system conveys information to the user by using haptograms. The SRM was also extended to handle multiple entity output sentences, i.e., the previous version's output: "*A <laptop> has been found <very close> to you. A <chair> has been found <very close> to you.*" has been integrated in a single sentence: "A <laptop> and a <chair> have been found <very close> to you. Finally, we developed a web interface call Knowledge Base Dashboard (KBD) to visualize the processes executed within the HIPI system and more specifically the communication between the VA tool, the iBeacon sensors, the KBS and the user.

4.1 Additions to the SUITCEYES Ontology

In this deliverable, the KB was extended with new concepts to fit the needs that emerged in this period of the project. For example, during the COVID-19 pandemic outbreak, there was a need for the DB-user to know if the person near them is wearing a facemask or not. This was translated in the KB as a data property called wearsFacemask, which refers to either a Known of UnknownPerson entity (described in D3.2, section 4.1.1).

Also, the ontology was enriched with more entities like rooms and objects to cover a wider variety of places so that, in combination with the information received from the VA tool, would enrich the environmental awareness of the DB-user by responding to queries. More specifically, regarding the added vocabulary to the ontology, we present below the current state of the entities and properties that exist in the SUITCEYES ontology. Note that these classes complement the ones that we have integrated from the Dem@care and the Seas ontology (presented in D3.2).

In Figure 4.1, we present the subclasses of the Object class that are the added objects to the ontology. In Figure 4.2, we present the additional Semantic_Space subclasses, which are the added scenes that the ontology can store. In Figure 4.3, we present the wearsFacemask data property.

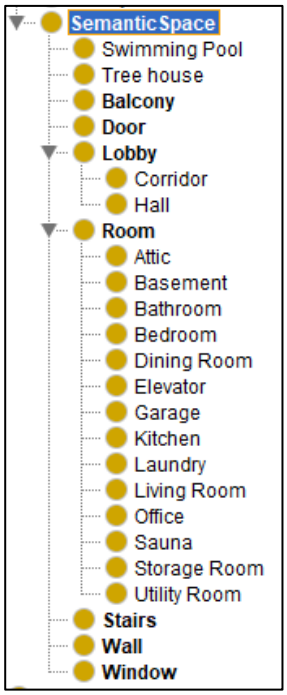


Figure 4.1: Object Class

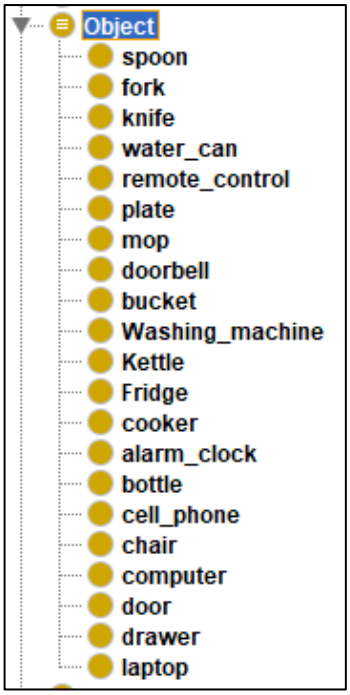


Figure 4.2: Semantic Space Class

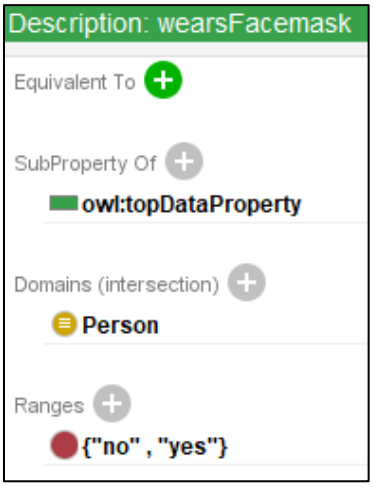


Figure 4.3: wearsFacemask data property

4.2 Knowledge Base Service (KBS)

In this deliverable, we present the development extensions and improvements of the KBS. The KBS is a JAVA application that implements the functions and procedures that receive information from the different components of the HIPI, and convey semantical information to the DB-user through the use of the KB. The KBS and KB interact through the Rdf4j library, which enables the KBS to receive the semantic reasoning process (inference) output and forward it to the HIPI library to be converted to haptograms that will reach the end-user.

4.2.1 KBS Communication Process with the HIPI system

The need for a change of the communication process emerged during the reporting period, due to the shutdown of the Realtime.co messaging framework that was used in the second version of the KBS. As also described in section 2.2.3, the messaging framework was replaced with the MQTT[8] protocol which is an OASIS standard messaging protocol for the Internet of Things (IoT) in combination with the Ably[9] broker, which is a publish/subscribe messaging platform with a suite of integrated services to deliver complete real time functionality directly to end-users.

Furthermore, communication between the user and the KBS was added through a haptic device presented in D7.5. In this way, the user can send queries to the ontology with the form of sentences as strings in a JSON file. The user can send predefined queries with the use of this

[8] https://mqtt.org
[9] https://ably.com

haptic device, like “Where is my <object>?”, “Where am I located?”, “Is the person near me wearing a mask?” and other queries in regards with the user’s spatial surrounding information.

In addition, the feedback of the ontology to the system during the active object search procedure was changed. In particular, instead of the actuator patterns that were sent (described in D3.2, section 4.3.1), the ontology now sends a JSON file that includes useful information about the surroundings of the user, including the last time an entity was detected, whether the user has moved or not and other information useful to the HIPI (more in section 4.2.4).

For each HIPI component, a different communication channel is created for the exchange of information. The information received and sent is in JSON format, which offers great flexibility and ease of use, because it is easy to understand and does not take into account the programming language or any other specifications.

The communication channel for the exchange of information between the KBS and the VA tool is called “VA_KBS_channel” and it is the way that the VA tool sends the detection information to the KB. This communication channel is “one-way”, meaning that the VA component only sends information and the KBS only receives information. The information in JSON format can be seen in Table 4.1a/b. This JSON has two more extra fields from the previous iteration of the project, the facemask and similarity fields. The facemask field can take to values “yes” or “no” and refers to if a detected person wears a facemask. The similarity field can take integer values, and low values means that the consecutive frames are almost identical (more on that on section 4.2.3).

The communication channel responsible for the exchange of information between the KBS and the iBeacons is called “iBeacon_KBS_channel”. The data sent there have remained the same since the iBeacons have specific capabilities and are presented in Table 4.1c

The channel responsible for receiving the queries of the end-user is called “User_to_Ontology_channel”. This is also a one-way channel which receives a JSON file with the query and the query timestamp. The information in JSON format can be seen in Table 4.1d.

The communication channel responsible for giving data for the Active Object Search (AOS) procedure is called “Active_Object_Search_channel” and is a one-way channel that the KBS uses to send latest detection information to the HIPI system so that the AOS module can operate. The information in JSON format can be seen in Table 4.1e.

We also have two more communication channels for our KDB. The first channel is called “Query_Answer_Interface_channel” and it sends the query and the answer to the interface so that they can be displayed. The second channel is called “Active_Object_Search_Command” channel and it is used to display the commands that the user receives when searching for an object, i.e. “Move forward” or “Turn right”, etc. The respective information in JSON format can be seen in Table 1d (same JSON as before) and Table 4.1f.

In the table below, we present the JSON format structure that the ontology receives and sends to the various components of the HIPI system.

Table 4.1: JSON structure of incoming messages from different components

(a) incoming message from the VA component

```
{
  "header": {
    "timestamp": "2021-03-22T11:42:31.154",
    "message_id": "VA_sksk232ksjfj2k32kola",
    "recipients": [
      "KBS"
    ],
    "sender": "VA"
  },
  "body": {
    "data": "http://160.40.50.243:8008/upload/storage/ VA_sksk232ksjfj2k32kola.json"
  }
}
```

(b) message structuring analysed data from the VA, stored in the provided URL

```
{
 "image": {
  "name": "31_03_21_09_20_24_424675",
  "scene_type": "office",
  "height": 480,
  "width": 640,
  "target": [
   {
    "confidence": 0.805,
    "distance": 2862.667884269728,
    "height": 159,
    "width": 112,
    "left": 184,
    "facemask": "no",
    "type": "chair",
    "top": 320
   }
  ],
  "scene_score": 0.0,
  "timestamp": "2021-03-31T09:20:24.424",
  "similarity": 15.0
 }
}
```

(c) incoming message from the iBeacon sensor

```
{
  "timestamp": "22-08-19 11:42:30",
  "data": [
    {
      "distance": 5390.0,
      "positionX": 0.0,
      "positionY": 0.0,
      "name": "chair",
      "rssi": 137,
      "message": "Immediate",
      "id": 5
    },
    {
      "distance": 4582.0,
      "positionX": 0.0,
      "positionY": 0.0,
      "name": "table",
      "rssi": 171,
      "message": "Near",
      "id": 1
    }
  ]
}
```

(d) incoming message from the DB-user using the haptic device

```
{
  "timestamp": "12-03-21 11:42:30",
  "query": “Where am I now located?”;
}
```

(e) Outgoing message from the KBS to the actuator module

```
{
  "timestamp": "18-93-21 09:47:05",
  "VA_detection": True,
  "offset": -25,
  "distance": 2.0}
}
```

(f) AOS command message sent to the KDB

```
{
 "entity": "laptop",
 "move_command": "Turn left"
}
```

4.2.2 Natural Language Output/Multiple Entity Reports

In this deliverable, describing the 3rd version of the SUITCEYES ontology, more rules were added to support the semantic spatial representation of the user's surroundings, as well as incorporating a set of rules to present multiple entities output in one sentence with the use of the KBS. These inferences were designed on the basis of specific scenarios derived from the user and technical requirements.

We extended the KB reasoning mechanism to efficiently handle the integration of similar content within the context of a produced natural language phrase. As described in D3.2, the previous version of the KB was capable of handling single entity output reports; this means that if, for example, more than one result is inferred from the reasoning mechanism, then the current version of rules produces one output sentence per detected entity, i.e., “A <laptop> has been found <very close> to you. A <chair> has been found <very close> to you.”. With the use of new

queries, the ontology now can produce multiple entity reports, such as: “A <laptop> and <chair> have been found <very close> to you.

This multiple entity natural language output can be produced by using a SPIN/SPARQL[10] function (that is the SPIN/SPARQL syntax of the user’s query, a JAVA function to group the reported entities together and to send a natural language output in JSON format to the haptic library, so that it can be translated to haptograms), and a simple INSERT SPIN/SPARQL function, so that this output can be stored in the ontology for future use.

In Figure 4.4, we present the query that corresponds to the user query: «Which objects are near me?». Note that we measure the distance in millimeters, that is why the distance value must be less than 5000 (5 meters).

```
SELECT ?detection ?timestamp ?slabel ?elabel ?leftright ?distance
    WHERE {
        ?detection a sot:Detection .
        ?detection sot:hasTimestamp ?timestamp .
        ?detection sot:detectsSemanticSpace ?scene .
        ?detection sot:detectsObject ?entity .
        ?entity rdfs:label ?elabel.
        ?scene rdfs:label ?slabel.
        ?scene rdf:type sot:SemanticSpace .
        ?entity sot:hasXGridSpace ?x.
        ?x sot:positionedInXGrid ?leftright.
        ?x sot:hasOffset ?offset.
        ?entity sot:hasSpatialContext ?sc.
        ?sc sot:definesAbsoluteDistance ?distance.
        FILTER(?distance < 5000)
        }
        ORDER BY desc(?timestamp)
```

Figure 4.4: SPIN/SPARQL function for query: Which objects are near me?

In Figure 4.5, we present the answer of the query. A door and a chair are detected near the user, on their left side.

sot:detection_655	"2021-03-31T10:10:36.436"^^xsd:dateTime	"office"	"chair"	"left"	"2831.4158"^^xsd:float
sot:detection_655	"2021-03-31T10:10:36.436"^^xsd:	"office"	"door"	"left"	"4537.9883"^^xsd:float

Figure 4.5: Answer of the query: Which objects are near me?

Our JAVA method, using simple rdf4j functions to retrieve the query’s result, groups the objects chair and door together and send the output: «A chair and a door was detected near you, on your

[10] Spin/SPARQL is the current industry standard to represent SPARQL rules and constraints on Semantic Web models

left side, X time ago», where X refers to the difference in time between the query's timestamp and the detection's timestamp. Finally, with the use of the rdf4j insert function, we store this output in the ontology.

4.2.3 Efficient storing of incoming data

We developed a way for a more efficient manipulation of data received from different components to the KB. One of the findings was that the KBS should handle the similar reports arriving every second in the KB, and populate only those that report a different situation than the previously reported one. This way, the population of duplicate information in the ontology will be avoided, something that is critical for the KBS response time.

The VA tool sends to the KBS information regarding the similarity of consecutive frames. In particular, the JSON file with the detection that we receive from the VA tool contains the field SIMILAR. If it is a "yes", then the previous frame has not changed drastically from the current frame. From this information, we can assume that every entity contained in the current frame is the same as the previous one.

We have implemented a method in java that checks the similarity field in the incoming JSON files. If the consecutive frames have a high degree of similarity, then with the use of SPIN/SPARQL queries we take the latest detection that was saved in the ontology and replace its timestamp datatype with the timestamp of the most current frame's detection. As the pictures of the detections are similar, the remaining information of the current detection can be omitted. In Figure 4.6, we present the query that we use to return the latest detection and the most important information, each time the "similar" field is yes and in Figure 4.7, we present the data that we receive after executing the query. In Figure 4.8, we present the SPIN/SPARQL function that we use to replace the timestamp datatype of the previous frame detection, with that of the latest frame's. The 'old_timestamp' in the DELETE command refers to the timestamp of the latest saved detection and the 'new_timestamp' in the INSERT refers to the timestamp of the current, similar frame.

```
SELECT ?detection ?timestamp ?slabel ?elabel ?leftright ?distance
    WHERE {
        ?detection a sot:Detection .
        ?detection sot:hasTimestamp ?timestamp .
        ?detection sot:providedBy ?sensor .
        ?detection sot:detectsSemanticSpace ?scene .
        ?detection sot:detectsObject ?entity .
        ?entity rdfs:label ?elabel.
        ?scene rdfs:label ?slabel.
        ?scene rdf:type sot:SemanticSpace .
        ?entity sot:hasXGridSpace ?x.
        ?x sot:positionedInXGrid ?leftright.
        ?x sot:hasOffset ?offset.
        ?entity sot:hasSpatialContext ?sc.
        ?sc sot:definesAbsoluteDistance ?distance.
        }
    ORDER BY desc(?timestamp)
    LIMIT 1
```

Figure 4.6: Spin Function for query that returns the latest detection

	detection	timestamp	slabel	elabel	leftright	distance
1	sot:detection_656	"2021-03-31T10:10:42.442"^^xsd:dateTime	"office"	"chair"	"left"	"2772.3657"^^xsd:float

Figure 4.7: Answer of latest detection query

```
DELETE { ?detection sot:hasTimestamp 'old_timestamp' }
INSERT { ?detection sot:hasTimestamp 'new_timestamp' }
WHERE
  { ?detection sot:hasTimestamp 'old_timestamp'
  }
```

Figure 4.8: SPARQL/SPIN function for replacing the timestamp

In real life scenarios, as a DB-user can look towards a certain direction for a long time, the aforementioned process can save a lot of storage space in the ontology. For example, if a user faces a scene that it has no new information for 30 seconds, we now store only 1 detection to the ontology instead of the 30 detection that were stored before, as the VA tool sends us detection data every second. This is 29 less detections in the span of 30 seconds, greatly decreasing the volume of the incoming saved data.

4.2.4 Active Object Search module

In this deliverable, we present the updated and improved version of the Active Object Search module (introduced in D3.2, Section 4.3.1). The changes that we implemented in this module were due to the new technical requirements of the HIPI system.

In particular, effort was put to make the Rasberry Pi (RPi) more independent and as a result the control decision and motor signals are, during this iteration of the HIPI system, computed locally on the RPi. Therefore, the fusion of the VA and iBeacon data is done on-board. In the previous iteration of the HIPI system, the data were captured by a Bluetooth scanner and were sent to the KB. There, the ontology did the processing and sent back motor commands.

In the current setup, the data are captured using the RPi 4 embedded Bluetooth antenna and filtered locally on the RPi, so there is no need for the KBS to combine the VA data and the Ibeacon data to produce motor commands, as the Ibeacon information is already stored locally. However, the iBeacon data are still needed in the ontology, as they help to answer simple queries like: «Where is my laptop?».

The AOS module works as following: First, the DB-user sends a query about a specific entity like «Find my laptop». This query activates the AOS procedure. The KBS continuously searches if the entity is detected in the latest detection sent by the VA tool. If the entity is detected, the KBS sends a JSON file (presented in Table 1e), with the relevant information back to the RPi, using the Active_Object_Search_channel introduced in Section 4.2.1. The requested information is a) the timestamp of the detection, b) a True/False value of if the requested entity is detected, c) an offset value that show if the entity is left or right in the frame, and d) the distance of the entity that the user is searching for. Using these values, the HIPI system determines the command that must be sent to the user as haptic feedback. The list of the current commands in natural language is: turn right, turn left, move forward, search and target reached.

4.2.5 Knowledge Base Dashboard for HIPI system visualization

In this deliverable, we present the web application that was developed, called Knowledge Base Dashboard (KBD). This application was developed to visualize the processes run on the HIPI system in real time, so that it can be a useful tool available to "experts", project members, knowledge base managers, testers of VA/Beacon components and others to test and demonstrate live the flow of the HIPI process.

In Figure 4.9, we present the KBD. It is consisted of four main components. The first component on the upper left is the Latest Detections table. This table is updated live with the latest detections received from the VA tool. It displays the most important information regarding the detections: detection id, scene, object, person detected, sensor that provided the detection and timestamp of the detection. A closer look of this component can be found on Figure 4.10.

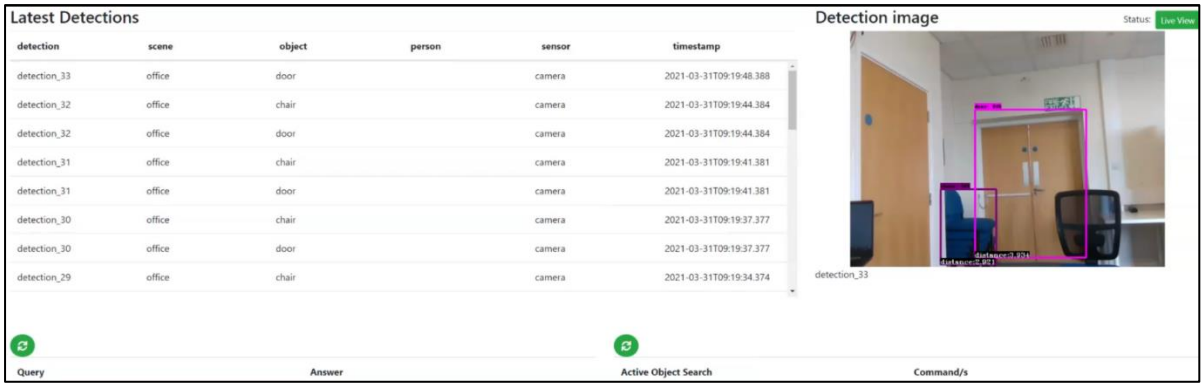


Figure 4.9: Knowledge Base Dashboard

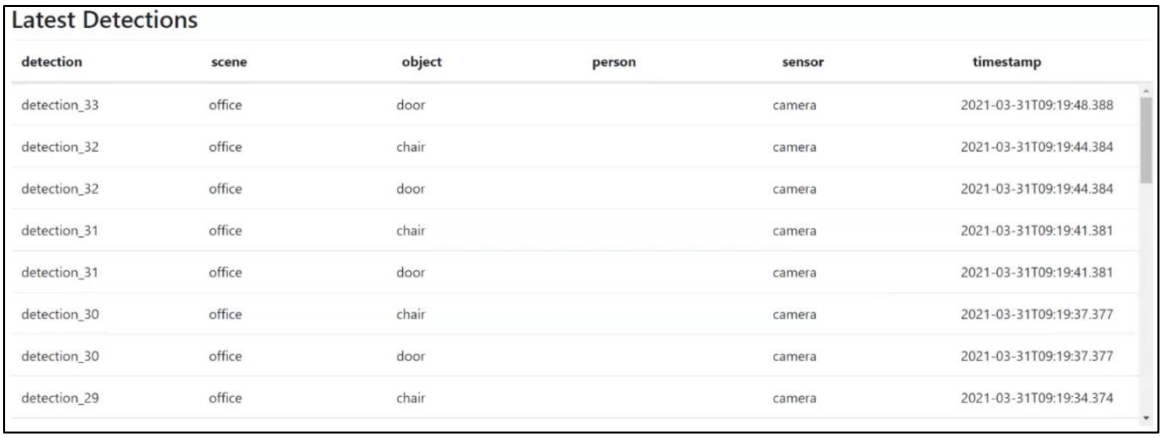


Figure 4.10: Latest Detections table of KDB

The second component of the KDB is the detection image on the upper right. There are displayed the detection pictures uploaded by the VA tool. There is also an option to always display the latest uploaded image, or to select and view a specific detection image. In Figure 4.11, we can see the detection image component.

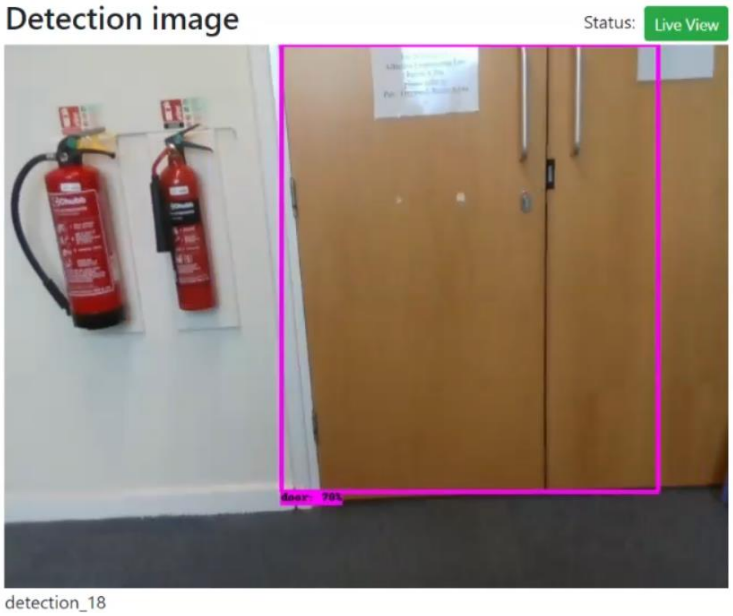


Figure 4.11: Detection Image Component of KDB

The third component of the KDB, placed on the lower left, is called the Queries and Answers component. There can be displayed in real time and in natural language, the queries that the user send to the system and the answer they receive. Figure 4.12 presents the Queries and Answers component.

Query	Answer
Where are the drawers?	The drawer was detected in the office , 0mins and 11 secs ago, on your left, 3.489824 m from you.

Figure 4.12: The Queries and Answers component of the KDB

Finally. the fourth component on the lower right, called the AOS component, displays in real time and in natural language the commands that the HIPI system sends to the user when they the AOS module is activated, i.e. when they search for an object. In Figure 4.13, we present the AOS component.

Active Object Search	Command/s
Searching for **drawers**	search
	move forward
	move forward

Figure 4.13: AOS component of the KDB

The KDB also proved to be a very useful tool for demonstration and debugging activities, as it visualizes the whole process of the HIPI system, like the queries sent, the answers received, the AOS project and the detections with all the information that is stored in the KDB.

4.3 Chapter Summary and Future Work

This chapter presented the third iteration (v3) of the SUITCEYES ontology that corresponds to the user needs, the requirements of the HIPI and the technical aspects of the involved components. First, we presented the additional conceptualisations of the ontology, which extend the representational capabilities of the second version. Next, we presented the changes that were applied to the different modules regarding the future specification that were set in D3.2. We also presented the KDB that was developed for the visualization of the whole process. Finally, we concluded with the new query additions of the HIPI system.

The following directions would further advance the ontology's functionality in the future:

- Extend the spatial entity relation capabilities of the ontology, i.e. say that the phone is on the table. This is implemented theoretically in the ontology, but methods should be added in cooperation with the VA tool so that we can deduce when an object is left/right from or on top/under another object.
- Add object ids so that the KBS can deduct which detections contain the same objects. This can be done in cooperation with the VA tool, so that we can keep only the most recent relevant information about an object. For example, if we detect a specific laptop in the bedroom, previous detections of the said laptop can be deleted from the KB, reducing the data volume overall. This decrease in data volume will also make the queries run faster on the KBS. For example, when the user asks "where is my laptop?", the query answer will contain less detections of laptops.

5 Conclusions

This deliverable presents the third and final version of the tools within WP3 for capturing, translating and semantically representing environmental cues. More specifically, we presented the third versions of the tools for visual analysis (chapter 2), perceptible haptic information (chapter 3) and semantic knowledge representation and reasoning (chapter 4). The aforementioned approaches comprise the final software module for the HIPI system and were tested using the final integrated prototype HIPI system presented in deliverable 4.3.

References

Chowdary, G. Jignesh, et al. "Face mask detection using transfer learning of inceptionv3." International Conference on Big Data Analytics. Springer, Cham, 2020.

Darányi, S., N. Olson, E. Lindell, N.-K. Persson, M. Riga, E. Kontopoulos and I. Kompatsiaris (2020). "Communicating Semantic Content for Persons with Deafblindness by Haptograms and Smart Textiles: Theoretical Approach and Methodology." International Journal on Advances in Intelligent Systems 13.

Durrant-Whyte, Hugh, and Tim Bailey. "Simultaneous localization and mapping: part I." IEEE robotics & automation magazine 13.2 (2006): 99-110.

Gayoso Martínez, Víctor, Fernando Hernández Álvarez, and Luis Hernández Encinas. "State of the art in similarity preserving hashing functions." (2014).

He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

Kawulok, Michal, Emre Celebi, and Bogdan Smolka, eds.Advances in face detection and facial image analysis. Springer, 2016.

Kuznetsova, Alina, et al. "The open images dataset v4." International Journal of Computer Vision (2020): 1-26.

Lahtinen, R. M. (2008). Haptices and Haptemes: A Case Study of Developmental Process in Social-haptic Communication of Acquired Deafblind People. PhD PhD, University of Helsinki.

Lahtinen, R. M. and R. Palmer (2010). "Environmental Description: For Visually and Dual Sensory Impaired People." A1 Management

Lennie, Peter, Joel Pokorny, and Vivianne C. Smith. "Luminance." JOSA A 10.6 (1993): 1283-1293.

Liu, Wei, et al. "Ssd: Single shot multibox detector."European conference on computer vision. Springer, Cham, 2016

Loey, Mohamed, et al. "A hybrid deep transfer learning model with machine learning methods for face mask detection in the era of the COVID-19 pandemic."Measurement167 (2021): 108288.

Nielsen, G., Ed. (2012). 103 Haptic Signals - a reference book, Danish Association of the Deafblind.

Redmon, Joseph, et al. "You only look once: Unified, real-time object detection."Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

Sandler, Mark, et al. "Mobilenetv2: Inverted residuals and linear bottlenecks." Proceedings of the IEEE conference on computer vision and pattern recognition. 2018.

Szegedy, Christian, et al. "Inception-v4, inception-resnet and the impact of residual connections on learning."Proceedings of the AAAI Conference on Artificial Intelligence. Vol. 31. No. 1. 2017.

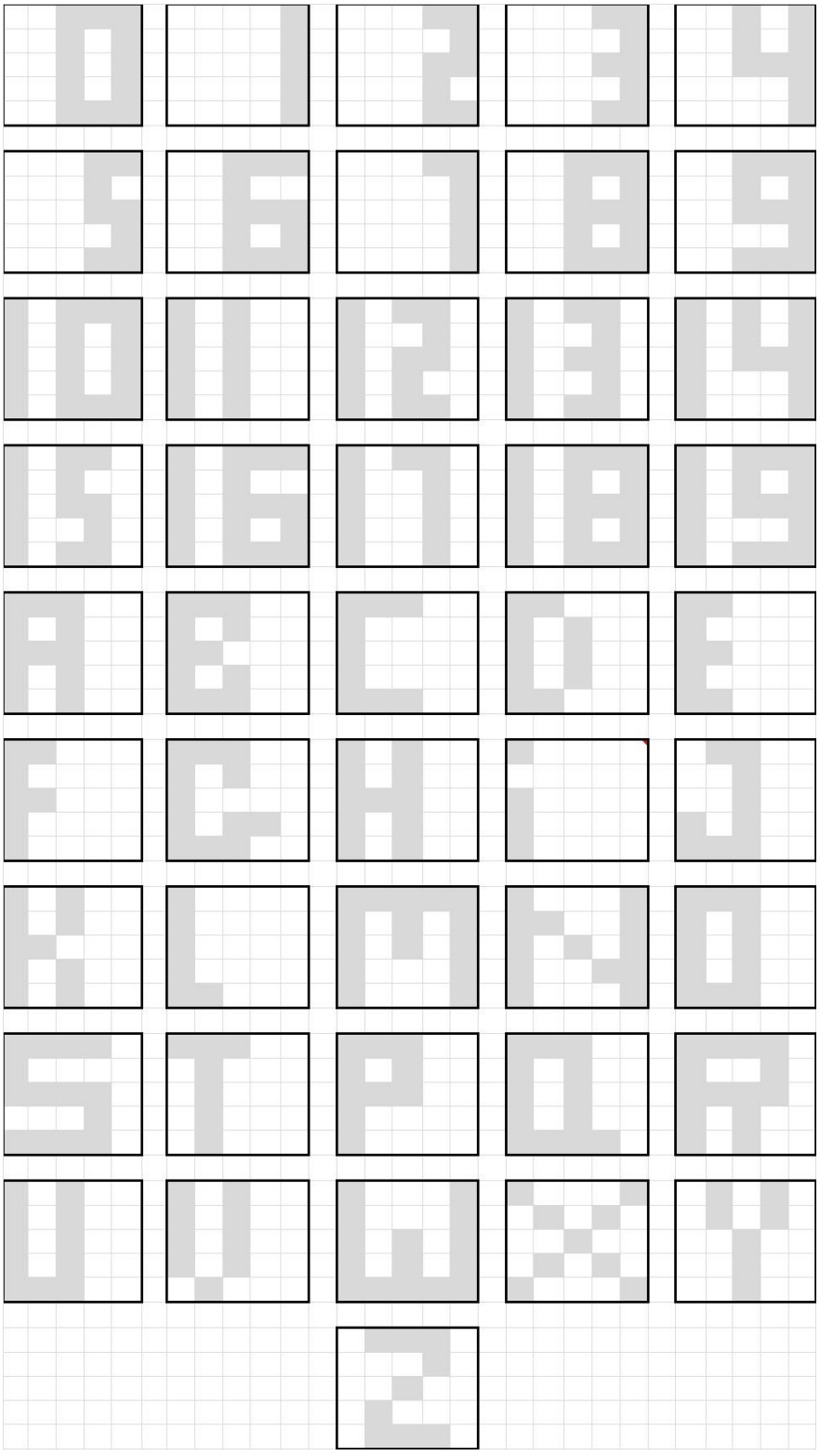
Szegedy, Christian, et al. "Rethinking the inception architecture for computer vision." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

Wang, Xin. "Laplacian operator-based edge detectors." IEEE transactions on pattern analysis and machine intelligence 29.5 (2007): 886-890.

Xiao, Jianxiong, et al. "Sun database: Large-scale scene recognition from abbey to zoo."2010 IEEE computer society conference on computer vision and pattern recognition. IEEE, 2010.

Zhou, Bolei, et al. "Learning deep features for scene recognition using places database." (2014).

Appendix A: The list of numbers between 0-19 and the Latin alphabet mapped to a 5x5 matrix



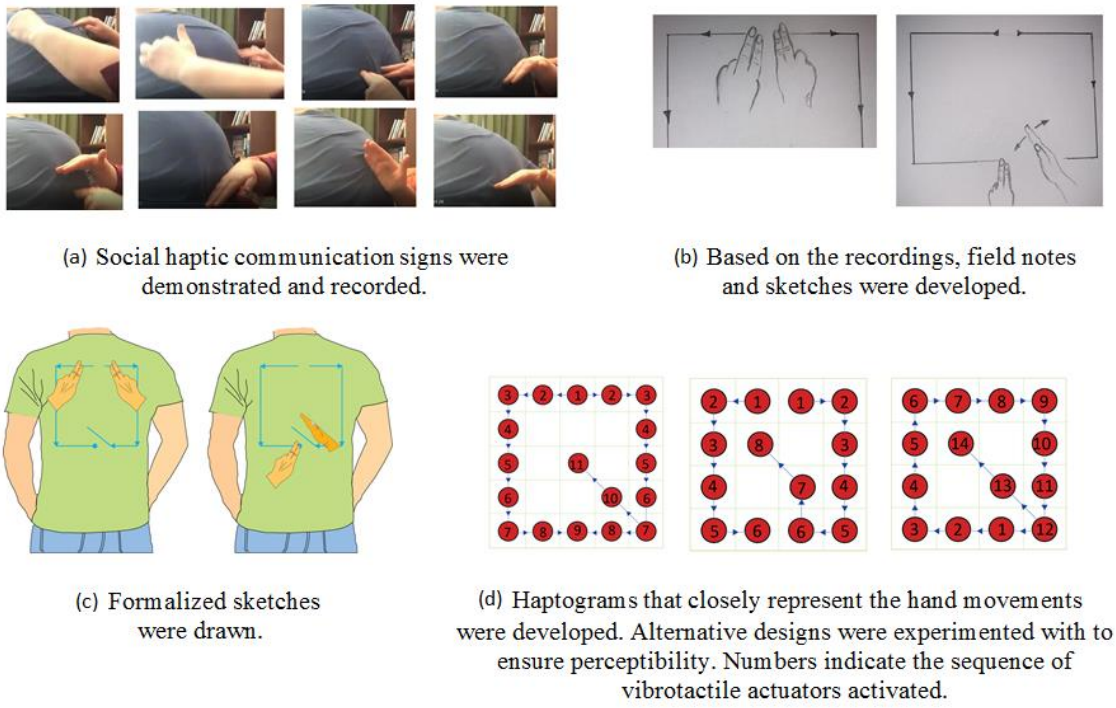
Appendix B: Ideation workshops and technological process of SHC adoption

Our work started by collecting various documented SHC signs (e.g., Nielsen 2012; Socialhaptiska signaler[11]; and other sources). These were compared with those individual words and more complex messages that are intended to be used by the SUITCEYES HIPI. Thereby we could single out concepts important for our work yet without documented SHC signs. The steps of the user-centred co-design process were as follows:

- We worked closely with the participants and reviewed SHC signs that were already defined by others. As some of these signs were based on visual sign language, they needed adjustments to adapt them more appropriately for people with dual sensory loss;
- We then went through a list of words of interest that lacked documented SHC signs. In these cases, the participants demonstrated the movements for those concepts. In most cases extended variations were shown, and background and related issues were discussed;
- These participatory design meetings and related demonstrations and conversations were recorded;
- SHC signs that were presented in the demonstrations were also reproduced by graphical sketches;
- The signs demonstrated, either directly based on the recordings, or by following the sketches, were converted to haptograms to closely resemble the movements made by hands in the demonstrations made by the participants. At that time, grids of 4x4 and 5x5 nodes mainly situated on the back and/or upper arm were being considered;
- While the participants could not test the actual vibrotactile patterns, the patterns of activation of vibrotactile motors were discussed with the participants for feedback and brainstorming.

Below we illustrate steps of the co-design process (Figure B.7a-c).

[11] https://socialhaptisk.nkcdb.se/



(a) Social haptic communication signs were demonstrated and recorded.

(b) Based on the recordings, field notes and sketches were developed.

(c) Formalized sketches were drawn.

(d) Haptograms that closely represent the hand movements were developed. Alternative designs were experimented with to ensure perceptibility. Numbers indicate the sequence of vibrotactile actuators activated.

Figure B.7a-c: Ideational steps in SHC-conform haptogram design

Required hardware for multi-panel haptogram display

The first test results with haptograms over a 3x3 grid were presented in D6.3. In D3.2, the example haptogram sets were generated over a 4x4 matrix of cells, and displayed on the back of the user. The combination of rows and columns in that way provided a two-dimensional (2D) arrangement of haptic actuators.

With ambitions to utilize other body parts as potential "screens" as well, the grid size for haptogram design needed to be extended beyond a 3x3 or 4x4 grid. Furthermore, the introduction of additional dimensions beyond 2D required exploration of variations in intensity of signals vs. the speed of movement which were not available in our earlier prototypes. For these reasons, new hardware designs were experimented with, and developed as follows:

- For the multi-panel experiments and access to an extended range of actuator motors, we had two new types of boards developed, one that supports 32 channels (Figure B.8) and one that supports 40 channels (Figure B.8);
- These boards can be used in standalone mode or in combinations;
- To create an even larger set of nodes for haptograms on different body parts, it is possible to daisy chain multiple boards (e.g., the 32-channel board x 2, or x 3 (etc.), achieving 64 or 96 channels, respectively). The number of connected boards is only limited by the power supply and the timing requirements. (Figure B.8);
- In the daisy chained case, it is fully possible to mix 32 and 40 channel boards together.

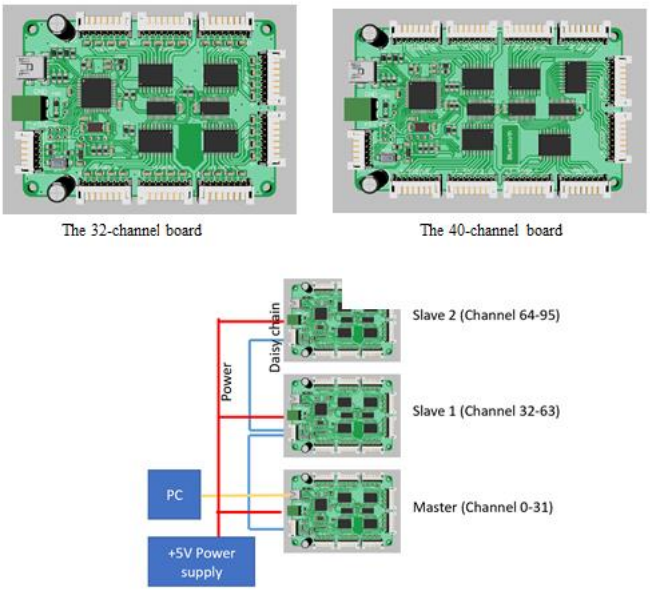


Figure B.8: Daisy chaining multiple boards