A Rule-Based Framework for Object Localization, Spatial and Situational Awareness with Natural Language Feedback for the Deafblind

Vasileios Kassiano, Anastasios S. Kesidis, Thanos G. Stavropoulos,
Spiros Nikolopoulos, and Ioannis Kompatsiaris
Information Technologies Institute (ITI)
Centre for Research & Technology Hellas (CERTH)
Thessaloniki, Greece
{vaskass, akesidis, athstavr, nikolopo, ikom}@iti.gr

***Abstract*—Deafblindness is a debilitating condition that hinders communication, awareness of the surroundings and ultimately independent living. Technological advancements in image analysis and haptics promise to support patients. Yet, machine-interpretable representations, interpretation and feedback to bridge the gap between perceiving the environment and communicating this to the user, are lacking. This paper presents a rule-based framework that supports object localization, spatial and situational awareness and natural language feedback for the deafblind. The framework utilizes ontologies as an interoperable data model to represent knowledge about the environment and rules to extract object localization, spatial and situational awareness as well as to form appropriate natural language responses. The rule set is expressed in the SPARQL Inferencing Notation (SPIN), which enables simplicity and flexibility in rule definition. A dashboard web application visualizes the streaming detections perceived from the environment and allows real-time queries and responses. A proof-of-concept scenario is realized with synthetic data from a realistic environment, showing use cases in detecting surroundings, locating an object and being aware of a situation (e.g., people wearing a facemask as a COVID-19 precautionary measure). In the future, the platform will support connection to a data exchange message bus to receive detections from image analysis and communicate actions towards haptic hardware to enable testing by patients.**

Keywords-ontology; rules; natural language; SPIN; situational awareness; spatial awareness; localization; deafblindness.

I. INTRODUCTION

Communication with and between users with deafblindness is constrained by the debilitating nature of this disability, ranging from congenital to acquired deafblindness, including worsening eyesight, hearing or both over time, and, ultimately, symptoms of ageing as well. Technology promises to facilitate such communication problems with advancements in image analysis [2] [3] and haptic technology [4] [5]. Yet, applications that integrate a machine-interpretable understanding of the users' surroundings and give comprehensive feedback to them are limited.

This paper presents a framework that provides object localization, spatial and situational awareness for the deafblind. The framework is based on ontologies to interoperable represent context (i.e., the environment, surroundings and situations) and rules to interpret, transform and formulate appropriate responses to user queries. The SPIN rule-engine provides flexible and extendable rule sets for finding an object, realizing surroundings in relevance to the user and becoming aware of the situation (e.g., people wearing facemasks as a precautionary measure against COVID-19). While the previous study in [1] presented the data model (ontology) and spatial awareness rules, this study extends the framework with object localization and situational awareness as well as a dashboard for visualizing the system's streaming detections, performing user-defined queries in real-time and getting responses. Proof-of-concept use cases are realized using this dashboard, demonstrating suitable responses inspired by the SUITCEYES project [6], to improve the Quality of Life of the deafblind using interactive technology. In SUITCEYES, the platform will additionally support connection to a message bus to receive real detections from cameras and image analytics as well as haptics, in order to provide feedback.

The rest of the paper is structured as follows: In Section II, related work regarding natural language (NL) and spatial awareness is presented. In Section III, the architecture of our system is introduced, including data collection and storing processes, as well as inference methods incorporated. The methods used for Spatial Information to Natural Language, for Facemask Feedback and for Object Localization can also be found in this section. In Section IV, our experiments and results produced by using synthetic data are shown, inspired by the data received from the SUITCEYES platform. Finally, in Section V conclusion and motivation for future work can be found.

II. RELATED WORK

An increasing number of Internet of Things (IoT) applications are used for healthcare purposes due to their ability to support living with various ailments, such as ageing and dementia [7] [8] [9]. These types of applications acquire knowledge from multiple sources and from continuous and heterogeneous data flows [10] [11]. Semantic technologies provide comprehensive tools and methods for representing knowledge and using inference to produce new knowledge from data. IoT environments are increasingly found in home healthcare technologies in actions that create better living conditions for the elderly, through the use of IoT

technologies, such as Active and Healthy Ageing (AHA) and Home Ambient Assisted Living (AAL).

Furthermore, in the case of deafblind people, the simple and accurate representation of their surrounding environment is one of the most basic needs for their quality of life. This can be achieved by providing the nature of their surroundings in natural language. Some of the most relevant work for language processing with ontologies are [12] [13].

KnowSense [7] is an activity monitoring system for elder people with dementia, deployed in controlled and diffuse environments. Semantic Web technologies, such as OWL 2, are widely used in KnowSense to display sensor and specific application observations, as well as to implement solutions for identifying activities and problems in everyday life activities (Instrumental activities of daily living, IADLs) with the aim of clinical evaluation of various stages of dementia. Description Logic (DL) reasoning for activity detection and SPARQL questions are used to extract clinical problems.

ACTIVAGE [8] is a large-scale pilot project, which aims to develop Smart Living solutions that strengthen active and healthy aging. The ACTIVAGE IoT Ecosystem Suite (AIOTES) project consists of a set of techniques, tools and methodologies (rule-based reasoning, interoperable ontologies, etc.) that increases semantic interoperability at different levels between heterogeneous IoT platforms. The approach uses different mechanisms of reasoning that can improve the understanding of patients' heterogeneous data and help generate new knowledge by providing services to end users.

Dem@Care [9] is a system based on heterogeneous sensors that provides support for independent living for elder people with dementia or similar health problems. This approach incorporates a heterogeneous set of detection methods and technologies, including video, audio, in addition to normal, environmental, and other measurements. Semantic technologies (e.g., rules-based reasoning) are used to process and analyze sensor data according to user requirements. This leads to feedback and decision support, which is communicated to end users through appropriately designed user interfaces. The support includes various clinical scenarios, both short (trials in hospital settings) and long term (daily living at work), for independent living.

Furthermore, semantic mechanisms are created and applied that process patients' original data and use it to generate new knowledge by offering new services for the benefit of end users such as caregivers, health professionals and patients.

In [14], a system for healthcare in Smart Home environments is developed which considers social relationship-based contexts to provide a fully personalized healthcare service.

An ontology-based sensor selection for real-world wearable activity recognition is presented in [15], in which the use of ontologies is proposed to thoroughly describe the wearable sensors available for the activity recognition process. This enables the semantic selection of sensors to support a continuity of recognition.

In [16], an extended version of a linguistic ontology is presented that works particularly with space. Language regarding space, spatial relationships and actions in space is covered and an ontological structure that relates such expressions with ontology classes is developed. Finally, examples of the ontology's results based on natural language examples are presented.

In [17], a project in which ontologies are part of the reasoning process used for information management and for the presentation of information is presented. Both accessing and presenting information are mediated via natural language and the ontologies are coupled with the lexicon used in the natural language component.

An approach to transform natural language sentences into SPARQL is proposed in [18], with the use of background knowledge from ontologies and lexicons. The results of this approach show that the diagnosis process and the data search for a broad range of users is improved.

In [19], an evaluation is made on how efficient the SPARQL query language is and the SPARQL Inferencing Notation (SPIN) when utilized to identify data quality problems in Semantic Web data automatically, and within the Semantic Web technology stack

In the case of deafblind patients, the simple and accurate representation of their environment is one of the most important needs. In [15], [16] and [17] even though forms of natural language processing through ontologies are proposed, they do not involve healthcare or wearable sensors. For the healthcare related work [4] [5] [6] [8], [13] and [14], the natural language presentation of information component is absent.

Meanwhile, as the COVID-19 pandemic crises developed worldwide, in [20], the need for precautionary measures to consider and adapt to people with disabilities is emphasized, with tailored recommendations. That includes people with deafblindness, who find it harder to maintain social distancing and assess potential threats in the surroundings.

The framework proposed in this paper, combines existing approaches to represent data and execute rules using semantic technologies for healthcare, but extends them with: 1) a representation of the surrounding environment for deafblind patients; 2) a set of rules to extract knowledge for object localization, situational and spatial awareness and provide responses in natural language; 3) a dashboard to visualize inputs and outputs and integrate with image and haptic technology to enable pilots. Specifically, existing data models and rule sets are greatly extended to support the more varied toolbox of rules expressed in SPARQL and SPIN notation, which are considered to be the state-of-the-art notation when it comes to semantic data management. Particularly, inspired from the pandemic, situational awareness includes rules to respond to whether people are wearing a mask or not.

III. THE PROPOSED FRAMEWORK

A. Requirements and System Specification

The requirements for the framework were derived from the SUITCEYES project, aiming to improve the Quality of

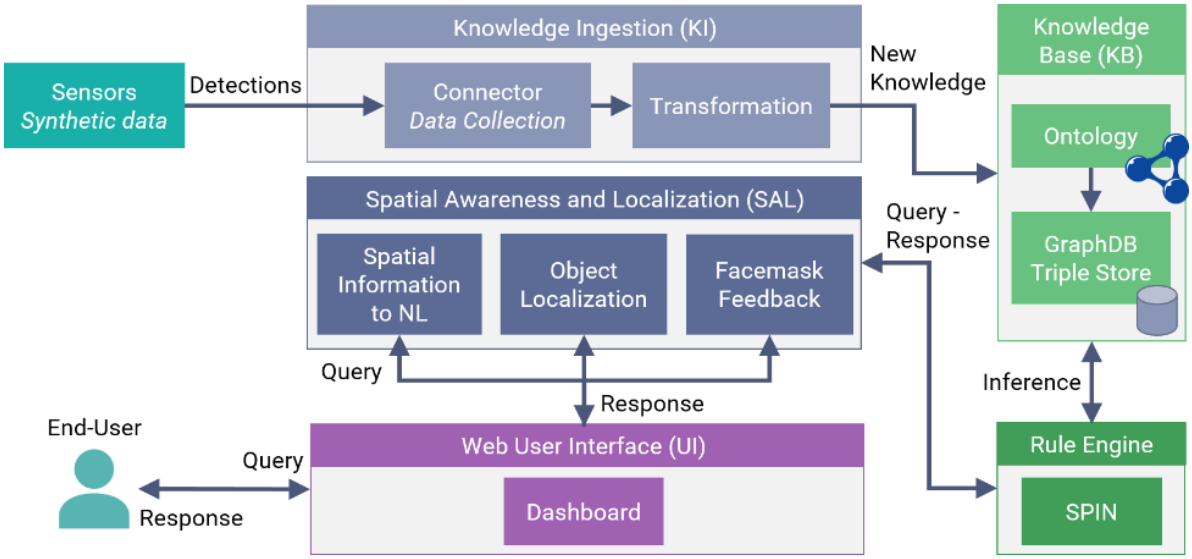

Figure 1: System Architecture

Life of people with deafblindness. As such, the project provided both functional and non-functional requirements such as the ability to exchange data coming from sensors, cameras and image analytics and respond back to haptics in a universal JSON format in real-time.

The "detections" from input devices contain information such as which object/person was detected, when the detection took place, what sensor made the detection and spatial information like where in the user's field of view is the object/person, e.g., left, right, in front of, etc. Finally, the measured distance is reported, if the sensor has this capability. Additionally, the system should receive and respond to queries, like "Where is my book?" and respond in natural language, comprehensively, in a manner that the user would understand (through translation to haptic devices).

As such, there is a need for a "brain" of the system, a toolbox of methods containing the appropriate rules for extracting knowledge from the incoming data, a knowledge/data base for storage and the methods for handling the overall procedure and the input/output.

The input that our system receives come from the end-user who sends queries like "Where is my PC?" to the dashboard. After examining the knowledge contained in the ontology, they receive the appropriate response, either in a natural language message or comprehensive directions and pointers, which can later be translated in haptic devices.

B. Architecture

According to the aforementioned requirements, the proposed framework was designed using a modular, extensible and service-oriented architecture shown on Figure 1. It consists of the following components:

- The sensors/synthetic data which provide information in the form of detections to be stored in the ontology.
- The Knowledge Ingestion (KI) component of our framework is responsible for receiving the incoming data from the sensors/synthetic data source and make the appropriate transformations (e.g., differential population) so that the data are ready to be uploaded to the ontology stored in GraphDB.
- The component that stores all data and rules, and executes the queries is the Knowledge Base (KB). In our KB the ontology with all the new knowledge is stored, in the GraphDB Triple Store database.
- The Spatial Awareness and Localization (SAL) component contains all methods that were developed to extract information from the ontology and to answer the user queries. These methods are: a) the Spatial Information to NL, which answer simple queries like: "Where is my laptop?", b) Object Localization (OL), which answer queries like "Locate my book" and c) Facemask Feedback, which answer queries like: "Is the person next to me wearing a facemask?".
- A Dashboard implemented as a Web User Interface (UI) application, which is responsible for visualizing the latest streaming detections as they arrive in the KB, and enables users to input queries and receive answers. The dashboard is intended for researchers and integrators to visualize results before integrating with their technology for data input (detections) or output (haptic messages).
- Finally, we use the SPIN rule engine to execute rules automatically, each time new data gets uploaded to our KB.

C. The Ontology – Data Model

As a primary data model, the framework utilizes the SUITCEYES ontology [4], presented in [1] and available online [24]. The ontology integrates heterogeneous, multimodal input from different sensors in a formal and semantically enriched basis, thus, user context-related

information, that can provide enhanced situational awareness and augment the user's navigation and communication capabilities. The ontology already includes representation of objects and activities from the Dem@Care ontology [9], sensors and sensor observations from the SOSA/SSN ontologies [21], persons and social associations from the Friend-Of-A-Friend (FOAF) specification [22]. In addition, core topological concepts of a building from SEAS (Smart Energy Aware Systems) [23] was integrated, which is a schema for describing the core topological concepts of a building, such as buildings, building spaces and rooms.

In this work, concepts and relations are extended to support new functionality. Some of the basic classes of the SUITCEYES ontology represent objects, spaces and people that can be detected as raw data form sensors, such as cameras, or processed data, through a visual analysis component. For example, some of the objects that can be found in the ontology are: computer, laptop, alarm clock, mug, table, chair and other everyday objects, while some of the spaces include: bedroom, bathroom, living room and other spaces that can usually be found in a home environment. We extended these concepts further from the ontology presented in [1], by adding more classes to the ontology that refer to objects, spaces and rooms that could be useful to any deafblind user. We also added a Boolean data property named "facemask" to the Person entity.

D. Knowledge Ingestion (KI) Component

The data containing the detections from sensors or from synthetic data arrive in our KI component. Each message that arrives represents an image frame that contains information about the scene, people and objects detected. An example with the kind of data that the KI component receives can be seen below:


```
{
 "image": {
            "target": [
                      {
                       "height": 176,
                       "top": 96,
                       "type": "person_unknown",
                       "width": 115,
                       "left": 407,
                       "confidence": 0.725,
                       "distance": 1322.01
                        },
                       {
                        "height": 134,
                        "top": 329,
                        "type": "computer",
                        "width": 195,
                        "left": 123,
                        "confidence": 0.597,
                        "distance": 1822.01
                        }
                                ],
  "timestamp": "2020-02-19T08:18:27.649",
  "scene_score": 1.0,
  "width": 640,
  "scene_type": "office",
  "height": 480,
  "name": "19_02_20_08_18_27_649204"
 }
```


In this example, we can see that an unknown person and a computer are detected in an office. We can also acquire information about the time of this detection, the distance from the source and various information about the position of the bounding boxes of the detected entities (height, top, width, and left) in pixels.

The KI component receives these messages and using a simple function prepares them to be stored in our ontology by creating a simple INSERT SPARQL query. This query contains all the meaningful information about the entities and scenes detected.

Before executing an INSERT query, another step of data transformation is performed. We developed an efficient approach to store the incoming data to the ontology. We call this approach "differential population" of the ontology, which means that instead of populating the ontology with every detection data received, a method is applied that checks if the incoming detection data is already included in the ontology. If true, then only the timestamp of the existing detection instance is changed, otherwise we add the new detection to the ontology. This algorithm of the differential population procedure is presented in Figure 2.

By using this technique, we limit the volume of data that are inserted in the ontology by only applying a simple check each time we receive a detection. This increases efficiency, considering that in a home environment the same objects could be detected in the same place multiple times (e.g., a TV almost never changes place in a home) and that many sensor systems continuously send data via their sensors.

E. Spatial Awareness and Localization (SAL) component

The SAL component is responsible for performing all the processes using semantic technologies to answer the user queries. It contains 3 main methods that answer different type of queries. These methods are the following:

Algorithm: **Differential Population**
Input: D (Detection type object)

for each D_i stored in ontology ***do***:
{
If D.detects_object == D_i.detects object and
D.spatialContext == D_i.spatialContext then
D_i.timestap = D.timestamp
}
End

Figure 2: Differential Population Algorithm

```
CONSTRUCT{
    ?detection sot:producesOutput ?output.
    ?output sot:refersToDetection ?detection.
    ?output a sot:Output.
    ?output sot:hasTextualDescription ?description .
}
WHERE{
    BIND (sospin:Function_SelectLatesObjectDetection() AS ?detection) .
    ?detection a sot:Detection .
    ?detection sot:detectsObject ?object .
    ?object a ?object_type.
    FILTER (?object_type = sot:Laptop).
    ?object rdfs:label ?objectName .
    ?object sot:hasSpatialContect ?spatialContext.
    ?spatialContext a ?spatialContextType .
    ?spatialContextType rdfs:SubClassOF sot:Spatialcontext .
    FILTER ((?spatialContextType - sot:RightSpatialCOntext) || (?spatialContextType =
sot:LeftSPatialContext)).
    BIND (sospin:Function_LeftRightContext(?spatialContext) AS ?leftright_annotation).
    FILTER ((?spatialContextType - sot:RightSpatialCOntext) || (?spatialContextType =
sot:LeftSPatialContext)).
    BIND (sospin:Function_LeftRightContext(?spatialContext) AS ?leftright_annotation).
    BIND (sospin:Function_CloseFarContext(?spatialContext) AS ?closefar_annotation).
    FILTER ((?spatialContextType - sot:CloseSpatialCOntext) || (?spatialContextType =
sot:farSpatialContext)).
    BIND (sospin:Function_CloseFarContext(?spatialContext) AS ?closefar_annotation).
    BIND (BNODE() AS ?output) .
    BIND (CONCAT(?objectName, "is on your", ?leftright_annotation, " side.", ?closefar_annotation, "to you.
") AS ?description).
}
```

Figure 3: SPARQL/SPIN rule for query #1

1) Spatial Information to Natural Language:

In Table 1, we present a list of indicative queries that are used in the ontology to provide the user with a natural language output regarding spatial information of their surroundings. In these queries, variables are used to cover a broad number of objects and spatial contexts. This kind of inference is achieved by using a set of ontological rules, written in SPARQL/SPIN notation, that run on top of the ontology, whenever a specific query is triggered by the user. Within the context of this scenario, a list of indicative queries were created that can dynamically change on specific aspects, i.e., to cover different entities of interest, different spatial relations (with respect to the distance of position left/right), etc.

Table 1: List of rules and their output

#	Query	Nat. Lang. Output
1	Where is my <object >?	Your <object > is on your < right/left spatial context> side, <close/far spatial context> to/from you.
2	How many <objects> are on my <right/left spatial context> side?	<# counted> objects, an <object1> and an <object 2> are on your < right/left spatial context > side.
3	Which <objects> are <spatial context> to/from me?	An <object1>, <object2> and object3> are located <spatial context> to/from you.

The variables in the above queries are given a specific value, depending on what the user wants to ask. For example, the first query can be transformed in natural language to: “Where is my laptop” and its output can be: “Your laptop is on your right side, close to you”. The implementation of this query is presented as an ontological rule written in SPARQL/SPIN syntax in Figure 3, using synthetic data, which we created, that include various objects and spatial contexts. Most of our ontological rules use SPARQL CONSTRUCT and DELETE/INSERT commands, in order to create new triples in the ontology and thus enrich the knowledge stored in the schema.

In Figure 4, the implementation of the #3 query is presented, which in natural language translates to “Which objects are close to me?” which using our synthetic data produces the output: “A laptop, a TV and a chair are located close to you”. For this query we have skipped the construct rule, which is the same as the #1 query.

2) Facemask Feedback

During the COVID-19 pandemic, it would be extremely useful for a deafblind user to ask if the person/people next to them wear a facemask. For this reason, we developed the ontology and implemented a SPIN query to perform this task.

From the ontology side, we added a Boolean data property to the Person entity called “facemask” which has value true if the person is detected wearing a facemask and

```
WHERE{
    BIND (sospin:Function_SelectLatesObjectDetection() AS ?detection) .
    ?detection a sot:Detection .
    ?detection sot:detectsObject ?object .
    ?detection sot:detectsObject ?object2 .
    ?detection sot:detectsObject ?object3 .
    ?object a ?object_type.
    ?object2 a ?object_type.
    ?object3 a ?object_type.
    ?object rdfs:label ?objectName .
    ?object2 rdfs:label ?objectName2 .
    ?object3 rdfs:label ?objectName3 .
    ?object sot:hasSpatialContect ?spatialContext.
    ?object2 sot:hasSpatialContect ?spatialContext2.
    ?object3 sot:hasSpatialContect ?spatialContext3.
    ?spatialContext a ?spatialContextType .
    ?spatialContext2 a ?spatialContextType2 .
    ?spatialContext3 a ?spatialContextType3 .
    ?spatialContextType rdfs:SubClassOF sot:Spatialcontext .
    ?spatialContextType2 rdfs:SubClassOF sot:Spatialcontext .
    ?spatialContextType3 rdfs:SubClassOF sot:Spatialcontext .
    FILTER ((?spatialContextType - sot:CloseSpatialCOntext) &&
        (?spatialContextType2 = sot:CloseSPatialContext)  &&
        (?spatialContextType3 = sot:CloseSPatialContext)).
    BIND (sospin:Function_CloseFarContext(?spatialContext) AS ?closefar_annotation).
    BIND (BNODE() AS ?output) .
    BIND (CONCAT(?objectName, ", ", ?objectName2, "and a ", ?objectName3, ", are ", ?closefar_annotation, "to you. ")
        AS ?description).
}
```

Figure 4: SPARQL/SPIN rule for query #3

false if not. In Figure 5, we present the facemask data property of the Person class.

If the query returns a person wearing a facemask, we offer this information in the form of an output file with the field "facemask: yes/no". The query fired to return the information of whether the person next to the user wears a facemask is presented in Figure 6.

As its name suggests, the variable "?facemask" will contain the Boolean value of whether the person nearest the user wears a facemask or not. We order the results in descending order regarding to the timestamp, as we want to take into account only the most recent detection and in ascending order regarding the distance, as the query refers to the person nearest to the user. The limit value on the last line of the query shown in Figure 6 can be changed to include more people wearing a facemask around the user.

3) *Spatial Awareness and Localization (SAL)*

For the purpose of locating an object, it would be useful if our ontology could answer queries of a more continuous nature. For example, in the previous iteration, if the user asked "Where is my laptop", they would get a single answer containing the laptop's location if it was found, or the information that the laptop was not detected. Then the user would have to repeat the query again and again, until the object is finally detected, i.e., when the object comes in the field of view of the camera that is attached on the user.

For this reason, we implemented SPIN/SPARQL queries that support SAL. These queries are continuously executed and the process ends only when the object that the user is searching for is found and it is very close to them, or when

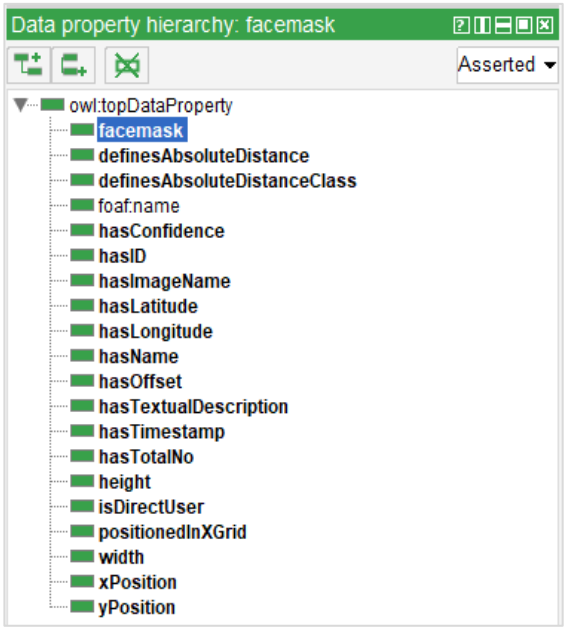

Figure 5: Facemask Data Property

```
SELECT ?facemask
WHERE {
    ?detection a sot:Detection .
    ?detection sot:hasTimestamp ?timestamp .
    ?detection sot:providedBy ?sensor .
    ?detection sot:detectsSemanticSpace ?scene .
    ?detection sot:detectsPerson ?person .
    ?scene rdf:type sot:SemanticSpace .
    ?person sot:hasSpatialContext ?sc.
    ?person sot:facemask ?facemask.
    ?sc sot:definesAbsoluteDistance ?distance.
    }
    ORDER BY desc(?timestamp) asc(?distance)
    LIMIT 1
```

Figure 7: Facemask Feedback Query

the user manually stops the search. The process of SAL is described below.

When the user tries to find a specific object, i.e., when a "Locate my <object>" query is received, the system performs the SAL procedure. During this procedure, the SPIN rules that try to look for this object in the database are fired consequently, every second that passes. The information that is received from the KB is then stored in an output file to be shared with the appropriate component of the system, which gives directions to the user. The directions given are based on an offset value that our query returns. This value that refers to pixels, ranges between -200 to 200 in a 400x400 picture that a sensor captures. For example, if the offset value is 100, it means that the object is on the right side of the user and the directions should guide the user to the right. The process stops when the distance variable that our query returns is less than 0.3 meters which is the value we assume that the user can reach the desired object with their hands. This threshold could be changed throughout the development of the system to correspond to specific objects, by experimenting with different setups and users.

Algorithm: **Spatial Awareness and Localization**
Input: User query in JSON format
Output: JSON file with object information
uq = user query
distance = 1;

```
While (distance > 0.3 || object != found)
{
        query response = perform query(uq);
        if (query_response has object detected)
                object = found;
        distance = query_response.distance;
        output = prepare_response_Json(query
        response);
}
End
```

Functions:
perform_query: sends the query to be executed in our GraphDB instance
prepare_response_JSON: takes the dataset that the query returns and creates the response JSON file

Figure 6: SAL Algorithm

```
SELECT ?detection ?timestamp ?slabel ?elabel ?leftright ?dis
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.
    ?entity a sot:Object.
    ?scene rdf:type sot:SemanticSpace .
    ?entity sot:hasXGridSpace ?x.
    ?x sot:positionedInXGrid ?leftright.
    ?entity sot:hasSpatialContext ?sc.
    ?sc sot:definesAbsoluteDistance ?distance.
    }
ORDER BY desc(?timestamp)
LIMIT 1
```

Figure 8: SAL query for Object Entity search

In SAL, the information that is sent to the system contains the following four kinds of data:

- Timestamp of the detection that the object was last detected.
- If the object was found in the latest detection, the detection field contains the field "detection" which holds a Boolean variable with value "True" in the response file. If the object is not found in the last detection, the value is "False".
- Object's orientation from the user in the form of pixel offset in the detection image. For example, if the offset is negative 100, the object is located on the left of the side of the field of view of the user.
- Object's distance from the user in meters.

The form of the response output can be seen below:

```
{
  "timestamp": "2021-05-17T08:42:00",
  "detection": True,

  "offset": -100,
  "distance": 2.0
}
```

The algorithm that performs the SAL procedure is presented in Figure 7.

The rules that are fired during SAL are presented below. Once again, we use SPIN/SPARQL notation to implement these rules and the result set is transformed into an output file with the appropriate information with the use of a simple JAVA function that we created. In Figure 8, we present the query fired when the user searches for an entity.

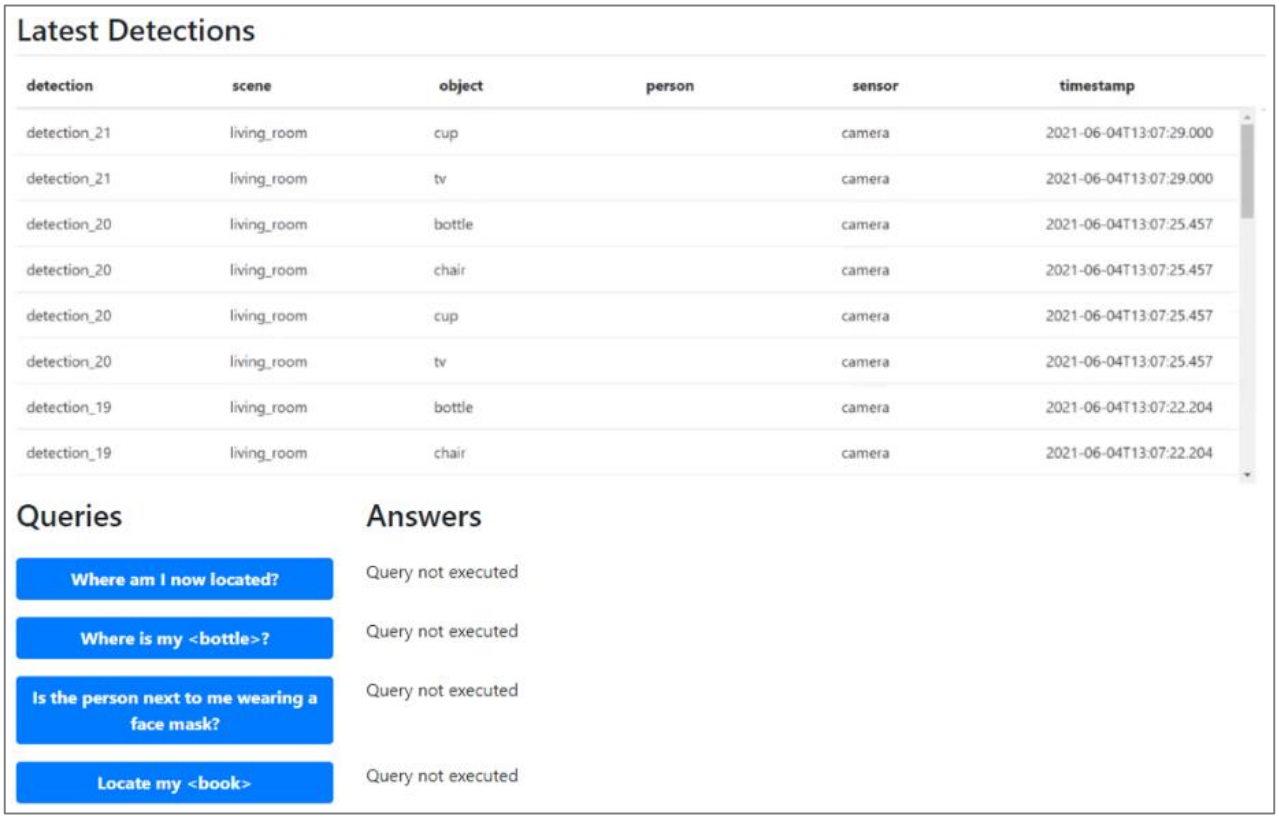

Figure 9: Dashboard showing latest detections as they stream into the platform and allowing queries and responses

The variable "?timestamp" contains the timestamp information that is needed for the output file. The "?elabel" variable contains the string value that describes the object entity. For example, the label of a Cup object of the ontology is "cup". This field helps us group some objects in the same category, e.g., when the user searches for a mug we assume that a cup would also suffice. This is why we have put the label "cup" in the Mug objects of the ontology as well, so when a query with the rdfs:label "mug" is fired, objects of both Mug and Cup classes would be returned.

The "?leftright" variable contains the value of the sot:positionedInXGrid data property. This value is measured as pixels in an integer, where 0 is the center of the picture, negative values mean left and positive right from the center. The "?leftright" variable gives us the offset value of the response.

The "?distance" variable, as the name suggests, contains the distance of the object to the user. The distance is measured in millimeters, which are then translated to meters, using a JAVA function.

Finally, we use the "ORDER BY desc (?timestamp)", to order them by descending timestamp, and the LIMIT 1 commands to get only the most recent detection of the searched entity. In the event of the entity not found, by using once again a JAVA function, we return the Boolean value "False" to be contained in the output, otherwise the value is "True".

The notion behind this procedure is that when an object is detected and it is very close to the user (e.g., <0.3m), the user can reach and find the object by touch. This is why our loop also checks the distance value of the detection, to decide whether to continue or not.

F. Web User Interface (UI) – Dashboard

For visualization purposes, along with making easier the process of testing and experimenting with our system, we developed a dashboard, which contains the most recent information about our KB and visualizes the queries that are sent by the user and the answers that they receive from our system. The dashboard is presented in Figure 9.

The dashboard contains two main components:

- The Latest Detections table, which visualizes the most important information about the latest detections stored in the ontology such as detection id, scene, object and person detected, sensor that made the detection and timestamp of the detection. The detections table is updated live with the latest detections received from the sensors or, in our case, the synthetic data.
- The Query and Answer component, which in this iteration of the dashboard contains 4 of the main queries that a user can send to our system. These queries are: 1) Where am I now located? 2) Where

is my <bottle>? 3) Is the person next to me wearing a facemask? and 4) Locate my <book>.

Note that the entities in <> are parameters, i.e., can be changed to include other entities.

IV. PROOF OF CONCEPT USE CASES

Utilizing the system and the ability to visualize data and input queries on the dashboard, we performed a set of use case scenarios to validate its correctness. The scenarios are inspired from the SUITCEYES project. Based on its requirements, we constructed a set of synthetic data in the form of input messages streaming in the platform and with a realistic sequence of objects and people entering, moving and leaving the scene, as well as scenes changing, emulating a person moving in a room with a wearable camera. The use case scenarios are the following:

i) A developer/integrator wants to test the KB platform for receiving detection data from their component connected to the message bus

ii) A developer/integrator wishes to receive natural language output from user queries to integrate into their haptic system

iii) A deafblind user wishes to know whether the person or people around them wears a facemask

iv) A deafblind user wishes to find an object and navigate to it

Each of the following subsections presents one scenario performed through the system.

A. Incoming Detection Data stored Visualization

For the purpose of visualizing the process of receiving data, we developed a simple dashboard interface which shows in real time the data stored in the ontology.

As we can see in the snapshot presented in Figure 9 from the synthetic data we received in the last three detections the scene is a living room, the objects detected are a cup, a tv, a bottle and a chair, no people were detected and the sensor was a camera. Note that this snapshot will be used as our basis for the state of the ontology during our queries experimentation.

B. Spatial Information to Natural Language Experiments and Results

We present the output of the SPARQL/SPIN rule shown in Figure 3 by using the Protégé software [25]. In Figure 10, we present the object and description returned by the query. The description will be used as the output to the user.

In Figure 11 and Figure 12, we present the objects returned from the query “Which objects are close to me” with their spatial context, and the output that the user will get as textualDescription.

We also use the queries and answer component of the dashboard we developed to receive queries and provide answers in real time. In Figure 13, we can see the queries sent and the answers received from our framework, regarding the state of the ontology described in Figure 9.

By receiving the expected results regarding our synthetic data, we validated the correctness of our queries.

We also used the same synthetic data to test the differential population procedure. In the example below, we have stored in our KB two similar detections: The first detection contains a laptop in the living room a detection with the same information, arrives from the SUITCEYES system with a more recent timestamp. This information, if stored twice in the ontology would be duplicated with the exception of the different times. In Figure 14, we present what the outcome would be without using our differential population method, i.e., the storing of 2 similar detections.

In Figure 15, the outcome of the same incoming detection is presented, but with using the differential population procedure. Here, only the first detection is stored, with its timestamp field changed to match that of the second, incoming detection.

This absence of almost identical detections could save a lot of space in real applications that continuously receive and store data.

C. Facemask Feedback experiments and results

In this section we present the process that is executed when a user asks if the person next to them wears a facemask. For this reason, we created synthetic data that contain people

object	description
Laptop	"Your Laptop is on your right side, close to you."^^<http://www.w3.org/2001/XMLSchema#

Figure 10: Query output of "Where is my laptop"

obj	spatialContext	obj2	spatialContext2	obj3	spatialContext3
Laptop	Close	Chair	Close	TV	Close

Figure 11: Objects and their Spatial Context from query#3

queryOutput	textualDescription
QueryOutput	"A Laptop, a Chair and a TV are close to you"^^<http://www.w3.org/2001/XMLSchema#string>

Figure 12: Query output of "Which objects are close to me"

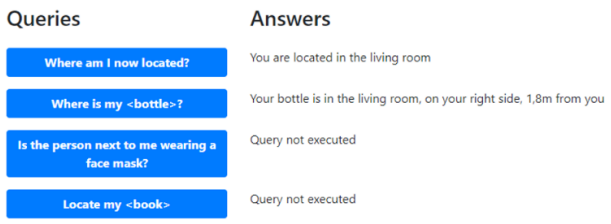

Figure 13: Query and Answer component of the dashboard

detection	obj_detected	where	timestamp
Detection_1	Laptop	Living_Room	"2020-08-25T13:18:30Z"^^<http://www.w3.org/2001/XMLSchema#dateTimeStamp>
Detection_2	Laptop	Living_Room	"2020-08-25T13:20:00Z"^^<http://www.w3.org/2001/XMLSchema#dateTimeStamp>

Figure 14: Detections saved without the differential population procedure

detection	obj_detected	where	timestamp
Detection_1	Laptop	Living_Room	"2020-08-25T13:20:00Z"^^<http://www.w3.org/2001/XMLSchema#dateTimeStamp>

Figure 15: Detections saved with the differential population procedure

```
sd:detection_5
  rdf:type sot:Detection ;
  sot:detectsPerson soa:john ;
  sot:detectsPerson soa:unknown_person_1 ;
  sot:detectsSemanticSpace soa:kitchen_1 ;
  sot:hasConfidence "0.77"^^xsd:float ;
  sot:hasTimestamp "2021-05-20T09:32:00.233"^^xsd:dateTime
  sot:providedBy soa:camera_1 ;
.
sd:john
  rdf:type sot:KnownPerson ;
  rdf:type sot:Person ;
  rdf:type foaf:Person ;
  sot:locatedInSemanticSpace soa:kitchen_1 ;
  sot:definesAbsoluteDistance 1500.00;
  sot:facemask "True";
  rdfs:label "John" ;
.
sd:unknown_person_1
  rdf:type sot:Person ;
  rdf:type sot:UnknownPerson ;
  rdf:type foaf:Person ;
  sot:definesAbsoluteDistance 2500.00;
  sot:locatedInSemanticSpace soa:kitchen_1;
  sot:facemask "False";
.
```

Figure 16: Latest detection received for facemask query

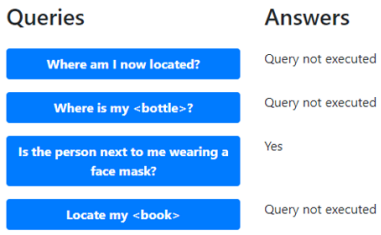

Figure 17: SAL dashboard Facemask Feedback Response

```
sd:detection_1
  rdf:type sot:Detection ;
  sot:detectsObject soa:laptop_1 ;
  sot:detectsObject soa:book_1 ;
  sot:detectsObject soa:cup_1 ;
  sot:detectsSemanticSpace soa:living_room_1 ;
  sot:hasConfidence "0.95"^^xsd:float ;
  sot:hasTimestamp "2021-06-15T09:40:00"^^xsd:dateTi
  sot:providedBy soa:camera_1 ;
.
sd:livingroom_1
  rdf:type seas:LivingRoom ;
  rdfs:label "living room" ;
.
sd:laptop_1
  rdf:type sot:Object ;
  rdf:type sot:Laptop ;
  sot:locatedInSemanticSpace soa:living_room_1 ;
  sot:definesAbsoluteDistance 3400.00;
.
sd:book_1
  rdf:type sot:Object ;
  rdf:type sot:Book ;
  sot:locatedInSemanticSpace soa:living_room_1 ;
  sot:definesAbsoluteDistance 5600.00;
.
sd:cup_1
  rdf:type sot:Object ;
  rdf:type sot:Cup ;
  sot:locatedInSemanticSpace soa:living_room_1 ;
  sot:definesAbsoluteDistance 4500.00;
.
```

Figure 18: Synthetic Database snapshot

that are either wearing or not wearing a facemask, in various distances from the user. The file that contains this synthetic data is presented in Figure 16 below. We have named this ontology file "synthetic data", so we use the preface "sd".

We notice that in this detection there are two people detected in the kitchen, John and an unknown person. John wears a facemask but the unknown person is not. Now, we assume that the user sends us a query asking whether the person closer to him wears a facemask. This query has been presented in Figure 8. Using a simple function, we produce a simple response file with the field "facemask" and the value "yes". We present the Query and Answer component of our dashboard in Figure 17.

We can modify the query and the output according to the user's needs. For example, if the user would like to know about the facemask status of all the people in his field of view, we can simply omit the LIMIT part of the query presented in Figure 6. Then by using another simple function we produce the message output shown below.

```
{
  "person": "John",
  "facemask": "yes",
  "person": "Unknown_Person_1",
  "facemask": "no"
}
```

D. Spatial Awareness and Localization queries experiments and results

For the purpose of experimenting with SAL queries we used a synthetic database that we know would accurately represent a series of detections in the SUITCEYES platform. Our synthetic data contains rooms and objects in those rooms that a user could choose to search for. In Figure 18, we can see part of this synthetic database. This part describes a detection which contains a living room as the recognized scene and a laptop, book and cup as the recognized objects in various distances.

Now, we are going to present the process that is being executed when a SAL query is asked, i.e., "Locate my <book>, by showing the queries and responses we receive from the GraphDB database. For this experiment, we receive a query from a user searching for his book. Our framework initiates SAL process by entering the loop and executing the query presented in Figure 19, on our GraphDB stored ontology. Note that this query is the query presented in Figure 8, with only the "?entity" variable changed to contain an object of the class Book.

The execution of the aforementioned query in combination with the function for the message output produces as a result the file shown in below:

```
{
  "timestamp": "2021-06-15T09:40:00",
  "detection": True,
  "offset": -13,
  "distance": 5.6
}
```

We present the Queries and Answer component of our dashboard where we verified the correctness of our responses in Figure 20.

As we can notice, our method reports back the data that are stored in our ontology database. As the distance value is not lower than 0.3, which we assume is the reaching distance for an object, after 1 second of the initial query execution, we perform again the same query. As our framework is designed to always receive new detections, we receive a detection in which the user has gone closer to the object after they receive directions based on our initial information. The new state of the detections in our ontology can be found in Figure 21.

In this latest detection we notice that the distance of the book object that the user is searching is decreased to 2.5 meters. We also notice that the cup object that was detected previously has not appeared in this detection, meaning that it is not in the field of view of the user. The output that we produce for this iteration of the OL procedure is presented below:

```
{
  "timestamp": "2021-06-15T09:40:10",
  "detection": True,
  "offset": 33,
  "distance": 2.5
}
```

In this output we can see again that we produce the correct information for the book object. We can also notice that the offset has changed into a positive value that means that the object is now on the right side of the field of view of the user. Once again, we verify our results using the dashboard in Figure 22.

The same procedure is repeated until the distance value becomes less than 0.3 meters. Then, we assume that the user can use his hands to find the object that they look for.

If in any time during this procedure the detection value becomes “False”, we just re-execute the query until we receive a detection which contains the object that the user is searching for.

Our SAL component is designed to work in combination with some module that gives some kind of directions to the user so that they can approach the object that they look for.

V. CONCLUSION AND FUTURE WORK

In this paper, we present a framework to support deafblind people using a rule-based toolbox of methods for object localization, spatial and situational awareness. The framework utilizes ontologies for uniform context data representation and the SPIN/SPARQL notation to extract knowledge and construct natural language responses. A dashboard visualizes inputs and outputs streaming over a universal message bus message exchange so as to enable developers integrate advanced detection methods from wearable cameras and user interfaces on haptic devices. The framework is validated through use cases scenarios showcased on its dashboard.

```
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.
    ?entity a sot:Book.
    ?scene rdf:type sot:SemanticSpace .
    ?entity sot:hasXGridSpace ?x.
    ?x sot:positionedInXGrid ?leftright.
    ?entity sot:hasSpatialContext ?sc.
    ?sc sot:definesAbsoluteDistance ?distance.
    }
ORDER BY desc(?timestamp)
LIMIT 1
```

Figure 19: Book SAL query

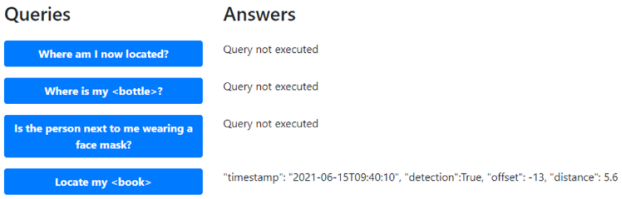

Figure 20: SAL response on dashboard

```
sd:detection_4
  rdf:type sot:Detection ;
  sot:detectsObject soa:laptop_1 ;
  sot:detectsObject soa:book_1 ;
  sot:detectsSemanticSpace soa:living_room_1 ;
  sot:hasConfidence "0.95"^^xsd:float ;
  sot:hasTimestamp "2021-06-15T09:40:10"^^xsd:dateTime
  sot:providedBy soa:camera_1 ;
.
sd:book_1
  rdf:type sot:Object ;
  rdf:type sot:Book ;
  sot:locatedInSemanticSpace soa:living_room_1 ;
  sot:definesAbsoluteDistance 2500.00;
.
```

Figure 21: Latest detection received

Figure 22: SAL updated response on dashboard

Such systems can greatly improve the quality of life of people with deafblindness especially regarding spatial awareness, but also situational awareness in the pandemic era, when people with disabilities need tailored measures and means to recognize their surroundings. Future work, includes integration of the framework with real data providers, such as sensors, cameras and the visual analysis. Moreover, a pilot phase includes deafblind users interacting with the system to

validate its acceptance and usefulness and to incrementally optimize the system for them.

ACKNOWLEDGMENTS

This work has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreements No. 78880814 SUITCEYES.

REFERENCES

[1] V. Kassiano, T. Stavropoulos, S. Nikolopoulos, I. Kompatsiaris, M. Riga, "Spatial Awareness for the Deafblind in Natural Language Presentation using SPIN Rules: A Use Case in the SUITCEYES Platform," The 12th International Conference on eHealth, Telemedicine, and Social Medicine, eTELEMED 2020, 21-25 November 2020, Valencia, Spain

[2] N. Olson, Nasrine, et al. "Sensor Technology, Gamification, HapticInterfaces in an Assistive Wearable," Journal on Technology and Persons with Disabilities 7 (2019): 79-87.

[3] O. Korn et al. "Empowering persons with deafblindness: Designing an intelligent assistive wearable in the SUITCEYES project," Proceedings of the 11th PErvasive technologies related to assistive environments conference. 2018.

[4] S. Darányi et al. "Static and Dynamic Haptograms to Communicate Semantic Content: Towards Enabling Face-to-Face Communication for People with Deafblindness." SEMAPRO 2019, The Thirteenth International Conference on Advances in Semantic Processing. International Academy, Research and Industry Association (IARIA), 2019.

[5] O. Korn, J. Gay, R. Gouveia, L. Buchweitz, A. S. Schulz, and M. Umfahrer. "Tactile navigation with checkpoints as progress indicators? only when walking longer straight paths." In Proceedings of the 13th ACM International Conference on PErvasive Technologies Related to Assistive Environments, pp. 1-8. 2020.

[6] The SUITCEYES project: https://suitceyes.eu/, Last access date 12/12/2021

[7] G. Meditskos, T. G. Stavropoulos, S. Andreadis, and I. Kompatsiaris, "KnowSense: A semantically-enabled pervasive framework to assist clinical autonomy assessment," in CEUR Worshop Proceedings, 2015.

[8] G. Fico et al., "Co-creating with consumers and stakeholders to understand the benefit of internet of things in smart living environments for ageing well: The approach adopted in the madrid deployment site of the activage large scale pilot," in IFMBE Proceedings, 2017, doi: 10.1007/978-981-10-5122-7_272.

[9] T. G. Stavropoulos, G. Meditskos, S. Andreadis, and I. Kompatsiaris, "Dem@Care: Ambient sensing and intelligent decision support for the care of dementia," in CEUR Workshop Proceedings, 2015.

[10] I. Maarala, X. Su, and J. Riekki, "Semantic Reasoning for Context-Aware Internet of Things Applications," IEEE Internet Things J., 2017, doi: 10.1109/JIOT.2016.2587060.

[11] A. Pliatsios, C. Goumopoulos, and K. Kotis, "Interoperability in IoT: A Vital Key Factor to Create the 'Social Network' of Things," in The Thirteenth International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies (UBICOMM 2019), 2019, pp. 63–69.

[12] J. A. Bateman et al. "A linguistic ontology of space for natural language processing," Artificial Intelligence 174.14 (2010): 1027-1071.

[13] D. Estival, C. Nowak, and A. Zschorn. "Towards ontology-based natural language processing," Proceeedings of the Workshop on NLP and XML (NLPXML-2004): RDF/RDFS and OWL in Language Technology. 2004.

[14] L. Haesung and J. Kwon. "Ontology model-based situation and socially-aware health care service in a smart home environment," International Journal of Smart Home 7.5 (2013): 239-250.

[15] C. Villalonga et al. "MIMU-Wear: Ontology-based sensor selection for real-world wearable activity recognition." Neurocomputing 250 (2017): 76-100.

[16] M. Sander et al. "Ontology-based translation of natural language queries to SPARQL." 2014 AAAI fall symposium series. 2014.

[17] G. Meditskos, S. Dasiopoulou, and I. Kompatsiaris, "MetaQ: A Knowledge-driven Framework for Contextaware Activity Recognition Combining SPARQL and OWL 2 Activity Patterns," Pervasive and Mobile Computing, vol 25, pp. 104-124, 2016.

[18] G. Meditskos and I. Kompatsiaris, "iKnow: Ontologydriven Situational Awareness for the Recognition of Activities of Daily Living," Pervasive and Mobile Computing, Vol 40, pp. 17-41, 2017.

[19] C. Fürber and M. Hepp. "Using SPARQL and SPIN for data quality management on the semantic web." International Conference on Business Information Systems. Springer, Berlin, Heidelberg, 2010.

[20] M. Mörchen, H. Kapoor, and S. Varughese. "Disability and COVID-19." Community Eye Health 33, no. 109 (2020): 10.

[21] K. Janowicz, A. Haller, S. JD Cox, D. Le Phuoc, and M. Lefrançois. "SOSA: A lightweight ontology for sensors, observations, samples, and actuators." Journal of Web Semantics 56 (2019): 1-10.

[22] M. M. A. Al-Mukhtar and A. T. A. Al-Assafy. "The implementation of foaf ontology for an academic social network," International Journal of Science, Engineering and Computer Technology, vol. 4.1, p. 10, 2014.

[23] M. Lefrançois, J. Kalaoja, T. Ghariani, and A. Zimmermann. "The SEAS Knowledge Model." PhD diss., ITEA2 12004 Smart Energy Aware Systems, 2017.

[24] The SUITCEYES Ontology:

https://mklab.iti.gr/results/the-suitceyes-ontology/, Last access date 12/12/2021

[25] The Protégé Software: https://protege.stanford.edu/, Last access date 12/12/2021