HDGI: A Human Device Gesture Interaction Ontology for the Internet of Things

Madhawa Perera[1,2](✉), Armin Haller[1], Sergio José Rodríguez Méndez[1], and Matt Adcock[1,2]

[1] Australian National University, Canberra, ACT 2601, Australia
{madhawa.perera,armin.haller,sergio.rodriguezmendez, matt.adcock}@anu.edu.au
[2] CSIRO, Canberra, AU 2601, Australia
{madhawa.perera,matt.adcock}@csiro.au
https://cecs.anu.edu.au/
https://csiro.au/

Abstract. Gesture-controlled interfaces are becoming increasingly popular with the growing use of Internet of Things (IoT) systems. In particular, in automobiles, smart homes, computer games and Augmented Reality (AR)/Virtual Reality (VR) applications, gestures have become prevalent due to their accessibility to everyone. Designers, producers, and vendors integrating gesture interfaces into their products have also increased in numbers, giving rise to a greater variation of standards in utilizing them. This variety can confuse a user who is accustomed to a set of conventional controls and has their own preferences. The only option for a user is to adjust to the system even when the provided gestures are not intuitive and contrary to a user's expectations.

This paper addresses the problem of the absence of a systematic analysis and description of gestures and develops an ontology which formally describes gestures used in Human Device Interactions (HDI). The presented ontology is based on Semantic Web standards (RDF, RDFS and OWL2). It is capable of describing a human gesture semantically, along with relevant mappings to affordances and user/device contexts, in an extensible way.

Keywords: Ontology · Gesture · Semantic web · Internet of Things · Gesture interfaces

1 Introduction

Gesture-based systems are becoming widely available hence widely explored as methods for controlling interactive systems. Especially in modern automobiles, smart homes, computer games, and Augmented Reality (AR)/ Virtual Reality (VR) applications, gestures have become prevalent due to their accessibility to

J. Z. Pan et al. (Eds.): ISWC 2020, LNCS 12507, pp. 111–126, 2020.
https://doi.org/10.1007/978-3-030-62466-8_8

everyone. Most of these gesture interactions consist of physical movements of the face, limbs, or body [19] and allow users to express their interaction intentions and send out corresponding interactive information [9] to a device or a system. However, most of the gestural interfaces are built based on a manufacturer's design decision.

Introducing "guessability of a system" in 2005, Wobbrock et al. [18] emphasize that "a user's initial attempts at performing gestures, typing commands, or using buttons or menu items must be met with success despite the user's lack of knowledge of the relevant symbols". Their study enables the collection of end users' preferences for symbolic input, and is considered as the introduction of Gesture Elicitation Studies (GES) [17]. Since then many researches have attempted to define multiple gesture vocabularies. However, a majority of them are limited in their scope and limited to specific uses. As a result, an impressive amount of knowledge has resulted from these GES. At present this knowledge is cluttered. There are multiple studies that show 'best gestures' for the same referent. Here, the referent is the effect of a gesture or the desired effect of an action which the gestural sign refers to [8]. Hence, there are redundant gesture vocabularies. If all the knowledge of GES is properly linked, researchers could find gesture vocabularies that are defined for similar referents. However, a lack of linked data in this area has resulted in researchers conducting new GES whenever they need a particular gesture-referent mapping instead of using existing knowledge. Hence, we see the necessity of a gesture ontology that can describe gestures with its related referents and facilitate automated reasoning.

Further, there currently exist several sensors, such as Microsoft Kinect, allowing out-of-the-box posture or movement recognition which allow developers to define and capture mid-air gestures and to use them in various applications. With the advancements in AR and VR, the use of gestural interfaces has increased as these immersive technologies tend to use more intuitive Human Compute Interaction (HCI) techniques. All these systems have the capability to detect rich gestural inputs. This has resulted in designers, developers, producers and vendors integrating gesture interfaces into their products contributing to a surge in their numbers and causing greater variation in ways of utilizing them. Riener et al. [13] also show that, most of the time, "system designers define gestures based on their own preferences, evaluate them in small-scale user studies, apply modifications, and teach end users how to employ certain gestures". Further, Riener et al. [13] state that this is problematic because people have different expectations of how to interact with an interface to perform a certain task. This could confuse the users who are accustomed to a set of conventional controls.

Most of the time, these systems have either binary or a few choices when it comes to gesture selection. Therefore, users do not have much of a choice even though the manufacturer defined gestures are undesirable or counter-intuitive. For example, if we take Microsoft HoloLens[1], its first version (HoloLens 1) has a 'bloom' gesture to open its 'start' menu. In contrast, in HoloLens 2, a user has to pinch their thumb and index finger together while looking at the start icon

[1] https://docs.microsoft.com/en-au/hololens/.

that appears near a user's wrist when they hold out their hand with their palm facing up, to open the start menu. Optionally they can also tap the icon that appears near the wrist using their other hand.

BMW's iDrive infotainment system expects users to point a finger to the BMW iDrive touchscreen to 'accept a call' whereas Mercedes-Benz' User Experience (MBUX) multimedia infotainment system uses the same gesture to select an icon on their touch screen. Further, online search engines currently do not provide sufficient information for gesture related semantics. For example, search query to retrieve 'gestures to answer a call in a car', would not provide relevant gesture vocabularies supported by different vendors. Designers/developers have to find individual studies separately and read/learn necessary data manually. Being able to retrieve semantics and refer to a central location which maps all the available gestures to the affordance of 'answering a call in a car' would be convenient for designers and developers in such situations.

Additionally, understanding semantics of these gestures and inter-mapping them will help to bring interoperability among interfaces increasing User Experience (UX). The problem is how to do this mapping. Our approach is to design an ontology to map existing and increasingly prolific gesture vocabularies and their relationships to systems with the intention of providing the ability to understand and interpret user gestures. Henceforth, users are individually shown the desired effect of an action (called a referent) to their preferred gestures.

Villarreal-Narvaez et al.'s [17] most recent survey paper shows that a majority of gestures are performed using the upper limbs of a human body (i.e. hands). Thereby keeping extensibility in mind, we designed a Human Device Gesture Interaction (HDGI) ontology to describe and map existing and upcoming upper limb related gestures along with relevant device affordances. This allows systems to query the ontology after recognizing the gesture to understand its referents without having to be pre-programmed. This further helps personalisation of gestures for particular sets of users. As such, a user does not have to memorize a particular gesture for each different system, which improves system reliability.

This paper describes the HDGI ontology and its sample usage and state of the art in this area. First, in Sect. 2 we discuss existing approaches to address the problem of ubiquitousness in human device gesture interactions. In Sect. 3, we describe the syntax, semantics, design and formalisation of HDGI v0.1, and the rationale behind such a design. In Sect. 4, we illustrate tools for mapping HDGI v0.1 to leap motion[2] and the Oculus Quest[3] devices. This serves as an evaluation of the expressive power of our ontology and provides developers and designers with a tool on how to integrate the HDGI ontology in their development. We conclude and discuss future work in Sect. 5.

2 Related Work

A large number of studies can be found dealing with the problem of hand gesture recognition and its incorporation into the design and development of gestural

[2] See https://www.ultraleap.com/.

[3] See https://www.oculus.com/quest/.

interfaces [16]. In most of these cases, gestures are predefined with their meaning and actions. Yet, the studies do seem to explore the capability of identifying the relationship beyond predefined mappings of a gesture. Thus, we see very few studies that have attempted to define and formalise the relationship between each gesture. A review conducted by Villarreal Narvaez et al. [17] in 2020 shows that GES has not yet reached its peak which indicates that there will be many more gesture-related vocabularies in the future which, consequently increases the need to have interoperability between them.

One approach that has been adopted by researchers is to define taxonomies, enabling designers and manufacturers to use standard definitions when defining gesture vocabularies. Following this path, Scoditti et al. [15] proposed a gestural interaction taxonomy in order to guide designers and researchers, who "need an overall systematic structure that helps them to reason, compare, elicit (and create) the appropriate techniques for the problem at hand." Their intention is to introduce system-wide consistent languages with specific attention for gestures. However, those authors do not map existing gesture vocabularies with semantical relationships. Following this, Choi et al. [3] developed a 3D hand gesture taxonomy and notation method. The results of this study can be used as a guideline to organize hand gestures for enhancing the usability of gesture-based interfaces. This again follows a similar approach to Scoditti et al. [15]. However, this research is restricted to 6 commands (43 gestures) of a TV and blind(s) that were used in the experiment. Therefore, further experiments with an increased number of commands is necessary to see the capability and adaptability of the proposed taxonomy and notation method. Also, this notation is using a numeric terminology which is not easily readable, unless designers strictly follow a reference guide that is provided. In addition, they mention that the size or speed of hand gestures have not been considered in their approach.

Moving beyond taxonomies there is also existing research using ontologies. Osumer et al. [12] have modelled a gesture ontology based on a Microsoft Kinect-based skeleton which aims to describe mid-air gestures of human body. Their ontology mainly focuses on capturing the holistic posture of the human body, hence misses details like the finger pose or movements and a detailed representation of the hand. In addition, the ontology is not openly shared, hence it prevents use and extensibility. In addition, their main contribution is to have a "sensor-independent ontology of body-based contextual gestures, with intrinsic and extrinsic properties" where mapping different gestures with their semantic relationships to affordances is not considered.

Khairunizam et al. [6] have conducted a similar study with the intention of addressing the challenge of how to increase the knowledge level of the computational systems to recognize gestural information with regard to arm movements. In their research, they have tried to describe knowledge of the arm gestures and attempted to recognize it with a higher accuracy. This can be identified as an interesting study where the authors have used Qualisys motion capture (MOCAP) to capture the movement of the user's right arm when they perform an arm gesture. However, their focus was mainly on recognizing geometrical gestures and the gesture set was limited to 5 geometrical shapes. Again, their ontological framework does not consider the mapping of other gestures that carry similar referents.

Overall, the attempts above have a different scope compared to our ontology. Our focus is not on modelling the infinite set of concepts, features, attributes and relationships attached to arm-based gestures. We do not consider gestures that do not carry a referent to a particular affordance of a device. Nonetheless, our ontology is extensible to allow the addition of emerging gestures with a referent to an affordance or to be extended to other body parts, i.e. extending the gestures beyond the upper limbs of the human body. As a best practise we have used existing ontologies whenever they fit and provided mappings to concepts and properties in these ontologies.

3 Human Device Gesture Interaction (HDGI) Ontology

The HDGI ontology models the pose and/or movement of human upper limbs that are used to interact with devices. This ontology; 1) Describes gestures related to device interactions and which are performed using a human's upper limb region; 2) Maps affordances and human gestures to facilitate devices/automated systems to understand different gestures that humans perform to interact with the same affordances; 3) acts as a dictionary for manufacturers, designers and developers to search/identify the commonly used gestures for certain affordances, and/or to understand the shape and dynamics of a certain gesture. The ontology is developed with a strong focus on flexibility and extensibility where device manufacturers, designers and users can introduce new gestures and map its relations to necessary affordances. Most importantly, this does not enforce designers and manufacturers to follow a standard but it maps the ubiquitousness in gesture vocabularies by linking them appropriately. The aim of this study is to define a semantic model of gestures combined with its associated knowledge. As such, GES becomes more permissive, which opens up the opportunity to introduce a shareable and reusable gesture representation that can be mapped according to the relationships introduced in HDGI.

We defined a new namespace https://w3id.org/hdgi with the prefix hdgi (registered entry at http://prefix.cc) for all the classes used in the ontology so as to be independent of external ontologies. However, we have provided relevant mappings to external ontologies where appropriate. We are using w3id.org as the permanent URL service. Furthermore, the relevant code, data and ontology are made available for the community via GitHub[4] allowing anyone interested to join as a contributor.

3.1 Design Rationale

We have arranged the classes and properties of the HDGI ontology[5] to represent human upper limb region gestures with their associated affordances

[4] https://github.com/madhawap/human-device-gesture-interaction-ontology.

[5] https://w3id.org/hdgi.

and context. The ontology is designed around a core that consists of seven main classes: hdgi:Gesture, hdgi:BodyPart, hdgi:Pose, hdgi:Movement, hdgi:Affordance, hdgi:Device, and hdgi:Human, establishing the basic relationships between those along with an hdgi:Observer and hdgi:Context classes. Figure 1 depicts this core ontology design pattern[6]. This pattern will be registered in the ontology design pattern initiative[7]. Please note that the ontology introduces all classes and relationships depicted in Fig. 1 in its own namespace, but for illustration purposes we use their equivalent classes/properties from external ontologies, when appropriate. All the classes and properties are expressed in OWL2 and we use Turtle syntax throughout our modelling.

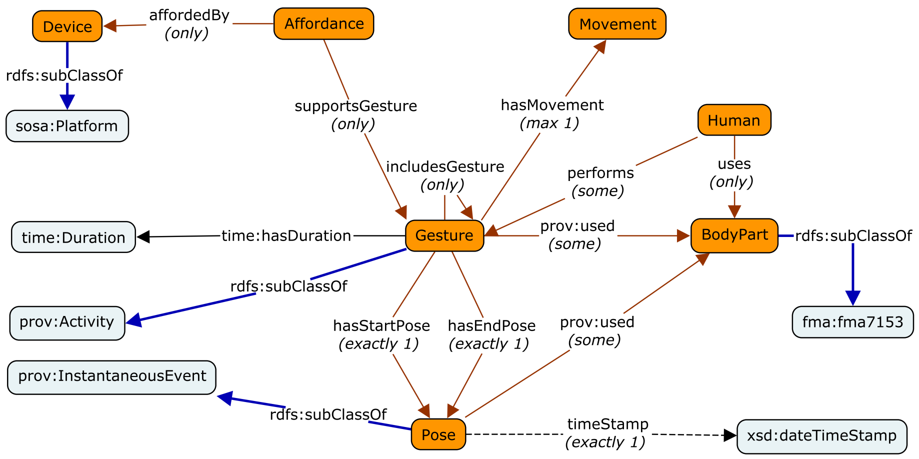

Fig. 1. The core structure of HDGI, showing relationships to external ontologies

We use global domain and range restrictions on properties sparingly, but as much as possible, use guarded local restrictions instead, i.e. universal and existential class restrictions for a specific property such that only for instances of that property with that class as the subject, the range of the property is asserted. This helps in the alignment of the ontology with other external ontologies, in particular, if they also use guarded local restrictions. We provide alignments to these ontologies as separate ontology files in Github at: human-device-interaction-ontology/ontologyAlignments[8].

[6] The prefixes denoted in the figure are: sosa: <http://www.w3.org/ns/sosa/>, time: <http://www.w3.org/2006/time#>, prov: <http://www.w3.org/ns/prov#>, fma: <http://purl.org/sig/ont/fma/>.

[7] See http://ontologydesignpatterns.org.

[8] https://github.com/madhawap/human-device-gesture-interaction-ontology/tree/master/v0.1/ontologyAlignments.

3.2 Core Classes and Properties

Gesture A hdgi:Gesture is defined in such a way that it distinguishes two atomic types of gestures, namely static and dynamic gestures. A dynamic gesture consists of exactly one start hdgi:Pose at a given time, exactly one end hdgi:Pose at a given time, an atomic hdgi:Movement, and involves a single hdgi:BodyPart at a time. However, since a gesture can have multiple poses and movements of multiple body parts, we provide a means to define a sequence of gestures. Since, the ontology is designed in a way that it can capture and describe individual body parts separately, a gesture that involves multiple movements and poses of body parts can be described using the object property hdgi:includesGesture that aggregates hdgi:Gesture and through their mapping to Allen time [1] puts them in sequence or concurrent. That is, a gesture can contain one or more gestures.

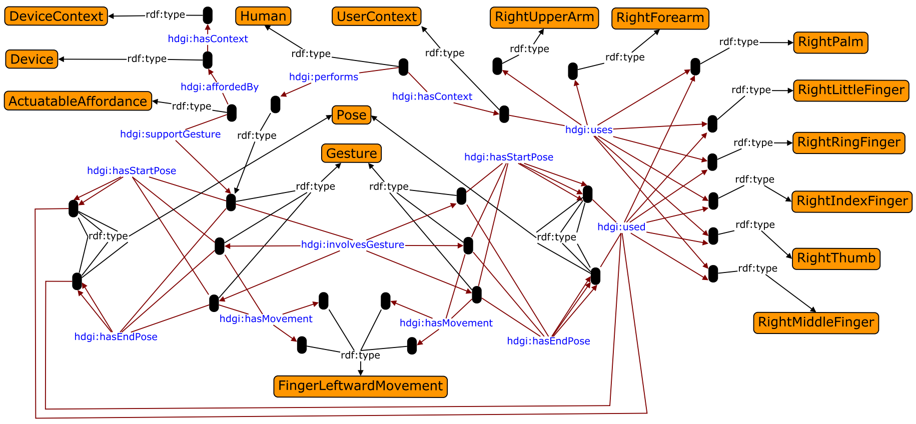

Fig. 2. HDGI dynamic gesture instance example

To give a concrete example of the modelling of a dynamic gesture, we use a ‘swipe gesture’ performed with the right hand (named ‘right hand swipe left’) illustrated below in Listing 1.1 and Fig. 2. As per the description above, ‘right hand swipe left’ consists of eight atomic gestures. Only some of these atomic gestures are shown in Fig. 2 and listed in Listing 1.1 and each of those include a single body part, a start pose and an end pose, with a movement.

For extensibility, we added several possible gesture subclasses such as hdgi:HandGesture, hdgi:ForearmGesture, hdgi:FacialGesture, hdgi:LegGesture, hdgi:UpperArmGesture etc. However, at this moment only hand, forearm, and upper arm gestures are modelled in detail in HDGI.

Listing 1.1. Gesture Right Hand Swipe Left

```
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix hdgi: <https://w3id.org/hdgi#> .

hdgi:Right_Hand_Swipe_Left rdf:type, hdgi:Gesture ;
              hdgi:includesGesture hdgi:Right_Forearm_Move_Left ,
                                   hdgi:Right_IndexFinger_Move_Left ,
                                   hdgi:Right_LittleFinger_Move_Left ,
                                   hdgi:Right_MiddleFinger_Move_Left ,
                                   hdgi:Right_Palm_Move_Left ,
                                   hdgi:Right_RingFinger_Move_Left ,
                                   hdgi:Right_Thumb_Move_Left ,
                                   hdgi:Right_UpperArm_Move_Left .

hdgi:Right_Forearm_Move_Left rdf:type, hdgi:ForearmGesture ;
                          hdgi:hasEndPose hdgi:Right_ForearmPose_2 ;
                          hdgi:hasMovement hdgi:Right_ForearmLeftward_Move ;
                          hdgi:hasStartPose hdgi:Right_ForearmPose_1 ;
                          hdgi:used hdgi:Sofia_RightForearm .

hdgi:Right_Palm_Move_Left rdf:type, hdgi:HandGesture ;
                       hdgi:hasEndPose hdgi:Right_PalmOutwardPose_2 ;
                       hdgi:hasMovement hdgi:Right_PalmLeftward_Move ;
                       hdgi:hasStartPose hdgi:Right_PalmOutwardPose_1 ;
                       hdgi:used hdgi:Sofia_Right_Hand_Palm .
```

BodyPart For modelling body parts, we reuse and extend concepts and classes in the Foundational Model of Anatomy (FMA) ontology [14]. Again, though we focus only on a human's upper limb region, the hdgi:BodyPart class is defined in an extensible way with the motive of allowing representation of further body parts to describe new poses in the future. We are not modelling all the biological concepts that are described in FMA, but only the relevant classes for HDI. While preserving FMA class definitions and structures, we define hdgi:UpperArm, hdgi:Forearm, hdgi:Palm, and hdgi:Finger as the basic building blocks of the 'upper limb region'. The hdgi:UpperArm class is an equivalent class to the `Arm` class in FMA. The hdgi:Finger class is further divided to represent each individual finger as hdgi:Thumb, hdgi:IndexFinger, hdgi:MiddleFinger, hdgi:RingFinger, and hdgi:LittleFinger and are mapped to the respective subclasses of a "Region of hand" in FMA. These fingers are further divided into left and right entities. Figure 3 depicts each of these sections of the 'upper limb region'. Thus, we define a gesture as a combination of one or more poses and/or movements involved by one or more of these eight sections of upper limb region.

Pose. Each body part can be involved in a pose. In other words, a `Pose` must hdgi:involves one hdgi:BodyPart at a point in time. For each body part there is a corresponding, potential pose. Stepping down a layer of abstraction, the hdgi:Pose class describes the exact placement of a pose in a 3D space, by modelling the 'position' and 'rotation' of a pose. The hdgi:hasPosition and hdgi:hasRotation relationships are used for this mapping; e.g. hdgi:ThumbCurled –> hdgi:hasPosition –> xPosition. In order to avoid the problem of having different origin points based on the gesture recognition device configurations, the HDGI ontology always considers relative positions. That is, upper arm positions are always relative to the shoulder joint (Refer to Fig. 3 - point A). The position

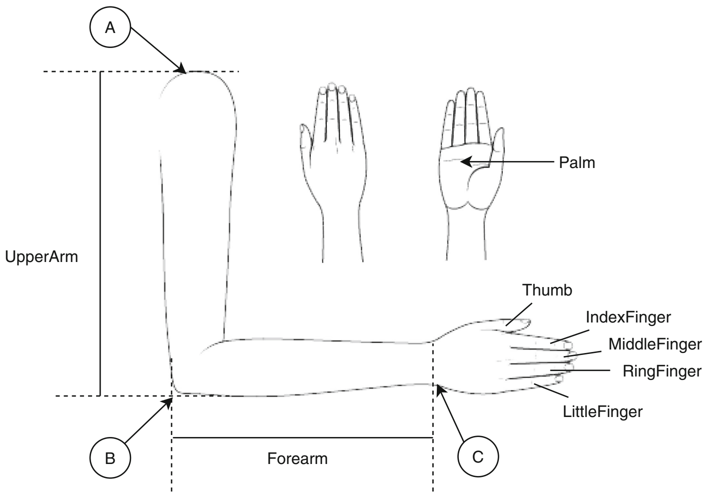

Fig. 3. Body parts of Upper limb region

of a hdgi:ForearmPose is always relative to the elbow joint (cf. Figure 3 – point B). Palm and finger positions are always relative to the wrist (cf. Figure 3 – point C). Further, the hdgi:Position class must describe the local coordinate system that its hdgi:xPosition, hdgi:yPosition, and zPosition values are based on. Thus, every hdgi:Position must have a hdgi:hasLocalCoordinateSystem object property with a hdgi:LocalCoordinateSystem as its range. This is to avoid problems, such as different SDKs/systems using slightly different coordinate systems. For example, Unity3D[9] is using a left-hand rule coordinate system where the Z-axis always points outwards from the users. In contrast, the leap-motion SDK uses a right-hand rule where the Z-axis is pointed inwards. In order to allow either type of modelling and to avoid unnecessary conversions steps, we separately model the hdgi:LocalCoordinateSystem class and hdgi:Position class relationship. The rotation of a pose can be represented in two different ways. Some systems use yaw (angle with y-axis), pitch (angle with x-axis), and roll (angle with z-axis) angles to describe the rotation of a 3D rigid body, whereas some systems use quaternions. By allowing support for both of these representations (yet one at a time), we keep our model flexible and able to model data received from different manufacturers/devices.

Further, a hdgi:Pose represents a static gesture. Thus, similar to the hdgi:Gesture class, a hdgi:Pose can contain one or more poses within itself. A hdgi:Pose always has a time stamp and involves a single body part at a time (thus, individual body parts can be modelled separately). Again, for extensibility, we added several possible poses as subclasses such as hdgi:LegPose, hdgi:FootPose, etc. However, at the moment we only model hdgi:UpperArm, hdgi:Forearm, hdgi:Palm, and each individual hdgi:Finger poses.

[9] See https://unity.com/.

Listing 1.2. Pose

```
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix hdgi: <https://w3id.org/hdgi#> .

hdgi:Right_ForearmPose_1 rdf:type, hdgi:ForearmPose ;
                      hdgi:hasPosition hdgi:Right_Forearm_Position_1 ;
                      hdgi:hasRotation hdgi:Right_Forearm_Rotation_1 ;
                      hdgi:used hdgi:Sofia_RightForearm ;
                      hdgi:timeStamp "2020-04-12T21:30:11-10:00"^^xsd:dateTimeStamp .

hdgi:Right_ForearmPose_2 rdf:type, hdgi:ForearmPose ;
                      hdgi:hasPosition hdgi:Right_Forearm_Position_2 ;
                      hdgi:hasRotation hdgi:Right_Forearm_Rotation_2 ;
                      hdgi:used hdgi:Sofia_RightForearm ;
                      hdgi:timeStamp "2020-04-12T21:31:31-10:00"^^xsd:dateTimeStamp .

hdgi:Right_PalmOutwardPose_1 rdf:type, hdgi:PalmOutwardPose ;
                          hdgi:used hdgi:Sofia_Right_Hand_Palm ;
                          hdgi:timeStamp "2020-04-12T21:30:11-10:00"^^xsd:dateTimeStamp .

hdgi:Right_PalmOutwardPose_2 rdf:type, hdgi:PalmOutwardPose ;
                          hdgi:used hdgi:Sofia_Right_Hand_Palm ;
                          hdgi:timeStamp "2020-04-12T21:31:31-10:00"^^xsd:dateTimeStamp .
```

Listing 1.2 provides an example of a pose modelling related to the gesture "Right Hand Swipe Left". The example models the start pose and the end pose of the right forearm and the right palm. As per the description above, each hdgi:Pose hdgi:used only hdgi:BodyPart and has exactly one hdgi:timestamp with a maximum of one hdgi:Position and a hdgi:Rotation (hdgi:Rotation could be modeled either using Euler angles (hdgi:xRotation (roll), hdgi:yRotation (pitch), hdgi:zRotation (yaw)) or hdgi:Quaternion based on received data). Listing 1.3 further explains the hdgi:Position and hasLocalCoordinateSystem mappings. Each hdgi:Position has maximum of one hdgi:xPosition, hdgi:yPosition and hdgi:zPosition and exactly one hdgi:LocalCoordinateSystem. Notice in hdgi:LocalCoordinateSystem, each axis direction is pre-known (enum), hence for hdgi:xAxisDirection it is either "leftward" or "rightward", for hdgi:yAxisDirection it is either "upward" or "downward", and for hdgi:zAxisDirection it is "outward" or "inward".

Listing 1.3. Position

```
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix hdgi: <https://w3id.org/hdgi#> .

hdgi:Right_Forearm_Position_1 rdf:type, hdgi:Position ;
                           hdgi:hasLocalCoordinateSystem hdgi:
                                ↪ NewForearm_LocalCoordinateSystem ;
                           hdgi:hasUnitOfMeasure hdgi:centimeters ;
                           hdgi:xPosition "0.0"^^xsd:float ;
                           hdgi:yPosition "30.211"^^xsd:float ;
                           hdgi:zPosition "19.021"^^xsd:float .


hdgi:Right_Forearm_Position_2 rdf:type, hdgi:Position ;
                           hdgi:hasLocalCoordinateSystem hdgi:
                                ↪ NewForearm_LocalCoordinateSystem ;
                           hdgi:hasUnitOfMeasure hdgi:centimeters ;
                           hdgi:xPosition "16.517"^^xsd:float ;
                           hdgi:yPosition "28.456"^^xsd:float ;
                           hdgi:zPosition "19.121"^^xsd:float .
```

```
hdgi:NewForearm_LocalCoordinateSystem rdf:type, hdgi:LocalCoordinateSystem ;
                                   hdgi:x-axisDirection "leftward" ;
                                   hdgi:y-axisDirection "upward" ;
                                   hdgi:z-axisDirection "outward" .
```

Movement. The hdgi:Movement class only relates to dynamic gestures and has no relationship to a hdgi:Pose. A hdgi:Movement consists of a predefined set of movements that we identified as sufficient to describe the movements of hdgi:UpperArm, hdgi:Forearm, hdgi:Palm, and hdgi:Finger. This is extensible for designers and developers to include their own new movements. As this is not tightly-coupled with other classes such as hdgi:Gesture, hdgi:Pose, and hdgi:BodyPart, the flexibility is there for customisations. Each hdgi:Movement is atomic (that is related to only one position change or one rotation change) and must have exactly a single hdgi:Duration. This can be derived from hdgi:timestamp difference between start hdgi:Pose and end hdgi:Pose.

Affordances. According to Norman [11] "the term affordance refers to the perceived and actual properties of the thing that determines just how the thing could possibly be used". Later on, this view has become standard in Human Computer Interaction and Design [2]. Further, Maier et al. [7] define affordances to be "potential uses" of a device. This implies that the "human is able to do something using the device" [2]. Hence affordances of a device can be stated as the set of "all potential human behaviours that the device might allow" [2]. Therefore, Brown et al. [2] conclude that "affordances are context dependent action or manipulation possibilities from the point of view of a particular actor". This highlighted the necessity for us to model both an hdgi:Affordance and a hdgi:Context (both hdgi:UserContext and hdgi:DeviceContext) class when modelling Human Device Gesture Interactions. As a user's choice of gestures is heavily based on their context [10], to understand the correct intent it is important that HDGI can map both the context and affordance. This helps systems to understand user specific gesture semantics and behave accordingly.

In gesture interactions, necessary affordances are communicated by the user to a device via a gesture that is supported by the device. If there is an openly accessible gesture affordance mapping with automated reasoning, we could integrate multiple gesture recognition systems to cater for user needs, and thereby increase user experience (UX). For example, assume that Device A has an affordance X and Device B has affordance Y. If a user performs a gesture which can only be detected by Device B but the user's intent is to interact with affordance X, by using the mappings in hdgi-ontology and the use of automated reasoning, Device B would be able to understand the user intent and communicate that to Device A accordingly. This further implies, that it is the affordance that should be mapped to a gesture rather than the device. This is modelled as hdgi:Affordance –> hdgi:supportsGesture –> hdgi:Gesture, where an affordance can have none to many supported gestures. A hdgi:Device can be a host to multiple affordances and the same affordance can be hosted by multiple devices. Hence, hdgi:Affordance –> hdgi:affordedBy –> hdgi:Device has cardinality of many to many. Here, hdgi:Device is a sub class of sosa:Platform. SOSA [5] (Sensor, Observation, Sample, and Actuator) is a lightweight but self-contained core ontology which itself is the core of the new Semantic Sensor Network (SSN) [4] ontology. The SSN ontology describes sensors and their observations, involved procedures, studied features of interest, samples, and observed properties, as well as actuators. We further reuse sosa:Sensor and sosa:Actuator and hdgi:ActuatableAffordance and hdgi:ObservableAffordance are sub-classes of sosa:ActuatableProperty and sosa:ObservableProperty.

In addition, the HDGI ontology models the relationship between hdgi:Device and a hdgi:DeviceManufacturer as there can be the same gesture mapped to different affordances by different vendors or the same affordance can be mapped to different gestures (refer to the BMW and Mercedes-Benz example in Sect. 1). We model this in HDGI through the hdgi:Device hdgi:manufacturedBy a hdgi:DeviceManufacturer relationship where a device must have just one manufacturer.

Listing 1.4 provides an example modelling of hdgi:Affordance, hdgi:Device and hdgi:Context relationships corresponding to the description above above.

Listing 1.4. Affordance and Device Mapping

```
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix hdgi: <https://w3id.org/hdgi#> .

hdgi:go_to_next_channel rdf:type, hdgi:ActuatableAffordance ;
                    hdgi:affordedBy hdgi:Television ;
                    hdgi:supportsGesture hdgi:Right_Palm_Move_Left .

hdgi:Television rdf:type, hdgi:Device ;
              hdgi:hasContext hdgi:Hotel_Room_Type_1 ;
              hdgi:manufacturedBy hdgi:Samsung .

hdgi:Hotel_Room_Type_1 rdf:type, hdgi:DeviceContext .

hdgi:Samsung rdf:type, hdgi:DeviceManufacturer .

hdgi:Hotel_Room_Type_1_Visitor rdf:type, hdgi:UserContext .

hdgi:Sofia rdf:type, hdgi:Human ;
         hdgi:hasContext hdgi:Hotel_Room_Type_1_Visitor ;
         hdgi:performs hdgi:Right_Palm_Move_Left ;
         hdgi:uses hdgi:Sofia_Right_Hand_Index_Finger ,
                 hdgi:Sofia_Right_Hand_Little_Finger ,
                 hdgi:Sofia_Right_Hand_Middle_Finger ,
                 hdgi:Sofia_Right_Hand_Palm ,
                 hdgi:Sofia_Right_Hand_Ring_Finger ,
                 hdgi:Sofia_Right_Hand_Thumb .
```

Most importantly, this is one of the major contributions in this ontology and when correctly modelled, this will help systems to automatically identify the semantics of a user's gesture and perform the necessary affordance mapping through an interconnected knowledge base instead of predefined one to one mappings. This allows gesture recognition systems to run gesture recognition, detection, mappings and communication separately in independent layers.

4 Device Mappings to HDGI

In addition to ontology building and annotating, it is equally important to consider its integration and documentation as a part of ontology engineering. Figure 4 illustrates a proof-of-concept implementation of the HDGI ontology. Here we wrapped a set of predefined SPARQL endpoints with RESTful APIs, in order to make the integration with third party Software Development Kits and Services easier and faster.

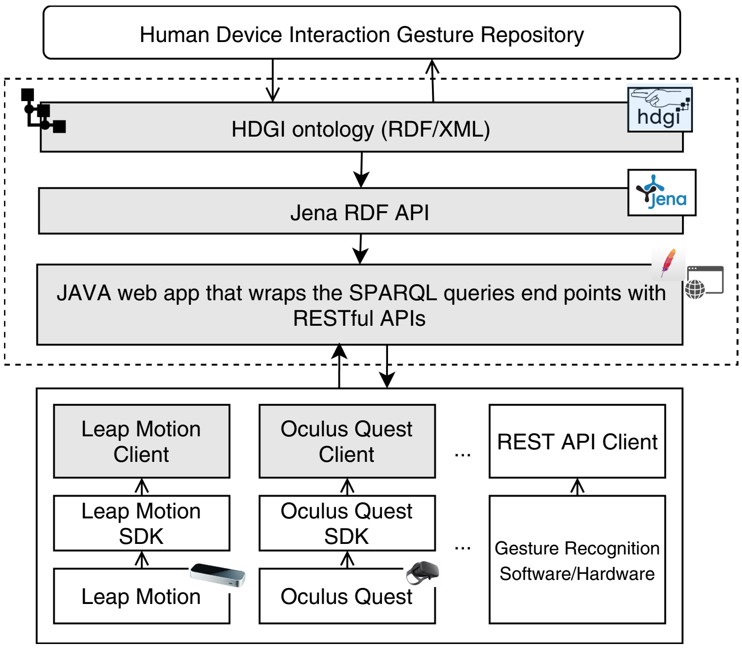

Fig. 4. HDGI ontology data integration workflow

HDGI-Mapping Service, is a fully API-driven RESTful web service, where designers, device manufacturers and developers can refer to one place - the HDGI-gesture repository - to find currently available and contemporary gestures and their relevant mappings to device affordances. In addition, APIs further allow them to define their own gesture vocabularies and map them and upload them to the gesture repository. This means that their gesture vocabularies will be easily accessible to the research community. We anticipate that this will help to reduce the redundant gesture vocabularies and increase the reuse of existing ones, eventually helping to reduce the ubiquitousness currently prominent in gestural interfaces.

In our study, we looked at the gesture vocabularies in the current literature and tried to map them into the ontology as a starting point. This allows using the HDGI-service endpoints to query about available gesture vocabularies. As we have made this an OpenSource project under Apache 2.0 license, anyone can contribute to the open GitHub code repository for further improvements. In addition, they can deploy this service in their own private cloud, if necessary. Either way, adhering to the HDGI ontology mappings will allow universal integration of gesture data instead of having a universal gesture standard that is not yet available and may never emerge.

Further information on HDGI mappings can be explored here[10]. Sample mapping service (the web application) code is available to anyone to download locally, and continue the integration with their gesture recognition software tools. The prerequisites to run the web application are Java version 1.9 or higher and an Apache Tomcat server. A 'how-to' documentation is provided here[11]. We have further added an API and architecture documentation which helps if someone needs to customize the web application itself, if they want to make customised SPARQL endpoints and to define new RESTful endpoints to suit any customizable needs, and to run as a private service. A complete API documentation can also be found here[12], and we are currently working on integrating the swagger UI, and Swagger codegen capabilities to the HDGI web app. Thus, users can get a comprehensive view of the API, understand endpoint structures and try it online in real-time. Further, with the integration of Swagger codegen, we will allow instant generation of API client stubs (client SDKs for APIs) from different languages including JavaScript, JAVA, Python, Swift, Android etc. which will make the integration of the APIs into different gesture recognition software/services even faster and easier.

5 Conclusion

This work presents the Human Device Gesture Interaction (HDGI) ontology: a model of human device gesture interactions; which describes gestures related to human device interactions and maps them with corresponding affordances. This is an initial step towards building a comprehensive human device gesture interaction knowledge base with the ultimate purpose of bringing better user experience. The HDGI ontology can assist gesture recognition systems, designers, manufacturers, and even developers to formally express gestures and to carry automated reasoning tasks based on relationships between gesture and devices affordances. While developing the ontology, we extracted elements observed from existing gesture vocabularies defined in previous studies. We also present a Web service interface (HDGI Mapping service) that can be integrated with existing gesture recognition systems.

The intention and scope of the HDGI ontology can be summarized as follows: 1) describe gestures related to HDI performed using the human upper-limb

[10] https://github.com/madhawap/human-device-gesture-interaction-ontology/blob/master/README.md.
[11] https://w3id.org/hdgi/mappings-docs.
[12] https://w3id.org/hdgi/mappings-api.

region; 2) map the relationship between affordances and a particular gesture based on the user context (therefore, devices could understand different gestures that human performs to interact with same affordances); and 3) act as a dictionary and a repository for manufacturers, developers, and designers to identify the commonly used gestures for certain affordances, to specify formally what a certain gesture means, and to introduce new gestures if necessary.

As future work, there are several possible extensions that can be made to the ontology by incorporating more gesture types such as facial gestures, head gestures etc. Further, we are planning to release and deploy the HDGI RESTful service in the Cloud and release API clients to leading hand-gesture supported systems such as Microsoft HoloLens 2, Microsoft Kinect, and Soli. Since gesture interactions in Mixed Reality (MR) are becoming increasingly popular, we plan to conduct several gesture elicitation studies using Microsoft HoloLens 2 especially to map gesture interactions in MR to the HDGI ontology.

References

1. Allen, J.F.: Maintaining knowledge about temporal intervals. Commun. ACM **26**(11), 832–843 (1983)
2. Brown, D.C., Blessing, L.: The relationship between function and affordance. In: ASME 2005 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, American Society of Mechanical Engineers Digital Collection, pp. 155–160 (2005)
3. Choi, E., Kim, H., Chung, M.K.: A taxonomy and notation method for three-dimensional hand gestures. Int. J. Ind. Ergon. **44**(1), 171–188 (2014)
4. Haller, A., et al.: The modular SSN ontology: a joint W3C and OGC standard specifying the semantics of sensors, observations, sampling, and actuation. Semanti. Web **10**(1), 9–32 (2019). https://doi.org/10.3233/SW-180320
5. Janowicz, K., Haller, A., Cox, S.J., Le Phuoc, D., Lefrançois, M.: Sosa: a lightweight ontology for sensors, observations, samples, and actuators. J. Web Semant. **56**, 1–10 (2019)
6. Khairunizam, W., Ikram, K., Bakar, S.A., Razlan, Z.M., Zunaidi, I.: Ontological framework of arm gesture information for the human upper body. In: Hassan, M.H.A. (ed.) Intelligent Manufacturing & Mechatronics. LNME, pp. 507–515. Springer, Singapore (2018). https://doi.org/10.1007/978-981-10-8788-2_45
7. Maier, J.R., Fadel, G.M.: Affordance-based methods for design. In: ASME 2003 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, American Society of Mechanical Engineers Digital Collection, pp. 785–794 (2003)
8. McNeill, D.: Hand and Mind: What Gestures Reveal About Thought. University of Chicago press (1992)
9. Mitra, S., Acharya, T.: Gesture recognition: a survey. IEEE Trans. Syst. Man Cybern. Part C Appl. Rev. **37**(3), 311–324 (2007)
10. Morris, M.R., et al.: Reducing legacy bias in gesture elicitation studies. Interact. **21**(3), 40–45 (2014)
11. Norman, D.A.: The Psychology of Everyday Things. Basic books (1988)
12. Ousmer, M., Vanderdonckt, J., Buraga, S.: An ontology for reasoning on body-based gestures. In: Proceedings of the ACM SIGCHI Symposium on Engineering Interactive Computing Systems, pp. 1–6 (2019)

13. Riener, A.: Gestural interaction in vehicular applications. Comput. **45**(4), 42–47 (2012)
14. Rosse, C., Mejino, J.: The Foundational Model of Anatomy Ontology. In: Burger, A., Davidson, D., Baldock, R. (eds.) Anatomy Ontologies for Bioinformatics: Principles and Practice, vol. 6, pp. 59–117. Springer, London (2007). https://doi.org/10.1007/978-1-84628-885-2_4
15. Scoditti, A., Blanch, R., Coutaz, J.: A novel taxonomy for gestural interaction techniques based on accelerometers. In: Proceedings of the 16th international conference on Intelligent user interfaces, pp. 63–72 (2011)
16. Stephanidis, C., et al.: Seven HCI grand challenges. Int. J. Hum. Comput. Interact. **35**(14), 1229–1269 (2019)
17. Villarreal Narvaez, S., Vanderdonckt, J., Vatavu, R.D., Wobbrock, J.O.: A systematic review of gesture elicitation studies: what can we learn from 216 studies? In: ACM conference on Designing Interactive Systems (DIS'20) (2020)
18. Wobbrock, J.O., Aung, H.H., Rothrock, B., Myers, B.A.: Maximizing the guessability of symbolic input. In: CHI'05 extended abstracts on Human Factors in Computing Systems, pp. 1869–1872 (2005)
19. Yang, L., Huang, J., Feng, T., Hong-An, W., Guo-Zhong, D.: Gesture interaction in virtual reality. Virtual Reality Intell. Hardware. **1**(1), 84–112 (2019)