Multimedia Tools and Applications (2023) 82:239–269
https://doi.org/10.1007/s11042-022-13254-8

Building a three-level multimodal emotion recognition framework

Jose Maria Garcia-Garcia[1] • Maria Dolores Lozano[2] • Victor M. R. Penichet[2] • Effie Lai-Chong Law[3]

Received: 11 December 2020 / Revised: 22 February 2022 / Accepted: 15 May 2022 /
Published online: 6 June 2022

Abstract
Multimodal emotion detection has been one of the main lines of research in the field of Affective Computing (AC) in recent years. Multimodal detectors aggregate information coming from different channels or modalities to determine what emotion users are expressing with a higher degree of accuracy. However, despite the benefits offered by this kind of detectors, their presence in real implementations is still scarce for various reasons. In this paper, we propose a technology-agnostic framework, HERA, to facilitate the creation of multimodal emotion detectors, offering a tool characterized by its modularity and the interface-based programming approach adopted in its development. HERA (Heterogeneous Emotional Results Aggregator) offers an architecture to integrate different emotion detection services and aggregate their heterogeneous results to produce a final result using a common format. This proposal constitutes a step forward in the development of multimodal detectors, providing an architecture to manage different detectors and fuse the results produced by them in a sensible way. We assessed the validity of the proposal by testing the system with several developers with no previous knowledge about affective technology and emotion detection. The assessment was performed applying the Computer System Usability Questionnaire and the Twelve Cognitive Dimensions Questionnaire, used by The Visual Studio Usability group at Microsoft, obtaining positive results and important feedback for future versions of the system.

Keywords Affective computing · Multimodal detector · Emotion recognition · Multimedia aggregator

✉ Jose Maria Garcia-Garcia
Josemaria.garcia@uclm.es

Extended author information available on the last page of the article

1 Introduction

With the first appearance of Affective Computing (AC) 25 years ago [55], and hence of automatic emotion recognition, the following set of questions arose: What channel broadcasts emotional information with the highest clarity? How are these different channels related? Are there any other mechanisms to infer someone's emotional state? Hundreds of experiments were carried out, most of them focused on finding emotion traits in facial expressions and in voices. Furthermore, some research was conducted into whether it was possible to discover how a person was feeling according to physiological data obtained from them. A later trend even studied (and still does) how to detect emotions or affective states based on behavior. For instance, Oehl et al. [51] analyzed the pressure applied on the steering wheel by drivers to detect stress and thus possible dangerous situations; Yamachi [69] studied how the movement xof the mouse could express anxiety in the subject; and Jaques et al. [31, 44] investigated how eye movement could be used to detect emotions which are important for a learning process, such as boredom or curiosity. Nevertheless, from the beginning there was another trend which looked at how emotion recognition could be greatly improved by *fusing* the results from several emotion recognizers. This type of detector is called a *multimodal emotion detector*, since it uses several kinds of emotional information to determine what emotion a person is expressing [53].

According to several emotion theories (emotions as expressions, emotions as embodiments), emotional episodes activate multiple physiological and behavioural response systems [2]. In other words, emotional expressions are revealed through several channels at the same time (face, voice, body posture, etc.), so it makes perfect sense to read these signals from those different channels simultaneously to perform a better emotion recognition. In fact, analysing input from a single modality can lead us to wrong results. For instance, facial expression and voice can be controlled by the person we are reading data from. Physiological signals cannot be consciously modified, but the information we get from them is very primal: is the person scared? Is the person excited? When a person writes something, they may not be feeling as their writing is reflecting. All these modalities provide us with information about how someone is feeling but attending to just one of them is misleading. Someone may be smiling, but their physiological signals might be low; someone might have a straight face while saying they are comfortable, but their voice can reveal they are very nervous; someone can be completely serious, and being on the edge of a panic attack; someone might be writing about being super excited about something, while being completely emotionless. All these examples have a common point: if we only pay attention to one modality of emotion expression, we might be losing information. That is the reason why multimodal detectors can be so powerful [12].

However, the mere use of a single detector of emotions in just one channel presents several challenges in itself, so multimodal (MM) systems have largely been ignored in practice. Although we can find different proposals for multimodal processing methods, multimodal frameworks, etc., there is still a gap regarding the *practical* use of these proposals. Besides, the variety of services in existence [19] for the detection of emotions from one channel or another makes it difficult for people to easily apply them, not to mention to integrate them all together.

Not only is choosing and *acquiring* or implementing these emotion recognition services a complicated task, but also making them work together presents intrinsic challenges. Having several emotion detection services working at the same time means that we have to spend additional time making them work *together*, as we have to wait until we have received data from all of them, analyse their behaviour and the data they produce, fuse these data correctly so the aggregation of the information produces more information, instead of hiding or degrading that which we already have, etc. This new challenge that arises when we try to use different emotion recognition services together is one of the reasons why the presence of multimodal detectors is so scarce in real-world applications (or scarcer than what might be expected given their recognized superiority over unimodal emotion detection systems) [49].

Another difficulty that researchers have to face when they wish to develop a multimodal system is the *lack of references*: the proposals available in the literature or in repositories such as Github are either very abstract (proposals of UML diagrams of an ideal multimodal system) or very specific (rigid systems prepared for a certain problem or necessity). This fact defines a gap in the available resources for developers and researchers who may have operational emotion detectors but who do not have the resources to develop a system which aggregates them altogether from scratch.

In the present work, we proposed a framework called **HERA (Heterogeneous Emotional Results Aggregator)** in order to try to fill the gaps in the existent literature regarding the lack of actual implementations of frameworks. In Fig. 1 we can see a simplified diagram showing the current situation regarding the integration of heterogeneous emotion detection technologies. This figure highlights the incompatibility problems derived from this integration and how the HERA framework would fit in this ecosystem to solve these issues.

HERA offers an architecture to manage different emotion detectors producing results in different formats and to aggregate these heterogeneous results into a single, richer result. As a proof-of-concept, we have modelled and implemented our proposed framework using JavaScript and the ExpressJS framework, in the form of a web server, so it is easier to integrate multimodal emotion detection in an existing project regardless of the technology that the researchers are already using. Concerning the data fusion, HERA also contains a proposal for a *three-level data fusion model* in which data is aggregated in three different phases. Firstly, each

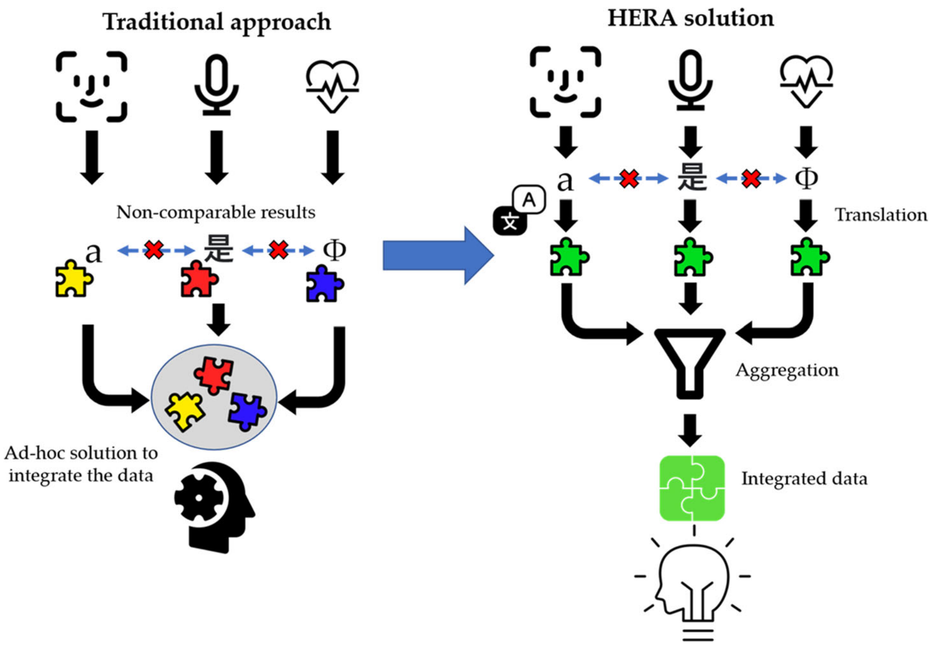

Fig. 1 Conceptual idea of the framework HERA

individual detector fuses all the data already gathered from previous requests. Secondly, detectors of the same type fuse their data together, using a data fusion strategy appropriated to these types of data. Once each group of detectors has aggregated its data and produced a single result, these data are aggregated again using a second strategy, one that fits this new context (fusing data coming from different types of affective channels and which can present contradictory values, for instance). Thus, we fuse the data in a sensible way, correcting possible discrepancies by fusing data from detectors of the same type and then being able to find new information when we fuse aggregated data considering its origin and circumstances. This aggregation process also involves a translation process, through which the heterogeneous results are translated into a common notation. For this purpose, we have used the PAD model [49] to represent the results obtained by each detector, so we store every result in the same format, regardless of the original format it had.

As it was mentioned above, this first implementation of our framework has been developed using modules and an interface-based programming approach, so it is easy to extend and adapt to particular cases. Thanks to this development approach, extending HERA is a matter of implementing new modules, without having to modify its core module at all. Also, thanks to its server nature, HERA can be deployed on any device capable of running Node (or Docker), so it can be integrated in an existent project with a minimal impact on the project's codebase.

Since the framework we propose is not only conceptual, but has been actually implemented in JavaScript as a tool for other developers, from now on we will refer to HERA indistinctly, with any of these terms: "framework", "tool", server or "system", depending on the aspect being highlighted.

It is important to point out that the key contribution of our work relies in the JavaScript implementation of the framework proposed (along with the modular structure and the data fusion framework proposed within HERA) and in the fact of offering a tangible tool to developers working in AC. While the problem of multimodality and emotion detection models has been tackled many times from many perspectives in literature before, we failed to find a proposal which actually offered an implementation for developers to easily integrate into their projects. Despite finding several frameworks which could fill the aforementioned gap in literature, they still present some deficiencies that we have tried to fix with our proposal. Since this is the goal of this proposal, the development of new data fusion algorithms or new forms of automatic classification is out of the scope of this paper.

In this paper, which is divided into six sections, we present a novel tool for integrating several emotion recognition services in the same place using a modular, scalable, fully customizable, and three-level multimodal system. In Section 2 we look at key aspects of multimodal systems, such as the kind of fusion they can use or the different systems that can be used to represent emotions. In Section 3, we present the implemented system and its different parts. In Section 4, we review the evaluation process of our multimodal system, together with the data we harvested during this process. Finally, we present several conclusions and future work in Section 5.

2 Background concepts and related works

The concept of multimodality already existed in HCI before Affective Computing came into being. Even when the technology to create these multimodal interactive systems was not yet available, HCI researchers were already testing how users preferred to interact with a system, simulating multimodality using Wizard-of-Oz systems, i.e., systems which seem to work on their own but, in reality, have 'a researcher (the "wizard") who simulates the system responses from behind the scenes' [45]. Early studies such as [28] discovered, back in 1991, that users preferred to interact through several channels at the same time. In [28], a study using a Wizard-of-Oz prototype was carried out in order to discover how the users preferred to perform a manipulation task in a three-dimensional space. Almost 60% of the participants preferred to interact with the system using both gestures and speech. This same phenomenon was described some years later in [50], where 95% of the users involved stated a preference for multimodal interaction.

When Affective Computing was defined for the first time in 1995 [54], the existent signal and data fusing knowledge became the perfect breeding ground for multimodal affective systems, i.e., systems that fuse data coming from different emotional channels (facial expression, voice tone, body gestures, electrodermal activity, etc.) to obtain a more accurate measurement, to correct anomalous samples from defective sensors, to detect affective states or behaviours that cannot be detected by attending to just one emotional channel, etc. In 1998, Chen et al. [8] implemented De Silva's proposed algorithm [61] to classify emotions expressed in 36 clips of synchronized audio/video, hence building the *first actual multimodal human emotion recognition system*. This marked the beginning of a new line of work in Affective Computing: *Multimodal Emotion Recognition (MMER)*.

This line of research has enormously advanced over the last years, taking advantage of the power that machine learning and big data offers to AC, what has led to the appearance of very different proposals to produce multimodal detectors using machine learning [11, 47, 63, 70–72]. While these works provide interesting approaches, the key contribution of this paper is to offer a software tool addressed to developers so that they can easily integrate these trained detectors into a bigger system, and this is the reason why the framework HERA does not include any type of machine learning algorithms in its implementation. Nevertheless, HERA's architecture has been designed to allow developers to ingrate their own automatic classifiers and models as local emotion detection services which behave just as the remote ones. In addition, although HERA does not use any form of machine learning to integrate data from different sources at this time, this could be included in future versions.

In the next subsection, we review how the data fusion is carried out, the different levels at which it can be performed and how these data can be represented. Lastly, we review the situation of multimodal affective technology today.

2.1 Types of data fusion

One of the inherent challenges in Multimodal Systems is *data fusion*. While being one of the key assets of MMER systems, data fusion is also one of the most difficult aspects to implement. In order to obtain added value from aggregating data from different sources, it is mandatory to combine them properly. In other words, you cannot just, for instance, calculate the average of all the data your emotion recognizers produce, but you must fuse these data following a fusion strategy. In the field of data fusion, there are *several fusion strategies*, each one taking place at a different point of the data processing flow [4, 43, 55]:

- **Raw-level fusion**. A system performs a raw-level data fusion when it works with the data at their purest level, i.e., as they are produced by the different sensors. Arrays of positions of each Facial Unit, numbers expressing the skin conductivity or temperature, the heart rate, an audio file: these data are raw data. One drawback of this kind of fusion is that sensors must be measuring the same magnitude in order to be able to fuse their outputs. You cannot fuse a number expressing someone's heart rate with a number expressing the frequency of someone's voice at a given time, but you can fuse the two measurements of two different heart rate sensors to reduce the Mean Squared Error (MSE).

- **Feature-level fusion**. When a system extracts the features expressed by the raw data and combines them, it performs a feature-level fusion. A feature can be some signal's mean during a certain window of time, the name of a posture or a facial expression represented by an array of mark positions, etc.
- **Decision-level fusion**. A system performs a decision-level fusion when it combines the final results produced by each classifier, that is, each emotion detector, these usually being expressed using the six basic emotions (joy, sadness, fear, anger, disgust and surprise) [15] together with percentages (joy 38%, sadness 10%, etc.) indicating how much each emotion is present in the data analysed, although there can be far more affective states identified. On this final level, we retrieve these affective states, together with their corresponding score, and compare them all together to obtain a high-level, richer, and more accurate result. Decision-level fusion is the most commonly used approach for multimodal HCI [2]. The main benefit of this approach over the other ones is that, at this point, all the results are (almost) on the same page, regardless of the format they had when they were first produced.
- **Hybrid fusion**. This approach is actually a combination of feature-level fusion and decision-level fusion. This type of approach arises when we fuse the features of two detectors, classify the results using an automatic classifier and then fuse the result of that classifier with the results produced by another emotion detector (another classifier).
- **Model-level fusion**. This approach is not exactly a data fusion approach since the data are not actually fused but fed into another classifier which has been trained to classify multimodal information.

Since HERA does not implement the emotion detectors, but just requests the results from them, we will always work on the decision-level, using the PAD format that we define below.

2.2 Emotion representation

Another defining feature of emotion recognizers is the way they represent the detected emotions. The definition of emotion itself was already a hot topic of discussion in antiquity, and this discussion is still fully alive today, a fact which has led to the coexistence of 92 emotion theories [34]. Despite this debate about what emotions are, how they are produced or how they are felt by a person, when it comes to representing them using values, there are several approaches which are currently being used [35, 40, 60, 66]:

- *Categorical models*. Also called *discrete emotion representation models,* these representation models use categories in the form of labels to represent emotions. With this type of models, a detected emotion is expressed in terms of the categories we are using. One of the most extended categorical models is the one proposed by Paul Ekman [15], which uses the famous six basic emotions to express an emotion. It is also very common to see the "neutral" category added to this model to represent the absence of emotion. Although these models are easy to understand, it may be difficult to choose the correct categories to accurately represent an emotion.

- *Dimensional models*. In order to alleviate the inflexibility of categorical models, dimensional models were proposed. Instead of having fixed categories of emotions, these models use dimensions which express aspects of an emotion, so what was expressed using a fixed set of categories and a set of percentages is now expressed as a point in a dimensional space (usually 2D or 3D). One of the most widely used dimensional approaches is the PAD (Pleasure, Arousal, Dominance) model, which was proposed in [46]. In this model emotions are expressed in three dimensions, each one in a range of values that usually goes from −100 to 100 or − 1 to 1. These dimensions convey how pleasant (100) or unpleasant (−100) a person's feelings about something are, how aroused (100) or relaxed (−100) a person feels and how dominant (100) or submissive (−100) a person feels while experiencing an emotion. In Fig. 2 we can see a graphic example of this model, in which fixed categories representing emotions are now sectors in a 3D space. This type of models allows us to work with machine learning algorithms in an easier way and also provides us with a form of representation that can become a layer of abstraction for different categorical models, as we will describe later in this paper.
- *Componential models*. This type of models could be considered categorical models as well, since they are built on fixed emotion categories, but they define another dimension of information, defining hierarchies of emotions (Plutchik model, OCC model) [35] or even adaptations for specific fields of research (Hourglass of emotions [5]):

After reviewing the different models, the classic PAD model was chosen as the standard model to represent emotions in our proposed framework. The reason behind this decision is twofold. Firstly, categorical and component models assume the orthogonality of emotions which, in contrast, are overlapping. In comparison, the dimensional model better accommodates the inherent overlapping nature of different emotions. Also, when it comes to

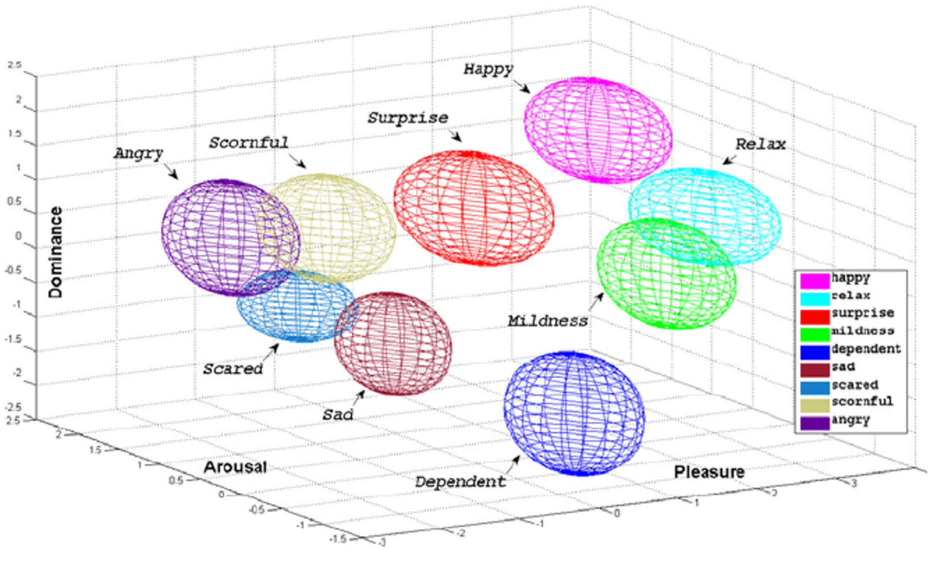

Fig. 2 PAD space [74]

operationalise these models, the dimensional model is easier to implement as well. Secondly, dimensional models help us overcome the heterogeneity of formats commonly used in the industry to represent results. Even when most of the emotion detection services which exist nowadays use a categorical system to represent emotions, they are all different. Hence, developers find themselves facing a problem of heterogeneity of results: they have different services producing results based on the same data, but each service expresses its results in a different way, with a different set of tags, using different ranges of values, what makes it impossible to merge those results directly. To provide a solution to this problem, we have included an aggregation mechanism in the proposal of our framework based on *translating* every result to a standard format. For this purpose, we have chosen PAD, which allows developers to express an emotion as a point in a 3D space. By homogenising all the results produced by each detector, we can aggregate them all together, producing a richer and more solid final result. In following sections, we will see how it was deployed in HERA.

2.3 Research challenges of multimodal emotion recognition systems

Multimodal systems have been the focus of much work in the last few years. New mechanisms for reading emotions have been developed, classification tools have become more powerful and more accessible, and many public databases with multimodal affective data to work with have been created [49, 55]. Over these years, the initial challenges have been met, but new, more complex challenges have arisen. Some of these challenges are explained below [4, 14]:

- *Limitation of the available data to train emotion detectors*. Since emotion detectors are just automatic classifiers, they rely on the datasets they are fed to be able to perform appropriately. For a long time, the shortage of multimodal affective databases limited the amount of works that could be done in this field, even though some multimodal databases have appeared in the last years (e.g., DEAP [36], RECOLA [57], BP4D+ [73]). There is also a problem with the use of prepared datasets, and it is that models trained with these data do not perform well in spontaneous environments [14, 59, 52], although there are proposals to fix this issue [33, 37, 73].
- *No context information being considered***.** MM systems work better when they consider an affective state in the context of previous ones, since this makes it possible to better consider new emotional responses. However, even when there are proposals to consider this information, contextual information is sometimes overlooked [49].
- *Irregular ecosystem*. Unless you have access to a set of trained models to detect emotions in different channels, you are forced to resort to existent, publicly available emotion detectors (assuming you do not have the time to learn how to develop your own model, to gather the media necessary to train it, and to improve it to reach an acceptable degree of accuracy). The ecosystem of the different detectors available over the Internet is quite extensive. For instance, you can find free open-source detectors which are offered via TensorFlow models, services offered on payment of a fee which let one analyse media resources through HTTP requests, etc. The problem here is that each service works in a specific way and has its own advantages, which makes the learning and integration process very slow at the beginning.

Even though these are some of the challenges researchers and developers working in the field of AC have to address, the irregularity of the ecosystem is the most difficult to overcome, especially when developers try to make different services work together. In order to solve this problem, we have developed HERA, a framework to integrate different emotion detection services together that only demands the developer to complete a small template of code in order to integrate a new service into a bigger ecosystem. In Section 3, we will describe how HERA is organised and how it is used.

As it was stated above, the main contribution of this proposal does not rely on the training of an automatic classifier or any other model, but in the development of a framework to integrate very heterogeneous technologies together. This approach does not use machine learning algorithms, as the goal is to merge results based on empirical knowledge.

2.4 Multimodal emotion recognition systems

As we mentioned above, the concept of multimodality has been around for quite a long time now and not just in theory. Multimodal systems have been deployed and tested in fields such as education [1, 13, 20, 27, 29, 32, 68], healthcare [6, 7], marketing [5, 30, 38, 42, 58] or gaming [22, 59].

However, most of the systems developed in these cases were completely ad-hoc, that is, they were developed thinking specifically about the use case they were going to be applied in, there being very little effort on the part of the academic community to develop more abstract systems or frameworks for developing multimodal emotion recognition systems.

Prior to the development of the HERA system, we reviewed the existing literature to study possible frameworks or systems that were created for this purpose. The related works we found are briefly described next. Gonzalez-Sanchez et al. proposed an agent-based software architecture of a generic multimodal framework using design patterns [23], a proposal that they implemented and tested in several real scenarios with different types of inputs [24].

Zheng et al. proposed a multimodal framework specifically to recognise emotion from EEG and eye movement [75].

Maël Fabien's team, in partnership with the French Employment Agency, developed a multimodal emotion recognition system in the form of a web app (using Flask) which analyses facial, vocal and textual emotions. This system is meant to be used to analyse the face, voice and written text of a person being interviewed for a job offer [17].

Myeongjang Pyeon started the development of a web-based interactive multimodal emotion recognition framework but left it unfinished and undocumented [56].

Alepis and Virbou [1] developed a proposal which is the closest to our proposal we have been able to find, although it is designed specifically for mobile phones and handheld devices.

The W3C Multimodal Interaction Working Group has proposed a framework to develop multimodal systems, that is, systems that allow the users to interact through different interaction channels [65]. Even though multimodality, when it comes to interaction, does not completely correspond to multimodality when it comes to emotion detection, the Multimodal Interaction Framework (MMI) presents a similar architecture to the one proposed in this work. Among the similarities with our approach, this framework (MMI) is also based on different components in charge of receiving input data to produce a result, in this case, multimodal interactions to produce a concrete action on the system. Nevertheless, in the MMI framework, the second level component performs semantic interpretations of the multimodal actions, whereas in our proposal the second level component performs the translation to a common format (PAD) of the different emotion recognition sources (facial, voice, etc.). Regarding emotions, W3C has also proposed a notation to represent them using XML, based on the different emotion models that we presented above [64].

Even though a lot of different implementations of multimodal systems can be found on Github, most of them cover specific use cases instead of giving the structure or tools necessary to create one of those systems. The HERA system aims to fill that gap. In the next section, a description of the system can be found, together with an explanation of the decisions that were made in order to develop it.

3 System description

HERA is a three-level multimodal emotion detection framework designed to manage different emotion detectors in one place, detect emotions in multimedia resources using these recognizers and aggregate results from each emotion detection service. Essentially, HERA (from now on also referred to as the server, the framework, the tool, or simply the system) works like a proxy for other services, these being a third-party emotion recognizer, a device streaming affective or physiological data, etc. Instead of communicating with the services directly, having to spend additional time synchronizing them, performing the authentication process that they may need, discriminating between them depending on the type of resource one wishes to analyse and so on, one can simply tell the system which services one wishes to use and provides it with the settings that those detectors need. When this setting is finished, HERA will simply wait for the analysis requests to arrive, processing them according to the type of media that the request contains and the kind of affective information that the system should seek in this media. HERA will automatically redirect the analysis request to the proxies that can handle these requests, later storing the results that each service produces.

It should be noted that this proposal is not a closed ecosystem with a limited number of services. Our project has been developed following an *Interface-Based Architecture* in order to increase the modularity and maintainability of the system. This means that every detector implements the same interface (see Detector subsection), so adding more detectors (proxies) to the system is a matter of implementing this interface for each of them.

In the next subsections, the HERA ecosystem is described in detail.

3.1 Development approach

This first version of HERA was developed in the form of an API REST using Node.js and the framework Express.js [16]. The development of this framework as an API was based on these principles:

- *Interoperability*. Developing the framework as a module of a specific technology or programming language would have restricted its access to possible users, since researchers or developers working in AC may already be working with a certain technology or framework. If this technology is not compatible with the technology used to develop our tool, or the developers do not know how to use it, they would not be able to use our system, making our tool available only to developers who are familiar with this technology or those that could integrate it with the technology they are already using. In order to facilitate access to HERA and expand our target user group, we developed this proposal as a *web server* (which can also be containerized using Docker), taking advantage of the universality of HTTP communications, which can be implemented with most of the technologies available nowadays, so HERA can be used in any project regardless of the underlying technology.

- *Independency*. When developers are working on an application that involves emotion detection, they have to use valuable time and resources in coding all the aspects related to emotion detection and processing, polluting the codebase with code that does not actually do anything in the app, but which manages aspects that it does not manifest in the final application view. With this approach, i.e., keeping all the emotion processing logic aside, on a separate server, developers just need to take care of capturing the media that they wish to analyse, communicating with HERA and handling the results that the system produces.
- *Easy to extend*. In order to make this framework an appealing multimodal system for developers to use, we need to design it so that it is able to work with many different services. However, we cannot code all the needed functionality to work with every possible device or service in existence. Even if HERA includes some well-known detectors (that is, their corresponding interface implementation), such Face++, which has already been implemented, developers may need to change how the system works with them, or they may need to communicate with a brand new emotion detection service, so this framework needs to be *easy to extend*, using the mechanisms provided by JavaScript in this case so it is simple to add support for new devices.

Based on these three precepts, the HERA system was developed as a web server with Node.js, using the Express framework to build the REST API infrastructure. Thus, our framework is designed to be deployed on a Node server and used as part of a bigger system, as the diagrams in the next figures (Fig. 3 and Fig. 4) show.

In the next subsection, the different artifacts and components of HERA are described in detail.

3.2 Architecture

In Fig. 3, we can see a standard UML deployment diagram which shows our framework in a simulated real context. As explained above, HERA is actually a Web server built with Node.js and ExpressJS, so it can run on a server or in a Docker container, for instance. Once the server is deployed, developers using this framework would communicate with it using HTTP requests, which work with the TCP/IP protocol. In the right-hand part of the diagram, we can see several services as examples of information sources for which HERA could act as a proxy. The first one represents a Generic Emotion Recognition Service, usually offered as an API which requires an access token that the API owners must issue for you previously. It is

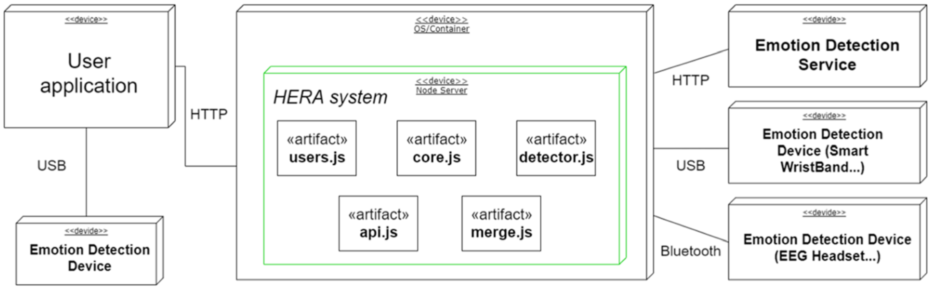

Fig. 3 Architecture - Deployment diagram

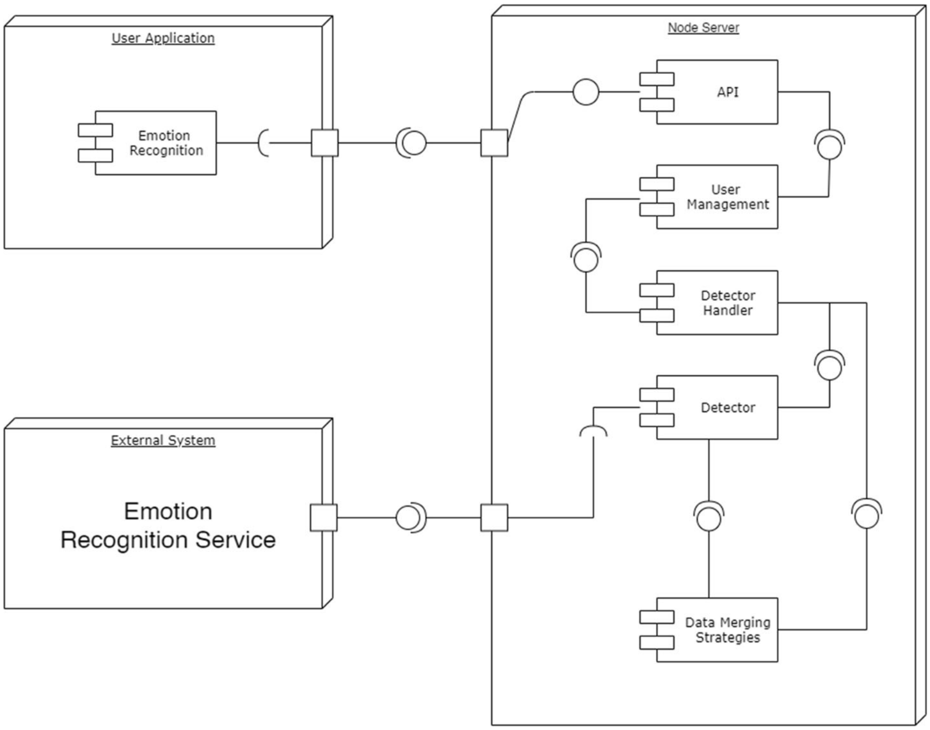

Fig. 4 Architecture - Component diagram

very common to pay for this kind of services to integrate emotion detection in an application, since the companies who makes them offer well-trained emotion recognizers as a service. These services usually work over HTTPS, receiving an item of multimedia which holds affective information (a picture or a video, a sound file, an excerpt from a text, etc.) and returning a set of results in a specific format, usually a categorical approach (the emotion detected is expressed in terms of Ekman's six basic emotions [15]). However, this service could also be a local server providing an emotion detection service implemented by the developers themselves using the *scikit-learn* package, for instance. The other three nodes represent physical devices that could be attached to the device running our server or to the User application directly. These physical devices could communicate their results via HTTP (through an intermediary system) or directly to HERA, using more direct protocols such as USB or Bluetooth. In the case of devices measuring physiological variables, the results would be *rawer*, requiring some processing on the server before they can be translated into a more understandable format, but we will look at this topic in the next subsection.

Returning to Fig. 3, we can see that the system is made up of five main artifacts:

- *api.js*. This file holds all the API-related logic. Even though there are a couple more files related to the Express framework and the API logic, the declaration of the different endpoints that the developers reach from their external applications can be found in *api.js*.
- *users.js*. In order to be able to keep track of requests coming from different applications (or from the same application but for different purposes), HERA implements sessions using the *users.js* artifact. This artifact contains logic to create new users, which are actually

sessions, using cookies, to add information (detectors) to these sessions, to refresh sessions and to delete them when a certain timeout is reached.

- *core.js*. This artifact contains the *core* functionality of the API, that is, the management of the different detectors, all of them organized in different groups depending on the affective channel in which they can detect affective information and the orchestration of the subsequent data fusion (after performing the analysis of media resources).
- *detector.js*. This artifact contains the key building block of the API: an abstraction for a device or a system providing affective information. For each emotion information provider, whether it be a physical device, a paid API offered by a third party or an automatic classifier trained to identify emotions running locally, HERA creates a *Detector* object, which will act as a proxy for this service, handling the requests that usually would be sent directly to it. Following the principle of keeping the system modular, each detector implements part of its functionality as an external module that is imported when the Detector object is created. We will review this in Subsection 3 of this section.
- *merge.js*. This artifact contains the different strategies that can be used to fuse the affective data. These strategies receive a list of PAD triplets, which come from the results produced by the different detectors and return a single PAD triplet. In addition, since the strategies also follow a *modular* approach, they can be defined by following another format or adding more elements (or properties) to the final result, creating new information based on that which we already have, and this is one of the main assets of HERA.

In Fig. 4 we can see a component diagram showing how the different artifacts work together to fulfil the users' requests:

- The *api.js* artifact from Fig. 4, here represented by the API component, exposes the API endpoints that users reach using HTTP requests. This API has been developed using Express.js, and the different endpoints can be reached using GET and POST requests. In the following subsections, we will specify how many endpoints HERA offers and what information they need to operate.
- As stated above, the system uses sessions to tell incoming requests apart and to store detector handlers, so the API component needs the functionality exposed by the *User Management* component, which corresponds with the *users.js* artifact.
- Likewise, the User Management component uses the logic offered by the *Detector Handler* component (implemented in the core.js artifact) to attend to the users' requests. This component contains logic to add new detectors to the current session, to organize them into groups depending on the input they can handle, to forward requests to the corresponding group and to fuse the data available in every detector.
- In order to use a detector in this framework, developers must have previously implemented a set of methods so this detector can be handled by the system (although HERA already includes several Detectors that are implemented in its repository). These methods constitute the *Detector* interface. In the following section we will see this interface in detail.
- Finally, both detectors and the Detector Handler of a session need access to the data fusion strategies, which are managed by the *Data Merging Strategies* component. Each detector use strategies to merge their own results into a single value (a PAD triplet), while the Detector Handler uses strategies at two different levels: first, it fuses the results coming from each category of detector, and afterwards it fuses together the results produced in this previous first fusion. We will see how this process works in the following subsection.

In the next subsections we will review how the data fusion is performed, how the detectors are integrated in the system and what would be the regular workflow of the framework.

3.3 The detector interface

As we mentioned at the beginning of Section 3, in order to be an effective tool, HERA must be able to work with all types of affective information providers, whether it be an API receiving requests over HTTPS and returning JSON objects with a certain content, a physical device streaming an array of data via a USB port, or a device communicating with the system through Bluetooth packages. Not only must the system deal with data in very different forms, but also with the idiosyncrasies imposed by each detector (authentication processes, gathering of data, etc.). In order to deal with this range of possibilities, HERA uses the D*etector* class as an abstraction layer, treating every different detector as a Detector object. Every detector has an *id* (which is usually its name), the *kind of content* in which it can find affective information (face, voice, body, etc.), a list of the media formats it supports (images, video, audio tracks, etc.), a *delay* property which indicates the average response time of the detector, the *address* where the detector delivers its service (if any), and two *lists* storing the results produced, one with the original results and one with the corresponding PAD translations. In an object-oriented (OO) programming language, Detector would be an abstract class and every specific detector would be a new class which would extend it, adding the functionality that this new detector could need. However, since basic JavaScript does not support abstract classes directly, HERA implements it using modules: every detector is implemented as a *module which must export three key functions*, which are associated with a Detector object in runtime. These three functions are:

- *initialize.* This method takes care of any initialization tasks that your detector needs. For instance, if this detector is a proxy for a third-party service offered via Internet, developers may need to obtain an *auth* token first. If this detector is a proxy for a bluetooth wristband, developers may need to put their discovery and connection code in this method. If developers do not need any initialization, they should just return a resolved promise (*Promise.resolve*).
- *extractEmotions.* This is the main method of every detector, the one in charge of performing the actual emotion detection, whether this be forwarding a media resource to an API over the internet, reading RAW data from a sensor, etc. This method receives three parameters, namely *context*, *media* and *callback*. *Context* is the environment from which the method is called, usually a Detector object. This is done so that these methods can be specified in a separate file, but results can be stored correctly in the corresponding Detector object. *Media* is a path to the file which holds the affective information, if there is any. If there is no file (maybe the detector reads some RAW data from a certain port or socket in this method), this parameter will stay unused for the sake of the aforementioned interface-based programming paradigm. The final parameter, namely *callback*, is an optional callback to handle the retrieved data, in case further manipulations are needed.
- *translateToPAD.* This method is one of the keys of this proposal, since it is the part of HERA that allows us to integrate heterogeneous emotion detectors. In order to actually be able to treat all the detectors in the same way, we need to translate all the results they produce to the same language. The chosen language for this task is the PAD format, which we introduced in Section 2. This method must implement the transformation of the

incoming results, so the affective results (whatever their format is) received by a detector are translated into a triplet of three numbers, each one being a value between −1 and 1, which stands for how negative or positive the expressed emotion is, how relaxed or excited the person is, and how passive or dominant that person feels while experiencing that emotion, respectively. In Fig. 5, we can see an example of the type of transformation this method does.

Regarding the organization of these different modules, it is highly recommended to keep them organised in folders according to *categories* (a `face` folder for modules of detectors which support facial-expression-based emotion detectors, a `voice` folder for voice-based emotion detectors, etc.), although this is not actually necessary since the path to a given detector's functions is fully specified in the request for setting up the different detectors.

In the next subsections we will see how this uniformized data is integrated and how these methods are associated with their corresponding detector.

3.4 Data fusion

In Section 2 we introduced one of the key aspects of a multimodal system, namely the fusion of different types of data coming from different sources of affective information. Since multimodal systems gather information with different formats coming from different sources, they need to perform a transformation and/or an integration task of this data in order to produce a coherent final result. Therefore, the first thing to do when a new result is produced is to translate it into its corresponding PAD value. Every detector keeps a list of all the results they have produced, so the data fusing challenge is *to turn a set of lists of PAD results into a single result while considering the context of all the detectors*, as some of these results may have been produced by *different detectors of the same type* analysing the same media file (e.g., two face-based emotion detectors analysing the same picture), by different detectors analysing media resources with *different formats* (e.g., one result produced by a face-based emotion detector and one result produced by a voice-based emotion detector), by detectors with a very different rate of result production (e.g., one detector has produced 3 results and another detector has produced 1.000 results), etc.

We have designed a fusion framework which fuses data three times, one per level:

- The first level is made up of the specific detectors that have been integrated into HERA, that is, emotion detection services which had had their Detector interface implemented. In other words, we say that an emotion detection service has been integrated into HERA

```
//Raw data, not necessarily following Ekman’s six basic emotions
const rawData = {
     “happiness”: 0.95432,
     “sadness”: 0.00000,
     “surprise”: 0.001245,
     “anger”:  1e-4545,
     “disgust”: 1e-135
}
const padData = translateToPAD(rawData);
//padData = [ 0.6584, -0.3207, 0.1249 ]
```

Fig. 5 Example of translateToPAD

when we have created a file which contains the three methods mentioned in the previous section to communicate with said detection service and to translate its responses. The fusion in this level is done by telling every detector to aggregate its results from the current session into one result. At this point, each detector has turned its list of results into a single PAD triplet.

- The second level is made up of the *categories of detectors* detecting emotions in the same modality. HERA organises detectors by placing those which can detect emotions in the same modality under the same group, this way, when a request of emotion detection over this category arrives, HERA forwards this request to each detector capable of fulfilling it. The fusion at this level is done by aggregating the results produced by the fusion in the previous level together, always grouping these results by category. At this point, HERA has a PAD triplet per category. Since the detectors of each category have all produced a result for each request received, aggregating all the results of the same category into one helps to decrease small discrepancies.
- In the third level of this data fusion framework, HERA aggregates the output of the previous level into a final single PAD triplet. The aggregation strategy at this level might be more complex than the one used in the previous levels, since now we might have to consider different weights for the different categories, metadata attached to the PAD triplet produced by each category, etc.

Figure 6 shows a sequence diagram that models the data fusion process throughout the whole system:

- The user sends a merging data request to HERA through the *results* endpoint (Fig. 7). As it is stated in the documentation, this request demands three parameters: a list of channels

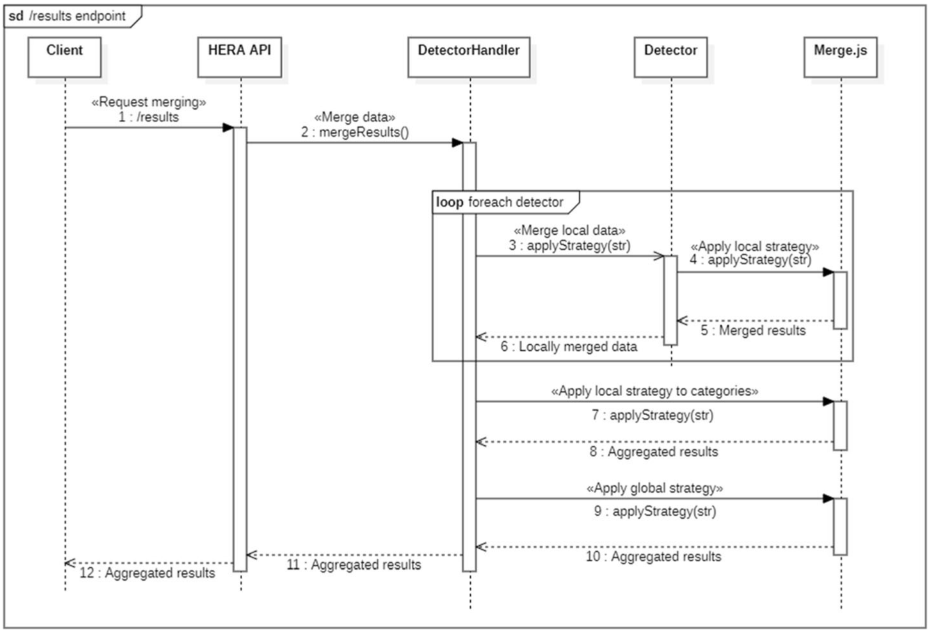
sd /results endpoint
Client
HERA API
DetectorHandler
Detector
Merge.js
«Request merging» 1 : /results
«Merge data» 2 : mergeResults()
loop foreach detector
«Merge local data» 3 : applyStrategy(str)
«Apply local strategy» 4 : applyStrategy(str)
5 : Merged results
6 : Locally merged data
«Apply local strategy to categories» 7 : applyStrategy(str)
8 : Aggregated results
«Apply global strategy» 9 : applyStrategy(str)
10 : Aggregated results
11 : Aggregated results
12 : Aggregated results

Fig. 6 Fusion strategy diagram

```
post( {
    url: 'http://localhost:3000/api/v1/results',
    body: {
        channelsToMerge: [ 'face' ],
        localStrategy: 'default',
        globalStrategy: 'default'
    },
    json: true
} )
```

Fig. 7 /results request endpoint

that are going to be merged, the name of the strategy that the API has to use to aggregate the local data and the name of the strategy that the API has to use to aggregate the data of detectors from different categories. In Fig. 7, we can see a request to this endpoint using JavaScript and the *request* package. In this snippet, we are asking HERA to fuse all the data from the "face" and "voice" channels. In our tool, a channel is a group of detectors which can extract emotions from a certain modality. For instance, the channel "face" a list of detectors which can read emotions from the face of a person (facial expression). The channels' names can be fully customized by the users of HERA.

- When HERA receives a "/results" request, it forwards the request to the Detector Handler component of the API, which keeps the detectors organized in *categories*. Since each detector on each category holds a list of results in PAD format, HERA must combine all these results into one. To do so, HERA will request every Detector object to aggregate its own results using the local strategy *localStrategy* that was attached to the request.
- Each Detector object keeps two lists of the results produced in the current session: one list contains the results in its raw form, while the other one contains the PAD version of those same results. When the *mergeResult* method is invoked, the Detector calls the *applyStrategy* function from the *merge.js* module, which applies the fusion strategy specified in the aforementioned request to a list of PAD results. The merge.js module has all the implemented strategies organised in a dictionary. In the context of HERA, a *strategy* is a function which receives a list of triplets, each triplet representing an emotion in a PAD space, and returns a single triplet. In Fig. 8, we can see an example of a strategy, which performs an average operation over the different coordinates of the PAD values of the list. Since HERA is implemented in JavaScript, we can also take advantage of the capabilities of the language to add extra properties and metadata to this final result.
- Once every detector has aggregated its data using the *localStrategy* strategy, they return this single PAD data to the Detector Handler object. At this point, HERA has a PAD triplet per Detector, being these detectors organised in categories. The next step is to aggregate

```
default: function( tripletsArray ) {
    const pleasure = tripletsArray.map( ( element ) => {
        return element[ 0 ];
    } );
    const arousal = tripletsArray.map( ( element ) => {
        return element[ 1 ];
    } );
    const dominance = tripletsArray.map( ( element ) => {
        return element[ 2 ];
    } );
    return [ mean( pleasure ), mean( arousal ), mean( dominance ) ];
}
```

Fig. 8 Strategy example

the PAD triplets from the same category into one. To do so, HERA will repeat the operation from the previous step: for each category, the Detector Handler object will invoke the *applyStrategy* method from merge.js, passing a list with as much results as detectors objects there are in that said category. Because of the similarity of results, the same *localStrategy* strategy from the previous step is used.

- Once HERA has obtained a PAD triplet per category, the Detector Handler will call the *applyStrategy* function again with these triplets as arguments, using the *globalStrategy* strategy this time. Usually, the *localStrategy* strategy will be an easier operation since the data to aggregate is much more similar, while the *globalStrategy* strategy will probably consider the context of the each triplet, possible metadata added to the triplets, etc., although the same strategy could be used in the different steps of the process.
- Using the *globalStrategy* strategy, a single PAD triplet, which could possibly hold additional metadata, is produced. This final result is sent back to the user, finishing the merging workflow.

3.5 System workflow

After having reviewed the different parts of the proposed framework, now we will see how all these pieces work together (Fig. 9). Once the server is running (deployed on a Node server and attending to requests in a specific port), the workflow is as follows:

- *Requesting an access token*. As we mentioned above, HERA uses sessions to distinguish where the requests are coming from and/or to group different sets of detectors. These sessions are implemented using cookies, which store an access token that the server provides us with. This access token must be included in any request directed to the server in the future, otherwise we will not be able to access its functionality, and it will warn us about the absence of an access token. In order to obtain our access token, we must send a request to the API's root (endpoint "/"). In response, the system will create our session on the server and send us in return our unique access token stored in the *userId* cookie. This id allows us to communicate with the rest of the endpoints. Trying to communicate with these endpoints without including this unique id in the request will return an error ("`Session wasn′ t initialized. Send request to "/" first`"). This id is linked to a `user` object which will store the detectors' proxies, the data returned by these, etc.
- *Setting up the server*. Once we have our access token, we can communicate with HERA so that it creates a proxy for each detector we wish to use in the current session. To do so, we must send a request to the "`/init`" endpoint, attaching to said request the information about the detectors we wish to use. This endpoint will link a `DetectorHandler` object to our user session and will create a `Detector` object for each detector specified in the request (this setting information can be included directly in the request or stored in a file on the server if we are always using the same detectors, in which case we must specify the route of this file in our request). Our system also supports a benchmarking process for each detector: if we provide each category of detectors with a folder containing resources that those detectors can analyse, setting up a detector at this endpoint will start a benchmarking task, in which all the resources from the corresponding folder will be analysed using this detector. From this task an average response time is calculated so that we can later filter detectors out depending on their response time.

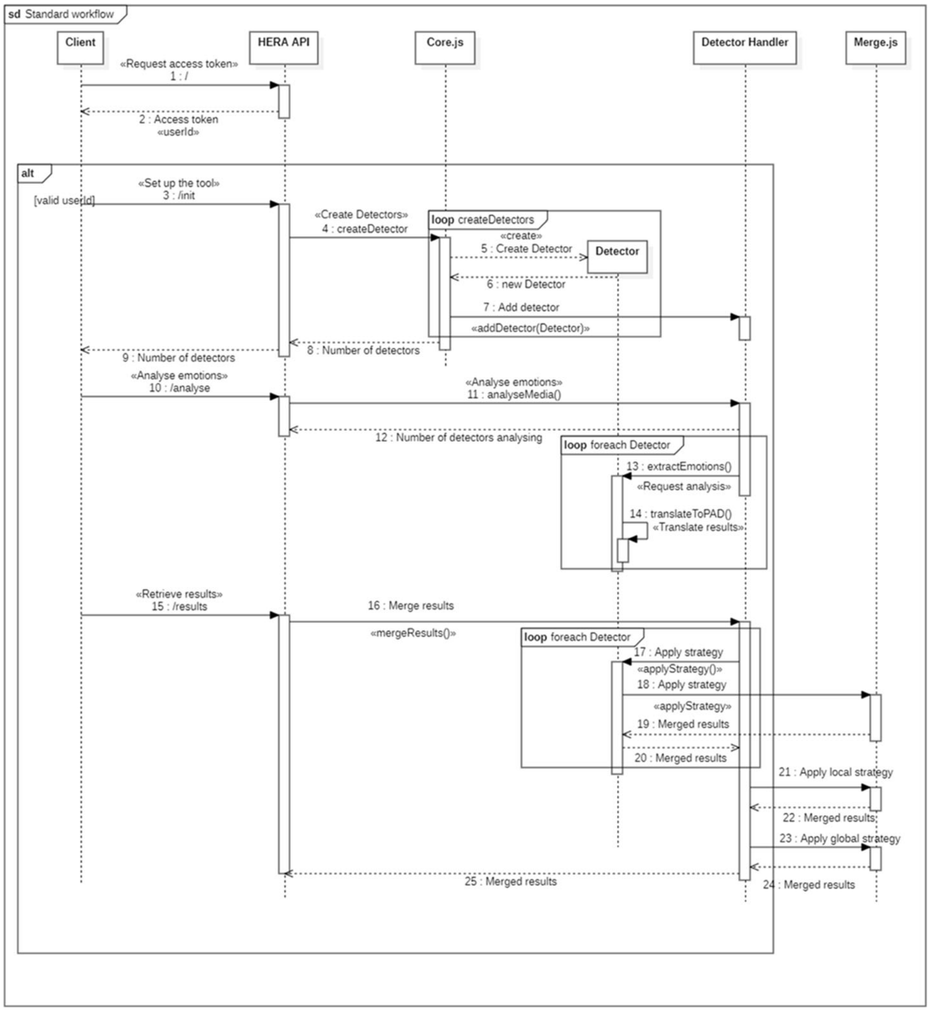

Fig. 9 System workflow

- *Requesting a media analysis looking for emotions*. After calling the previous endpoint, HERA is ready to handle our emotion analysis requests. In order to request an analysis, we must reach the "`/analyse`" endpoint, sending to it the resource we wish to analyse, if any (the server may need to pick this resource itself from a socket or a USB port), the type of resource it is and how the affective information is coded, that is, whether it has to analyse the resource by looking for faces, for voices, etc. HERA uses this information to forward the requests to the detectors capable of taking this request, all of them organized in categories under our `DetectorHandler` object. This endpoint uses the `extractEmotion` function of each detector to retrieve the corresponding results, and the `translateToPAD` functions to store the results. The response to this endpoint does not contain the results, since they must be requested through the next endpoint.

- *Asking for results*. Finally, once the detectors have produced the required results, developers would poll on the HERA server to ask for the integration of the produced results using the "`/results`" endpoint. In order to communicate with this endpoint, we have to indicate what data merging strategies HERA should use to aggregate the results and what channels should be aggregated together. In this way, this endpoint can be used to retrieve the RAW results for a single channel, a PAD triplet with additional properties produced by aggregating all the existent results, etc.

The framework also includes another endpoint which developers can use to filter detectors out once they have been created. Thus, developers have an additional tool to control their detectors.

- *Filtering detectors*. After creating the detectors, we may wish, for instance, to filter out detectors which are too slow, or maybe we have lost one of our affective channels and we wish to remove that category of emotion detectors. To do so, we can send a request to the "`/setup`" endpoint, indicating which categories we are actually going to need or a threshold that the average response time of each detector must satisfy (to remove detectors which take longer than that threshold to complete a request). As per the rest of the API, this endpoint can be modified to add options and filters.

4 Framework evaluation

An evaluation of the system was conducted with seven software developers, all of them with previous experience in the development of applications and in the use of APIs. In this section, the different aspects of the assessment are presented.

HERA was developed under the hypothesis that separating the emotion processing logic from the main application, and providing a modular data fusion framework would help developers to integrate emotion detection in their applications more quickly and more easily, and also give them the possibility of further exploiting the affective information they could gather. In this evaluation we seek to prove this hypothesis.

4.1 Participants and context

For this experiment we recruited seven software developers aged between 22 and 35 years old, all of them with experience in the development of server-side applications and in the use of third-party APIs. Any of them had previous experience with emotion-related technologies, although two of them were familiar with AC concepts. We decided to recruit developers with no previous knowledge on emotion detection technologies so that they could focus on the process of integrating technologies together from a beginner's perspective. According to Jakob Nielsen, even five participants are sufficient to detect 85% of the usability problems in a system [48]. Although seven participants may seem a small number, as the main goal was to assess the validity of the system by specialists in the field of software development, we considered more important to have representative participants than to recruit many people with similar skills repeating the same tasks.

The setup of the experiment consisted of two devices. The first device was a web server deployed on the Amazon Web Services (AWS) platform, which is accessible at all times. This platform was responsible for keeping a HERA server instance running. Evaluation tasks involving HTTP requests to the system were carried out by directing those requests to this server. The second computer was the computer used by each developer participating in this test to perform the corresponding evaluation tasks. Prior to the evaluation, each developer was asked what equipment they would rather use, and all of them decided to use their personal computer, even though they were also offered a computer which had all the software they might need to complete the evaluation tasks. The HERA server was launched using Node.js v12.14.0 and Express v4.16.0, and the evaluation tasks were performed using Node.js v12.14.0 and two different Python versions (2.7 and 3.7).

4.2 Evaluation metrics

After reviewing previous studies on API usability assessment [2, 10, 26, 62, 67], especially the API evaluation framework used by The Visual Studio Usability group at Microsoft [9], we decided to adopt a combination of the Twelve Cognitive Dimensions Questionnaire and the Computer System Usability Questionnaire (CSUQ) [3, 25, 41]. The Cognitive Dimensions framework was proposed in 1989 as a framework to evaluate the different aspects (dimensions) of a programming language, but in the last few years it has been applied to specific products, such as APIs, for instance. These dimensions are the following: abstraction level, learning style, working framework, work-step unit, progressive evaluation, premature commitment, penetrability, API elaboration, API viscosity, consistency, role expressiveness and domain correspondence. To apply this framework, we prepared a questionnaire with twelve questions, one per dimension, so that participants could manifest their impressions of the API regarding each dimension. For example, does the API expose all its functionality, or does it expose only certain abstractions of different functionalities? How easy is to make a change? Does the API work in a consistent way? This is a sample of the kind of questions that the users had to answer. In line with [9], we gave this questionnaire to the participants before and after performing the evaluation, to contrast their expectations of how the API should work against their impressions of how it actually works. Regarding the CSUQ, this is a questionnaire of 19 questions, all of them answered with a 7-point Likert scale, about the usability of the product being tested [39]. Altogether, users complete three questionnaires: one to check their expectations, one to gather their impressions of the tool (using the cognitive dimensions framework), and finally one to measure the usability of the tool.

4.3 Experimental design

After considering previous works on the assessment of APIs [2, 10, 26, 62], we designed the following evaluation framework:

(i) *Research design*. The experiment was designed as a within-subjects experiment, so users could appreciate the process of using the HERA system against the use case of not having it, that is, having to communicate with several emotion detectors manually, sync the return of results and the posterior fusion of data. Hence, each participant is put through three programming exercises which are essentially identical. This exercise consists of a set of tasks whose final goal is to communicate with two emotion detection services, so they analyse a media file looking for emotional responses. The participants wrote three programs to: (1) communicate with two different emotion detection services, namely Face++ and DummyDetector; (2) request an emotion analysis of a given media file, a picture in this case; and (3) fuse the different results of each detector into a single metric. For the first exercise, the participants chose their preferred programming language to communicate with these emotion recognition services. For the second exercise, they used the same language to write requests as if they were going to be sent to our server, while for the third exercise they had to use a different programming language to actually communicate with our server, although all of them used JavaScript or Python. The second exercise serves as a training process in the use of HERA, in which users learn how to use it, which involves communicating with the different endpoints, configuring the proxies on the server, and asking for an analysis and the corresponding results. In the final exercise, the users recovered the requests they wrote in the second exercise and adapted them to be sent to the server that was actually deployed, so that it would be able to act as a proxy for the Face++ and DummyDetector services.

(ii) *Intervention*. An instance of the server was launched using the Amazon Web Services Elastic Beanstalk service, which allows us to easily deploy a web application or infrastructure. As was stated in the instructions script that was given to the participants, interaction with the system would be carried out through HTTP requests directed to the AWS server. The participants were also offered a computer prepared with all the software needed to complete the different programming tasks, but they were also offered the possibility of using their own laptops if they were more comfortable programming on their own computers.

The whole evaluation process was divided into four parts.

(i) *Introduction to the Test*. The participants were introduced to the HERA System and to the context in which it exists, covering such things as the existence of different emotion recognition systems, how every system is different, how there is no norm for how things should be done, etc. In this first phase, they were given a document with instructions on what they had to do during the assessment, and information they would need in the process. After reading this script, they would complete the Twelve Cognitive Dimensions Questionnaire, answering the questions by thinking *only* in terms of the expectations they had about how the HERA System *should* work.

(ii) *First part of the Test*. In this part of the test, the participants had to communicate with Face++ and DummyDetector to ask for the emotions presented in a picture of a face. For this task, they used the programming language they preferred in order to communicate with the different servers, to store the results and to combine them in some way (they were advised to take the mean of the results representing the emotion "happiness" according to each different service).

(iii) *Training with HERA*. Once they had finished the first part of the evaluation, they performed a *simulation* using HERA. Using the programming language of their preference again and the documentation from the evaluation script and from the framework's repository [18], they had to write a structure for a set of generic requests aimed at each endpoint of the API.

(iv) *Second part of the Test*. In this second part, the participants communicate with the HERA system in order to ask *it* to carry out the previous task for them. The participants communicate with an instance of the system deployed on AWS to authenticate themselves, to set up the different proxies for each detector (Face++ and DummyDetector), to ask for an emotion recognition analysis on a media file (a picture) and a subsequent

Table 1 Participants data

Participant	Total time	Time first phase	Time training phase	Time second phase	Errors	Assistances
Participant #1	0:52:28	0:25:55	0:21:20	0:05:13	0	0
Participant #2	0:59:05	0:27:32	0:23:51	0:07:41	1	0
Participant #3	1:21:11	0:39:46	0:23:24	0:18:01	1	4
Participant #4	0:50:50	0:20:36	0:20:55	0:09:19	1	0
Participant #5	0:50:33	0:14:42	0:30:30	0:05:21	0	5
Participant #6	0:34:52	0:15:34	0:14:06	0:05:12	1	2
Participant #7	0:37:55	0:14:24	0:17:27	0:06:04	0	2

fusion of the results. This second part is done with a programming language that is different from the one used in the previous exercise. The goal of this is to make sure that the users focus on the behaviour of the framework, and that they are not distracted by the idiosyncrasies of a specific programming language.

(v) *Completing usability questionnaires*. After completing the three programming exercises, the participants completed the Twelve Cognitive Dimension Questionnaire again, this time answering the questions by considering their actual impressions of the tool, and the Computer System Usability Questionnaire. In this last questionnaire, a set of demographic questions were included.

The data collected during the evaluation were subsequently analysed, and the outcomes are described below.

4.4 Evaluation outcomes and discussion

During the evaluation, we measured the time it took the participants to finish each part, together with the errors they made (regarding the use of HERA, not the libraries they need to make the HTTP requests), and the times they needed help from the evaluator. In Table 1 we can see the time each participant needed to finish each part, the errors they made and the times they needed assistance (the participants' data have been anonymized).

In Fig. 10 we can see the average time the participants needed to finish each part. This parameter shows a dramatic reduction in the time needed to finish a task, thus validating part of

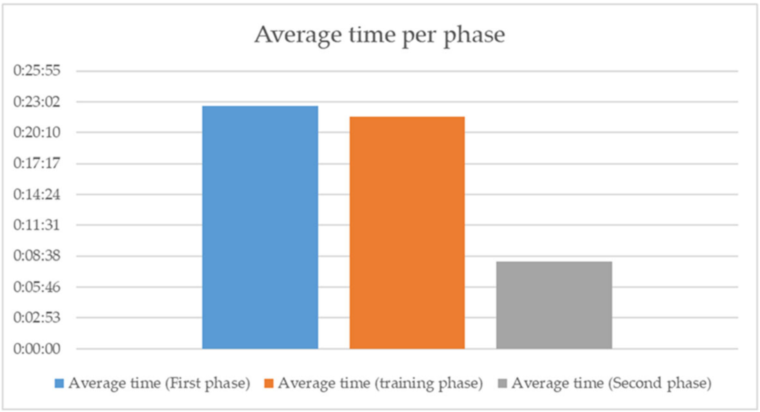

Fig. 10 Average time spent by phase

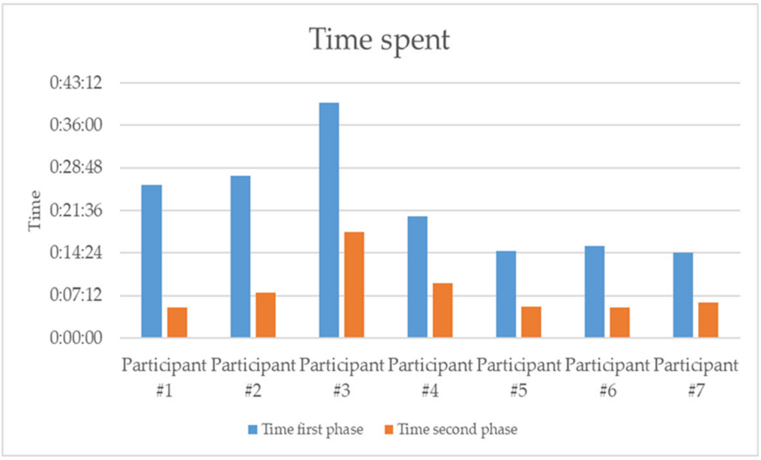

Fig. 11 Time spent per phase and user

our initial hypothesis, namely that after a previous training phase, which is not very expensive in terms of time, the use of HERA becomes very straightforward. Also, we must bear in mind that, since the evaluation is highly scripted, the participants save a lot of time in the first phase that is not reflected in the data gathered. The tasks of studying how to integrate emotion recognition in the system, how to gather the media, and analysing the different response formats of each detector (without having an evaluator helping you to dissect an HTTP response or an intricate JSON object) are all highly time-consuming, and they are *not* reflected in the first column of the figure, even though this phase already took the longest on average.

In Fig. 11 we can see a more detailed view of the data from Table 1 (omitting the time from the training phase). In this figure, each pair of columns represents the time it took a participant to finish the first and second phase. It is clear that the first phase took much longer to complete (around 20 minutes), in comparison with the second one (between 5 and 7 minutes). Due to the characteristics of this sample (small groups, little data dispersion), we carried out a Mann–Whitney U test to mathematically assess this time difference between the two phases, which led us to state that the two sets of data are significantly different with an alpha value of 0.05 ($U = 3.0$, $p = 0.00364$).

Finally, we can see the results gathered by the different questionnaires in Fig. 12 and Fig. 13. In Fig. 12 we can see the average response for each question. Since this questionnaire was composed by placing the positive value of the answer as the highest value (7 in this case), the higher the average response, the more positive the answer is, and the higher the total score, the higher the user satisfaction is. In this case, the average response for almost all the questions is above 5, with an average standard deviation of 0.9993, excepting the ninth question.

This question, which asks about the clarity of error messages returned by the API, has a standard deviation of 1.66 (a high value for such a narrow set of options). This difference of opinions was actually explained by the participants who gave a low value as an answer in later interviews.

The two users who answered that question with the lowest values had several issues with the HTTP library that they were using (bad URLs, wrong way of attaching data to the request, etc.), which gave them message errors they did not fully understand, due to a lack of knowledge of that particular library. Since they associated these vague errors with our server

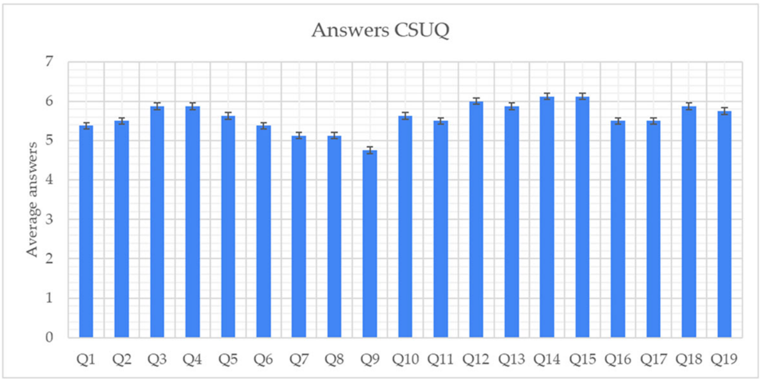

Fig. 12 Average answers of the Computer System Usability Questionnaire

(when the server was not even receiving the request), they gave a low score in that field. The rest of the questions received, on average, good scores. The total average score of this questionnaire is 106.5 out of a total of 133, which equals to 8.01 out of 10. This mark, along with the small deviation for each answer, can be interpreted as an agreement from the participants regarding the high usability of the system.

Finally, in Fig. 13, we can see the two spider diagrams we obtained by plotting the answers given to each dimension of the Twelve Cognitive Dimensions Questionnaire. By following the approach in [9], we surveyed the participants before and after, in order to be able to contrast their expectations and their impressions of the application. On average, we can see how our framework met most of their expectations, even improving upon what they were expecting, especially in the dimensions of Learning style, Premature Commitment, API Viscosity, Role Expressivity and Domain Correspondence.

With these results, we can conclude that the recruited developers found the tool useful to integrate technologies they were not familiar with. It is important to remark that this was the goal of this assessment. Since there are no other tools like HERA on literature to reduce the slope of the learning curve when it comes to integrate different emotion detectors and develop multimodal detectors, developers usually find it difficult to start developing this kind of detectors. This may lead to the development of defective multimodal systems or even to the abandonment of a project, in favour of a simpler unimodal detector. Although we reviewed some projects about multimodal detectors in section 2.4, these were designed ad-hoc for specific purposes or platforms, so the effort of trying to adapt these to a new scenario might be

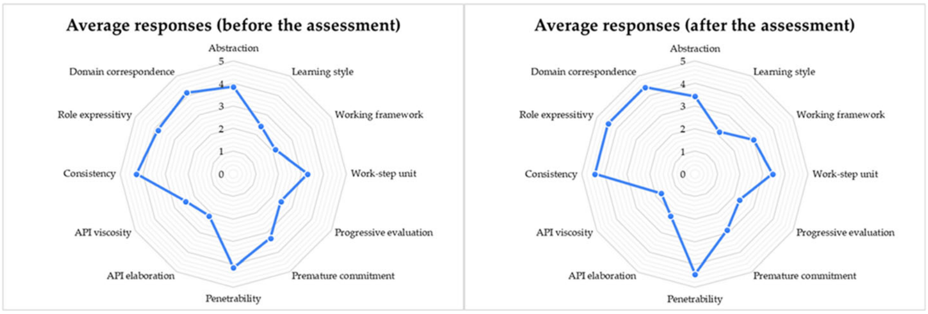

Fig. 13 Average responses of Twelve Cognitive Dimensions Questionnaire

even bigger than the effort of researching multimodality properly and develop a solution from the ground up. HERA is an effort to assist those developers in the elaboration of multimodal systems. However, further assessment with more archetypes of developers is still needed to assess the tool.

Regarding the application of HERA, this framework has been designed as a separate application that can be launched anywhere thanks to the underlying technology that we have used in its development. This approach just adds another entity to any existent architecture, leaving the mainframe of this existent application cleaner than it would be if all the emotion-related logic had to be written from scratch in the system. Some of the scenarios that would benefit from the use of HERA are the following:

- *Applications with educational purposes*. The idea of HERA actually spawned during the development of multimedia applications with education purposes [20, 21], when the development of said applications exposed the limitations and difficulties of building multimodal systems. In the context of these previous works, a tool deployed in the backend of the application would have helped us to develop the system, even if it had needed some customization.
- *Applications handling multimedia content*. Part of HERA needs to be capable of handling different types of multimedia input, being it in the form of a stream, a file, etc. Even if this first version has been more focused on the fusion of data, the proper management of multimedia information can be a valuable asset of HERA on its own.
- *Applications involving business processes*. Some of the activities that companies carry out, like costumer attention, can harvest great benefits from opinion mining and other forms of emotion detection. A system like HERA can help develop a more tailored solution to analyse the input of clients.

5 Conclusions

In this paper we have reviewed HERA, a framework developed in JavaScript to support the creation of multimodal emotion detectors and their subsequent integration with other applications, and the evaluation process that we carried out to test the first version. HERA has been developed while following the principles of modularity, independency and extendibility, since it is offered as an open-source tool that other developers are encouraged to adopt and help grow. With this tool, we strive to fill a gap in the existent literature, bringing together theory and implementation, and supporting our proposal in a standard format, namely the PAD classification system. The system proposed in this paper is designed to offer an easy way of including emotion recognition in an existing system in a straight-forward way, and reducing the usual workload involved in this task. Integrating our framework in another application's infrastructure is a matter of launching a Node server running the framework and sending HTTP requests to it, with this having only a minor impact on the app source code. Since the framework has been developed over Express and is open source, a new web application could be developed by even using the framework source code as a starting point. Although using this framework is not completely immediate, since it requires a previous setup and user training phase, we have proved empirically that the benefits of using this framework, in terms of time and effort, are worth this preparation phase. We tested the tool with real users, who used it to develop a small demo, which revealed some of HERA's strong and weak points.

One of the current challenges is to go on developing HERA so that it can support more forms of interactions and new detectors. Even though one of its key aspects is that it is easy to extend, the more features it has from the beginning, the more likely it is that new developers use it, thus enlarging the community of HERA's users, which will help us to find bugs and make HERA bigger, safer and more solid. In order to tackle this challenge, we will review existent mainstream emotion detection services to integrate them into HERA. This extension process will also include popular handheld and wearable devices, which will contribute to extend the communication capabilities of HERA.

This first version of the framework is specially focused on the fusion of data, but HERA can still be improved to increase its capabilities regarding the management of media input, so that developers can stream multimedia data to HERA and request analysis over these continuous streams, instead of making punctual requests each time.

When it comes to the fusion of data, HERA can greatly benefit from having a wide variety of algorithms to aggregate data all together. We will consider different usages of this tool to develop algorithms to cover the different needs of future developers.

Lastly, the management of results within HERA can open the door to machine learning, that could be applied once we have gathered enough results, previously validated by experts. This could help to triple validate the aggregated data and/or to increase even more the richness of the data.

Funding Open Access funding provided thanks to the CRUE-CSIC agreement with Springer Nature.

Availability of data and material Any additional data or material could be provided by the authors if needed.

Code availability The code of this framework is available at [18].

Declarations This work has been partially supported by the national project granted by the Ministry of Science, Innovation and Universities (Spain) with reference RTI2018–099942-B-I00 and by the regional project (ref: SBPLY/17/180501/000495) granted by the regional government (JCCM) and the European Regional Development Funds (FEDER).

Conflicts of interest/Competing interests On behalf of all authors, the corresponding author states that there is no conflict of interest.

References

1. Alepis E, Virvou M (2012) Multimodal object oriented user interfaces in mobile affective interaction. Multimed Tools Appl 59(1):41–63
2. Arroyo I, Cooper DG, Burleson W, Woolf BP, Muldner K, Christopherson R (2009) Emotion sensors go to school. Front Artificial Intel App 200(1):17–24. https://doi.org/10.3233/978-1-60750-028-5-17

3. Blackwell AF and Green TRG (2000) “A Cognitive Dimensions Questionnaire Optimised for Users,” Proc. 12th Work. Psychol. Program. Interes. Gr., no. April, pp. 137–154.
4. Calvo RA, D’Mello S (2010) Affect detection: an interdisciplinary review of models, methods, and their applications. IEEE Trans Affect Comput 1(1):18–37
5. Cambria E, Grassi M, Hussain A, Havasi C (2012) Sentic Computing for social media marketing. Multimed Tools Appl 59(2):557–577
6. Chao X, Zhiyong F (2008) A trusted affective model approach to proactive health monitoring system. Proc - 2008 Intern Sem Fut BioMed Inform Engin, FBIE 2008:429–432. https://doi.org/10.1109/FBIE.2008.52
7. Chen J, Hu B, Li N, Mao C, and Moore P (2013) “A multimodal emotion-focused e-health monitoring support system,” in Proceedings - 2013 7th International Conference on Complex, Intelligent, and Software Intensive Systems, CISIS 2013, pp. 505–510, https://doi.org/10.1109/CISIS.2013.92.
8. Chen LS, Huang TS, Miyasato T, and Nakatsu R 1998 “Multimodal human emotion/expression recognition,” in Proceedings Third IEEE International Conference on Automatic Face and Gesture Recognition, pp. 366–371.
9. Clarke S (2020) “Measuring API Usability”, Dr. Dobb's: The World of Software Development, May 01, 2004. Accessed on: February 12, Available at:https://www.drdobbs.com/windows/measuring-api-usability/184405654
10. Clarke S, Becker C (2003) Using the Cognitive Dimensions Framework to evaluate the usability of a class library. Proc First Jt Conf EASE PPIG, no. April:359–366
11. Dai W, Liu Z, Yu T, and Fung P (2020) “Modality-transferable emotionembeddings for low-resource multimodal emotion recognition,”.
12. Darekar RV, Dhande AP (2016) Enhancing effectiveness of emotion detection by multimodal fusion of speech parameters. Intern Conf Electri, Electron Optimi Tech, ICEEOT 2016:3242–3246. https://doi.org/10.1109/ICEEOT.2016.7755303
13. S. D’Mello, A. Graesser, and R. W. Picard, “Toward an affect-sensitive auHERAutor,” IEEE Intell Syst, vol. 22, no. 4, pp. 53–61, Jul. 2007, https://doi.org/10.1109/MIS.2007.79.
14. D’Mello SK, Kory J(2015) “A review and meta-analysis of multimodal affect detection systems,” ACM Computing Surveys, vol. 47, no. 3. Association for Computing Machinery, 01-Feb-2015.
15. Ekman P (1999) Basic emotions. In: Handbook of cognition and emotion, vol ch. 3. John Wiley & Sons, New York, pp 45–60
16. Express, “Fast, unopinionated, minimalist web framework for Node.js”(2020). Accessed on: April 10th, 2020. Available at: https://expressjs.com/
17. Fabien Mäel (2019) “Multimodal-Emotion-Recognition”, June 28, 2019. Accessed on: March 31, 2020. Available: https://github.com/maelfabien/Multimodal-Emotion-Recognition
18. Garcia-Garcia, Jose Maria, “HERA system: Three-level multimodal emotion recognition framework to detect emotions combining different inputs with different formats. Accessed on: April 10th 2020. Available at: https://github.com/josemariagarcia95/hera-system
19. Garcia-Garcia JM, Penichet VMR, and Lozano MD (2017) “Emotion detection: a technology review,” in Proceedings of the XVIII International Conference on Human Computer Interaction - Interacción ‘17, pp. 1–8.
20. Garcia-Garcia JM, Penichet VMR, Lozano MD, Garrido JE, Lai-Chong Law E (2018) Multimodal affective computing to enhance the user experience of educational software applications. Mob Inf Syst 2018(10):10. https://doi.org/10.1155/2018/8751426
21. Garcia-Garcia JM, Cabañero M e del M, Penichet VMR, and Lozano MD(2019) “EmoTEA: Teaching Children with Autism Spectrum Disorder to Identify and Express Emotions,” in Proceedings of the XX International Conference on Human Computer Interaction - Interacción ‘19, pp. 1–8, https://doi.org/10.1145/3335595.3335639.
22. Gilleade KM, Alan D, and Allanson J (1997) “Affective videogames and modes of affective gaming: assist me, challenge me, emote me,” 2005, .D. L. Hall and J. Llinas, “An introduction to multisensor data fusion,” Proc IEEE, vol. 85, no. 1, pp. 6–23.
23. Gonzalez-Sanchez J, Chavez-Echeagaray M-E, Atkinson R, Burleson W (2011) Affective computing meets design patterns: A pattern-based model for a multimodal emotion recognition framework. Proc 16th Eur Conf Pattern Lang Programs - Eur 11, no. July:1–11. https://doi.org/10.1145/2396716.2396730
24. J. Gonzalez-Sanchez, M. E. Chavez-Echeagaray, R. Atkinson, and W. Burleson, “ABE: An agent-based software architecture for a multimodal emotion recognition framework,” Proc - 9th Work IEEE/IFIP Conf Softw Archit WICSA 2011, no. May 2014, pp. 187–193, 2011, https://doi.org/10.1109/WICSA.2011.32
25. Green TRG (1989) Cognitive dimensions of notations. In: Sutcliffe A, Macaulay L (eds) People and computers V. Cambridge University Press, Cambridge, UK, pp 443–460
26. Green TRG, Petre M (1996) Usability analysis of visual programming environments: a ‘cognitive dimensions’ framework. J Vis Lang Comput 7(2):131–174. https://doi.org/10.1006/jvlc.1996.0009

27. Gupta SK, Ashwin TS, Guddeti RMR (2019) Students' affective content analysis in smart classroom environment using deep learning techniques. Multimed Tools Appl 78(18) Multimedia Tools and Applications:25321–25348
28. A. G. Hauptmann and P. McAvinney, "Gestures with speech for graphic manipulation," Int J Man Mach Stud, vol. 38, no. 2, pp. 231–249, Feb. 1993.
29. Hung JC-S, Chiang K-H, Huang Y-H, Lin K-C (2017) Augmenting teacher-student interaction in digital learning through affective computing. Multimed Tools Appl 76(18) Multimedia Tools and Applications:18361–18386
30. Jaiswal S, Virmani S, Sethi V, De K, Roy PP (2019) An intelligent recommendation system using gaze and emotion detection. Multimed Tools Appl 78(11):14231–14250
31. Jaques N, Conati C, Harley JM, Azevedo R (2014) "Predicting Affect from Gaze Data during Interaction with an Intelligent Tutoring System," in Intelligent Tutoring Systems. Springer, Cham, pp 29–38
32. Jarraya SK, Masmoudi M, Hammami M (2021) A comparative study of autistic children emotion recognition based on Spatio-temporal and deep analysis of facial expressions features during a meltdown crisis. Multimed Tools Appl 80(1):83–125
33. Khanh TLB, Kim S-H, Lee G, Yang H-J, Baek E-T (2021) Korean video dataset for emotion recognition in the wild. Multimed Tools Appl 80(6):9479–9492
34. Kleinginna PRJ, Kleinginna AM (1981) A categorized list of emotion definitions, with suggestions for a consensual definition. Motiv Emot 5(3):263–291
35. Kołakowska A, Landowska A, Szwoch M, Szwoch W, Wróbel M (2015) Modeling emotions for affect-aware applications. In: Wrzycza S (ed) Information systems development and applications. Faculty of Management University of Gdańsk, Poland, pp 55–69
36. Koelstra S, Muhl C, Soleymani M, Jong-Seok Lee A, Yazdani T, Ebrahimi T, Pun A, Nijholt IP (2012) DEAP: a database for emotion analysis; using physiological signals. IEEE Trans Affect Comput 3(1):18–31
37. Kossaifi, Jean, Robert Walecki, Yannis Panagakis, Jie Shen, Maximilian Schmitt, Fabien Ringeval, Jing Han et al (2019) "SEWA DB: A Rich Database for Audio-Visual Emotion and Sentiment Research in the Wild." IEEE Transactions on Pattern Analysis and Machine Intelligence.
38. Kumar A, Garg G (2019) Sentiment analysis of multimodal twitter data. Multimed Tools Appl 78(17): 24103–24119
39. Lewis JR (2018) Measuring perceived usability: the CSUQ, SUS, and UMUX. Int J Hum Comput Interact 34(12):1148–1156. https://doi.org/10.1080/10447318.2017.1418805
40. Landowska A (2018) Towards new mappings between emotion representa-tion models. Appl Sci 8(2):274
41. Lewis JR (1995) IBM computer usability satisfaction questionnaires: psychometric evaluation and instructions for use. Int J Hum Comput Interact 7:57–78
42. Li Z, Fan Y, Jiang B, Lei T, Liu W (2019) A survey on sentiment analysis and opinion mining for social multimedia. Multimed Tools Appl 78(6):6939–6967
43. Mansoorizadeh M, Moghaddam Charkari N (2010) Multimodal information fusion application to human emotion recognition from face and speech. Multimed Tools Appl 49(2):277–297
44. Maat L, Pantic M (2007) Gaze-X: Adaptive, affective, multimodal interface for single-user office scenarios. Lect Notes Comput Sci (including Subser Lect Notes Artif Intell Lect Notes Bioinformatics) 4451 LNAI: 251–271. https://doi.org/10.1007/978-3-540-72348-6_13
45. Martin B, Hanington B (2012) Universal methods of design: 100 ways to research complex problems, develop innovative ideas, and design effective solutions. Rockport Publishers, Beberly (Massachusetts), pp 204–205
46. Mehrabian A, Russell JA (1974) An approach to environmental psychology. The MIT press
47. Mittal T, Guhan P, Bhattacharya U, Chandra R, Bera A, Manocha D (2020, 2020) EmotiCon: Context-Aware Multimodal Emotion Recognition Using Frege's Principle. In: IEEE/CVF conference on computer vision and pattern recognition (CVPR), Seattle, WA, USA, pp 14222–14231. https://doi.org/10.1109/CVPR42600.2020.01424
48. Nielsen J, Landauer T (1993) A mathematical model of the finding of usability problems. Proceedings of the Interact'93 and CHI'93 Conference on Human Factors in Computing systems; 1993 Apr. ACM, Amsterdam, the Netherlands. New York, pp 24–29
49. Osman H. Al and Falk TH (2017) "Multimodal Affect Recognition: Current Approaches and Challenges," in Emotion and Attention Recognition Based on Biological Signals and Images, InTech.
50. Oviatt S, DeAngeli A, and Kuhn K (1997) "Integration and synchronization of input modes during multimodal human-computer interaction," in Proceedings of the SIGCHI conference on Human factors in computing systems - CHI '97, pp. 415–422.
51. Oehl M, Siebert FW, Tews T-K, Höger R, Pfister H-R (2011) Improving human-machine interaction - A non-invasive approach to detect emotions in car drivers. Lect Notes Comput Sci (including Subser Lect Notes Artif Intell Lect Notes Bioinformatics) 6763 LNCS, no. PART 3:577–585
52. Pantic M, Sebe N, Cohn JF, Huang T (2005) Affective multimodal human-computer interaction. Proc 13th ACM Int Conf Multimedia, MM 2005 , no. January:669–676

53. Patwardhan AS (2018) “Multimodal mixed emotion detection,” in Proceedings of the 2nd International Conference on Communication and Electronics Systems, ICCES 2017, 2018, vol., pp. 139–143, https://doi.org/10.1109/CESYS.2017.8321250.
54. Picard RW (1995) Affective Computing. MIT Press 321:1–16
55. Poria S, Cambria E, Bajpai R, and Hussain A (2017) “A review of affective computing: from unimodal analysis to multimodal fusion,” Inf. Fusion.
56. Pyeon Myeongjang (2018) “IEMo: web-based interactive multimodal emotion recognition framework”, Abril 30, 2018. Accessed on: April 28, 2020. Available at: https://github.com/mjpyeon/IEMo
57. Ringeval F, Eyben F, Kroupi E, Yuce A, Thiran JP, Ebrahimi T, Lalanne D, Schuller B (2015) Prediction of asynchronous dimensional emotion ratings from audiovisual and physiological data. Pattern Recogn Lett 66:22–30
58. Rousidis D, Koukaras P, Tjortjis C (2020) Social media prediction: a literature review. Multimed Tools Appl 79(9–10):6279–6311
59. Sekhavat YA, Sisi MJ, and Roohi S (2020) “Affective interaction: using emotions as a user interface in games”, Multimedia Tools and Applications. Multimedia Tools and Applications, Affective interaction: Using emotions as a user interface in games.
60. Sethu V, Provost EM, Epps J, Busso C, Cummins N, and Narayanan S 2019 “The ambiguous world of emotion representation,”.
61. Silva L. C. De, Miyasato T, and Nakatsu R (1997) “Facial emotion recognition using multi-modal information,” Proc. ICICS, 1997 Int. Conf. Information, Commun. Signal Process. Theme Trends Inf. Syst. Eng. Wirel. Multimed. Commun. (Cat. No.97TH8237), vol. 1, no. May 2014, pp. 397–401.
62. L. C. De Silva, Pei Chi Ng (2000) “Bimodal emotion recognition,” in Proceedings Fourth IEEE International Conference on Automatic Face and Gesture Recognition (Cat. No. PR00580), pp. 332–335.
63. Siriwardhana S, Kaluarachchi T, Billinghurst M, Nanayakkara S (2020) Multimodal emotion recognition with transformer-based self supervised feature fusion. IEEE Access 8:176274–176285. https://doi.org/10.1109/ACCESS.2020.3026823
64. W3C, emotion markup language, (May 22, 2014). Accessed on: February 17th, 2020. Available: https://www.w3.org/TR/emotionml/
65. W3C, multimodal interaction framework, multimodal interaction working group, (May 06, 2003). Accessed on: February 17th, 2020. Arvailable: https://www.w3.org/TR/mmi-framework/
66. Wang Z, Ho S-B, Cambria E (2020) A review of emotion sensing: categorization models and algorithms. Multimed Tools Appl 79(47–48):35553–35582
67. Wijayarathna C, Arachchilage NAG, Slay J (2017) “Using Cognitive Dimensions Questionnaire to Evaluate the Usability of Security APIs,” no. 2004.
68. Woolf B, Woolf B, Burelson W, Arroyo I(2007) “Emotional Intelligence for Computer Tutors,” Suppl. Proc. 13TH Int. Conf. Artif. IN-TELLIGENCE Educ. (AIED 2007), (PP, pp. 6–15.
69. Yamauchi T (2013) “Mouse Trajectories and State Anxiety: Feature Selection with Random Forest,” in 2013 Humaine Association Conference on Affective Computing and Intelligent Interaction, pp. 399–404.
70. Zhao S et al. (2020) "Discrete Probability Distribution Prediction of Image Emotions with Shared Sparse Learning," in IEEE Transactions on Affective Computing, vol. 11, no. 4, pp. 574–587, 1 Oct.-Dec, https://doi.org/10.1109/TAFFC.2018.2818685.
71. Zhao S, Gholaminejad A, Ding G, Gao Y, Han J, Keutzer K (2019) Personalized emotion recognition by personality-aware high-order learning of physiological signals. ACM Trans Multimed Comput Commun Appl 15, 1s, article 14, (February 2019):18. https://doi.org/10.1145/3233184
72. Zhao S, Ding G, Gao Y, Han J(2017) “Approximating discrete probability distribution of image emotions by multi-modal features fusion,“in Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17, pp. 4669–4675
73. Z. Zhang, J. M. Girard, Y. Wu, X. Zhang, P. Liu, U. Ciftci, S. Canavan, M. Reale, A. Horowitz, H. Yang, J. F. Cohn, Q. Ji, and L. Yin, “Multimodal spontaneous emotion Corpus for human behavior analysis”, 2016.
74. Zhang S, Wu Z, Meng HM, Cai L (2010) Facial expression synthesis based on emotion dimensions for affective talking avatar. Smart Innov Syst Technol 2010(1):109–132. https://doi.org/10.1007/978-3-642-12604-8_6
75. W. L. Zheng, W. Liu, Y. Lu, B. L. Lu, and A. Cichocki, “EmotionMeter: a multimodal framework for recognizing human emotions,” IEEE Trans Cybern, vol. 49, no. 3, pp. 1110–1122, Mar. 2019, https://doi.org/10.1109/TCYB.2018.2797176.

Affiliations

Jose Maria Garcia-Garcia[1] · **Maria Dolores Lozano**[2] · **Victor M. R. Penichet**[2] · **Effie Lai-Chong Law**[3]

Maria Dolores Lozano
Maria.Lozano@uclm.es

Victor M. R. Penichet
Victor.Penichet@uclm.es

Effie Lai-Chong Law
lai-chong.law@durham.ac.uk

1 Informatics Research Institute of Albacete, 02006 Albacete, Spain

2 Department of Computing Systems, Universidad de Castilla-La Mancha, 02006 Albacete, Spain

3 Department of Computer Science, Durham University, Durham DH1 3LE, UK