EmoCam: capturing emotions using non-invasive technologies

D. Freitas, J.E. Muñoz and S. Bermudez i Badia

Madeira Interactive Technologies Institute, Funchal, Portugal
Faculty of Exact Sciences and Engineering, University of Madeira, Funchal, Portugal

Abstract. Affective computing uses human emotions to adapt applications to users' moods, affections and behavior. Measuring these emotions is often challenging due mainly the invasiveness of the technology needed for emotion's sensing. Novel approaches based on non-invasive approaches utilize computer vision and machine learning algorithms to recognize emotions and facial expressions just by using cameras. Two promising technologies have been recently developed and freely released looking for brings emotional intelligence to our systems: the AffectivaSDK and the IntelReal Sense Camera. These tools allow capturing information of eight different emotions, five facial expressions and fourteen hand gestures. This bachelor project embraces the design and development of a software toolkit, the EmoCam Panel which allows to access all of the data from these affective tools and stream it to video games developed in Unity3D, thus enabling game developers and researchers include emotional intelligence in their applications. A proof of concept was developed using a custom-made endless runner video game in which three different adaptive rules were defined for modifying game events through emotional responses. Moreover, the EmoCam panel can be connected with the tools developed in the NeurorehabLab for serious games, for health investigation, thus extending the capabilities of these tools via including emotion sensing technologies.

Keywords: Facial recognition; Emotions; Affective Computing; RealSense; Webcam; Video games; Unity; Hand-gestures.

1 Introduction

Reading user's emotions data is always a complex task considering the multiplicity of variables that can directly influence the acquisition and interpretation processes. Researchers often use invasive technologies such as electroencephalography (EEG) or facial electromyography (facial EMG) to collect bioelectrical signals from the body, looking for a posterior signal filtering and data interpretation stages *feed*. A different alternative to interpret the emotional state of humans is using images or videos, recorded with simple cameras in a non-invasive way. This technology uses computer algorithms that can identify key landmarks on the user's face, thus detecting his expressions and associate them to a probability of having an emotion [7]. By using emotion recognition techniques, novel computer systems can assist users in order to avoid frustrations and enhance human computer interactions. Thus, it is possible to create affective-responsive strategies, creating applications that can be affectively-adjustable to each user. This affective computing area has been rapidly growing in the last decade, allowing the development of several hardware and software tools which are commonly available to different communities, such as researchers and game developers [7]. Nevertheless, the adoption of this technology is a slow process, demanding users be familiar with very specific software-related terms such as SDKs (Software Development Kits), IDE (Integrated Development Environments) and communication protocols, therefore limiting a widespread use of this powerful technology.

The main goal of this project is to develop a software toolkit which can integrate two of the most novels and advanced developments in the affective computing field: the Affectiva SDK and the RealSense Camera from Intel. These tools, allow the collection of affective data, face data, expressions, physiological data and hand movements. The final result should be integrated into existent software tools developed at the NeurorehabLab, a research group which studies the intersection of technology, neuroscience and clinical practice, mainly driving by serious gaming approaches. Through this integration, the affective data could be sent to any game engine, enabling game developers to use this data in a simple and clean way, without dealing with data synchronization and SDKs integration. During this project, we will propose an architecture that will have the affective computing panel, the methodology to integrate it with the NeurorehabLab tools and a proof of concept for an affective video game.

2 Emotion recognition

There are mainly two ways to detect user's emotions: *direct* (using technologies that measures user's facial muscle movements or analyzing physiological signals in the body) and *indirect* (through direct observation or using non-invasive technologies) [7]. It seems obvious that, direct technology provides a more accurate measurement; however despite the use of invasive technologies, researchers and developers also need to have deep knowledge about facial anatomy and morphology (muscles' movement), as well as the physiological phenomena.

Examples of direct technologies:

- **Facial EMG**: using electrodes attached to the user's face, to detect discharges produced by facial muscle movements;
- **Electroencephalography (EEG)**: using electrodes placed along the scalp to detect voltage fluctuations resulting from ionic current within the neurons of the brain [4].

Examples of indirect technologies:

- **Observer judgments**: expressive behaviors are shown to observers who are asked to rate the expressions on emotion scales [7];

– **Webcam**: using computer vision algorithms to identify face expressions and then emotions.

This project is centered in the use of indirect (or non-invasive) technologies, more specifically camera devices for detecting human emotions through sophisticated computer vision algorithms. Particularly, we used the Affectiva SDK, a software tool that can be used to recognize a set of emotions only by using webcams and the RealSense Camera from Intel, which is a powerful depth camera which incorporates multiple algorithms for face expression and hand gesture detection.

2.1 Affectiva SDK

As seen before, it is hard to define what an emotion is [7], but us, humans, express our feelings by emotions at every moment. We express and communicate emotion through facial expressions, using our facial muscles.

Affectiva is an emotion measurement technology that was created in 2009 in MIT's Media Lab, whose main goal is to develop a way that a computer can predict/recognize human emotions based on facial expressions (for example, brow raise, eye widen or lip pucker) as input to calculate the likelihood of an emotion. With this SDK is possible to detect twenty facial expressions and detect eight emotions: anger, contempt, disgust, engagement, fear, joy, sadness and surprise.

But how is it possible to detect facial expressions? It is important, now, to define what is *deep learning*. Deep learning is an area of research that allows computers to learn from experience and understand the world in terms of hierarchy of concepts, with each concept defined in terms of its relation to simpler concepts. By gathering knowledge from experience, this approach avoids the need for human operators to formally specify all of the knowledge that the computer needs. The hierarchy of concepts allows the computer to learn complicated concepts by building them out of simpler ones [3].

Computer vision algorithms identify key landmarks on the face and then Affectiva SDK uses the concept/techniques of machine learning (such as: neural nets) to facial expression classification and face detection, analyzing the image's pixels. This is why they collect more than five million faces from 75 countries and from people aged under 18 years old to 65 or plus, amounting to over twelve billion emotion data points, being today the most extensive database of face points. Please note that, according to them, it is also possible to detect, and with a heightened confidence level, emotions from people using glasses.

Having a system that can detect facial expression, detecting emotions is a straight forward task, using the below table 1 from [1] and using any optical sensor, device camera or standard webcam.

Emotion	**Increase likelihood**	**Decrease likelihood**
Joy	Smile	Brow raise
Anger	Brow furrow, lid tighten, eye widen, chin raise, mouth open and lip suck	Inner brow raise, brow raise and smile
Disgust	Nose wrinkle and upper lip raise	Lip suck and smile
Engagement	Brow raise, brow furrow, nose wrinkle, lip corner depressor, chin raise, lip pucker, lip press, mouth open, lip suck and smile	N. a.
Surprise	Inner brow raise, brow raise, eye widen and jaw drop	Brow furrow
Fear	Inner brow raise, brow furrow, eye widen and lip stretch	Brow raise, lip corner depressor, jaw drop and smile
Sadness	Inner brow raise, brow furrow and lip corner depressor	Brow raise, eye widen, lip press, mouth open, lip suck and smile
Contempt	Brow furrow and smirk	Smile

Table 1. Calculation of emotions.

This SDK is freely available for multiple platforms: iOS, Android, Web, Windows, Linux, macOS, Unity and Raspberry Pi.

2.2 RealSense Camera SR300

Hardware Description: Is a powerful camera developed by Intel formerly known as Intel Perceptual Computing which enables color stream video, over USB 3.0, with a resolution of 720p (at 60 frames) or 1080p (at 30 frames). Besides this, it has a depth sensor that can stream high quality 3D depth videos in a distance of, approximately, 20 cm to 150 cm, an infrared camera and an infrared laser projector [6]. The camera provides synchronized color, depth and IR video stream data to the client system at same frame rate or with a different frame rate, and has several applications like: face analytics and tracking, and hand and finger tracking.

The image 1 shows the main components of the camera that are needed to process the images:

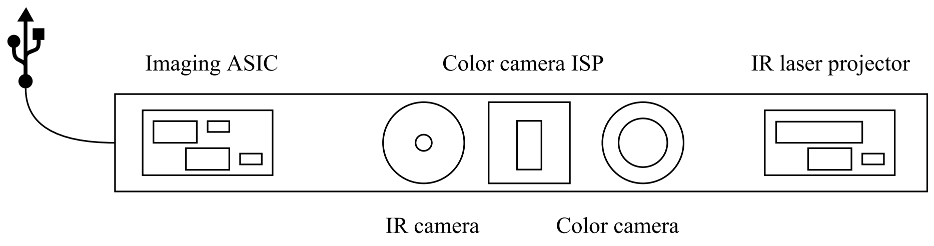


Fig. 1. Main architecture of RealSense SR300.

The IR projector and IR camera operate together using coded light patterns to produce a two-dimensional array of monochromatic pixel values. These values are processed by imaging application specific integrated circuits (ASIC) that generate depth and/or infrared video frames, which are transmitted to the client system via USB 3.0.

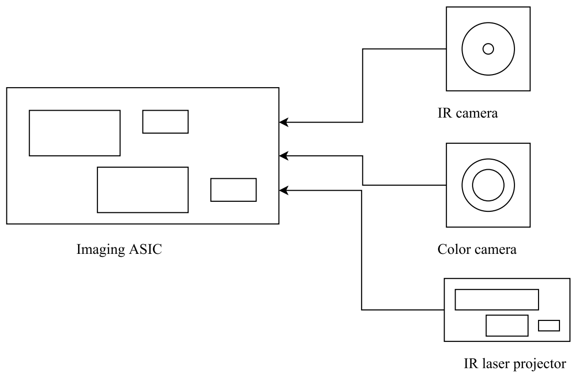


Fig. 2. Imaging application specific integrated circuits.

The color camera consists of a chromatic sensor and an image signal processor which captures and processes chromatic pixel values. These values generate color video frames which are transmitted to the imaging ASIC and then transmitted to the client system via USB 3.0 [6].

2.3 Intel RealSense SDK

With RealSense cameras, Intel also developed a SDK for accessing the data using multiple scripting languages. With this SDK, it is possible to create applications that we can naturally interact with, making them more immersive.

This SDK is very powerful and it is capable of facial recognition, detecting facial expressions, hand gestures, hand/finger tracking, 3D scanning, voice recognition and, but not only, physiological data (in this case, the heart rate).

Natively, this SKD is programed in C++, but because of interface modules, it supports C# , Java, JavaScript (for web applications), Processing and, as expected, C++.

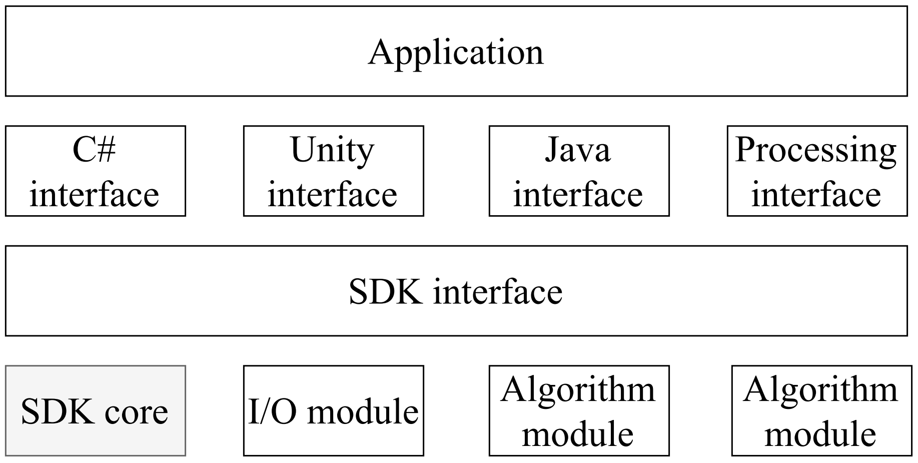


Fig. 3. Intel RealSense SDK architecture.

Most of the SDK functionalities were available to the C++/C# developers, which resulted in undue disadvantages to game engines such as Unity3D and other application development frameworks. To address this problem, Intel RealSense SDK architecture defines a wrap layer, providing uniform access to the core and middleware capabilities through the specially designed interfaces for each framework [8], as shown in image 4. This gives the user freedom to choose any programing language.

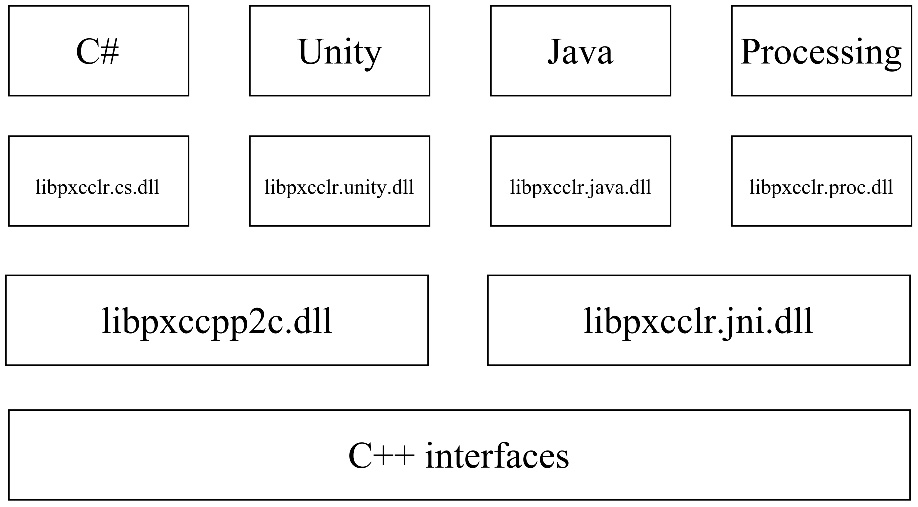


Fig. 4. SDK access across different application development frameworks.

The SDK core is responsible for organizing the execution pipeline and manages two types of SDK modules [5]:

- **I/O module**: get the data from the camera and send the data to an output device or to an algorithm module;
- **Algorithm module(s)**: includes algorithms that are responsible for all capabilities of the camera/SDK, like face tracking, voice recognition... note that, it is not only two modules as shown in 3, but n, each of one for each capability.

It is possible to have multiple algorithm modules running at the same time with the same pipeline, for this, SDK core has a built-in manager for each pipeline. It is also possible to have multiple cameras in one application; each of them have a pipeline and consequently a manager for each.

2.4 Class hierarchy

In figure 5 the class hierarchy of Intel RealSense SDK is shown.

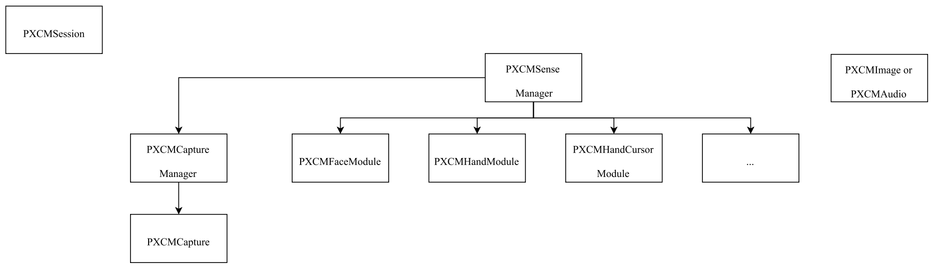


Fig. 5. Intel RealSense SDK class hierarchy.

The class *PXCMSenseManager* is responsible for the organization and management of the execution pipeline [8]. *PXCMCaptureManager* manages camera devices and respective streams, for example, both color or depth stream. All algorithm modules presented in figure 3 are inherited from class *PXCMSenseManager* and will be explained in detail in below sections.

3 Communication and integration with video games

Motivation #1: despite of both SDKs being accessible to everyone, the implementation in, for example, a video game, is not straight forward and not generic (depends on the game engine); researchers and developers need to spend research time to understand how SDKs work and implement/develop a solution from *scratch* - that is why there are few studies about affective computing systems using this frameworks.

For tackling this issue, we hold on the tools already developed by the NeurorehabLab which allow a fluid communication between information from multiple sensors and video games developed in Unity3D. In the below sections, we will introduce two of these tools in which this affective technology, that we called the *EmoCam Panel*, can be easily integrated.

3.1 RehabNet CP

RehabNet CP is a tool that was developed by NeuroRehabLab [9]. The main objective of RehabNet CP is to provide a simple way to enable game developers to stream data from hardware sensors to applications, enabling, for example, low cost at-home rehabilitation solutions. RehabNet CP was implemented in Unity and acts as a device router. Today, there are lots of sensors already integrated with RehabNet CP such as: Oculus Rift; Microsoft Kinect version 1 and version 2...

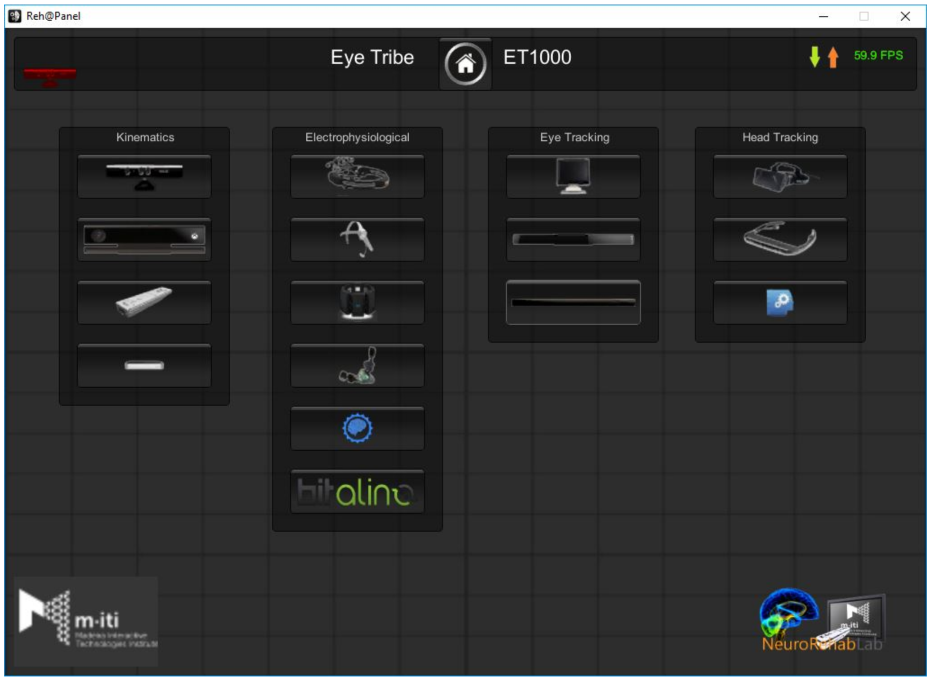


Fig. 6. Devices integrated with RehabNet CP.

The communication is established using UDP (User Datagram Protocol) communication to enable continuous and fast data transmission being both geographically and technologically accessible from everywhere [9].

3.2 Biocybernetic Loop Engine

The Biocybernetic Loop Engine (BL Engine) is one more tool that was developed by NeuroRehabLab; this application enables the creation of physiologically modulated video games by means of wearable sensors, creating **`If then...`** **`else`** rules, for example: **`If`** *`heartRate`* **`>`** *`limit`* **`then`** *`difficulty++`* **`else`** *`difficulty-`* , in an easy manner, changing application's variables with the information that was received by various sensors, this is called *adaptive rules*.

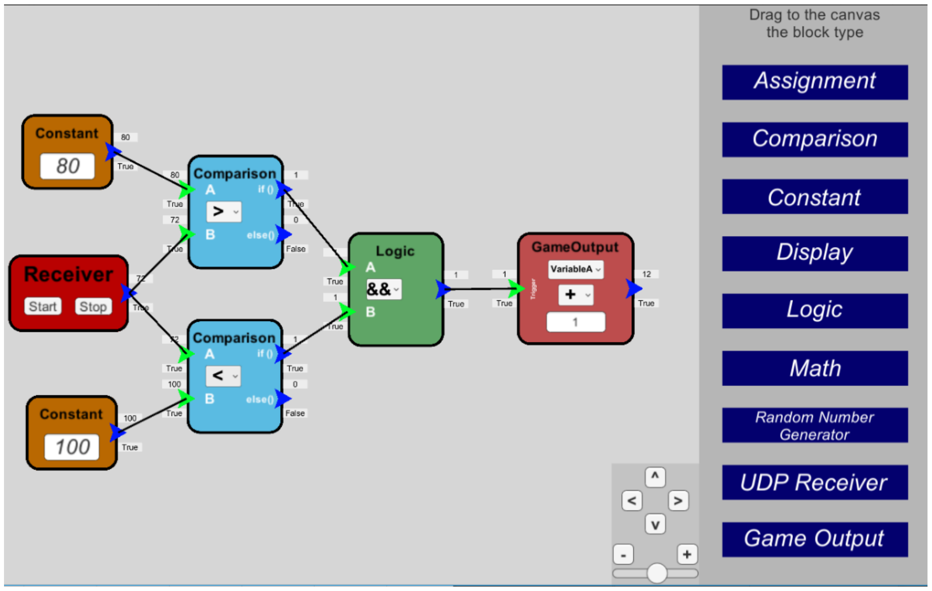


Fig. 7. Creation of adaptive rules using BL Engine.

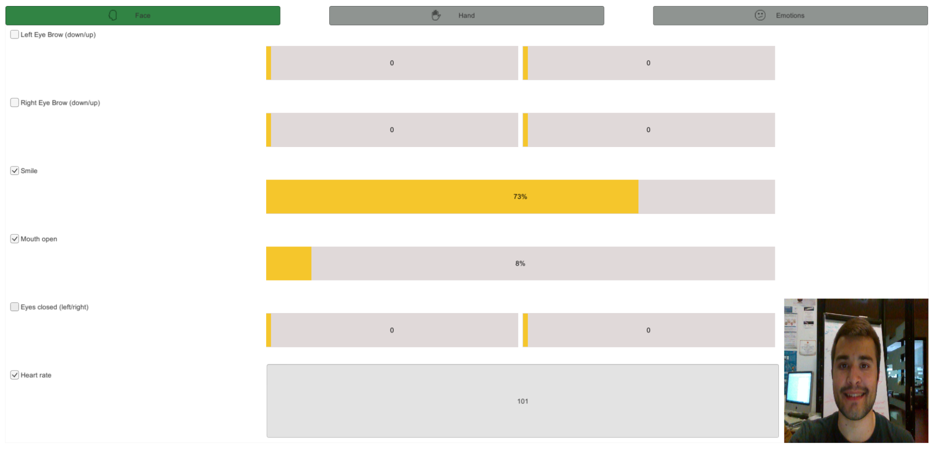
In addition to the creation of adaptive rules, this application also includes a signal acquisition panel, helping the visualization of multiple cardiac-related sensors.

Similar to the other tool, there are some supported cardiac-related sensors, such as: Biosignal Plux; Bitalino DIY Kit...; some of these devices are already built-in and some are external clients (like an Android smartwatch) and uses UDP to stream data to the signal processing. The data can be sent to the application in two ways: *native*, when the data comes from built-in devices or *UDP*, when external clients.

Motivation #2: for RehabNet CP, despite of the big set of sensors, none of them can track simultaneously face data, hand movements or user emotions. Our contribution is to develop a panel that can track all of this data and can be easily integrated in RehabNet CP. The same thing happens with BL Engine, in this case, sending information though UDP to the tool, will help developers to create more adaptive rules, using, for example, user's emotions.

4 EmoCam framework

EmoCam Panel is a toolkit consisting of three sub-panels: *Face Expression*, *Hand-gestures* and *Emotions Recognition*. Every item has a checkbox that, if is on, then it starts to track and send, through UDP, to the RehabNet CP using a specific protocol; if it is disabled, then it stops to track and interrupts the transmission. The retrieved information can be seen in two ways: in a text box, for heart rate and hand position, and in a progress bar for more quantitative information, for example: emotions. On the bottom right, there is a space for color image stream from the camera.



☐ Left Eye Brow (down/up) 0 0

☐ Right Eye Brow (down/up) 0 0

☑ Smile 73%

☑ Mouth open 8%

☐ Eyes closed (left/right) 0 0

☑ Heart rate 101

Fig. 8. Face Expression sub-panel screenshot.

The panel will be developed in C# using the game engine Unity 5.5. To send data to game engines, we will use the protocol UDP with client-server model. As a proof of concept, will be developed, also in Unity and using C# , an endless runner game that will be adjusted to the player's emotions and physiological data using both SDKs.

4.1 Architecture

The RealSense Camera will send data to the two SDKs that will be embedded in EmoCam Panel (explained above). This panel is responsible for getting the information from the SDKs and sending it to RehabNet CP and to BL Engine, using a specific protocol, and export data to a CSV file(s). Then, the two tools will send the data to the video game, where it can be joined with data from other sensors/devices and then interpreted by the video game.

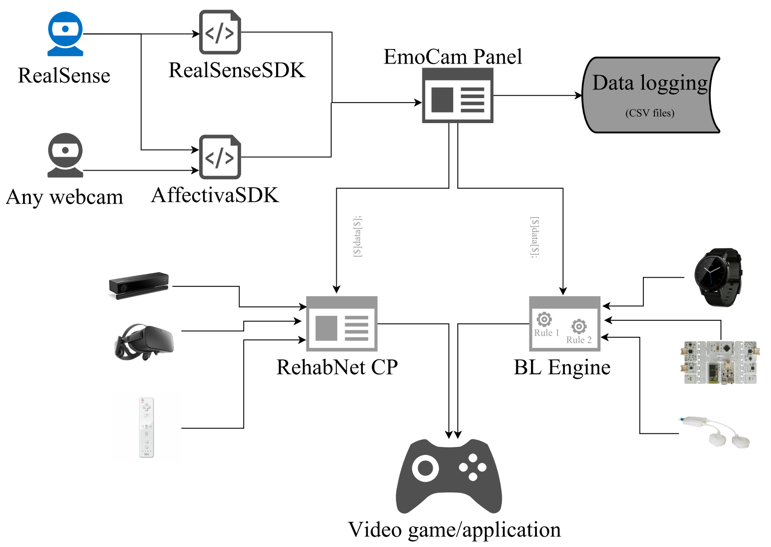


Fig. 9. System architecture.

Thus, process of data acquisition and transmission is easier, it can be integrated into any gaming engine (Unity, Unreal, ...), that supports UDP and it allows to create affective games. Besides this, into one solution it is possible to integrate data from gestures, facial expressions and emotions.

This panel has five files:

- ***ColorStream.cs***: to get the color stream form RealSense Camera;
- ***RealSense.cs***: to get the face and hand information from Intel RealSense SDK;
- ***PlayerEmotions.cs***: to analyze and to return user emotions from Affectiva SDK;
- ***Client.cs***: file that implements the client side of UDP, sending the information to an IP address following an already created protocol;
- ***UINavigation.cs***: file responsible for hiding/showing sub-panels.

4.2 General flowchart

In this section two flowcharts will be shown, one for each SDK.

4.3 RealSense SDK flowchart

Once the panel is started, it creates an instance of *SenseManager* to organize the pipeline by starting, stopping and pausing the operations of its various modalities, using this command: `PXCMSenseManager.CreateInstance()`; this implicitly creates a session internally. After this, it is needed to check whether or not *SenseManager* was instantiated.

The second step is to enable all the algorithm modules, for example: this command `EnableHand()` enables the *PXCMHandModule*, that tracks user hands.

Now, it is needed to update module's configuration and, for consequence, the internal configuration of the camera. Because the code is different to each module, in the dedicated section, that part of the code will be explained in detail. If all configuration is done properly, it is needed to initialize the pipeline using this command: `Init()`. To check if the pipeline was successfully initialize, this comparison was used: `_status != pxcmStatus.PXCM_STATUS_NO_ERROR`. The camera is ready to retrieve and show data through RealSense SDK.

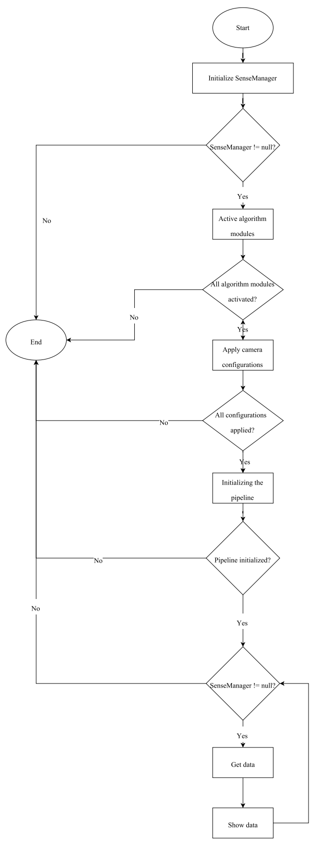


Fig. 10. General flowchart for RealSense SDK.

4.4 Affectiva SDK flowchart

To start analyzing user's emotions, the first step is to add a component, via Unity interface, that implements the detector. In this case, we added the detector to the scene's *Main Camera*. Because it is needed to track all emotions, on the configuration of the detector, in the *Emotions* select box, the option *Everything* was chosen.

Because the RealSense Camera can't stream data, at the same time, to both SDKs, it was not needed to add the component *Camera Input*. The solution that was found was to use the object image from the color stream object of the sub-panel.

In the script, the first step is to find the detector and then the camera stream object, using these commands, respectively: `FindObjectOfType<Detector>()` and `FindObjectOfType<ColorStream>()`. In the *Camera Input* component is needed to specify the sample rate, in this case, the sample rate was defined as a variable; then it starts a co-routine that calls every sample rate seconds (it was set to 20, this is, it is called every $\frac{1}{20} = 0.05$ seconds) a function that *takes* the image form the color stream's object.

After having the color stream, it will be passed to the detector that will analyze, each frame and retrieve the emotions value ready to show.

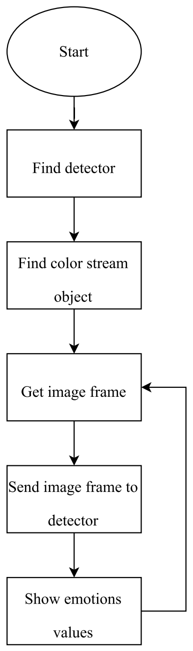


Fig. 11. General flowchart for Affectiva SDK.

4.5 Face Expression sub-Panel

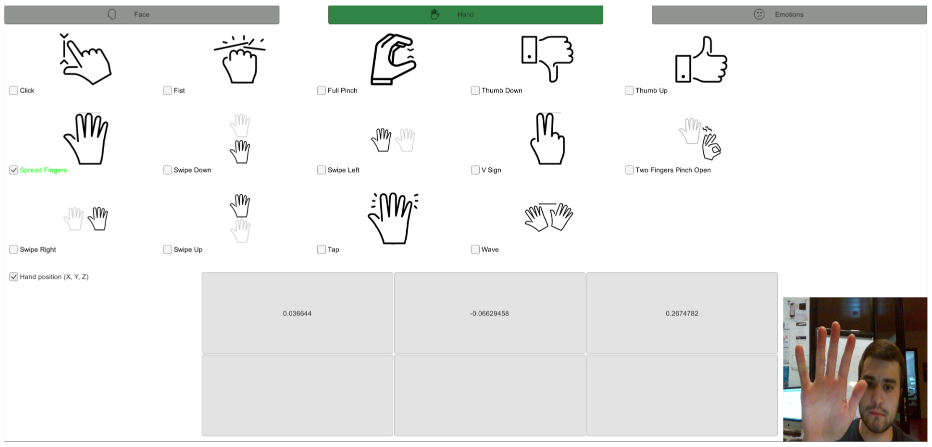
This is the sub-panel (image 8) responsible for retrieving all the information about the user's face, that can be traceable with RealSense Camera.

The first two items give information about the location of the eye brow, either left or right, being that, the left progress bar for the down level and the right for the upper level. It is also possible, in the following two items, to track the mouth's information like smile and how much is the user's mouth open. The next item gives information about eyes closure, in this case, the left progress bar is for the left eye and the right progress bar for the right eye. To conclude this sub-panel, the *Heart rate* item gives physiological information about the user, in this case, the heart rate.

Intel does not give detail information about how it is possible to track face information, but they use the same technique that Affectiva SDK uses (machine learning). To detect the heart rate, RealSense SDK detects color changes and uses those color changes to determine if a pulse happened.

Implementation: in each frame, there is a cycle that iterates through every face detected. Each face is identified by an identifier making it possible to retrieve information about a specific face. The `FaceData` interface manages the face tracking module output data, like pulse detection or expressions. Having output data, is very straight forward to retrieve information about the heart rate (`QueryHeartRate()`) and about expressions (`QueryExpressions()`). Each expression is identified by a constant, for example, `EXPRESSION_BROW_RAISER_LEFT`, to get the expression result for left eye brow upper level.

4.6 Hand-Gestures sub-Panel



☐ Click
☐ Fist
☐ Full Pinch
☐ Thumb Down
☐ Thumb Up
✓ Spread Fingers
☐ Swipe Down
☐ Swipe Left
☐ V Sign
☐ Two Fingers Pinch Open
☐ Swipe Right
☐ Swipe Up
☐ Tap
☐ Wave
✓ Hand position (X, Y, Z)

0.036644 | -0.06629458 | 0.2674782

Fig. 12. Hand-Gestures sub-panel screenshot.

The *Hand-Gestures* sub-panel aggregates all the information about the hands of the users. It will show a list of all gestures that RealSense SDK can detect. Similarly to the *Face Expression* sub-panel, to start/stop tracking a gesture it is necessary to check the checkbox of each gesture. If a gesture is detected, then return the value *True*, otherwise, *False*; the detection of the gestures acts like a trigger, giving always *False* and, at the moment that a gesture is detected, gives *True*. To simplify the use of the UDP already defined protocol, the *True* value was coded as *1* and the *False* value as *0*. Note that will be sent always *0* and, unless the respective gesture was detected, will be sent only one *1*. Once again, Intel does not give detail information about how RealSense SDK detect gestures.

The last section of this sub-panel is the hand position. RealSense SDK uses three types of coordinates system:

- **Image coordinates**: a two-dimensional system in pixels;
- **World coordinates**: a three-dimensional system that returns the geometry location of the detected hand(s) in World coordinates and in meters: x meters to the right of the sensor, y meters above the sensor and z meters away from the sensor;
- **Adaptive coordinates**: a three-dimensional system representing the relative position of each axis inside the adaptive bounding box boundaries.

Due to future possible application uses, in this sub-panel the world coordinates system was used. There are three text boxes, one for each axis: x, y and z. Please note that, RealSense SDK can identify either if it is a left or a right hand that is shown.

Implementation: in each frame, the hand module is retrieved using `QueryHand()` and it is updated with the current hand data. In this information, there is the number of gestures detected, then a cycle iterates through all the gestures that have his specific check box on. The procedure for the hand position is similar, but there is a `for` cycle that iterates through all the number of detected cursors (which means hands). For each hand it is possible to detect the body side (`QueryBodySide()`) and his World coordinates (`QueryCursorPointWorld().x`, in this case, for the x axis).

4.7 Emotions Recognition sub-Panel

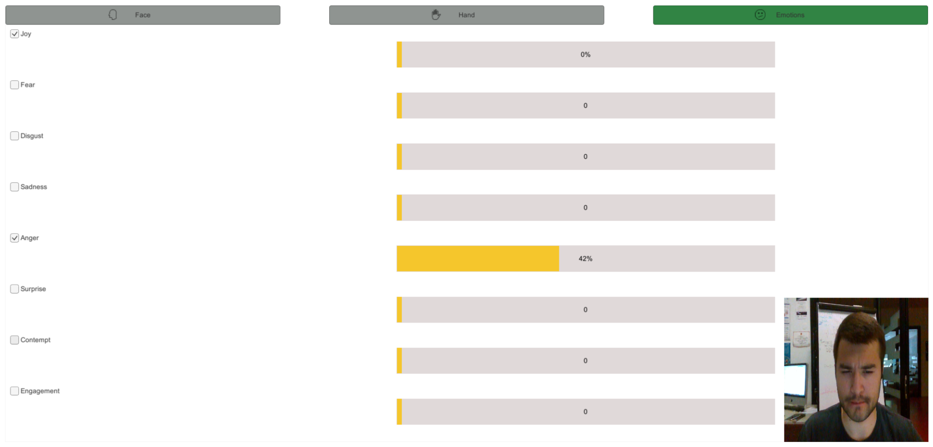


Fig. 13. Emotions Recognition sub-panel screenshot.

This is the last sub-panel and it has all the information about user's emotion, being the only sub-panel that works with Affectiva SDK. As other qualitative data, the emotions are measured and sent (UDP) in the interval $[0, 100]$. As explained later, there is eight emotions that can be inferred, has shown in the picture 13. The implementation for this sub-panel was already explained in the previous sections.

4.8 Color stream

As seen in the previous screenshots, there is always a space where the user can see the color stream of the camera. This helps the user to better adjust the camera position, the light exposure and the range of the camera.

Implementation: the color stream is made at a resolution of 640x480 and it is rendered in a specific object dedicated to that effect, but the color stream resolution can support up to 1920x1080 pixels. The SDK also gives access to left, right, and IR camera feeds. The stream type and the resolution are specified here, for example: `EnableStream(PXCMCapture.StreamType.STREAM_TYPE_COLOR, 640, 480)`; in this case, it will be a color stream at a resolution of 640x480.

The RealSense SDK is very consistent so, to get the color stream, it is necessary to call the function `QuerySample()` and update/set the texture of the object.

4.9 UDP Server

The User Datagram Protocol (UDP) is a transport layer protocol defined for use with the IP network layer protocol. The service provided by UDP is an unreliable service that gives no guarantees for delivery [2].

The implementation in C# is very straight forward, only two commands are needed: `IPEndPoint(IPAddress.Parse(sendIP), port)` - specifying the IP to send the data - and `new UdpClient()` to create the UDP client.

The data is sent within a protocol that was already defined and in use by the two tools. The protocol as the following format:

$$[\$]device\_type, [\$\$]device\_name, [\$\$\$]data\_type, data\_name, X, Y, Z, P;$$

Example of use (for the left-hand position):

$$[\$]tracking, [\$\$]realsense, [\$\$\$]handposition, lefthand, X, Y, Z, 0;$$

5 Summary

The following table, summarizes all the functionalities/features grouped by devices.

EmoCam Panel		
Face expressions	**Hand-gestures**	**Emotions**
Intel RealSense	**Intel RealSense**	**Any webcam (Affectiva SDK)**
Eyebrow position	Click	Anger
Smile	Fist	Contempt
Mouth open	Full pinch	Disgust
Eyes closure	Spread-fingers	Engagement
Heart rate	Swipe left	Fear
	Swipe right	Joy
	Swipe up	Sadness
	Tap	Surprise
	Thumb down	
	Thumb up	
	Two fingers pinch open	
	V sign	
	Wave	

Table 2. Summary of all the features.

6 Limitations

RealSense Camera, RealSense SDK and Affectiva SDK have some limitations:

– Drivers incompatibility: although in the minimum requirements it is mentioned that compatibility with Windows 10, the RealSense Camera's drive was not recognized by a computer with Windows 10, but the same computer with Windows 10 Pro, works. If the computer has connection to Internet thorough an Internet cable, the drivers do not work, but if it has connection through Wireless, it works;
– It is only capable of tracking one head and one pair of hands;
– The heart rate value is very unstable and sensitive to light changes;
– Detection hand gestures happens when the hand-to-camera distance is approximately 20 cm to 115 cm;
– Facial expressions and heart rate estimations can be done when the head-to-camera distance is approximately 30 cm to 60 cm;
– RealSense Camera only works with USB 3.0 that can output 5 V, if it doesn't, it is necessary to buy an USB hub;
– For Unity, Affectiva SDK can only infer the emotions of one user;
– Although both of the SDKs are very consistent, the community of developers is not very active yet.

7 Proof of concept

The last step to conclude the project was to develop a proof that the Emocam Panel works. To do this, we developed a game that was introduced in one workshop promoted by NeuroRehabLab and it is based in a *Endless Runner* video game.

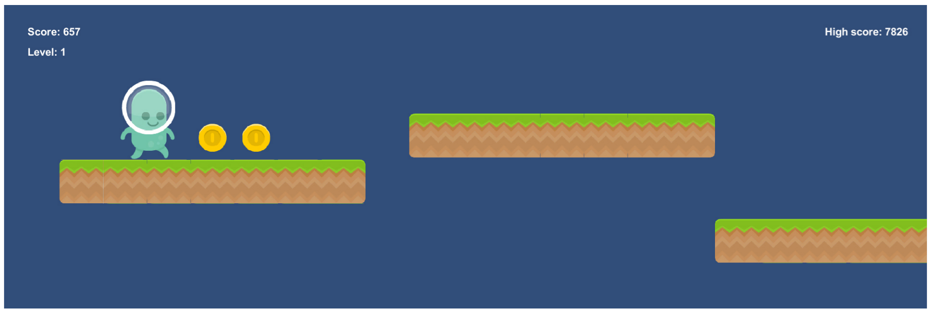


Fig. 14. Proof of concept - *Endless Runner*.

The main goal of this typology of game is to stay alive as long as possible; at each step that the avatar gives, the score increases, and the same thing happens when a coin is grabbed. If the avatar drops off from the platforms, that are generated randomly, the game is over.

Three different adaptive rules were designed and implemented in the video game using the EmoCam Panel:

- **Rule 1:** this rule is dedicated to measuring the level of attention of the player. At the end of each video game stage a mean is made of the level of a attention (or any emotion that can be choose by the developer). Then, this mean is divided by 100 and multiplied by the jump force and the size of the avatar. Increasing the jump force, makes easier to jump between platforms and increasing the size of the avatar, preventing him from dropping off the platforms;
- **Rule 2:** with this rule is possible to adjust the avatar's speed according to the user's heart rate. If the heart rate increases from the last section, that percentage is added to the avatar's speed, the same thing happens when heart rate decreases, that percentage is subtracted from avatar's speed. If the avatar's speed is too high, the game will be more difficult, because the platforms will be shown faster and the user will have difficulties to jump in the precise position. If the avatar's speed is too low, then the probability of the avatar to fall between two platforms will increase. So, the main objective is to maintain a certain level of heart rate;
- **Rule 3:** similar to the first rule, but this rule is dedicated to the happiness of the player. Once again, the developer can choose what happiness is for him, choosing a positive emotion to track (such as Joy and Surprise). If the mean of the emotion is higher than a specific threshold, then the probability of generating platforms with coins also increases, otherwise the probability decreases. Obviously, if the probability of showing coins is increased, the player is going to score more points.

With this proof of concept, it was possible to test the correct functioning of the two SDK's at the same time.

In this case, it was only used the RehabNet CP to stream the data to the video game. Note that, a game script that reads and receives the protocol is needed, and changes the variables according to what developer wants to track/control.

Although all the rules were initially tested, due time limitations, a playtesting session is presented as a future work.

8 Conclusion and future work

Emotion recognition systems which use non-invasive technology have the potential to augment human computer interactions via including affective information for both input control and system adaptation. In this direction, the EmoCam Panel, is a unique tool that can be used by any researcher or game developer that wants, for example, to transform a conventional hand-held video game interaction into an emotionally adaptive-game, making this transformation easy to implement.

The result of this project is a panel, that used simple webcams or the advanced and low-cost Real Sense Camera from Intel, for track and recognize users' expressions, emotions and physiological informations as well as hand gestures for a natural interaction. With the proof of concept, we test the feasibility of using such information to intelligently adapt an endless-runner video game considering player's emotions and physiological responses. Although the EmoCam Panel integrates the UDP communication protocol, it is not still fully integrated with the RehabNet CP and the BL Engine. Efforts should be made to integrate these tools in order to boost the development of novel affective serious games for health.

References

1. Affectiva: Mapping Expressions to Emotions (2017), `https://developer.affectiva.com/mapping-expressions-to-emotions`
2. Fairhurst, G.: The User Datagram Protocol (UDP) (2008), `http://www.erg.abdn.ac.uk/users/Gorry/course/inet-pages/udp.html`
3. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press (2016), `http://www.deeplearningbook.org`
4. Henry, J.C.: Electroencephalography: Basic principles, clinical applications, and related fields, fifth edition. Neurology 67(11), 2092–2092–a (2006), `http://www.neurology.org/content/67/11/2092.2.short`
5. Intel: Intel RealSense SDK Getting Started Tutorial
6. Intel: Intel RealSense Camera SR300 - Product Datasheet (2016), `http://www.mouser.com/pdfdocs/intel_realsense_camera_sr300.pdf`
7. L. Meiselman, H.: Emotion Measurement. Woodhead Publishing (2016)
8. Rao, M.: Intel RealSense SDK Architecture. Intel
9. Vourvopoulos, A., Faria, A.L., Cameirão, M.S., i Badia, S.B.: Rehabnet: A distributed architecture for motor and cognitive neuro-rehabilitation. In: 2013 IEEE 15th International Conference on e-Health Networking, Applications and Services (Healthcom 2013). pp. 454–459 (Oct 2013)