Multimedia Tools and Applications (2022) 81:8995–9012
https://doi.org/10.1007/s11042-022-12068-y

Applying color recognition techniques to achieve low-cost portable digital board functionalities

Alba Cotarelo[1] · **Jose Manuel Redondo**[1]

Received: 27 October 2020 / Revised: 21 September 2021 / Accepted: 4 January 2022 /
Published online: 9 February 2022

Abstract
Digital teaching support devices, like digital boards, are becoming increasingly popular to improve concept learning, as they allow manipulating represented objects in new and very innovative ways if compared with traditional boards. However, their acquisition cost (device, software licenses, additional hardware), and the lack of deployment flexibility to adapt to certain classroom configurations, are the main problems that prevents them to be used in a substantial number of educational environments or slow their adoption rate. This research shows that a color recognition algorithm, applied to projected images captured by consumer-grade webcams, can successfully provide functionalities equivalent to typical digital boards on low-budget educational environments. This way, a pointer object is recognized by its color, and its actions are interpreted at real time to be converted to standard UI actions in the computer screen, as a standard input device. The prototype has been also designed to be usable in classroom environments of different sizes, available space, or configuration, maximizing reutilization of existing elements in the classroom. The research prototype was successfully deployed in a real classroom using modest hardware, inexpensive requirements, and very flexible setup options, adapting both to student and teacher needs.

Keywords Digital board · Color recognition · Portability · Low-cost

1 Introduction

Digital media [12, 32] and teaching support devices are widely used in classrooms nowadays to aid teachers to better explain different types of contents. One of the most popular support devices are digital boards [25]. They are used in classrooms with students of a wide range of ages. Typical implementations use different techniques to enable certain "writing" devices (pointers) to achieve multiple effects when they touch their surface. Examples are painting colors, drawing lines or shapes, moving objects, or produce different graphical effects.

✉ Jose Manuel Redondo
redondojose@uniovi.es

Alba Cotarelo
UO251336@uniovi.es

[1] Computer Science Department, University of Oviedo, Oviedo, Spain

Interaction with objects represented in the board can be very rich, displaying concepts in ways that are impossible for traditional boards. These enhanced interaction possibilities can be used to reinforce learning, cognition, and motivation [1, 7].

Some boards are interactive and can display contents by themselves [22], acting like a multitouch TV screen. Others need external projectors and custom software running in a classroom computer [28]. This software manages the displayed content and the hardware to achieve optimal interaction and calibration. The board surface turns into something like a tablet device with a much greater display size, so contents can be viewed by all classroom students. However, using digital boards also have several shortcomings:

– *Cost*. Adoption rate of digital boards is increasing continually in the recent years, and this trend is expected to continue in the future [33] due to its general benefits [21]. However, cost issues make them to be adopted at a slow rate in some countries that cannot easily afford this kind of investment [5], especially in public schools [37]. This also leaves privately funded schools with a potential competitive advantage. Price of digital boards needing an external projector starts from 400€ [28]. Self-displaying interactive boards are more expensive. For example, *Promethean* (reference brand in this field) boards prices starts from 1000€ [22]. Some models require its own custom computer, further increasing implementation costs and preventing reusing an existing one in the classroom.

  For example, the school we used for our initial prototype deployment could only afford to have digital boards in roughly 60% of their classrooms, putting them in a competitive advantage. However, as the acquisition of these devices was done during several course years for budget reasons, some of them are beginning to become obsolete or started malfunctioning. Paying the maintenance of these devices is proving to be a challenge for schools, especially nowadays, as they must spend a substantial part of their budget money to maintain the mandatory COVID-19 protocols. Consequently, some of these previously acquired devices cannot be replaced or repaired on time, limiting their usage even if they are installed in a classroom. This reduces the effective number of digital boards available to students. To further increase this problem, cheaper devices (like the ones offered to schools by book publishers at a reduced cost) are reported to be less durable than those bought directly to the vendors. A device based on easy to acquire and cheap commercial hardware will greatly contribute to solve these maintenance and acquisition problems that several models of existing boards have.
– *Portability*. Digital board sizes are typically similar to traditional ones, both needing to be installed and properly secured in a permanent classroom. This means that, normally, students from other classrooms have to move to the one with the installed device to take advantage of it (and better justify the investment). This potentially compromises class scheduling, and it is also strongly discouraged in the current COVID-19 health crisis. However, not doing it puts some classrooms in a competitive advantage, as we said in the previous point. A device based in hardware that can be easily moved between classrooms will address this shortcoming of existing boards.
– *Physical space*. Digital boards are often a complement to traditional ones, and some classrooms may not have enough space for both. For example, a concrete device that was offered to several schools required the exclusive use of a classroom wall, which was not possible in several schools and also prevents portability (previous shortcoming). This device was also very expensive (roughly 4000€), further limiting its potential acquisition. The ideal solution to this problem of existing boards has to be based on small hardware and will reuse all possible elements already present in the classroom.

– *Hardware*. Software for digital boards could be designed to be installed in standard PCs. However, classroom computers may be very outdated, not meeting its minimum hardware / OS requirements. Thus, acquiring these devices without a careful planning may lead to not operational or limited functionality boards. Also, the impossibility of upgrading classroom computer hardware due to budget restrictions may also impede using many currently available digital board models in the market. To solve this shortcoming of existing board models, the solution must be able to work in the current typical hardware available in school classrooms, even if it is very outdated or has a limited processing power

Digiboard is a research project that aims to develop a technique that overcomes the described problems. It implements the typical point-and-click functionalities of a digital board, minimizing hardware costs by trying to maximize reusing existing equipment. This research targets classrooms with potentially very old PCs. It also assumes that the classroom has an image projector connected to the PC with a typical setup to display multimedia presentations, like *PowerPoints*. The only additional hardware requirement is a standard consumer-grade webcam, a very cheap device that can be easily moved to any location. The initial intended audience of this research project are students under 12 (up to primary school ages in Spain).

To emulate digital board functionalities our research prototype captures the projected image with the webcam. Then, it uses a custom color recognition algorithm to interpret an element of the captured image as the "pointer". It then maps the pointer location in the projected image to the PC screen coordinate system, and interprets the user behavior to convert it to a typical GUI user action, such as move, click, or drag and drop, that is performed on the computer OS. This way, camera recorded images are used to translate user movements into actions that have effects in the software currently running in the classroom PC. In other words, image recording turns into an input device, achieving the desired digital board functionalities.

The main contribution of this paper is to describe how the *Digiboard* prototype has been designed, built, and calibrated resolving all technical difficulties found. It also describes how it has been successfully deployed in real classrooms using the available hardware, enabling digital board features in a much cheaper, flexible, and usable way. The prototype was incorporated into the weekly classroom activities of a Spanish school that only had enough budget to acquire physical digital boards for some classrooms, achieving our initial research goals.

The rest of this paper is structured as follows. Section 2 describes related work using different devices able to capture user movements to translate them into actions, while Section 3 describes how *Digiboard* works. Section 4 deals with the prototype internal architecture, and the implementation challenges we had to solve are detailed in Section 5. Finally, Section 6 describes how we evaluated the prototype on a real classroom, while Section 7 describes the conclusions and future work.

2 Related work

As we said, this research uses a standard webcam to recognize colors in images, translating this to GUI actions. This decision was taken after studying other options that could enable us to achieve the same functionalities.

2.1 Wearable devices and smartphones

The first type of devices we considered to simulate pointer behavior were wearable devices or smartphones. Our previous research [38] demonstrated that they can be attached to the body to successfully track user movements. However, this previous experience also demonstrated that wearable devices are expensive and may be difficult to carry for the intended audience, making them not adequate to classroom environments. Smartphones are extremely popular (although not as common in the intended audience age ranges), but not commonly taken to classrooms. Installing custom software into private phones was also discarded due to the extra complexity it supposes.

2.2 Eye trackers

These devices are successfully used in different research areas (market research, human behavior in simulations (e. g. driving simulators), *Human Computer Interaction* (HCI), website testing, medical research, neuroscience, psychology...), as they have a very high precision if compared with other solutions. Additionally, there are also successful applications in learning and education [18]. They also enable the most intuitive and effortless way of emulating a digital board, using the students eye movement to place the pointer, as they only have to face the eye tracker device.

Tobii, a world-leading company in research-oriented eye tracking devices, produces a wide range of products, from glasses to camera-based devices. In the context of this paper, its most portable implementation, the *Tobii Pro Nano* [29] could be used to achieve seamless interaction with the screen. However, although this solution complies with our portability and physical space requirements, it is also very expensive (around 3500€, 11000€if we also purchase a software license). Therefore, it could not be easily acquired by most schools. This device must also be fixed to the classroom PC, which complicates its setup if it is not a laptop. Finally, its system requirements were very high for the typical PC present in classrooms of our country. Although cheaper and not research-oriented eye tracking devices exist, they also had a substantial cost, and the hardware requirements problem.

2.3 Gaming-oriented devices

We also studied adapting consumer-grade gaming-oriented image-capturing devices in the market to our needs. Although they were created for gaming, they are also used in academic research applications [36].

A very popular device is *Kinect* from *Microsoft*, that uses a special custom camera device with sensors to recognize elements in an image, also including its depth. Thus, it is able to capture a 3-D representation of the scene. This way, *Kinect* can return the 3-D coordinates of any object used as pointer, and depth, instead of color, can be used to determine if the user is interacting with a certain surface, also enabling gesture recognition. This principle allowed *Kinect* to be used for research [35], and it has been even adapted to educational applications [10].

The main problem of using *Kinect* is that we still had to develop an efficient algorithm to differentiate the pointer from other objects (especially if they are similar) and from the user body in a 3-D space. While this is technically possible, we also found that the "classic" *Kinect* has been discontinued and replaced by a more expensive new version, the *Azure Kinect* (400USD), very recently. This new device is aimed at enterprise users, covering

needs in markets such as logistics, robotics, health care, retail, and volumetric capture applications. However, its price is still very high, and its hardware requirements are again out of line with the PCs we expect to work with.

Another similar gaming-oriented device is the *Playstation Move* system. This device has also been used in different research projects and applications [24]. It is composed by a camera that records images while the user is using a custom controller in front of it. This controller has a glowing sphere on top that can change colors, so it is very easy to distinguish it from the other elements in the image. The glowing light is recognized as a pointer, and its movements can be easily translated to different GUI actions. As in *Kinect*, this system only works with programs (video games) that have been specifically designed to use it.

Unfortunately, *Playstation Move* is nowadays discontinued as a stand-alone product and bundled as a non-essential accessory for the newer *Playstation VR* technology. *Playstation VR* (300USD) is a consumer-grade gaming virtual reality technology that is also used for research applications [16]). It also uses a camera to recognize user movements and enrich the interaction with VR-enabled games, able to be displayed in its custom virtual reality glasses. Apart from the cost, something like this could not be easily deployed into a classroom due to lack of flexibility and, again, hardware requirement problems.

There are also PC-only solutions with the same approach that could be used in our scenario. For example, we can find the *Asus Xtion PRO Live* (200USD), a custom camera able to track movements and identify RGB colors. It is aimed to develop games and interactive applications, but it has been also used for research [9, 27]. All of these solutions have the same cost and hardware shortcomings as the described ones.

2.4 Non-specialized consumer cameras

After studying these devices, we decided that using non-specialized consumer cameras to record images and identify elements in them was the only viable way to achieve our goals. However, we first needed to ensure that these cameras can fulfill our needs. There are successful research projects using cameras for color-based object detection [17], including real-time moving objects [13] and images with similar colors [26]. This leaves which is the minimum resolution that the camera needs to fulfill our purposes as the only question to be answered. As we will describe in Section 3, our experiments demonstrated that, in our scenario, the precision requirements were not high, and even a basic (and very cheap) VGA webcam could be successfully used for our purposes.

Therefore, there are multiple device types able to recognize elements in images through cameras so their movement can be translated to user interactions. All of them could be used to implement our research but, in the end, the hardware and/or development kits cost, infeasibility of classroom deployment, or the lack of enough processing power on average classroom computers left webcams as the only suitable device that fulfills our needs. The specialized devices offer more advanced and powerful recognition functionalities, but they largely exceed what it is really needed for this research application.

3 Digiboard design

Digiboard is based on color recognition techniques over images recorded by a webcam. Instead of using existing research to perform color recognition (e. g. [23, 30]), we developed a custom algorithm that focus just on performance to be usable with very outdated hardware

(see Section 6). This is because we first developed an initial research test prototype of an *Air Piano* application. This was a testbed of the feasibility of efficient color recognition using home-made devices and images displayed on unusual surfaces. It used a ceiling-mounted webcam pointing to the floor, where piano keys were emulated using white paper sheets and black duct tape. This was installed in a public corridor, and people hovered cardboard shapes (with red duct tape strapped around) over the emulated keys. After fine-tuning the color recognition algorithm, the application successfully played the sounds corresponding to the appropriate key in a variety of situations. This experience was then carried over to this research. More technical details about this custom color recognition algorithm are covered in Section 3.1.

This custom algorithm needs an initial calibration procedure to set an "special" color that represents the user-controlled pointer. Once set, every time this color is recognized in an object present in the recorded images, that object is treated as the "pointing device". We designed this inspired by the *Playstation Move* custom controller idea (see Section 2.3) but enabling any object to be potentially used as a pointer device once its color is correctly set. As we saw in our *Air Piano* prototype, this allows to create inexpensive pointers, such as pieces of tape strapped to a finger or a simple paper ball at the end of a wood stick. This is was implemented to minimize *Digiboard* costs and to adapt to children classroom environments: different pointing devices can be built by the students with materials already available in the classroom as part of their activities.

Once the pointer color is set, gesture-recognition algorithms will interpret the user actions. The first *Digiboard* prototype modeled three basic ones: click on a location, simple pointer movement within the displayed image, and object drag-and-drop. This enables object manipulation the same way a traditional digital board does. The gesture-recognition software will run in the classroom computer with the camera attached, not requiring additional software.

Using the typical computer-projector setup to implement our system also facilitates changing the surface in which the image is projected. Portable projectors, or those with rotating mountings, allows to use *Digiboard* on any available surface (tables, walls...). This allows more flexibility than a typical digital board, especially in small classrooms or environments with restricted space. It is also easier to integrate it in the class routine. As the information source to interpret actions is the captured image colors, the surface material is not a variable to consider. However, apart from color, the usable size of the projection must be also calibrated. Next section describes calibration in detail.

3.1 Calibration procedure

The initial calibration procedure has two steps: (1) establishing a bounding box and (2) set the pointer color. The first step deals with the actual size of the image displayed by the projector. Setting up a bounding box is necessary to know the webcam position relative to the projection. This way, it does not matter if the webcam records much more than the actual projection (See Fig. 1) or cuts the image that the projector displays: the bounding box will be the only area considered to identify user actions. Figure 1 shows how the user is expected to specify the bounding box by just clicking and dragging the mouse forming a rectangle.

Once the bounding box is established, the next step is to set the "pointer" color we mentioned earlier. Our experience revealed some limitations on color recognition that we must consider in this step:

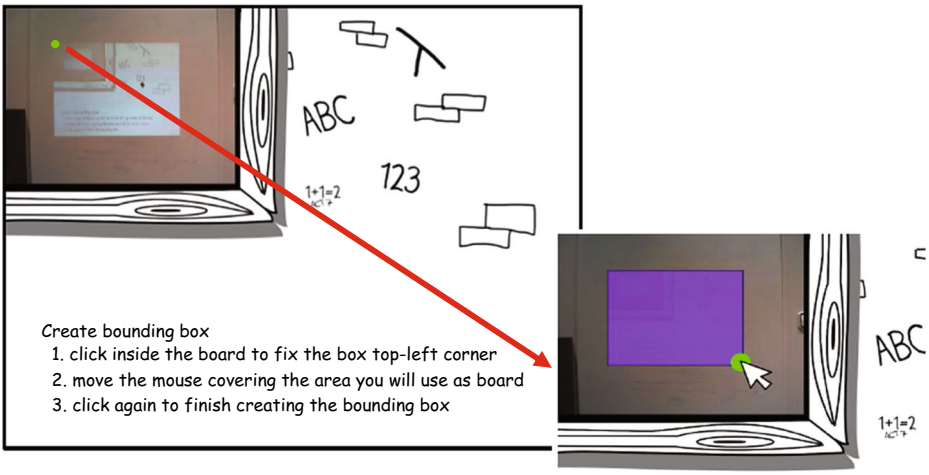

Fig. 1 Specifying the initial bounding box in the *Digiboard* calibration

- There are colors that are more easily recognized than others depending on the classroom light conditions. After experimentation, we found that red usually works best, although any color can be used.
- The size of the pointing device is important. A very small object could not be properly recognized in the recorded images. Fortunately, tests with objects like adhesive post-it notes, or small balloons strapped to wooden sticks were successful, so this turned out not a specially restrictive issue in our context.
- Persons (or objects) in the camera view port cannot wear clothes of be of the same color as the pointing device, as they could be recognized as additional pointers, generating false interactions. This limitation can be easily fixed by moving objects, the camera, or choosing a different pointer color, for example.

Specifying the pointer color uses the same approach as the bounding box: click in the camera image so the color in the clicked area will be used. We discovered that this process also had some technical problems, and its solution will be described on Section 5.

This two-step calibration process is just needed once if the installation conditions do not change. Recalibration could be necessary if, for example, the camera is moved to a different place (another bounding box will be needed). Also, if room lightning conditions change drastically (e. g. *Digiboard* activities are not made at the same hours of the day, the classroom artificial lightning is switched off/modified...), or a different pointer color is needed, a new calibration process will be required.

3.2 Digiboard functionality

Once calibrated, *Digiboard* can start to be used as a typical digital board. Pointer movements will be interpreted and converted to typical GUI actions that will be mapped from the bounding box area to the actual computer screen. As we said, the first prototype implemented three actions:

- *Movement*: If you move the pointer object through the bounding box displayed area, this will be treated as moving the mouse on the screen.

- *Single click*: Stopping the pointer into a position for a small amount of time is equivalent to left clicking into that position (a "press" event is generated). "Press" and "release" events are also notified with a sound (apart from the GUI changes produced by the click).
- *Drag and Drop*: This is an extension of the previous action. If the user decides to move the pointer just after the "press" sound is played, then the clicked object will be dragged through the image projection. When the user stops moving the pointer, a "release" event is generated producing the drop action, also playing the corresponding sound notification. Single clicks are distinguished from drag and drops if no movement in the pointer is produced during a certain time threshold after the "press" sound notification. More details about how this was implemented can be found in Section 5.4.

Apart from these typical actions, the application can be minimized to continue working in the background, so the teachers can interact with other applications to continue their class. *Digiboard* does not need exclusive access to the screen to work.

This way, our research maps user actions to standard *Windows* events, as the ones that can be produced by any typical input device. This allows us to interact with a great deal of applications, either educational or not, as for the OS *Digiboard* is just another input device. Section 6 details some of the application sources we successfully used.

4 Digiboard architecture

The application architecture is divided in three main blocks (see Fig. 2). Each block has a defined purpose to simplify maintenance. These are:

- `DigiboardUI`: stores all classes related to the *Digiboard* user interface. These classes control the different execution phases we described in the previous section and simulate the navigation and interface elements of the application. This includes buttons and toggle buttons we use, for example, to let the user activate or deactivate sound notifications.
- `BoardManager`: this module contains all classes related to bounding box creation, color selection, and recognition. They are used to select the pointer color and to track pointer movements along the bounding box. Therefore, this module contains the core functionality of *Digiboard*, and the more technically challenging functionalities (see Section 5).
- `MouseService`: provides all functionalities related to mouse movement and clicking on screen elements. It also processes mouse position to translate coordinates from camera images to the computer screen according to their current resolution (see Section 5.1).

4.1 Frameworks and tools

To implement *Digiboard* we not only needed a typical consumer webcam, but also multiple software libraries and frameworks. Regarding the camera, after several tests we found that using just a standard VGA video capture resolution (640x480) was enough to achieve our goals. This resolution can be achieved by any camera in the market, and even by cameras

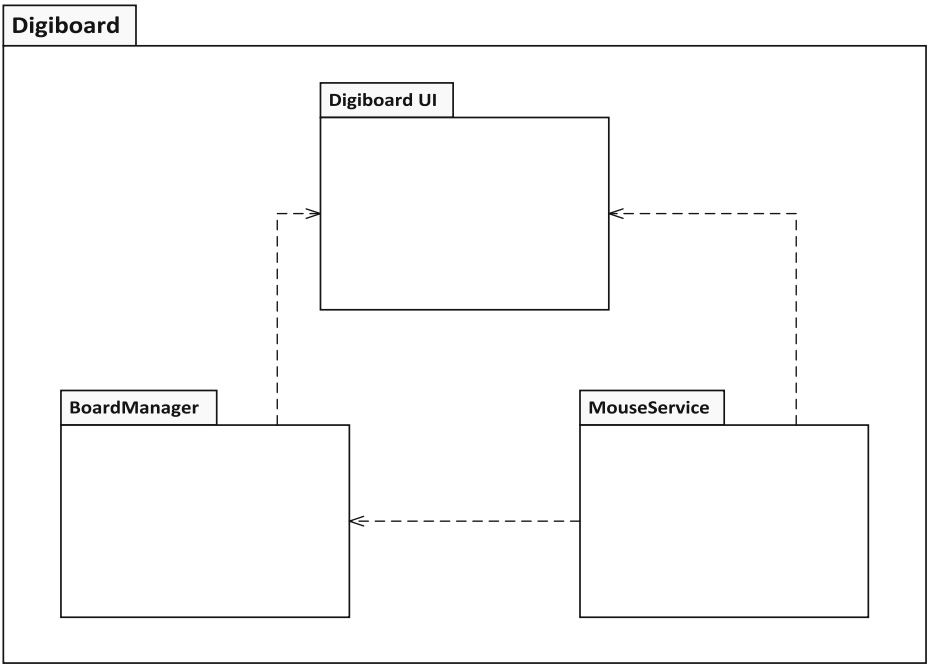

Fig. 2 Application architecture

bought several years ago. This is convenient when processing power is very limited (see Section 6), as image processing at low resolutions can be done with an acceptable response time much easier. It also saves costs, as almost any working camera can be reused for this purpose.

Regarding libraries and frameworks, this project is developed with the Java-based `Processing 3.5.3` [31] for *Windows*. It is a free open source framework focused on visual arts. *Processing* programs run by iterating over a main method that draws each frame over the previous one and provides methods for capturing mouse or keyboard events. We also used some additional libraries:

- `Capture,ProcessingRef`: Stores and manipulates image frames recorded by a camera device. We used it for capturing webcam images and manage each frame pixels to detect which ones has the chosen color pointer (see Section 5.2).
- `ControlP5` [34]: This library provides interface control elements, such as buttons and toggle buttons, that were used to create the *Digiboard* UI.
- `SoundCipher` [3]: provides musical notes with different pitches among other features. Sound notifications are implemented with this library.

Several special `Processing` data types were also needed to properly create interface elements, such as `PFont`, `PImage`, and `PVector` (to simplify the code dealing with screen coordinates). All of them are described as part of the `Processing` reference documentation [14]. Finally, the Java `Robot` class [20] emulates the mouse pointer movement and interaction in the computer (the three GUI actions described in Section 3.2).

5 Research prototype implementation challenges

The implementation of some classes of the `BoardManager` package offered several technical challenges. This section describes them and their solutions.

5.1 Coordinate mapping

As we described, *Digiboard* converts the coordinates obtained from the captured image inside the calibrated bounding box to the corresponding ones in the computer screen. We process camera images frame by frame, so the system returns pointer coordinates in real time. We found several problems to get the right formula to translate bounding box to screen coordinates.

Figure 3 shows an example of a bounding box captured by the camera that is smaller than the actual PC screen size. Let us suppose that the bounding box resolution is 20x13 pixels. To start translating coordinates from the bounding box plane to the screen plane, we enclosed the bounding box in a point matrix, as shown in Fig. 4. This way, if the user wants to select the `Notepad++` icon, the system needs to detect the pointer clicking at the coordinate (4, 5) of the bounding box (see Fig. 4).

The computer monitor resolution will be bigger than the camera, so this (4, 5) coordinate must be mapped to a corresponding equivalent "click zone" in the actual monitor screen. If, for example, we suppose that the PC monitor resolution is 30x17 pixels, the system is expected to move the mouse to one of the coordinates with dots in Fig. 5: (6, 6), (6, 7), (6, 8), (7, 6), (7, 7) or (7, 8).

We need a formula able to convert the camera coordinates into computer screen coordinates, so the mouse pointer moves were the user is expecting. We implemented one based in proportions and distance to make it very simple and efficient, so it could work on real-time scenarios with potentially very limited processing power. This way, we added to the original coordinate a percentage of itself. This percentage is obtained from the distance the

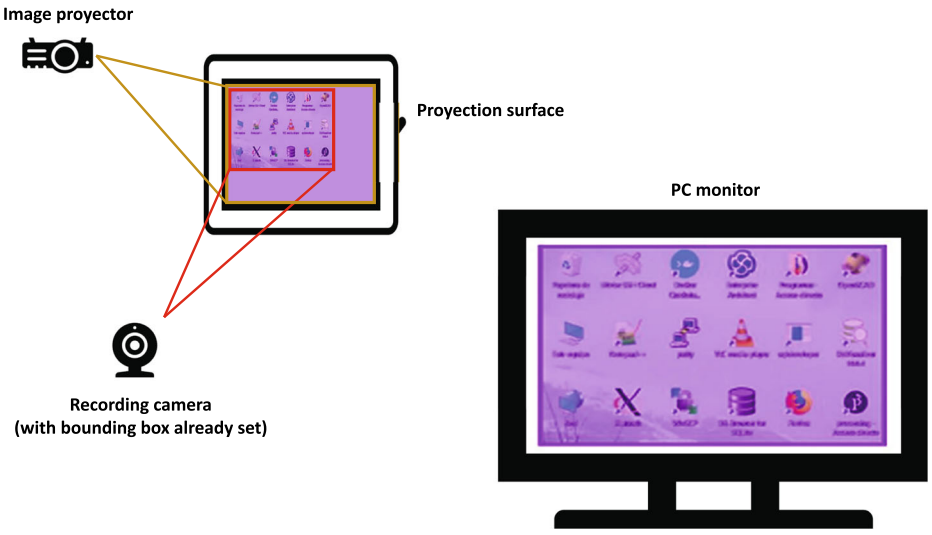

Fig. 3 Bounding box resolution VS actual screen resolution

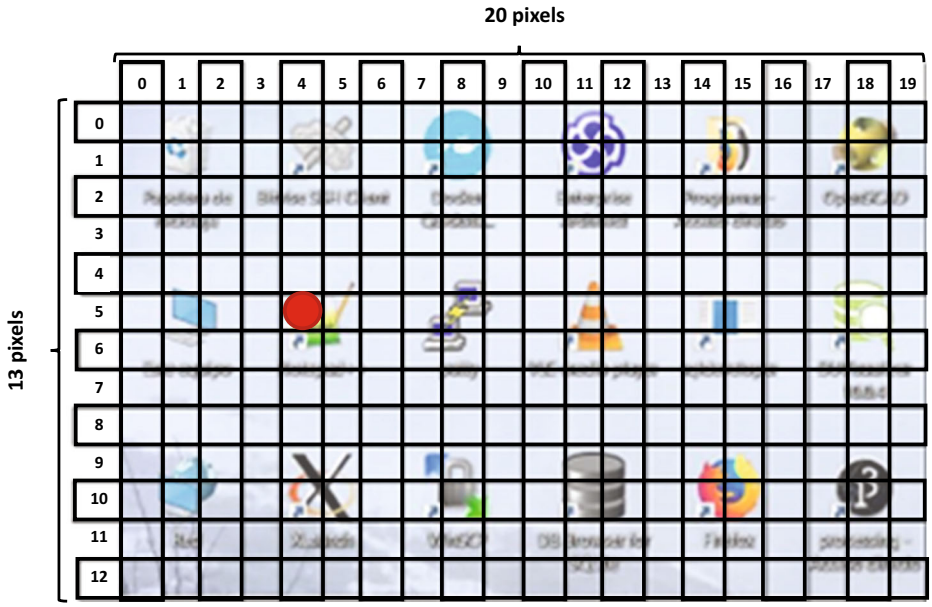

Fig. 4 Bounding box point matrix

coordinate must increase related to the original resolution camera. If (X_0, Y_0) is the coordinate where color pointer has been detected, (X_w, Y_w) the resolution of the camera and (X_s, Y_s) resolution of the monitor, the formula is:

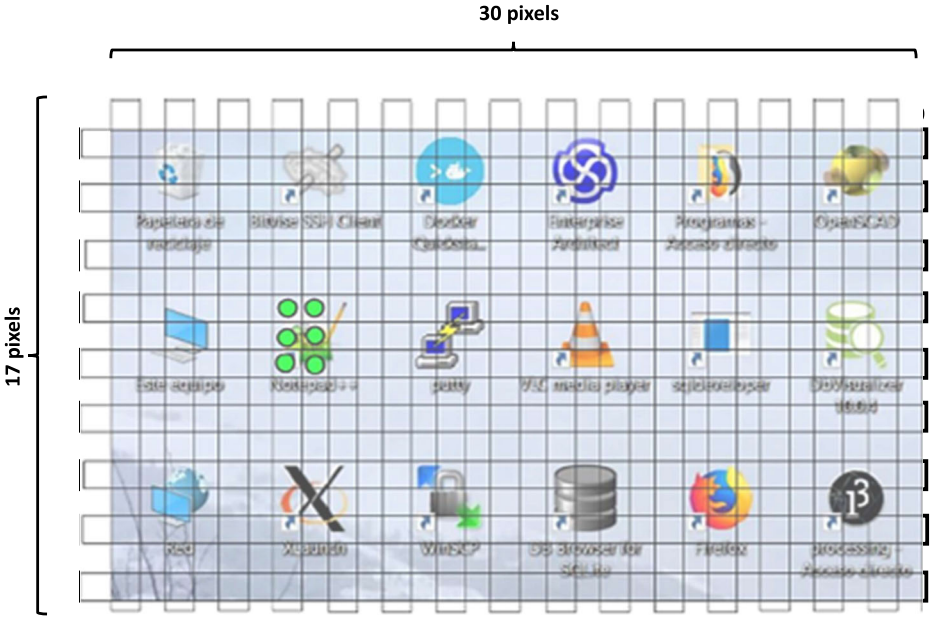

Fig. 5 Pointer mapped to the screen resolution

- New X coordinate: $X = X_0 + X_0 \times \frac{|X_w - X_s|}{X_w}$
- New Y coordinate: $Y = Y_0 + Y_0 \times \frac{|Y_w - Y_s|}{Y_w}$

So, the new coordinates for our example would be (6, 7) and the system would click on that pixel. This matches one of the coordinates we predicted.

- New X coordinate: $X = 4 + 4 \times \frac{|20-30|}{30} = 6$
- New Y coordinate: $Y = Y_0 + Y_0 \times \frac{|13-17|}{13} = 6.538 \simeq 7$

Unfortunately, this did not work at first with real resolutions. Tests found that the more the mapped pixel was away from (0, 0), the bigger the mapping errors were. Fortunately, we found that this problem does not occur if the camera and PC resolutions are proportional. We decided that he first prototype will enforce this proportionality, so the previous formula can be used to translate coordinates efficiently. As we will see in Section 6, this was no issue in our deployment. Although camera and PC were displaying different resolutions, setting proportional ones was very easy.

5.2 Light dependency

The light in the room was found to be a very important factor to correctly recognize pointer colors. To recognize them properly, the bounding box area must receive a regular amount of light in all its surface. This means that if we set up the color pointer in a specific place with a concrete light intensity, the system would only recognize that color in areas with a similar amount of light. If this problem is not properly handled, we could find that the mouse pointer is lost in certain moments, so it is possible that certain user actions may not be properly captured depending on room lightning conditions.

To solve this, we implemented a double color pointer setting, storing two tonalities of the same color, `A` and `B`. During the calibration phase (Section 3.1), the user is prompted to detect the color pointer in one corner of the bounding box. Then, it is told to move the pointer to the opposite corner, that normally have the highest probability of receiving a different amount of light, detecting it again. Pointer color is then recognized taking both values into consideration (see next subsection). Our tests found that this effectively increases the accuracy of the pointer recognition to acceptable values, even when both tonalities of the color are clearly different due to the lightning conditions.

5.3 False positives recognizing the pointer color

So, the calibration process requires using two tonalities of the pointer color, `A` and `B`, to perform an appropriate color detection. This calibration process has two filters to improve efficiency. The first filter uses both color tonalities independently, determining if the detected color is close enough to `A` or `B`. An adequate threshold was established via experimentation. This way, only a handful of image pixels passes it and are processed by the second filter. This one uses the three RGB components of the colors [2] to subtract the color of the remaining pixels from `A` and `B`. For each pixel Xf_1 that passed the first filter, $A - Xf_1$ and $B - Xf_1$ are calculated obtaining a series of values Zf_2. The pixels with the pointer color are those that have lowest values of Zf_2. This way, the coordinates of the pointer in the projected image are determined.

Our experimentation with this two-phase pointer location filtering process found occasional recognition errors on certain room conditions. This causes the pointer to become a

little uncontrolled. They happened because sometimes the pointer color was recognized in coordinates far from the ones in which the pointer was recognized in the last analyzed frame.

To solve this problem, we treated this as a false positive. Before moving the mouse, the system will calculate the distance from the last detected pointer location to the newly detected one. If the destination is considered "very far" (a threshold was also established beforehand via experimentation), the system discards it: the mouse does not move, and the next frame is analyzed. Our tests showed that this solution solves the problem without compromising user interaction with the displayed screen.

5.4 Clicking

A traditional digital board usage requires the user to perform clicks over the elements it displays, using various techniques depending on the nature of the board. Therefore, *Digiboard* must clearly distinguish user clicks from user pointer movement. To obtain the desired clicking effect, *Digiboard* recognizes clicks when the pointer remains static in the same place for a certain short amount of time. This is because recognizing a color and immediately treating it as a click would make objects to be constantly dragged along the board surface. Leaving the pointer static a very short amount of time models a "tap" gesture that represents clicks in an intuitive way, and without using extra hardware or requirements that will make our project more expensive or difficult to deploy.

This "tap" gesture is inspired in the behavior of typical videogames programmed for the *Eye Toy* or the *Kinect* (see Section 2.3). To implement this "tap-to-click" gesture, *Digiboard* only fires the "press" action in the corresponding coordinates of the computer UI when a few pointer color recognitions are detected in a very short amount of time. These consecutive color recognitions must be also detected in very close coordinates (a threshold was again established beforehand). Our evaluation with real users (next section) showed that this gesture was intuitive and well received by the users.

6 Evaluation

Once our internal tests finished successfully, we deployed the *Digiboard* prototype in a real classroom of a Spanish school (C. P. Ventanielles [11]) to see how students and teachers use it. We obtained permission to deploy it in a classroom of 22 children with ages in the 6-7 range. Due to the flexibility of *Digiboard*, one of the spare tables in the classroom was used to place the existing portable projector. We also bought a `Logitech c170` webcam (23 euros), placing it on top of the projector to finish the setup. Note that the webcam was the only cost required to use *Digiboard* in this school, as the other elements are very common in most classes. This school could not provide a webcam for our tests, but in case there were some available, the prototype could have been used without any cost. Figure 6 shows a schema of the layout we used.

To complete the required elements to start out tests, we created the pointer with a ruler and red duct tape that were already in the classroom. Figure 7 shows this color pointer being handled by two of the classroom students. Finally, we set the camera resolution to 640x480 and the classroom PC resolution to 1280x960 (proportional, as indicated in Section 5.1).

Unfortunately, the *Digiboard* prototype could not be used in our first attempt as, the classroom PC was more outdated than we have initially predicted. This PC had an `Intel Pentium D 960` CPU (2 core, 3.6Ghz, 2006) [6], a 500GB HDD, 4GB of RAM, and

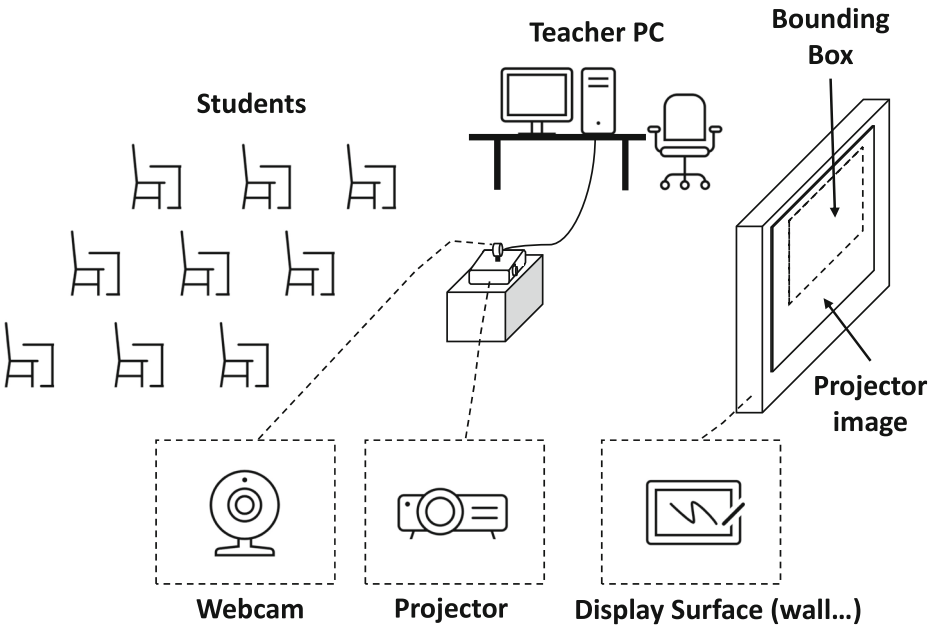

Fig. 6 *Digiboard* testing deployment schema on a real classroom

Windows 7 Professional. This OS was also a 32-bit version, which rendered 1GB of the system RAM not addressable.

Finding a 14-year-old computer with a 32-bit OS in service in this classroom forced us to resolve some additional technical challenges to be able to finally deploy the prototype, as we were told that the same setup was also used in several other schools in our province. The `Processing` Java library (see Section 4) was the first one, as it was compiled for 64-bit architectures. Fortunately, it has the option to be recompiled to 32 bits, so a proper version could be generated for our prototype. Once the prototype could be started, we found that this 32-bit version has excessive RAM consumption problems. To solve them, we needed to recompile *Digiboard* again incrementing the RAM available to `Processing`, as this is a compile-time option. Further testing showed that removing a background image we had in the GUI was also necessary to reduce the memory footprint even further. Once these adjustments were made, we were able to run the prototype correctly on that hardware.

Once deployed, we selected potential activities that could be used with *Digiboard*. It successfully enabled us to work adequately with many educational games or applications that were created for physical digital boards, achieving one of our main objectives. Applications like *Google JamBoard* (which also sells their own software [19]), and others that were kindly referenced by multiple members of the educational community in Spain that liked our approach, could be correctly used. One of the most important application sources referenced was *Mundoprimaria* [15], that contained many applications available for the intended audience. Several of them were not specifically designed for digital boards, but as *Digiboard* behaves like a traditional input device, *Digiboard* could be successfully used to interact with them. Figure 7 shows an actual classroom deployment of *Digiboard*, interacting with

Fig. 7 A *Mundoprimaria* application being used in *Digiboard* (Source: *Digiboard* TV Report of the TPA (*Televisión del Principado de Asturias*) [8])

an application from this repository. In the end, thanks to *Digiboard*, this classroom could now access to additional educational materials they could not use before, at a minimal cost.

This real classroom deployment was initially thought as a testing ground to identify and address errors or problems we could not predict during development, like the hardware-related problems we described. However, the reception of the prototype was very successful, both from teachers and students. Because of it, the school asked us to leave the working prototype in the classroom and incorporated it as part of the student daily routine, so it became fully operational.

Digiboard usage was scheduled as a 1-hour weekly activity in the student timetable. This period was used to review concepts explained on several classes during the week, using *Digiboard* to provide more advanced examples via the applications of the repositories we mentioned. Apart from this, the *Science* teacher also incorporated *Digiboard* in their classes to do some exercises provided by the classroom book publishers targeted to digital boards. It is important to remark that these materials could only be used on that classroom thanks to *Digiboard*, as it lacked a physical one. This way, we prevented leaving them unused because the inability to deploy them.

This continuous usage allowed us to gain substantial feedback and adjust the prototype using the input provided by students and teachers, improving the overall product. The successful deployment was also picked by a local TV channel [8] that made a report about it (see Fig. 7), which also motivated us to make a promotional video that had a moderate success in social networks. This classroom deployment started on December 2019 and was abruptly interrupted by the closure of schools in Spain due to the COVID-19 health crisis at the beginning of March 2020. Unfortunately, the complications derived from this forced change to remote teaching also supposed that we could not provide to the different project stakeholders (such as students or teachers) a survey to acquire quantitative feedback about the experience working with our prototype, that could have been used to better identify its stronger and weaker points and to develop future improvements. The authors plan to resume the acquisition of quantitative feedback once the classroom conditions are closer to the previous state and the school grants us the required permissions.

Qualitative feedback collected during that period was positive. The *Science* teacher reported that he could successfully complete the digital board activities we described. The classroom main teacher also reported that the students were more motivated to complete concept review exercises during the scheduled 1-hour weekly review classes. Teachers also required a small initial training before they could successfully use *Digiboard*, but this could be easily solved creating a "how-to" zine [32] and a list of common Questions & Answers elaborated from the questions made by the teachers during *Digiboard* deployment.

Teachers also reported that students adapted to *Digiboard* usage very quickly. System reaction time was considered adequate for the abilities of children of the target age. It was also remarked that children required some concentration and patience to get used to the device at the beginning, but that turned helpful to improve their concentration abilities and motor/coordination skills. Since these tests were made in hardware with very limited processing power, reaction time should improve if newer machines are used. Collecting the system reaction time is a very useful feature that will be incorporated in the future to profile the application and detect potential problems.

Students reported that one of the most successful features was the sound notifications played when actions take place. *Digiboard* has a different sound when the user clicks than when the user drops (if a drag and drop action is detected). All children reported that they immediately got used to these different sound notifications, so they know when any action is produced, and therefore do not lose track of what is happening in the activity. Finally, there were no problems reported with colors of children clothes and the color pointer, as children adapted to not to wear anything of the chosen pointer color on their "Digiboard day" (see Fig. 7).

7 Conclusions and future work

Our research demonstrates that consumer-grade webcams and color recognition algorithms in recorded images can be effectively used to emulate the behavior of a digital board, even in very old hardware. This is done identifying pointer movements and translating them to typical user UI actions. This allows to successfully emulate the behavior of a traditional digital board, but at a minimal cost and with a greater degree of flexibility. Thanks to this, more students could use interactive activities that otherwise were inaccessible due to cost or lack of space considerations, thus achieving our research objectives.

Future work will expand the initiative to two more educational centers that have shown interest on it, complementing the feedback we obtained from the initial one. Also, four more teachers all over Spain have expressed interest on deploying the *Digiboard* prototype in some of their classrooms, once the COVID-19 health crisis allows to do so. This will allow us to gain more user feedback to improve the project. We are also considering some technical improvements, such as trying to remove or ease the requirement of putting the PC resolution proportional to the webcam, add an explicit GUI zone to make pointer color selection easier, and implement more typical digital board actions.

Among these future improvements, one of the more interesting ones is the ability to record the work of students while interacting with *Digiboard*. As the webcam is recording the image enclosed in the bounding box that was previously selected, the prototype could record each video frame of the ongoing session adding to each one the current coordinates of the pointer (as calculated in Section 5.1). The coordinate representation can be different for movement, clicks, and drag and drops, so the prototype can obtain a video of all work and

actions performed by each student. The interface can also be easily modified to incorporate a "Begin Record" button, and the prototype can also incorporate a "Record by default" option to make this feature easier to use.

Finally, although the COVID-19 health crisis interrupted our prototype deployment, it also gave us more opportunities to work with this research in the future. We are exploring ways of using the *Digiboard* functionalities to enable remote teaching applications, like the ones that are possible with *Google JamBoard*, [4]. We are also exploring the feasibility of using a TV screen instead of a projected image. This way, teachers may be able to share their screen in applications such as *Zoom* or *Teams*, showing an educational application that the student must interact with. Once this is done, a properly calibrated webcam at the student home will interact with the TV screen, and the student actions will be sent remotely to a remote *Digiboard* UI action service in the teacher computer that will reproduce student actions remotely. This way, the teacher could be able to see the results of the student interactions at home from the classroom. This could be very useful in scenarios where part of the students will be at home during certain time periods, as happened in our country due to the COVID-19 health crisis.

Acknowledgements The authors wish to thank the principal, management staff, and teachers of the *C. P. Ventanielles* public school (Spain) for allowing us to make the test deployment of our prototype in their classrooms, and giving us very valuable feedback to deploy this research work successfully.

Funding This research was not supported by any funding sources.

Code Availability Current prototype is closed source, but we are evaluating other options in the future.

Declarations

Conflict of Interests The authors declare that they have no conflict of interest.

References

1. Abdullah A, Mokhtar M, abu samah N, Mohamad Ashari Z, Jumaat N, Farzeeha D, Mun SHM (2019) Active learning using digital smart board to enhance primary school students' learning. Int J Interact Mob Technol (iJIM) 13:4–16. https://doi.org/10.3991/ijim.v13i07.10654
2. Augustine D, Jose S, NT N (2015) Comparitive study on color recognition methods. Int J Comput Sci Trends Technol (IJCST), 3
3. Brown AR (2020) SoundCipher: A music and sound library for processing. explodingart.com/soundcipher/
4. Classroom NE (2020) How to use Google JamBoard for remote teaching. www.youtube.com/watch?v=S9m4HCjOkcA
5. Cohen H (2019) IoT in education: The smartboard. www.iotforall.com/iot-education-smartboard
6. Corporation I (2020) Intel ARK. ark.intel.com/
7. Davidovitch N, Yavich R (2017) The effect of smart boards on the cognition and motivation of students. Higher Educ Stud 7:60. https://doi.org/10.5539/hes.v7n1p60
8. del Principado de Asturias G (2020) Radio Televisión del Principado de Asturias. www.rtpa.es/
9. Diddeniya I, Wanniarachchi WK, Gunasinghe H, Premachandra C, Liyanage J (2020) An autonomously guided differential drive robot base using Asus Xtion Pro Live. In: International conference on image processing and robotics
10. Dosil AG (2013) Design and implementation of an educational game control module through Microsoft Kinect. Master's thesis, Universidad Carlos III de Madrid
11. Educastur (2020) Colegio Público Bilingüe Ventanielles. alojaweb.educastur.es/web/cpbventanielles
12. Evans J (2019) Your Linux Toolbox, vol 1. Starch Press, San Francisco

13. Faek F, Elias V (2015) Real time motion and color detection. J Zankoi Sulaimani 17:195–204. https://doi.org/10.17656/jzs.10437
14. Foundation P (2020) Processing library reference. processing.org/reference/
15. Gesfomedia S (2020) Mundo Primaria, el mayor portal de material educativo gratuito. www.mundoprimaria.com/
16. Habgood J, Wilson D, Moore D, Alapont S (2017) HCI lessons from PlayStation VR. In: Proceeding CHI Play '17 extended abstract, pp 125–135. https://doi.org/10.1145/3130859.3131437
17. Hunud H, Mohd Mustafah Y (2013) Colour-based object detection and tracking for autonomous quadrotor UAV. IOP Conf Series Mater Sci Eng, 53. https://doi.org/10.1088/1757-899X/53/1/012086
18. IMotions (2017) EyeTracking: The complete Pocket Guide. IMotions
19. Inc G (2020) Google for Education: Bring learning to life with JamBoard. edu.google.com/intl/en/products/jamboard
20. Inc O (2020) Java Robot class. https://docs.oracle.com/en/java/javase/15/docs/api/java.desktop/java/awt/Robot.html
21. Kaur D (2018) How smart class technology is benefiting education sector. www.entrepreneur.com/article/322587
22. Ltd P (2020) Promethean world. www.prometheanworld.com/
23. M P P, Hema C, Krishnan P, Radzi S (2009) Color recognition algorithm using a neural network model in determining the ripeness of a banana. In: Proceedings of the International Conference on Man-Machine Systems (ICoMMS)
24. Magnuson M (2011) Measuring the earth's rotation with a Playstation Move controller. The Physics Teacher 49:254–254. https://doi.org/10.1119/1.3566048
25. Mun SH, Abdullah AH (2016) A review of the use of smart boards in education. In: 2016 IEEE 8th International Conference on Engineering Education (ICEED), pp 120–125. https://doi.org/10.1109/ICEED.2016.7856056
26. Navada B, Kv SSP, Shetty H (2014) An image processing technique for color detection and distinguish patterns with similar color: An aid for color blind people. In: Proceedings of international conference on circuits, communication, control and computing, I4C, p 2014. https://doi.org/10.1109/CIMCA.2014.7057818
27. Nock C, Taugourdeau O, Delagrange S, Messier C (2013) Assessing the potential of low-cost 3D cameras for the rapid measurement of plant woody structure. Sensors (Basel Switzerland) 13:16216–33. https://doi.org/10.3390/s131216216
28. pizarras digitales (2020) Pizarras digitales y pantallas interactivas. www.pizarras-digitales.com/
29. Pro T (2020) Tobii Pro Nano. www.tobiipro.com/es/products/tobii-pro-nano/
30. Rachmadi RF, Purnama IKE (2015) Vehicle color recognition using convolutional neural network. arXiv:1510.07391
31. Reas C, Fry B (2014) Processing, A Programming Handbook for Visual Designers and Artists, vol 1, 2nd edn. MIT Press, Cambridge
32. Redondo López JM (2021) Improving concept learning through specialized digital fanzines. In: 2021 IEEE/ACM 43rd International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET), pp 134–143. https://doi.org/10.1109/ICSE-SEET52601.2021.00023
33. Research GV (2020) Interactive whiteboard market size, share & trends analysis report by technology (resistive membrane, electromagnetic pen), by form factor, by projection technique, by application, by region, and segment forecasts, 2020 - 2027. Tech. Rep. GVR-2-68038-396-6, Grand View Research
34. Schlegel A (2020) ControlP5: A GUI (graphical user interface) library for Processing. www.sojamo.de/libraries/controlP5/
35. Shotton J, Fitzgibbon A, Cook M, Sharp T, Finocchio M, Moore R, Kipman A, Blake A (2011) Real-time human pose recognition in parts from single depth images. In: Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR), vol 2011, pp 1297–1304. https://doi.org/10.1109/CVPR.2011.5995316
36. Tanaka K, Parker J, Baradoy G, Sheehan D, Holash J, Katz L (2012) A comparison of exergaming interfaces for use in rehabilitation programs and research. J Canad Game Stud Assoc 6:69–81
37. Upadhye A (2019) From blackboards to digital boards: a transformation of government-run schools. www.thehindu.com/education/schools/from-blackboards-to-digital-boards-a-transformation-of-government-run-schools/article28200443.ece
38. Zaleski O, Navarro M, Díaz S, Redondo JM, Labrador M (2018) Clinical gait assessment comparison: Smartphone-based versus inertial measurements units. In: Southeastcon, vol 2018, pp 1–5. https://doi.org/10.1109/SECON.2018.8478977