






# Mobile Augmented Reality Framework - MIRAR

João M. F. Rodrigues<sup>1</sup>(✉) , Ricardo J. M. Veiga<sup>1</sup>, Roman Bajireanu<sup>1</sup>,  
Roberto Lam<sup>1</sup> , João A. R. Pereira<sup>1</sup>, João D. P. Sardo<sup>1</sup>,  
Pedro J. S. Cardoso<sup>1</sup> , and Paulo Bica<sup>2</sup>

<sup>1</sup> LARSyS (ISR-Lisbon) and ISE, University of the Algarve, 8005-139 Faro, Portugal  
{jrodrig,rlam,pcardoso}@ualg.pt, ricardojorge@martinsveiga.com,  
romanmsn.com@hotmail.com, jandreperreira00@gmail.com, joao\_dps@outlook.com

<sup>2</sup> SPIC - Creative Solutions, Loulé, Portugal

**Abstract.** The increasing immersion of technology on our daily lives demands for additional investments in various areas, including, as in the present case, the enhancement of museums' experiences. One of the technologies that improves our relationship with everything that surrounds us is Augmented Reality. This paper presents the architecture of MIRAR, a Mobile Image Recognition based Augmented Reality framework. The MIRAR framework allows the development of a system that uses mobile devices to interact with the museum's environment, by: (a) recognizing and tracking on-the-fly, on the client side (mobile), museum's objects, (b) detecting and recognizing where the walls and respective boundaries are localized, as well as (c) do person detection and segmentation. These objects, wall and person segmentation will allow the projection of different contents (text, images, videos, clothes, etc.). Promising results are presented in these topics, nevertheless, some of them are still in a development stage.

**Keywords:** Augmented reality · Object recognition  
Wall detection · Human detection · HCI

## 1 Introduction

Augmented Reality (AR) [2] is a technology that, thanks to the mobile devices increasing hardware capabilities and new algorithms, quickly evolved in the recent years, gaining a huge amount of users. AR empowers a higher level of interaction between the user and real world objects, extending the experience on how the user sees and feels those objects by creating a new level of edutainment that was not available before. The M5SAR: Mobile Five Senses Augmented Reality System for Museums project [48] aims for the development of an AR system that acts as guide for cultural, historical and museum events. This is not a novelty, since almost every known museum has its own mobile applications (App), e.g. [31,57]. While the use of AR in museums is much less common, it

is also not new, see e.g. [25, 44, 54, 59]. The novelty in the M5SAR project is to extend the AR to the human five senses, see e.g. [48] for more details.

This paper focus on MIRAR, Mobile Image Recognition based Augmented Reality framework, one of the M5SAR's modules. MIRAR focuses on the development of a mobile multi-platform AR [2] framework, with the following main goals: (a) to perform "all" computational processing in the client-side (mobile device), minimizing, this way, costs with server(s) and communications; (b) to use real world two- and three-dimensional (2D and 3D) objects as markers for the AR; (c) to recognise environments, i.e., walls and its respective boundaries; (d) to detect and segment human shapes; (e) to project contents (e.g., text and media) onto different objects, walls and persons detected and displayed in the mobile device's screen, as well as enhance the object's displayed contents, by touching on the device's screen regions on those objects; and (f) to use the mobile device's RGB camera to achieve these goals. A framework that integrates these goals is completely different from the existing (SDK, frameworks, content management, etc.) AR systems [1, 8, 32, 33, 40].

The MIRAR sub-module for object recognition and environment detection presented in this paper is AR marker-based, often also called image-based [9]. AR image-based markers allow adding pre-set signals (e.g., from paintings, statues, etc.) easily detectable in the environment, and the use computer vision techniques to sense them. There are many image-based commercial AR toolkits (SDK) such as Catchoom or Kudan [8, 32], and AR content management systems such as Catchoom or Layar [8, 33], including open source SDKs [1]. Each of the above solutions have pros and cons. Between other problems, some are quite expensive, others consume too much memory (it is important to stress that the present application will have many markers, at least one for each museum piece), and others take too much time to load on the mobile device.

The increasing massification of AR applications brings new challenges to the table, such as the demand for planar regions detection ("walls"), with the more popular being developed within the scope of Simultaneous Localization And Mapping (SLAM) [4, 12]. Usually, the common approach for image acquisition of 3D environments uses RGB-D devices or light detection and ranging (LIDAR) sensors [30, 47, 55, 60]. There are also novelty advances within environment detection, localization or recognition, either using Direct Sparse Odometry [13], or using descriptors, like ORB SLAM [37] or even Large-Scale Direct Monocular SLAM [14]. However, as mentioned, the MIRAR framework focuses on mobile devices with only monocular cameras. Following this, an initial study of an environment detection sub-module was presented in [43], being the purposed method a geometric approach to the extracted edges of a frame. It should be considered that a frame is always captured from a perspective view of the surrounding environment, with the usual expected environment being characterized by the existence of numerous parallel lines which converge to a unique point in the horizon, called vanishing point [11, 53].

The last topics addressed in the MIRAR framework regards the detection of human shapes in real world conditions. This continues to be a challenge in

computer vision due to the existence of multiple variants, e.g., object obstructions, light variations, different viewpoints, the existence of multiple humans (occlusions), poses, etc., nevertheless, the detection of human shapes is an area with many studies and developments [17, 39, 52, 56, 61].

In summary, the MIRAR’s object recognition sub-module uses images from the museum’s objects, and the mobile device’s camera to recognise and track on-the-fly, on the client-side, the museum’s objects. The environment detection and recognition sub-module is supported upon the same principles of the object’s recognition, but uses images from the environment, walls, to recognise them. Finally, the human detection and segmentation uses Convolutional Networks for the detection and an image processing algorithm for foreground (person) extraction. The main contribution of this paper is the integration of these three topics into a single mobile framework for AR.

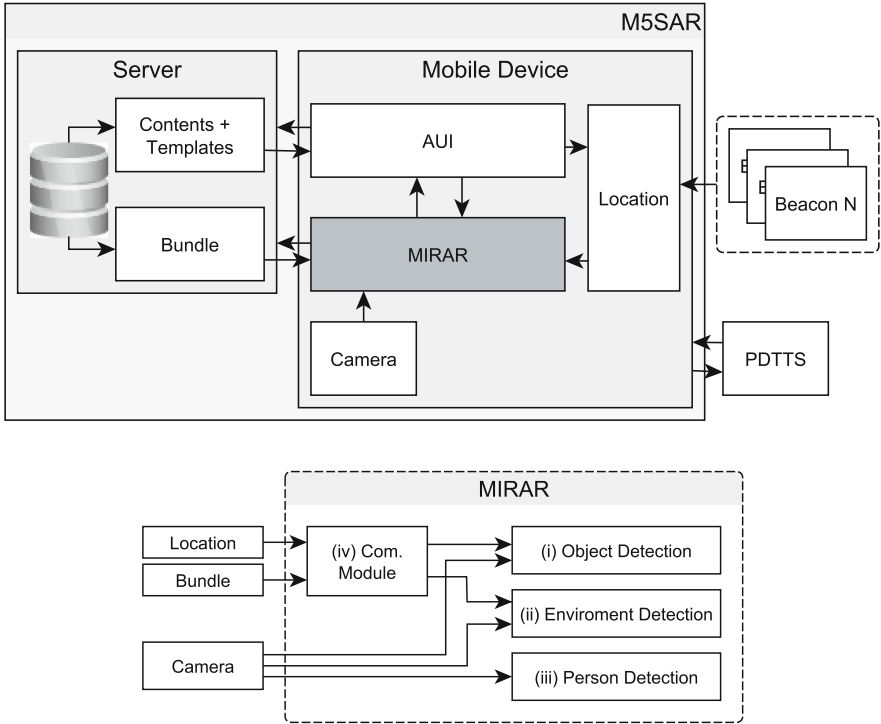
The paper is structured as follows: The MIRAR framework and architecture is introduced in Sect. 2. Section 3 presents the main MIRAR’s sub-module, namely the object detection, followed by the wall detection sub-module in Sect. 4 and the human shape detection in Sect. 5. The paper concludes with a final discussion and future work, Sect. 6.

## 2 MIRAR Framework

Before detailing the MIRAR framework it is important to give a brief overview of the M5SAR system, shown on top of Fig. 1. On the figure’s left side, the basic communications flow between the server and mobile device is outlined (a detailed description is out of the scope of this paper) and, on the right side, the simplified diagram of the mobile App and the devices “connected” (via bluetooth) with the mobile device is shown. The displayed Beacons [16] are employed in the user’s localisation and the Portable Device for Touch, Taste and Smell Sensations (PDTTSS) [51] used to enhance the five senses. In summary, the M5SAR App architecture is divided into three main modules: (A) Adaptive User Interfaces (AUI), see [48]; (B) Location module, a detailed explanation is out of this paper’s focus, and (C) MIRAR module (see Fig. 1 bottom).

The MIRAR has four main features: (a) the detection and recognition of museum objects, triggering a card in the (M5SAR) App [48]; (b) the detection, recognition and tracking of objects as the user moves along the museum, allowing to touch different areas of the objects displayed in the mobile screen and showing information about that region of the object, MIRAR sub-module (i); (c) detection and modelling of the museum walls, and projecting information into the detected walls (e.g., images, movies, text) related with the recognized object’s epoch, sub-module (ii); (d) detection of persons that are moving in the museum, and, for instance, to dress them with clothes from the object’s epoch, sub-module (iii).

Sub-modules (i) and (ii) need to communicate with the server, i.e., the MIRAR module sends the user’s position to the server, based on the previous object detections and the localisation given by the beacon’s signals. From the



**Fig. 1.** Top: overall simplified system architecture. Bottom: MIRAR block diagram.

server, the MIRAR module receives a group of object markers (image descriptors; see next section), here called bundles, that contain all the objects available in the located room or museum section. In a way to minimise communications, the App stores in the memory (limited to each device’s memory size) the bundles from the previous room(s), museum section(s), and as soon as it detects a new beacon signal it downloads a new bundle. Older bundles are discarded in a FIFO (first in, first out) manner.

It is also important to stress that, since the sensor used to acquire the images from the environment is the mobile’s camera, in order to save battery, the camera is only activated when the AR option is selected in the UI. When the activation occurs, the user can see the environment in the mobile screen and effectuate the previously mentioned actions. As an additional effort to save battery, the device will enter a low-power state if the user turns the phone upside down, by dimming the phone’s screen and interrupting the processing.

As final remarks, the App was implemented using Unity [58], the computer vision algorithms were deployed using the OpenCV [38] library (Asset) for Unity, and tests and results consider that the mobile device is located inside a museological space. The next section will present the object detection and tracking module.

### 3 Object Detection, Recognition and Tracking Module

The object detection sub-module aim at detecting objects present in the museum, being the algorithm divided in 2 components: (a) detection and recognition, and (b) tracking. While the recognition is intended to work on every museum object, the tracking will only work in masterpieces<sup>1</sup>. The masterpieces' tracking allows to place contents in specific parts of the UI, so that the user will touch on those areas in order to gain more information about a particular region of the detected object.

Before describing this module in further details, it is important to distinguish from templates and markers. Here, templates are images (photographs) of the objects stored in the server's database (DB) while, on the other hand, a marker is the set of features (keypoints) with their respective (binary) descriptors for a certain template, see Fig. 2 and [43]. The authors' employ the ORB descriptor for keypoint detection and descriptors implementation [19, 43, 50].

A generic image recognition and tracking algorithm for AR has the following main steps: (1) extract the markers (keypoints and descriptors) from a template; (2) extract keypoints and compute descriptors from query images (i.e., for each mobile device camera's frame); (3) match the descriptors of both the template and query; and, when needed, (4) calculate the projection matrix to allow perspective wrapping of images, videos, and other contents.

An initial recognition algorithm was presented in [48], with further advances presented in [43], as follows. Similar to [3], in Step (1) it is utilised the image to extract keypoints and compute descriptors. The borders are the exception (e.g., the painting frame) which were removed, since usually there is no relevant information in those areas. Nevertheless, the templates are processed in different scales (image sizes): starting at the pre-defined camera frame size,  $640 \times 480$ px (pixels), the templates are scaled up and down (by a  $1/3$ ), resulting in a total of 3 scales per template. To further increase the framework's performance, these markers continue to be created on a server and sent to the client (mobile device) on demand, to be de-serialized. Step (2) from the frame acquired by the camera, the keypoints are simply extracted and their respective descriptors computed (using the ORB descriptor).

Regarding Step (3), the query image descriptors are (3.1) brute-force matched, using K-Nearest Neighbours (KNN), with  $K = 2$ , against the descriptors of the available markers. Next, (3.2) the markers' descriptors are matched to the query's descriptors. Following, (3.3) a ratio test is performed, i.e., if the two closest neighbours of a match have close matching distances (65% ratio), then the match is discarded [3], because this would be an ambiguous match. This ratio evaluation is the test where most matches are removed. For this reason, this test is performed first to improve performance later on. Then, (3.4) it is performed a symmetry match where only the matches resulting from the KNN in (3.1) that are present in (3.2) are accepted. After this, a (3.5) homography refinement is

---

<sup>1</sup> Masterpieces are objects that have an enlarged (historical and cultural) value in the museum's collection.



**Fig. 2.** Top: Example of existing objects in Faro Municipal Museum. Bottom: examples of detected and tracked markers with the corresponding axis.

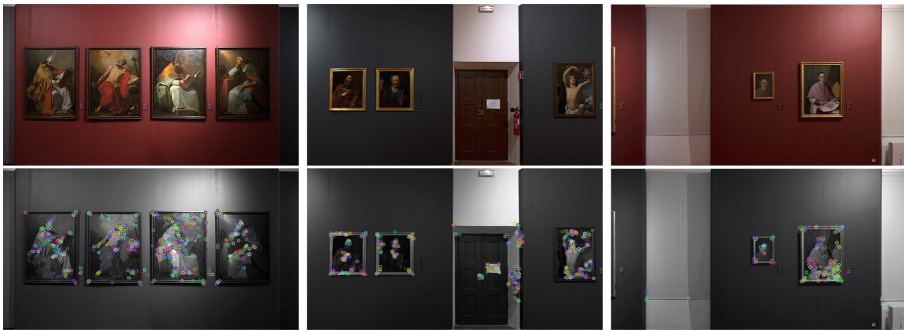
applied. This refinement uses the RANSAC method to verify if the matched keypoints in the query image maintain the same configuration between them (same relative position) as they had in the template image. If any of the keypoints stay out of this relation, then they are considered outliers and removed from the match set. (3.6) If after all of these refinements there are at least 8 matches left, then it is considered as a valid classification.

In the (3.6.i) classification stage the query image is compared using Brute-Force (BF) to all marker scales for each of the available templates. This, in turn, returns a classification based on the count of (filtered) matches, when there are at least 8 descriptors matches. The marker that retrieved the most number of matches is considered the *template to be tracked*. Afterwards (3.6.ii), if the tracking stage is necessary, i.e., if a masterpiece is present, the matching only occurs with the markers of the 3 scales of the *template to be tracked* previously selected in classification phase. If the object (*template to be tracked*) is not visible in the scene for 1 second then it is considered lost and the recognition process initiates again. Last, but not least, Step (4) of the generic algorithm is done using perspective wrapping (pose estimation) in order to place content on the same plane as the detected image (marker).

Figure 2 bottom shows some examples of tests done in the Faro Municipal Museum where the classification number is shown in red. For more algorithm details and results see [43].

## 4 Environments Detection

As previously mentioned, the objective of this sub-module is to be able to discern the location and position of the walls of a given environment, and afterwards replace them with other contents. The algorithm presented here does not yet supports our investigation over this subject mentioned in [43], although it will eventually merge with the preceding work. In the present case, similar to Sect. 3 markers are used (keypoints and descriptors) for each template, with the bundles being previously generated. For this sub-module, the templates are of the entire walls, and not only of the museum’s object, see Fig. 3 top row, which allows for the retrieval of the expected wall shape using a pose estimation algorithm. Various implementations of pose estimation have already been presented for 3D objects using RGB-D sensors [6] or through monocular images [46], and urban environments [22]. The main contribution of this sub-module is the initial implementation of the wall estimation for primarily indoor detection and recognition while the user navigates through museums, which are presented in this paper, with a future user localization feature already being developed. Regarding our implementation of wall estimation, it is important to note that the aim for this sub-module is a seamlessly fully integrated AR application for mobile devices; therefore, the presented algorithm is focused and adjusted for performance on smartphones.



**Fig. 3.** Top: Example of templates. Bottom: Extracted keypoints location for each template.

Contrary to what was presented for the object detection, instead of using ORB descriptors, we found that BRISK [34] descriptors perform better for this task, which will be explained later in Sect. 4.1. For comparison between markers, not only Brute-Force was tested, but also the *Fast Library for Approximate Nearest Neighbours* (FLANN) [36]. The necessity of evaluating both matchers for this task will be posteriorly explained.

The current algorithm, after the bundle has been created and loaded, is applied to each frame from the mobile device camera as follows: (1) A number of the most significant keypoints are retrieved (filtered) and the respective



descriptors computed; (2) Optimal matches are found and filtered; (3) Pose estimation is performed after discarding the defective homography; (4) Polygons corresponding to the matched templates are superimposed on the frame.

Beginning with Step (1), all the keypoints (using the BRISK keypoint detector) found from the frame provided are ordered by their response, which defines the ones with stronger information, and only an amount of keypoints is maintained, which in our case was  $Num_{KP} = 385$  (this number was empirically computed, see Sect. 4.1). More keypoints will not improve results and it increases computational time. If this number is decreased significantly, many “template (wall) - frame” matches are lost. Afterwards, the respective descriptor for each keypoint is computed using the BRISK descriptor.

In Step (2), before searching through all the stored templates, it is verified if the location of the user is known through the previous frames, thus allowing for the matching search to begin with the surrounding templates. The method used for matching was K-Nearest Neighbours (the same as in Sect. 3), with  $K = 2$ , either by BF matching, or using FLANN. While the BF compares all the retrieved markers’ descriptors from the frame with the stored markers from the templates, it was created an index with FLANN that uses multi-probe LSH [35]. The parameters used were 6 hash tables, with 12 bits hash key size, and 1 bit to be shifted to check for neighbouring buckets. The number of times we defined for the index to be recursively traversed was 100, as we observed a good balance between additional processing time and the increased precision. It is important to refer that, as opposed to BF, FLANN does not return a complete matching between the markers, but instead it gives an approximate correspondence. The remaining matches are filtered through the Lowe’s ratio test, where we discard the pairs with close matching distances (65% ratio), allowing only the more distinct ones to remain. Subsequently, if at least 10 good matches are found, then the perspective transform is retrieved through the homography refinement using the RANSAC method, where the original pattern of keypoints from the templates are compared with the ones from the frame, considering the ones with the same configuration as inliers, and the others as outliers.

Regarding Step (3), for the pose (wall) estimation templates to be properly found, the perspective matrix must be found valid. It should be noted that the existing planes across the provided frames will be randomly presented with acute perspective angles, or at deeper distances. Concerning the templates’ format, for this sub-module we chose to include the desired full wall delimitations to be found, even if the regular walls did not offer relevant information to be retrieved, with the keypoints gathered in clusters along the museums’ objects, see in Fig. 3.

The chosen template format after the pose estimation returned the approximated horizontal limits of the walls. In order to improve accuracy and performance, it was necessary to discard the non relevant perspective matrices. To do so, we analysed the matrix extracted from the homography and applied a group of tests. We calculated the determinant of the top left  $2 \times 2$  matrix and limited the output between 1 and 100, given that, with the perspective transform, if the values of said determinant were to be negative there would be an inversion,



and as the templates were created for the expected projection, there should be none. The limit of 100 was imposed because in case there was a large value for the determinant, then the aspect ratio would have been overly deformed. Furthermore, after finding the coordinates, their order is compared against the original template, e.g., if the  $(x_0, y_0)$  of the template is on the top left and the  $(x_1, y_1)$  on the lower left; then, after the perspective transform, this orientation should remain. Afterwards, it is verified if the angles between each 3 points are not overly convex, as they are expected to be nearly perpendicular. Finally, as the environment/room is “regular”, which means the presence of vertical walls without deformations (no circular walls) or extensive 3D artwork, all the non vertical resulting polygons with an error of 15% are discarded.



**Fig. 4.** Example of a sequence of frames with matched templates. See also Fig. 6.

The last Step (4), the retrieved coordinates are converted into polygons that are superimposed upon the original frame for each of the matched templates, corresponding to the expected surface of the wall in the environment; a sequence of the output results can be seen in Fig. 4. With this outcome it is possible to project content not only replacing the walls, as presented before in [43], but also to present floating AR content. It is important to stress that when the angle between the wall and the mobile user is “too sharp” is not yet possible to find the boundaries of the wall, which can be shown if Fig. 4 bottom row.

## 4.1 Tests

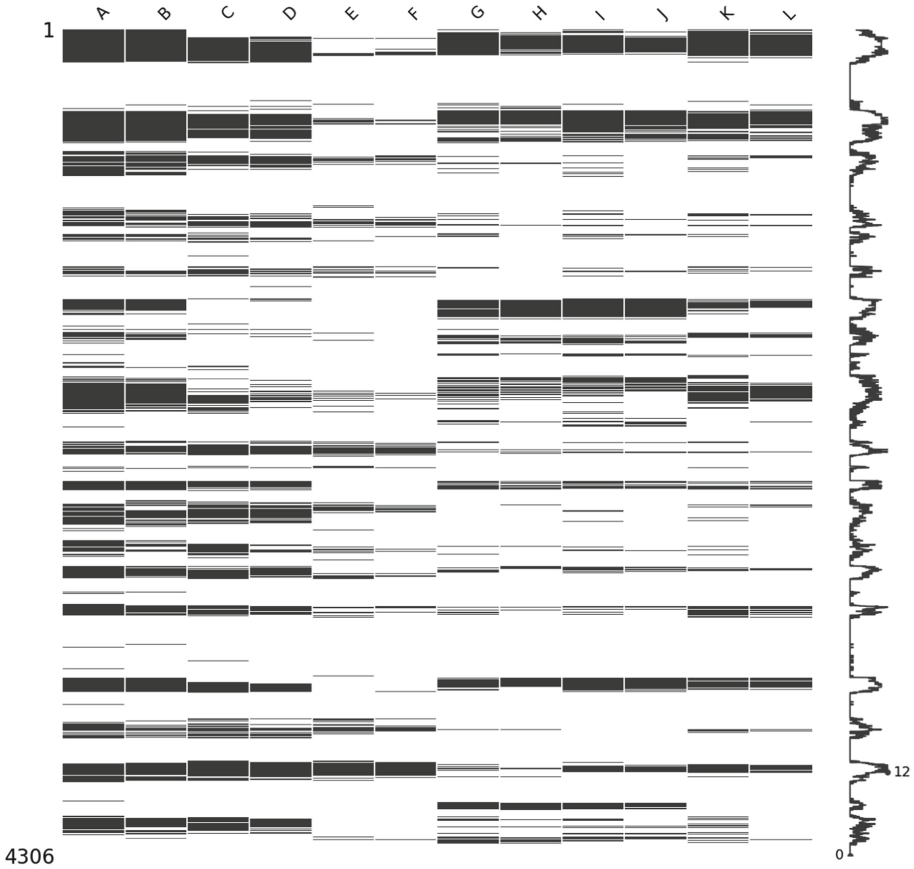
In order to test the reliability of the algorithm, tests were done before converting the algorithm to the mobile platform; nevertheless, all the videos used for the tests were acquired by mobile devices (smartphones and tablets). The tests were done using a desktop computer with an Intel CPU i5-6300 running at 2.4 GHz with the algorithm limited to run in single-thread. The videos consisted of a total amount of 4,306 frames of expected user navigation through the museum, with both the horizontal and vertical orientations used. An additional video containing persons in between the camera and the walls also showed good results, as seen in Fig. 6. It is important to note that it is expected that this sub-module will not always detect and recognize the environment in all the frames; therefore, the most important measure of success is the amount of frames with valid matches found.

The tests were conducted in the following ways: the templates' width size between 320px and 640px; the frame's width size between 640, 480 and 320px; and increasing the number of minimal good matches, starting with 10 and using steps of 5. All the tests were performed using BF and FLANN.

Regarding the variation of the minimal good matches between markers to begin the template matching, as expected, with the increase of this threshold the amount of frames with a found template match were dramatically reduced, while it was observed the maintenance of a similar processing time, either for each frame as for each matched frame, showing little to no variation. The results in terms of "pose" estimation of the polygons over the output frame were also improved. Upon reviewing these results we decided to use the minimal value of 10, given that it returned the highest number of frames matched without overly increasing the undesired results; for example, changing this value to 15 reduced the frames matched by 35 – 40%. With the variation of the templates width size, it was expected to add additional detected matches for when the wall is distant and is presented smaller on the frame. The results showed that even when it occasionally happens, it doesn't justify adding different scales of templates for this sub-module at the current version in exchange of processing time performance, as it was presented in [48].

The obtained processing times for the current algorithm, while reducing the templates and frames width, decreased from  $28,3 \pm 13,8$ ms to  $17,9 \pm 5,9$ ms with BF, and from  $33,2 \pm 14,2$ ms to  $24,1 \pm 17,8$ ms. Even though it presented some improvement on the time performance, the amount of frames matched dropped 73 – 63% respectively.

Considering the necessity of a higher rate of matching, the performance of different templates and frames sizes were compared. The illustration in Fig. 5 presents the amount of frames matched with the colour black within the total frames of different expected user interaction videos. On the left is shown the frame number (1, ..., 4306), and on the right, the histogram of matched template-frame along the different widths and matching algorithms, namely FLANN and BF. The intent of this comparison, other than reporting the total of matched frames for each different test, was to analyse if the scale factor would introduce new



**Fig. 5.** Illustration of the number of matched frames (in black) with templates, changing the widths of both, i.e., from A-F is shown for each pair Flann (A, C and E) and BF (B, D and F) the matched templates with a width of 640px, the frames' width varies from 640 (A and B), to 480 (C and D) and 320px (E and F). From G-L the same but now with the template with the width of 320px. At right, an histogram of the total of matches for each frame is shown.

results, either by reducing the frame width while retaining a larger template width, which is shown from A-F, or by reducing the template width, while again changing the size of the frame, as can be seen from G-L, with a template width of 320px. The test were also divided in FLANN (A, C, E, G, I and K) and BF (B, D, F, H, J and L). The variation of frames' width is organized as 640 (A and B), 480 (C and D) and 320px (E and F), with the same from G-L.

Using the illustration shown in Fig. 5 it becomes easier to analyse the effect of the different parameters, being one of the main comparisons the use of FLANN or BF for feature matching. Is it relevant to note that, as BF needs a complete certainty for matching, which would be achieved more easily if the desired matching

template was in full frame, FLANN does not, which introduces the possibility of the existence of false positives. As a continuous outputs of false positives from the same template is uncommon, it became easier to discard them. Although FLANN versus BF is usually restricted to a large amount of keypoints, in our case for 640px templates the average was  $384.5 \pm 119.48$  keypoints per image, where FLANN surpasses BF in processing time performance, for our case it was primarily used with the intent of retrieving additional matched frames. Regarding the obtained results, FLANN returned additional matched frames with 38% of the total number of 4.306 frames matched, and BF returned 32% for 640px templates, while for 320px we obtained 20% and 15%, respectively. Although there is an increase in processing time for FLANN in order of 17% facing BT, when analysing the results shown in Fig. 5, it is possible to verify a more sparse occurrence of matched frames for FLANN versus BF, allowing for a higher probability of matching while the user navigates the museum, which will be used in order to recalibrate (in the future) the user localization and focus, improving the projection of content tracking and stability.

Focusing on the 640px width templates, it is possible to observe the expected lower matching while reducing the frames' width, for the total of available markers for each frame was reduced against the templates average number of extracted keypoints. With the 320px templates, the results showed a different outcome. While the total matched frames was also reduced, it became almost invariable across the tests, which means that with a lower processing time the same results would be achievable. One interesting point is noticeable near the medium point, where there was more matched frames acquired with lower templates' width. This is explained with the distance of the matching template, i.e., if the frame's width is 640px and the templates' 320px, if the respective location of the template is inside the frame at distance, it will be closest to the lower templates' width than the larger, and as we are using BRISK descriptors, even if they are invariant to scale, there is a threshold to the maximum of that invariance. In conclusion, for these results it can be seen that in the future the implementation of different scales of templates to improve the localization tracking may be needed.

As referred before, for this module a BRISK descriptor was used instead of ORB. Although the amount of frames with matched templates was similar in between, ORB performed slightly better, with 771 frames with match against 726 of BRISK, from a total amount of frames of 1.857. The outcome of said matches presented worse projected polygons, meaning that even with the homography additional sanity tests, the occurrence of bad homographies increased. Furthermore, an increase in the occurrence of false positives with a factor of ten times between ORB and BRISK was observable, which largely contributed to the bad homographies received. The performed times from ORB to BRISK lowered from  $37.8 \pm 9.5$ ms to  $27.1 \pm 10.4$ ms while using FLANN, and  $36.7 \pm 9.6$ ms to  $26.0 \pm 8.5$ ms with BF. Is important to observe that for our tests using ORB, FLANN does not seem to affect the performance times. Lastly, the amount of average keypoints retrieved from the templates actually decreased from ORB to BRISK, with a

total of 5.824 keypoints, with an average of  $485.3 \pm 24.4$  keypoints per template, to a total of 4.614 keypoints, with an average  $384.5 \pm 119.5$ , which shows ORB being more consistent than BRISK for the amount retrieved, although it did not prove that with the additional keypoints the results would improve.

As the objective of this sub-module is the recognition and detection of the walls throughout the visit within some rooms of the museum, it is expected a loss of tracking/recognition and its recovery at a slightly different location or angle, which demands that the recognized template shape be the as close as possible to the desired, every time there is a match within frames. Since the amount of good expected results with BRISK surpassed the ORB descriptor, we decided to perform the current algorithm using only the BRISK descriptor, while the algorithm for object detection shown at Sect. 3 remains using ORB. The different outcomes between both the descriptors for this challenge could be due to the fact that the BRISK descriptor is invariant to scale, while the ORB is not. Furthermore, while for object detection we used scaled versions of the templates for matching, as the object is expected to fill the screen of the mobile device, with this module there was an additional inclusion of different scales, as it should be considered that the user will be navigating the different rooms of the museum; therefore, the templates, in this case the walls, will appear on the mobile device with different geometric shapes and distances; see also [43].

Additional examples of results obtained can be observed in Fig. 6, where is possible to see on top the algorithm working with vertical frames and some of the still occurring bad outcomes retrieved from faulty perspectives transforms from the homography. On the 2nd row, results of the current algorithm can be observed, while the view is partially obstructed with people. Additionally, it is important to remark that this module is being ran only from the frames obtained, without additional sensors or 3D information, and so, the results obtained with obstructed views are a welcoming result for the current implementation.

In Fig. 6 bottom row it is possible to observe the results of part of the former algorithm [43] applied to the retrieved frames. While with the former implementation the process would begin by elimination of all non relevant lines, or in our former case, all the non vertical, horizontal and vanishing lines, the future fusion of both developments will be more focused in only retrieving the nearest lines to the already extracted polygons through the Hough Transform [26], improving the polygons veracity to the actual walls' shapes and adding a level of longer distance detection, either by additional calculations through the use of the vanishing point together with the vertical and horizontal lines, as can be seen in [43], or with an eventual introduction of a pre-known room shape, which, combined with the user localisation, would achieve better results, presenting the opportunity for the use of indoor 3D models, further increasing the user immersion with AR.

On the next section we introduce an initial study for the detection and segmentation of the human shape.



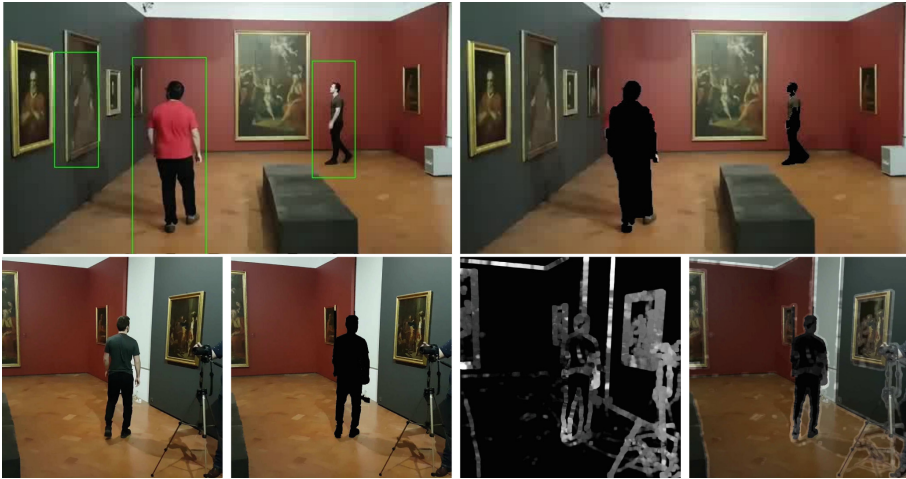
**Fig. 6.** Top row, examples of different vertical matching outcomes. Middle row, initial results of matching while the view is obstructed by persons. Bottom row, examples of the Hough Transform [26] applied to different frames.

## 5 Human Shape Detection

Human shape detection, as mentioned in the Introduction, already presents its challenges; furthermore, for this sub-module we have to consider the detection in real-time on a mobile device, while the user is moving through six degrees of freedom (6DOF), which will increase the level of complexity [5, 42]. Recent researches approach the human shape detection either by a top-down or a bottom-up method. Top-down means that the persons' shape are first detected and afterwards an estimation of their poses is achieved [23, 24, 41], while with bottom-up the humans' limbs are individually detected, generating groups of body parts in order to form humans' poses [7, 17]. For the initial study of this module we used a top-down approach, being the objective the detection and segmentation of the humans' shapes, allowing for the projection of AR content over it, as for example, the ability "to dress" the museums' users with clothes corresponding to the desired surrounding epoch of the museums' objects.

In order to overcome the complex challenges imposed by the detection of human shapes in video, captured by a moving camera of a mobile device, we used a convolutional neural networks (CNN), built in TensorFlow [21]. To detect human shapes in a video feed with reasonable rate of *fps* we used a single shot





**Fig. 7.** Top row-left, example of human detection with SSD-Mobilenet. Right, the result of human segmentation by GrabCut. Bottom row, from left to right, original image (frame), human segmentation, computed optical flow of two consecutive frames and overlap of segmentation and optical flow (the two former images).

detection (SSD) network, and we used the MobileNet model for the neural network architecture; as its name suggests, it is designed for mobile applications [27]. The other technique used in the process of human detection/segmentation is the GrabCut algorithm [42]. It has a limitation where it needs to define the foreground and background areas; hence, we propose a fully-automatic human segmentation method by using the bounding box as a basis for the foreground and background areas.

The algorithm for this module is executed for each frame following these steps: (1) Apply SSD-Mobilenet [29], used for human detection, which outputs a bounding box around the detected humans (see Sect. 5.1 for the justification). (2) Resize the extracted bounding box by an increase of 10%; the original bounding box would cut some parts of the human shape in some image conditions, thus this improves the foreground precision. Step (3) follows with a cut of the input image, the cut is made with twice (2x) the size of the initial bounding box, with the same centre, and inside, the cropped area is used as background. Finally, (4) we use the GrabCut [49] algorithm for human shape segmentation.

## 5.1 Tests

Three models were tested in the mobile device for the human shapes detection, SSD-Mobilenet [28], YOLO [45], and DeepMultiBox [15]. Empirical tests were done in a Museum (Faro Municipal Museum) using an ASUS Zenpad 3S 10 tablet, showing that in real world conditions the SSD-Mobilenet presented a better accuracy and speed, validating what is mentioned in [28]. Initially, we



used COCO [10] frozen pre-trained weights for SSD-Mobilenet. The evaluation setup, as mentioned, consisted on a ASUS Zenpad 3S 10 tablet and a windows machine with an Intel i7-6700 CPU @ 3.40 GHz. A total of 86 frames of data were run 15 times through the network, with their performance being recorded. The resolution of the input frames was 640px and the spatial size of the convolutional neural network was 320px. To filter out weak detections we use a confidence threshold of 0.25. Our tests for this model, using the tablet, returned an average processing time of 346.0ms, while the computer achieved 33.7ms, being these values only regarding Step (1).

After the human detection, and for the remaining Steps (2-4) in our algorithm, with the use of GrabCut for segmentation, we achieved an average processing time of 127.3ms using the computer. In Fig. 7 top row is possible to observe the output frame showing promising results regarding human segmentation. An observable disadvantage for this module within the museum environment can also be observed with a painting of a person being detected as living person. Although it is a good feature, for this task it is an undesirable result. Furthermore, in Fig. 7 bottom row is possible to inspect that, when the conditions provide a discriminative foreground and background areas, the GrabCut algorithm can perform with high precision.

To solve the problem caused by mis-segmentation of the limbs of a segmented person, as for example in the left image, where the arms are indistinguishable from the torso, we started to apply Gunnar Farneback's algorithm to compute dense optical flow between two consecutive frames [18,20]. This allows to complement the GrabCut segmentation process by using the consistency of pixel values in two frames. Using Farneback's algorithm allows to estimate the optical flow in a sequence of frames, and it is possible to use it to locate the borders of limbs that do not appear in the GrabCut segmentation. The algorithm shows an optical flow field with distinguished values between torso and arms because they have different speed movements, see Fig. 7 bottom right.

## 6 Conclusions

This paper presents the current Mobile Image Recognition based Augmented Reality (MIRAR) framework architecture. Even in its current state, MIRAR had already presented good results in the object detection, recognition, and tracking sub-modules. The integration with the new approach for the wall detection and recognition shows satisfactory results, taking in consideration that it is still a work in progress. For the human shapes detection, initial results were shown; nevertheless, more consistent tests need to be performed in different museum conditions.

For future work, the recognition of 3D objects is an immediate focus in terms of creating a robust bank of tests, and so is the refinement of the object recognition and tracking module. This can be achieved by refining the matches with homography and trying to find an optimised set of keypoints from multiple scales. For the wall detection, the focus will be on improving the stability and

further filtering the occasional bad results, introducing pre-tuned templates to increase the range of detection, while preserving performance, the inclusion of a tracking system, and a merger with the previous work presented on edges detection for geometric prediction to stabilise the resulting polygons, reaching for a predictive localization of the surrounding indoor environment. The current different choices of descriptors between objects and wall detection will also be addressed.

For the human shapes detection, the segmentation done with the use of the GrabCut algorithm needs to be complemented in order to acquire a good human segmentation, since it will allow the projection of contents onto those shapes/persons. In the future we plan to use optical flow estimation (with initial results already shown) in the final segmentation process in order to improve the segmentation results. Additional work needs to be done to reduce the execution times of the detection and segmentation.

As a final conclusion, the MIRAR shows, even in this current stage, promising results, and it is expected to be an excellent tool to give a more impactful relation between the museum's user and the museum's objects.

**Acknowledgements.** This work was supported by the Portuguese Foundation for Science and Technology (FCT), project LARSyS (UID/EEA/50009/2013), CIAC, and project M5SAR I&DT nr. 3322 financed by CRESC ALGARVE2020, PORTUGAL2020 and FEDER. We also thank Faro Municipal Museum and the M5SAR project leader, SPIC - Creative Solutions [[www.spic.pt](http://www.spic.pt)].

## References

1. Artoolkit.: ARtoolKit, the world's most widely used tracking library for augmented reality (2017). <http://artoolkit.org/>. Accessed 16 Nov 2017
2. Azuma, R., Bailiot, Y., Behringer, R., Feiner, S., Julier, S., MacIntyre, B.: Recent advances in augmented reality. *IEEE Comput. Graph. Appl.* **21**(6), 34–47 (2001)
3. Baggio, D.L.: *Mastering OpenCV with Practical Computer Vision Projects*. Packt Publishing Ltd, Birmingham (2012)
4. Bailey, T., Durrant-Whyte, H.: Simultaneous localization and mapping (SLAM): part II. *IEEE Robot. Autom. Mag.* **13**(3), 108–117 (2006)
5. Bhole, C., Pal, C.: Automated person segmentation in videos. In: 21st International Conference on Pattern Recognition (ICPR), pp. 3672–3675. IEEE (2012)
6. Buch, A.G., Kraft, D., Kamarainen, J.-K., Petersen, H.G., Krüger, N.: Pose estimation using local structure-specific shape and appearance context. In: IEEE International Conference on Robotics and Automation (ICRA), pp. 2080–2087. IEEE (2013)
7. Cao, Z., Simon, T., Wei, S.E., Sheikh, Y.: Realtime multi-person 2D pose estimation using part affinity fields. In: CVPR, vol. 1, no. 2, p. 7 (2017)
8. Catchoom.: Catchoom (2017). <http://catchoom.com/>. Accessed 16 Nov 2017
9. Cheng, K.-H., Tsai, C.-C.: Affordances of augmented reality in science learning: suggestions for future research. *J. Sci. Educ. Technol.* **22**(4), 449–462 (2013)
10. COCO.: COCO - common objects in context (2018). <http://cocodataset.org/>. Accessed 14 Jan 2018

11. Duan, W.: Vanishing points detection and camera calibration. Ph.D. thesis, University of Sheffield (2011)
12. Durrant-Whyte, H., Bailey, T.: Simultaneous localization and mapping: part I. *IEEE Rob. Autom. Mag.* **13**(2), 99–110 (2006)
13. Engel, J., Koltun, V., Cremers, D.: Direct sparse odometry. *IEEE Trans. Pattern Anal. Mach. Intell.* **40**(3), 611–625 (2017)
14. Engel, J., Schöps, T., Cremers, D.: LSD-SLAM: large-scale direct monocular SLAM. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) *ECCV 2014*. LNCS, vol. 8690, pp. 834–849. Springer, Cham (2014). [https://doi.org/10.1007/978-3-319-10605-2\\_54](https://doi.org/10.1007/978-3-319-10605-2_54)
15. Erhan, D., Szegedy, C., Toshev, A., Anguelov, D.: Scalable object detection using deep neural networks. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2147–2154 (2014)
16. Estimote.: Create magical experiences in the physical world (2017). <https://goo.gl/OHW04y>. Accessed 04 Apr 2017
17. Fang, H., Xie, S., Lu, C.: RMPE: Regional multi-person pose estimation. *arXiv preprint* (2017)
18. Farneback, G.: Two-frame motion estimation based on polynomial expansion. In: Bigun, J., Gustavsson, T. (eds.) *SCIA 2003*. LNCS, vol. 2749, pp. 363–370. Springer, Heidelberg (2003). [https://doi.org/10.1007/3-540-45103-X\\_50](https://doi.org/10.1007/3-540-45103-X_50)
19. Figat, J., Kornuta, T., Kasprzak, W.: Performance evaluation of binary descriptors of local features. In: Chmielewski, L.J., Kožera, R., Shin, B.-S., Wojciechowski, K. (eds.) *ICCVG 2014*. LNCS, vol. 8671, pp. 187–194. Springer, Cham (2014). [https://doi.org/10.1007/978-3-319-11331-9\\_23](https://doi.org/10.1007/978-3-319-11331-9_23)
20. Fleet, D., Weiss, Y.: Optical flow estimation. In: Paragios, N., Chen, Y., Faugeras, O. (eds.) *Handbook of Mathematical Models in Computer Vision*, pp. 237–257. Springer, Boston (2006). [https://doi.org/10.1007/0-387-28831-7\\_15](https://doi.org/10.1007/0-387-28831-7_15)
21. Google.: TensorFlow - an open-source machine learning framework for everyone (2018). <https://www.tensorflow.org/>. Accessed 14 Jan 2018
22. Hallquist, A., Zakhor, A.: Single view pose estimation of mobile devices in urban environments. In: *2013 IEEE Workshop on Applications of Computer Vision (WACV)*, pp. 347–354. IEEE (2013)
23. He, K., Gkioxari, G., Dollár, P., Girshick, R.: Mask R-CNN. In: *IEEE International Conference on Computer Vision (ICCV)*, pp. 2980–2988. IEEE (2017)
24. Hernández-Vela, A., Reyes, M., Ponce, V., Escalera, S.: Grabcut-based human segmentation in video sequences. *Sensors* **12**(11), 15376–15393 (2012)
25. HMS.: Srbija 1914/augmented reality exhibition at historical museum of Serbia, Belgrade (2017). <https://vimeo.com/126699550>. Accessed 04 Apr 2017
26. Hough, P.V.: Method and means for recognizing complex patterns. Technical report (1962)
27. Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H.: Mobilenets: Efficient convolutional neural networks for mobile vision applications. *CoRR*, abs/1704.04861 (2017)
28. Huang, J., Rathod, V., Sun, C., Zhu, M., Korattikara, A., Fathi, A., Fischer, I., Wojna, Z., Song, Y., Guadarrama, S., et al.: Speed/accuracy trade-offs for modern convolutional object detectors. *arXiv preprint arXiv:1611.10012* (2016)
29. Huang, J., Rathod, V., Sun, C., Zhu, M., Korattikara, A., Fathi, A., Fischer, I., Wojna, Z., Song, Y., Guadarrama, S., et al.: Speed/accuracy trade-offs for modern convolutional object detectors. In: *IEEE CVPR* (2017)

30. Hulik, R., Spanel, M., Smrz, P., Materna, Z.: Continuous plane detection in point-cloud data based on 3D hough transform. *J. Vis. Commun. Image Represent.* **25**(1), 86–97 (2014)
31. InformationWeek.: Informationweek: 10 fantastic iPhone, Android Apps for museum visits (2017). <https://goo.gl/XF3rj4>. Accessed 04 Apr 2017
32. Kudan.: Kudan computer vision (2017). <https://www.kudan.eu/>. Accessed 16 Nov 2017
33. Layar.: Layar (2017). <https://www.layar.com/>. Accessed 16 Nov 2017
34. Leutenegger, S., Chli, M., Siegwart, R.Y.: BRISK: binary robust invariant scalable keypoints. In: *IEEE International Conference on Computer Vision (ICCV)*, pp. 2548–2555. IEEE (2011)
35. Lv, Q., Josephson, W., Wang, Z., Charikar, M., Li, K.: Multi-probe LSH: efficient indexing for high-dimensional similarity search. In: *Proceedings of the 33rd International Conference on Very Large Data Bases*, pp. 950–961. VLDB Endowment (2007)
36. Muja, M., Lowe, D.G.: Fast matching of binary features. In: *2012 Ninth Conference on Computer and Robot Vision (CRV)*, pp. 404–410. IEEE (2012)
37. Mur-Artal, R., Montiel, J.M.M., Tardos, J.D.: ORB-SLAM: a versatile and accurate monocular slam system. *IEEE Trans. Rob.* **31**(5), 1147–1163 (2015)
38. OpenCV.: OpenCV (2017). <http://opencv.org/>. Accessed 04 Apr 2017
39. Ouyang, W., Wang, X.: Joint deep learning for pedestrian detection. In: *2013 IEEE International Conference on Computer Vision (ICCV)*, pp. 2056–2063. IEEE (2013)
40. Pádua, L., Adão, T., Narciso, D., Cunha, A., Magalhães, L., Peres, E.: Towards modern cost-effective and lightweight augmented reality setups. *Int. J. Web Portals (IJWP)* **7**(2), 33–59 (2015)
41. Papandreou, G., Zhu, T., Kanazawa, N., Toshev, A., Tompson, J., Bregler, C., Murphy, K.: Towards accurate multiperson pose estimation in the wild. *arXiv preprint [arXiv:1701.01779](https://arxiv.org/abs/1701.01779)*, 8 (2017)
42. Park, S., Yoo, J.H.: Human segmentation based on grabcut in real-time video sequences. In: *IEEE International Conference on Consumer Electronics (ICCE)*, pp. 111–112. IEEE (2014)
43. Pereira, J.A.R., Veiga, R.J.M., de Freitas, M.A.G., Sardo, J.D.P., Cardoso, P.J.S., Rodrigues, J.M.F.: MIRAR: mobile image recognition based augmented reality framework. In: Mortal, A., Aníbal, J., Monteiro, J., Sequeira, C., Semião, J., Moreira da Silva, M., Oliveira, M. (eds.) *INCREaSE 2017*, pp. 321–337. Springer, Cham (2018). [https://doi.org/10.1007/978-3-319-70272-8\\_27](https://doi.org/10.1007/978-3-319-70272-8_27)
44. Qualcomm.: Invisible museum (2017). <https://goo.gl/aS0NKh>. Accessed 04 Apr 2017
45. Redmon, J., Farhadi, A.: Yolo9000: better, faster, stronger. *arXiv preprint [arXiv:1612.08242](https://arxiv.org/abs/1612.08242)* (2016)
46. Riba Pi, E.: Implementation of a 3D pose estimation algorithm. Master’s thesis, Universitat Politècnica de Catalunya (2015)
47. Ring, J.: The laser in astronomy. *New Sci.* **18**(344), 672–673 (1963)
48. Rodrigues, J.M.F., Pereira, J.A.R., Sardo, J.D.P., de Freitas, M.A.G., Cardoso, P.J.S., Gomes, M., Bica, P.: Adaptive card design UI implementation for an augmented reality museum application. In: Antona, M., Stephanidis, C. (eds.) *UAHCI 2017*. LNCS, vol. 10277, pp. 433–443. Springer, Cham (2017). [https://doi.org/10.1007/978-3-319-58706-6\\_35](https://doi.org/10.1007/978-3-319-58706-6_35)
49. Rother, C., Kolmogorov, V., Blake, A.: Grabcut: Interactive foreground extraction using iterated graph cuts. In: *ACM Transactions on Graphics (TOG)*, vol. 23, no. 3, pp. 309–314. ACM (2004)

50. Rublee, E., Rabaud, V., Konolige, K., Bradski, G.: ORB: an efficient alternative to SIFT or SURF. In: Proceedings of International Conference on Computer Vision, pp. 2564–2571. IEEE (2011)
51. Sardo, J.D.P., Semião, J., Monteiro, J.M., Pereira, J.A.R., de Freitas, M.A.G., Esteves, E., Rodrigues, J.M.F.: Portable device for touch, taste and smell sensations in augmented reality experiences. In: Mortal, A., Aníbal, J., Monteiro, J., Sequeira, C., Semião, J., Moreira da Silva, M., Oliveira, M. (eds.) INCREaSE 2017, pp. 305–320. Springer, Cham (2018). [https://doi.org/10.1007/978-3-319-70272-8\\_26](https://doi.org/10.1007/978-3-319-70272-8_26)
52. Sermanet, P., Kavukcuoglu, K., Chintala, S., LeCun, Y.: Pedestrian detection with unsupervised multi-stage feature learning. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 3626–3633. IEEE (2013)
53. Serrão, M., Shahrabadi, S., Moreno, M., José, J.T., Rodrigues, J.I., Rodrigues, J.M.F., du Buf, J.M.H.: Computer vision and GIS for the navigation of blind persons in buildings. *Univ. Access Inf. Soc.* **14**(1), 67–80 (2015)
54. SM.: Science museum - atmosphere gallery (2017). <https://vimeo.com/20789653>. Accessed 04 Apr 2017
55. Sousa, L., Rodrigues, J.M.F., Monteiro, J., Cardoso, P.J.S., Semião, J., Alves, R.: A 3D gesture recognition interface for energy monitoring and control applications. In: Proceedings of ACE 2014, pp. 62–71 (2014)
56. Tian, Y., Luo, P., Wang, X., Tang, X.: Deep learning strong parts for pedestrian detection. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 1904–1912 (2015)
57. TWSJ.: The wall street journal: Best apps for visiting museums (2017). <https://goo.gl/cPTyP9>. Accessed 04 Apr 2017
58. Unity.: Unity3D (2018). <https://unity3d.com/pt>. Accessed 10 Jan 2018
59. Vainstein, N., Kuflik, T., Lanir, J.: Towards using mobile, head-worn displays in cultural heritage: user requirements and a research agenda. In: Proceedings of the 21st International Conference on Intelligent User Interfaces, pp. 327–331. ACM (2016)
60. Xiao, J., Zhang, J., Adler, B., Zhang, H., Zhang, J.: Three-dimensional point cloud plane segmentation in both structured and unstructured environments. *Rob. Auton.Syst.* **61**(12), 1641–1652 (2013)
61. Zhang, L., Lin, L., Liang, X., He, K.: Is faster R-CNN doing well for pedestrian detection? In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9906, pp. 443–457. Springer, Cham (2016). [https://doi.org/10.1007/978-3-319-46475-6\\_28](https://doi.org/10.1007/978-3-319-46475-6_28)