Playzones: A robust detector of game boards for playing visual games with robots

Arnaud Ramey
UC3M – Madrid, Spain
arnaud.a.ramey@gmail.com

Miguel A. Salichs
UC3M – Madrid, Spain
salichs@ing.uc3m.es

***Abstract*— We present here a generic vision-based method for detecting board games. This provides a convenient development tool for playing games with robots.**

This method only requires the user to draw a thick black square,called playzone, around the game board. The algorithm localizes this zone in the image provided by the camera and compute a corrected version of the image. The whole process requires a small computation time.

I. INTRODUCTION

Social robotics aims at making robots partners for applications such as entertainment, education, and assistance for daily tasks. Playing with robots is a whole field of investigation. To make the experience intuitive and enjoyable for the user, the process of interaction with the robot must be as straightforward as possible. The ultimate goal would be that the user could play with the robot in a similar manner he or she would with another human player. As such, the modifications and alterations applied to the rules of the game to make it playable with a robot should be as light as possible.

However, it is not easy to develop such interaction methods. The field investigated in this paper is board games. The colors, shape, size, and more generally the visual appearance of such games vary a lot between them. However, the development of such games for robotics is easier if is supplied a method for spotting where the game is in the data supplied by the sensors.

In this paper, we present a solution for a fast vision-based detection mechanism for board games. The robustness and efficiency of this solution has been verified through extensive experimentation. Furthermore, its extensibility has been proven by applying it for two board games, tic-tac-toe and hangman.

II. RELATED WORK

The need to locate a zone of interest in an image is one of the challenges of computer vision.

The classical way of locating a given spot or object is to use some vision-based features. Some popular ones are the SIFT and SURF markers ([2]), or more recently the BRIEF markers ([3]). These markers present a small computation time, robustness to light conditions and are scale-independent. However, they are highly bound to the visual appearance of the zone of interest. That is, there cannot be some common interface able to locate board games for different games based on such features.

An approach to tackle this problem is the use of fiducial markers ([5]; [6]). It consists in a system of patterns added to the environment. Some vision-based algorithms locate these visual markers and succeed to estimate their position and orientation. As such, information about the position in the environment of the marked objects is obtained. A system such as ARTag was proven robust to poor lightning condition. However, the production of these markers requires some knowledge that the average end-user might not have.

Turning the robot into a board game partner has been performed already. However, most proposed methods were ad-hoc solutions, using special devices. For instance, the checkers playing robot presented in [9] uses a given configuration with a camera mounted on top of the game and a constant illumination. The board game position is manually annotated by the user at the beginning of the game. In [11], the toy robot developed in the project IROMEC, used to help through games children with special needs, is able to play some board games, but with a special interface. In [15], it is made use of a giant checkerboard. The robots, namely Sony AIBOs, as well as the human players, are used as pieces of this checkerboard. In [13], a framework for playing games with AIBOs has been developed. The vision analysis is based on the fast detection of ellipses and lines, the detected objects are then projected into a bird view perspective. Such a system is applied for playing tic-tac-toe. However, the extensibility of this system to other games is limited.

III. PROBLEM STATEMENT

A. Hardware specifications

"Maggie" is one of the research robots in the Robotics Lab of the University Carlos III, in Madrid. It is a platform for studying human-robot interaction, that is the ways to adapt the robotics potentials to provide to human users new ways of working and entertaining. An illustration of Maggie is below, on Figure 1. Here is a summary of the extensive description of Maggie in [1].

Maggie is designed as a 1.35 meters-tall girl-like doll. Her base is motorized by two differentially actuated wheels and a caster wheel on both sides. It is also equipped with 12 bumpers, 12 infrared optical sensors and 12 ultrasound sensors. Above the base, a laser range finder (Sick LMS 200) has been added. The upper part of the robot incorporates the interaction modalities. On top of the platform, there is an anthropomorphic robot head with an attractive design. The head has two degrees of freedom, while each arm has one

Fig. 1: The robot Maggie.

degree of liberty. Invisible touch sensors are integrated in several parts of the body, such as the shoulders, the hands and the head.

These features are illustrated in Figure 2.

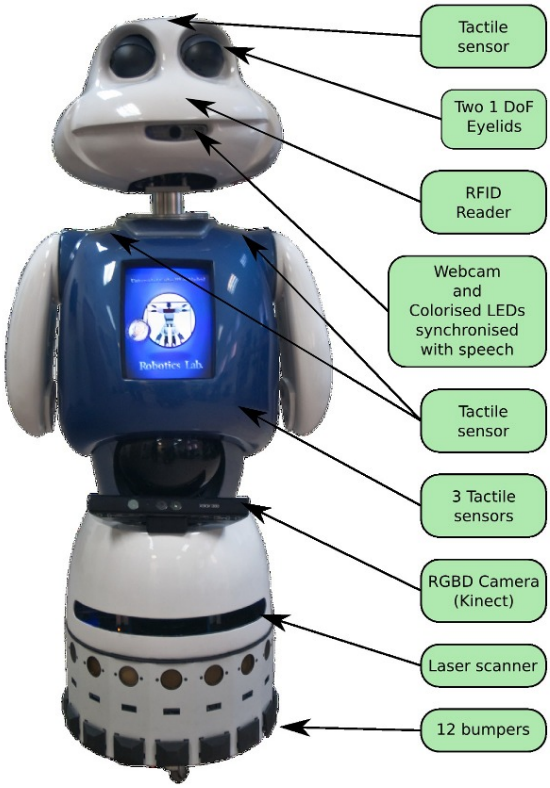

Fig. 2: The hardware equipping Maggie.

Maggie relies on a main computer hidden inside her chest and controlling all its skills. A *tablet PC* on her belly allows information display or user interaction through touch inputs. For image acquisition, the camera (hidden in the mouth), is a Logitech QuickCam Pro 9000, with a resolution of 640x480 pixels and a frame rate of 30 fps. A Microsft Kinect is mounted at the belt, enabling acquisition of depth images. The integration of this device on Maggie is presented in [10].

B. Software architecture

The architecture in Maggie relies on the AD organization (Automatic-Deliberative), as described in [1], which handles **skills** relying on **primitives**.

Primitives are in direct communication with the physical devices of the robot, and send elementary orders to them. This includes the base actuator, the head actuator, the laser sensor, the tactile sensors, etc.

A skill is the ability of Maggie to do a specific action. It relies on the data supplied by the sensors (such as the touch sensors on the body, the camera, the laser, etc.). The actions triggered by a skill can be numerous: move a part of the body, say something, connect to a website, etc. The fundamental inputs and outputs are made using primitives. The algorithms that will be presented here were implemented as skills in the global architecture of Maggie.

The voice generator is provided by professional software wrapped into the global architecture of Maggie as a skill. The generated sound is then emitted through loudspeakers situated in the neck of Maggie. It is possible to communicate with Maggie using the voice: the sound is relayed through a headset and understood by Maggie with a speech recognizer using a grammatical based knowledge.

IV. APPROACH

The idea behind our solution is explained in subsection IV-A. In subsection IV-B, we will give an overview of how the algorithm to find the board game is structured.

The algorithm for finding the playzone goes by several independent steps, each of them using different mathematical concepts. That is why, in subsection IV-C, subsection IV-D and subsection IV-E we will explain the detail under each part of the algorithm.

A. Overview of the method

As seen before, the diversity of the appearance of board games prevents from using a generic algorithm for finding them. Some additional information, or *markers*, has to be added. That is why we choose to indicate visually the border of the board game using a thick, black, square-shaped marker around the game. It then looks like a black squared frame surrounding the game. [1]

The algorithm to find this marker is then vision based. It will find the black shape in the image, and transform its content to fit the square shape corresponding to a top (bird) view.

[1] This black square can be produced by the end user through several means. For instance, it can be drawn, using any black marker, on the sheet of paper that is used for the board game. This method presents the advantage of needing very little hardware and not being constrained by the size of the game. It is also possible to build a black frame of cardboard, wood or any other rigid material. This then can be reused for different instances of a game, but constrains the maximal size of the games, that have to fit into this frame.

B. Description of the algorithm

Let I be the image supplied by the camera. Typically, I is a VGA image, that is a 640×480 image with three layers. A sample is visible in Figure 3.

Fig. 3: An example of acquired frame I. We want to get a bird's view perspective of the game board, marked with the black thick marker.

At the same time, we will describe the algorithm and we will provide captures from the effective treatment of an image by the program.

Our purpose here, as presented in subsection IV-A, is to find the playzone in I and obtain a rectified version of it in a normalized image PZ. We want the image PZ to be of size (W, H) where $W, H \in \mathbb{N}$ are values chosen by the user, for instance $W = 300$ and $H = 300$.

1) First of all **we transform the image I to a gray-scale version** I_g ([14]).
2) Then, **we threshold** I_g with into a monochrome image I_t. We use an adaptive threshold, as described in [8]. This technique uses a variable threshold for each pixel, depending on the neighbor colors. As such, it allows stressing the local contrast generated by the black shape of the marker. This is more robust to light conditions than a simple threshold. A sample is in Figure 4.

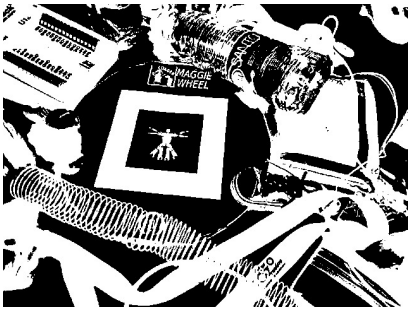

Fig. 4: I_t, the acquired frame, thresholded

3) **We get all the connected components of** I_t, using the disjoint-sets data structure and union-find algorithm explained in subsection IV-C, as shown on Figure 5.
4) **We select the best connected component** P of I_t, using the MHD distance explained in subsection IV-D. A sample is visible in Figure 6.
5) **We get the corners of this component**. We use some algorithms presented in subsection IV-E. Cf Figure 7.

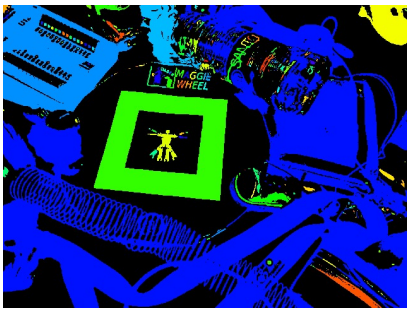

Fig. 5: Illustration of all the connected components of I_t: each one is of a different color. However, as there are many components in this image, and the image is only 256 colors, it is possible that distinct components have the same color. The components are actually stored as vectors of points (corresponding to the non-black pixels).

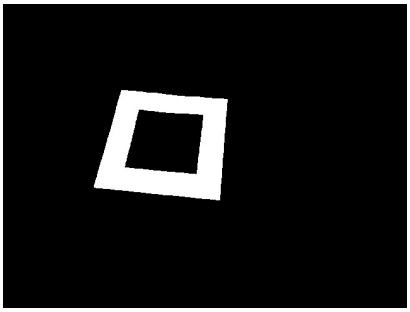

Fig. 6: Illustration of P, the best kept component of the thresholded image I_t.

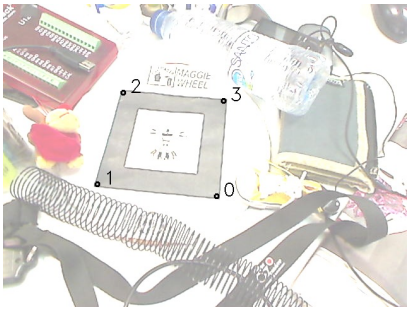

Fig. 7: The outer corners obtained through geometrical analysis.

6) **We find the inner corners of the playzone**. The playzone is shaped as a black squared frame. This is then done by evaluating the average thickness of the black pixels on the four sides (north, south, east, west) of the quadrilateral shape found in the previous step. Cf Figure 8.
7) **We compute the homography** matching these four corners in I to the four corners $(0, 0)$; $(0, H)$; (W, H); $(W, 0)$ of the normalized square image P.
To make sure that each corner of I is associated to the good one of P, we previously sort these four corners of I according to the angle they form with the center of the component.
The resulting homography is a 3×3 real matrix that

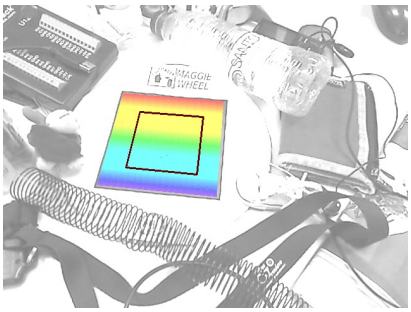

Fig. 8: An illustration of the process to obtain the inner corners of the playzone. The outer shape of the playzone is swept by an horizontal scan-line which evaluates the median thickness of the black border on the left and right border. A vertical scan-line gives the up and down thicknesses. The upper left inner corner, for instance, is located at the intersection of the average left border and upper one. On this illustration, the horizontal scan-lines are represented. Their color goes from red, for the first scan-line, to blue, for the last one.

we will name H (further explanation, see [12]).

8) **We rectify the image** I, by applying the homography matrix H: the result is our target image PZ.
We also apply a 180^o rotation to get the image from the player point of view. For instance, if, later, Maggie refers to the left of the board, it will be the left side of the board for the player, which is much more natural. We finally obtain a top view of the board game, without the marker: this was our goal. A sample is visible in Figure 9.

Fig. 9: Final image PZ, rectified and cropped. Note the 180^o rotation.

Now comes some additional information about some key steps of the algorithm.

C. Finding components: Union Find using the Disjoint-Sets data structure

We consider a set of points $P = \{p = (x, y) \in \mathbb{N}^2\} \in \mathbb{N}^{2\mathbb{N}}$. In fact, these points represent the non-null points of an image.

A **connected component** of P is a subset C of P points such as $\forall c \in C, \exists \tilde{c} \in C, \|c - \tilde{c}\|_{L_1} = max(\mid c.x - \tilde{c}.x \mid, \mid c.y - \tilde{c}.y \mid) = 1$.

That corresponds to the usual definition of connected components for non-oriented graphs, supposing that two points at an Euclidean distance of 1 on P are two linked nodes in the corresponding graph. This equivalence is illustrated in Figure 10.

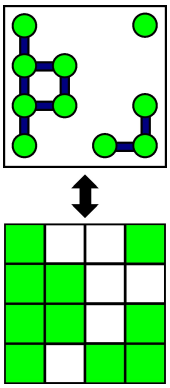

Fig. 10: Illustration of the correspondence between connected components in images and graphs.

Now, we look for a quick way to find every connected component of P. More accurately, it is to obtain a partition $\{P_i, 1 \leq i \leq n\}$ of P, where n is the number of connected components of P and $\forall i \in [1;\ n]$, P_i is a connected component. This is illustrated in Figure 11. There are many ways to make it, we chose to use the **Union-Find** algorithm with the **Disjoint-Sets** data structure.

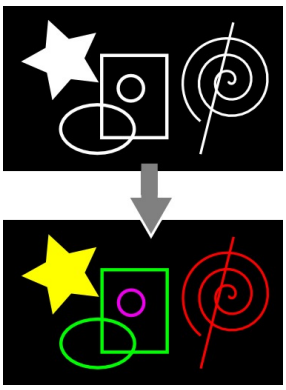

Fig. 11: The goal of this part is to extract the components. Above, the original image: each non-black point represents a point of P. Under are the connected components $\{P_i\}$. Each component has been drawn in a different color.

a) Disjoint sets: The disjoint-sets data structure was first presented in [7] in 1964. A set is a structure of tree, in which each node has a reference to his father (while it is references to his sons in the classic tree structure). A disjoint-set forest is a list of sets.

b) Union Find: A union-find algorithm supplies a disjoint-set forest two functions: `union` and `find`. The former allows the fusion of two sets, while the latter finds the root of the tree in which is located the supplied argument.

c) The algorithm: we can give a pseudo-code version of the algorithm.

---

proc $FindComponents$(List of point P, int $width$, int $height$) $\equiv$
computing the disjoint set

```
  foreach p = (x, y) in P; do
    createNode (x, y)
  od
  for y := 0 to height step 1 do
      for x := 0 to width step 1 do
          Node curr = Node (x, y);
          Node up = Node (x, y − 1);
          Node left = Node (x − 1, y);
          if x > 0 ∧ y > 0 ∧ exists up ∧ exists left
            then Union(up, curr);
                 Union(left, curr);
             else if x > 0 ∧ exists left
                    then Union(left, curr); fi
             else if y > 0 ∧ exists up
                    then Union(up, curr); fi
          fi
      od
  od
                                          now create the lists
  create A, an array of size width × height of blank point lists;
  for y := 0 to height step 1 do
      for x := 0 to width step 1 do
          Node curr = Node (x, y);
           add (x, y) to A[curr.father];
      od
  od
                                          now collect the results
  create R, a list of list of points;
  for y := 0 to height step 1 do
      for x := 0 to width step 1 do
          if A[y ∗ width + x] not empty
            then add A[y ∗ width + x] to R; fi
      od
  od
  return R;
.
```

D. Recognizing the best one: Modified Hausdorff Distances

Let us consider a list of connected components as $CC = \left\{C \in \mathbb{N}^{2\mathbb{N}}\right\} \in (\mathbb{N}^{2\mathbb{N}})^{\mathbb{N}}$, obtained for instance with the method described in subsection IV-C. We also consider a model component $M = \left\{m = (x, y) \in \mathbb{N}^2\right\} \in \mathbb{N}^{2\mathbb{N}}$.

The objective in this section is to select the component of CC which matches the best with the shape of M.

First, we re-normalize all the components of CC. The reason of doing so is that we cannot get any prior information about the size of the playing zone. From now, the components of CC are considered as normalized.

We have now to compare a normalized set of points, M, to every normalized set of points of CC. The idea is to compute a distance between M and every component CC of C, and to keep the member of C which realize the best result.

In [4] is made a comparison between the different ways of computing a distance between two sets of points. According to these findings, we use the distance D_{22}, defined as:

$$\forall A, B \in \mathbb{R}^{2\mathbb{N}},$$
$$D_{22}(A, B) = max d_6(A, B), d_6(B, A)$$
$$\text{with } d_6(A, B) = \frac{1}{absA} \sum_{a \in A} d(a, B)$$
$$\text{and } d(a, B) = \min_{b \in B} \|a - b\|$$

Using this distance, we can now iteratively compute the distance of each component of C and keep only the best. It is possible to add a maximum threshold value to avoid returning a component when the playzone is nowhere to be found on the screen.

In pseudo-code, such an algorithm would be written:

```
proc d_6(A, B) ≡
  r = 0;
  foreach a in A; do
    d_ab = INFINITY;
    foreach b in B; do
      d_ab = min(a − b, d_ab); od
    r = r + d_ab;
  od
  r = r/ | A |;
  return r;
.
proc d_22(A, B) ≡
  return max(d_6(A, B), d_6(B, A));
.
proc selectComponent(CC, M) ≡
  best = INFINITY;
  foreach C in CC; do
    if d_22(C, M) < best
      then best = d_22(C, M);
           rep = C;
    fi
  od
  return rep;
.
```

d) Examples: Let us provide two examples to show the capacities of this method.

Example 1: Table I shows a few test images and their distances with the model image, obtained with our comparator.

TABLE I: Results of the image comparator on an example. Above, the model image of our comparator. Below, the test images and their distance.

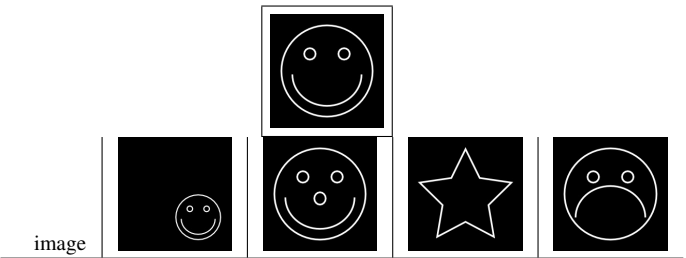

image				
distance	0.608	3.605	16.866	3.863

Image number 1, which is the same as the model with a scaling, gets the most little distance. [2]
Image number 3, very different from the model, gets a higher mark and is immediately discarded by the comparator.

Example 2: we apply the same method to a real input image obtained from the robot camera. The model image is a monochrome playzone shape. The results are gathered in Table II. We can see that the correct corresponding connected component gets the lowest mark from the comparator.

[2]We could wonder why we do not get a distance of zero with this image. The reason is simple: the scaling of the image did not preserve the ratio (border thickness) / (image width). Therefore, the rescaled image has a thicker border, and so some points are distant from the points of the model image.

TABLE II: Results of the image comparator on a real image.
Above on the left, the model image of our comparator: the shape of the playzone.
Above on the right, the thresholded input image, where each component is drawn with a different color.
Below, some connected components and their distance.

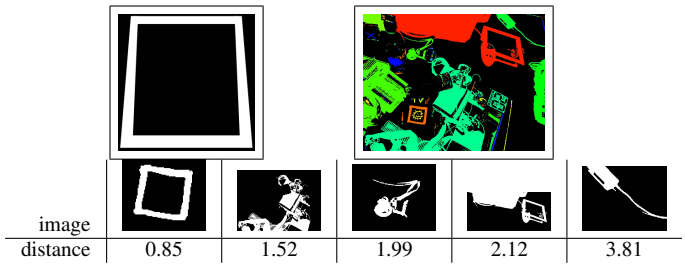

image					
distance	0.85	1.52	1.99	2.12	3.81

E. Finding the corners of a connected component

We consider a set of points P representing a connected component, according to the definition seen in subsection IV-C.

We know this component is the representation of a square with a thick border (our playzone), on which was applied a homography (distortion due to the projection matrix of the camera).

The corners detection can be obtained through several different algorithms in our implementation. These are geometric algorithms, most of them based on the analysis of the function that associates a direction in $[0, 2\pi[$ to the furthest point of the center of the connected component in that direction.

V. EXPERIMENTAL RESULTS

The previously presented method has been implemented on the robot Maggie seen in subsection III-A. It relies on the AD architecture seen in subsection III-B.

Samples of the algorithm applied to a tic-tac-toe game and a hamgman game is visible in Figure 12. and Figure 13 (the latter being visible at the end of the article).

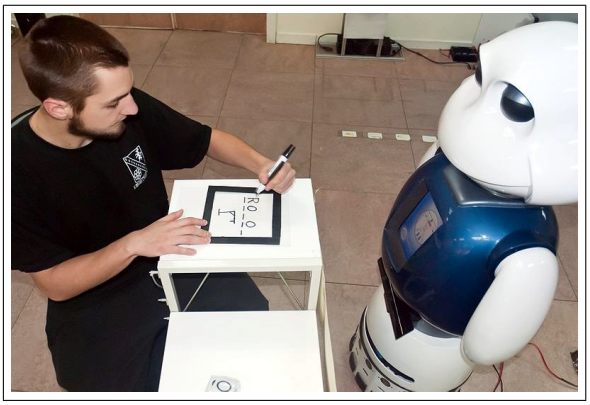

Fig. 12: A sample image of a human player using the playzone mechanism for interacting with Maggie.

A database of around 70 sample images, provided with the camera mounted on the robot, has been used to measure the performances of the algorithm. These images are regular 640×480, 3-channels images. The pictures present different point of views and lighting conditions, some of them show a clean background while others are full of objects. Some pictures are challenging enough so as to reaching a 100% success rate over the whole database is not reachable.

Each image was first manually annotated. The user specifies via a graphical user interface the correct position of the four corners of the playzone for each image.

A. Time performance

The database previously presented was evaluated on different CPUs.

The average times of detection over the whole database is visible in Figure 14.

For modern desktop PC or laptops, the average computation time is less than 50 milliseconds. This allows the use of the playzone mechanism for real-time applications. without any further modifications.

For CPUs similar to the ones found on embedded computers, however, the computation time is over 100 milliseconds. This is unfortunately not fast enough for a real time detection without lagging.

However, such a processing time is satisfying if used with turn-to-turn games. For instance, for a checkers game detecting the playzone in a fraction of seconds does not present an handicap for the interaction experience of the player.

Another bypass could be to lower the resolution of the input image. As a drawback, however, the resolution of the final corrected playzone content is also reduced. This can trigger problems for games with a high level of details.

A detail of the costs of each step is presented in Figure 15 (visible at the end of the article). We can notice the most time-costly steps are the computation of the connected component and the final rectification of the image. These two steps almost represent two thirds of the total computation time.

B. Noise resistance

On each image of the database, we apply a noise of given amplitude. That is, we blend this image with a uniform noise image with a given transparency index.

For amplitude of 0, the image is not altered at all, while 1 is equivalent to a purely noised image. This is shown on Figure 16 (visible at the end of the article).

Each image of the set is submitted to the playzone detector, and the found location of the playzone is compared with the manually annotated one. We consider that a playzone is correctly detected if four corners are found and the distance of each corner to the four correct corners is less than 5% of the longest vertex of the correct playzone.

The results are visible in Figure 17. Note that for a noise greater than 85%, the detection is not performed successfully anymore. We can however add that the image is hardly understandable even for an human eye.

The successful detections rate drops under 80 % of its peak value (obtained for images with no noises) on our image database for a noise superior to 50 %.

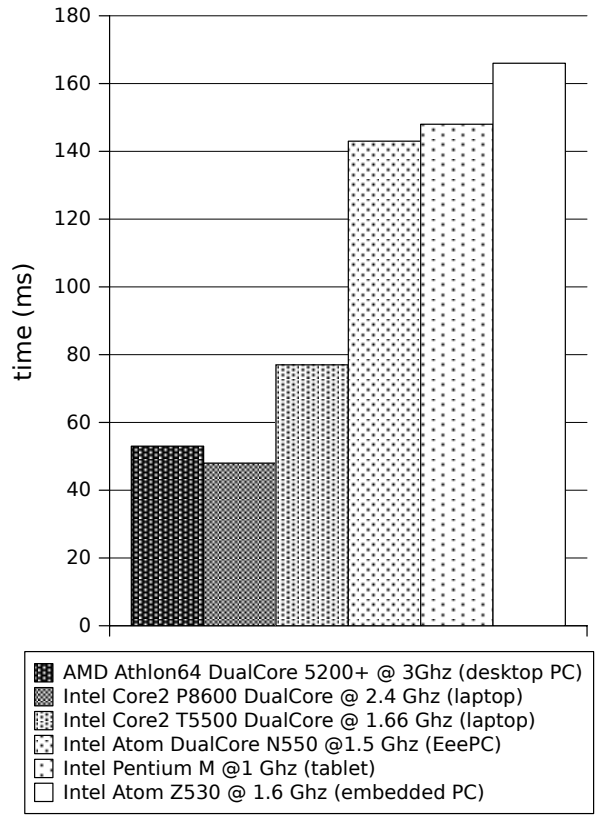

Fig. 14: The average detection time for the database of images. The images are of size 640×480.

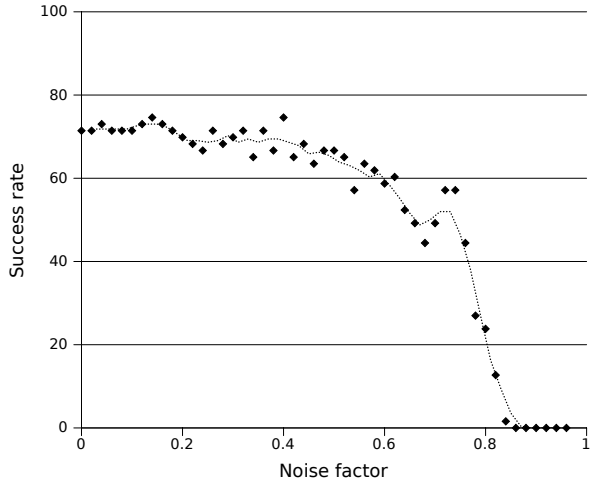

Fig. 17: Effect of a noise applied to the input images database. As the noise applied is randomly generated, the obtained curve is not continuous.

VI. CONCLUSIONS AND FUTURE WORKS

The presented method allows having a fast and robust detector for board games. Thanks to the use of the additional marker, we manage to make abstraction of the diversity of appearances that such a game can present.

However, the time performance does not allow a real-time detection of the playzone for less powerful computers such as embedded computers. We are currently addressing this issue using some tracking algorithms. Furthermore, the presented method does not cope well with partial occlusions of the playzone, for instance by the human user arm. The tracking also aims at solving this issue by searching partial playzones in the vicinity of the last detected position.

Some further optimization is also ongoing, especially concerning one of the slowest steps, the computation of the connected components of the image.

An integration of the whole playzone module is currently ongoing with ROS, the Robot Operating System [3].

VII. ACKNOWLEDGMENTS

The authors gratefully acknowledge the funds provided by the Spanish Government through the projects called Peer to Peer Robot- Human Interaction (R2H), of MEC (Ministry of Science and Education), and A New Approach to Social Robotics (AROS), of MICINN (Ministry of Science and Innovation).

REFERENCES

[1] R. BARBER AND M. SALICHS, *A new human based architecture for intelligent autonomous robots*, Elsevier, 2002, pp. 85–90.

[2] H. BAY, A. ESS, T. TUYTELAARS, AND L. VAN GOOL, *Speeded-up robust features (surf)*, Comput. Vis. Image Underst., 110 (2008), pp. 346–359.

[3] M. CALONDER, V. LEPETIT, C. STRECHA, AND P. FUA, *Brief : Binary robust independent elementary features*, Computer, 6314 (2010), pp. 778–792.

[4] M.-P. DUBUISSON AND A. JAIN, *A modified hausdorff distance for object matching*, in Pattern Recognition, 1994. Vol. 1 - Conference A: Computer Vision Image Processing., Proceedings of the 12th IAPR International Conference on, vol. 1, Oct. 1994, pp. 566 –568 vol.1.

[5] M. FIALA, *Artag, a fiducial marker system using digital techniques*, Computer Vision and Pattern Recognition, IEEE Computer Society Conference on, 2 (2005), pp. 590–596.

[6] M. FIALA, *Comparing artag and artoolkit plus fiducial marker systems*, in Haptic Audio Visual Environments and their Applications, 2005. IEEE International Workshop on, oct. 2005, p. 6 pp.

[7] B. A. GALLER AND M. J. FISHER, *An improved equivalence algorithm*, Commun. ACM, 7 (1964), pp. 301–303.

[8] A. JAIN, *A fast karhunen-loeve transform for digital restoration of images degraded by white and colored noise*, Computers, IEEE Transactions on, C-26 (1977), pp. 560 –571.

[9] D. LEWIS AND D. G. BAILEY, *A checkers playing robot*, Massey University, Palmerston North, NZ, November 15-16 2004.

[10] A. RAMEY, V. GONZÁLEZ-PACHECO, AND M. A. SALICHS, *Integration of a low-cost rgb-d sensor in a social robot for gesture recognition*, in Proceedings of the 6th international conference on Human-robot interaction, HRI '11, New York, NY, USA, 2011, ACM, pp. 229–230.

[11] B. ROBINS, E. FERRARI, AND K. DAUTENHAHN, *Developing scenarios for robot assisted play*, in Robot and Human Interactive Communication, 2008. RO-MAN 2008. The 17th IEEE International Symposium on, aug. 2008, pp. 180 –186.

[12] J. SEMPLE AND G. KNEEBONE, *Algebraic projective geometry*, Oxford classic texts in the physical sciences, Clarendon Press, 1998.

[13] E. TIRA-THOMPSON, N. HALELAMIEN, J. WALES, AND D. S. TOURETZKY, *Tekkotsu: Cognitive robotics on the Sony AIBO*, Citeseer, 2004.

[14] W. WHARTON AND D. HOWORTH, *Principles of Television Reception*, Pitman Paperbacks Series, Pitman Publishing, 1971.

[15] M. XIN AND E. SHARLIN, *Exploring human-robot interaction through telepresence board games*, Lecture Notes in Computer Science, (2006), pp. 249 – 261.

[3] http://ros.org

Fig. 13: Application of the playzone method to two popular board games: tic-tac-toe and hangman. The image analysis of the board game is much easier in the corrected, cropped image than in the raw input one.

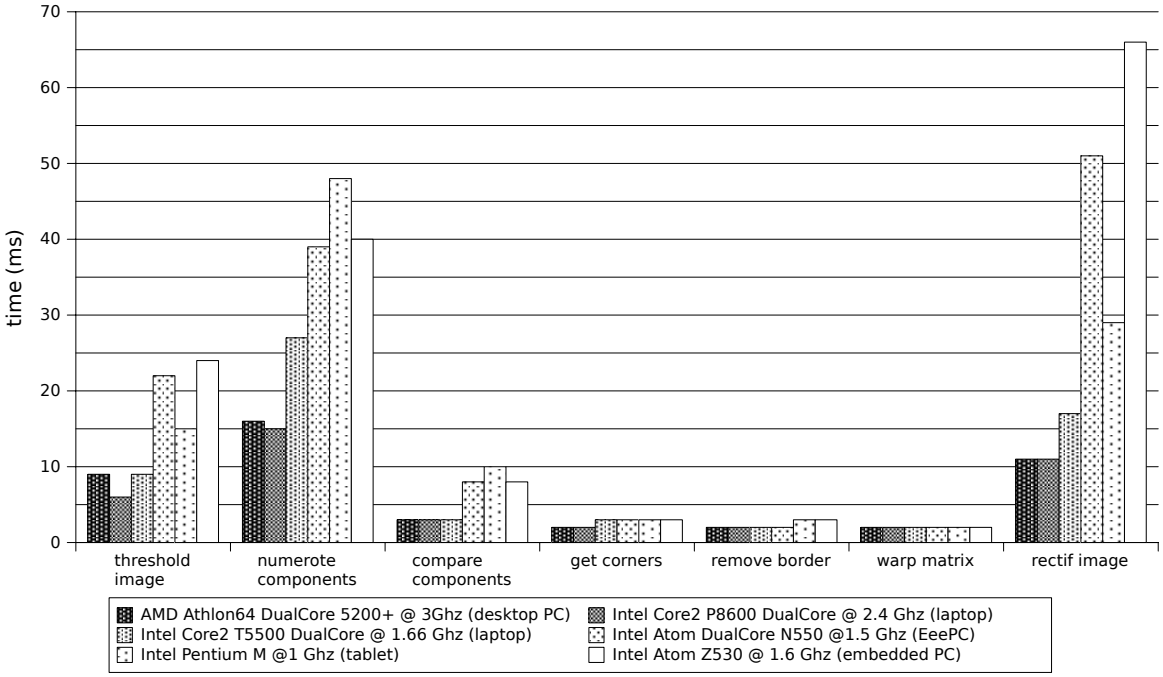

Fig. 15: The times costs for each step, in milliseconds. The total detection and rectification time corresponds to the sum of the times for each step.

Fig. 16: Effects of the noise factor on a sample image. The first noised picture corresponds to a 50% noise, the second to a 75% noise. With such a noise, the playzone becomes hard to recognize even for the human eye.