Robust Real Time Color Tracking

Mark Simon, Sven Behnke, and Raúl Rojas

Free University of Berlin
Institute of Computer Science
Takustr. 9, 14195 Berlin, Germany
{simon|behnke|rojas}@inf.fu-berlin.de
http://www.fu-fighters.de

Abstract. This paper describes the vision system that was developed for the RoboCup F180 team FU-Fighters.
The system analyzes the video stream captured from a camera mounted above the field. It localizes the robots and the ball predicting their positions in the next video frame and processing only small windows around the predicted positions. Several mechanisms were implemented to make this tracking robust. First, the size of the search windows is adjusted dynamically. Next, the quality of the detected objects is evaluated, and further analysis is carried out until it is satisfying. The system not only tracks the position of the objects, but also adapts their colors and sizes. If tracking fails, e.g. due to occlusions, we start a global search module that localizes the lost objects again. The pixel coordinates of the objects found are mapped to a Cartesian coordinate system using a non-linear transformation that takes into account the distortions of the camera. To make tracking more robust against inhomogeneous lighting, we modeled the appearance of colors in dependence of the location using color grids. Finally, we added a module for automatic identification of our robots.
The system analyzes 30 frames per second on a standard PC, causing only light computational load in almost all situations.

1 Introduction

In the RoboCup Small-Size (F180) league, five robots on each team play soccer on a green field marked with white lines. The ball is orange and the robots, as well as the goals, are marked either yellow or blue. In addition to the yellow or blue team marker (a ping-pong ball centered on top of the robot), further markers are allowed, as long as they have different colors (refer to [6] for more details).

The robots are controlled by an external computer connected to a camera mounted above the field, such that the entire field is visible, as shown in Figure 1. The task of the vision system is to compute the positions and orientations of the robots, as well as the position of the ball. The behavior control software uses this information to operate the robots, relying on visual feedback. Since the robots and the ball move quickly and vision is usually the only input for behavior control, a fast and reliable computer vision system is essential for successful

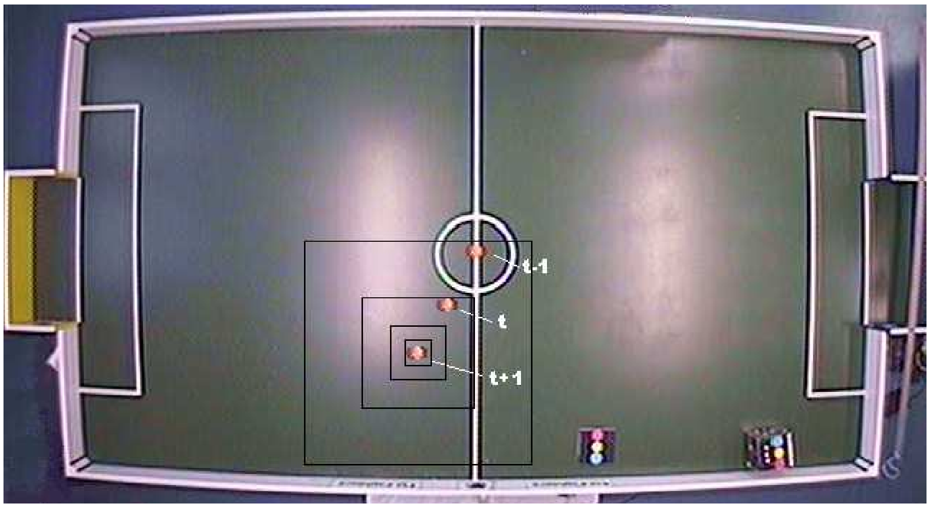

Fig. 1. A typical camera image, showing the field with shadows at the walls and reflections in the center. The linear ball prediction and a variable search window are shown too. The robots are marked with three colored dots.

play. Further information about the overall system and the hierarchical reactive control of the F180 team FU-Fighters can be found in [1] and [2].

Appropriate coloring of the interesting objects partially simplifies the vision problem, but does not make it simple. There are several problems. First, the interesting objects are not always visible. The ball can be occluded, due to the central camera mount and vertical sides of robots. Also, the robots can be occluded by people that move them manually, e.g. to place them at their kickoff positions.

Inhomogeneous lighting is also problematic. There are shadows from the robots, the walls, and the referee, as well as highlights from the spot lighting. These conditions are far from a homogeneous diffuse lighting that would give the objects a constant appearance.

The vision problem is also complicated by the undefined background next to the field and the variability of the robots markers. All colors are allowed for the markers, as long as they are different from the reserved ones, and also the number and form of the markers changes from team to team. The vision system must be able to adapt quickly to the different markers of the opponents.

The non-linearity of the cameras wide angle optical system has also to be taken into account. The straight walls of the field appear to be convex in the captured image. One has also to correct for the height of the objects when mapping them to a 2D standard coordinate system.

The remainder of the paper is organized as follows. In the next section, we describe the robust tracking method, we developed. Then, we explain the non-linear coordinate transformation used. Section 4 presents improvements that we incorporated into the system prior the Melbourne competition: (a) the color map approach that models the appearance of colors dependent on the position; and (b) automatic identification of our robots.

2 Robust Tracking

The input for our vision system consists of an analog video stream produced by an NTSC S-Video camera mounted above the field. We capture the image using a PCI frame grabber at a resolution of 640×480 pixels in RGB format. The frame rate is 30fps. This produces an enormous data rate of 26MB/s from which we want to extract the few bytes relevant to behavior control. We need to estimate the positions of the ball and the opponent robots, as well as the positions and orientations of our robots.

The vision system analyzes only those parts of the image containing the field. The background is ignored. This reduces the data rate, but it would still not be possible to analyze the entire field in every frame without using special purpose hardware.

We therefore decided to develop a tracking system that predicts the positions of the interesting objects and analyzes only small windows around them. With a high probability, the objects will be within these small windows and we do not have to process most parts of the image.

Many other teams at RoboCup'99 relied on special hardware, like FPGAs or DSPs to process the entire image in real time [3, 4].

2.1 Ball and Robot Models

The orange ball, and the blue and yellow team markers appear as colored dots of about constant size in the image. The form of these dots is not always circular, since moving objects are captured by the camera at different positions for different half images. This causes significant differences between the odd and even image lines. The lighting and the ball shape of the objects produces highlights and shadows. However, most of the pixels belonging to an object have similar colors.

We model the ball and the team marker with its position and size, as well as its color in HSI space. In addition to the team marker, we put two colored dots on top of our robots, such that they form a line from the left to the right wheel (refer to Figure 1 for a top down view). These dots are modeled in the same way as the ball and the team markers.

The robot model contains also information about the distance of the dots and the orientation of the line. For the tracking of other team's robots we have further models with only one or two dots. We initialize the models by clicking with the mouse on the dots and we assign an ID to each of our robots.

2.2 Variable Search Windows

At any given time, most objects move slowly on the field and only some objects might move fast. This allows to predict the positions of dots from the difference of the last two positions. We use linear prediction clipped at the borders of the field (see Figure 1). If an object moves fast, this prediction is not very accurate. Therefore, we need a larger search window for this situation. However, if the object stands still, the prediction will be very good and a small search window suffices.

We developed an algorithm for dynamic adjustment of the search window sizes. If we find the dot within its window, we slowly decrease the window's side length for the next frame. If the search is not successful, we increase the size by a factor of two and search again, as illustrated for the ball in Figure 1. The size is always limited between a minimal (e.g. 16×16) and a maximal value (e.g. 128×128) and the searched region is clipped by the rectangle containing the field. During regular play, the average size of the search windows is close to the minimal value. Large windows are only necessary, when objects move very fast (e.g. the ball has been kicked) or dots cannot be segmented (e.g. due to the lighting or occlusions).

One problem when enlarging the search windows for the robot's dots is that one can find dots of the same color that belong to different robots. To prevent this, we use two heuristics. First, we remove dots of found robots from the image by "painting" them black. Second, we enlarge a search window only after all dots have been searched for in their small window.

2.3 Color Segmentation

To find a colored dot in its search window, we first determine the pixel that has the smallest RGB-distance to the color of the dot's model. We assume that this pixel belongs to the dot and try to segment the remaining pixels from the background by analyzing a quadratic window with a side length of about twice the dot's diameter that is centered at the selected pixel.

We use two methods for color segmentation. The first method that is illustrated in Figure 2 works in HSI color space [5]. Since we are looking for colorful dots, we apply a saturation mask and an intensity mask to the window. Only pixels that are saturated and neither too bright nor too dark are analyzed further. The final segmentation decision is done using the hue distance to the model's color. Working in HSI space has the advantage that the method is quite insensitive against changes in intensity. However, it can only be applied to saturated dots. If the model dot is not saturated or the HSI segmentation fails, we try to segment the dot in RGB color space. This backup method is needed, since some teams might use unsaturated markers, and also since saturated markers may appear as unsaturated dots when viewed in shadow or hot spots. Here, we use only the RGB-distance between the pixels color and the color of the model. All pixels of similar color are segmented as belonging to the dot.

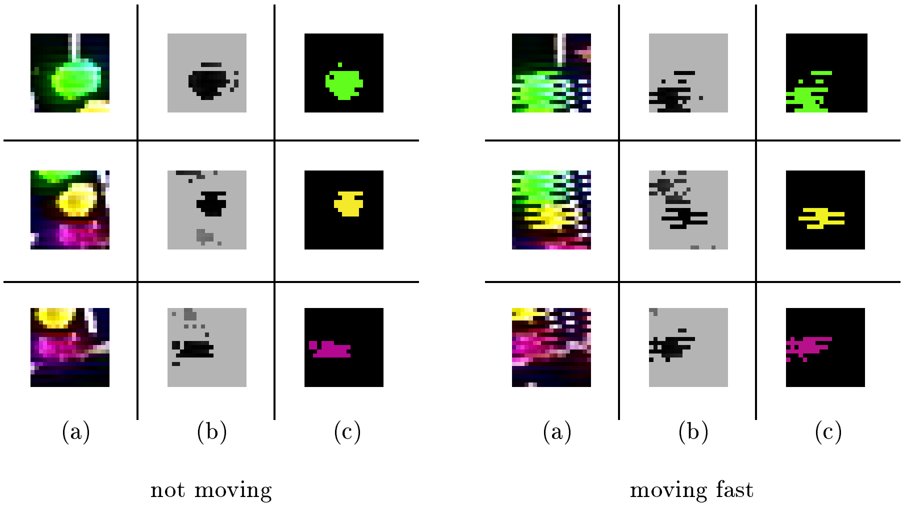

Fig. 2. Segmentation of colored dots in HSI space. Shown are (a) the original images, (b) the saturation/intensity masks, (c) and the segmented pixels, that have been selected using the hue distance to the model.

A quality measure is computed for the dot from the similarity to its model. If the size and the color are similar to the model, then the quality is high. The dot's position is updated if its quality is high enough and the dot is part of a valid robot, or if it is the ball. We estimate the position of the dot with sub-pixel resolution as the average location of the segmented pixels. If the quality is good, the size and average color of the dot are adapted slowly, as long as they do not deviate significantly from the initial model.

The segmentation does not explicitly take the shape of the dot into account. If enforces compact dots by searching many pixels in a small quadratic window.

2.4 Robot Search

To find a robot, all dots that belong to its model are searched for. If all dots can be localized, we compute the robot's quality from the degree of similarity to its model. We take into account the quality of the individual dots and their geometry, e.g. the distance between them.

For our robots, where three colinear equidistant colored markers are present, we check if the dots found fulfill this geometric constraint. If only one of the three dots is not found with sufficient quality, we can exploit the redundancy of our model. From the locations of the two dots found, we know where the third one should be and can check if it is really there.

We update the model of the robot only if the quality of the robot found is high enough. The position is adapted faster and more often than the geometry.

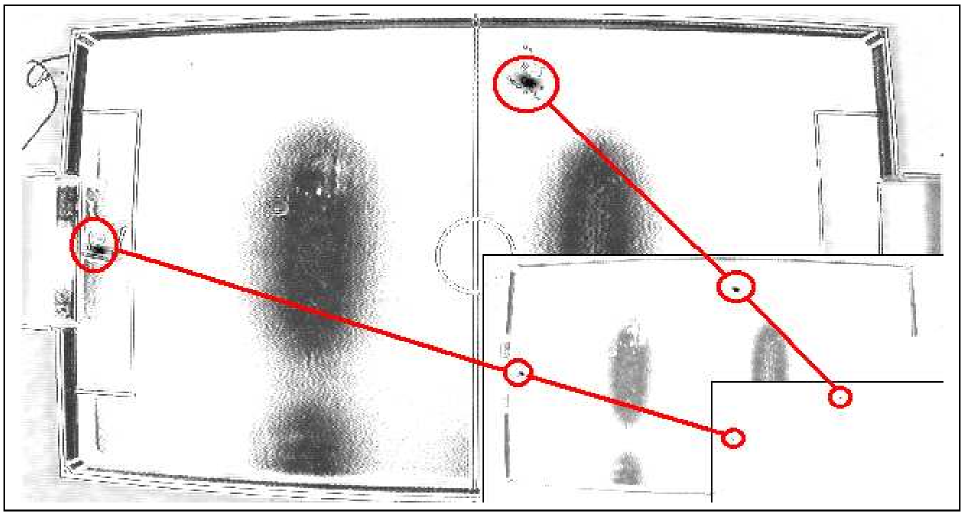

Fig. 3. Global search for a color. Shown are the differences to the desired color at three resolutions. Marked are two dots that have the most similar colors.

To prevent the tracking of non-robots, we count how frequently the quality of the robot found is low. If the robot is found for a certain time with low quality, then it is considered lost and a global search is performed to find it again.

If we lose objects, we assume that they are still at the position where we saw them last and signal this to the behavior control. If we lose the ball next to a robot that is near to the goal line, we assume that this robot occludes the ball and update its position together with the robot. This feature proved to be useful in penalty situations.

2.5 Global Search

The global search method is needed if robots are lost, e.g. due to occlusions. It is not needed for the ball, since we enlarge its dynamic search window until it covers the whole field.

Global search analyzes the entire field after the found objects have been removed from the image. We search for the colors of the dots contained in the models of the lost robots and try to combine them to valid robots.

To reduce pixel noise, we spatially aggregate the color information as follows. First, we compute the distance of all pixels to the desired colors in RGB color space. Next, we subsample these distance maps twice, taking each time the average of four pixels, as shown in Figure 3. Now, we find the best positions for each color, taking into account minimal dot distances. Using the lists of dots found, we search for combinations of dots that fulfill the geometrical constraints of the robot models. This search starts with the dots that have the highest quality. If more than one robot from a team is lost than we assign the found robots to the models that have the closest positions.

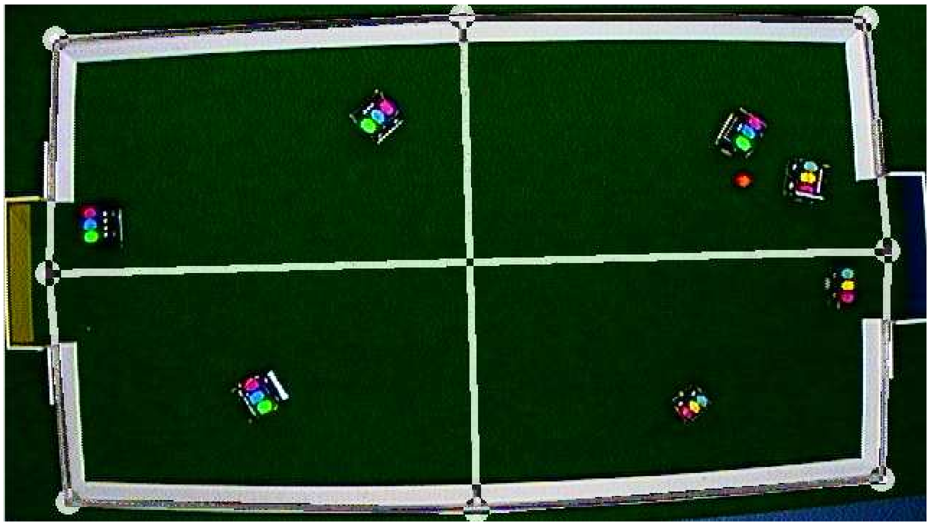

Fig. 4. Eight points parameterize the non-linear coordinate transformation.

We avoid calling the costly global search routine in every frame, since this would reduce the frame rate and would therefore increase the risk of loosing further objects. It would also add overhead to the reaction time of the overall system. Fortunately, the global search is needed mostly when the game is interrupted, e.g. for a kickoff. Then, people moving robots by hand cause occlusions that trigger the global search. To avoid this interference, the FU-Fighters robots position themselves for kickoff.

During regular play, the processor load caused by the 30Hz computer vision is as low as 30% on a Pentium-II-300, leaving enough time for behavior control and communication that run on the same machine.

3 Coordinate Transformation

All tracking is done in pixel coordinates, but the behavior control needs the positions of the objects in a standard coordinate system. The origin of that system is located in the middle of the field and its length corresponds to the interval $[-1, 1]$. The width of the standard system is chosen such that the aspect ratio of the field is preserved.

A linear coordinate transformation is not sufficient, since the camera produces distortions. We model them for the four quadrants of the field using eight positions located at the fields corners and in the middle of the walls (see Figure 4). The central point is computed from the intersection of the two lines connecting the wall centers that become the axis of the standard system.

In each quadrant, we perform a bilinear interpolation between its corners. We also account for the different heights of the objects by multiplying the trans-

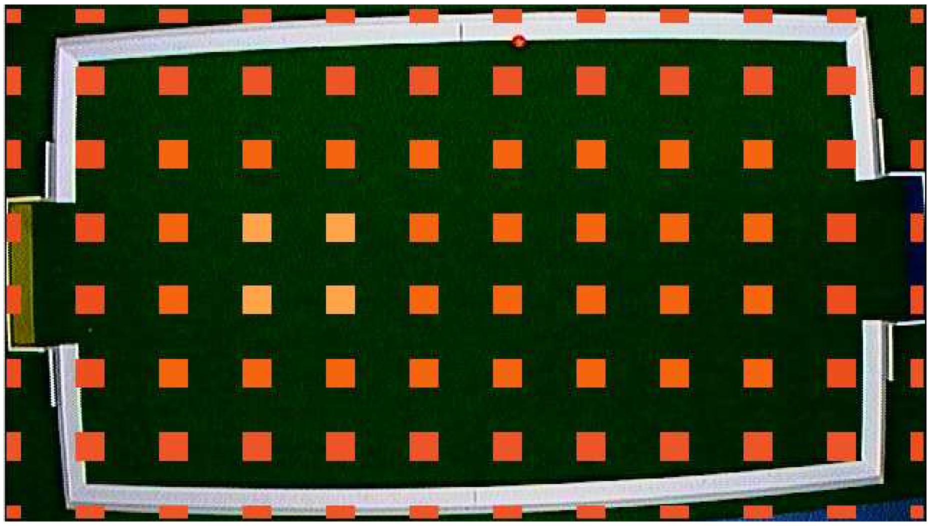

Fig. 5. Color map for the orange ball with a resolution of 12×8. The color is darker near the walls and brighter in the center. A hot spot is visible in the middle of the left half.

formed coordinates with a suitable factor. The described transformation models the non-linearities of the camera sufficiently.

4 Improvements

4.1 Color Maps

When adapting the color of individual dots, one assumes that the appearance of a color only depends on the dot itself. This is not correct, since its appearance also depends on the possibly inhomogeneous lighting. Therefore, modeling the inhomogeneities should improve the tracking system.

The idea is to maintain for each color a map that models its appearance in dependence of the location on the field. We assume that the lighting changes slowly with position and use a low-resolution grid (e.g. 12×8).

We initialize these maps uniformly and edit it at extreme positions, e.g. by clicking at a colored marker when it is in a hot spot or in a shadowed region. A map for the orange ball might look like the one shown in Figure 5 after initialization.

When the tracking system is running, we adapt the maps automatically every time objects have been found. This approach can cope with slow global changes of the lighting as well as local inhomogeneities that vary slowly in space and time.

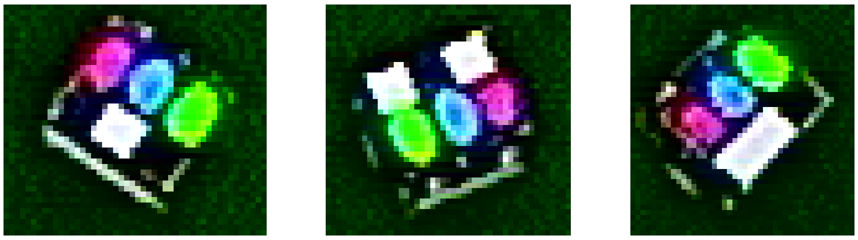

Fig. 6. Robot identification. Three different binary codes are shown, represented by the black and white markers located next to the colored markers.

4.2 Robot Identification

When localizing our robots they have to be mapped to the behavior processes that control the individual robots. Our first approach was to use identical robot markers and to maintain the mapping by initializing it manually and tracking the robots all the time. This works fine, as long as tracking looses less than one robot at a time. However, when robots are occluded or leave the field, the danger of exchanging two of them is high.

Such permutations cannot be corrected by the vision system described so far. Both robots get then the wrong visual feedback, which does not lead to useful behavior. To deal with this problem, we developed an automatic identification module.

Since the number of different colors is limited, we decided to use additional black and white markers, arranged in a binary pattern. Three markers are placed next to the three colored dots, as can be seen in Figure 6. To avoid uniform markers, we use only the six code words that contain at least one black and at least one white marker. The resulting code is similar to the one described in [4].

Identification is done only after successful localization. We compute the positions of the markers from the positions of the colored dots and estimate their intensity using a Gaussian filter that has approximately the size of the markers.

Then, we order the three filter answers. Since both black and white markers are present, the brightest response must be white and the darkest one must be black. We only have to decide if the remaining marker is black or white. This can be done by comparing it to the average of the extreme responses.

Due to interlace problems, we do identification only if robots don't move too fast. We also check the investigated regions for homogeneity, to make sure that we are looking at the inside of a marker. Identification works reliably even when lighting changes. It significantly reduces the time needed for manual initialization of the system.

5 Summary

This paper described the tracking of robots and the ball for our RoboCup Small-Size team. We implemented several mechanism that make the system robust and fast, such as dynamic search windows, global search, color maps and robot identification. One important aspect was the careful design of the robot's markers, such that no combination of the markers from two different robots can be seen as a robot and such that there is some redundancy, for the case that one marker cannot be localized.

The FU-Fighters used the described tracking system in Amsterdam, where we won the European Championship 2000 and the improved system during the RoboCup'2000 competition in Melbourne, where we finished second, next to Big Red from Cornell University. The system delivered reliable information even in situations where other teams failed to localize the ball or the robots, e.g. when the ball was inside the wall's shadow or the referee was projecting a shadow on the field.

In the future, we plan to use a digital camera that can be connected to the PC via an IEEE-1394 link. This camera produces an uncompressed YUV 4:2:2 image with 640×480 pixels at a rate of 30Hz. The digital link, as well as the progressive scan should increase the image quality and therefore simplify interpretation.

We are also investigating the possibility of adding an omni-directional camera to the robots for local vision. The image will initially be transferred via a wireless analog video link to an external PC, where it can be analyzed. We plan to apply the tracking principle to edges, e.g. the ones between the green field and the white walls to localize the robots. If this can be done with a low computational load, it would be possible to implement the local computer vision on a small, low power on-board computer, such as a PDA.

References

1. Peter Ackers, Sven Behnke, Bernhard Frötschl, Wolf Lindstrot, Manuel de Melo, Raul Rojas, Andreas Schebesch, Mark Simon, Martin Sprengel, and Oliver Tenchio. The soul of a new machine. Technical Report B-12/99, Freie Universität Berlin, 1999.
2. Sven Behnke, Bernhard Frötschl, Raul Rojas, Peter Ackers, Wolf Lindstrot, Manuel de Melo, Mark Preier, Andreas Schebesch, Mark Simon, Martin Sprengel, and Oliver Tenchio. Using hierarchical dynamical systems to control reactive bahaviors. In *Proceedings IJCAI'99 - International Joint Conference on Artificial Intelligence, The Third International Workshop on RoboCup – Stockholm*, pages 28–33, 1999.
3. S. Cordadeschi, editor. *RoboCup' 99 Small-Size Team Descriptions.* http://www.ida.liu.se/ext/robocup/small/intro.
4. Paulo Costa, Paulo Marques, Antonio Moreira, Armando Sousa, and Pedro Costa. Tracking and identifying in real time the robots of a F-180 team. In *Proceedings IJCAI'99 - International Joint Conference on Artificial Intelligence, The Third International Workshop on RoboCup – Stockholm*, 1999.
5. A.K. Jain. *Fundamentals of Digital Image Processing.* Prentice-Hall, 1989.
6. The RoboCup Federation. *RoboCup Regulations & Rules.* http://www.robocup.org.