# Visualizing Robot Behaviors as Automated Video Annotations: A Case Study in Robot Soccer

Danny Zhu[1] Manuela Veloso[1]

***Abstract*— Autonomous mobile robots continuously perceive the world, plan or replan to achieve objectives, and execute the selected actions. Videos of autonomous robots are often naturally used to aid in replaying and demonstrating robot performance. However, plain videos contain no information about the ongoing internals of the robots. In this work, we contribute an approach to automate the overlay of visual annotations on videos of robots' execution to capture information underlying their reasoning. We concretely focus our presentation on the complex robot soccer domain, where the high speed of the robots' execution results from action planning for collaboration and response to the adversary.**

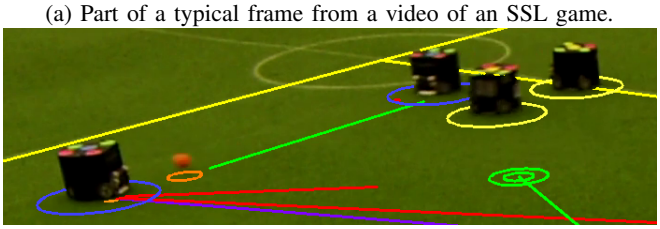
## I. Introduction

Consider watching a video of robot soccer in the RoboCup Small Size League (SSL), as depicted in Figure 1. The game is fast and the ball is small, so it is hard to tell what is happening for anyone does not already know the game very well. It can be hard even to tell which robots are on each team, as shown by the blue and yellow dots on top. Even if we do know the robots well, there is still a lot that we cannot deduce by just looking at the robots moving; the ball and the teams are still hard to follow. Videos are informative, because they capture a recording of what actually happened in the world, but there is much information about the robots' behaviors that is not captured by the video.

In this work, the robots are necessarily controlled by some computer algorithm; we contribute a process for extracting information about their operation while they run and storing it for display on a video. It would not be possible to do something similar with a video of humans, whose internal "algorithms" are inaccessible to the outside world.

We propose and discuss a means of organizing the information generated by an autonomous robot in a manner suitable for translation to visualizations. Overall, our main goal is to combine real systems of mobile autonomous robots with the ability to visualize reasoning algorithms, so that we can capture and reveal what the algorithms of autonomous robots are doing. We wish to display the behavior of complex autonomous agents in tandem with the real world, enhancing what were originally initially plain, uninformative videos, and we do so by developing the ability to draw extra algorithmic information on videos, as shown in Figure 1. Much of the internal state of the robots involves physical locations on the ground, e.g., where the ball is, where to pass to, and where the opponents are. By making the video look as if all of that information had actually been displayed on the ground around the robots, we can more clearly show what happened and why the robots did what they did.

(a) Part of a typical frame from a video of an SSL game.



(b) The same frame as above, with visual annotations overlaid.

Fig. 1: An example of an image from an SSL game, first without and then with extra annotations.

Since an algorithm controlling robots may process many different pieces of information per second, it is also important to be able to reduce what such visualizations display in a way that follows the robot algorithm. Accordingly, we introduce the application of multiple levels of detail to robot visualizations. We describe a method of organizing the information created by a run of a robot algorithm into a form that mirrors the structure of the algorithm. By organizing the information into a graph structure, we make it easy to select and filter information that relates to individual components.

In this paper, we demonstrate an application of these ideas using a RoboCup SSL team. However, the same concepts can be extended to work with other autonomous mobile robots.

## II. Related Work

Our work consists of two main parts: organizing the information generated by an autonomous robot as it executes tasks and projecting that information onto a video of the robot performing those tasks. In this paper, we discuss the first aspect and provide a summary of the second. We concretely ground our visualizations using the RoboCup SSL [10].

Batory and O'Malley [2] previously examined the organization of software into separate components that relate to each other in a graph structure, similar to how we organize the information generated by an autonomous agent. They provide a general definition of software components and the

[1] Danny Zhu and Manuela Veloso are with the Computer Science Department, Carnegie Mellon University, Pittsburgh, Pennsylvania 15213, USA dannyz@cs.cmu.edu, mmv@cs.cmu.edu

interfaces between them. Collberg et al. [4] take a related approach to the organization of software, using inheritance, program flow, and call graphs, and develop visualizations for depicting the change of software over time. Compared to this earlier work, we incorporate the spatial aspects of autonomous robots into the system, allowing us to perform an augmented reality-style visualization. We also take the conceptual structure of the program and, rather than visualize it directly, use it to organize a separate set of visualizations generated during the execution of the program.

Augmented reality systems are displays that show the real world itself along with additional information on top. Collett [5] developed a system to depict robots' sensor information in an augmented reality display. We use similar ideas of video manipulation, but additionally focus on higher-level aspects of robotic planning, A popular commercial product related to augmented reality appears in television broadcasts of American football games ([1], [7]). Many such broadcasts add a virtual down line or other information for the benefit of viewers; the displays move and are occluded as if they were present on the playing field, much like the work presented here. Such displays are mature and have high precision in positioning and occlusion, but they require pan/tilt sensors on all cameras and a crew of humans in order to keep track of changes in lighting conditions.

## III. Planning in the RoboCup Small Size League

The SSL is a RoboCup league in which teams of six robots play soccer using a golf ball in a $9\,\mathrm{m} \times 6\,\mathrm{m}$ field. Each team has an offboard computer that controls all of that team's robots over radio. Overhead cameras provide localization; SSL-Vision [13] processes the camera input to detect the robots and ball. As compared to other RoboCup leagues, the high-fidelity global vision and centralized planning allow teams to focus on coordination and high-level teamwork, as opposed to perception, locomotion, or distributed planning. RoboCup SSL games are fast-moving, with the robots traveling at up to $3\,\mathrm{m/s}$ and the ball at up to $8\,\mathrm{m/s}$.

An example of a specific planning algorithm used by our SSL team is selectively reactive coordination (SRC) [8], which is our method of determining how to position attacking robots so as to maximize the probability of scoring goals.

In brief, the steps of the SRC algorithm are as follows:

1) Compute zones, which are subsets of the field.
2) Assign the zones to the available attacking robots.
3) For each attacking robot, compute an individual action within its assigned zone.

The algorithm includes multiple steps in sequence, where details of the positions and regions involved in each stage are not deducible from the final behavior of the robots.

At any given moment in time, the SRC algorithm uses a set of zones $\{z_i\}$, which are rectangular subsets of the field space. Sets of zones are chosen offline and manually; the algorithm chooses a set of zones at each timestep. If there are $n$ available attacking robots, the algorithm designates $n - 1$ as support attackers and assigns them to zones; the remaining robot is the primary attacker, which has no zone; instead, it attempts to manipulate the ball. To assign the primary attacker and the support attacker zones, the algorithm computes a cost for each robot and each role, then finds the best overall assignment of robots to roles.

Next, each support attacker chooses a pass reception locations within its zone. Each location is the point within the zone leading to the highest probability of successfully making a pass. The computation of probability depends on several factors, such as the distance from each point to the opponent goal. Regardless of the point chosen, each support attacker moves to a predefined default location within its zone; they wait there until the algorithm chooses a pass receiver, which moves to its chosen location.

Finally, the team executes the pass using a pass-ahead algorithm [3], which determines the timing of the receiver's movement relative to that of the robot with the ball in order to receive the pass while leaving as little as time as possible for the opponent team to intercept the ball.

## IV. Visualizations of Planning

We introduce the idea of augmenting a robot planning algorithm to generate visualizations describing aspects of its planning. While running, the robot records information relating to the planning; each time the control algorithm executes, the generated visualization information is gathered together for real-time or offline display. In our system, that information is then drawn onto a video, as shown in Figure 1b. We discuss visualizations that consist of geometric shapes, which can be drawn onto the video directly; it is also possible to use symbolic information, which would then have to be transformed into drawings in a further step. For the SRC algorithm, we choose the following pieces of information to visualize:

- the zones
- the default location within each zone
- the location chosen by each support attacker
- which robot is the primary attacker

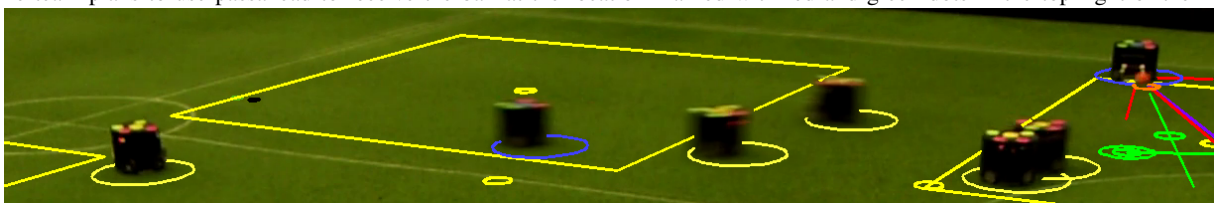
For a planning algorithm such as SRC that runs on real robots, showing aspects of the plan that will take time to execute on top of the current state of the robots blends the future and present aspects of the planning and execution in a way that is impossible to achieve with a plain video alone.

Figure 2 shows a particularly visible example of this correspondence across time, as related to SRC. The first image shows the primary attacker planning to pass to a location in the upper right of the image; the second image shows a point in time a few seconds later, after the intended receiver has received the ball at the planned location.

### A. Information hierarchy

We adapt the idea of layered disclosure [9], which applies to textual logging information, and only with a single dimension of filtering. With layered disclosure, lines of textual information generated by an autonomous agent are given numerical levels corresponding to the level of detail that they represent. For example, levels 1–10 are for high-level goals, while levels 41–50 are for sensory perceptions.

(a) The team plans to use passahead to receive the ball at the location marked with red and green dots in the top right of the image.



(b) A teammate robot has received the ball at the planned location and is about to shoot it into the goal.

Fig. 2: Visualizations showing the planning and subsequent execution of a pass between two robots.

We extend the idea of a single dimension of detail to a hierarchical system of logging information. The structure of the information hierarchy should mirror the layout of the algorithm in some sense. The idea of software as a graph of components has been explored, though the means of creating the abstract structure in our case may be different from that in the other work. We assume generally that we have a directed acyclic graph (DAG) describing the reasoning structure, where each node has some set of visualization information associated with it. This DAG might be generated based on a similar notion of components within the reasoning algorithm of the agent as described above; alternatively, the nodes could correspond to particular pieces of information and the information dependencies between them.

When working with visualizations, having a principled means of filtering information is even more important than with text logging; spatial visualizations may overlap with each other, causing them to interfere in a way that is not present with text. Our solution is to use the conceptual organization of the agent to filter the set of visualizations shown at the same time.

### B. Geometric primitive mapping

At any time during execution, an agent's reasoning algorithm may generate visualizations based on its state at that point. The algorithm applies some means of mapping an aspect of its reasoning into geometric primitives, each of which is one of a few simple shapes: circle, line, rectangle, or ellipse. Each instance of a geometric primitive has a color and the appropriate geometric parameters.The parameters of a primitive are some function of values in the reasoning, such as local variables at the point in the algorithm generating the primitive. The mapping may be chosen by anyone involved with the robots, perhaps (but not necessarily) a developer.

A typical use of the geometric primitives for robot soccer is to display positions on the field derived from calculations within the soccer algorithm. For example, we might show a good location at which to receive a pass by drawing a circle around it, or show where a robot should go by using line segments to draw an arrow from its current location to that location. For the SRC algorithm in particular, we define the following mappings from information to geometric primitives; Figure 2 shows examples:

- the zones map to yellow rectangles
- the default locations map to yellow circles
- the computed locations to receive passes map to small red circles (top right of Figure 2a)
- the locations to which support attackers should drive to receive passes map to small green circles (top right of Figure 2a)

Figure 2b also demonstrates lines depicting parts of the calculations for making the subsequent shot on the goal.

We also use circles of a fixed size to represent the robots; doing so is appropriate here, since all of the robots are of essentially identical size and shape. If the robots were not all the same, we could easily use different shapes for each one, given a suitable shape for each type of robot.

## V. SSL Logging and Drawing

In this section, we describe the specifics of the SSL platform on which we have implemented our visualizations, including the format of log file available and the current interfaces available for viewing log files.

### A. Log description

The control algorithm of the team runs a complete iteration 60 times per second; each iteration produces a heterogeneous collection of information about the reasoning it performs. The information produced by each iteration comprises:

- raw positions as received from SSL-Vision [13]
  - for each robot: position and angle
  - for the ball: position
- positions and velocities predicted by Kalman filter [11]
  - for each object: position, angle, and velocity
- the commands sent to each robot by our soccer team

  - for each robot: velocity, angular speed, kick state
- a hierarchically-organized textual log [9]
  - lines of text, each indented corresponding to its depth; we form a tree where the parent of each line is the closest previous line with a smaller indent
- a list of drawing primitives with parameters
  - list of circles, lines, rectangles, and ellipses

We have developed a graphical interface to display the information from a log and enable us to interpret the reasoning behind the SSL robot behaviors, shown in Figure 3. The interface combines all aspects of the log into an integrated display and is essential to our efforts to improve the intelligence of our robots.

### B. Additional drawings created by the viewer itself

The viewer application, in addition to depicting the pre-generated drawings, computes additional information to draw based on the information in the log. Currently, that information comprises (a) object trails, (b) navigation targets, (c) open angles to goals, and (d) navigation obstacles. The viewer draws these pieces of information itself, visually distinct from the drawings from the soccer team. These pieces of information could be pre-generated as well, but, for practical reasons, they are treated separately. These extra pieces of information can be toggled on and off independently, so they present a simple form of information selection.

One reason that not everything can be computed by the viewer is randomness; each random choice would need to be explicitly recorded and reused. There is also the matter of amount of computation. The information drawn by the viewer comprises object positions and the results of short computations using information in the log. That information would be redundant to include in the log, so we allow it to be recomputed. On the other hand, reproducing everything would involve rerunning the entire soccer program, which would be impractical. The set of information redrawn reflects a balance between these two extremes.

### C. Information selection

Our robot agents are complex and produce information at a high rate. Thus, it is necessary to devise means of reducing what is displayed. For the textual information, we already use an extension of layered disclosure. We produce a tree-shaped hierarchy of information at different levels, but instead of having only a single dimension for filtering, we allow for selection of subtrees which is persistent across frames.

For the drawing information, we create a hierarchy that mirrors the text hierarchy. When we collapse a node in the hierarchy of the text log, all the drawings created while the execution of the program was within that node are hidden as well. Figure 4 shows examples of one scene drawn at different levels of detail, with offense- and defense-related drawings selected independently.

## VI. Constructing a video

We can take the graphical portion of the viewer application and insert it into the video as the natural means of combining the drawn visualizations with real video of the robot actions. Doing so involves taking the instances of drawing primitives from a log and drawing them most informatively onto a video. For the SSL, we choose to place the drawings in the video so that they appear in the appropriate places on the ground and are occluded by the objects on the ground.

### A. Drawing with different levels of transformation

An important kind of information that we display consists of the positions of the robots and ball on the field at the present time in the video. A basic means of depicting this information would be to show, for each object, some identifying string (e.g., the ID number for robots) printed directly onto the video at the location of the object. We could instead draw, e.g., a circle in the image centered at that point. In the complete version, we instead draw a circle that is transformed to the correct perspective and masked so that it appears to be present on the ground in the frame of the camera and occluded by real objects on the field. Compared to the simpler approaches, doing so has the advantage of providing a more natural appearance for the drawings and avoiding drawing on top of parts of the video that contain the objects we are interested in watching. These different means of distinguishing robots are shown in Figure 8.

### B. Projection and masking

Here, we summarize the process of transforming the drawing primitive instances into drawings on top of a video. The process is described in more detail in [12]. The drawing process for each frame consists of two main steps: projecting the drawings from the coordinates used by the robots into the frame of the video and computing which pixels to alter.

The projection step consists of computing and applying the homography to map the coordinates of points from world space into image space [6]. The field coordinates have their origin at the center of the field, with the $x$-axis along the line between the goals; the image coordinates are determined by the position of the camera. A homography is represented by a $3 \times 3$ matrix. Given a homography $H$ and the world coordinates $\begin{pmatrix} x & y \end{pmatrix}^T$, if $H \begin{pmatrix} x & y & 1 \end{pmatrix}^T = \begin{pmatrix} x' & y' & z' \end{pmatrix}^T$ then the corresponding image coordinates are $\begin{pmatrix} \frac{x'}{z'} & \frac{y'}{z'} \end{pmatrix}^T$. A homography can be constructed from four or more correspondences (points with known world and image coordinates); currently, we manually specify correspondences for each video to process. There are well-known algorithms [6] for robustly computing a homography from correspondences.

Because a homography does not preserve distances, it can, e.g., map circles to ellipses. We approximate circles and ellipses as polygons with a large number of vertices, which means that this warping happens automatically. Meanwhile, since homographies do preserve lines, we need only project the endpoints of a line and the vertices of a rectangle, and then draw straight lines between them in the image.

The other step is deciding where in the image to draw on. We wish to make the drawings appear present on the ground, so we find which pixels show the ground. Since our

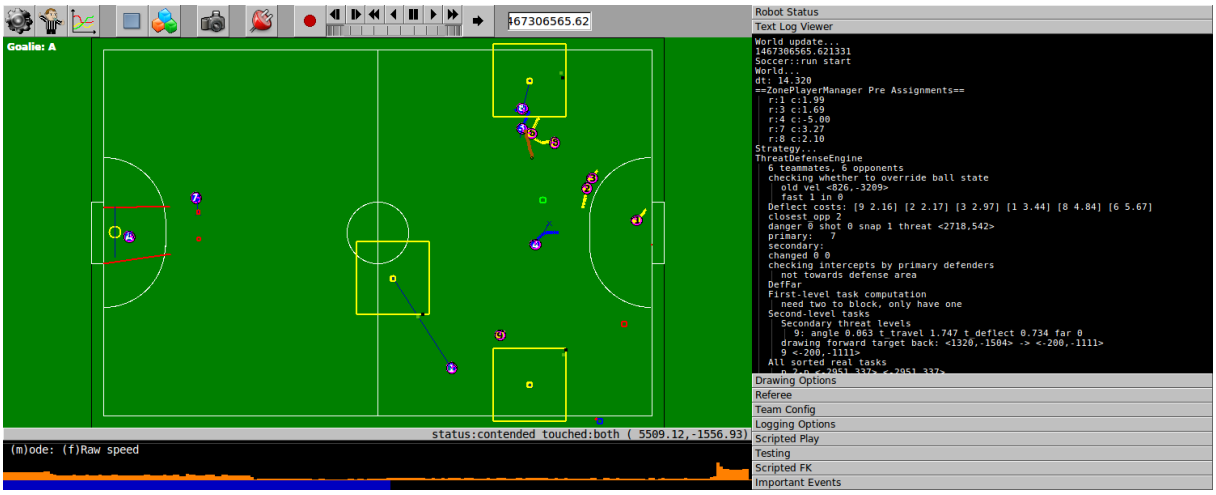
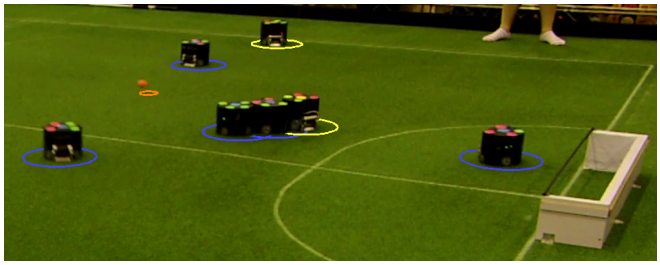
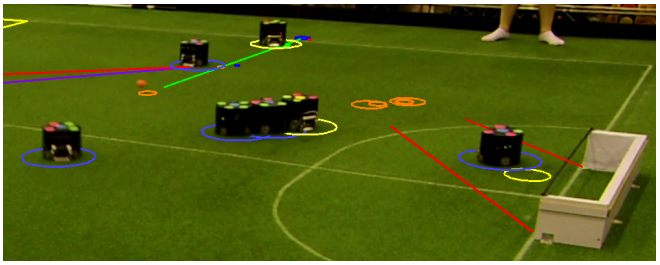


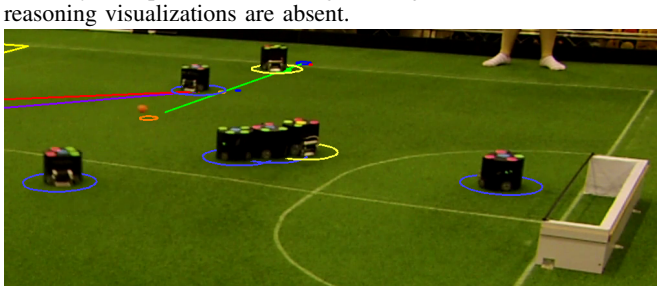

Fig. 3: An example of our viewer's display, showing the textual log information on the right. The graphical display on the left shows the robots, labeled with their unique team color and ID number, and additional drawings generated by our team.



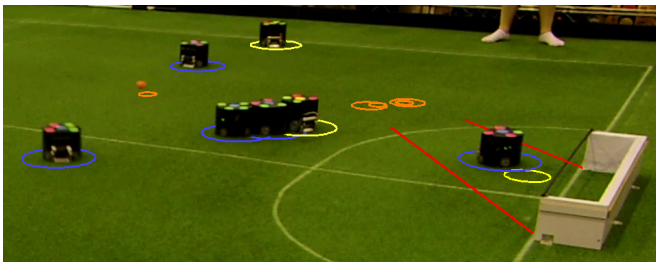
(a) Only the positions of the game objects are annotated; the reasoning visualizations are absent.



(b) Both defense- and offense-related annotations are included.



(c) In addition to the object positions, offense-related annotations are drawn. The ball is about to be kicked by the blue robot near it; the annotations include lines showing the projected path of the ball (green), the orientation for the robot to be facing during the kick (purple), and the predicted path of the ball afterward (red).



(d) In addition to the object positions, defense-related annotations are drawn. The drawings include the extents of the open angles to the goal from the position of the opponent best able to receive the ball (red) and desired positions for extra defenders (orange).

Fig. 4: An example of a single image annotated with different levels of detail.

applications have taken place on green SSL fields, we have been able to use a mostly pixelwise process to perform this masking step. Primarily, we use two thresholding steps in HSV color space to find the green of the field and the white of the field line markings, combined with morphological operations to smooth edges and fill in gaps.

### C. Drawing accuracy and limitations

So far, we have used a homography to map between the video image and the world coordinates. For a homography to be completely accurate, the video must have been taken using a camera and lens with no distortion (i.e., they map straight lines to straight lines). While some classes of lenses are prone to distortion (e.g., wide-angle lenses) or intentionally distorted (e.g., fisheye lenses), distortion is generally not desirable; the videos we have produced, on a consumer video camera, have no noticeable distortion, as is typical for modern products. In any case, it is straightforward to compute and reverse the distortion of any camera setup, given the ability to take new images with it [6], so we do not consider camera distortion to be a major issue.

Also, our current implementation requires a fixed camera for each video, since it assumes a constant homography

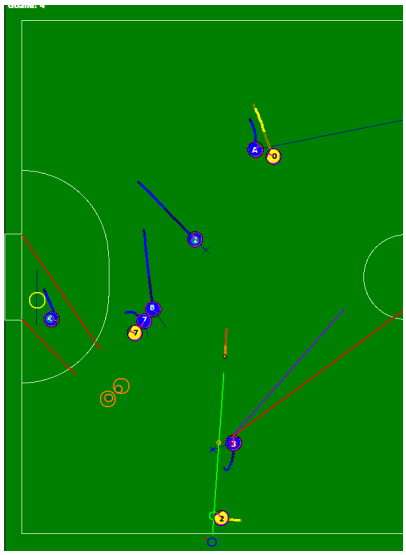


Fig. 5: The graphical portion of the display shown by our viewer application for the same frame as shown in Figure 4.

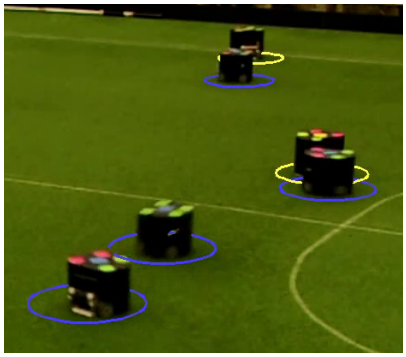


Fig. 6: An example of robots at different distances from the camera properly generating circles of different sizes according to the camera perspective.

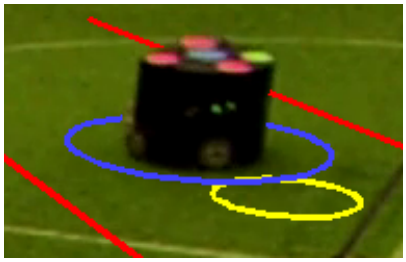


Fig. 7: An example of the masking process allowing a robot to appear to cover a line drawn on the field.

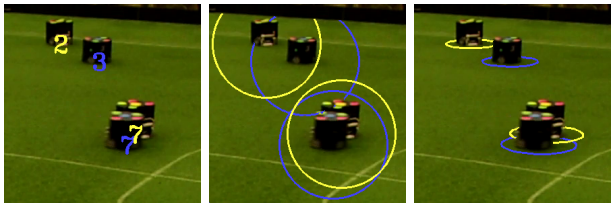


Fig. 8: Possible ways to mark robots in an annotated video.

across all frames of a video, which allows us to specify the coordinates for a video with a single set of correspondences. It is possible, however, to analyze the video and track points on the field over time, which would allow us to construct a moving homography over time. The rest of the drawing process could then proceed exactly as before, using the varying homography.

## VII. Conclusion

We have described a method for organizing visualization information generated by the execution of an autonomous mobile robot, as well as results of projecting that information onto uninformative videos of robot execution. We grounded the ideas of the method in a concrete implementation involving a team of the SSL, but the ideas used, both for organization and for visualization, are generalizable to other mobile robot algorithms.

## Acknowledgments

This work is partially supported by AFRL and DARPA under agreement #FA8750-16-2-0042 and NSF grant IIS1012733. The views and conclusions contained in this document are those solely of the authors.

## References

[1] 1st and Ten Graphics, 2014. https://www.sportvision.com/football/1st-ten%C2%AE-graphics.

[2] Don Batory and Sean O'Malley. The design and implementation of hierarchical software systems with reusable components. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 1(4):355–398, 1992.

[3] Joydeep Biswas, Juan Pablo Mendoza, Danny Zhu, Benjamin Choi, Steven Klee, and Manuela Veloso. Opponent-driven planning and execution for pass, attack, and defense in a multi-robot soccer team. In *Proceedings of the 2014 International Conference on Autonomous Agents and Multi-Agent Systems*, pages 493–500. International Foundation for Autonomous Agents and Multiagent Systems, 2014.

[4] Christian Collberg, Stephen Kobourov, Jasvir Nagra, Jacob Pitts, and Kevin Wampler. A system for graph-based visualization of the evolution of software. In *Proceedings of the 2003 ACM symposium on Software Visualization*, pages 77–ff. ACM, 2003.

[5] Toby H. J. Collett. *Augmented Reality Visualisation for Mobile Robot Developers*. PhD thesis, University of Auckland, 2007.

[6] David A. Forsyth and Jean Ponce. *Computer Vision: A Modern Approach*. Prentice-Hall Englewood Cliffs, 2003.

[7] Stanley K. Honey, Richard H. Cavallaro, Jerry N. Gepner, Edward G. Goren, and David B. Hill. Method and apparatus for enhancing the broadcast of a live event. US Patent number 5917553 A.

[8] Juan Pablo Mendoza, Joydeep Biswas, Philip Cooksey, Richard Wang, Steven Klee, Danny Zhu, and Manuela Veloso. Selectively reactive coordination for a team of robot soccer champions. *Proceedings of AAAI-16 (2016, to appear)*, 2016.

[9] Patrick Riley, Peter Stone, and Manuela Veloso. Layered disclosure: Revealing agents' internals. In *Intelligent Agents VII: Agent Theories Architectures and Languages*, pages 61–72. Springer, 2001.

[10] RoboCup Federation. Small Size League official homepage. http://wiki.robocup.org/Small_Size_League.

[11] Manuela Veloso, Joydeep Biswas, Philip Cooksey, Steven Klee, Juan Pablo Mendoza, Richard Wang, and Danny Zhu. CMDragons 2015 extended team description, 2015.

[12] Danny Zhu and Manuela Veloso. Virtually adapted reality and algorithm visualization for autonomous robots. In *RoboCup 2016: Robot World Cup XX*. Springer, 2017.

[13] Stefan Zickler, Tim Laue, Oliver Birbach, Mahisorn Wongphati, and Manuela Veloso. SSL-Vision: The shared vision system for the RoboCup Small Size League. In *RoboCup 2009: Robot Soccer World Cup XIII*, pages 425–436. Springer, 2009.