Automated Creation of Augmented Reality Visualizations for Autonomous Robot Systems

Danny Zhu

May 4, 2016

Thesis Proposal

Contents

Abstract

We believe that it is essential to be able to understand the reasoning of autonomous robots as it relates to their behavior, and to be able to convey this reasoning to others. Videos of robots are often naturally used to aid in understanding robot performance; plain videos contain no information about the processing or behavior of the robots, but such videos can be combined with extra information to improve their usefulness. Overall, my goal is to combine real systems of mobile robots with tools for understanding algorithms, so that the behavior of complex autonomous agents can be made more understandable. Concretely, my thesis will investigate the addition of visualizations onto initially plain, uninformative videos.

The work I have done so far relating to the issues of analyzing and understanding robot behavior has primarily progressed in two directions. The first is the development of an automatic referee (autoref) for the RoboCup SSL, a robot soccer league. This work involves creating a formal model of the rules and progression of the game as a hybrid automaton, along with a modular implementation designed to resemble the structure of the model and of the rules themselves. The individual components cooperate to produce the overall flow of the game. The autoref is an autonomous agent that observes the behavior of other agents, makes inferences based on that behavior according to a given set of rules, and communicates with the other agents to alter their behavior in turn. Understanding the decisions of the autoref is of great interest.

I have also developed mixed reality visualizations for autonomous mobile robots. For both the robot soccer team and a quadrotor navigation problem, I have produced animations that combine a video taken of the real robot, executing some plan or sequence of plans, with visualizations revealing the details of the plan appropriately geometrically registered to the video. The aim of such visualizations is to present the progress of the robots' planning algorithms so as to improve human understanding of their behavior.

The use of augmented and mixed reality has been explored in robotics before. Others have created augmented reality displays intended to aid the operator of a robot in the face of latency or poorly positioned camera views, as well as visualizations similar to my own for displaying data collected by robots as appropriately-registered overlays on a video of the robot. The focus of this work was on showing the low-level sensor data and using the visualizations to aid humans in performing tasks using the robots; I intend to move on to elucidating higher-level aspects of the planning done by autonomous robots.

For my thesis, I propose to further generalize the visualizations I have created; I would like to allow dynamically changing what objects are visualized for a given plan, creating the ability to zoom into specific regions of the space, show multiple levels of detail of the plan, or filter the visualizations corresponding to different portions of the plan, according to a user's choices. Doing so will extend the idea of layered disclosure for text, which our team already uses heavily, to these visualizations. Meanwhile, the autoref currently lacks any visualization capabilities, or indeed any user interface at all. However, its modular design will aid in adding future visualization, and applying ideas of visualization from the other domains to its processing of game rules will be important for its development; if such an autoref is to be practical to use, it must be easy to understand by spectators and players, as requested. I will evaluate the effectiveness of different visualization choices I will develop within the robot soccer and autoref domains.

Chapter 1

Introduction

1.1 Motivation

Imagine watching a game of robot soccer in the RoboCup Small Size League (SSL), as depicted in Figure 1.1. The game that the robots play is fast, and the ball is small, so it is hard to tell what is happening if one 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. That is true while watching robots in person, but it gets worse when watching a video of the robots that has been captured by some camera somewhere. Then the ball can really get lost in the video and the teams may be even harder to follow. Videos are still very informative, because they are the only way to capture a recording of what actually happened in the world. But would it not be even better if the videos had some extra information about the underlying algorithms that determine the robots' behaviors to help us follow what happened?

That is where this thesis comes in. We propose to develop the ability to draw extra information on the videos, as shown in Figure 1.2, to capture and reveal what the algorithms of autonomous robots are doing. 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 make it easier to tell what happened and why the robots did what they did. In this thesis, we will break down the interesting information from a robot system into three categories, consisting of the information related to (a) the current state of the world, (b) the future plans of the robots, and (c) the reasons that the robots chose to make a particular decision in the past. Because the robots are controlled by a computer program, it is possible to record exactly what their algorithms are doing at every moment. The methods of this thesis will then combine that information with a video by drawing informative objects on it, such as circles to mark locations on the ground or lines to describe a region or path; we keep the original view of the robots from the video, and then add drawings.

Robot soccer teams nowadays are controlled by complicated programs, and must consider a lot of information in order to make sure to be playing competitively. For instance, one common architecture that many teams use is Skills, Tactics, and Plays (STP) [17]. Skills decide the exact low-level commands for one robot to do in order to do one small, specific thing; tactics put together multiple skills and connecting logic to make a robot fill in a

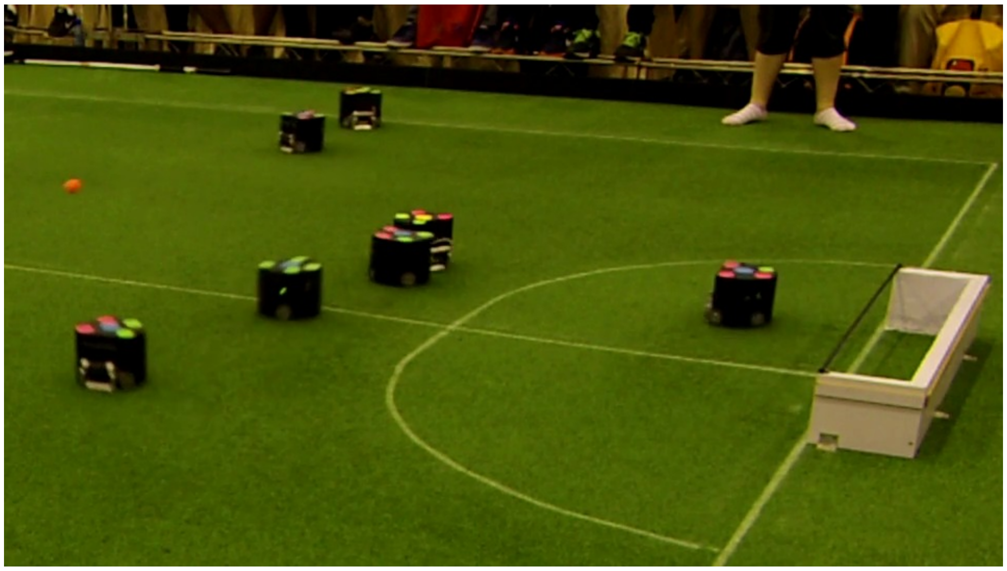


Figure 1.1: A typical frame from a video of an SSL game.

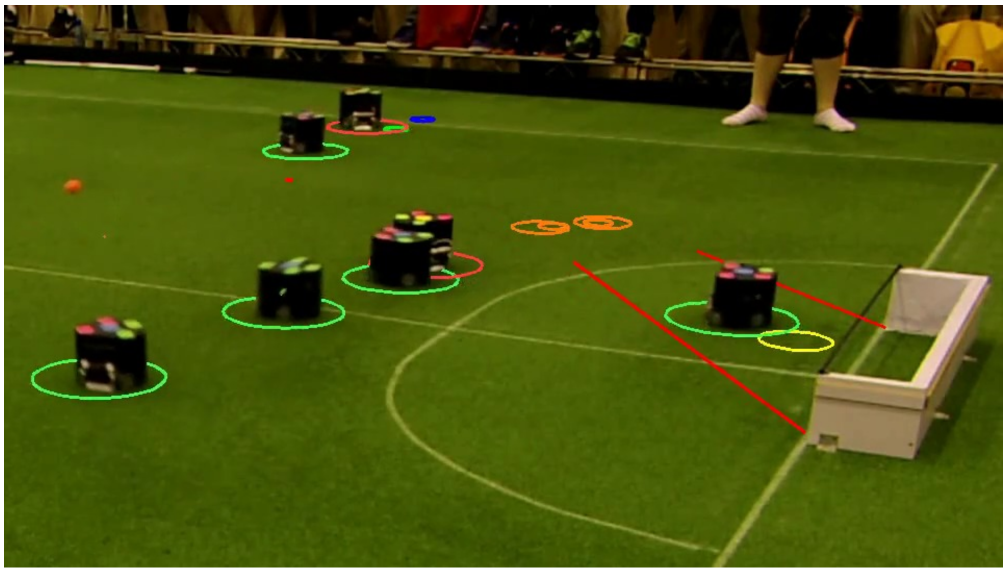


Figure 1.2: The same frame shown in Figure 1.1, with additional informative annotations drawn on top of it.

higher-level role for the team; finally, plays describe how to assign tactics to each robot on the team. Every skill, tactic, and play that is active at a given moment has to make decisions constantly in order to figure out the right action to take. Altogether, there is a lot of information that the system generates that might need to be conveyed later on; however, not all of it will necessarily need to be displayed all the time. In this thesis, we will produce frameworks that enable the creation of flexible visualizations that make it possible to filter the set of displayed visualizations in a sensible, principled way, and have more or less information visible according to what one might need at any given moment. In order to produce these flexible visualizations, we need to store information during robot execution and be able to interpret it later. In general, this thesis will take the execution of a complex robotic system and encode the decision and history information in log files that are later combined with videos.

To be clear, the intended focus of this thesis will not be on performing user studies to determine the choices that lead to the best understanding in any particular situation; rather, we intend to provide the ability to make such decisions easily in general situations.

Now we do need to address the reason that something new like what we have described could be used for robotic applications. After all, although there are robots involved, there is ultimately a computer executing a program to control them. We, and other robot developers, do in fact use print statements and other text-based displays to an extent, but one immediate problem is that there can be a lot of information to display, so you would need a lot of print statements. Besides, the robots really do change what we need; we can't simply ignore the fact that these programs have physical components in the real world. Any method that draws on a screen in isolation, even if it includes graphical elements, as in Figure 1.3, is separated from the robots themselves, and that gap makes gets in the way of its ability to convey information. Each part of the program execution is tied to a particular state of the robot in the world, and enabling the display of the direct physical connection with that state

In summary, this thesis will address the following question:

> Given a video of autonomous robots, how can information about the decisions and actions of the robots be extracted from a representation of their reasoning and flexibly projected onto the video?

1.2 Approach

In this thesis, we will explore several aspects of the problem of generating informative visualizations from records of robot execution. We will aim to create videos that not only show directly what the robots are doing, but also depict inferences about the actions and the reasoning behind them.

It will be necessary to devise ways of representing and presenting the stored data in a way that allows for flexibility in display and usage.

Augmented reality visualizations The visualizations we propose to automatically create are properly registered with the locations of objects in the world and are subject to occlusion by objects in the scene. In order to create such visualizations, we must therefore

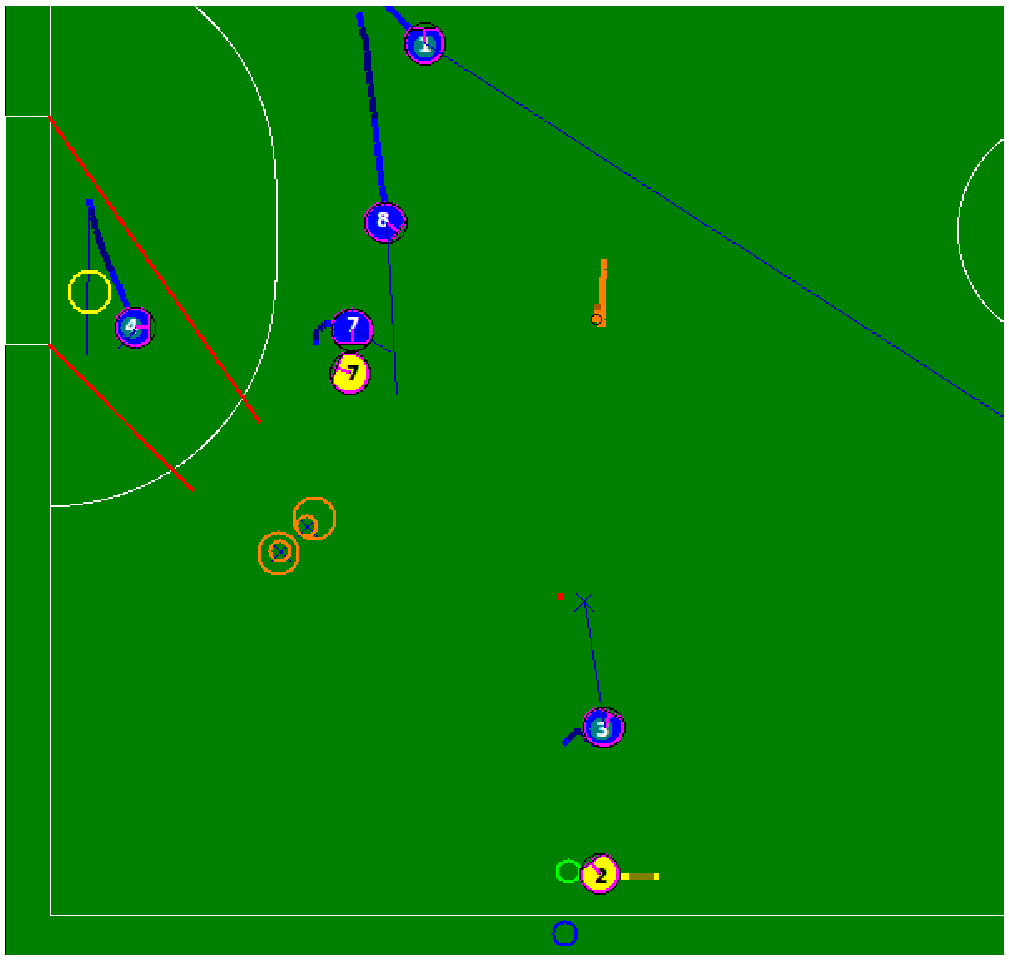


Figure 1.3: The two-dimensional view generated by our existing viewer software for the same moment depicted in Figure 1.1.

have three main pieces of information: the mapping from spatial coordinates in the world to image coordinates in the video, enough understanding of the structure of the scene to handle the occlusion, and additionally the mapping between the times recorded in the execution logs and time in the video, so that all events are synchronized. For the cases we have examined already, we have been able to generate this information manually or with straightforward means; we will continue to study computing them in general for the proposed applications.

For the case of planar drawings (where the plane is usually the floor), the coordinate mapping is given by a homography [25]. In general, a homography is determined by four or more point or line correspondences between two coordinate systems; so far, we have mostly considered cases of a static camera, where the homography remains constant throughout the video; for such videos, we supply the correspondences manually. In general, if the camera moves during the video, we will need to track how the coordinate mapping changes over time, and if we use visualizations that are three-dimensional, we will need to know more about the camera parameters than a homography captures. There exist methods to address the problem for various circumstances; we propose to devise a method that incorporates the particular knowledge available in this domain, i.e., the positions of the moving objects in the scene, in order to automatically produce the coordinate mapping as a function of time.

The occlusion information is also straightforward to determine in some cases, but not so in general. For the situations we have implemented so far, we have only needed to detect the pixels of a distinctively-colored floor; we have used color thresholding followed by morphological operations to do so. For other cases, such as if we use three-dimensional visualizations, we will need more detailed information. Since producing this kind of full understanding of a scene in general is itself a major topic of computer vision, we do not expect to make advancements in the state of the art for addressing the entire problem; however, we propose to incorporate and adapt relevant algorithms for our videos. For our specific problem of realistic visualization, we will also need to make decisions about how to handle the case of a real object blocking the view of an added drawing.

Finally, we need the mapping from the times recorded in the logs of execution (usually given in some absolute scale, such as seconds since the Unix epoch) and time in the video (which starts at 0); this mapping is almost always an additive offset. Currently, we determine the offset manually for each visualization, but the automatic trajectory matching that we propose will compute the time offset as well.

We propose to categorize the displayable information for any given robotic system into three groups, based on their relation to the current time. One category contains information directly related to the *present*: tracking or sensor information that makes up the current estimate of the state of the world. Then there is the information related to the *future*: the plans of the robots that have yet to be executed. Finally, there is the information from the *past*, showing events from previous windows of time that have some effect on the decisions in the present.

Multiple levels of detail Given the large amount of information that we might want to display for a run of a robot system, we must be careful not to overload someone looking at the system by displaying too much information.

The concept of *layered disclosure* [47] addresses this kind of problem in text-based logging systems. It describes tagging each individual piece of information within a log with a “layer”, which is a numeric value that indicates the abstraction level to which that information belongs (e.g., overall, abstract goals and immediate sensory perceptions at immediate ends of the spectrum, with intermediate, short-term goals in the middle), with selective display and the elements placed into a meaningful hierarchy. We propose to apply a generalization of layered disclosure to the graphical elements in our visualizations, allowing for the selection of not only linearly-ordered levels of importance, but also grouping by other criteria, such as location within the program or relevance to a high-level goal of the robotic system.

Implementation and evaluation on real robot systems In order to demonstrate the feasibility of the algorithms and systems we develop, we will create and test implementations relating to the RoboCup SSL, on both the team itself and an automatic referee system we have worked on. The SSL robots in particular already store and use a rich set of graphical visualization primitives, but their display is simplistic and lacking in the extra capabilities detailed above.

We propose to not only create our own implementations, but also to enable others to create similar videos for any other robot system. To this end, we will develop concrete algorithms that can be used in existing projects to allow the development and use of visualizations with a minimum of additional effort.

1.3 Expected contributions

I expect to make the following contributions with my thesis:

1. A unified representation and interface for robot decision and position information.
2. Algorithms to process videos and use the proposed data representations to overlay graphical depictions of robot planning.
3. User interfaces that expose the ability to easily navigate the data representations and show multiple levels of detail.
4. Demonstrations of augmented reality visualizations using the RoboCup SSL robots, autonomous referee, and other autonomous systems.
5. Software that enables the creation and use of the proposed visualizations with other robotic systems.

Chapter 2

Related work

This chapter presents an overview of related work. In Section 2.1, we discuss the basics of the RoboCup SSL, as well as work related to our development of an automatic referee for SSL games. In Section 2.2 and Section 2.3, we discuss the fields of visualization and mixed reality.

2.1 RoboCup SSL

The RoboCup SSL promotes research in multi-agent coordination in real-world adversarial domains. In this league, teams of six robots play a scaled-down version of FIFA soccer with a golf ball in a field of size $9\,\text{m} \times 6\,\text{m}$. Each robot is controlled via radio commands sent by its team's offboard computer. Four overhead cameras observe the field and detect the positions of the robots and ball with high frequency and fidelity; as compared to other RoboCup leagues, the global vision and centralized planning mean that teams focus on hardware development, coordination behaviors, and high-level teamwork strategy, as opposed to perception, locomotion, or methods of distributed planning. Games in the SSL are fast-paced, with the robots traveling up to 3 m/s and kicking the ball at speeds of up to 8 m/s.

The overarching target of the RoboCup Federation, which includes several other soccer leagues, based on widely varying robot types, as well as some non-soccer competitions, is [51]

> By the middle of the 21st century, a team of fully autonomous humanoid robot soccer players shall win a soccer game, complying with the official rules of FIFA, against the winner of the most recent World Cup.

The various leagues are intended to collectively spur development in the different areas of development that will ultimately be necessary to produce such a team. The humanoid soccer leagues, for example, focus on bipedal walking and ball manipulation; the SSL focuses on centralized planning and fast reactions in a dynamic environment; the MSL, which uses roughly human-sized wheeled robots and a normal soccer ball, focuses on decentralized planning and sensing without dealing with the difficulties of legged robots; finally, the simulation soccer leagues (2D and 3D) use simplified models of the world to allow teams to focus more on strategy and artificial intelligence without having to maintain hardware.

2.1.1 Physical components

Although there are few restrictions on the physical design of SSL player robots, the primary one being that each robot must fit within a cylinder of diameter 180 mm and height 150 mm,

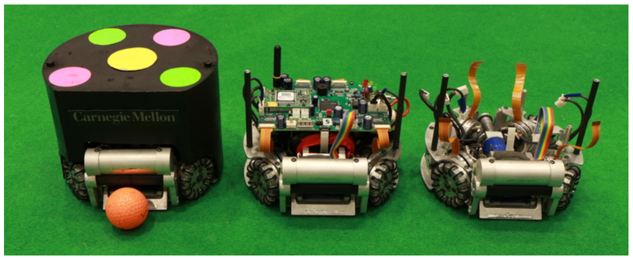


Figure 2.1: The robots of CMDragons, showing (left) a full robot with a ball on its dribbler, (middle) a robot without its top cover, and (right) without its main electronics board.

most teams have converged toward very similar designs; Figure 2.1 demonstrates a typical design[1]. The robot has an omnidirectional drive system based on four omniwheels. For ball manipulation, it has a dribbler, a horizontal rubber-coated bar which rotates to put backspin on the ball, allowing the robot to hold the ball for some time while driving around. Finally, the robot has both a flat kicker, which kicks the ball straight forward at high speed, and a chip kicker, which kicks the ball at an angle of about 45° above horizontal. Use of the chip kicker enables robots to pass or clear the ball even when there are opponents in front of them.

The ball is a typical golf ball, except that it is colored bright orange for ease of detection by the vision system. According to the rules, it should be approximately 43 mm in diameter and have a mass of approximately 46 g.

The overhead cameras are mounted on scaffolding constructed around and above the field. The league has standardized on FireWire 800 cameras. All of the cameras are connected to a neutral computer running the SSL Vision System (SSL-Vision) [71], which processes the images from the cameras to extract the positions of the robots and ball on the field. Each camera transmits sixty frames per second to SSL-Vision; each time a frame is received from any camera, SSL-Vision broadcasts the detected positions from that camera to the team computers.

The referee's decisions are transmitted to the teams via another computer running a specialized program, the *referee box* (or refbox). To ensure low latency and avoid bandwidth constraints between the team computers, referee computer, and vision computer, all of them are connected via a wired Ethernet network specific to the playing field.

2.1.2 RoboCup game analysis and refereeing

In addition to the development of the competitive RoboCup teams, there has also been some work on automatic analysis of games played in different RoboCup leagues, both online and offline. Examples of the goals of such work include classifying the roles of the robots in the game, following the ball, providing commentary on the progress of the game, and making decisions related to refereeing the game.

[1]We thank Mike Licitra for the mechanical design of the robots and for the design and fabrication of the robot electronics.

Vail et al. [63] investigated using conditional random fields for activity recognition of different robots based on a stream of their positions over time, much like the data available in the SSL. For automatic refereeing, it is also necessary to perform recognition of a sort: a referee's fundamental task is to interpret the situations described in the rules and determine when they are happening. However, most of the relevant situations for refereeing are amenable to simple handmade detectors. More advanced techniques for classification may be useful for more difficult situations, such as those that are based on attributing intent or negligence to the teams.

Many other authors have addressed the problem of online processing of RoboCup games, in pursuit of various aims. A simple example of such an aim is the automated cameraman for the RoboCup Middle Size League (MSL) described by RFC Stuttgart [33]. The cameraman shares information with a team to keep a camera pointed toward an object of interest at all times during the game. The object of interest is usually the ball, but may instead be a goal or goalie when a robot makes a shot on goal.

A more involved task is producing engaging, human-like commentary of games in real time. Rocco [67] and Mike [60], commentators for simulation games, use hierarchical declarative event systems to produce discrete events which may be the topic of utterances. CM-Cast [64] is a commentary system for soccer games played by Sony AIBO robots; it is embodied in two humanoid robots which can autonomously move around the field, localize themselves, track the ball, detect game events, and give commentary appropriately. All of these commentators revolve around a concept of discrete "events" derived from the continuous state of the game, similar to the event detectors we describe later. Commentating requires more understanding of the specific game under observation than does acting as a cameraman, but less than full refereeing; it shares the problems of event detection with refereeing, but the focus of such systems is typically more on utterance selection and audience interaction.

There have also been other efforts to provide automated refereeing for RoboCup games. For many years, the simulation leagues have had official autorefs integrated with their simulation servers, which have grown to be encompass many rules at this point, although games still require human referees for the more subjective events. In both the 2D [21] and 3D [57] simulation leagues, the autorefs can handle

- managing the length of game halves,
- resetting robots to legal locations when the game is stopped,
- detecting when a free kick is taken,
- awarding the proper kicks when the ball leaves the field or enters a goal, and
- detecting offsides situations.

Additionally, the 2D referee can detect backpasses, but relies on a human referee to detect players surrounding the ball or a team blocking its own goal with too many players. The 3D referee can detect when the players have formed a cluster around the ball that should be broken up and when a team's goal is being blocked by too many of that team's robots; in either case, it teleports the offending robots outside the field.

The autoref we describe, when running with a simulation of real robots, is similar to these referees in scope. However, referees that are designed purely for simulation and execute within the simulator itself have the advantage of being able to rely upon having full, direct access to the state of the world, in terms of both reading (i.e., determining the current positions of world objects) and writing (i.e., instantaneously resetting the robots or ball to desired positions). As a result, detection of many events is much less error-prone for such referees.

For real robots, Arenas et al. [6] developed a refereeing robot that watched games of the RoboCup 2-Legged Standard Platform League and the Humanoid League. By necessity, that work mainly focused on the lower-level aspects of event detection from vision, since those leagues do not have the standard centralized vision system of the SSL, but the ultimate aim is equivalent to ours for those leagues. For the MSL, Tech United Eindhoven [54] briefly described an autoref that can detect when the ball leaves the field, automatically signal the robots to stop play, and wait for a human to position the ball in the correct location. Their algorithm is like a simplified version of the one described here. The RoboCup Logistics League (RCLL) has also recently gained an autoref [43], which was initially developed at the same time as the work described here. The kinds of events that need to be detected for that league are rather different, but the ultimate goals described there are much the same: taking a difficult job off human referees, ensuring objectivity in judging, and benchmarking the systems used in the game. The RCLL autoref communicates with team robots and neutral machines over the network, awarding points when robots complete particular tasks using the game pieces, which abstractly represent materials in a factory. The communication the autoref receives consists of discrete messages, primarily about completed tasks, so that it does not need to perform its own processing of real-world data to extract relevant events, as ours does. However, the rule-based system at the core of the RCLL autoref achieves a similar purpose to the structures described here, and could be used as the basis of as alternate implementation.

Finally, at least two existing projects have worked toward automatic refereeing for the SSL specifically. At RoboCup 2014, the SSL released the technical challenge of detecting certain relevant events from the game, such as robots colliding with excessive force or kicking the ball too fast [50]. ER-Force, the SSL team of University of Erlangen-Nuremberg, has published the code it developed for the challenge [52]. Also, the open-source project `ssl-autonomous-refbox` [58] uses Prolog to declaratively define the rules as predicates over game states. The ER-Force code can only handle detecting those isolated events, while `ssl-autonomous-refbox` is limited to observing a game and lacks the ability to handle the transitions related to restarting the game after a stop; our autoref provides a more complete solution than either of them by being able to handle a full game from start to stop, transmitting the appropriate commands to teams.

2.2 Visualization

Visualization, in the most general sense, refers to any means of using graphics or images to convey information. Visualizations have been made on an ad hoc basis for a long time, but organized study of the principles behind it has become much more common in recent times.

A seminal modern figure in visualization is Edward Tufte; his book *The Visual Display of Quantitative Information* [62] provides a history of visualization, as well as extensive guidelines on designing useful and comprehensible illustrations of all kinds of data. Tufte codified and demonstrated general principles that are widely applicable to both print- and computer-based visualizations, such as removing extraneous material on the page and ensuring that the variation in a visual display has only as many dimensions as the data do.

There are several categories of science-related visualization; the most relevant to our work are scientific visualization and information visualization. Information visualization involves displaying abstract data that do not have an inherent spatial representation that can be mapped directly onto a display [26]. Scientific visualization involves displaying data that do have some inherent spatial relations [26]; the data are often measurements of some physical process.

Much work has also focused on visualization as applied to computer programs. This work is generally referred to as *software visualization*, of which *algorithm animation* is a part. Software visualization refers to any method of visualizing aspects of computer programs, such as the relationships between components, the text of the code itself, or a program's behavior over time [10, 46].

Animation has long been an invaluable tool for understanding algorithms. In algorithm animation, the changes over time in the state of a program are transformed into a sequence of explanatory graphics. Done well, such a presentation can explain the behavior of an algorithm much more quickly than words. BALSA [16] is an influential early framework for algorithm animation. It introduced the idea of *interesting events*, which arise from the observation that not every single low-level operation in an algorithm should necessarily be visualized. With BALSA, an algorithm designer inserts calls to special subroutines inside the algorithm; when execution of the algorithm reaches the calls, an interesting event is logged, along with the parameters to the call. The designer then writes a separate renderer that processes the log of interesting events into an actual animation to display.

More recently, there have been several websites that collect examples of algorithm animation and provide tools for creating new animations [2, 3, 66]. They follow BALSA's paradigm of interesting events and provide easy-to-use environments for implementing and displaying algorithms.

We are performing a form of algorithm animation here, but with a slightly different focus: most animations are designed with education in mind and delve into the details of an abstract algorithm, rather than executing on a physical robot. Additionally, the typical lifecycle of an algorithm on a mobile autonomous robot is different from the standalone algorithms that are usually animated. In most animations, a single run of the algorithm from start to finish results in an extended animation, and each interesting event corresponds to some interval of time within it. For an autonomous robot, algorithms are instead often run multiple times per second, with each run contributing a tiny amount to the behavior of the robot, and it makes sense for each frame of an animation to depict all the events from one full run. An example of this kind of animation comes from the visual and text logging systems employed by teams in the RoboCup Small Size League, a robot soccer league [35]. The algorithms behind the teams in the league consist of many cooperating components, each with its own attendant state and computations running 60 times per second. As a result, understanding the reasons behind any action that the team takes can be challenging, necessitating the development of powerful debugging tools in order for a team to be effective.

For text-based log data, Riley et al. [47] developed the idea of "layered disclosure," which involves structuring output from an autonomous control algorithm into conceptual layers, which correspond to high- and low-level details of the agent's state. Teams have also developed graphical logging in tandem with the text output: the algorithms can output lists of objects to draw, and we can view both this drawing output and text output either in real time or in a replay fashion. The existence of these tools speaks to the need for informative debugging and visualization when dealing with autonomous agents.

2.3 Mixed reality

Mixed reality is a general term originally defined by Milgram and Kishino [41] to encompass any system that "involves the merging of real and virtual worlds". The term was introduced to extend and generalize the idea of *augmented reality*, which was rapidly coming into use to describe new systems at the time. Milgram defined a reality-virtuality continuum; fully virtual and fully real environments exist at opposite ends of the spectrum, with combinations in the middle. Augmented reality systems are those near the reality end; they are typically displays, often but not necessarily head-mounted [40], that show the real world itself (either directly or through a video) along with additional information on top.

2.3.1 Augmented reality

Augmented reality has seen a wide variety of uses, both in relation to robotics and otherwise; it allows a user's view of reality to be enhanced without being entirely replaced. Azuma [7] listed a variety of uses of augmented reality, including those for medicine, manufacturing, robotics, entertainment, and aircraft control. One common use of augmented reality within robotics is to provide enhanced interfaces for teleoperated robots, such as interfaces for controlling a robotic arm that overlay 3-D graphics onto a video view of the arm [34, 42]. Such interfaces can a view of the predicted trajectory of the arm after a sequence of not-yet-executed commands, allowing for much easier control, e.g., in the presence of signal delay between the operator and the arm.

Our past work shares the 3-D registration and drawing that are common to augmented reality systems, although most definitions of augmented reality require an interactive or at least live view, which we do not currently provide. Amstutz and Fagg [5] developed a protocol for communicating robot state in real time, along with a wearable augmented reality system that made use of it. They demonstrated an example application that overlaid information about nearby objects onto the user's view, one that showed the progress of a mobile robot navigating a maze, and one that showed the state of a robotic torso. Though they focused mainly on the protocol aspect, their work has similar motivations and uses to ours.

An instance of a popular commercial product related to augmented reality appears in television broadcasts of American football games [1, 29, 53]. Many such broadcasts add a virtual down line or other information for the benefit of viewers; the displays are designed to be located and occluded as if they were present on the playing field. Such displays are mature and capable of high precision in positioning and occlusion, but they require pan and tilt sensors to be attached to all cameras and assistance from a crew of several humans in order to keep track of changes in lighting conditions.

Other recent work has also involved using physical projections to aid in the understanding of the behavior of autonomous robots. For instance, Chadalavada et al. [20] demonstrated a robot that signaled its intentions by projecting its planned future path on the ground in front of it. They found that humans in the robot's vicinity were able to plan smoother trajectories with the extra information, and gave the robot much higher ratings in attributes such as predictability and reliability, verifying that describing an agent's internal state is valuable for interacting with humans.

One prior publication deserves particular note here: the doctoral thesis of Collett [22] described an augmented reality visualization system which shares many characteristics with we focus on combining the drawing with the ability to display and animate details such as the future plan of the robot, rather than only the low-level sensor and state information from the robot.

2.3.2 Augmented virtuality

Toward the other end of the reality-virtuality continuum is *augmented virtuality*, a term also introduced by Milgram and Kishino [41], in which the primary environment is virtual; we also discuss here systems in which the real and virtual components are on roughly equal footing, which are often simply referred to as "mixed reality" in the literature.

Robert and Breazeal [48] and Robert et al. [49] demonstrated two systems based on mixed reality: an educational robot that appears to children to disappear and reappear inside a screen, and a game where players control a physical robot that bounces a projected ball to a wall, where it appears within a projected world.

The concept of hardware-in-the-loop testing (HIL) shares clear similarities with augmented virtuality, though its origins are different. Hardware-in-the-loop testing involves emulating a complex mechanical system by constructing part of it and simulating the rest, with appropriate actuators and sensors to interface between the two portions [23]. With an appropriate design, HIL greatly reduces the expense of evaluating a new device by reducing the amount of hardware that must be built, shortening the time needed for development cycles, and enabling the reproducible generation of extreme situations that would be difficult to produce for an entire system. HIL is widely used in many disciplines, including automotive engineering and space robotics [23, 31, 59].

2.4 Camera pose tracking

The general problem of determining the viewpoint of a video stream as a function of time is of great interest in computer vision, and particularly so for augmented reality applications. Techniques to extract this information generally fall under the umbrella of *visual odometry* [44], which is widely used for localization by autonomous mobile robots; *match moving* [11] refers to a similar topic, though the name is usually used in the context of cinema special effects, and includes more manual, labor-intensive techniques.

Augmented reality applications often use fiducial tracking, which places designated objects in the environment to provide tracking of the pose of the camera with respect to the objects. Such objects may be active, such as patterns of LEDs [8, 9], or passive, such as the high-contrast printed patterns [28].

The alternative to fiducial tracking is natural feature tracking, which instead tracks the motion of arbitrary objects in the environment. Algorithms such as optical flow [30], point trackers [56, 61] and interest point matching [37] are basic building block algorithms that enable one to estimate correspondences between two images; applied to the successive frames of a video, these algorithms and variants of them form the basis of many tracking systems.

Others have also investigated tracking the motion of a camera specifically for sports videos, similar to the videos of the SSL field with which we have worked so far. Unlike the football broadcasts mentioned previously, these methods take as input only a video stream, with no additional sensor information about the movement of the camera; Hayet et al. [27] and Okuma et al. [45] developed algorithms to process videos of soccer and hockey games, respectively, and compute the homography between the playing surface and the video for each frame of the video. Both work by computing incremental homographies between successive frames of the input video by computing local matches between frames, followed by global matches to models of the markings on the playing surface.

Chapter 3

Previous work

Our previous work has largely focused on working with CMDragons, our RoboCup SSL team, in two main capacities. Section 3.1 describes our work on creating and maintaining the team's current defensive strategy, which has been used at several RoboCup tournaments. We have also developed an automated referee (autoref), discussed in Section 3.2, for the SSL, which is capable of interpreting gameplay according to the rules of the game, and enforcing the rules accordingly; the autoref represents a detailed and novel analysis of the robotic teams in a game. More directly in the direction of this thesis, we have also begun developing augmented reality visualizations of the type we discussed in Section 1.2. In order to reduce the scale of the problem to be studied from the complexity of the SSL, we began by examining a system that demonstrates the bare minimum of autonomy: the problem of a single quadrotor navigating around obstacles in a two-dimensional domain; Section 3.3 discusses this work. Our prior experience with real autonomous systems motivates our interest in creating means of analyzing and describing their behavior.

3.1 RoboCup SSL team

Our SSL team, CMDragons, builds upon extensive research from previous years in areas including computer vision, path and dribbling planning, execution monitoring, and team architecture [13, 17–19, 39, 70]. This legacy enabled our 2015 team to focus on the problem of coordinated decision-making for a team in a highly dynamic adversarial domain. Furthermore, while previous years have seen a focus on stopped-ball plays [12], recently our team has focused its research efforts on regular gameplay team planning. As a result, our offensive strategy relied heavily on teamwork, with 245 pass attempts and 194 completions over the tournament.

To achieve such coordination, the team algorithms are divided into various layers:

1. The top team layer, consisting of a play system [14] that, based on functions of the state of the game, partitions the robots into a defense subteam and an offense subteam, and provides the robots in each team with further specialized roles.
2. Each of these two subteams creates coordinated plans to maximize the team's probability of scoring and not being scored upon, respectively.
3. Each robot individually selects actions that are consistent with the coordinated plan and maximize the probability of success.

4. The robots execute these actions through new reusable skills [12].

3.1.1 Threat-based defense

Our personal focus has been on the defensive capabilities of the team; we have been in charge of developing algorithms for the defense subteam for several years. The defense relies on a threat-based evaluator [38], which computes the *first-level threat* T_1 and several *second-level threats* T_2^i on our goal: T_1 is the location from which the opponent can most immediately shoot on our goal — i.e., the location of the ball, if it is stationary, or the location of the robot that will receive the ball, if the ball is moving. Threats T_2^i are locations of opponent robots that might receive a pass from the T_1 robot, and then shoot on our goal.

The available defenders are positioned based on the locations of the threats. *Primary defenders* (PDs), of which there are usually one or two, move near the defense area, acting as the line of defense before the goalie; *secondary defenders* (SDs) move further out, intercepting passes and shots by the opponent earlier on. The PDs typically defend against the T_1, blocking open goal angles between the ball and the goal; if there are two PDs, but only one is needed to block a potential shot, the other moves elsewhere on the defense area to guard a T_2. (This condition can occur when the ball is close to one of the near corners of the field.) The SDs guard against the various T_2s; each one positions itself on a line either from a T_2 to the goal (to block a shot) or from the T_1 to a T_2 (to block a pass). There are never more than two PDs, and there are never any SDs unless there are two PDs. Figure 3.1 shows the T_1 and T_2^i, and the corresponding PD and SD assignments for a particular state of the world.

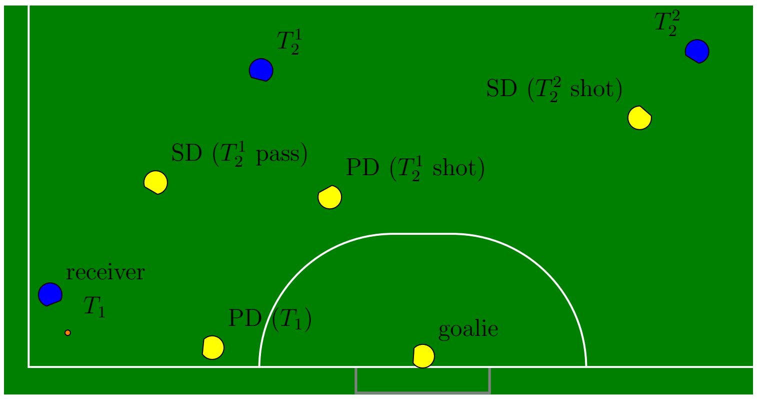


Figure 3.1: Example defense (yellow) evaluation and role assignment, as a function of the ball (small orange circle) and opponent (blue) locations. Parenthesized text indicates the type of task assigned to each defender.

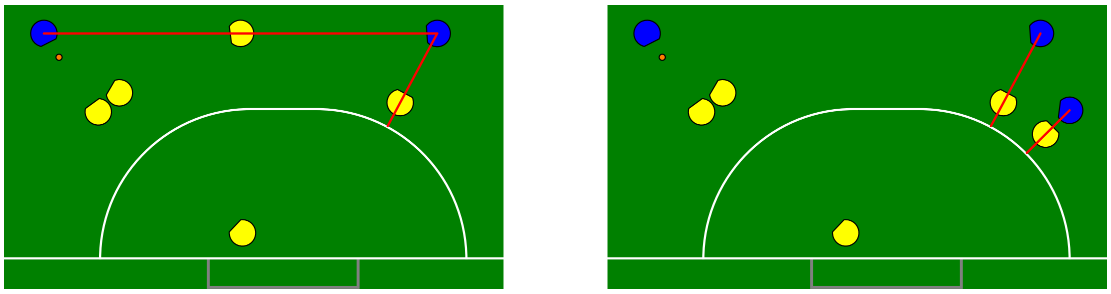
The aim of the goalie and the defenders guarding the T_1 is to entirely block the open angle from the threat to the goal; if this is not possible, they position approximately so that they block as much of it as possible.

Here, we have described the static positioning behavior of the defense, which would determine the behavior of the robots when the opponents and ball are static. In practice, we need additional behaviors in order to respond well and anticipate future conditions when the field is not static. Especially important is handling any moment when the ball is heading directly toward our goal or about to be received by an opponent. We have developed a complex and empirically tested set of behaviors to intelligently handle defense coordination.

3.1.2 Defense positioning illustration: 5 defense robots

We illustrate the defense's coordination behavior when the play system has chosen the most defensive play, which has 5 robots assigned to defense. In this case, one defense robot is the goalie, 2 robots are PDs, and 2 are SDs; we illustrate the case in which both PDs are needed to guard T_1. Thus, only the behavior of the SDs varies as a function of the opponent's offense.

Two opponents: Figure 3.2a shows the positioning of a 5-robot defense against 2 offense opponents. Since there is only one T_2, the two SDs block the shot from and the pass to that threat.



(a) 5 defenders vs. 2 opponents **(b)** 5 defenders vs. 3 opponents

Figure 3.2: Positioning of the defense in response to attacking opponents. Red lines indicate the geometrical relationships that define the target locations of the SDs.

Three or more opponents: Figure 3.2b shows the positioning of a 5-robot defense against 3 offense opponents. Our defense always ranks shot-blocking tasks as more valuable than pass-blocking tasks; thus, both SDs position to block shots from T_2^1 and T_2^2. If there are more than three opponents and hence more than two T_2^i, our algorithm ranks the T_2^i based on the size ϕ of their shooting angle on the goal, ignoring robots, and our SDs mark the highest-ranked threats. If multiple robots have an angle ϕ larger than a pre-defined threshold $\phi^{\max}$, they are ranked based on an estimate of the time t_{goal} it would take to complete a shot on our goal, where lower times are ranked higher. Time t_{goal} is calculated as $t_{\text{goal}} = t_{\text{pass}} + t_{\text{deflect}} + t_{\text{kick}}$, where

$$t_{\text{pass}} = \frac{d(\boldsymbol{x}_b, \vec{x}_T)}{v_{\text{pass}}}, \qquad t_{\text{deflect}} = t_c \cdot \begin{cases} 0, & \theta \le \theta_{\min} \\ 1, & \theta \ge \theta_{\max} \\ \frac{\theta - \theta_{\min}}{\theta_{\max} - \theta_{\min}}, & \text{else} \end{cases}, \qquad t_{\text{kick}} = \frac{d(\vec{x}_g, \vec{x}_T)}{v_{\text{kick}}}.$$

t_{pass} estimates the time to pass to the threat using the distance $d(\boldsymbol{x}_b, \vec{x}_T)$ between the ball location $\boldsymbol{x}_b$ and the threat location $\vec{x}_T$, and the estimated opponent passing speed v_{pass}.

estimates the time to deflect the ball to our goal based on the angle θ formed by $\boldsymbol{x}_b$, $\vec{x}_T$, and the goal center point $\vec{x}_g$: if a one-touch deflection is possible (i.e., $\theta \leq \theta_{\text{min}}$), then $t_{\text{deflect}} = 0$; otherwise, t_{deflect} increases with θ, as defined by constants t_c and θ_{max}. t_{kick} estimates the shot time similarly to t_{pass}, using the opponent's shooting speed v_{kick}.

3.2 Automated referee

In the leagues of RoboCup, the majority of the research effort to date has been on the competitive performance of the autonomous teams in aspects of motion planning and team strategy. However, another critical component of a soccer game is the referee. In current SSL games, refereeing is done by humans, who use a "referee box" that passes their calls to the robots. We have created an automated referee (autoref) [69] for SSL games, towards enabling games to proceed with little or no human supervision. The goal is to move closer to the eventual full automation of complete games with real robots. The technical challenges include the clear definition of the rules of the game in terms of features to be extracted from the visual perception, temporal sequencing, and corresponding calls and game management. We have developed a description of a game of the SSL as it is relevant to an autoref, by categorizing the rules of the game and presenting the structure of a game as a hybrid automaton. We then described the complete autoref as an agent using a modular event-based architecture, following up on the automaton as a guideline, to keep track of the state of a game and issue referee commands accordingly [68].

3.2.1 Game and autoref formalization

To concretize the structure of the game, we used the formalism of hybrid automata. We omit here a full description of hybrid automata in general; see Alur et al. [4] for the mathematical formalism we followed. For our purposes, we may think of a hybrid automaton as a finite state machine augmented with variables in addition to the overall state (although states are usually referred to as "locations" for hybrid automata). The additional variables may be discrete or continuous; each transition between states occurs when an associated predicate on the variables is satisfied. Within one location, the continuous variables change over time by following one of a set of trajectories associated with a state; the trajectories are typically described as the set of solutions to differential equations. When transitions occur, the variables may make discontinuous jumps to new values.

For the automaton describing an SSL game specifically, the locations, which are described in Table 3.1, capture the idea that, at any given moment in time, the game is either running or stopped, and each of those entails a distinct set of conditions that are relevant to refereeing. We depict the set of locations and transitions between them in Figure 3.3. When a transition occurs, a new command may be transmitted to the teams. When the game is in the GAME_ON location, the ball is in play and the robots are actively trying to score goals; during GAME_OFF, the ball is being placed for the next restart of play; during SETUP, the ball has been placed and the robots may set up for a kick. Each auxiliary variable describes some distinct part of the game state or history that may be relevant to future refereeing decisions. The variables include quantities such as the last robot to touch the ball, the com-

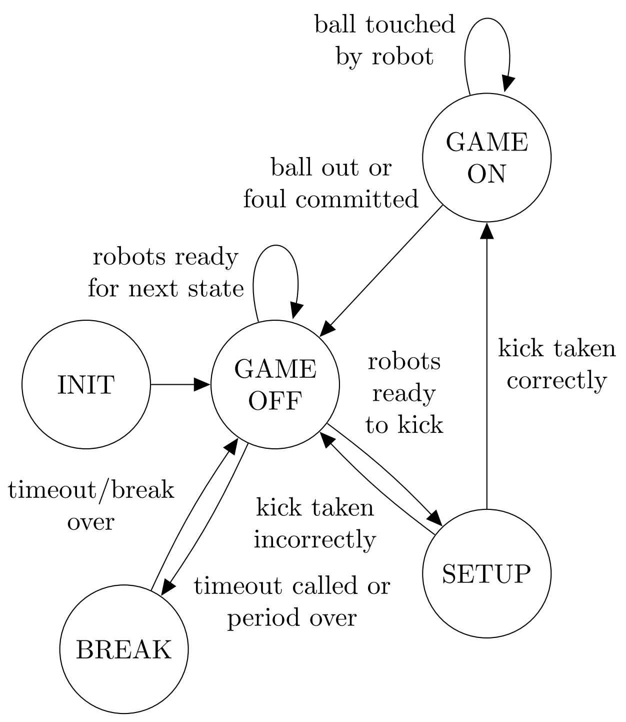


Figure 3.3: A simplified representation of the automaton representing a game of the SSL.

mand with which to resume the game after a stoppage, and the amount of time remaining in the current half.

We have described this automaton as describing the operation of an autoref, but there is a direct correspondence to the rulebook and to what human referees must do while they observe the game. They are also essentially modeling the same automaton: they must use their observations to estimate when the true state of the game ought to change, so that they can issue the appropriate commands.

3.2.2 Usage and evaluation

We implemented a program based on the formalism described above to actually act as a referee for games of the SSL; it approximately tracks the evolution of the automaton described above in terms of a set of separate *event detectors*, each of which receives the sequence of world states and outputs an object describing how the state of the automaton should be updated. The algorithm implements a discrete approximation of the true automaton, since it can only receive discrete updates of the world state.

In this kind of architecture, an event is any of the conditions associated with an edge in the automaton. The events we have currently implemented are enough to handle all phases of a normal game, including free kicks and half time, though only a subset of possible fouls are detected.

In order to test the abilities of the autoref, we performed two types of evaluation. First, to test the overall stability and rule adherence of a game as judged by the autoref, we set the autoref up to run repeated simulated games, with two copies of the CMDragons team playing, and collected statistics about the results of those games. Second, to test the ability of the individual event detectors to operate on real data, we ran some of the detectors on data from games of RoboCup 2014 and compared the results to the calls made by the human referees at those games.

Although not all of the rules of the SSL are currently implemented, the ones that are make up the great majority of rules that come into effect during games in practice. With the currently implemented events, it is possible to run an essentially full game in simulation, with kickoffs, appropriately awarded free kicks for ball exits and some fouls, and checks to ensure that the progress of the game is not delayed by a slow team or stuck ball. As a result, the autoref makes it feasible to run and gather information from a far greater number of rules-compliant games than would be feasible to monitor by hand.

As a simple demonstration of this capability, we ran many simulated games comparing two slightly different versions of our own CMDragons soccer team. In some tests, we enabled or disabled some new feature of the team, so that we could check whether having that feature gave the team an advantage. In others, one version of the team was handicapped, either by having fewer robots or by having the speed and acceleration of its robots restricted to a fraction of their normal values. Overall, we have run thousands of games, and can run well over one thousand games continuously without supervision. We found that, when one team is handicapped, the other has an noticeable advantage; while this is no surprise, the ability to verify this through experiment is made possible only by the presence of an autoref. The average scores as functions of the degree of handicap are shown in Figure 3.4; the error bars indicate the standard error of the mean after 100 games.

We also ran some tests to determine the effect of software features on performance. One of the features we tested was a zone-based attack strategy we developed and used in the RoboCup tournament last year; the other was a method for a robot to quickly turn while dribbling the ball, which we used for only part of RoboCup because it was unreliable on real robots [65]. The results are shown in Table 3.2; both features provided an advantage, as we hoped. However, this is one case where the limitations of the simulation environment must be kept in mind: the fast turning behavior worked very well in simulation, but we disabled it in reality because it did not work as well. The zone-based attack seems less likely to be dependent on the specifics of the simulation.

3.3 Augmented reality visualizations

We have already created some augmented reality visualizations, which we will be aiming to extend and generalize for this thesis. We have worked with two domains so far: a quadrotor navigation problem using real and virtual obstacles and the more complex setting of a RoboCup SSL team. In our work so far, the contents of the visualizations are a set of geometric drawing primitives, with positions given in the coordinates used by the control programs.

Examples of our visualizations can be found at `https://youtu.be/P4WgT8yOTGY` and `https://youtu.be/R3XqbiDzFoo`.

Location	Description
GAME_ON	The game is actively being played. This location typically ends when a robot commits a foul or the ball leaves the field.
GAME_OFF	The game has been stopped for one of the preceding reasons, and the robots must not come near the ball while the referee is positioning it. This location ends once the robots are ready to kick, which is usually taken to be when they stop moving or after some time has passed.
SETUP	The command to take a kick has been given, but the kick has not been taken yet. This location ends once the kick is taken, or too much time has passed without a kick.
BREAK	One team has called a timeout or the game is in a break between periods. This location only ends if the team ends the timeout or time in the break period runs out.

Table 3.1: The high-level automaton locations of an SSL game.

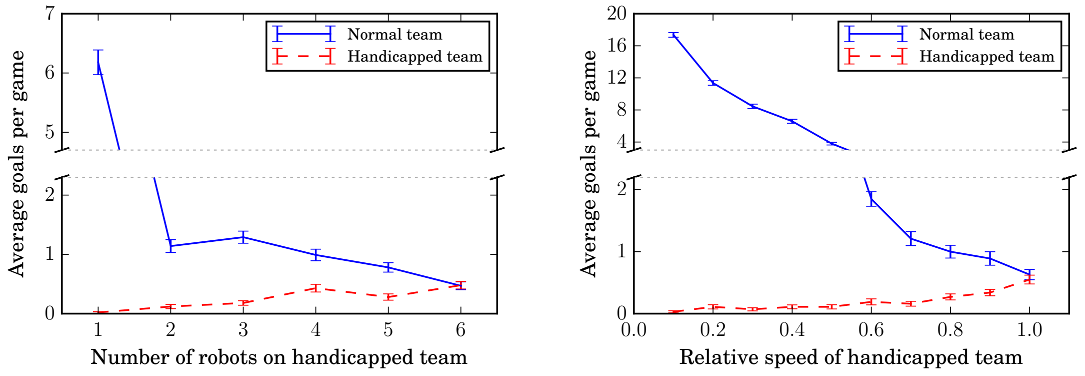


Figure 3.4: Comparisons of two versions of our team, one handicapped by having either fewer robots or reduced speed and acceleration. Each data point depicts the mean score and standard error of the mean of the two teams across 100 full games.

	Zone-based attack		Turn & dribble	
	Win %	Goals	Win %	Goals
feature on	38.00	0.58 ± 0.05	42.00	0.70 ± 0.05
feature off	12.67	0.18 ± 0.02	10.67	0.22 ± 0.03

Table 3.2: Comparisons of two versions of our team, differing by the presence or absence of a new feature. "Win %" indicates the percentage of games won by that team, and "Goals" indicates the mean number of goals per game, as well as the standard error of the mean. Each comparison is based on 300 games.

3.3.1 Quadrotor navigation domain

As an initial testbed for this visualization scheme, we implemented a control system for a quadrotor using the rapidly-exploring random tree (RRT) [36] algorithm; we then visualized information about the running of the algorithm.

The RRT algorithm finds a path from a start state to a goal state within some state space, parts of which are marked as obstacles and are impassable. The most basic form of the algorithm consists of iterating the following steps, where the set of known states initially contains only the given start state:

1. generate a random state r
2. find the closest known state c to r
3. extend c toward r, creating e
4. if e is directly reachable from c, add e to the set of known states, recording that c is its parent

The iteration terminates when a state is found sufficiently close to the goal state. The entire path is then constructed by following the chain of parents to the start state.

The simplest state space for an RRT consists of the configuration space of the robot, with dynamics ignored; states are connected simply by drawing lines in the space. Although this is a great simplification, it is straightforward in concept and implementation, and often leads to sufficiently good results.

Since the quadrotor can accelerate in any direction without yawing to face it, we chose to ignore orientation and take states to be the locations within a rectangular region on the ground. As is typical for a kinematic RRT, the random state generation simply chooses a location uniformly at random within the set of configurations and the metric is Euclidean distance between locations.

A common optimization is to simplify the returned path by skipping directly from the start state to the last state in the path which is directly reachable from it [18]. This causes the robot to navigate directly toward the first turn on the path; without it, the randomness of the RRT means that the first step in the path may take the robot in widely varying directions between timesteps. After each run of the RRT, the quadrotor attempts to fly toward the point resulting from this optimization.

For a robot running the RRT algorithm, each of the steps naturally relates to positions within the working area, which makes such a navigation problem a good fit for our augmented reality visualizations.

3.3.2 RoboCup SSL team domain

Since its inception, our RoboCup SSL team has developed a rich logging system that uses a combination of text-based [47] and graphical logging elements to help diagnose problems and understand the robots' behavior. The team uses the graphical elements to illustrate a variety of information about planning, such as desired positions of the robots, planned pass directions, and regions to guard. We took the existing graphical elements wholesale and displayed them all as the basis for a first version of a visualization. The drawing elements consist of lines, circles, and rectangles; each instance of one of these types has the necessary geometric parameters and a color associated with it.

3.3.3 Masking

In both domains, the ground surface was an SSL playing field, and we wanted to draw only on the field surface, not on top of the quadrotor or any obstacles on the field. In order to do so, we need to detect which pixels in the video are actually part of the field in each frame. Since the majority of the field is solid green, a simple chroma keying (masking based on the color of each pixel) mostly suffices. We convert the frame to the HSV color space [32], which separates hue information into a single channel, making it more robust to lighting intensity changes and well-suited for color masking of this sort. We simply took all the pixels with hue values within a certain fixed range to be green field pixels, providing an initial estimate of the mask.

To account for the field markings, which are not green, we applied morphological image transforms [55] to the mask of green pixels. The idea is that the markings form long, thin holes in the green mask, and there are generally no other such holes in our setup; therefore, a dilation followed by an erosion with the same structuring element (also known as a closing) fills in the field markings without covering any other pixels. We also applied a small erosion beforehand to remove false positives from the green chroma keying. The structuring elements for all the operations were chosen to be iterations of the cross-shaped 3×3 structuring element, with the number of iterations chosen by hand.

3.3.4 Coordinate transformation

Since we are only concerned with a plane in world space, we need a homography to give the desired coordinate transformation, if we assume an idealized pinhole camera [25]. A pinhole camera projects the point (x, y, z) in the camera's coordinate system to the image coordinates $\left(\frac{x}{z}, \frac{y}{z}\right)$, so the coordinates (u, v) are the image of any point with coordinates proportional to $(u, v, 1)$. Suppose that the ground coordinates $(0, 0)$ correspond to the coordinates $\vec{q}$ in the camera's coordinate system, and that $(1, 0)$ and $(0, 1)$ correspond to $\vec{p}_x$ and $\vec{p}_x$ respectively. Then, for any x and y, the ground coordinates (x, y) correspond to the camera coordinates

$$x\,\vec{p}_x + y\,\vec{p}_y + \vec{q} = \begin{pmatrix} \vec{p}_x & \vec{p}_y & \vec{q} \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}.$$

Thus, multiplying world coordinates by the matrix $\begin{pmatrix} \vec{p}_x & \vec{p}_y & \vec{q} \end{pmatrix}$ and dividing by the third coordinate of the result takes ground coordinates to the corresponding image coordinates; the resulting transformation is a homography. Homographies are typically computed from a set of points or lines with known coordinates in both the image and the world; there are well-known algorithms to perform this computation. We used the function for this purpose, `findHomography`, from the OpenCV library [15]; the function minimizes the backprojection error [25].

We primarily used a stationary camera; for such videos, we manually annotated several points based on one frame of the video and used the resulting homography throughout the video. We used intersections of the preexisting SSL field markings as easy-to-find points with known ground coordinates. We also implemented a line-tracking algorithm similar to the one by Hayet et al. [27], which allows some tracking of the field with a moving camera. Our algorithm takes the pixels which were detected to be field marking pixels and uses RANSAC [24] to fit a new position for each line.

3.3.5 Drawing

Finally, with the above information at hand, drawing the desired shapes is straightforward. To draw a polygon, we transform each vertex individually according to the homography, which gives the vertices of the polygon as it would appear to the camera. Since a homography maps straight lines to straight lines, the polygon with those vertices is in fact the image of the original polygon. Then we fill in the resulting distorted polygon on the video, only changing pixels that are part of the field mask. For the quadrotor domain, we used crosses and circles (represented as many-sided polygons) as primitives; the SSL domain already has line segments, circles, rectangles, and stadia as primitives. Figure 3.5 shows an example of an frame from an output video for the quadrotor, along with the image created by drawing the same primitives directly in a 2-D image coordinate system; Figure 1.2 shows the same for the SSL.

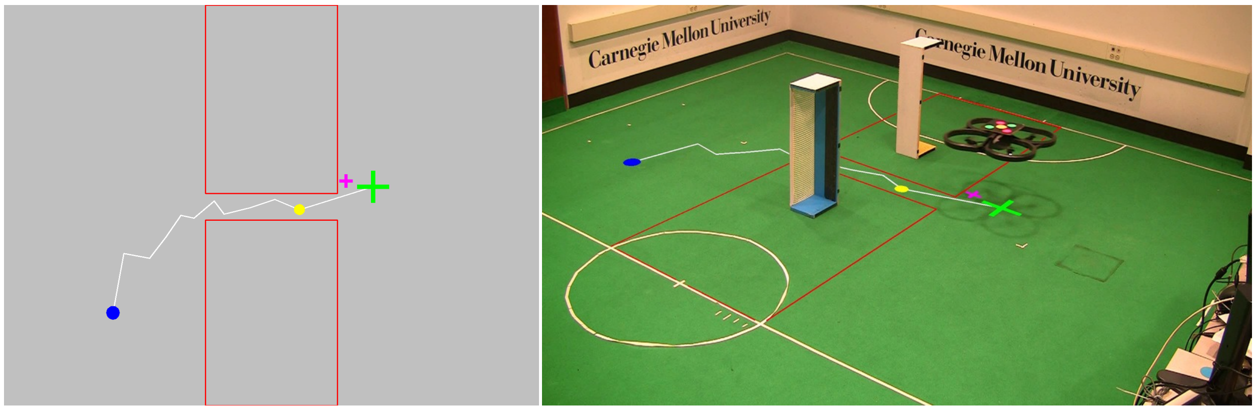


Figure 3.5: Left: A 2-D visualization generated from the interesting events of one run of the RRT, using a domain based on physical obstacles. Right: The result of drawing that visualization onto the corresponding frame from the video of the real robot.

Chapter 4

Proposed work

The goal of this thesis is to develop abstractions and algorithms to enable the creation of augmented visualizations that expose internal details of the planning and execution of autonomous agents and mobile robots.

We have been deeply involved in a complex robotic system, the SSL team, and begun to study the problem of visualization in more depth on a simpler domain that still demonstrates interesting autonomy, the quadrotor navigation problem; in this thesis, we will return to more complex systems and develop generalized methods that can handle such systems.

4.1 Visualization-focused execution logging for autonomous agents

The work of this thesis will depend on having a uniform representation of the information logged from the execution of an autonomous agent. For the visualizations we have produced so far, we used storage methods tailored to the individual applications; in the case of the SSL visualizations, we used the extensive but low-level logging format already available.

In general, every robotic system will have its own collection of kinds of relevant information that should be recorded. For instance, the quadrotor navigation problem has the current and target positions, the positions explored during planning, and the sequence of positions in the final path found. The SSL team has a large set of information of different kinds from all levels of the STP hierarchy. Although there are differences between different systems, there are similarities too; we propose to examine the commonalities between the debugging and visualization needs of the different robotic systems available for us to work with. This examination will then guide us in designing general representations that capture the needs of general robotic systems. We anticipate that the results will store *metadata*, such as the algorithmic component that generated a particular piece of information, or information relating to the location in the hierarchy of the control algorithm. The metadata will then be used to determine conditions under which each piece of information will be displayed.

Practically, once we have such a uniform representation and abstract data structures that incorporate the quantitative execution data along with any necessary metadata, we will also need concrete algorithms that manipulate such representations; we also propose to make our approach available for integrating with existing robotic projects.

When showing a particular visualization, we must address the issue of displaying only the needed information. The purpose of the extra metadata that we propose to store along with the basic execution information is to allow selecting of a subset of the execution information to display; we plan to create means of performing this selection that works well with the structure of generic robot autonomy.

4.2 Video processing for camera information extraction

A key element of creating the visualizations we have described is the processing of a video to extract time, transformation, and masking information so that we can add annotations to it. Currently, for the sake of generality, we plan to assume that there is no additional pose sensor information associated with the input video; we might get such information if the video were recorded from a camera attached to a virtual reality headset. We will assume that only the video itself is available, so our approach will need to extract the necessary information directly from the video.

As discussed in Section 2.4, many methods exist to extract motion information from videos; we intend to additionally make use of the information available from the execution logs. We propose to devise a method to estimate the time offset and homography automatically in the case that reasonably precise positioning information is available for the robots in the system, based on correlating the observed motion in the video with the recorded positions.

4.3 Demonstrations with real robotic systems

In order to demonstrate the practicality of the algorithms and representations discussed in this thesis, we will use them to create a variety of visualizations using the robotic systems available to us. These systems include at least the SSL team and quadrotors; we have already done some work with these two robotic systems, and intend to apply the proposed work to them as well. We will also integrate this work with our autoref software and possibly the CoBot robots, as available. We propose to show that the introduction of the abstractions and libraries of this thesis enable the creation of accurate visualizations from autonomous robots generally across robotic domains.

4.4 Display and interfaces for multiple levels of detail

As part of the overall system for using the visualizations we propose, we need to be able to display the set of available pieces of information for any particular use with a robotic system.

We propose to, as part of the implementation part of this thesis, create user interfaces for manipulating and displaying the stored visualizations for a system. Such an interface would need to expose the possible data filters alongside a display of the video and the overlaid representations. In general, we propose to make it possible for others to create flexible robotic systems, without dictating what particular choices are best for any particular person or situation. Accordingly, we consider detailed user studies to be out of scope for this thesis.

4.5 Summary and timeline

In summary, we propose to build upon our extensive experience with autonomy in complex robotic systems and contribute automated visualization to capture their reasoning by annotating videos of the robots' execution. My proposed timeline for this work is:

- Fall 2016
  - Formalize the commonalities in logging needs across different robotics domains
  - Design the abstract specifications of storage formats to accommodate these commonalities
- Spring 2017
  - Produce concrete algorithms and implementations that enable the creation of our augmented reality visualizations
- Summer 2017
  - Demonstrate a variety of visualizations with different robotic systems
  - Create interfaces for displaying and interacting with the created visualizations
- Fall 2017
  - Write and defend thesis

Bibliography

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

[2] Algomation, 2014. URL http://www.algomation.com.

[3] Algoviz, 2009. URL http://www.algoviz.org.

[4] Rajeev Alur, Costas Courcoubetis, Thomas A Henzinger, and Pei-Hsin Ho. Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems. In *Hybrid Systems*, Lecture Notes in Computer Science, pages 209–229. Springer, 1993.

[5] Peter Amstutz and Andrew H. Fagg. Real time visualization of robot state with mobile virtual reality. In *IEEE International Conference on Robotics and Automation*, volume 1, pages 241–247. IEEE, 2002.

[6] Matías Arenas, Javier Ruiz-del Solar, Simón Norambuena, and Sebastián Cubillos. A robot referee for robot soccer. In *RoboCup 2008: Robot Soccer World Cup XII*, pages 426–438. Springer, 2009.

[7] Ronald T. Azuma. A survey of augmented reality. *Presence*, 6(4):355–385, 1997.

[8] Ronald T. Azuma and Gary Bishop. Improving static and dynamic registration in an optical see-through HMD. In *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques*, pages 197–204, 1994.

[9] Michael Bajura and Ulrich Neumann. Dynamic registration correction in video-based augmented reality systems. *IEEE Computer Graphics and Applications*, 15, 1995.

[10] Thomas Ball and Stephen G. Eick. Software visualization in the large. *Computer*, 29 (4):33–43, Apr 1996. ISSN 0018-9162. doi: 10.1109/2.488299.

[11] Craig Barron. Matte painting in the digital age. In *ACM SIGGRAPH 98 Conference abstracts and applications*, page 318. ACM, 1998.

[12] 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 *AAMAS*, 2014.

[13] Joydeep Biswas, Juan Pablo Mendoza, Danny Zhu, Steven Klee, and Manuela Veloso. CMDragons 2014 extended team description paper. In *Proceedings of the 18th RoboCup International Symposium*, 2014.

[14] Michael Bowling, Brett Browning, Allen Chang, and Manuela Veloso. Plays as team plans for coordination and adaptation. In *RoboCup 2003 Symposium*, pages 686–693, 2004.

[15] Gary Bradski. OpenCV. *Dr. Dobb's Journal of Software Tools*, 2000.

[16] Marc H. Brown and Robert Sedgewick. A system for algorithm animation. In *Proceedings of the 11th Annual Conference on Computer Graphics and Interactive Techniques*, pages 177–186. ACM, 1984.

[17] Brett Browning, James Bruce, Michael Bowling, and Manuela Veloso. STP: Skills, tactics, and plays for multi-robot control in adversarial environments. *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, 219(1):33–52, 2005.

[18] James Bruce and Manuela Veloso. Real-time randomized path planning for robot navigation. In *Proceedings of the 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 3, pages 2383–2388. IEEE, 2002.

[19] James Bruce and Manuela Veloso. Fast and accurate vision-based pattern detection and identification. In *Proceedings of the 2003 IEEE International Conference on Robotics and Automation*, volume 1, pages 1277–1282. IEEE, 2003.

[20] Ravi Teja Chadalavada, Henrik Andreasson, Robert Krug, and Achim J. Lilienthal. That's on my mind! robot to human intention communication through on-board projection on shared floor space. In *European Conference on Mobile Robots*, 2015.

[21] Mao Chen, Klaus Dorer, Ehsan Foroughi, Fredrik Heintz, Zhanxiang Huang, Spiros Kapetanakis, Kostas Kostiadis, Johan Kummenje, Jan Murray, Itsuki Noda, Olivor Obst, Pat Riley, Timo Steffens, Yi Wang, and Xiang Yin. *RoboCup Soccer Server Users Manual*, 2003. URL http://sourceforge.net/projects/sserver/files/rcssmanual/9-20030211/.

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

[23] Hosam K. Fathy, Zoran S. Filpi, Jonathan Hagena, and Jeffrey L. Stein. Review of hardware-in-the-loop simulation and its prospects in the automotive area. In *Proc. SPIE*, May 2006.

[24] Martin A. Fischler and Robert C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

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

[26] Michael Friendly and Daniel J. Denis. Milestones in the history of thematic cartography, statistical graphics, and data visualization, 2001. URL http://www.datavis.ca/milestones.

[27] Jean-Bernard Hayet, Justus Piater, and Jacques Verly. Robust incremental rectification of sports video sequences. In *British Machine Vision Conference (BMVC'04)*, pages 687–696, 2004.

[28] William A. Hoff, Khoi Nguyen, and Torsten Lyon. Computer vision-based registration techniques for augmented reality. In *Proceedings of Intelligent Robots and Computer Vision*, 1999.

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

[30] Berthold K. Horn and Brian G. Schunck. Determining optical flow. *Proc. SPIE*, 0281:319–331, 1981. doi: 10.1117/12.965761. URL http://dx.doi.org/10.1117/12.965761.

[31] R. Isermann, J. Schaffnit, and S. Sinsel. Hardware-in-the-loop simulation for the design and testing of engine-control systems. *Control Engineering Practice*, 7(5):643–653, May 1999. ISSN 0967-0661.

[32] George H. Joblove and Donald Greenberg. Color spaces for computer graphics. In *Proceedings of the 5th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '78, pages 20–25, New York, NY, USA, 1978. ACM.

[33] Uwe-Philipp Käppeler, Oliver Zweigle, Kai Häußermann, Hamid Rajaie, Andreas Tamke, Andreas Koch, Bernd Eckstein, Fabian Aichele, Daniel DiMarco, Adam Berthelot, Thomas Walter, and Paul Levi. RFC Stuttgart team description 2010, 2010. URL ftp://ftp.informatik.uni-stuttgart.de/pub/library/ncstrl.ustuttgart_fi/INPROC-2010-97/INPROC-2010-97.pdf.

[34] Won S. Kim. Virtual reality calibration and preview/predictive displays for telerobotics. *Presence: Teleoperators and Virtual Environments*, 5(2):173–190, 1996.

[35] Hiroaki Kitano, Minoru Asada, Yasuo Kuniyoshi, Itsuki Noda, and Eiichi Osawa. RoboCup: The robot world cup initiative. In *Proceedings of the first international conference on Autonomous agents*, pages 340–347. ACM, 1997.

[36] Steven M. LaValle. Rapidly-exploring random trees: A new tool for path planning. Technical Report 98-11, Iowa State University, 1998.

[37] David Lowe. Object recognition from local scale-invariant features. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, volume 2, pages 1150–1157, 1999.

[38] Juan Pablo Mendoza, Joydeep Biswas, Danny Zhu, Richard Wang, Philip Cooksey, Steven Klee, and Manuela Veloso. CMDragons 2015: Coordinated offense and defense of the SSL champions. In *RoboCup 2015: Robot World Cup XIX*, pages 106–117. Springer, 2015.

[39] Juan Pablo Mendoza, Manuela Veloso, and Reid Simmons. Detecting and correcting model anomalies in subspaces of robot planning domains. In *AAMAS*, 2015.

[40] Paul Milgram and Herman Colquhoun. A taxonomy of real and virtual world display integration. *Mixed reality: Merging real and virtual worlds*, pages 1–26, Mar 1999.

[41] Paul Milgram and Fumio Kishino. A taxonomy of mixed reality visual displays. *IEICE Transactions on Information and Systems*, 77(12):1321–1329, 1994.

[42] Paul Milgram, Anu Rastogi, and Julius J. Grodski. Telerobotic control using augmented reality. In *IEEE International Workshop on Robot-Human Communication*, pages 21–29, Jul 1995.

[43] Tim Niemüller, Gerhard Lakemeyer, Alexander Ferrein, S Reuter, D Ewert, S Jeschke, D Pensky, and Ulrich Karras. RoboCup Logistics League sponored by Festo: A competitive factory automation testbed. In *Proceedings of the 16th International Conference on Advanced Robotics — 1st Workshop on Developments in RoboCup Leagues*, 2013.

[44] D. Nister, O. Naroditsky, and J. Bergen. Visual odometry. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages 652–659, June 2004.

[45] Kenji Okuma, James J. Little, and David G. Lowe. Automatic rectification of long image sequences. In *Asian Conference on Computer Vision*, Jan 2004.

[46] Blaine A. Price, Ian S. Small, and Ronald M. Baecker. A taxonomy of software visualization. *Journal of Visual Languages and Computing*, 4:211–266, 1992.

[47] 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.

[48] David Robert and Cynthia Breazeal. Blended reality characters. In *Proceedings of the seventh annual ACM/IEEE international conference on Human-Robot Interaction*, pages 359–366. ACM, 2012.

[49] David Robert, Ryan Wistorrt, Jesse Gray, and Cynthia Breazeal. Exploring mixed reality robot gaming. In *Proceedings of the fifth international conference on tangible, embedded, and embodied interaction*, pages 125–128. ACM, 2011.

[50] RoboCup 2014 Technical Challenges. URL http://robocupssl.cpe.ku.ac.th/robocup2014:technical_challenges.

[51] RoboCup objective. URL http://www.robocup.org/about-robocup/objective/.

[52] Robotics Erlangen autoref. URL https://github.com/robotics-erlangen/autoref.

[53] Roy J. Rosser, Yi Tan, Howard J. Kennedy, Jr., James L. Jeffers, Darrell S. DiCicco, and Ximin Gong. Image insertion in video streams using a combination of physical sensors and pattern recognition, 2000. US Patent number US6100925 A.

[54] F.B.F. Schoenmakers, G. Koudijs, C.A. Lopez Martinez, M. Briegel, H.H.C.M. van Wesel, J.P.J. Groenen, O. Hendriks, O.F.C. Klooster, R.P.T. Soetens, and M.J.G. van de Molengraft. Tech United Eindhoven team description 2013: Middle Size League, 2013. URL http://www.techunited.nl/media/files/TDP2013.pdf.

[55] Jean Serra. *Image analysis and mathematical morphology*. Academic Press, Inc., 1983.

[56] Jianbo Shi and Carlo Tomasi. Good features to track. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 593–600, 1994.

[57] Simspark wiki: Soccer simulation. URL http://simspark.sourceforge.net/wiki/index.php?title=Soccer_Simulation&oldid=3043.

[58] ssl-autonomous-refbox. URL https://code.google.com/p/ssl-autonomous-refbox.

[59] Ryohei Takahashi, Hiroto Ise, Atsushi Konno, Masaru Uchiyama, and Daisuke Sato. Hybrid simulation of a dual-arm space robot colliding with a floating object. In *IEEE International Conference on Robotics and Automation*, pages 1201–1206, May 2008. doi: 10.1109/ROBOT.2008.4543367.

[60] Kumiko Tanaka-Ishii, Hideyuki Nakashima, Itsuki Noda, Kôiti Hasida, Ian Frank, and Hitoshi Matsubara. MIKE: An automatic commentary system for soccer. In *Proceedings of the International Conference on Multi Agent Systems*, pages 285–292. IEEE, 1998.

[61] Carlo Tomasi and Takeo Kanade. Detection and tracking of point features. Technical report, Tech Report CMU-CS-91-132, Carnegie Mellon University, 1991.

[62] Edward R. Tufte. *The visual display of quantitative information*. Graphics Press, 2001.

[63] Douglas Vail, Manuela Veloso, and John Lafferty. Conditional random fields for activity recognition. In *Proceedings of the 6th International Joint Conference on Autonomous Agents and Multiagent Systems*, page 235. ACM, 2007.

[64] Manuela Veloso, Nicolas Armstrong-Crews, Sonia Chernova, Elisabeth Crawford, Colin McMillen, Maayan Roth, Douglas Vail, and Stefan Zickler. A team of humanoid game commenters. *International Journal of Humanoid Robotics*, 5(03):457–480, 2008.

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

[66] Visualgo, 2011. URL http://www.visualgo.net.

[67] Dirk Voelz, Elisabeth André, Gerd Herzog, and Thomas Rist. Rocco: A RoboCup soccer commentator system. In *RoboCup-98: Robot Soccer World Cup II*, pages 50–60. Springer, 1999.

[68] Danny Zhu and Manuela Veloso. Event-based automated refereeing (under review).

[69] Danny Zhu, Joydeep Biswas, and Manuela Veloso. AutoRef: Towards real-robot soccer complete automated refereeing. In *RoboCup 2014: Robot World Cup XVIII*, pages 419–430. Springer, 2015.

[70] Stefan Zickler and Manuela Veloso. Efficient physics-based planning: sampling search via non-deterministic tactics and skills. In *AAMAS*, pages 27–33, 2009.

[71] 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.