Team Edinferno
Description Paper for RoboCup 2011 SPL

Subramanian Ramamoorthy, Aris Valtazanos, Efstathios Vafeias, Christopher Towell, Majd Hawasly, Ioannis Havoutis, Thomas McGuire*, Seyed Behzad Tabibian, Sethu Vijayakumar, Taku Komura

School of Informatics
The University of Edinburgh
10 Crichton Street
Edinburgh EH8 9AB
United Kingdom

Abstract. This paper outlines the organization and architecture of a robotic soccer team, *Team Edinferno*, developed at The University of Edinburgh. This paper serves as the qualification document for the 2011 RoboCup Standard Platform League competitions. We are a completely new team aspiring to compete in international competitions from 2011, so the team and software architecture have been built from scratch. We are also potentially the first ever SPL team from the United Kingdom. Our primary research interests are centered on issues of robot learning, especially for effective autonomous decision making and strategic behaviour in continually changing worlds. This is supported by solid foundations in robotic locomotion and full-body behaviours, on-board computer vision and communications software infrastructure.

1 Introduction and Team Composition

Team Edinferno is a new team, potentially the first Standard Platform League team from the United Kingdom. The team consists of graduate and undergraduate students and more experienced robotics researchers from the School of Informatics at The University of Edinburgh. We come from a strong research group studying robot learning (within the Institute of Perception, Action and Behaviour), situated within a very diverse and active community of AI and computer science researchers - one of the largest and best in the UK. The team is lead by Dr. S. Ramamoorthy, who has extensive background in robotics and machine learning, in academia and industry. Research within our group is organized around the central theme of understanding and developing autonomous decision making mechanisms in continually changing and strategically rich environments. The Standard Platform League affords a very tangible and interesting domain for our research work.

2 Software Modules and Core Capabilities

As a new team, we have invested significant efforts towards establishing all the basic modules required of a soccer playing agent. Although we have developed all of these

* Visiting Undergraduate Student from Universität Siegen, Germany

modules from the ground up, we have paid attention to ideas coming out of other established teams - as described in their description documents and technical reports. As seen in our supporting video, we have achieved all the basic capabilities required for us to be able to play in the SPL and look forward to that opportunity. The core architecture of our agent is summarized in figure 1.

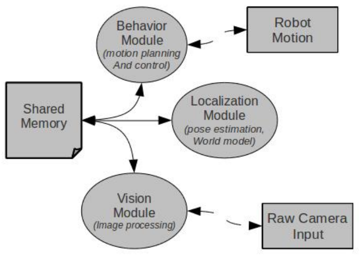


Fig. 1. Software architecture - major modules.

The core infrastructure is based on the existing Naoqi architecture. Every module exists in its own thread and runs on the robot's main broker (Nao's central program). The Behaviour module communicates with both Vision and Localization modules. The processes are invoked through normal function calls, while data is exchanged through shared memory. Each module publishes its result into ALMemory (based on a blackboard architecture). So, Behaviour module calls the vision updates, calls Monte Carlo localization update when Vision is done and also provides motion information (odometry updates). The Vision module posts results of object recognition to shared memory. The Localization module updates the global position of the robot in the shared memory block to be available to the other modules.

On-board Vision Our vision module uses color segmentation as the basic operation, exploiting colour cues to find ball, field lines, goal posts, etc. We train our segmentation algorithm, in the YUV colour space, using sample images obtained under various conditions. This yields binary images for each base colour. In these images, we search for contours using OpenCV functions. This allows us to handle two types of vision tasks. A simple task is something like finding waistbands on robots, which we approach as a region finding operation - using bounding boxes. More complex tasks include circle detection, which require a bit more processing as discussed below. We also include operations for finding and using the horizon as well as calibrating against fixed systematic errors (e.g., in head pitch angle).

The Vision module provides the localization module with the key features it needs: goal posts' position, cross spots' and circle position. Circle detection works by finding intersections between all lines that are perpendicular to the contour lines of white regions. The distance of the intersection to the contour needs to be roughly the same as the radius of the field circle. So, intersections are clustered and the cluster with the highest score is used to define the circle. All of this computation is defined in field space, based on projections from image space.

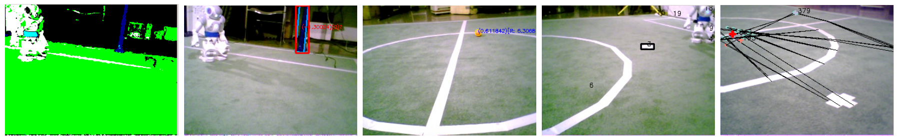


Fig. 2. Representative output of the Vision module - real time detection of key features based on colour segmentation.

Probabilistic Localisation Our Monte Carlo localization is based on visual features that are detected in real time - the locations of goal posts, cross spots and the circle. We implement an augmented version of a particle filter algorithm [1] to avoid the standard problems of the kidnapped robot and false convergence. The particle filter uses a motion model that draws on odometry information. Once the particle filter has been initialized (which only takes a few steps and scans), we are in a position to average the beliefs for the purposes of further decision making, e.g., motion planning. Our supporting video includes complete demonstration sequences showing this module in action.

Locomotion and Behaviour The architecture of the Behaviour module is summarized in figure 3.

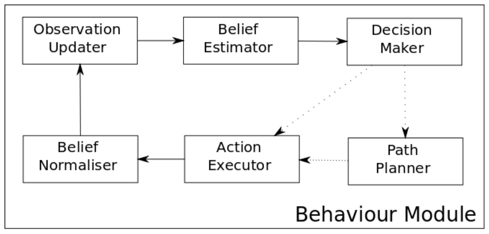


Fig. 3. Architecture of the Behaviour module.

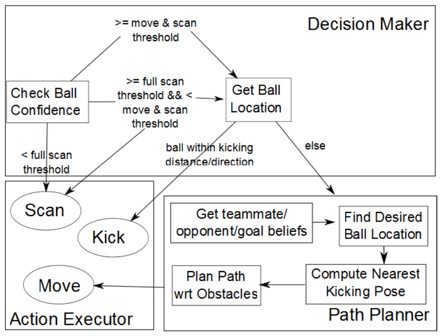
The key operations within this module are as follows:

- Observation Updater: polls the vision module for the latest image coordinates of relevant objects and converts into world coordinates, relative to an egocentric reference frame. World coordinates are computed using a distance/direction estimation function, which accounts for the robot's head yaw and pitch angles, as well as the focal length of its camera. For every visible object, the observation updater generates a new observation, which is the simplest representation of the robot's current perception of the world (e.g. "a pink robot at a distance of 0.5 metres"). If the ball is visible, this also computes desired head yaw and pitch angles for the next frame, in order to center the camera image.
- Belief Estimator: generates the robot's current beliefs on the state of the other objects. A belief is generated by matching a current observation to a past belief (e.g. "the pink robot at a distance of 0.5 meters is the pink robot seen at the previous frame at a distance of 0.9 meters"). Beliefs also contain a confidence attribute; if an object is visible, the robot's confidence on the state of this object is set to 1.0,

otherwise, for every frame that the object is not visible, this confidence gradually decays. In order to generate a new belief for another robot, the belief estimator passes each robot observation through a particle filter, which computes the likelihood of the current observation given the previous beliefs and the robot's motion and sensory model.

– Decision Maker: uses the robot's current beliefs to decide which high level action should be taken. The chosen action depends on the robot's role (attacker, defender, goalkeeper) and its levels of confidence on the various features and attributes of interest (robot's own location in the field, ball location, location of the other robots). Several branches of the decision making state machine also depend on some confidence thresholds, e.g., depending on their confidence on the location of the ball, attackers may choose to "move" towards it, "move while scanning with their head", or "stop and do a full head scan". Examples of other actions include "kick with left foot", "get up from back" and "scan for teammates". These actions are also clustered; for example, moving to the ball and moving while avoiding another robot both fall under the same "move" action label.
– Path Planner: computes the trajectory to take towards a point of interest. It first computes a desired target pose, e.g., if the robot knows ball and goal location, a target pose is a pose that will allow it to kick *towards* the goal. For kicking-related actions, there are four candidate target poses (for straight left kick, right straight kick, left side kick and right side kick), and the path planner selects the closest one based on current pose. Then, a path is constructed from start to end pose. If there are no obstacles, the path is linear - created by interpolation. Otherwise, for each obstacle, the path planner computes various landmark points which are outside a *safety* distance from the obstacle. If multiple obstacles are present, only the landmarks that are *safe with respect to all obstacles* are retained. Then, the planner selects the landmark which minimises the distance to the target pose, and splits the generated path into two parts (current pose$\rightarrow$safe landmark, safe landmark$\rightarrow$target pose).
– Action Executor: executes the chosen action. Kicking and get-up actions comprise predefined motion sequences. For moving actions, the action executor takes the first point of the computed path, and computes the velocity vector (x, y, θ) with which the robot should move to reach this point. For scanning actions, the action executor keeps track of the last angle pair scanned by the robot, and updates these angles accordingly.
– Belief Normaliser: updates all the robot's estimates (observations, beliefs, and particles) based on the robot's most recent movement. This movement is computed through odometry and corresponds to changes in the x-y coordinates and the orientation of the robot. Using this movement, estimates are translated and normalised so that they are in the correct position, relative to the robot's new (transformed) frame of reference.

Communication and Coordination Certain kinds of communication are mandatory. So, we implement the ability to enable robots to obey both the GameController and the button interface. Button presses are correctly and automatically sent back to the GameController. Of more interest is the ability to communicate a shared state and use this for coordinated motion. We have the ability to share localization and belief information between our robots, in a compact form. This enables robots to implement simple



Decision Maker
>= move & scan threshold
Check Ball Confidence
>= full scan threshold && < move & scan threshold
Get Ball Location
< full scan threshold
ball within kicking distance/direction
else
Scan
Kick
Move
Action Executor
Get teammate/ opponent/goal beliefs
Find Desired Ball Location
Compute Nearest Kicking Pose
Plan Path wrt Obstacles
Path Planner

Fig. 4. Decision making architecture - *attacker* module.

heuristics such as allowing the best placed robot to take on the role of striker, avoiding on-field conflicts[1].

3 Key Research Contributions and Future Directions

Strategic and Multi-objective Motion Planning Based on the solid foundation already fully implemented in our agents, we are exploring more sophisticated ways to make strategic decisions in response to opponent strategies. Our current work [2] uses a description of possible opponent strategies in terms of reachable sets, based on models of dynamics and system performance limitations (e.g., maximum velocity along each direction). It is possible to pre-compute template reachable sets for key motion hypotheses. Then, by utilizing real time information from particle filter based estimation of opponents' positions, we are able to select - *online* - the 'best' reachable set and plan safe paths with respect to this. This is already implemented and one particular benefit of this approach is that it allows us to overcome the limitation of sparse, slow and partial observability by utilizing a more sophisticated, predictive, model of the opponent. We are currently exploring ways to achieve tighter coupling between positional estimation and strategy hypotheses, and also taking into account multi-objective performance criteria.

Full-body Humanoid Robot Behaviours A long standing strength within our research group is in the area of machine learning for humanoid locomotion [3, 4] and full-body humanoid behaviours [5, 6]. It may be worth noting that this work [5] was considered as a finalist for the *RoboCup Best Paper Award* at IROS 2010. In our current implementation, shown in supporting video, we build on the existing ALMotion engine as this was the quickest way for us to get the team going. This has already been extended in specific cases such as for directional kicks and the goal keeper. In future, we plan

[1] We have full and state of the art functionality, as illustrated in our supporting video and discussed in following section, for avoiding static and dynamic obstacles. Here, we refer to the ability to avoid the situation in the first place!

to implement more of our research outputs in specific roles such as focussed kicks for passing, maintaining stability while kicking powerfully and adjusting body postures in tight situations such as for the goal keeper and when many robots are nearby.

Team-level Strategy Design and Learning (related to our 2D Simulation League team) In addition to the SPL team, we also plan to enter the 2D Simulation League. This latter league allows for *significant* strategic sophistication. We are exploring a layered architecture where by fusing locally-sensed observations we build a shared compact representation of world state and game dynamics. This representation is encoded in predicate-like statements. Distributed local control is used to drive the system in this predicate space to the desired goal states. The local controllers are designed and learnt off-line for specific tasks and subgoals, embedding domain knowledge. Using reinforcement learning mechanisms, the controllers are optimized against various classes of opponents and standard team mate behaviours. To maintain robustness in online interactions, online estimation of the quality and robustness of the controllers is utilized to compensate for unknown opponents and situations.

In current and future work, we are adapting specific ideas from this simulation league framework to improve the coordinated behaviour of our SPL agents. Currently, with few exceptions, SPL agents seem to view opponents as essentially 'noise' to be overcome. We hope to be able to go a little bit further in the direction of understanding opponent intent and reacting to or otherwise gainfully utilizing this information.

4 Conclusions

Team Edinferno is a new SPL team, potentially the first one from the United Kingdom. The underlying research work builds on strong background in robot learning and aims to advance the state of the art of autonomous decision making in continually changing worlds. Although the team is still in formative stages (e.g., in possession of only two RoboCup version Naos), we have successfully implemented fully functional agents demonstrating all required functionality. We plan to deploy this in live competition at the Mediterranean Open in March 2011. This will help us become fully ready for the main RoboCup event in Istanbul.

References

1. S. Thrun, W. Burgard, D. Fox, *Probabilistic Robotics*, MIT Press, 2006.
2. A. Valtazanos, S. Ramamoorthy, Online motion planning for multi-robot interaction using composable reachable sets, Under Review.
3. S. Ramamoorthy, B.J. Kuipers, Trajectory generation for dynamic bipedal walking through qualitative model based manifold learning. In *Proc. Int. Conf. Robotics and Automation*, pp. 359-366, 2008.
4. I. Havoutis, S. Ramamoorthy, Motion synthesis through randomized exploration on submanifolds in configuration space. In J. Baltes et al. (Eds.): *RoboCup Symp. 2009, LNAI 5949*, pp. 92-103, 2010.
5. I. Havoutis, S.Ramamoorthy, Constrained geodesic trajectory generation on learnt skill manifolds. In *Proc. Int. Conf. Intelligent Robots and Systems*, 2010.
6. C. Towell, M. Howard, S. Vijayakumar, Learning nullspace policies, In *Proc. Int. Conf. Intelligent Robots and Systems*, 2010.