Kouretes 2013 SPL Team Description Paper*

Nikolaos Kargas, Nikolaos Kofinas, Evangelos Michelioudakis,
Nikolaos Pavlakis, Stylianos Piperakis,
Nikolaos I. Spanoudakis, Michail G. Lagoudakis

Intelligent Systems Laboratory, Technical University of Crete, Chania, Greece
www.kouretes.gr

1 Team Information

Team Kouretes was founded in 2006 and participates in the main RoboCup competition ever since in various leagues (Four-Legged, Standard Platform, MSRS, Webots), as well as in various local RoboCup events (German Open, Mediterranean Open, Iran Open, RC4EW, RomeCup) and RoboCup exhibitions (Athens Digital Week, Micropolis, Schoolfest). In May 2010, the team hosted the 1st official SPL tournament in Greece (with three invited teams) within the Hellenic Conference on Artificial Intelligence (SETN). The team has been developing its own (publicly-available) software[1] for the Nao robots since 2008. Distinctions of the team include: **2nd** place in MSRS at RoboCup 2007; **3rd** place in SPL-Nao, **1st** place in SPL-MSRS, among the **top 8** teams in SPL-Webots at RoboCup 2008; **1st** place in RomeCup 2009; **6th** place in SPL-Webots at RoboCup 2009; **2nd** place in SPL at RC4EW 2010; and **2nd** place in SPL Open Challenge Competition at RoboCup 2011 (joint team Noxious-Kouretes).

Team Kouretes is led by Michail G. Lagoudakis (associate professor) and Nikolaos I. Spanoudakis (laboratory staff). In 2013, the team counts five ECE student members (brackets indicate the main research area of each member):

Nikolaos Kofinas	postgraduate	[Forward and Inverse Kinematics]
Stylianos Piperakis	postgraduate	[Omnidirectional Walk Engine]
Nikolaos Kargas	undergraduate	[Local State Estimation]
Nikolaos Pavlakis	undergraduate	[Global State Estimation]
Evangelos Michelioudakis	undergraduate	[Multi-Robot Coordination]

2 Team Research

2.1 Software Architecture and Communication

The team's code is based on MONAS [5], a software architecture developed in-house to address the needs of the team. MONAS provides an abstraction layer from the Nao robot and allows the synthesis of complex robotic teams as XML-specified Monas modules and/or statechart modules. Monas modules focus on specific functionalities (vision, motion, etc.); they are executed concurrently, each one of them at any desired frequency completing a series of activities at

* The team has been supported by the Technical University of Crete, the Grant MCIRG-CT-2006-044980, Chipita S.A.–Molto, and the Telecom Systems Institute.

[1] Team Kouretes public code repository: https://github.com/kouretes/Monas

each execution. Statechart modules [6] are based on the Agent Systems Engineering Methodology (ASEME), whereby the target team behavior is specified through a series of model-based transformations as a statechart. Statecharts are executed using a generic multi-threaded statechart engine that provides the required concurrency and meets the real-time requirements of the activities on each robot.

MONAS relies on our intra-robot and inter-robot messaging system, called NARUKOM [9]. NARUKOM is based on the publish/subscribe paradigm and supports multiple ways of communication, including point-to-point and multicast connections. The data shared between nodes/threads are stored on local blackboards which are transparently accessible from all nodes/threads. Communication is achieved through messages tagged with appropriate topics and relayed through a message queue implemented using Google protocol buffers to facilitate the serialization of data and the structural definition of the messages. Three types of messages are supported: (i) *state*, which remain in the blackboard until they are replaced by a newer message of the same type, (ii) *signal*, which are consumed at the first read, and (iii) *data*, which are time-stamped to indicate the precise time the values they carry were acquired. Data messages have expiration times, so that they are automatically removed when they are no longer needed.

2.2 Kinematics

We have completed our own analytical derivation and implementation of the forward and inverse kinematics for the whole body of the Nao robot (head, arms, legs) [2,3]. The proposed solution was made feasible through a decomposition into five independent problems (head, two arms, two legs), the use of the Denavit-Hartenberg method, and the analytical solution of a non-linear system of equations. The main advantage of the proposed inverse kinematics compared to existing numerical approaches is its accuracy, its efficiency, and the elimination of singularities. The implemented NAO kinematics library[2] is optimized for real-time, on-board execution and offers center-of-mass calculation. Forward kinematics are used to determine the exact camera position in the physical space given the readings from the joint encoders. This allows the precise synchronization of a camera image with the exact camera position given the joint angles at the very moment the image was obtained. The synchronization leads to significant improvements in estimating the distance and bearing of recognized objects. In addition, inverse kinematics of the head are used to point the camera to any desired field position; this feature is particularly useful in turning the head towards an estimated ball position after a wide scan for landmarks while walking.

2.3 Walk Engine

We have recently started working on our own omnidirectional walk engine, KWALK, which offers robust behavior when disturbances and ground anomalies are present. The proposed walk engine architecture is shown in Figure 1.

[2] Library download link: www.github.com/kouretes/NAOKinematics

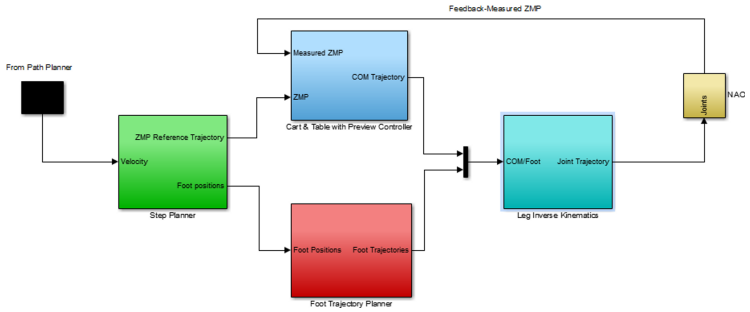

Fig. 1. The architecture of KWalk for stable omnidirectional locomotion.

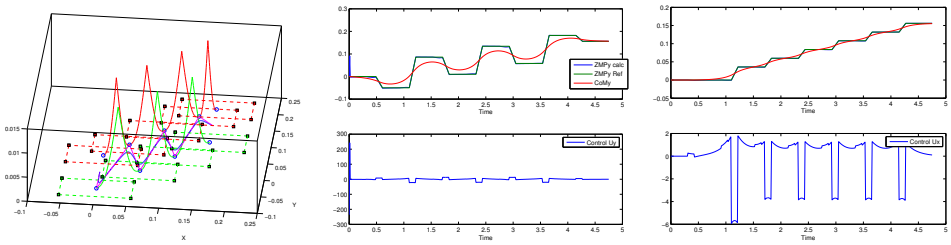

Fig. 2. Planned steps and COM, ZMP, and control along the lateral and sagittal planes.

First, we compute the position of each step on the plane using only the desired torso's velocities (longitudinal, lateral, rotational) and the physical constraints the robot induces. Then, the reference Zero Moment Point (ZMP) trajectory of the generated walking pattern is computed. Next, given the desired foot placements on the plane, the trajectories of the feet are computed. Figure 2 (left) shows seven consecutive diagonal steps computed by the step planner and the foot trajectories generated by the foot trajectory planner using a 5-th order polynomial interpolation. An efficient way to approach NAO's leg dynamics is the Cart-and-Table model, in which the Center of Mass (COM) of the NAO is modeled and approximated as a cart on a table. These dynamics are directly controlled with a Preview Controller treated as a ZMP tracking servo problem. The improvement compared to the common Linear-Inverted-Pendulum model is significant, especially in the present of disturbances. Figure 2 shows the computed COM and ZMP trajectories contrasted with the reference ZMP trajectory for both planes, lateral (middle) and sagittal (right), along with the corresponding control policy. Finally, given the foot trajectories and the COM trajectory, we employ the inverse kinematics for the two legs to determine the correct leg joint values, shown in Figure 3 for both legs.

Given the foot center positions on the plane and the foot orientations about the vertical axis, instead of using an ordinary way of interpolation, such as piecewise Bezier curves, for computing the foot trajectory in space, we study two ways of rigid body interpolation, which compute optimal trajectories by

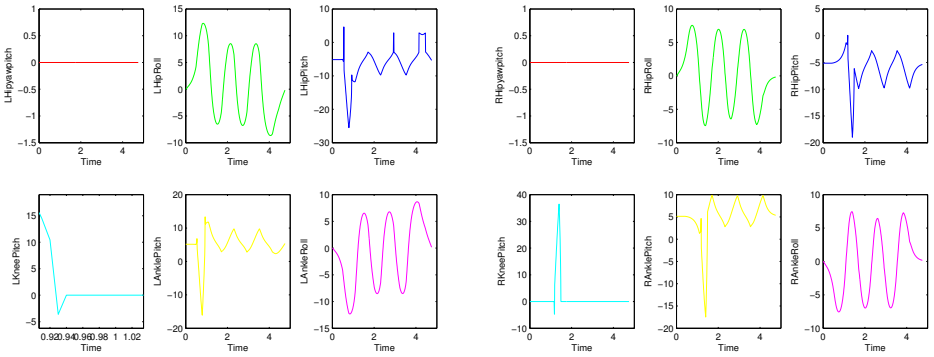

Fig. 3. Nao leg joint values (left and right) for the diagonal steps in Figure 2.

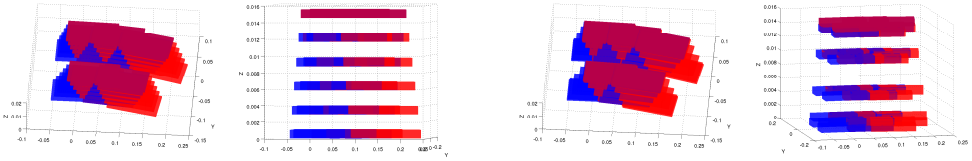

Fig. 4. Foot trajectories from minimizing geodesic distance (left) or jerk (right).

minimizing integral function costs involving left-invariant Riemannian metrics. The first (Figure 4, left) is the geodesic curve (shortest path in SE(3)) and the second (Figure 4, right) is the minimum jerk curve, whereby the accelerations and the velocities of the foot in the initial and ending points in space are zero. The resulting trajectories are smooth in both cases.

2.4 Vision

Object recognition is accomplished by KVision [4], a light-weight image processing method for visual object recognition focusing on reliability and efficiency. The vision pipeline uses real-time sensory information from the joints along with the robot's kinematic chain to determine the exact camera position in the 3-dimensional space and subsequently the view horizon and the sampling grid, so that scanning is approximately uniformly projected over the ground (field), not over the image matrix. Special attention is paid on the precise synchronization of images with robot joint values using time-stamped data messages. The next phase employs our auto-calibrated color recognition scheme on the pixels of the sampling grid to identify regions of interest. In the last phase, detailed analysis of the identified regions of interest seeks potential matches for the corresponding target object. These matches are evaluated and filtered by several heuristics, so that the best match (if any) in terms of color, shape, and size for a target object is finally extracted. Then, the corresponding object is returned as perceived, along with an estimate of its current distance and bearing as well as appropriate head joint angles for fixating and tracking it.

2.5 Local State Estimation

For self-localization, a different approach, based on Extended Kalman Filter (EKF), was recently implemented replacing our previous Monte-Carlo localization approach with particle filters. The benefit of EKF, besides being computationally more efficient, is that it enables easier integration of robot and ball positions into a shared world state model, which can be used for developing team strategies or informing uncertain robots about their position. The ambiguity in landmark observations made it necessary to track multiple hypothesis in order to cope with the data association problem. When performing an ambiguous measurement that may correspond to L different landmarks, each of the existing models is split into L models. To alleviate the exponential growth of the population of models, less-likely models are discarded, while redundant ones are merged. Another issue that had to be resolved was the existence and the variance of systematic errors related to robot odometry. The state of the filter was augmented with three parameters resulting in a six-state filter. The augmented filter accounts for systematic errors, such as drift, since the true value of each parameter is being estimated online and is used when applying the odometry motion model in the prediction step of the algorithm.

A separate four-dimensional EKF filter is used to track the position and the velocity of the ball in the field relatively to the observer robot. We assume that the ball's motion straight and subject to constant decceleration due to friction.

2.6 Global State Estimation

A new module, Shared World Model, was created and embedded into our architecture. This module is responsible for collecting all the local world state beliefs, performing filtering in order to integrate them, and producing a consistent belief about the global state of the game. This module runs on each robot, monitors the UDP network, and listens for local belief messages from all the robots (including self) containing information about their pose and the ball observation (if any). For each incoming message, a filtering step is performed in order to incorporate the new information into the current global belief (currently, position of all teammates and ball position). Therefore, sharing and fusing the local beliefs, enables each robot to correct local errors and decide what to do based on a better estimation of the global state of the game. We chose to use an Extended Kalman Filter (EKF) for global state estimation, because it fits our needs and it is proven to produce near-optimal results, if the models are realistic and the linearization of the models is performed carefully. Each robot needs to maintain a $(3N + 2)$-dimensional Kalman filter, where N represents the number of robots in the team; three variables (x_i, y_i, θ_i) for the pose (location and orientation) of robot i and two variables (x_b, y_b) for the location of the ball. An update step takes the observation $(robot_x, robot_y, robot_\theta, ball_x, ball_y)$ and updates only the blocks of the Kalman matrices that correspond to this specific robot. The Kalman matrices for our problem are described below:

$\hat{\mathbf{x}}$: $(3N + 2) \times 1$, represents the global state we need to estimate.

$\mathbf{P}$: $(3N+2) \times (3N+2)$, represents the uncertainty of the filter.
$\mathbf{F}$: $(3N+2) \times (3N+2)$ is the transition model.
$\mathbf{Q}$: $(3N+2) \times (3N+2)$, is the transition noise covariance.
$\mathbf{H}$: $3 \times (3N+2)$ (no ball observation), $5 \times (3N+2)$ (with ball observation) is the observation model.
$\mathbf{R}$: 3×3 (no ball), 5×5 (with ball), is the observation noise covariance.
$\mathbf{z}$: 3×1 (no ball), 5×1 (with ball), represents the observation.
$\tilde{\mathbf{y}}$: 3×1 (no ball), 5×1 (with ball), the innovation matrix capturing the measurement residual (difference between measurement and estimated value).
$\mathbf{S}$: 3×3 (no ball), 5×5 (with ball), represents the innovation noise covariance.
$\mathbf{K}$: $(3N+2) \times 3$ (no ball), $(3N+2) \times 5$ (with ball), represents the optimal Kalman gain and determines the effect of the innovation matrix on the current belief.

The EKF update equations we used are summarized below:

$$\begin{aligned} \textit{Predict}: \quad \hat{\mathbf{x}}_{k|k-1} &= \mathbf{F}_k \hat{\mathbf{x}}_{k-1|k-1} \\ \mathbf{P}_{k|k-1} &= \mathbf{F}_k \mathbf{P}_{k-1|k-1} \mathbf{F}_k^\top + \mathbf{Q}_k \end{aligned}$$

$$\begin{aligned} \textit{Update}: \quad \tilde{\mathbf{y}}_k &= \mathbf{z}_k - \mathbf{H}_k \hat{\mathbf{x}}_{k|k-1} \\ \mathbf{S}_k &= \mathbf{H}_k \mathbf{P}_{k|k-1} \mathbf{H}_k^\top + \mathbf{R}_k \\ \mathbf{K}_k &= \mathbf{P}_{k|k-1} \mathbf{H}_k^\top \mathbf{S}_k^{-1} \\ \hat{\mathbf{x}}_{k|k} &= \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k \tilde{\mathbf{y}}_k \\ \mathbf{P}_{k|k} &= (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k) \mathbf{P}_{k|k-1} \end{aligned}$$

The state transition and observation matrices are the following Jacobians:

$$\mathbf{F}_{k-1} = \left. \frac{\partial f}{\partial \mathbf{x}} \right|_{\hat{\mathbf{x}}_{k-1|k-1}} \qquad \mathbf{H}_k = \left. \frac{\partial h}{\partial \mathbf{x}} \right|_{\hat{\mathbf{x}}_{k|k-1}}$$

where f is the state transition function and h is the observation function. Since f and h are non-linear in general, we perform a linearization about the current state to produce $\mathbf{F}$ and $\mathbf{H}$ in order to be able to use the Kalman filtering equations.

2.7 Team Coordination

Coordination among robots on the same team is crucial for RoboCup games. We have recently started working on our coordination method, which offers mechanisms for positioning on the field and role assignment. The formation generator component is responsible for generating a set of positions on the field, based on the global estimated ball position as provided by our shared world model. Specifically, the formation type (offensive, defensive) is determined dynamically, depending on which half of the field the ball lies in, and a number of candidate positions that represent special roles (supporter, attacker, defender, ...) is generated. Then, the role assignment component uses a utility function to evaluate mappings of players (robots) to positions (roles) and decide which mapping is

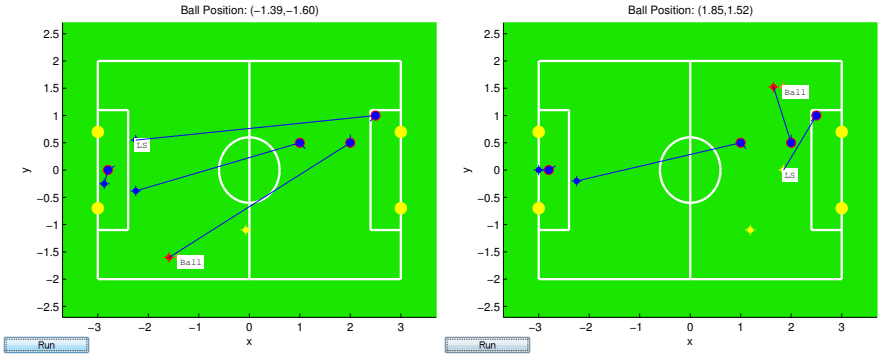

Fig. 5. Team formation and role assignment for defense (left) and offense (right). Current robot positions are shown in blue circles, formation positions are shown in blue circles with crosses, and role assignment is shown through the connecting lines.

best for the current game situation. Note that the candidate positions/roles may be more than the robots, therefore the number of possible mappings may be extremely large. Role assignment aims essentially looks for the set of positions with greatest importance with respect to the current situation. The utility function combines a variety of features: distance of a robot from a candidate position, desired orientation at a candidate position, probability of collision with other robots given a target position, a global field scoring function, and a measure of robot health/stability (hinted by the number of times a specific robot has fallen or has been penalized). The selected mapping is available to all robots and each robot is responsible for executing a behavior that implements the assigned role. Figure 5 shows two examples of team formation and role assignment.

2.8 Behavior Specification

The behavior of our team is specified using the Agent Systems Engineering Methodology (ASEME) as it was recently adapted to address the challenges of robotic behavior specification [6]. ASEME suggests a strict hierarchical decomposition of the desired robot behavior into smaller activities until a level of provided base activities is met. Each design step is supported by a formal model and the transition from one design phase to another is assisted by automatic model transformations. Briefly, the process begins with the specification of a set of liveness formulas (analysis phase) which are converted to an initial statechart model; the statechart is subsequently enriched (design phase) and is converted to source code. Our own Computer-Aided System Engineering (CASE) tool, Kouretes Statechart Editor (KSE) [7,8], enables the developer to design the statechart model either graphically, from scratch, or by first writing the liveness formulas and transforming them automatically to an abstract statechart. KSE also supports the graphical editing of the statechart model, the addition of transition expressions, the validation of the final statechart model indicating which

elements are incorrect (if any), and the automatic source code generation for compilation and execution on the robot. The final statechart model specifies the intra- and the inter-agent control for achieving the desired team behavior by accessing directly the functionality provided by our base activities: RobotController, Vision, Localization, ObstacleAvoidance, MotionController, HeadHandler, Sensors, LedHandler, Communication.

2.9 Monitoring and Debugging

To facilitate software development, we have developed a monitoring and debugging tool, called KMonitor [1], which monitors messages sent by the robots over the network using the UDP packet multicasting protocol and displays the transmitted information in a user-friendly way. The graphical user interface is composed by various tabs offering different monitoring views for inspection of vision, localization, obstacle avoidance, and decision making. The user can select one or more of the detected active robots to monitor and check one or more desired graphical features to visualize. KMonitor integrates the required functionality and views under a single intuitive graphical user interface, facilitating the user in switching quickly between different views in order to monitor different aspects of the robot software. Finally, KMonitor is easily extensible and fully parameterizable to accommodate future needs.

References

1. Karamitrou, M.: KMonitor: Global and Local State Visualization and Monitoring for the Robocup SPL League. Diploma thesis, Technical University of Crete, Greece (2012), available at: www.intelligence.tuc.gr/lib/downloadfile.php?id=432
2. Kofinas, N.: Forward and Inverse Kinematics for the NAO Humanoid Robot. Diploma thesis, Technical University of Crete, Greece (2012), available at: www.intelligence.tuc.gr/lib/downloadfile.php?id=430
3. Kofinas, N., Orfanoudakis, E., Lagoudakis, M.G.: Complete analytical inverse kinematics for NAO. In: Proceedings of the 13th International Conference on Autonomous Robot Systems and Competitions (ROBOTICA) (2013)
4. Orfanoudakis, E.: Reliable Object Recognition for the RoboCup Domain. Diploma thesis, Technical University of Crete, Greece (2011)
5. Paraschos, A.: Monas: A Flexible Software Architecture for Robotic Agents. Diploma thesis, Technical University of Crete, Greece (2010)
6. Paraschos, A., Spanoudakis, N.I., Lagoudakis, M.G.: Model-driven behavior specification for robotic teams. In: Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS) (2012)
7. Topalidou-Kyniazopoulou, A.: A CASE Tool for Robot-Team Behavior-Control Development. Diploma thesis, Technical University of Crete, Greece (2012)
8. Topalidou-Kyniazopoulou, A., Spanoudakis, N.I., Lagoudakis, M.G.: A CASE tool for robot behavior development. In: Proceedings of the 16th RoboCup International Symposium (2012)
9. Vazaios, E.: Narukom: A Distributed, Cross-Platform, Transparent Communication Framework. Diploma thesis, Technical University of Crete, Greece (2010), available at: www.intelligence.tuc.gr/lib/downloadfile.php?id=352