M. Knopp, C. Aykın, J. Feldmaier and H. Shen, "Formation control using GQ(λ) reinforcement learning", 2017 26th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN), Lisbon, 2017, pp. 1043-1048. DOI: 10.1109/ROMAN.2017.8172432

Formation Control using GQ(λ) Reinforcement Learning

Martin Knopp[1], Can Aykın, Johannes Feldmaier, and Hao Shen

***Abstract*— Formation control is an important subtask for autonomous robots. From flying drones to swarm robotics, many applications need their agents to control their group behavior. Especially when moving autonomously in human-robot teams, motion and formation control of a group of agents is a critical and challenging task.**

In this work, we propose a method of applying the GQ(λ) reinforcement learning algorithm to a leader-follower formation control scenario on the e-puck robot platform.

In order to allow control via classical reinforcement learning, we present how we modeled a formation control problem as a Markov decision making process. This allows us to use the Greedy-GQ(λ) algorithm for learning a leader-follower control law. The applicability and performance of this control approach is investigated in simulation as well as on real robots.

In both experiments, the followers are able to move behind the leader. Additionally, the algorithm improves the smoothness of the follower's path online, which is beneficial in the context of human-robot interaction.

I. Introduction

A special class of multi-robot formation control is the motion control of a group of robots led by a human leader. For example, the formation control of robotic wheelchairs which are accompanied by human caregivers [1], or the multi-robot formation control in shared rescue missions [2].

A basic problem when considering multi-robot scenarios is how to maintain a specific shape while moving around. This problem has been addressed in the literature by model based (e.g. [3], [4], [5]), potential-field based [6], graph based [7], or combined approaches [8]. A lot of these approaches yield non-linear control laws which are difficult to implement on low-cost hardware.

Reinforcement learning, on the other hand, is based on the idea of an agent trying out different actions and learn from the outcome which behavior is optimal for a specific situation. The designer of a control task therefore has to provide feedback in the form of a scalar reward function which provides information about the performance of the last step the agent decided to take [9]. Most reinforcement learning algorithms can be implemented on embedded hardware platforms as they rely heavily on linear functions which are of lower computational complexity compared to the aforementioned non-linear control laws.

Reinforcement learning could be a useful approach for formation control. Either on the low level side—like keeping distance and angle based on direct sensor data—or on the high level side—like the shape of the formation or leader selection. The foundation for our idea of applying reinforcement learning to formation control is laid out in this work by the investigation of the applicability on the low level side: a simple leader-follower formation which uses proximity sensors as input and motor speeds as actions. We employ the GQ(λ) algorithm [10] which is a state of the art reinforcement learning algorithm.

For our experiments, we use e-puck robots, small educational, open-source tabletop robots. They provide a standard Linux environment and are therefore particularly well suited for first experiments with formation control algorithms. A simplified model of them is also included in the V-REP simulator by Coppelia Robotics [11], which we used for initial simulations without the uncertainties of a real environment. Additionally, for a better reproducibility, we also replaced the human leader with an e-puck robot. In order to transfer our experiments to the real world, the human leader should be equipped, for example with an active infra-red beacon, so that the distance sensors of the e-pucks can detect and identify the leader at a feasible range. Even though e-puck robots are quite tiny and somewhat fragile, the available sensor data and possible actions are applicable to most other robotic platforms, or they could be outfitted accordingly for little cost. Our requirements for the leader are that it moves at a speed the followers are able to keep pace with and that it is detectable by our agents. Depending on the needs of the actual scenario this can either be ensured by the human leader watching out for its robotic followers or by using adequate sensors and motors on the robots, so that loosing their leader is unlikely.

The rest of this paper continues with an brief introduction to reinforcement learning, the robots used and simulation software and the actual formation task in our investigation. We describe the design considerations necessary to apply reinforcement learning to the formation task and requirements for a usable implementation. We conclude with our results within the simulation and on the real robots.

A. Reinforcement Learning

Reinforcement learning is a machine learning technique that, in its simplest form, is based on the assumption that an agent (in our case the e-puck robot) acts within a Markovian environment. This means that it is in some state s out of a set of states S and can proceed to another state by choosing an action a from a set of actions A. It chooses its action according to some behavior policy $\pi_b(a|s)$. For each of these state-action-state-transitions, the agent observes a reward r. The agent uses this reward to update the value function estimation V(s) or Q(s,a) (depending on whether we are interested only in states or state-actions) to better represent

[1]Corresponding author: Martin.Knopp@tum.de

All authors are with the Department of Electrical and Computer Engineering, Technical University of Munich, 80333 Munich, Germany

the real value of the state or state-action pair. The value function is then used to find a better policy π_b which allows the agent to maximize the accumulative reward.

The value function estimation asymptotically converges to its true values and therefore, also, the policy becomes optimal.

Many of the properties of classical reinforcement learning are rather impractical for the use on real world problems. Commonly, real world problems are non-episodic so that the theoretically infinite number of iterations have to be handled properly, and additionally the state as well as the action space can have a large number of elements. Fortunately, a good enough solution is feasible for almost all practical problems by discretizing the state and action space, and by determining suitable terminal states. One can therefore divide the state and action space using techniques like *tile coding* and approximate the value function to greatly reduce the computational complexity of the algorithms [12]. For this work on leader-follower formation control, we combined linear function approximation with single binary feature tile coding.

To be more concrete, we used the GQ(λ) algorithm from Maei and Sutton [10] and combined it with an ϵ-greedy behavior policy to balance between exploration and exploitation. This approach is described by Maei et al. in their paper about off-policy learning with function approximation [13] and also in the recent technical report by White and Sutton about the implementation of GQ(λ) [14].

B. The e-puck Robot

The e-puck robot is a small table-top, open-source robot (approx. 10 cm diameter) which was developed at the *École polytechnique fédérale de Lausanne (EPFL) [15]*. Its design goal was to provide a comparatively cheap research and education robot. As a consequence, its proximity sensors are based on infrared diodes and phototransistors. This limits their maximum range to approximately 15 cm, depending on the infrared reflectance of the surface they are trying to detect. This is important for this work, because a-priori knowledge about the leader's position is required for setting up the tile coder. Three sensors, which are similar to these proximity sensors, are integrated into the base of the robot and allow it to distinguish bright and dark floor. They can be used to detect edges when driving forward or to follow a line.

For moving around, the e-puck uses two stepper motors which drive its two wheels. The mechanical setup of them is symmetric and therefore allow the robot to turn itself on the spot by setting the motor speeds to opposing values.

Due to the open-source nature of the e-puck, a number of hardware extensions are available. One of it is the Gumstix Overo Computer On Module (COM) extension, which allows one to use a ARM-based embedded computer for controlling the robot. The COM module runs an embedded Linux and is able to connect to a wireless LAN. We are using this embedded Linux together with an existing Python API to implement our learning algorithms directly on the e-puck. In this way, we have a rapid prototyping platform that allows us to quickly test different learning approaches without having to care about the low-level hardware control. On the other hand this also introduces some processing delays due to the serial communication of the Overo COM with the e-puck base, which are taken into account by our algorithm.

C. The V-REP Simulator

V-REP is a versatile open-source robot simulation framework. What makes it extremely useful for our work is its remote API which allows access to almost all simulation parameters via an outside program. Our particular use case is the control of the simulated robots via external scripts. We implemented a Python wrapper interface that emulates the Overo COM API for interfacing with V-REP. In that way, only very small changes are necessary to port software from simulation to real hardware.

D. Robot Task Description

To investigate the feasibility of Greedy-GQ(λ) for formation control on e-puck robots, we specified a leader-follower scenario: the first robot is configured statically to follow a line on the ground. This function is readily available in V-REP and also on the real robot. In our experiments, this line is jittered and shall represent data which is comparable to the output of a SLAM based path planner that was used to plan a path in a dynamic environment while tracking a human leader.

The following robot(s) observe their respective leader via the two front facing infrared distance sensors. The sensors on the side of the robot are too sparsely distributed to allow a sufficient precise tracking of other robots. This problem might be solvable by using the optional range and bearing extension, but this investigation was outside of the scope of this work. Therefore, more advanced formations like side-by-side following have been postponed for future work.

II. Theoretical Considerations

In this section, the theoretical ideas, limitations and decisions that are necessary for our proposed formation control setting are stated.

A. Design of State and Action Space

The dimensions of the state and action space are rather straight-forward. Our robots have two motors and the speed of each one can be set independently. Therefore the action space is two-dimensional.

The state space is pretty similar as we are considering a two-dimensional path by the leader which the robot should follow. Due to the mechanical size of a robot and the maximum sensor range, we can quickly decide on an optimal leader-follower distance. It is in the middle between the maximum and minimum allowed distance. The deviation from this optimal distance, the distance error, is the first dimension of our state space. The optimal region for the follower to detect the leader is therefore a circle. To build an actual formation, we need to specify the position on

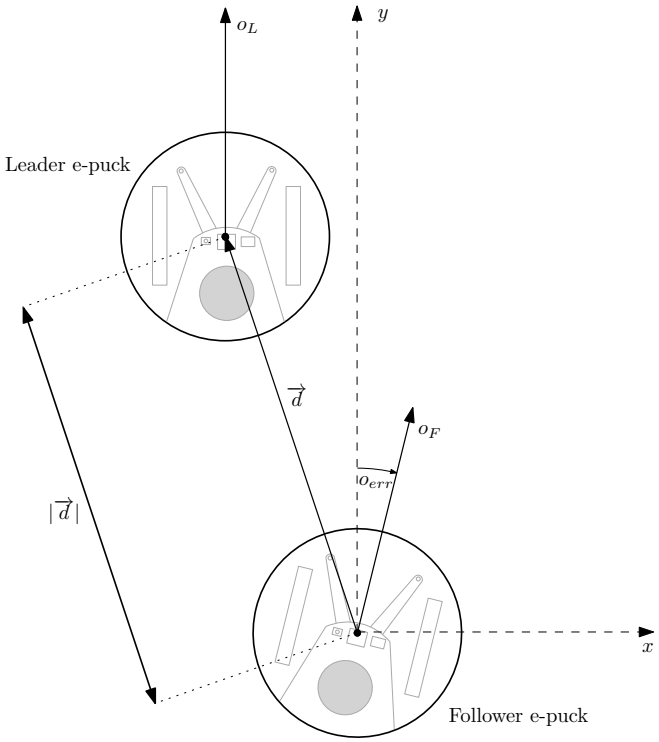

Fig. 1. Illustration of the error measures $|\vec{d}|$ and o_{err} for our leader-follower scenario.

this circle. By considering this, it is reasonable to use the angular error as the second dimension. The geometrical considerations of these error measures are shown in Figure 1.

All four dimensions (states and actions) can be considered to be continuous (actually, they are based on discrete sensor and actor values, but from an implementation point of view, there is no difference). Therefore featurization is required, for which we chose static binary single feature tile coding. For the action space, uniform linear tile coding was applied. For the state space, linear tile coding proved to be feasible during simulations in V-REP. As the (raw) values of the proximity sensors on the real e-puck show an exponential scaling, logarithmic tiling is of much better use here. It also yielded better results when used for the angular orientation error.

The number of tiles is a tradeoff between precision and computational complexity. As the (first) leader moves at a constant speed, the followers only have a limited time for computing before the leader moves out of range. The simulations in V-REP were done on a standard PC, and the tiling of the state space was chosen to be 9×9. This tiling is illustrated in Figure 2. The action space had a 7×7 tiling. For learning on the actual robot, these sizes had to be reduced to 5×5 and 3×3, respectively. This reduction is necessary due to the much lesser processing power of the Overo Computer On Module used on the e-puck.

B. Absolute and Relative Position Calculation

To calculate the distance and angular error, two general approaches are possible: calculation via absolute coordinates

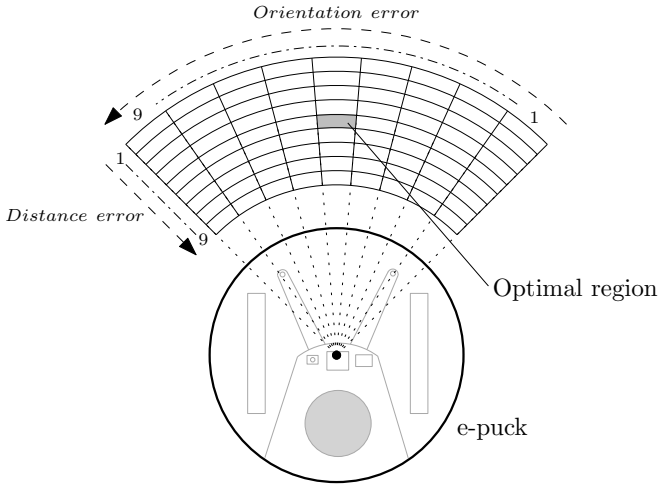

Fig. 2. The 9×9 tile coding used for the representation of the state space when error calculations are based on absolute coordinates.

and direct derivation from the follower's sensors. The former one is also available on the real robots by using a global camera tracking system which is implemented with ArUco markers [16] attached to the robots. This method of error calculation is very useful for tuning and developing the learning algorithms as the symmetry, range, and noise of the proximity sensors can be neglected. On the other hand, assuming a global observer to be available in any real scenario is not realistic and justifies investigation of the relative calculation of the error measures.

C. Resetting the Agent

As we are considering our task to be episodic, we need some kind of reset algorithm in case our robot loses its leader and therefore gets into a terminal state. One early approach was to reset the whole simulation into its starting state. This approach has the major drawback of requiring human interaction in the real environment. It also leads to some overtraining as the agents iterate over the same section of the path over and over again.

Therefore, our proposed reset function works by reaching a terminal state while still being within the robots sensor range (this requirement is not needed within V-REP and when using the camera observer as absolute coordinates are available at any time). The follower then turns towards the leader and moves forward at maximum speed until it reaches its optimal working range again and can restart learning in a new episode.

D. Performance Improvements

In general, the used process of reinforcement learning is a cycle that starts with the observation of a certain state s_t, selects an action a_t, waits until the action is completed, observes the resulting state s_{t+1}, updates its value function estimation accordingly and starts the next cycle. As the update of the value function estimation is the most computationally intensive step, we introduced the idea of an update buffer. It caches the observations of s_{t+1} and the

reward and delays the value function estimation to the next time the robot waits for the execution of its chosen action. This has of course some consequences, as it introduces an aspect similar to offline learning to an online learning process, and also requires an additional calculation cycle after reaching a terminal state to correctly update the value function estimation. It must be noted that these changes do not change the applicability of the algorithm, but just have to be taken into account. The processing speed-up also greatly improves learning efficiency as there is no blind time in which the leader continues to move away from the follower.

III. Experiment and Results

This section describes the results of the aforementioned experiments in the simulation scenario as well as on the real robot. The main goal of using a learning algorithm for leader follower control was achieved.

Additionally we have observed that the path of the followers is a smoothed variant of the leader's path initial line-following leader. The artificial juddering behavior resulting from disturbances and inaccuracies of the SLAM process is therefore reduced. In this way, the followers behave to what human observers would expect, and consider as "natural". This should improve the acceptance in non-expert interactions with humans who might perceive juddering robot actions as malfunctions.

A. Simulation

As mentioned in Section I-D, our scenario for reinforcement learning based leader-follower formation control implements the leader as a simple static line follower with no learning at all. In that way we make the leader move arbitrarily without requiring advanced path planning and map recognition, but do not lose generality of the results.

We allow up to four followers which try to learn how to follow the robot directly in front of them. This leads to a worm-like line formation, a screenshot of such a formation in the V-REP simulator is shown in Figure 3.

The most interesting observation is that the robots start to stabilize their behavior from the first follower towards the end of the line formation. This allows the conclusion that a constant, stable movement is beneficial for our learning approach, as the followers show a very juddering movement during the learning phase which only smooths to a certain degree after convergence.

Depending on the desired formation, it is also possible to transfer the learned parameters to agents which shall show the same behavior, i.e. if all robots should follow the robot directly in front of them, it is sufficient to learn the control parameters on the first follower and run a greedy policy on all other robots which exploit the learned state-action values from the first one.

Most of the time during our simulations with a total number of about 1.3 million steps, the followers stayed very close to the optimal region (98.7% are within a 3×3 area on the right side). The detailed statistic of where the followers detected the leader is shown in Figure 4. We suppose that

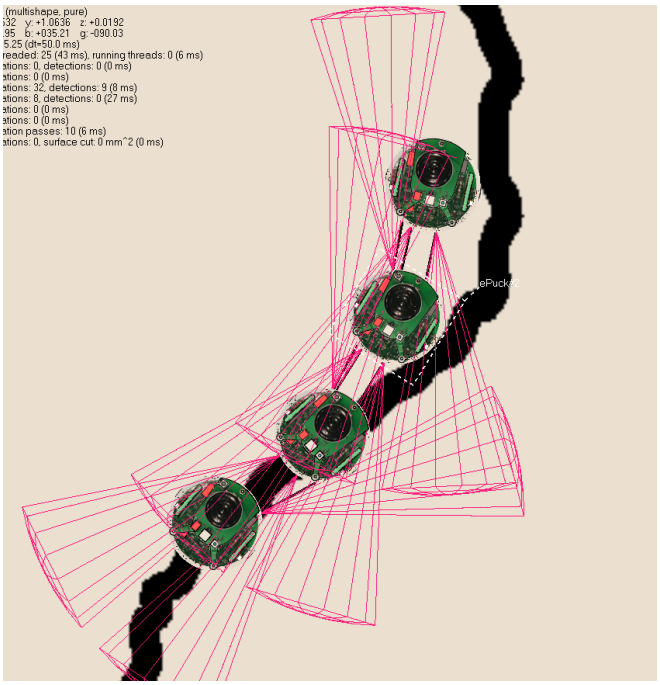

Fig. 3. Simple four agent line formation in V-REP: the first leader can be found in the lower left corner. The red cones are showing active proximity sensors.

the shift to the right side is caused by the concrete scenario, as the overall path is a closed circular loop.

B. Real World Scenario

Evaluating Greedy-GQ(λ) on real e-pucks is not an easy task. As mentioned before, the processing power of the Overo COM is much lower than that of a normal PC, so the number of calculations has to be heavily reduced. To achieve that, we lowered the number of tiles as described in the theory section. This reduces the smoothness, and also the learning efficiency, by a certain amount but is absolutely necessary to ensure that the follower can keep up with the

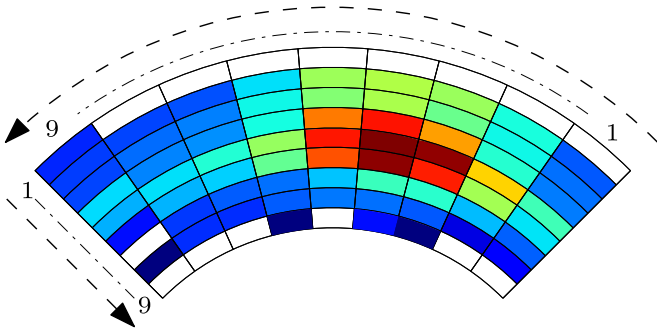

Fig. 4. Distribution of visited states at around 1.3 million steps. In 74% of all cases, the leader was detected in the dark red area right to our optimal region. It stayed within the orange/red 3×3 area on the right side of the optimal region in 98.7% of the time. To make the rarely visited states discernible, a logarithmic scale is used: red and orange values are in the range of 30%–1%, green values are at around 0.1%, and blue values are another magnitude below that. The non-colored states were never visited in this experiment.

leader.

The proximity sensors of the e-puck are a much bigger problem. The mentioned maximum range of 15 cm can be considered as an optimal result under perfect conditions. All of the sensors are susceptible to environmental lighting and none of the sensors are matched. A basic query of them returns their raw A/D conversion values, which are practically a measure of how much infrared light is coming into the phototransistor. As each of the sensors is a little bit different, even with no objects around the robot and homogeneous lighting conditions, practically none of the values are the same. This is compensated for by offset calibration which makes the range functions return 'zero' under the described conditions. Therefore, it is also important to avoid strong (infrared) light coming from one direction as this would render the robots unable to detect meaningful range information as soon as they change their angle towards the light source. As the returned values depend on the incoming infrared light, they also depend on how good or bad the illuminated surface is reflecting back the light. The standard robot housing has a particularly bad reflectance and is only detected when almost touching the other robot. We solved this problem by using cut out coffee paper cups to increase the sensor range to usable values. This modification is shown in Figure 5 with the leader and its coffee cup on the right side and the follower on the left side. A different approach would be to attach retro-reflective tape to the e-pucks. If more spread out formations are desired, this might as well become necessary. The resulting absolute sensor readout values are not important as the learning will adapt to them, these modifications are required to get any difference in the readouts when approaching another robot at all.

The raw A/D conversion of the incoming light also leads to an exponential dependency between the distance of the leader-follower pair and the resulting values. The logarithmic tile coding of the distance error in the real-robot scenario takes this non-linearity into account.

Despite of all these sensor and computational issues, the learning algorithm itself showed its capability to learn the right parameters for staying within the desired line formation also on imperfect hardware. Nevertheless, it takes much more time until it reaches a stable state and the robots move a lot less smoothly than in the simulation with its more precise sensors and larger state and action spaces. However, due to the transferability of learned parameters between different robots, this issue can be moderated as follows.

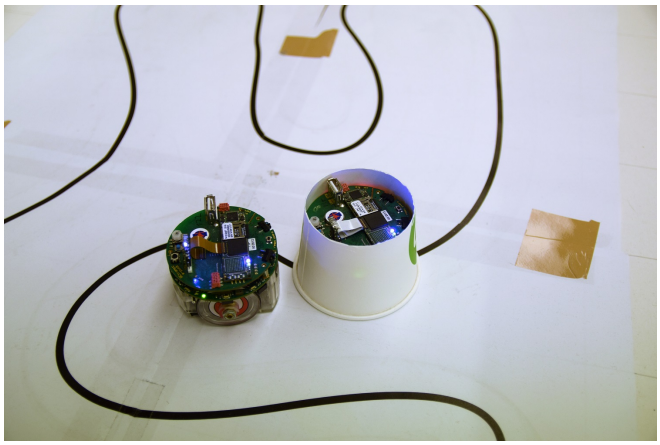

Fig. 5. Leader-follower formation on actual hardware. The leader is placed into a white cardboard coffee cup to improve infrared reflectance and thus proximity sensor range.

In settings with a global tracker, absolute coordinates are available. They are both handled by a similar linear tile coding in the simulation and on the real hardware, which allows the transfer of the value function estimation. We looked into this approach only briefly, as this requires the same state-action space configuration as in the simulator. As noted before, learning with these large spaces is not possible, so the real robots could only exploit the learned behavior from the simulation. Nevertheless, this lead to promising results considering the smoothness of movements and the stability of the whole formation. We did not expect such good results due to the variances of the detected positions in the camera images.

These results show that improving the proximity sensor representation might help to create a scenario in which formation driving could be trained in a simulated scenario in order to skip the time consuming initial training in reality. Two approaches seem reasonable: making the simulated sensors to behave like the real ones or preprocessing the sensor data to return the actual detected distance. The former approach could be desirable as it might also allow simulating the use of the proximity sensors as a mean of infrared communication. On the other hand, this might also be out of the scope of most simulations and the return of an actual detected distance would also be helpful for further debugging directly on the robot.

IV. Conclusion

Our findings show that Greedy-GQ(λ) can be used to let robots learn how to keep a line-like formation. It also shows that the required learning parameters can be adapted from a simulation, even though the simulation still differs significantly from the actual scenario in reality. Compared to classical control approaches, our results are quite similar and comparable, with the drawback that our initial training phase takes some time, which is not necessary in the classical case. Despite of that, the emerging "natural" behavior encourages us to look deeper into this direction. Therefore we are adapting our algorithms to larger robots which are able to follow humans and interact with them in a leader follower scenario. We are also preparing a study to precisely evaluate the increased acceptance of the robot's behavior by non-experts to get a deeper insight into our intuitive assumptions.

Generally, by exploiting reinforcement learning to solve a classical control problem, we have introduced an additional solution to the leader-follower task. In order to support future developments in the domain of multi-robot and human interaction scenarios, we tried to emphasize the practical difficulties in implementing and transferring Greedy-GQ(λ)

learning to real hardware. We believe that reinforcement learning based algorithms are useful in non-deterministic and dynamic environments – such as real world scenarios.

Reinforcement learning can also be extended to higher level tasks and other existing ideas in formation control (e.g. rigid graphs) by changing the reward function to include these requirements, or by layering multiple learning algorithms on top of each other. This flexibility and the observed behavior characteristics encourage us to further pursue reinforcement learning in the area of human-robot formation cooperation.

Our long term goal would be a framework that allows us to provide basic instructions like a desired shape, and have a self organizing group of robots that can adapt to these instructions and the requirements of the environment. Such a framework would have to include features like formation splitting to get around obstacles or dynamic leader assignment if an abrupt change of direction becomes necessary. These functions would be essential for the dynamic control of support robots in rescue missions.

REFERENCES

[1] M. Arai, Y. Sato, R. Suzuki, Y. Kobayashi, Y. Kuno, S. Miyazawa, M. Fukushima, K. Yamazaki, and A. Yamazaki, "Robotic wheelchair moving with multiple companions," in *The 23rd IEEE International Symposium on Robot and Human Interactive Communication*, 2014, pp. 513–518.

[2] J. Saez-Pons, L. Alboul, J. Penders, and L. Nomdedeu, "Multi-robot team formation control in the GUARDIANS project," *Industrial Robot: An International Journal*, vol. 37, no. 4, pp. 372–383, 2010.

[3] M. Uchiyama and P. Dauchez, "A symmetric hybrid position/force control scheme for the coordination of two robots," in *Proc. 1988 IEEE International Conference on Robotics and Automation*, vol. 1, Philadelphia, PA, Apr. 1988, pp. 350–356.

[4] W. B. Dunbar and R. M. Murray, "Model predictive control of coordinated multi-vehicle formations," in *Proc. 41st IEEE Conference on Decision and Control*, Las Vegas, NV, Dec. 2002, pp. 4631–4636.

[5] ——, "Distributed receding horizon control for multi-vehicle formation stabilization," *Automatica*, vol. 42, no. 4, pp. 549–558, Apr. 2006.

[6] S. Monteiro and E. Bicho, "A dynamical systems approach to behavior-based formation control," in *Proc. 2002 IEEE International Conference on Robotics and Automation (ICRA '02)*, vol. 3, Washington, DC, May 2002, pp. 2606–2611.

[7] C. Yu, B. D. O. Anderson, S. Dasgupta, and B. Fidan, "Control of minimally persistent formations in the plane," *SIAM Journal on Control and Optimization*, vol. 48, no. 1, pp. 206–233, Feb. 2009.

[8] F. Dorfler and B. Francis, "Geometric analysis of the formation problem for autonomous robots," *IEEE Trans. Automat. Contr.*, vol. 55, no. 10, pp. 2379–2384, Oct. 2010.

[9] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1238–1274, Sept. 2013.

[10] H. R. Maei and R. S. Sutton, "GQ(λ): A general gradient algorithm for temporal-difference prediction learning with eligibility traces," in *Proc. Third Conference on Artificial General Intelligence (AGI 2010)*, Lugano, Switzerland, Mar. 2010, pp. 91–96.

[11] E. Rohmer, S. P. N. Singh, and M. Freese, "V-REP: a versatile and scalable robot simulation framework," in *Proc. IEEE International Conference on Intelligent Robots and Systems (IROS 2013)*, Tokyo, Japan, Nov. 2013, pp. 1321–1326.

[12] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT Press, 1998.

[13] H. R. Maei, C. Szepesvari, S. Bhatnagar, and R. S. Sutton, "Toward off-policy learning control with function approximation," in *Proc. 27th International Conference on Machine Learning (ICML 2010)*, Haifa, Israel, June 2010, pp. 719–726.

[14] A. White and R. S. Sutton, "GQ(λ) Quick Reference Guide," University of Alberta, Tech. Rep., July 2014.

[15] F. Mondada, M. Bonani, X. Raemy, J. Pugh, C. Cianci, A. Klaptocz, S. Magnenat, J.-C. Zufferey, D. Floreano, and A. Martinoli, "The e-puck, a robot designed for education in engineering," in *Proc. 9th Conference on Autonomous Robot Systems and Competitions (Robotica 2009)*, Castelo Branco, Portugal, May 2009, pp. 59–65.

[16] S. Garrido-Jurado, R. Muñoz-Salinas, F. Madrid-Cuevas, and M. Marín-Jiménez, "Automatic generation and detection of highly reliable fiducial markers under occlusion," *Pattern Recognition*, vol. 47, no. 6, pp. 2280–2292, Jan. 2014.