Human-Machine Interface for Multi-agent Systems Management Using the Descriptor Function Framework

Giovanni Franzini(✉), Stefano Aringhieri, Tommaso Fabbri, Matteo Razzanelli, Lorenzo Pollini, and Mario Innocenti

Automation, Robotics and Autonomous Systems Laboratory,
Department of Information Engineering, University of Pisa,
Largo Lucio Lazzarino 1, 56122 Pisa, Italy
{giovanni.franzini,stefano.aringhieri,tommaso.fabbri}@for.unipi.it,
matteo.razzanelli@ing.unipi.it,
{lorenzo.pollini,mario.innocenti}@unipi.it

Abstract. Human-machine interfaces for command and control of teams of autonomous agents is an enabling technology for the development of reliable multi-agent systems. Tools for proper modelling of these systems are sought in order to ease the creation of efficient interface that allow a single operator to control several agents, as well as monitor the execution state of the tasks the team is demanded to accomplish. If humans are present in the environment, the agents must sense their presence and collaborate with them toward the mission accomplishment. In this context, the descriptor function framework is a versatile tool that allows the human integration at two levels: the development of human-machine interfaces and the achievement of human-machine teaming. In this paper, we show how such results can be obtained and we propose a possible architecture for the framework implementation.

Keywords: Multi-agent system · Autonomous system · Cooperative system · Human-machine interaction · Human-autonomous system teaming

1 Introduction

Control of multi-agent systems is a challenging task that has drawn the attention of the scientific community over the last two decades, thanks to the growing interest in their potential applications in the airborne, maritime and ground domains. Teams of autonomous agents may for instance support humans in all those missions and activities that require the systematic coverage or monitoring of wide areas. Bearing in mind this type of scenarios, humans and autonomous agents should cooperate and coordinate their actions, so that the available resources are efficiently used and the mission is successfully accomplished. Hence, the study of frameworks for multi-agent system modelling and control is of primary interest, in order to achieve human-machine teaming.

J. Hodicky (Ed.): MESAS 2016, LNCS 9991, pp. 25–39, 2016.
DOI: 10.1007/978-3-319-47605-6_3

The *descriptor function framework*, developed originally at the University of Pisa and at the US Air Force Munitions Directorate since 2010 [1], is a mathematical modelling instrument that provides a simple yet effective way for describing agents capabilities and coordinate their deployment in the field. Conversely to most of the solutions proposed in literature, the descriptor function formalism can handle teams composed by *heterogeneous agents*, that is agents with different capabilities with respect to the demanded task. By means of the same formalism, it is possible to describe the desired team deployment, or, in other words, the way the team must execute the task. The benefit is twofold. First, using the same mathematical tool for both agents modelling and task description, the agents perceive the execution status of the task and the deployment of their team-mates, so that they coordinate their actions. Second, different types of tasks can be modelled within the descriptor function framework without the need of ad hoc solutions. For example, in [2] the deployment is achieved using a geometric approach; in [3] the area coverage problem was addressed using Voronoi cell decomposition of the environment; in [4] the target assignment task was solved by means of heuristic algorithms for deployment within a temporal deadline. Within the descriptor function framework, all these problems can be solved without changing the agents control strategy. The desired deployment needs only to be translated into an appropriate descriptor function.

Over the years, the descriptor function framework has proven to be a versatile instrument for the control of team of autonomous agents. However, the authors never considered the presence of humans in the field, working on the same task assigned to the agents. The aim of this paper is to show how the descriptor function framework can be extended to consider this possibility. In particular, as for the autonomous agents, we model humans by means of descriptor functions. In this way, the autonomous agents can perceive the humans and how they are contributing to the task execution. As a result, human-machine cooperation is achieved.

In addition, we also show a possible architecture for the development of a human-machine interface for command and control of the autonomous agents team. Through this interface, an operator can coordinate the agents deployed in the field, as well as monitor the execution state of the tasks assigned. Exploiting the descriptor function formalism, we also introduce the possibility of direct steering of the deployed agents. More specifically, a single operator can steer the whole team, moving all the agents at the same time. Such a capability may be useful, for example, in emergency situation where it is necessary to have a direct control of the agents movements, or can be used to move the team toward another region of interest.

The paper is organized as follows. In Sect. 2 we provide an overview of the descriptor function framework. For the sake of convenience, in the discussion we do not focus our attention on the mathematics behind the framework. The interested reader shall refer to the references provided throughout the discussion for further information about the mathematical details. In Sect. 3 we discuss how humans and their presence in the field can be introduced into the description

framework, in order to achieve the human-machine cooperation. In Sect. 4 we present a possible software implementation for the framework, with reference to the *Robot Operative System* (ROS) [5] developing environment. Section 5 concludes the paper.

2 The Descriptor Function Framework

Descriptor functions are a mathematical model for describing the distribution of resources over the environment. As a matter of fact, an autonomous agent is a carrier of resources of a certain type. The task that a team of autonomous agents has to accomplish defines how these resources must be deployed in the environment. Therefore, descriptor functions are a unifying tool to model both the agents capabilities and the task we demand the team to accomplish. The use of such a modelling tool simplifies the management of large groups of agents and provide to the users a powerful yet intuitive language for handling the interactions with teams of autonomous agents.

The basis of the descriptor function framework can be found in the original works by Niccolini et al. [1,6]. The framework has been continuously improved during the years, extending its capabilities and strengthening its theoretical foundations [7,8].

2.1 Framework Overview

The descriptor function framework is based on three main definitions:

- **Agent**: the entity that operates in the environment in order to achieve the mission objectives.
- **Task**: coordination in space of the agents working on the same objective.
- **Mission**: coordination in space and time of the tasks.

Generally, mission objectives are translated into one or more tasks that the team must complete toward the mission accomplishment. Hence, a task indicates to the agents how they should deploy in the field in order to fulfil a given mission sub-objective.

Broadly speaking, teams may be composed by agents with different capabilities of executing the mission's tasks, i.e. teams may be composed by *heterogeneous* agents. In order to cope with heterogeneous teams, agent's characteristics are quantified by means of the *agent descriptor functions* (ADF). Given a point of the environment, the ADF describes the agent capability of executing a given task at that point. Hence, the ADF is a function of the agent relative position from that point. Mathematically, we define the ADF of an agent i for the task k as follows,

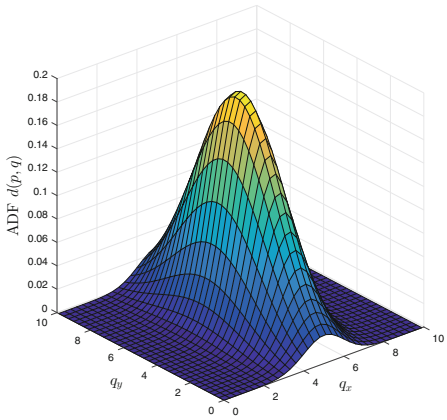
$$d_i^k(\boldsymbol{p}_i, \boldsymbol{q}) \colon \mathcal{P} \times \mathcal{Q} \to \mathbb{R}^+ \tag{1}$$

where $\mathbb{R}^+ = \{x \in \mathbb{R} \colon x \geq 0\}$, $\mathcal{Q} \subseteq \mathbb{R}^n$, with $n = \{2, 3\}$, is the domain of the environment and $\boldsymbol{p}_i \in \mathcal{P}$, with $\mathcal{P} \subseteq \mathbb{R}^m$, is the agent state vector (typically its

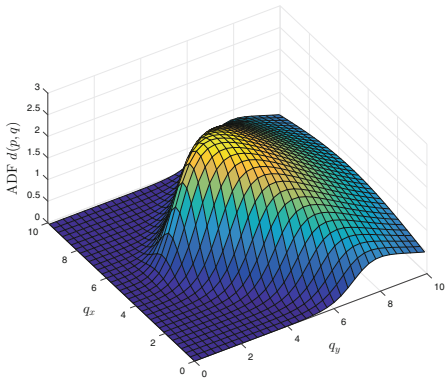
position and orientation). For each agent, we define a number of ADFs equal to the tasks composing the missions. If the agent is unable of executing a task, the relative ADF is zero all over the environment.

Two examples of ADFs are shown in Fig. 1. Gaussian functions, like the one shown in Fig. 1(a) and (b), provide a simple tool for describing various type of omnidirectional sensors. More complex functions can be used to model sensor with limited field-of-view, see e.g. Fig. 1(c) and (d). The level curves indicates the sensor's intensity and may be a measure of the quality of the information acquired from that point. Extensive details about ADF design and their mathematical formalization can be found in [7].

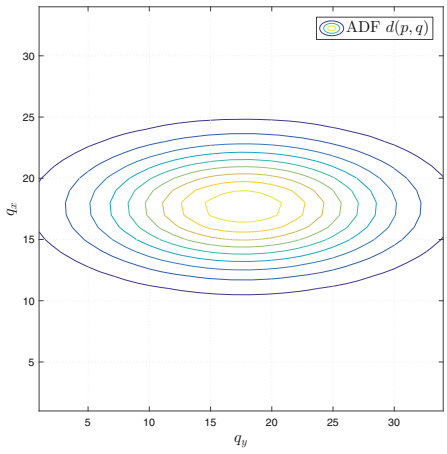
As mentioned at the beginning of this section, within the descriptor function framework agents are viewed as carriers of resources. In order to execute a task,



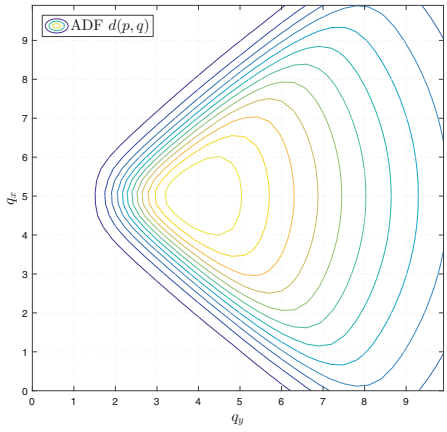
(a) Gaussian ADF.



(b) ADF with field-of-view.



(c) Gaussian ADF: level curves.



(d) ADF with field-of-view: level curves.

Fig. 1. Examples of ADFs.

we must define how these resources should be deployed in the environment. To this end, we express the desired deployment using the *task descriptor function* (TDF). The TDF relative to a task k is defined as,

$$d_*^k(t, \boldsymbol{q}) \colon \mathbb{R}^+ \times \mathcal{Q} \to \mathbb{R}^+ \tag{2}$$

Note that the TDF may change with time in case of time-varying tasks.

The current state of execution of the task is given by the *current task descriptor function* (CTDF), that is equal to the sum of all the ADFs relative to that task. The CTDF for the task k is defined as

$$d^k(\boldsymbol{p}, \boldsymbol{q}) = \sum_{i \in \mathcal{N}} d_i^k(\boldsymbol{p}_i, \boldsymbol{q}) \tag{3}$$

where $\mathcal{N}$ is the set of the agents.

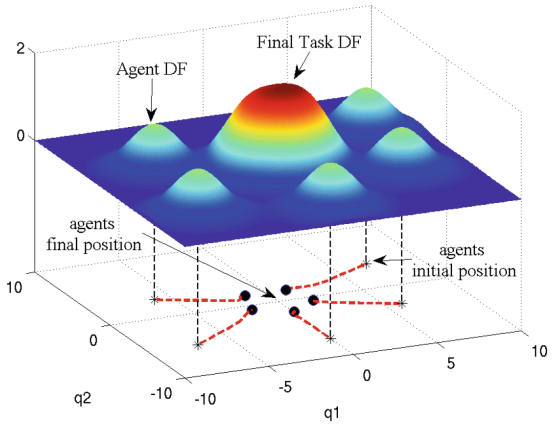


Fig. 2. Current task descriptor function and task descriptor function (taken from [6]).

Figure 2 shows an example of TDF for a *static coverage* task (see Sect. 2.3). With this type of TDF we demand the agents to cover a particular region of the environment. In this example the TDF is a two-dimensional Gaussian function, meaning that the centre of the area requires more resources than its boundaries.

The agents, in order to execute the task, must know the error of the current deployment with respect to the desired one. This information is provide by the *task error function* (TEF), which is given, for example, by the difference between the TDF and the CTDF,

$$e^k(t, \boldsymbol{p}, \boldsymbol{q}) = d_*^k(t, \boldsymbol{q}) - d^k(\boldsymbol{p}, \boldsymbol{q}) \tag{4}$$

Negative values of the TEF denote a lack of resources in the area. Conversely, positive value of the TEF indicates an excess of resources that can be redistributed over the environment.

Task execution can be formulated as an optimization problem: the agents must found the optimal deployment that minimizes a given cost function of the TEF. Using this approach, the same control strategy can be adopted by the agents for tasks execution and many different coordination problems can be handled by the framework simply by changing the TDF. In the past, control laws based on cost function gradient-descent were proposed for agents with single-integrator dynamics [1]. The use of potential field-based control was also considered [8].

The use of control laws based on TEF minimization naturally enables inter-agent cooperation. The agents will avoid regions where the error is low, i.e. areas already covered by other agents, and, as a result, they coordinate their actions during the task execution.

Therefore, the knowledge of the task current execution state, i.e. of the TEF, enables agents self-organization and adaptation. Agents coordinate their actions in order to minimize the TEF (*task-level self-organization*) and participate to the accomplishment of those task for which they can provide resources (*mission-level self-organization*) [1]. Clearly, once the TEF can be estimated, the agents can achieve full autonomy and the whole approach can be decentralized. TEF estimation problem is addressed in [6] by means of consensus-based algorithms.

In the above discussion we focused our attention on how the execution of a task is handled using the descriptor function framework. However, a mission may be composed by more than one task, that must be assigned to the agents according to their capabilities and their priority. Task assignment problem was also analysed by the authors of the descriptor function framework. Biologically inspired methodologies were adopted to address this problem. In particular, the algorithms proposed in [6,9] allow each agent of the team to autonomously decide which task it should execute according to its capabilities and its knowledge of the current task status.

2.2 Obstacle and Collision Avoidance

When moving in a structured or unstructured environment, agents must avoid collision with the obstacles (*obstacle avoidance*), as well as among themselves (*collision avoidance*). In the literature, numerous algorithms have been proposed to address these problems based, for example, on online path planning techniques, potential fields and geometric methods (see e.g. [10–12]). These methods can be readily introduced in the descriptor function framework, so that the agents can safely operate in real scenarios.

In [7] the authors adopted a potential field method to prevent inter-agent collision and obstacles avoidance by agents coordinated with the descriptor function framework.

The presence of obstacles can be also directly included in the descriptor function framework by means of *obstacles descriptor functions* (ODF). The idea is to introduce for each obstacle an ODF, such that the area occupied has no interest for the agents [13]. The TEF is then modified as follows,

$$e^k(t, \boldsymbol{p}, \boldsymbol{q}) = d^k_*(t, \boldsymbol{q}) - d^k(\boldsymbol{p}, \boldsymbol{q}) - \sum_{j \in \mathcal{O}} d_{obs,j}(\boldsymbol{p}, \boldsymbol{q}) \tag{5}$$

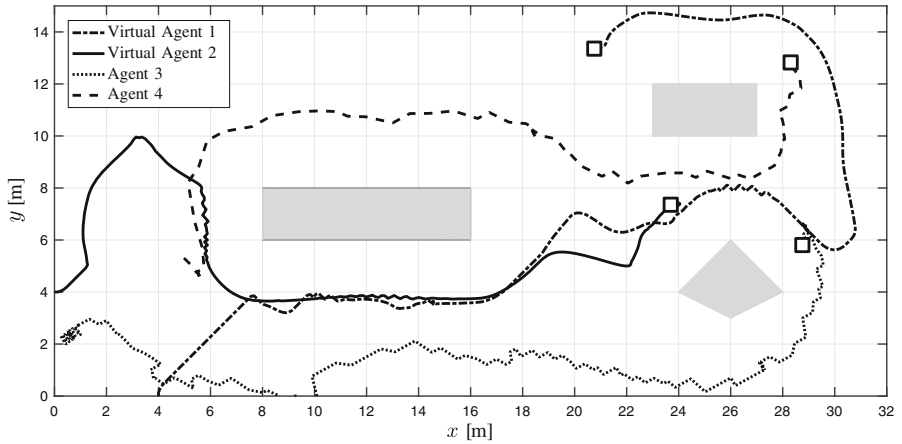
where $\mathcal{O}$ is the set of the obstacles in the environment and $d_{obs,j}(\boldsymbol{p}, \boldsymbol{q})$ is the ODF for the obstacle j. Note that the ODF depends on the agents positions. As the agents get closer to the obstacles, the relative ODF increases in order to reduce the TEF in that area. As a result, the agents move away from the obstacles toward regions with higher TEF value.

2.3 Tasks Examples

In this section, we present a list of tasks successfully implemented and tested using the descriptor function framework. For each task, we provide references for those readers interested in their mathematical formulations, as well as in the technical details about the implementation.

- **Uniform Deployment** [1,6]**.** The agents are demanded to spread over a prescribed area, so that the resources are uniformly distributed. This problem usually arises in sensors networks, where a given number of sensor must be deployed in the environment in order to cover the assigned area.
- **Static Coverage** [6,8]**.** Similar to the uniform deployment, the static coverage requires to deploy the agents in the environment in order to cover a given region. However, regions may have area of highest interest that should be covered first by the agents.
- **Effective Coverage** [6,7]**.** The effective coverage problem, originally formulated in [11], models the task of exhaustive research over a given area. The ADF here describes the efficiency of the agent in sensing the area. The task is complete when all the area of interest is explored. This could be applied to search and rescue missions.
- **Dynamic Coverage** [6,7]**.** This task requires the agents to continuously explore the environment, in order to keep updated their information. An example of dynamic coverage task may be the surveillance and patrolling of a sensitive area.
- **Target Assignment** [6,8]**.** Given K static targets and N agents, we require that at least each target is covered by an agent if $K \leq N$. Otherwise, each agent must cover one target, so that N of them are under control.

The tasks described above were tested by means of extensive simulations. Experimental tests were also conducted to understand the applicability of the framework to real mission scenarios. A test-bed composed by ground vehicles, built at University of Pisa, has been developed over the years to conduct experimental campaigns. A description of an earlier version of the test-bed is presented in [6,14]. In Fig. 3 is shown the execution of a task of effective coverage using these vehicles. The experiment was performed in July 2015 with two real agents (the two vehicles) and two virtual agents simulated by a PC. We introduced three obstacles in order to test also the agents obstacle avoidance algorithms. As can be seen in Fig. 3a, the agents successfully explored the operative space, avoiding collisions among them and with the obstacles.



(a) Agents trajectories at the end of the effective coverage task (squares denote agents final positions).

(b) Agent 3 and 4 executing the effective coverage task.

Fig. 3. Effective coverage experimental test (July 2015).

3 Human-Machine Interaction Within the Descriptor Function Framework

3.1 Modelling the Human Presence

The descriptor function abstraction is a simple and powerful tool that confer to the framework a unique reconfigurability property. The same functions can be used to consider the presence of humans, working on the same task executed by the autonomous agents, within the framework. In the following, we will refer to them as *human agents*.

Human agents capabilities with respect to a task can be described using the same formalism adopted for the autonomous agents, i.e. through ADFs. Hence, we define the *human agent descriptor function* (HADF) for the task k,

$$d^k_{hum}(\boldsymbol{p}, \boldsymbol{q}) \colon \mathcal{P}_{hum} \times \mathcal{Q} \to \mathbb{R}^+ \tag{6}$$

Here $\boldsymbol{p} \in \mathcal{P}_{hum}$, where $\mathcal{P}_{hum} \subseteq \mathbb{R}^{m_h}$, denotes the human agent state vector, that may be composed by its position and orientation in the environment. Since the human agents collaborate with the autonomous agents operating in the environment, the CTDF must consider their contribution to the task execution. We then modify the CTDF as follows,

$$d^k(\boldsymbol{p}, \boldsymbol{q}) = \sum_{i \in \mathcal{N}} d^k_i(\boldsymbol{p}_i, \boldsymbol{q}) + \sum_{j \in \mathcal{H}} d^k_{hum,j}(\boldsymbol{p}_j, \boldsymbol{q}) \tag{7}$$

where $\mathcal{H}$ is the set of the human agents and $\boldsymbol{p}_j$ their positions. Doing so, the TEF computation remains unchanged.

The introduction of the HADF enables the interaction between humans and autonomous agents. The latter can now sense the presence of humans through the TEF. Collision or damage to humans is prevented thanks to the collision avoidance mechanism discussed in Sect. 2.2. Furthermore, the human-machine cooperation is easily achieved. The TEF now reflects the actions taken by the human agents. Since the autonomous agents use the TEF to execute the task, coordination with the current human agents deployment naturally arises, without changing their control strategy.

In order to efficiently coordinate both the human and the autonomous agents, the same control strategy adopted by the autonomous agents can be used to suggest to the human agents how to contribute to the task. This information can be provided, for example, by means of a personal mobile devices. Doing so, the full coordination of humans and autonomous agents is achieved.

3.2 Autonomous Agents Management and Direct Steering

One of the most important features of the descriptor function framework is the possibility to perform different tasks by simply modifying the TDF. In this way, the agents control strategy does not need modifications. Hence, the framework can be used to develop human-machine interfaces that provide to an operator a simple way for monitoring and control a team of autonomous agents.

A possible architecture for the implementation of a command and control interface for team of autonomous agents based on the descriptor function framework is proposed in Fig. 4. The interaction between the operator and the team is handled via an interface that allows to monitor the team status and the task completion level. The operator can order to execute a set of tasks, i.e. assign the mission, choosing among the tasks presented in Sect. 2.3. The resulting list can be either organized by the operator, so that the tasks are executed in a given order, or can be handled by the agents, that according to their internal heuristic and the assigned task priority organize their execution (see Sect. 2.1). The interface can be also enhanced allowing the operator to define new tasks. However, TDF, ADFs and HADFs generation must be handled properly.

Using the architecture described above, we now introduce a new capability to the descriptor function framework: the direct steering of a team of autonomous

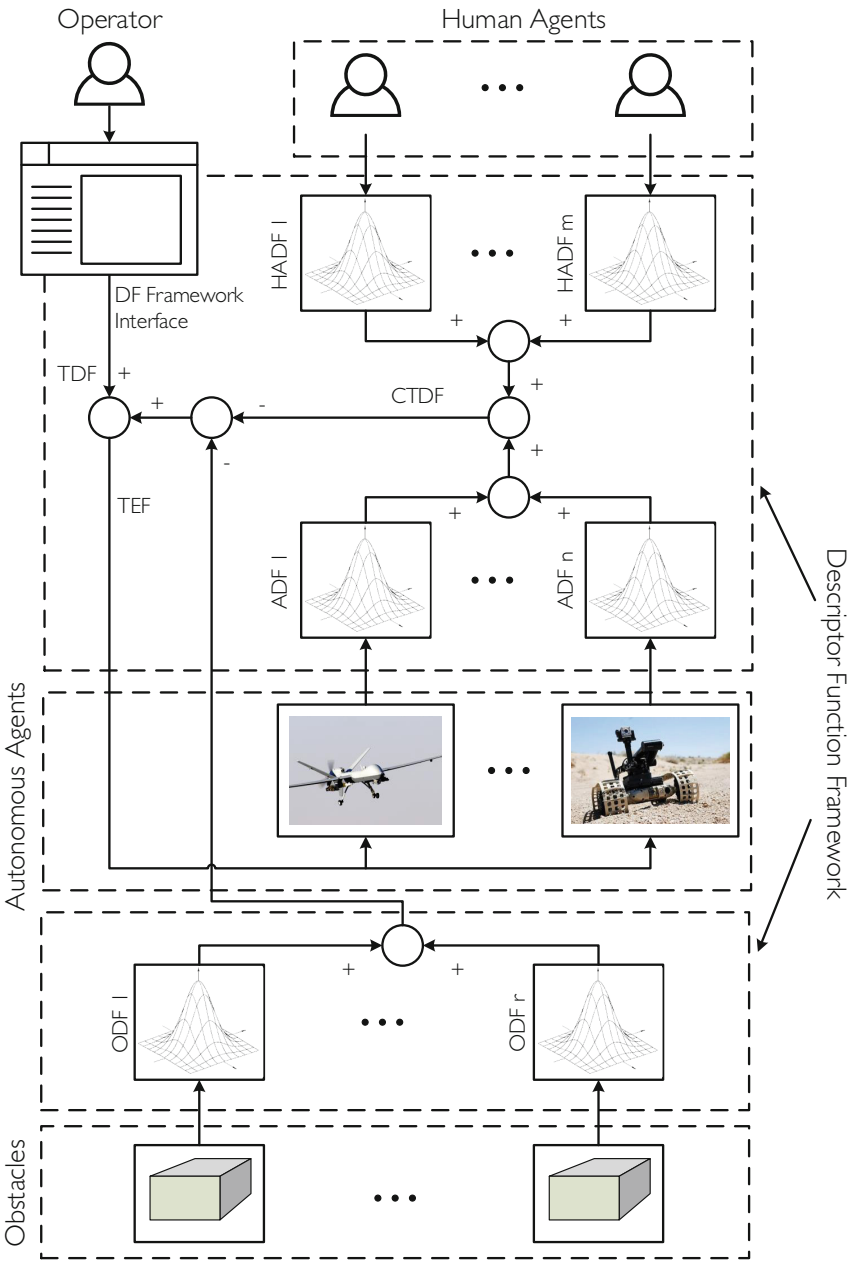


Fig. 4. Descriptor function framework architecture for command and control of team of autonomous agents.

agents. Such a capability may be useful in all that situations where the operator requires to take the direct control of the team, e.g. in response to an emergency situation or in case he needs to move the team to another geographical area. This problem was addressed by the original authors of the descriptor function framework in [14] by means of an abstraction based control in order to simplify the team steering by an operator. Some of the ideas presented in that paper inspired the creation of the descriptor function formalism.

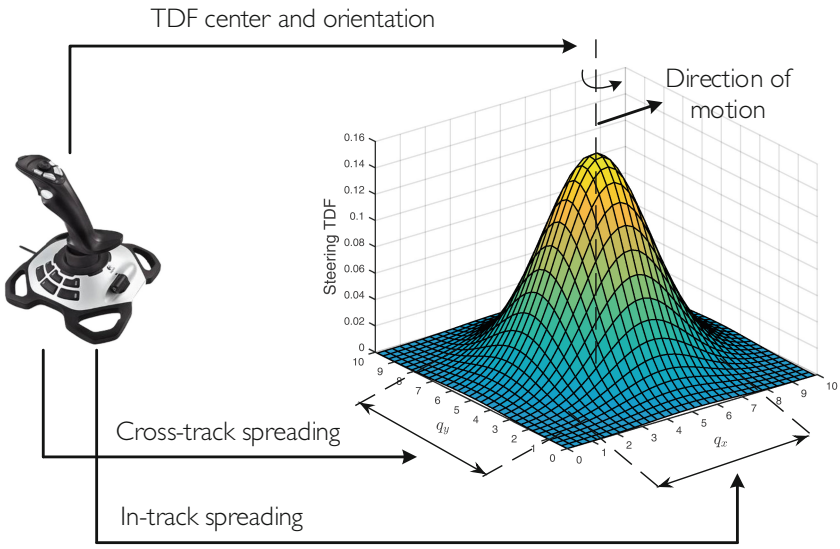


Fig. 5. TDF for autonomous agents steering.

We introduce a new TDF for team steering, which is shown in Fig. 5. The centre of this TDF, as well as its shape and dimensions, are controlled by the operator through, for example, a joystick. In this way, the operator can gather the agents and decide their spreading around the TDF centre. The TDF shown in the figure is a two-dimensional Gaussian function. Changing the standard deviations values, the operator can control the agents spreading along the in-track and cross-track directions. Since the framework integrates algorithms for obstacle and collision avoidance, during the steering the agents will not collide between them and collision with obstacles are prevented. Hence, the control of the team is significantly simplified, even in cluttered environments.

4 Implementation of the Descriptor Function Framework for Human-Machine Interaction

The descriptor function framework, with the enhancements presented in this paper, was implemented using the *Robot Operating System* (ROS) [5]. The C++

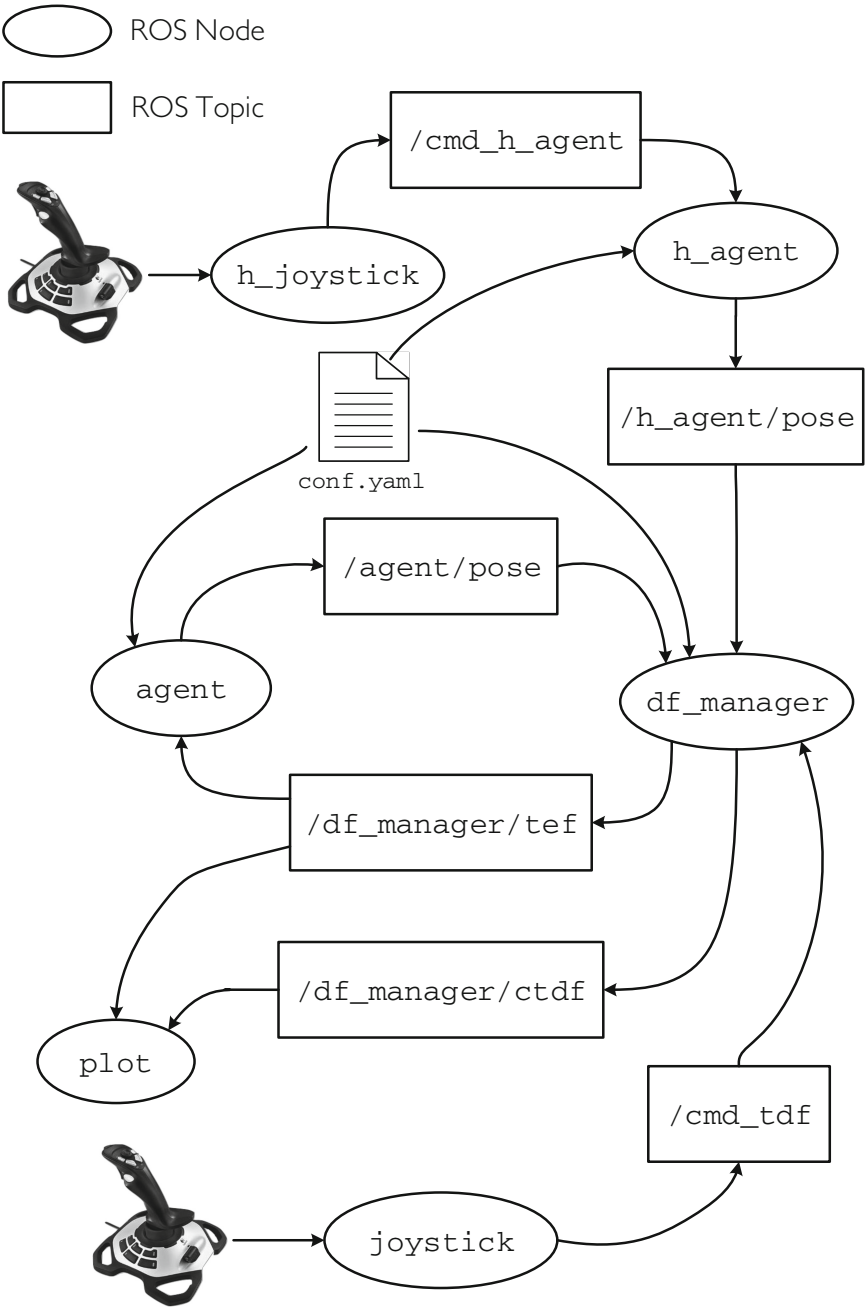
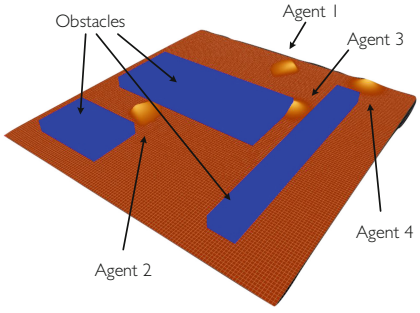


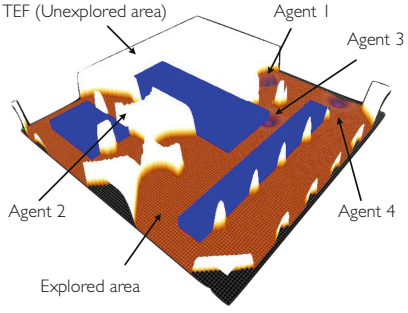
Fig. 6. ROS simulation architecture.

language was chosen for the implementation of the framework. Python was preferred for the development of viewers that help the operator in monitoring the task execution.

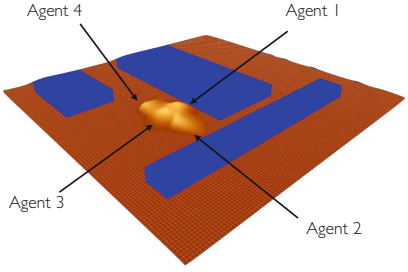
The architecture of the ROS implementation is shown in Fig. 6. The `df_manager` node computes the CTDF and the TEF using the current agents positions, their ADFs and HADFs. The autonomous agents are implemented using the `agent` node. The agents read the TEF from the topic `/df_manager/tef`, compute the control law and simulate their dynamics. The agents then write the new position into the `/agent`i`/pose` topic, where i is the agent's ID. Human agents are implemented using the node `h_agent` and their positions are published in the topic `/h_agent`j`/pose`, where j is the human agent's ID. The `plot` node is used to visualize the CTDF, as well as the TEF, in order to monitor the agents deployment and the level of completeness of the task. For the agents direct steering, we introduced the `joystick` node, that reads the input from a joystick and publish it into the topic `/cmd_pos`. The information written here are used by the `df_manager` in order to update the TDF during the steering. In like manner, the node `h_joystick` can be used to move the human agents using a joystick, in order to simulate their movements in the field.



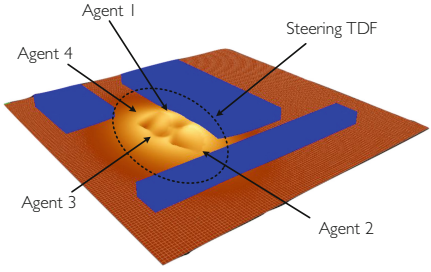
(a) Effective coverage: CTDF.



(b) Effective coverage: TEF.



(c) Direct steering: CTDF.



(d) Direct steering: TEF.

Fig. 7. Examples of task execution.

The configuration file `conf.yaml`, written according to the YAML standard [15], contains the simulation and the agents parameters.

Figure 7 shows some simulations performed with the framework and the developed interface. In particular, the figure shows four screenshots of the CTDF and TEF viewers provided by the `plot` node.

In Fig. 7(a) and (b) are shown, respectively, the CTDF and the TEF during an effective coverage task. Looking at the TEF, the operator visualize the regions of the environment not yet explored by the agents. The CTDF, instead, shows how the resources, i.e. the agents, are deployed in the field.

Figure 7(c) and (d) show the same information, this time during the direct steering of the agents. The former shows how the agents gather around the TDF controlled by the joystick. The latter figure, instead, give to the operator an idea about the joystick TDF and its dimensions, helping him during the steering of the agents.

5 Conclusions and Future Works

In this paper we discussed how human-machine teaming can be achieved within the descriptor function framework. Using the same formalism adopted for the autonomous agents, we can model the human activities in the field by means of descriptor functions. The autonomous agents can perceive the human presence and coordinate their actions accordingly. We also showed how the framework lends itself well for the development of human-machine interfaces that enable the control and monitoring of large teams of autonomous agents. Furthermore, by means of the tool provided by the framework it is possible by a single operator to take the direct control of the autonomous agents and steer a team of agents in structured as well unstructured environments.

Current and future activities are directed to two main areas. The first is the improvement of distributed control and a better analytical formulation of the relationship between the human operator and the autonomous agents. The second is additional validation using ground as well as aerial vehicles.

References

1. Niccolini, M., Innocenti, M., Pollini, L.: Near optimal swarm deployment using descriptor functions. In: 2010 IEEE International Conference on Robotics and Automation, pp. 4952–4957. IEEE (2010)
2. Lee, G., Chong, N.Y.: A geometric approach to deploying robot swarms. Ann. Math. Artif. Intell. **52**(2), 257–280 (2009)
3. Cortès, J., Martinez, S., Karatas, T., Bullo, F.: Coverage control for mobile sensing networks. IEEE Trans. Robot. Autom. **20**(2), 243–255 (2004)
4. Carpin, S., Chung, T.H., Sadler, B.M.: Theoretical foundations of high-speed robot team deployment. In: 2013 IEEE International Conference on Robotics and Automation, pp. 2033–2040. IEEE (2013)
5. Robot Operating System (ROS). http://www.ros.org/

6. Niccolini, M.: Swarm abstractions for distributed estimation and control. Ph.D. dissertation, University of Pisa (2011)
7. Braga, A.F, Innocenti, M., Pollini, L.: Multi-agent coordination with arbitrarily shaped descriptor functions. In: 2013 AIAA Guidance, Navigation, and Control Conference. AIAA (2013)
8. Niccolini, M., Pollini, L., Innocenti, M.: Cooperative control for multiple autonomous vehicles using descriptor functions. J. Sens. Actuator Netw. **3**(1), 26–43 (2014)
9. Niccolini, M., Innocenti, M., Pollini, L.: Multiple UAV task assignment using descriptor functions. In: 18th IFAC Symposium on Automatic Control in Aerospace, IFAC Proceeding Volumes, vol. 43, no. 15, pp. 93–98. IFAC (2010)
10. LaValle, S.M.: Planning Algorithms. Cambridge University Press, Cambridge (2006)
11. Hussein, I.I., Stipanovich, D.: Effective coverage control for mobile sensor networks with guaranteed collision avoidance. IEEE Trans. Control Syst. Technol. **15**(4), 642–657 (2007)
12. Pollini, L., Cellini, M., Mati, R., Innocenti, M.: Obstacle avoidance for unmanned ground vehicles in unstructured environments. In: 2007 AIAA Guidance, Navigation, and Control Conference. AIAA (2007)
13. Innocenti, M., Pollini, L., Franzini, G., Salvetti, A.: Swarm obstacle and collision avoidance using descriptor functions. In: 2016 IEEE Multi-Conference on System and Control. IEEE (2016)
14. Pollini, L., Niccolini, M., Rosellini, M., Innocenti, M.: Human-swarm interface for abstraction based control. In: 2009 AIAA Guidance, Navigation, and Control Conference. AIAA (2009)
15. YAML. http://yaml.org/