S. Silvestrini, M. Lavagna
Neural-Based Predictive Control for Safe Autonomous Spacecraft Relative Maneuvers
Journal of Guidance Control and Dynamics, Vol. 44, N. 12, 2021, p. 2303-2310
doi:10.2514/1.G005481

Neural-based Predictive Control for Safe Autonomous Spacecraft Relative Maneuvers

Stefano Silvestrini * and Michèle Lavagna†
Polytechnic University of Milan, Via La Masa 34, 20156, Milan, Italy

I. Nomenclature

ANN	=	Artificial Neural Network
FMA	=	Feature Matching Approach
GNC	=	Guidance, Navigation and Control
IRL	=	Inverse Reinforcement Learning
$LRNN$	=	Layer Recurrent Neural Network
$LSTM$	=	Long-Short Term Memory
$MBRL$	=	Model-based Reinforcement Learning
MLP	=	Multi-Layer Perceptron
MPC	=	Model Predictive Control
$NARX$	=	Nonlinear Autoregressive Exogenous Model
QP	=	Quadratic Programming
RNN	=	Recurrent Neural Network

II. Introduction

DISTRIBUTED space systems composed of several micro-satellites flying in formation are becoming increasingly attractive for the space community. As the number of satellites increases, an unprecedented level of autonomy is required to perform Guidance, Navigation and

Part of the work has been presented at AIAA Scitech 2020 Forum - 6-10 January 2020 - Orlando, FL - Paper 1918
*PhD Candidate, Department of Aerospace Science and Technology, stefano.silvestrini@polimi.it
†Full Professor, Department of Aerospace Science and Technology, michelle.lavagna@polimi.it

Control (GNC) tasks for formation maintenance and reconfiguration. Model-predictive control is a powerful control strategy that, if combined with convex programming, guarantees optimality and closed-loop control for trajectory generation and control actuation. It has already been studied for distributed reconfiguration [1, 2], nevertheless such planning algorithms are dependent on the accuracy of the dynamical model and they might fail if the on-board dynamical model does not include correct gravitational models, perturbations or other nonlinear terms. In order to maintain a high level of accuracy in the dynamical modelling used for planning, as well as a low computational effort, there is a need for an adaptive algorithm that can cope with partially known environment, whose computational burden does not scale linearly with the modelling accuracy. Classic Reinforcement Learning and meta-Reinforcement Learning have become truly powerful lately, due to their generalization capabilities. Nevertheless, this paper proposes a different approach that is based on hybrid solutions, featuring both AI-methods and classical algorithms. One reason is that classic Reinforcement Learning requires huge and extensive training campaign where thousands of episodes are presented to the agent to be able to learn the correct actions. The reward drives the learnt policy so maintaining an analytical structure of the cost function avoids the need for re-training the agent if the reward was different. In addition, the meta-reinforcement learning approach would certainly be able to handle the collision avoidance constraint. For instance, Gaudet et al. [3] demonstrated that the agent learns to quickly adapt to novel control problems by learning over a wide range of scenarios. Nevertheless, the scenario of distributed collision-free reconfiguration with different target state may lead to nearly infinite configurations that could challenge the reinforcement learning paradigm. For this reason, in this work, only one block (dynamics reconstruction) is left to AI-based method coupling the classical framework.

The proposed Model-Based Reinforcement Learning (MBRL) is developed for controlling a formation configuration and generating trajectories for distributed reconfigurations. This approach enables autonomous quasi-optimal reconfiguration in unknown or unmodelled environments as well as fuel-efficient control strategies for formation maintenance, leveraging the incremental knowledge of the environment.

In a distributed architecture no information is globally available to all agents regarding the planning of each element, therefore each spacecraft needs to predict future maneuvers of potentially colliding agents. Each spacecraft can only measure the relative state. When an impulsive reconfiguration is performed, there are no means to estimate the future trajectory of neighboring spacecraft. Usually, the collision avoidance constraint is enforced by assuming the formation to evolve naturally or along predefined trajectories, which are known by the system [4, 5]. Such assumption is quite restricting when dealing with systems that implement a distributed GNC architecture. The consequence is that the maneuvers may be executed by one spacecraft at a time, preventing the formation to evolve simultaneously. Two novel approaches are here presented to solve the shortcoming and enable the formation to maneuver safely and simultaneously, namely Inverse Reinforcement Learning and Long Short-Term Memory network trajectory forecasting. Inverse Reinforcement Learning (IRL) is employed for impulsive trajectory prediction of neighbouring satellites. Such method was originally developed for imitation learning [6], and then extended to learn demonstrated behaviours and the underlying cost function [7]. Demonstrations of practical real-world performance with three applied case-studies have been given recently [7, 8]. Concerning space applications, Linares [9] proposed an approach based on IRL to determine the behaviour of space objects. The IRL method requires a user-defined cost function structure. Moreover, the method becomes more accurate as the number of observations and predictions grows. To improve the capability of the algorithm for short-term scenarios, a complementing method is based on LSTM recurrent neural networks, which is trained sequentially to predict the temporal behaviour of the observed states. The algorithm scheme used in this paper is shown in Fig. 1. Concerning the whole algorithm architecture, the main features are:

- Each spacecraft senses the environment with relative measurements and learns the underlying dynamics by supervised learning.
- Each spacecraft observes the others, it performs LSTM-IRL and feeds such predictions to the collision avoidance path constraint used in the MBRL.
- Each spacecraft plans its own reconfiguration trajectory using Model-Based Reinforcement

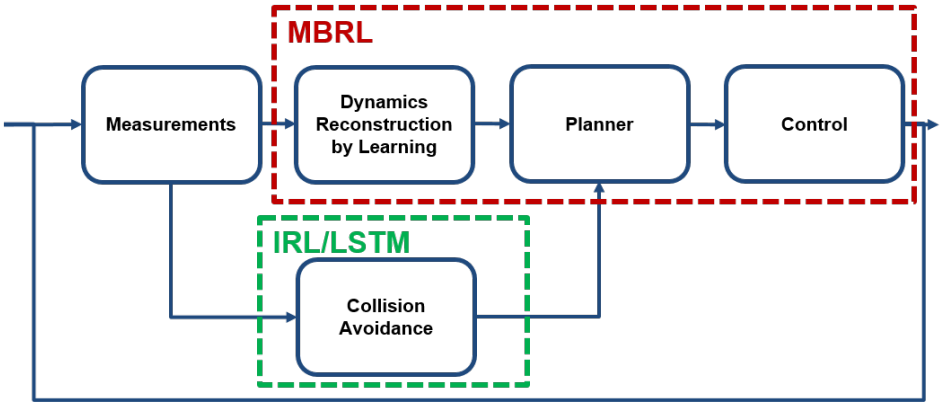

Fig. 1 Overall algorithm architecture for neural-based planning and control for collision-free coordinated spacecraft reconfigurations.

Learning (MBRL), which performs an optimization exploiting the dynamics reconstructed by supervised learning. The model is encapsulated in an Artificial Recurrent Neural Network (RNN). The collision avoidance constraint takes as input the predicted trajectory of neighboring agents.

The justification for this work and the goal of the paper is to develop an algorithm that addresses the following features:

1) To develop a neural-based reconstruction algorithm for system dynamics identification, which allow an autonomous spacecraft to refine the on-board dynamical model as it flies, coping with unmodelled perturbations and nonlinearities. It is achieved by supervised learning of a Recurrent Neural Network.
2) To develop a planning algorithm that can adapt to the perturbed environments using the neural reconstructed dynamics. This prevents the failure of traditional algorithms in presence of unmodelled terms in the dynamical model enhancing the autonomy and flexibility of the spacecraft. The task is performed using the developed Model-Based Reinforcement Learning method.
3) To develop a relative trajectories prediction algorithm to ensure collision-free simultaneous reconfigurations. This is required when coordinated maneuvers are needed, where the hypothesis of the formation evolution according to natural dynamics does not hold. The

neighbouring trajectories are predicted by Inverse Reinforcement Learning and Long-Short Term Memory to guarantee safe reconfigurations.

The paper is structured in four main parts. Section III describes the Model-based Reinforcement Learning algorithm. Section IV describes the trajectory prediction algorithms for collision-free maneuvers. Section V presents the results for the numerical validation. Finally, section VI draws the conclusions.

III. Model-based Reinforcement Learning for Trajectory Planning

A. Spacecraft Formation Dynamics Learning

A Recurrent Neural Network is employed for dynamics reconstruction. Let us assume having a system that evolves according to the equation $\dot{\mathbf{x}} = \mathcal{F}(\mathbf{x}, \mathbf{u})$. Furthermore, it is assumed that the observation is equal to the state for the sake of simplicity, i.e. $\mathbf{y} = \mathbf{x}$. The algorithm can easily be extended to different measurement models by introducing the measurement function $h(\mathbf{x})$ or its linear matrix version $\mathbf{H}$. The system dynamics can be learned using an Artificial Recurrent Neural Network (RNN) trained by standard Levenberg-Marquardt back-propagation algorithm. One effective strategy is to use an analytical model coupled with an ANN to reconstruct uncertain constant parameters (e.g. spherical harmonics coefficients in [10]). Another approach is to couple the neural network with an estimation algorithm to reconstruct only the perturbation terms, taking as basis the linearized natural dynamics [11, 12]. Differently from the above methods, here the whole dynamics is encapsulated into an Artificial Neural Network. The neural dynamics reconstruction lies on the theoretical foundation of the universal approximation theorem of Artificial Neural Networks [13]. Recurrent Networks have the capability of handling time-series data efficiently. The connections between neurons form a directed graph, which allows an internal state memory. This enables the network to exhibit temporal dynamic behaviours. When dealing with dynamics identification, it is crucial to exploit the temporal evolution of the states, hence RNN shows superior performances with respect to MLP [14]. In detail, a Nonlinear Autoregressive Exogenous Model (NARX) recurrent network is employed. It is worth remarking that the NARX model uses control

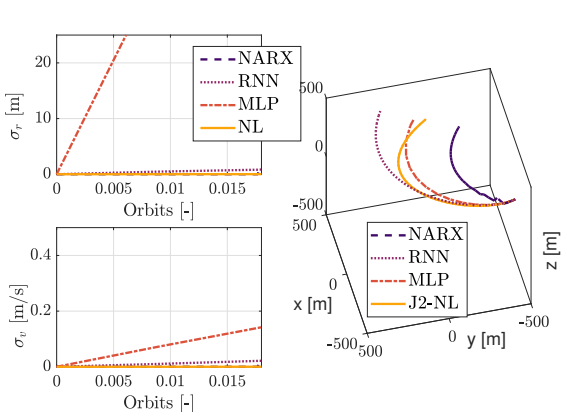

Fig. 2 Comparison between NARX, LRNN and MLP dynamical propagation in the Local-Vertical-Local-Horizontal frame. The plot demonstrates the superior performance of RNN for dynamics reconstruction using coarse initialization training.

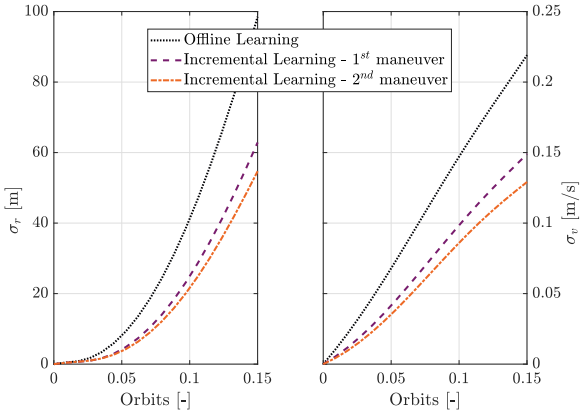

Fig. 3 Comparison between pre-trained (offline) and refined (online) network prediction for $\mathcal{N} = 100$ planning steps.

action as inputs and state as output, given a certain n-delay of the training data. The n-delay of the training data means that the network is fed with a n-long sequence of precedent states. The NARX network shown in Fig. 4 is particularly suited for the task, being able to make prediction when used in *closed-loop* architecture. Fig. 2 shows the different propagation of the feed-forward (MLP), a standard Layer-Recurrent Neural Network (LRNN) and the proposed NARX, pre-trained equally. The reference orbit is a 6 am – 6 pm Sun-synchronous orbit of 775 km altitude, the initial relative state is $\delta\chi = [100\ 100\ 200\ 300\ 200\ 0]/a$ expressed in Relative Orbital Elements. The prediction position and velocity accuracy, compared with the analytical unperturbed nonlinear (NL) and J2-perturbed model (NL-J2), demonstrates the superior performance of recurrent networks in dynamics reconstruction. The offline pre-trained NARX is constantly updated on-board while performing operations, as described later (see Eq. 1). Fig. 3 shows the improvement for the same open-loop prediction of $\mathcal{N} = 100$ planning steps between the pre-trained network (offline) and the network generated after performing one reconfiguration simulation. It is important to remark that the prediction reported in Fig. 3 is open-loop, namely a forward propagation, which is not representative of the closed-loop utilization of the network in the guidance and control algorithm. As the agents

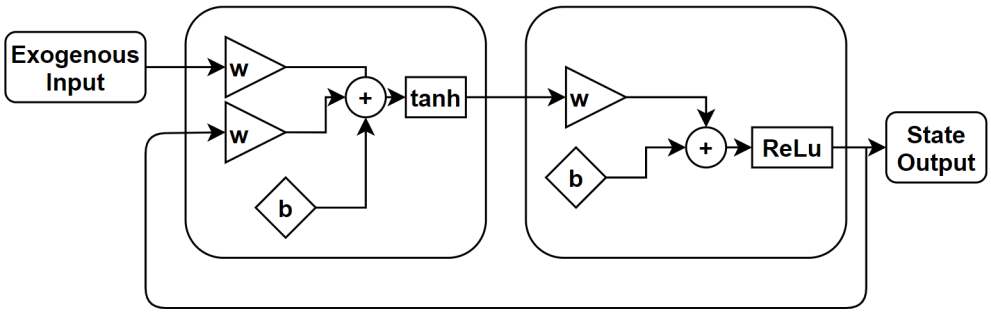

Fig. 4 Nonlinear Autoregressive Exogenous Model used for on-board dynamical model reconstruction.

keep performing relative orbital maneuvers the knowledge of the actual dynamics is refined online, which can be used for a incremental performance planning. The NARX network is a 2-layer, 2-step delay with the $10-6$ neurons shown in Fig. 4. The process of learning can be initiated off-line based on one of the well-known relative dynamical model. The pre-training is performed presenting to the network a set of 1000 trajectories generated using the Clohessy-Wiltshire model. The dynamical model can be represented as $\mathbf{x}_{k+1} = \tilde{\mathcal{F}}_{T_s}(\mathbf{x}_k, \mathbf{u}_k)$, where the transition matrix is associated to the sampling time T_s. The model is learned using the network $\tilde{\mathcal{N}}_{T_s}$ trained by back-propagation. In this paper, the well established Levenberg-Marquardt algorithm is employed for minimizing:

$$\min_{\mathbf{w}} \sum_{k} ||\tilde{\mathcal{N}}_{T_s}(\mathbf{x}, \mathbf{u}, \mathbf{w}) - \mathbf{y}_{k+1}||^2 \tag{1}$$

where $\tilde{\mathcal{N}}$ is the Artificial Neural Network model at the current learning step, hence the dependency on the weights $\mathbf{w}$. The vector $\mathbf{y}$ is the observation vector, here assumed to be equal to the state $\mathbf{x}$.

B. Neural Planning and Control

Model predictive control (MPC) is an optimization-based guidance and control strategy, which merges the advantage of optimization and closed-loop control [1]. Typically, the objective function includes the quadratic difference between the target state and the spacecraft state, $\mathbf{x}_k$ at each time

step, and a quadratic term representing the control effort [1], as in Eq. 2.

$$\mathcal{J}(\mathbf{x}_k, \mathbf{u}_k) = \left[(\mathbf{x_{k+N}} - \mathbf{x_k^*})^{\mathbf{T}} \hat{S} (\mathbf{x_{k+N}} - \mathbf{x_k^*}) + \sum_{i=1}^{N-1} (\mathbf{x_{k+i}} - \mathbf{x_k^*})^{\mathbf{T}} S (\mathbf{x_{k+i}} - \mathbf{x_k^*}) + \sum_{i=0}^{N-1} \mathbf{u_{k+i}^T} R \mathbf{u_{k+i}} \right] \quad (2)$$

where S, R are positive semi-definite matrices, $\hat{S}$ is the solution of the Discrete Algebraic Riccati Equation (DARE), N is the number of prediction steps, k indicates the current time step, i the time steps within the receding horizon. At each time step, the optimization problem can be expressed as:

$$\begin{aligned} &\min_{\mathbf{u}} \ \mathcal{J}(\mathbf{x}_k, \mathbf{u}_k) \\ &\text{subject to } \ \mathbf{x_{k+i+1}} = \tilde{\mathcal{F}}_{T_s}(\mathbf{x_{k+i}}, \mathbf{u_{k+i}}), \ i{=}1,..,N \\ &\qquad\qquad \mathbf{u_{min}} < \mathbf{u_i} < \mathbf{u_{max}}, \ i{=}1,..,N \\ &1 - (\mathbf{x_{p,k+i}} - \mathbf{x^o_{j,k+i}})^{\mathbf{T}} P (\mathbf{x_{p,k+i}} - \mathbf{x^o_{j,k+i}}) < 0, \ i{=}1,..,N, j{=}1,..,m, j{\neq}\, p \end{aligned} \quad (3)$$

where k indicates the current time step, i the time step within the receding horizon. The last constraint refers to the collision avoidance and will be thoroughly discussed in Section IV. The objective function is minimized respecting the dynamics constraint and the maximum thrust limit. The optimization variable is the control history $\mathbf{U}_k$, which is the set of control actions for each time step within the receding horizon. $\mathbf{U}_k = [\mathbf{u}_k, ..., \mathbf{u}_{k+N-1}]$ is a stacked vector containing the decision variables for the optimization problem, namely the control action for each discretization time step. The Model-Based Reinforcement Learning (or Neural-Model Predictive Control) exploits the neural dynamics reconstruction presented in section III.A to enhance the robustness, effectiveness and flexibility of the traditional MPC. The difference is that the dynamical model used in the optimization is encapsulated in an Artificial Neural Network model. The latter is mathematically tractable, and it adapts continuously as the spacecraft flies. In its pre-trained form, the AI-model approximates the linear dynamics (i.e. Clohessy-Whiltshire model). In the MBRL framework, the dynamics used in the MPC in Eq. 3 is replaced by the neural model $\mathcal{N}_s$. Such strategy creates a highly-coupled learning and planning algorithm that resembles the approach of reinforcement learning but does not prevent the exploitation of the underlying physics. As anticipated, the controller is extended to

feature the collision avoidance constraint. The constraint is nonlinear, hence the problem transforms into a quadratic optimization with nonlinear constraints. In this paper the MBRL is compared with a traditional MPC controller. The MPC implementation uses the same optimal control problem of MBRL, namely the same cost function and constraints. The analytical dynamical model used is the Clohessy-Whiltshire linearized dynamics. Both approaches are solved using *sequential quadratic programming (sqp)*. The *sqp* has been selected for the medium-scale problem in the receding horizon optimization as well as for its efficiency and robustness.

IV. Relative Trajectory Prediction for Collision Avoidance

As stated in section II, one critical task for distributed operation is to safely maneuver avoiding collision between agents. The distributed architecture does not allow the agents to communicate their own state, planned trajectories and target position to the other elements of the formation. This means that when the satellites perform impulsive maneuvers they have no means to predict the relative trajectories of the other agents. It is critical that, based on few relative observation between agents, the reconfiguration trajectory of neighboring satellites could be predicted. In this way, collision avoidance constraints can be taken into account by the each spacecraft's MBRL algorithm in the shape of a keep-out-ellipsoid. The path constraint for satellite p is added as:

$$1 - (\mathbf{x_{p,k+i}} - \mathbf{x^o_{j,k+i}})^{\mathbf{T}} P (\mathbf{x_{p,k+i}} - \mathbf{x^o_{j,k+i}}) < 0,\ i{=}1,..,N, j{=}1,..,m, j{\neq}\, p \tag{4}$$

where m is the number of agents, k indicates the current time step, i the time step within the receding horizon. The $x^o_{j,k+i}$ vector is the predicted trajectory derived by the collision avoidance algorithm. The non-convex constraint in Eq. 4 is convexified by unfolding the ellipsoid into a tangent plane, as done in previous work [14]. Convex optimization guarantees the convergence of the algorithm to a global minimum. As the number of satellites grows, the constraint becomes more challenging to fulfill, given that the set of feasible positions reduces dramatically. Inverse Reinforcement Learning and Long-Short Term Memory are two methods for generating predictions

of future trajectories based on few observations. The two approaches were thought to be used interchangeably, nevertheless, the results will show the benefit of coupling the two algorithms. The methods addressing the collision avoidance are focused on the prediction of impulsive trajectories of neighboring agents. The planning itself is carried out by MBRL, aided by the prediction made by the IRL-LSTM. At each GNC time step, the IRL runs to generate the neighboring agents trajectory prediction, such prediction is fed to the MBRL that uses the trajectory prediction to incorporate the convexified collision avoidance constraint.

A. Inverse Reinforcement Learning

In general, the cost function of an optimal control problem for trajectory planning in the horizon (t, T) can be described as the summation of a running intermediate cost and the terminal state penalty. The cost function can be recast into a feature-based expression where the cost is a linear combination of f nonlinear features. In principle, each state-control pair can represent a feature. In this paper the optimizer is described as an optimization over a linear combination of features ϕ, which represent cumulative cost along feasible trajectories and terminal state penalty. The feasible trajectories are those respecting optimization constraint and dynamics: the set of state-control pairs represents a *policy* π, borrowing a term from the Reinforcement Learning world. Using such approach, the cost function can be rewritten [6]:

$$\mathbf{w} = [w_1, w_2, ..., w_f, w_T]^T, \ \mu(\pi) = \begin{bmatrix} \{\sum_t^{T-1} \phi(\mathbf{x_t}, \mathbf{u_t})\}_{1,...,f} \\ \phi_T(\mathbf{x_T}) \end{bmatrix} \rightarrow \text{cost: } \mathcal{J} = \mathbf{w}^T \cdot \mu(\pi) \qquad (5)$$

The above formulation is necessary to introduce and discuss the Feature Matching Approach for Inverse Reinforcement Learning.

1. Feature-Matching Approach

The concept of Inverse Reinforcement Learning is to estimate a cost function that delivers an optimal trajectory compatible with an *expert* demonstrated trajectory, called $\tilde{\gamma} = \{(\mathbf{x_t}, \mathbf{u_t})\}_t^T$ for

simplicity. The demonstrated trajectory is the reconfiguration path followed by the neighbouring agents. The estimated cost function translates into trajectories through the application of an optimal control problem. Such optimal control problem is hand-crafted: in this paper, the MBRL described above is used as planner, where the control history (i.e. control action for each horizon step) is the only decision variable. It is important to note that the expert cost function, parametric in the set of features, is not known, but we assume the demonstration to be optimal for the estimated one. Each trajectory is generated by a policy π, which characterizes the state-control pairs at each instant in time, ideally. In detail, given a set of N_o observations of the demonstrated trajectory $\mathcal{D}_{\tilde{\gamma}} = \{(\mathbf{x_i}, \mathbf{u_i})\}_{i=1}^{N_o}$ we need to find a cost function $\mathcal{J}$, under which the demonstrated trajectory looks optimal according to the estimated cost function. It means that the demonstrated trajectory is the result of an optimization of a certain cost function that we want to estimate, namely the *expert cost function*. Feature Matching Approach solves for the weights in Eq. 5 by attempting to match the cumulative feature cost demonstrated by the *expert* under the optimal policy $\tilde{\pi}$ and the policy π^* based on the estimated cost function. The FMA can be expressed in compact form:

$$\mathcal{J}(\tilde{\gamma}|\tilde{\pi}) < \mathcal{J}(\gamma|\pi) \rightarrow \mathbf{w}^T \cdot \mu(\tilde{\pi}) < \mathbf{w}^T \cdot \mu(\pi),\ \forall \gamma, \pi \tag{6}$$

where it is stated that the demonstrated trajectory is optimal with respect to the estimated cost function. The demonstrated trajectory owns significant information about the structure of the cost function, hence the algorithm significantly benefits if the discrepancy between estimated optimal trajectory and expert one is integrated in the optimization [15, 16]. Loss augmentation represents a cost gap structured margin: $\mathcal{L}_{\tilde{\gamma}}(\pi) = \sum_{i=1}^{N_o} ||\tilde{\mathbf{x}}_i - \mathbf{x}_i||^2$. By inserting the loss augmentation in Eq. 6 we obtain the expression for the FMA for IRL:

$$\begin{aligned} &\min_{\mathbf{w}} \ ||\mathbf{w}||^2 \\ &\text{subject to} \quad \mathbf{w}^T \cdot \mu(\tilde{\pi}) < \mathbf{w}^T \cdot \mu(\pi) - \mathcal{L}_{\tilde{\gamma}}(\pi) \end{aligned} \tag{7}$$

Eq. 7 represent a convex optimization. Nevertheless, the set of policies the algorithm sweeps is theoretically not finite. This makes the optimization intractable, in particular for computational constraints of several interesting applications in spacecraft GNC. We may suppose that there exists a policy π^* that minimizes the right-hand side of the constraints in Eq. 7. Then, the constraints in Eq. 7 can be written as $\mathbf{w}^T\mu(\tilde{\pi}) < \min_{\pi}\{\mathbf{w}^T\mu(\pi) - \mathcal{L}_{\tilde{\gamma}}(\pi)\}$ without any loss of generality. We can place the constraint into the cost function. This yields an unconstrained optimization, which can be solved using gradient-based algorithms:

$$\min_{\mathbf{w}} \frac{\eta}{2}||\mathbf{w}||^2 + \left(\mathbf{w}^T\mu(\tilde{\pi}) - \min_{\pi}\{\mathbf{w}^T\mu(\pi) - \mathcal{L}_{\tilde{\gamma}}(\pi)\}\right) \tag{8}$$

where η is a user-defined coefficient. The FMA-IRL is actually a nested optimization problem. The outer loop is unconstrained, whereas the inner loop, which is a path-planning optimization, may be constrained, both linearly and non-linearly. The cumulative feature cost for the formation spacecraft trajectories can be expressed as:

$$\mu(\pi) = \begin{bmatrix} \mathbf{X}_k^T\mathcal{S}_s\mathbf{X}_k \\ \mathbf{U}_k^T\mathcal{R}_r\mathbf{U}_k \\ (F\mathbf{X}_k)^T\hat{\mathcal{S}}(F\mathbf{X}_k) \end{bmatrix} \tag{9}$$

where $\mathbf{X}_k$, $\mathbf{U}_k$, $\mathcal{S}_s$ and $\mathcal{R}_r$ are the stacked vectors and matrices to shorten the summation in Eq. 5. Note that no target state guess is inserted, making the formulation insensitive to such parameter. The target state guess is a-priori estimation of the target state, i.e. the objective state of the formation. From Eq. 9, the weights vector is $\mathbf{w} = [w_s, w_r, w_{\hat{s}}]^T$.

2. Inner Loop: Fast QP-Model Predictive Control

The nested optimization resulting from the FMA for trajectory prediction relies on an inner loop, calculating the trajectory generated by the estimated cost function. The inner *policy* evaluation is performed using a fast MPC recast in QP formulation. Such strategy allows a rapid convergence

of the inner loop, limiting the computational burden of the whole algorithm. In particular, the MPC resembles the architecture presented in Section III with the additional term entailing the gap structured margin $\mathcal{L}_{\tilde{\gamma}}(\pi)$. The proposed implementation uses the Clohessy-Whiltshire linearized natural dynamics for state prediction. In general, this approach can be extended to a neural reconstructed dynamics. The QP formulation is:

$$\begin{aligned} &\underset{\mathbf{U}_k}{min}\ \frac{1}{2}\mathbf{U}_k^T \mathcal{Q}\mathbf{U}_k + \mathcal{H}\mathbf{U}_k \\ &\text{subject to } \mathcal{V}\mathbf{U}_k < \mathcal{W} \end{aligned} \tag{10}$$

where $\mathbf{U}_k = [\mathbf{u}_k, ..., \mathbf{u}_{k+N-1}]$ is a stacked vector containing the decision variables for the optimization problem, namely the control action for each discretization time step. The matrices of the QP formulation can be written as $\mathcal{Q} = 2w_r\mathcal{R}_r + 2w_{\hat{s}}\Omega^T\mathcal{S}_{\hat{s}}\Omega - 2\Omega\mathcal{P}_p\Omega$ and $\mathcal{H} = 2\hat{\mathbf{x}}_k^T\Psi\mathcal{S}_{\hat{s}}\Omega + 2\mathbf{X}_\gamma\mathcal{P}_p\Omega$, where $\mathbf{X}_\gamma$ is the stacked vector of the demonstrated trajectory, $\mathcal{P}_p$ is a stacked identity matrix. The matrices Ω and Ψ represent the stacked state transition matrix and control matrix for the linearized relative dynamics. The demonstrated trajectories for each agent is generated using the MBRL approach. Each agent demonstrates its own trajectory to all the other elements of the formation.

3. Outer Loop: Unconstrained Optimization

The cumulative feature cost is a function of the policy, which is basically the sequence of control action output of the MBRL. The inner loop generates an optimal policy (control action sequence) based on the current estimate of the cost function (current weights). The optimal policy is used to calculate the cumulative feature cost used in Eq. 8. The outer loop is a convex unconstrained minimization problem that can be solved by *quasi-newton* methods with gradient descent. In the proposed implementation, the gradient reads $\nabla_w = \eta\mathbf{w} + (\mu(\tilde{\pi}) - \mu(\pi))$, which completes the FMA-IRL algorithm.

B. Neural-Sequential Trajectory Forecasting

The trajectory prediction for collision avoidance based on IRL is a promising algorithm, which delivered good results during the numerical validation test campaign. Although being simple and effective, IRL presents two drawbacks for working out the delicate task:

- the cost features have carefully been designed by analyzing the dynamics of the spacecrafts. In particular, it is assumed that all the spacecrafts executes the formation reconfiguration using an MBRL algorithm with the same cost function.
- the unknown cost function is assumed to be quadratic in the control effort, hence it is required at least an estimate of the observed control thrust impulse. A possible way to solve this issue is approximating the control term by a discrete velocity differential between two subsequent observations, provided that they are close in time.

As stated above, Recurrent Networks have the capability of handling time-series data efficiently. From this premises, the use of RNN can be extended also for trajectory identification of neighbouring satellites. Differently from the dynamics reconstruction used in MBRL, the network for neighbouring satellites, does not need the control input, since it tries to estimate the future trajectory by only observing the relative state. The type of recurrent neural network used for neighbouring satellites trajectory prediction is a Long Short-Term Memory network [17]. The LSTM network is used for on-board prediction of future state given the observed trajectory history (i.e. observed relative measurements). In this paper, a sequence of observed relative states is used to perform the online training. As the number of observations grows, the prediction becomes more accurate, nevertheless even limited observations (e.g. as low as 10) guarantee an acceptable prediction for the short-term horizon. The LSTM can be trained to make predictions based on time sequence data. Given the demonstrated trajectory $\tilde{\gamma} = \{(\mathbf{x_t}, \mathbf{u_t})\}_t^T$, where we neglect the control $\mathbf{u_t}$, the proposed sequential supervised learning is performed by feeding the network with the gathered input states and compared with the same states shifted of one time step: namely $\mathbf{x}_{in,LSTM} = \{\tilde{\gamma}\}_{(t,t+(N-1)\cdot T_s)}$, $\mathbf{y}_{out,LSTM} = \{\tilde{\gamma}\}_{(t+T_S,t+N\cdot T_s)}$. In order to obtain a more robust fit and to prevent the training from diverging, it is required to standardize the training data to have zero mean and unit variance.

Finally, the complete algorithm reads as Algorithm 1.

Algorithm 1 Model-Based Reinforcement Learning with Collision-Avoidance Prediction

Pre-train system description through dynamical model $\mathbf{x}_{k+1} = \tilde{N}(\mathbf{x}_k, \mathbf{u}_k)$, k discretized time step
Acquire target position $\mathbf{x}^*$
while $\mathbf{x}_k \neq \mathbf{x}^*$ **do**
    Observe system state at time k: $\mathbf{x}_k$
    Acquire formation geometry: $\mathbf{x}_i - \mathbf{x}_k \ \forall i = 1, .., m$, m number of spacecraft in the formation
    Execute trajectory prediction of neighboring agents (IRL-LSTM)
    Generate predictions of future states X_P of neighboring agents
    Solve Optimal Control Problem with cost $\mathcal{J}(\mathbf{x}_k, \mathbf{u}_k)$ using neural-dynamics $\tilde{N}$ for N time steps, including X_P in the collision avoidance constraint
    Execute $\mathbf{u}_k$
    Observe system state at time instant $k + 1$ and perform incremental learning of $\tilde{N}$
end while

V. Numerical Simulations and Results

The proposed algorithm architecture is reported in Fig. 1 and tested in two application scenarios, representative of in-plane and out-of-plane relative reconfiguration. The selected scenarios are tight formations with 1 N thrust limit, requiring constant relative position and velocity control to satisfy the strict requirements on relative states. The error with respect to the target state is indicated as $\sigma_p = \|\mathbf{x} - \mathbf{x}^*\|$. The numerical simulations features:

- Each spacecraft plans its own trajectory using the MBRL algorithm. The planner entails a collision avoidance constraint in the optimization scheme. The Guidance & Control sample time is 60 s. Each spacecraft measures only the relative state of all the other agents of the formation. The predicted trajectories of each neighbouring satellites is delivered by the LSTM and IRL algorithms and is fed to the constrained MBRL.
- The *true* dynamics used to carry out simulations is a fully nonlinear J2-perturbed relative model with respect to the reference orbit reported in Tab. 2.
- The simulation is stopped when the target configuration is reached, i.e. when position and velocity errors are below 10 m and 0.1 $\frac{m}{s}$, respectively.

A summary of the employed neural networks is reported in Tab. 1

Table 1 Artificial Neural Networks used in MBRL and collision-avoidance specifications.

Type	NARX	LSTM
Number of Layers	2 (2-delay)	2
Type of Layers	tanh-lin	sequential-lstm
Hidden Units	10+6	100
Input Size	3 (u_x, u_y, u_z)	3 (x, y, z)
Output Size	6	3

Table 2 Reference Orbit - Orbital Elements.

h [km]	e [-]	i [^{o}]	ω [^{o}]	Ω [^{o}]	θ [^{o}]
775	0.01	98	0	0	0

A. In-Plane Maneuver

Three satellites fly in rigid formation, i.e. a formation in which the relative distances and orientation among the satellites remain fixed. The aperture plane is perpendicular to the orbital plane. The initial condition consists in the three satellites at the vertices of an equilateral triangle of 13 m side lying on y-z plane. Yaw rotations of the aperture plane are possible as long as the relative formation between the arrays is kept. The numerical results for a yaw rotation of 30°, shown in Fig. 5, are reported in Tab. 3. For close reconfiguration, the MBRL outperforms the traditional MPC both in terms of fuel consumption and time of flight, as reported in Tab. 3. The required Δv is $\sim 5\%$ lower when using MBRL. It is interesting to test the algorithm in a hypothetical scenario where the relative distances are two order of magnitude larger. The difference in accuracy and required Δv between the compared algorithms becomes more significant, as shown in Fig. 6. This is due to the

Table 3 Δv and time of flight for MBRL and MPC.

FR	**MBRL**			**MPC**		
	SC1	SC2	SC3	SC1	SC2	SC3
Δv_T $[\frac{m}{s}]$	1.2	1.0	1.0	1.3	1.1	1.1
Time of Flight $[s]$	600	600	600	660	660	660

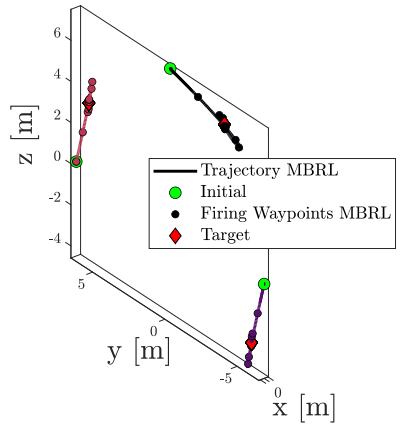

Fig. 5 Formation reconfiguration following 30° yaw rotation of the synthetic aperture.

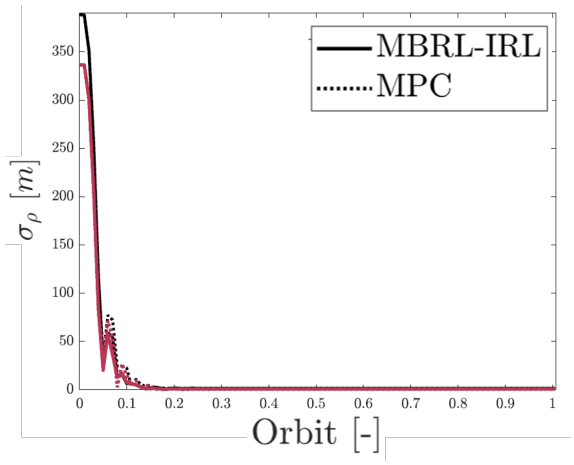

Fig. 6 Large reconfiguration following 30° yaw rotation of the synthetic aperture.

fact that nonlinearities become more and more relevant as the formation grows in size. The MBRL utilizes the neural reconstructed dynamics that captures these unmodelled terms.

B. Out-of-Plane Maneuver

The cross-track formation can be achieved by fixing the relative position or exploiting natural relative orbits, which owns a cross-track harmonic motion. The relative orbital elements, in particular δe and δi completely defines the central bounded relative orbit [18, 19]. The simulated reconfiguration is a transfer between two cross-track relative orbits with parallel and perpendicular relative inclination/eccentricity vector, namely from $\boldsymbol{\delta e} \perp \boldsymbol{\delta i}$ to $\boldsymbol{\delta e} \parallel \boldsymbol{\delta i}$ with $a\delta i = a\delta e = 2\ km$. In such reconfiguration, far from the target, the MPC fails in trajectory control and the controller diverges, whereas MBRL completes the reconfiguration using $\Delta v \sim 30\ \frac{m}{s}$, as shown in Fig. 7. The MPC failure is due to the large distance and the cross-track requirement, which is different from the other test cases. Nevertheless, if one could know beforehand the environment, the MPC could be tuned to solve the reconfiguration successfully. The main point is that, if an MPC scheme (with its associated tuned parameters) is used without modifications in all the scenarios, the output can fail given the low-adaptivity. The MBRL still uses the same tuning for all the scenarios presented but it is still able to perform all the reconfigurations successfully.

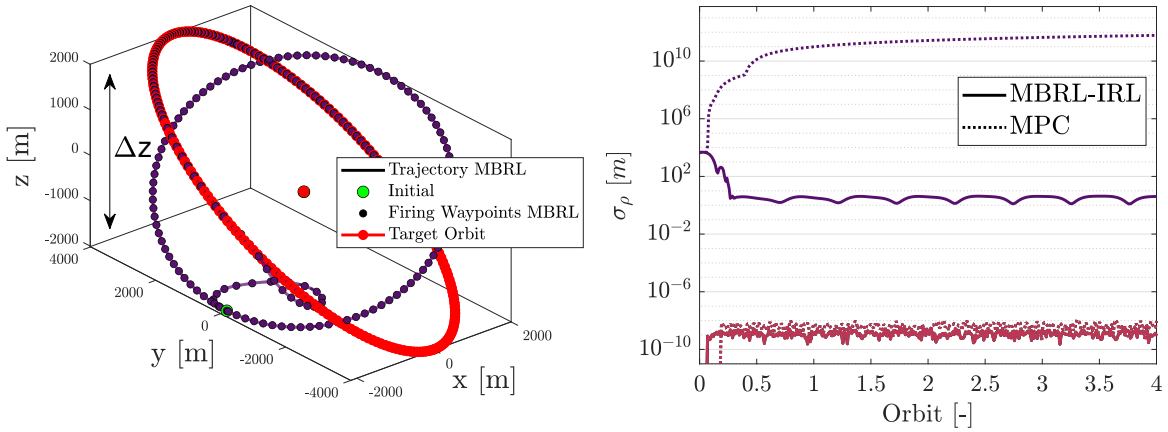

Fig. 7 Cross-track reconfiguration from $\delta e \perp \delta i$ to $\delta e \parallel \delta i$ with $a\delta i = a\delta e = 2\ km$.

C. Collision Avoidance Algorithms

Two algorithms have been presented to cope with the lack of knowledge on the neighbouring satellites trajectories while performing reconfiguration. If the formation is reconfiguring using impulsive maneuvers the zero-impulse natural dynamics is no longer valid, indeed the keep-out-zone limit may often be intersect, violating the constraint due to wrong trajectory prediction. Fig. 8 shows a challenging reconfiguration where the spacecraft swap the along track positions separated by 200 m. As shown, the relative distance between the satellites falls below the Keep-Out-Zone limit of 100 m when the prediction of neighbouring agents is carried out using natural dynamics. On the other hand, coupling the MBRL planner with an impulsive trajectory identification algorithm, such as IRL, allows a safe reconfiguration.

The comparison simulation consists of a sample of $\mathcal{N}_o$ observation of a neighbouring satellite's trajectory. These observation are processed by the IRL algorithm to approximate a cost function, whose optimization delivers a predicted trajectory shown in Fig. 9. Simultaneously, the observations are used to train the LSTM network, which predicts future states. The number of prediction is set to $\mathcal{N}_p = 5$, whereas $\mathcal{N}_o = 10$. The two algorithms are beneficial with respect to natural dynamics prediction for controlled reconfiguration, used in literature. The prediction accuracy for the proposed architecture is $< 10\ m$ compared to $\sim 100\ m$ if using prediction based on the system natural dynamics evolution. A more comprehensive analysis has been conducted to map the behaviour

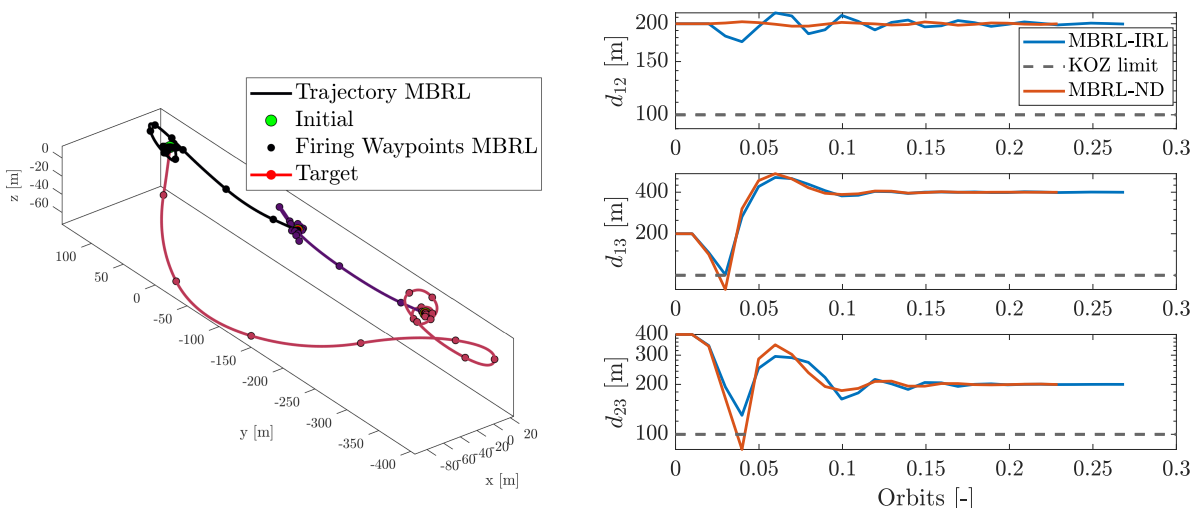

Fig. 8 Close intersecting reconfiguration. The relative distances between the agents are shown. MBRL-ND coupled with natural dynamics prediction for collision avoidance violates the constraints during close approach.

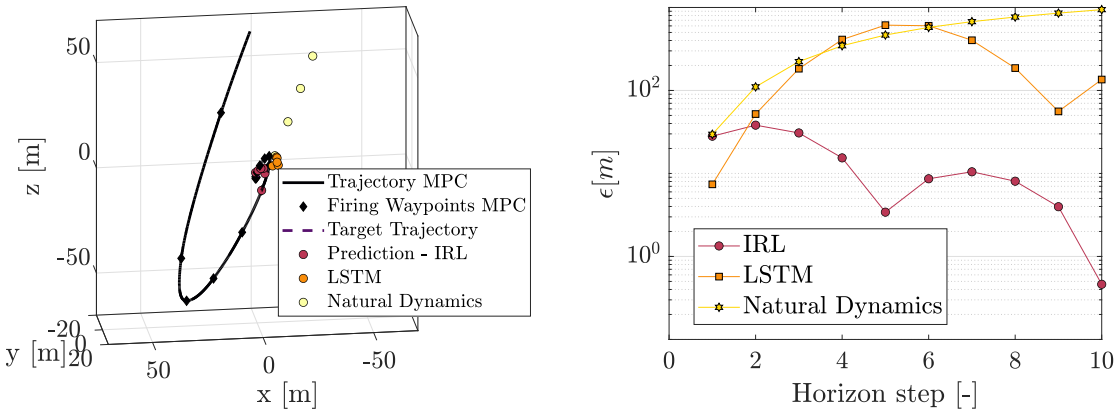

Fig. 9 Neighbouring satellite's trajectory. Predicted trajectories based on IRL, LSTM and natural dynamics are shown in colored dots. The error plot is shown on the right.

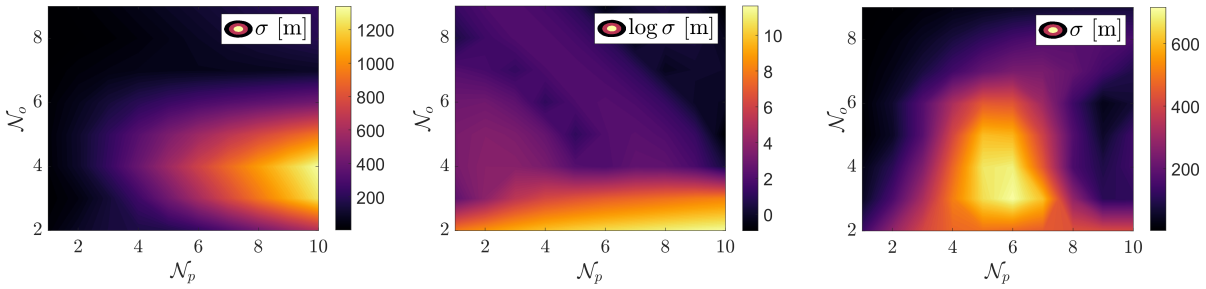

Fig. 10 Map of RMS error of the neighbouring agent trajectory prediction based on CW-model, IRL algorithm and LSTM, left to right, as a function of the number of observations and predictions.

of the algorithm for different combination of $\mathcal{N}_p$ and $\mathcal{N}_o$ in the same scenario described before. Fig. 10 shows the difference in the presented algorithms. On one hand, short-horizon predictions based on few observations are better resembled using LSTM. On the other hand, the IRL accuracy is higher when a larger observation set is used. A combination of the two approaches may be the best solution. The simulations are run using a Laptop computer using Intel(R) Core(TM) i7-6500U CPU@2.50GHz. The computational time is very limited for IRL ~ 1 s, whereas the network takes approximately ~ 2.5 s to perform the training. The computational burden scales linearly with the number of satellites, given that each trajectory prediction is based on disjoint sets of observations. Nevertheless, as mentioned in Section IV, a large number of collision avoidance constraints (one for each agent) poses threats to the algorithm convergence. All the numerical routines can be optimized for on-board implementation. One important remark is that Recurrent Neural Networks currently lack deep support for the implementation on space-grade boards, thus it is expected that the computational loads results may significantly improve.

VI. Conclusion

This paper presents an innovative strategy for the guidance and control of distributed formation solving the shortcomings that arise from the incomplete representation of the dynamical environment as well as the lack of knowledge of future trajectories of neighbouring satellites during coordinated maneuvers. The trajectory and control is generated using a Model-Based Reinforcement Learning

(MBRL) approach. Two algorithms, namely Inverse Reinforcement Learning (IRL) and Long-Short-Term Memory (LSTM) network, are explored to guarantee collision-free operations. Inverse Reinforcement Learning (IRL) reconstructs the cost function of each agent and predicts trajectories of concurring agents during the reconfiguration. The second method exploits a Long Short-Term Memory recurrent network to capture the dynamics and predict the trajectory. In this way, both natural and thrust dynamics is managed when enforcing the collision-avoidance constraint. Long Short-Term Memory demonstrated more accurate predictions for the short-term horizon whereas Inverse Reinforcement Learning for the longer term. This result leads to the conception of a coupled architecture where the first prediction is given by the Long Short-Term Memory and the successive by the Inverse Reinforcement Learning. The results show that the proposed algorithms perform correctly and solve reconfiguration scenario that are challenging, or even fatal, for traditional algorithms. Being computationally light, online Neural Network aided algorithms can be deployed in micro-satellites, where the computational power is limited. Also, the algorithms adapt to the environment capturing the unmodelled terms delivering successful control where traditional and not adaptive algorithms may fail.

References

[1] Wahl, T., and Howell, K., "Autonomous Guidance Algorithms for Formation Reconfiguration Maneuvers," *AAS/AIAA Astrodynamics Specialist Conference*, Columbia River Gorge, Stevenson, Washington, August 21 - 24, 2017.

[2] Morgan, D., Chung, S. J., and Hadaegh, F. Y., "Model predictive control of swarms of spacecraft using sequential convex programming," *Journal of Guidance, Control, and Dynamics*, Vol. 37, No. 6, 2014, pp. 1725–1740. https://doi.org/10.2514/1.G000218.

[3] Gaudet, B., Linares, R., and Furfaro, R., "Six degree-of-freedom body-fixed hovering over unmapped asteroids via LIDAR altimetry and reinforcement meta-learning," *Acta Astronautica*, Vol. 172, 2020, pp. 90 – 99. https://doi.org/https://doi.org/10.1016/j.actaastro.2020.03.026, URL http://www.sciencedirect.com/science/article/pii/S0094576520301545.

[4] Chu, J., "Dynamics, Distributed Control And Autonomous Cluster Operations Of Fractionated Spacecraft," Ph.D. thesis, TU Delft, 2015. 9789461865113.

[5] Di Mauro, G., Spiller, D., Bevilacqua, R., and Curti, F., "Optimal Continuous Maneuvers for Satellite Formation Reconfiguration in J2-perturbed Orbits," *2018 Space Flight Mechanics Meeting*, Vol. 0216, No. January, 2018, pp. 1–20. https://doi.org/10.2514/6.2018-0216.

[6] Abbeel, P., and Ng, A. Y., "Apprenticeship Learning via Inverse Reinforcement Learning," *Proceedings of the 21 st International Conference on Machine Learning*, Banff, Canada, 2004.

[7] Finn, C., Levine, S., and Abbeel, P., "Guided cost learning: Deep inverse optimal control via policy optimization," *33rd International Conference on Machine Learning, ICML 2016*, Vol. 1, 2016, pp. 95–107.

[8] Ratliff, N. D., Silver, D., and Bagnell, J. A., "Learning to search: Functional gradient techniques for imitation learning," *Autonomous Robots*, Vol. 27, No. 1, 2009, pp. 25–53. https://doi.org/10.1007/s10514-009-9121-3.

[9] Linares, R., and Furfaro, R., "Space Objects Maneuvering Detection and Prediction via Inverse Reinforcement Learning," *Advanced Maui Optical and Space Surveillance (AMOS) Technologies Conference*, 2017, p. 46.

[10] Pasquale, A., Silvestrini, S., Capannolo, A., and Lavagna, M., "Non-Uniform Gravity Field Model on Board Learning During Small Bodies Proximity Operations," *70th International Astronautical Congress (IAC 2019)*, 2019, pp. 1–11.

[11] Pesce, V., Silvestrini, S., and Lavagna, M., "Radial basis function neural network aided adaptive extended Kalman filter for spacecraft relative navigation," *Aerospace Science and Technology*, Vol. 1, 2019, p. 105527. https://doi.org/10.1016/j.ast.2019.105527.

[12] Silvestrini, S., and Lavagna, M., "Neural-aided GNC reconfiguration algorithm for distributed space system: development and PIL test," *Advances in Space Research*, Vol. 67, No. 5, 2021, pp. 1490–1505. https://doi.org/10.1016/j.asr.2020.12.014.

[13] Lippmann, R., *Neural Networks, A Comprehensive Foundation*, Vol. 05, 2005. https://doi.org/10.1142/s0129065794000372.

[14] Silvestrini, S., and Lavagna, M., "Model-based Reinforcement Learning for Distributed Path Planning," *15th Symposium on Advanced Space Technologies in Robotics and Automation*, ESA, ESTEC, Noordwijk, 2019.

[15] Ratliff, N. D., Bagnell, J. A., and Zinkevich, M. A., "Maximum margin planning," *23rd International Conference on Machine Learning*, 2006, pp. 729–736. https://doi.org/10.1145/1143844.1143936.

[16] Taskar, B., Guestrin, C., and Koller, D., "Max-Margin Markov Networks," *Advances in Neural Information Processing Systems 16*, edited by S. Thrun, L. K. Saul, and B. Schölkopf, MIT Press, 2004, pp. 25–32. URL http://papers.nips.cc/paper/2397-max-margin-markov-networks.pdf.

[17] Hochreiter, S., and Schmidhuber, J., "Long Short-Term Memory," *Neural Computation*, Vol. 9, No. 8, 1997, pp. 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735.

[18] Silvestrini, S., Pesce, V., and Lavagna, M., "Distributed Autonomous Guidance , Navigation and Control loop for Formation Flying Spacecraft Reconfiguration," *5th CEAS Conference on Guidance, Navigation and Control*, Milan, 2019, pp. 1–19.

[19] D'Amico, S., and Montenbruck, O., "Proximity Operations of Formation-Flying Spacecraft Using an Eccentricity/Inclination Vector Separation," *Journal of Guidance, Control, and Dynamics*, Vol. 29, No. 3, 2006, pp. 554–563. https://doi.org/10.2514/1.15114, URL http://arc.aiaa.org/doi/10.2514/1.15114.