i am gonna provide you with information for a research paper, we're creating a new algo for swarm robots combining centralized, decentralized algos with deep reinforcement learning. I need you to generate Introduction, abstract & bunch of other things required for a research paper
Information 1:
So for control of the Swarm there are generally two ways, Decentralized Control & Centralized Control.
In Centralized Control, there is a central system to which all the members of the swarm send their data to and it makes decisions for all the members.
In Decentralized Control, robots communicate directly with nearby neighbors or with a subset of the swarm. In such systems, each robot typically has access to information from nearby robots or those within its communication range.
But each method has their drawbacks, so to overcome those we could combine these two methods.
Instead, there can be Centralized controller making decisions but also nearby robots can share data and if the sub swarm deems that the decision made by the Centralized controller is not ideal they can work on their own decision based on the data shared within the sub swarm.
Information 2:

Recently, deep reinforcement learning (RL) methods have been applied successfully to multi-agent scenarios. Typically, the observation vector for decentralized decision making is represented by a concatenation of the (local) information an agent gathers about other agents. However, concatenation scales poorly to swarm systems with a large number of homogeneous agents as it does not exploit the fundamental properties inherent to these systems: (i) the agents in the swarm are interchangeable and (ii) the exact number of agents in the swarm is irrelevant. Therefore, we propose a new state representation for deep multi-agent RL based on mean embeddings of distributions, where we treat the agents as samples and use the empirical mean embedding as input for a decentralized policy. We define different feature spaces of the mean embedding using histograms, radial basis functions and neural networks trained end-to-end. We evaluate the representation on two well-known problems from the swarm literature\Emdashrendezvous and pursuit evasion\Emdashin a globally and locally observable setup. For the local setup we furthermore introduce simple communication protocols. Of all approaches, the mean embedding representation using neural network features enables the richest information exchange between neighboring agents, facilitating the development of complex collective strategies.

Keywords: deep reinforcement learning, swarm systems, mean embeddings, neural networks, multi-agent systems

1Introduction
In swarm systems, many identical agents interact with each other to achieve a common goal. Typically, each agent in a swarm has limited capabilities in terms of sensing and manipulation so that the considered tasks need to be solved collectively by multiple agents.

A promising application where intelligent swarm systems take a prominent role is swarm robotics (Bayındır, 2016). Robot swarms are formed by a large number of cheap and easy to manufacture robots that can be useful in a variety of situations and tasks, such as search and rescue missions or exploration scenarios. A swarm of robots is inherently redundant towards loss of individual robots since usually none of the robots plays a specific role in the execution of the task. Because of this property, swarm-based missions are often favorable over single-robot missions (or, let alone, human missions) in hazardous environments. Behavior of natural swarms, such as foraging, formation control, collective manipulation, or the localization of a common 'food' source can be adapted to aid in these missions (Bayındır, 2016). Another field of application is routing in wireless sensor networks (Saleem et al., 2011) since each sensor in the network can be treated as an agent in a swarm.

1. High state and observation dimensionality, caused by large system sizes.
2. Changing size of the available information set, either due to addition or removal of agents, or because the number of observed neighbors changes over time.

Most current multi-agent deep reinforcement learning methods either concatenate the information received from different agents (Lowe et al., 2017) or encode it in a multi-channel image, where the image channels contain different features based on a local view of an agent (Sunehag et al., 2017; Zheng et al., 2017). However, both types of methods bare major drawbacks. Since neural network policies assume a fixed input dimensionality, a concatenation of observations is unsuitable in the case changing agent numbers. Furthermore, a concatenation disregards the inherent permutation invariance of identical agents in a swarm system and scales poorly to large system sizes. Top-down image based representations alleviate the issue of permutation invariance, however, the information obtained from neighboring agents is of mostly spatial nature. While additional information can be captured by adding more image channels, the dimensionality of the representation increases linearly with each feature. Furthermore, the discretization into pixels has limited accuracy due to quantization errors.

In this paper, we exploit the homogeneity of swarm systems and treat the state information perceived from neighboring agents as samples of a random variable. Based on this model, we then use mean feature embeddings (MFE) (Smola et al., 2007) to encode the current distribution of the agents. Each agent gets a local view of this distribution, where the information obtained from the neighbors is encoded in the mean embedding. Due to the sample-based view of the collected state information, we achieve a permutation invariant representation that is furthermore invariant to the number of agents in the swarm / the number of perceived neighbors.

Mean feature embeddings have so far been used mainly for kernel-based feature representations (Gretton et al., 2012), but they can be also applied to histograms or radial basis function (RBF) networks. The resulting models are closely related to the "invariant model" formulated by Zaheer et al. (2017). However, compared to the summation approach described in their paper, the averaging of feature activations proposed in our approach yields the desired

invariance with respect to the observed agent number mentioned above. To the best of our knowledge, we are the first to use mean embeddings inside a deep reinforcement learning framework for swarm systems where both the feature space of the mean embedding as well as the policy are learned end-to-end.

Policies are learned in a centralized-learning / decentralized-execution fashion fashion, meaning that during learning data from all agents is collected centrally and used to optimize the parameters as if there was only one agent. Nonetheless, each agent only has access to its own perception of the global system state to generate actions from the policy function. We compare our representation to several deep RL baselines as well as to optimization-based solutions, if available. Herein, we perform our experiments both in settings with global observability (i.e., all agents are neighbors) and in settings with local observability (i.e., agents are only locally connected). In the latter setting, we also evaluate different communication protocols (Hüttenrauch et al., 2018) that allow the agents to transmit additional information about their local graph structure. For example, an agent might transmit the number of neighbors within its current neighborhood. Previously, such additional information could not be encoded efficiently due to the poor scalability of the histogram-based approaches.

2.1Deep RL

Recently, there has been increasing interest in deep reinforcement learning for swarms and multi-agent systems in general. For example, Zheng et al. (2017) provide a many-agent reinforcement learning platform based on a multi-channel image state representation, which uses Deep Q-Networks (DQN) (Mnih et al., 2015) to learn decentralized control strategies in large grid worlds with discrete actions. Gupta et al. (2017) show a comparison of centralized, concurrent and parameter sharing approaches to cooperative deep MARL, using TRPO (Schulman et al., 2015), DDPG (Lillicrap et al., 2015) and DQN. They evaluate each method on three tasks, one of which is a pursuit task in a grid world using bitmap-like images as state representation. A variant of DDPG for multiple agents in Markov games using a centralized action-value function is provided by Lowe et al. (2017). The authors evaluate the method on tasks like cooperative communication, navigation and others. The downside of a centralized action-value function is that the input space grows linearly with the number of agents, and hence, their approach scales poorly to large system sizes. A more scalable approach is presented by Yang et al. (2018). Employing mean field theory, the interactions within the population of agents are approximated by the interaction of a single agent with the average effect from the overall population, which has the effect that the action-value function input space stays constant. Experiments are conducted on a Gaussian squeeze problem, an Ising model, and a mixed cooperative-competitive battle game. Yet, the paper does not address the state representation problem for swarm systems.

Sunehag et al. (2017) tackle the “lazy agent” problem in cooperative MARL with a single team reward by training each agent with a learned additive decomposition of a value function based

on the team reward. Experiments show an increase in performance on cooperative two-player games in a grid world. Rashid et al. (2018) further develop the idea with the insight that a full factorization of the value function is not necessary. Instead, they introduce a monotonicity constraint on the relationship between the global value function and each local value function. Results are presented on the StarCraft micro management domain.

2.2Optimization-Based Approaches for Swarm Systems

To provide a concise summary of the most relevant related work, we concentrate on optimization-based approaches that derive decentralized control strategies for the rendezvous and pursuit evasion problem considered in this paper. Ji and Egerstedt (2007) derive a control mechanism preserving the connectedness of a group of agents with limited communication abilities for the rendezvous and formation control problem. The method focuses on high-level control with single integrator linear state manipulation and provides no rules for agents that are not part of the agent graph. Similarly, Gennaro and Jadbabaie (2006) present a decentralized algorithm to maximize the connectivity (characterized by an exponential model) of a multi-agent system. The algorithm is based on the minimization of the second smallest eigenvalue of the Laplacian of the proximity graph. An approach providing a decentralized control strategy for the rendezvous problem for nonholonomic agents can be found in the work by Dimarogonas and Kyriakopoulos (2007). Using tools from nonsmooth Lyapunov theory and graph theory, the stability of the overall system is examined. A control strategy for the pursuit evasion problem with multiple pursuers and single evader that we investigate in more detail later in this paper was proposed Zhou et al. (2016). The authors derive decentralized control policies for the pursuers and the evader based on the minimization of Voronoi partitions. Again, the control mechanism is for high-level linear state manipulation. Furthermore, the method assumes visibility of the evader at all times. A survey on pursuit evasion in mobile robotics in general is provided by Chung et al. (2011).

2.3Analytic Approaches

Another line of work concerned with the curse of dimensionality can be found in the area of multi-player reach-avoid games. Chen et al. (2017), for example, look at pairwise interactions between agents. This way, they are able to use the Hamilton-Jacobian-Isaacs approach to solve a partial differential equation in the joint state space of the players. Similar work can be found in (Chen et al., 2014a, b; Zhou et al., 2012).

3.1Trust Region Policy Optimization

Trust Region Policy Optimization is an algorithm to optimize control policies in single-agent reinforcement learning problems (Schulman et al., 2015). These problems are formulated as Markov decision processes (MDPs), which can be compactly written as a tuple
. After each step, the agent receives a reward , which judges the quality of its decision. The goal of the agent is to find a policy that maximizes the cumulative reward achieved over a certain period of time.

In TRPO, the policy is parametrized by a parameter vector containing the weights and biases of a neural network. In the following, we denote this parametrized policy as

. The reinforcement learning objective is expressed as finding a new policy that maximizes the expected advantage function of the current policy
 linear or neural network baselines are used that are fitted to the Monte-Carlo returns, resulting in an estimate
 for the advantage function. The objective is to be maximized subject to a fixed constraint on the Kullback-Leibler (KL) divergence of the policy before and after the parameter update, which ensures that the updates to the policy parameters
θ
 are bounded, in order to avoid divergence of the learning process. The overall optimization problem is summarized as
.
The problem is approximately solved using conjugate gradient optimization, after linearizing the objective and quadratizing the constraint.

3.2Mean Embeddings
Our work is inspired by the idea of embedding distributions into reproducing kernel Hilbert spaces (Smola et al., 2007) from where we borrow the concept of mean embeddings. A
 can be represented as an element in a reproducing kernel Hilbert space by its expected feature map (i.e., the mean embedding),

, such as Gaussian RBF or Laplace kernels, mean embeddings can be used, for example, in two-sample tests (Gretton et al., 2012) and independence tests (Gretton et al., 2008). A characteristic kernel is required to uniquely identify a distribution based on its mean embedding. However, this assumption can be relaxed to using finite feature spaces if the objective is merely to extract relevant information from a distribution such as, in our case, the information needed for the policy of the agents.

4Deep Reinforcement Learning for Swarms
The reinforcement learning algorithm presented in the last section has been originally designed for single-agent learning. In order to apply this algorithm to the swarm setup, we switch to a different problem domain and show the implications on the learning algorithm. Policies in this context are then optimized in a centralized–learning / decentralized–execution fashion.

4.1Problem Domain
The problem domain for our swarm system is best described as a swarm MDP environment (Šošić et al., 2017). The swarm MDP can be regarded as a special case of a decentralized partially observable Markov decision process (Dec-POMDP) (Bernstein et al., 2002) and is constructed in two steps. First, an agent prototype is defined as a tuple

The model encodes two important properties of swarm networks: First, all agents in the system are assumed to be identical, and accordingly, they are all assigned the same decentralized policy
π
. This is an immediate consequence of the two-step construction of the model, which implies that all agents share the same internal architecture. Second, the agents are only partially informed about the global system state, as prescribed by the observation model
O
. Note that both the transition model and the observation model are assumed to be invariant to permutations of the agents in order to ensure the homogeneity of the system. For details, see (Šošić et al., 2017).

4.2Local Observation Models
The local observation introduced in the last section is a combination of observations
 an agent makes about local properties (like the agent's current velocity or its distance to a wall) and observations of other agents. In order to describe the observation model used for the agents, we use an interaction graph representation of the swarm. This graph is given by nodes
, we require an efficient encoding that can be used as input to a neural network policy. In particular, it must meet the following two properties:

• The encoding needs to be invariant to the indexing of the agents, respecting the unorderedness of the elements in the observation set. Only by exploiting the system's inherent homogeneity we can escape the curse of dimensionality.
• The encoding must be applicable to varying set sizes because the local graph structure might change dynamically. Even if each agent can observe the entire system at all times, the encoding should be applicable for different swarm sizes.
4.3Local Communication Models
In addition to perceiving local state information of neighboring agents, the agents can also communicate information about the interaction graph
$\mathscr{G}$
 (Hüttenrauch et al., 2018). For example, agent
j
 can transmit the number of perceived neighbors to agent
i
. Furthermore, the agents can also perform more complex operations on their local neighborhood graph. For example, they could compute the shortest distance to a target point (such as an evader) that is perceived by at least one agent within their local sub-graph. Hence, by using local communication protocols, observation
4.4Mean Embeddings as State Representations for Swarms
In the simplest case, the local observation

 that agent
i
 receives of agent

. However,

can also contain more complex information, such as relative velocities or orientations. A straightforward way to represent the information set

is to concatenate the local quantities
into a single observation vector. However, as mentioned before, this representation has various drawbacks as it ignores the permutation invariance inherent to a homogeneous agent network. Furthermore, it grows linearly with the number of agents in the swarm and is, therefore, limited to a fixed number of neighbors when used in combination with neural network policies.

To resolve these issues, we treat the elements in the information set
O
i
as samples from a distribution that characterizes the current swarm configuration, i.e.,

. We can now use an empirical encoding of this distribution in order to achieve permutation invariance of the elements of

as well as flexibility to the size of . As highlighted in Section 3.2, a simple way is to use a mean feature embedding, i.e.,

defines the feature space of the mean embedding. The input dimensionality to the policy is given by the dimensionality of the feature space of the mean embedding and, hence, it does not depend on the size of the information set
O
i
any more. This allows us to use the embedding

as input to a neural network used in deep RL. In the following sections, we describe different feature spaces that can be used for the mean embedding. Figure 1 illustrates the resulting policy architectures with further details given in Appendix F.

4.4.1Neural Network Feature Embeddings
In line with the deep RL paradigm, we propose to use a neural network as feature mapping

whose parameters are determined by the reinforcement learning algorithm. Using a neural network to define the feature space allows us to handle high dimensional observations, which is not feasible with traditional approaches such as histograms (Hüttenrauch et al., 2018). In our experiments, a rather shallow architecture with one layer of RELU units already performed very well, but deeper architectures could be used for more complex applications. To the best of our

knowledge, we present the first approach for using neural networks to define the feature space of a mean embedding.

4.4.2Histograms

An alternative feature space are provided by histograms, which can be related to image-like representations. In this approach, we discretize the space of certain features, such as the distance and bearing to other agents, into a fixed number of bins. This way, we can collect information about neighboring agents in the form of a fixed-size multi-dimensional histogram. Herein, the histogram bins define a feature mapping
ϕ
HIST
using a one-hot-coding for each observed agent. A detailed description of this approach can be found in our previous work (Hüttenrauch et al., 2018). While the approach works well in discrete environments where each cell is only occupied by a single agent, the representation can lead to blurring effects between agents in the continuous case. Moreover, the histogram approach does not scale well with the dimensionality of the feature space.

4.4.3Radial Basis Functions

A specific problem of the histogram approach is the hard assignment of agents into bins, which results in abrupt changes in the observation space when a neighboring agent moves from one bin to another. A more fine-grained representation can be achieved by using RBF networks with a fixed number of basis functions evenly distributed over the observation space. The resulting feature mapping
ϕ
RBF
is then defined by the activations of each basis function and can be seen as a “soft-assigned” histogram. However, both representations (histogram and RBF) suffer from the curse of dimensionality, as the number of required basis functions typically increases exponentially with the number of dimensions of the observation vector.

4.5Other Representation Techniques

Inspired by the work of Mordatch and Abbeel (2018), we also investigate a policy function that uses a softmax pooling layer instead of the mean embedding. The elements of the pooling layer

5Experimental Results

Our experiments are designed to study the use of mean embeddings in a cooperative swarm setting. The three main aspects are:

1. How do the different mean embeddings (neural networks, histograms and RBF representation) compare when provided with the same state information content?
2. How does the mean embedding using neural networks perform when provided with additional state information while keeping the dimensionality of the feature space constant?

3. How does the mean embedding of neural network features compare against other pooling techniques?
In this section, we first introduce the swarm model used for our experiments and present the results of different evaluations afterwards. During a policy update, a fixed number of
K
trajectories are sampled, each yielding a return of
5.1Swarm Models
Our agents are modeled as unicycles (a commonly used agent model in mobile robotics; see, for example, Egerstedt and Hu, 2001), where the control parameters either manipulate the linear and angular velocities
and orientation
ϕ
. In case of double integrator dynamics, the agent is additionally characterized by its current velocities. The exact state definition and kinematic models can be found in Appendix A. Note that these agent models are more complex than what is typically considered in optimization-based approaches, which mostly assume single integrator dynamics directly on
$\boldsymbol{x}$
. Depending on the task, we either opt for a closed state space where the limits act as walls, or a periodic toroidal state space where agents exceeding the boundaries reappear on the opposite side of the space. Either way, the state is bounded by

We study two different observation scenarios for the agents, i.e., global observability and local observability. In the case of global observability, all agents are neighbors, i.e.

which corresponds to a fully connected static interaction graph. For the local observability case, we use
Δ
-disk proximity graphs, where edges are formed if the distance
i
–
y
j
)
2
between agents
i
and
j
is less than a pre-defined cut-off distance
d
c
for communication, resulting in a dynamic interaction graph. The neighborhood set of the graph is then defined as

where
. To make the solution compatible to the double integrator agent model, we make use of a PD-controller (see Appendix A for details). The reward function for the problem can be found in Appendix E.1.

We evaluate different observation vectors

which are fed into the policy. To compare the histogram and RBF embedding with the proposed neural network approach, we restrict the basic observation model (see below) to a set of two features: the distance

. This restriction allows for a comparison to the optimization-based consensus protocol, which is based on displacements (an equivalent formulation of distance and bearing). To show that the neural network embeddings can be used with more informative observations, we further introduce an extended set and a communication (comm) set. These sets may include relative orientations

or relative velocities
epending on the agent dynamics), as well as the own neighborhood size and those of the neighbors. An illustration of these quantities can be found in Figure 2.

Refer to caption
Figure 2:Illustration of two neighboring agents facing the direction of their velocity vectors

, along with the observed quantities, shown with respect to agent
i
. The observed quantities are the bearing
.
5.2.1Global Observability
First, we study the rendezvous problem with 20 agents in the global observability setting with double integrator dynamics to illustrate the algorithm's ability to handle complex dynamics. To this end, we compare the performances of policies using histogram, RBF and neural network embeddings on the basic set, as well as neural network embeddings on the extended set. The observations
and bearing
ϕ
i
,
j
. In the extended set, which is processed only via neural network embeddings, we additionally add neighboring agents' relative orientations
θ
i

.
The results are shown in Figure 3(a). On first sight, they reveal that all shown methods eventually find a successful strategy, with the histogram approach showing worst performance. Upon a closer look, it can be seen that the best solutions are found with the neural network embedding, in which case the learning algorithm also converges faster, demonstrating that this form of embedding serves as a suitable representation for deep RL. However, there are two important things to note:

• The differences between the approaches seem to be small due to the wide range of obtained reward values, but the NN+ method brings in fact a significant performance gain. Compared to the NN and RBF embedding, the performance of the learned NN+ policy is
better in terms of the average return of an episode (Figure 3(a)) and almost twice as good (
) in terms of the mean distance between agents at the steady state solution after around 200 time steps (Figure 6(a)). Furthermore, the NN+ embedding reaches the mean distance achieved by the NN and RBF embeddings roughly 20 to 30 time steps earlier, which corresponds to an improvement of
~
25
%
.
• Although the performance gain of NN+ can be partly explained by the use of the extended feature set, experiments with the same feature set using the histogram / RBF approach did not succeed to find solutions to the rendezvous problem; hence, the corresponding results are omitted. The reason is that the dimensionality of the input space scales exponentially for the histogram / RBF approach while only linearly for the neural network embedding, which results in a more compact feature representation that keeps the learning problem tractable.
Together, these two observations suggest that the neural network embedding provides a suitable learning architecture for deep RL, whereas the histogram / RBF approach is only suited for low-dimensional spaces.

Refer to caption
(a)20 agents with global observability
Refer to caption
(b)20 agents with local observability
Figure 3:Learning curves for the rendezvous task with different observation models. The curves show the median of the average return
G
¯
based on the top five trials on a log scale. Legend: NN++: neural network mean embedding of comm set, NN+: neural network mean embedding of extended set, NN: neural network embedding of basic set, RBF: radial basis function embedding of basic set, HIST: histogram embedding of basic set, CONCAT+: simple concatenation of extended set.
Figure 5 shows a visualization of a policy using the neural network mean embedding of the extended set. After random initialization, the agents' locations quickly converge to a single point.

Figure 6 shows performance evaluations of the best policies found with each of the mean embedding approaches. We plot the evolution of the mean distance between all agents over 1000 episodes with equal starting conditions. We also include the performance of the PD-controller defined in Appendix A. It can be seen in Figures 6(a) and 6(c) that the policies using the neural network embeddings decrease the mean distance most quickly and also find the best steady-state solutions among all learning approaches. While the optimization-based solution (PD) eventually drives the mean distance to zero, a small error remains for the learning-based approaches. However, the learned policies are faster in reducing the distance and therefore show a better average reward. Although the optimization-based policy is guaranteed to find an optimal stationary solution, the approach is build for simpler dynamics and hence performs suboptimally in the considered scenario. Note, that the controller gains for this approach have been tuned manually to maximize performance.

In order to show the generalization abilities of the embeddings, we finally evaluate the obtained policies (except for the concatenation) with 100 agents. The results are displayed in Figure 6(b). Again, the neural network embedding of the extended set is quickest in reducing the inter-agent distances, resulting in the best overall performance.