Joint Proceedings of Modellierung 2020 Short, Workshop and Tools & Demo Papers
130 Workshop on Models in AI

Learning of Utility Functions for the Behaviour Analysis in Maritime Surveillance Tasks

Mathias Anneken[1,*] Sebastian Markgraf [1,*] Sebastian Robert[2,3] Jürgen Beyerer[1,2,3]

Abstract: For detecting suspicious activities in the maritime domain such as illegal, unreported and unregulated fishing, it is crucial to counter the increasing amount of available information regarding vessels at sea with sophisticated algorithms and user interfaces. Deep learning and other data driven approaches create good results, but for an operator to be able to intervene suspicious activities, the supporting algorithms must be explainable and transparent. One possibility is to use simulations based on utility functions and behaviour models derived from the field of human behaviour modelling like game theory. Here, an expensive and time-consuming way is to model these utility functions by experts.

As this is not always feasible or the behaviour patterns might not be easily expressed by experts, this work follows a different approach by utilizing inverse reinforcement learning in order to estimate the utility functions for different ship types. For this study, data based on the automatic identification system (AIS) is used for comparing the behaviour of cargo vessels and fishing boats.

Keywords: maximum entropy inverse reinforcement learning; maritime domain; agents; decision support; spatiotemporal data

1 Introduction

The amount of traffic at sea is large and steadily increasing: In 2018 80 % of all international goods were carried by sea. This is only possible due to a fleet of over 94,000 vessels, with a combined tonnage of 1.92 billion dwt. [Un18]. In order to ensure the safe voyage of all vessels, different methods and technologies were introduce to handle the traffic, e. g., the automatic identification system (AIS) for collisions avoidance or analysing the spatiotemporal data of actors.

Due to better quality and availability of sensor systems, the amount of information available for analysing and monitoring the behaviour of entities in surveillance tasks is steadily

[1] Vision and Fusion Laboratory (IES), Karlsruhe Institute of Technology (KIT), mathias.anneken@kit.edu, sebastian.markgraf@student.kit.edu
[2] Fraunhofer IOSB, Karlsruhe, Germany, sebastian.robert@iosb.fraunhofer.de,juergen.beyerer@iosb.fraunhofer.de
[3] Fraunhofer Center for Machine Learning
[*] equal contribution, alphabetical order

increasing. This is also true for the maritime domain: satellites, radio communication, radar and other technologies make it easy to acquire a huge amount of data, which might result in an information overload for operators like coast guards. In order to make the most of the data, it is crucial to support the operators with reliable algorithms. For an operator not only the performance of an algorithm alone is of importance, but also the possibility to grasp an understanding about why the algorithms have decided on a particular result. This means the algorithm has to be transparent and explainable.

One possible approach for an explainable algorithms is to rely on utility functions (including reward and cost functions) for explaining certain behaviour. In [AFB17], a game theoretic approach is used for modelling rational behaviour of different vessels in the maritime domain. The underlying utility functions are hand crafted and need to be specifically tailored to mimic the behaviour of the vessels. As this process is tedious and error-prone, this paper shows a data-driven method to learn an utility function for different vessel types based on inverse reinforcement learning. This method is also applicable to other domains, e. g., car traffic.

2 State of the art

In general, utility functions are used to train an agent for a specific problem. The agent assesses actions by evaluating the utility in order to choose a favourable one. Mostly, this function is tailored by hand and then directly used to learn and predict behaviour of agents. Here, the approach is vice-versa: By learning the utility function, the behaviour will be mimicked. A popular example is inverse reinforcement learning (IRL), as introduced by Ng and Russell [NR00] for learning a reward function from expert trajectories. Some open challenges of the original approach include the insufficient heuristics to decide for a reward function and the limitation to small state spaces.

Wulfmeier et al. [Wu17] learn cost functions directly from sensor inputs. Their own deep maximum entropy IRL algorithm is used in conjunction with a convolutional neural network for autonomous driving in complex, urban environments. This allows them to skip the feature extraction and feed the sensor map around a vehicle directly to the algorithm. The learned cost map is given to a path planning algorithm and trajectories of the vehicle can be calculated corresponding to the current sensor input. Their data consists of over one year of vehicle trajectories.

An extension to maximum entropy IRL given by Ziebart et al. [Zi08] extends the original approach by using the maximum entropy as a deciding factor for a reward function. Maximum entropy describes the amount of assumptions made about the functions. The authors showed that using the least amount of assumptions leads to the most general reward function.

Wulfmeier et al. [WOP15] further extend maximum entropy IRL to use a neural net for the approximation of the reward function and describe the current state of the art approach: Deep maximum entropy IRL.

Hirakawa et al. [Hi18] predict animal trajectories by learning their underlying reward structure. They use GPS data collected from bird flights to create cost maps depending on the current terrain type. Afterwards, these cost maps are used to predict paths with an optimal control algorithm. This approach is novel by using the time as an explicit part of the state. This increases the state space's dimension vastly, but allows them to predict indirect trajectories instead of only direct paths.

3 Foundations

This section will briefly introduce the necessary foundations for implementing IRL. In general, each vessel is modelled in this work as an agent. The agent will recognize its environment and perform actions to influence it as described by Norvig and Russell [RN16]. For this purpose the agent perceives the environment with some kind of sensor. This information is used to build an internal representation of all important entities around the agent. Based on the state, the agent will perform actions to alter the environment. The transitions between states do not have to be deterministic. Norvig and Russell [RN16] differentiate between different kinds of agents. For reinforcement learning, especially the learning agent is crucial.

3.1 Markov Decision Process

A (finite) Markov Decision Process (MDP) models a decision process. It is given by a tuple $(\mathcal{S}, \mathcal{A}, T, R)$ with

- the state space $\mathcal{S}$: Set of all possible states of the environment
- the action space $\mathcal{A}$: Set of all possible actions of the agent
- the reward R: Abstract reward value for each state
- the transition probability T: Probability for state transition depending on the current state and chosen action
- the discount factor γ: Trade-off between near vs. far away rewards

As the MDP fulfils the Markov property, the next step depends only on the current state and chosen action. The solution to a MDP is called a policy π which maps a given state to an (in some measure) optimal action: $\pi : \mathcal{S} \rightarrow \mathcal{A}$. As this paper deals with stochastic policies, it will not map to a single action, but rather a probability distribution over possible actions.

3.2 Inverse reinforcement learning

While reinforcement learning estimates an optimal policy π^* from a given MDP with reward function R, IRL tries to map a given set Z of expert trajectories $\zeta = [s_1, s_2, \ldots, s_n]$ of an unknown optimal policy π^* to the underlying reward function R of the expert. Each trajectory ζ is given by consecutive states s_i. The states are composed of a two-dimensional position and a discrete timestamp $s_i = (x_i, y_i, t_i)$.

Here, the reward function is assumed to be linear in its features:

$$R(\zeta, \theta) = \sum_{s_j \in \zeta} \theta^T f(s_j) \tag{1}$$

with $f(s_j)$ as feature vector and θ^T as the corresponding weight vector. $f(\zeta)$ denotes the sum over all states in a trajectory for a feature:

$$f(\zeta) = \sum_{s_j \in \zeta} f(s_j) \tag{2}$$

Following [Hi18], the expected reward is then defined by

$$\begin{aligned} \mathbb{E}_{p_\pi(\zeta)}[R(\zeta, \theta)] &= \mathbb{E}_{p_\pi(\zeta)}\left[\sum_{s_j \in \zeta} \theta^T f(s_j)\right] \\ &= \theta^T \, \mathbb{E}_{p_\pi(\zeta)}[f(\zeta)] \end{aligned} \tag{3}$$

with $p_\pi(\zeta)$ as the probability of trajectory ζ given the policy π. The expected feature vector $\mathbb{E}_{p_\pi(\zeta)}[f(\zeta)]$ is then matched with the empirical count of observed behaviours Z from the optimal policy π^*:

$$\mathbb{E}_{p_{\pi^*}(\zeta)}[f(\zeta)] = \frac{1}{|Z|} \sum_{\zeta \in Z} f(\zeta) = \bar{f} \tag{4}$$

Following [NR00], this yields a set of solutions containing degenerate reward functions as well, e. g., $R = 0$. Filtering the reward functions through heuristics yields a useful result.

3.3 Maximum entropy inverse reinforcement learning

Maximum entropy inverse reinforcement learning (MaxEntIRL) is an extension to the general IRL approach. As its name suggests, it utilizes the maximum entropy as described by [Ja57]. In [Zi08], Ziebart et al. tackle the problem by defining all possible paths in the MDP as a probability distribution. Although many paths match feature expectations, by using [Ja57] the paths yielding higher rewards should be exponentially more likely to be

chosen. This results in the probability distribution with the partition function $\mathbb{Z}$ as given in [Zi08]:

$$\begin{aligned} \mathbf{P}(\zeta \mid \theta) &= \frac{1}{\mathbb{Z}(\theta)} \exp\left(\sum_{s_j \in \zeta} \theta^T f(s_j)\right) \\ &= \frac{\exp\left(\sum_{s_j \in \zeta} \theta^T f(s_j)\right)}{\int_{\zeta'} \exp\left(\sum_{s'_j \in \zeta'} \theta^T f(s'_j)\right) d\zeta'} \end{aligned} \tag{5}$$

By estimating the log-likelihood, the optimal weights θ^* can be calculated:

$$\theta^* = \underset{\theta}{\operatorname{argmax}}\, L(\theta) = \underset{\theta}{\operatorname{argmax}} \frac{1}{|Z|} \sum_{\zeta \in Z} \log \mathbf{P}(\zeta \mid \theta) \tag{6}$$

The gradient is then given by:

$$\nabla L(\theta) = \bar{f} - \sum_{\zeta \in Z} \mathbf{P}(\zeta \mid \theta) = \bar{f} - \sum_{\zeta \in Z} \sum_{s_j \in \zeta} f(s_j) D_\zeta(s_j) \tag{7}$$

with D_ζ as the expected state visitation frequency. Following [Zi08], the expected state visitation frequency can be calculated by using forward and backward passes. The algorithm calculates backwards from terminal states and keeps track of the probability mass associated with each branch by calculating the partition function. This leads to state frequencies which add up to the total frequency counts.

Following [KW97], the weights are updated using the exponentiated gradient with a learning rate λ

$$\theta \leftarrow \theta e^{\lambda \nabla L(\theta)} \tag{8}$$

until they converge.

4 Preprocessing of AIS data

This sections describes the necessary preprocessing steps regarding the AIS input data in order to be suited for further usage during MaxEntIRL. This includes filtering for the correct location as well as the ship types and the mapping to the grid. The dataset [Ra18] contains AIS data from the Celtic sea, the Channel and the Bay of Biscay in the vicinity of Brest, France, as shown in Fig. 1 (longitude between 10.00 and 0.00, and latitude between 45.00 and 51.00). The data was recorded over a period of six month from 01/10/2015 till 31/03/2016. Additional to the AIS data, complementary material, like lists of ports, weather and ocean conditions, and fishing areas, is attached. The AIS data can be split into dynamic (18.648.556 decoded messages) and static (1.032.187 decoded messages) parts.

The dynamic part consists of the position, speed over ground, true heading, course over ground, rate of turn, and navigational status. The static part includes the Maritime Mobile Service Identity (MMSI), ship type, destination, and draught.

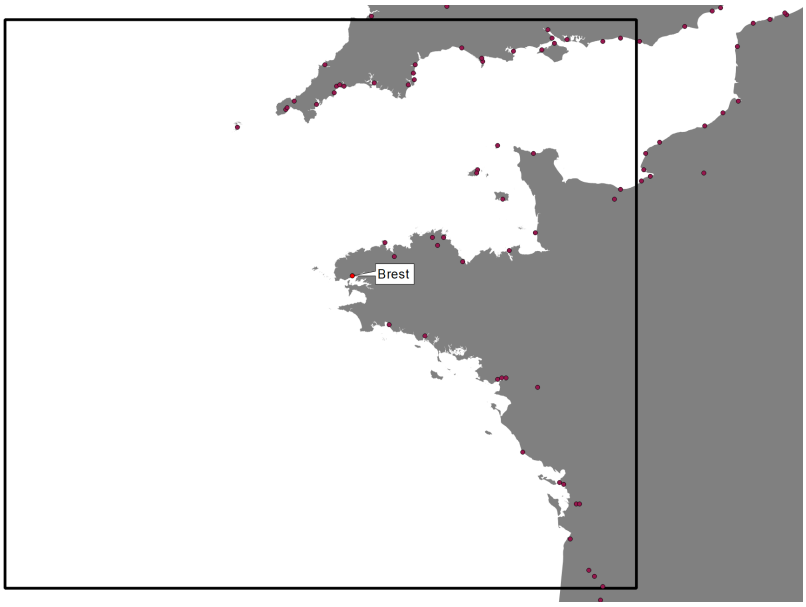


Fig. 1: The area around Brest indicated by the black rectangle (longitude between 10.00 and 0.00, and latitude between 45.00 and 51.00) is used. The dots indicate ports as listed in the World Port Index.

4.0.1 Filtering for ship type

As only fishing boats and container vessels are analysed, a filter is used on the static data.

4.0.2 Splitting by length

As the complete trajectories of each vessel is not suitable for the learning task, the trajectories will be split by using their length as discriminating factor. Training on shorter trajectories will increase the learning speed due to the smaller state space and therefore faster state visitation calculation.

Furthermore, the positional information provided by AIS data can be quite noisy as described in [SA17]. This low reliability regarding the position can be compensated for in many cases by splitting the trajectory based on some threshold for the length. Often the position errors span great distances and thus will be split into shorter trajectories. Too short trajectories will be discarded in the last step.

4.0.3 Splitting by time

As the input data is partly from a port area, many vessels are either anchored or moored, which implies, that they will not move in a meaningful way for discerning their type: The position will be centred around a single point (in port) or on a circle (anchorage area). Thus considering the time dimension including the vessels in port will just create longer trajectories. By applying a threshold on the time, the trajectories are split in ports (and general stopping manoeuvres).

4.0.4 Filtering rivers

As vessels on rivers are not free in their navigation, the learning of their decision process will not yield additional information on their contribution to the utility function. By using [WS96] as database for geo-referenced information on rivers, the trajectories which have more than 80 % of their path close to a river will be removed.

4.0.5 Mapping to grid

Using a continuous state space would imply a huge state space resulting in challenges regarding the computability. Thus, the state space is reduced by discretization the position.

4.0.6 Split round trips

Predicting trajectories performing round trips (start and end point are nearly the same) poses a challenge to MaxEntIRL. The logical path in these cases would be the direct path between start and goal, nearly without any movement at all. Therefore, these trajectories are split into two different trajectories.

4.0.7 Filtering short trajectories

Only longer trajectories covering a certain number of grid cells will be used in the evaluation. Therefore, all trajectories shorter than a threshold are omitted.

5 Creating feature maps

For the analysed areas, MaxEntIRL needs to be fed with feature maps representing certain properties of the environment. The feature maps will be used to estimate the utility for the

agent based on Equation (1). All different types of feature maps are shown in Fig. 2. As the traffic on sea is analysed, geographical information like the coastline is crucial. Here, a shapefile from [WS96] is used. A shapefile will provide geo-referenced geometrical objects for describing e. g., the outlines of continents and islands or the path of rivers.

5.0.1 Constant Feature Map

In order to punish longer trajectories, a constant feature map as depicted in Fig. 2a is used. By increasing the weight for this feature, the prediction will prefer shorter trajectories.

5.0.2 Coast Feature Map

A map of the coast is generated by using the shapefile. The coastline is directly mapped to the grid. The resulting coast feature is shown in Fig. 2b. All landmasses have a constant cost in order to avoid (fatal) collisions.

5.0.3 Special Feature Maps

Some other special feature maps are generated for fishing areas and traffic separation areas. The fishing areas are bundled in [Ra18]. Originally they were released by the European Commission Joint Research Centre[4]. The traffic separation scheme denotes the areas, the normal sea traffic are not allowed to enter, or it has to follow a specific direction. Here, OpenSeaMap[5] data is used.

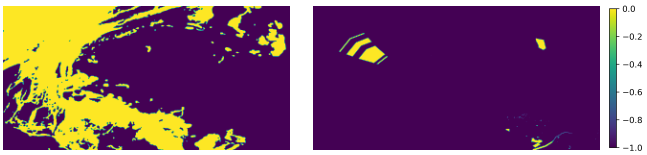
5.0.4 Distance to Coast Maps

Several other coastline related features are derived. First of all, the distance to the coast is used resulting in a feature map rewarding to keep greater distance to the coast. On the distance an exponential function with three different variances $\sigma^2 = \{1, 4, 10\}$ is applied. This results in a total of three maps. One for $\sigma^2 = 1$ is given in Fig. 2e.

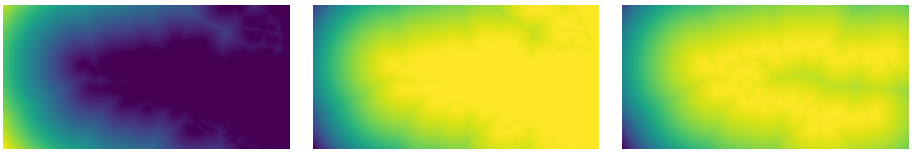
As many vessels try to keep a certain distance from the coast, another feature based on the inverse of the first class of distance to coast maps is introduced, again with the same three variances. The feature map is shown in Fig. 2f.

[4] https://bluehub.jrc.ec.europa.eu/webgis_fish
[5] https://www.openseamap.org

(a) Constant. (b) Coast.



(c) Fishing areas. (d) Traffic Separation Areas.



(e) Distance to coast for $\sigma^2 = 10$. (f) Distance to coast inverted for $\sigma^2 = 1$. (g) Distance to harbour for $\sigma^2 = 1$.

Fig. 2: All generated feature maps.

5.0.5 Distance to Harbour Maps

The last class of feature maps is based on the distance to harbours. It is depicted in Fig. 2g. The generation is similar to the other distance related features. The idea is to specify the distance a vessel likes to keep from the nearest harbour.

6 Evaluation

This section discusses the results for different ship types, namely container vessels and fishing boats.

6.1 Resulting Weights

In Fig. 3, the different weights for container vessels and fishing boats are depicted in blue and orange. For both types, the weights are quite similar: First of all, the greatest weight for both types is on the coast feature map (Fig. 2b), in order to avoid trajectories over land. The

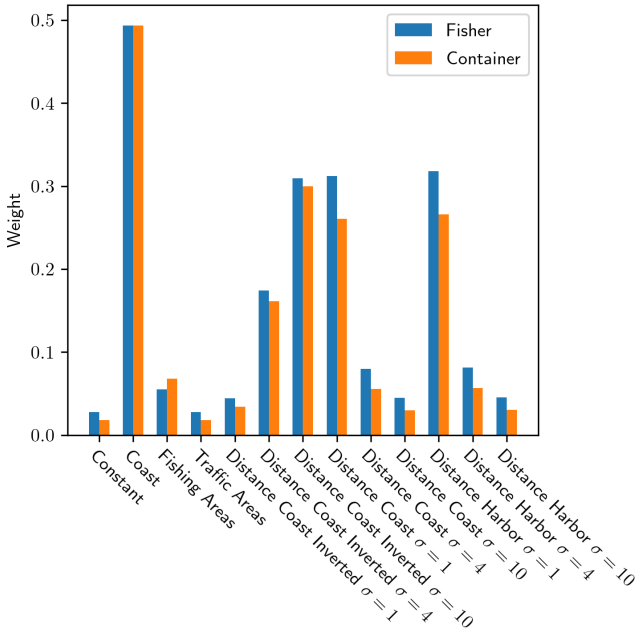


Fig. 3: Resulting weights for the different feature maps.

second highest weight is on the different distance maps. Possibly in order to keep a safe distance to the coast.

The feature map for the traffic separation scheme (Fig. 2d) has only a small weight. The problem here is, that there is only one small traffic separation scheme in the used database. Thus, the small weight is of no surprise. More unexpected is the small weight on the constant feature map (Fig. 2a), as this feature was intended for ensuring shorter trajectories. Probably, the combination of other features, which are constant over larger parts of the area, has this specific effect.

By combining the weighted feature maps utilizing Equation (1), an estimation for a static reward function can be calculated. This map is depicted for the different types in Fig. 4. As expected with the similarity of the weight distribution between the two types, these maps look quite alike.

6.2 Testing the Utility Function

The section before is giving a qualitative assessment of the resulted training. At first glance, it seems to match the general expectations. For a quantitative evaluation the utility function

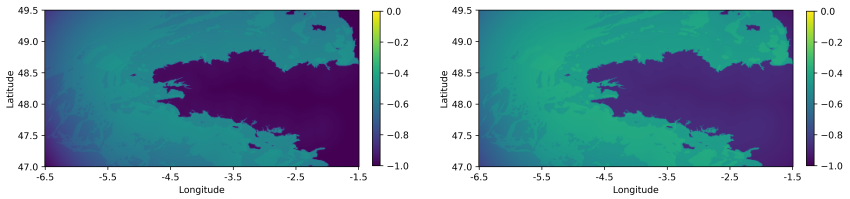


Fig. 4: Resulting weighted combination of feature maps for fishing boats (left) and container ships (right).

is used for predicting actual trajectories. For this prediction, the start and end state of a real trajectory is used. The trajectory between these two states is estimated by calculating the best policy using the learned reward functions. For each step, the action with the highest probability is taken until the end state is reached. This will result in an indirect trajectory. For comparison, a linear interpolation between the start and end state is used. This will always result in the shortest path between these two states, but it might not respect land masses which are not traversable.

For comparing the distance between the predictions based both on the learned utility function and the linear interpolation and the ground truth, dynamic time warping (DTW) is used in order to account for the difference in timing. As a distance metric the euclidean distance is used.

In Tab. 1 the resulting distances for the different vessel types are shown. For fishing boats, the MaxEntIRL predictions yield better results compared to the linear interpolation. This means in average MaxEntIRL predictions are closer to the original trajectory and thus mimic the behaviour better than the linear interpolation. One major advantage is the avoidance of land as seen e. g. in Fig. 5. Thus, the algorithm has learned that fishing boats want to keep a certain distance to the coast.

For the container vessels the results of MaxEntIRL are slightly worse compared to the linear interpolation. One reason could be the often straight and direct routes of the container vessels far away from the coast. Thus, by just following a clear line, the approximation is already quite near to the original trajectory. The MaxEntIRL seems to be at an disadvantage for this task.

Tab. 1: Resulting distances to the original trajectory using dynamic time warping.

Method	Fisher	Container
Linear interpolation	3.9 ± 2.7	1.0 ± 0.7
MaxEntIRL	2.0 ± 1.2	1.9 ± 0.9

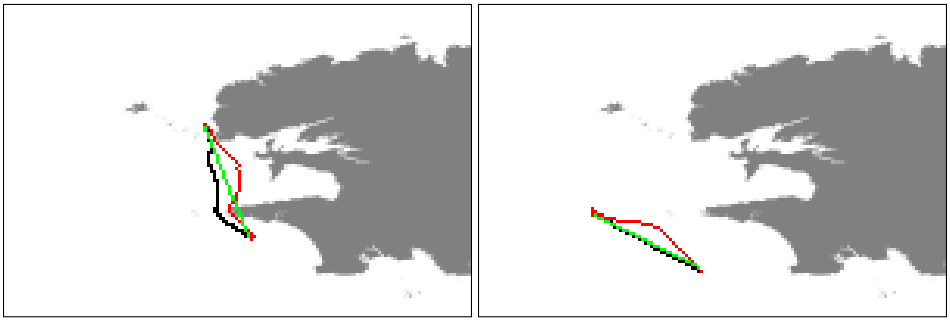


Fig. 5: Trajectory prediction for a fishing boat (left) and container vessel (right). The red line is the prediction, green a linear interpolation, and black the original trajectory. Each pixel represents one grid cell.

7 Results

Utility functions for two different vessel types in the maritime domain are learned by using MaxEntIRL. These functions are successfully used for predicting trajectories. The training is based on filtered AIS data together with different feature maps created based on geographical and nautical information.

This method was used for fishing boats and container vessels. For both types, feature maps representing the preferences for each type are generated. The results for fishing boats are superior to the comparison algorithm. The results for the direct trajectories of container vessels are less promising. One reason might be the linear combination of the feature maps together with only considering a static environment. Later means, that e. g., other vessels are not considered.

As the area is discretized, thin landmasses might be mapped to only a few cells. As traversing the grid diagonally is possible, the predicted trajectory might be infeasible. This has nothing to do with the learning algorithm but rather with the limited resolution which is used in order to have a manageable state space size.

8 Conclusion

This work explored the possibility of learning utility functions for maritime agents instead of crafting them manually. Here, an approach for learning utility functions for birds was transferred to the maritime context. This results shows the feasibility of learning utility functions for maritime agents.

While the first results look promising, the actual usage of this method in a real operational environment is still in the future. The shown method is limited in its learning capabilities and

thus will not sufficiently convey the necessary information to operators to really support their work. Nevertheless, by further extending this version and exploring different approaches, a better understanding of the behaviour of maritime agents can be achieved.

9 Future work

The usage of only linear combination of feature maps drastically limits the possibilities of the algorithm. Further, the algorithm itself chooses to only fully use a limit number of the provided features. Another future challenge is the inclusion of dynamic features, i. e., the position data of other vessels, weather conditions, sea state, etc. In reality the operator of a vessel will take all these dynamic factors into account for making decisions.

Further, only data from a small area is considered as a first application. Transferring the method to other areas in the maritime domain or even other domains might give more inside into the possibilities of the learned utility functions. In order to enhance these utility function, the algorithm needs to be extended, e. g., by using the deep learning approach described by Wulfmeier et al. in [WOP15]. This (and other deep learning approaches) will enable the usage of non-linear functions. Further, the usage of dynamic data instead of the static maps will be possible.

In order to have a better comparison, in future work, other path planing algorithms like A* or similar could be used to avoid obstacles. To improve the results for container vessels, deploying a deep approach, which can leverage specific areas the container vessels use, would be needed.

The trajectory estimation alone is not enough to detect scenarios of interest, e. g. illegal, unreported and unregulated fishing or the smuggling of contraband. For these kind of situations, the prediction together with the utility function itself could be used for detecting the deviation of vessels from their normal behaviour.

Acknowledgment.

Underlying projects to this article are funded by the WTD 81 of the German Federal Ministry of Defense. The authors are responsible for the content of this article. This work was developed in the Fraunhofer Cluster of Excellence “Cognitive Internet Technologies”.

Bibliography

[AFB17] Anneken, Mathias; Fischer, Yvonne; Beyerer, Jürgen: A Multi-agent Approach to Model and Analyze the Behavior of Vessels in the Maritime Domain. SCITEPRESS, pp. 200–207, 2017.

[Hi18] Hirakawa, Tsubasa; Yamashita, Takayoshi; Tamaki, Toru; Fujiyoshi, Hironobu; Umezu, Yuta; Takeuchi, Ichiro; Matsumoto, Sakiko; Yoda, Ken: Can AI predict animal movements? Filling gaps in animal trajectories using inverse reinforcement learning. Ecosphere, 9(10):e02447, 2018.

[Ja57] Jaynes, Edwin T.: Information theory and statistical mechanics. Physical review, 106(4):620, 1957.

[KW97] Kivinen, Jyrki; Warmuth, Manfred K.: Exponentiated Gradient versus Gradient Descent for Linear Predictors. Information and Computation, 132(1):1 – 63, 1997.

[NR00] Ng, Andrew Y.; Russell, Stuart J.: Algorithms for Inverse Reinforcement Learning. In: Proceedings of the Seventeenth International Conference on Machine Learning. ICML '00, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp. 663–670, 2000.

[Ra18] Ray, Cyril; Dréo, Richard; Camossi, Elena; Jousselme, Anne-Laure: Heterogeneous Integrated Dataset for Maritime Intelligence, Surveillance, and Reconnaissance. 2018.

[RN16] Russell, Stuart J; Norvig, Peter: Artificial intelligence: a modern approach. Malaysia; Pearson Education Limited,, 2016.

[SA17] Sotirov, Svetoslav; Alexandrov, Chavdar: IMPROVING AIS DATA RELIABILITY. 2017.

[Un18] United Nations Conference on Trade and Development: , Review of Maritime Transport 2018, 2018.

[WOP15] Wulfmeier, Markus; Ondruska, Peter; Posner, Ingmar: Deep Inverse Reinforcement Learning. CoRR, abs/1507.04888, 2015.

[WS96] Wessel, Paul; Smith, Walter: A global, self-consistent, hierarchical, high-resolution shoreline database. Journal of Geophysical Research, 101:8741–8743, 04 1996.

[Wu17] Wulfmeier, Markus; Rao, Dushyant; Wang, Dominic Zeng; Ondruska, Peter; Posner, Ingmar: Large-scale cost function learning for path planning using deep inverse reinforcement learning. The International Journal of Robotics Research, 36(10):1073–1087, 2017.

[Zi08] Ziebart, Brian; Maas, Andrew; Bagnell, J. Andrew; Dey, Anind K.: Maximum Entropy Inverse Reinforcement Learning. pp. 1433–1438, 2008.