Motion Prediction Based on Multiple Futures for Dynamic Obstacle Avoidance of Mobile Robots

Ze Zhang, Emmanuel Dean, Yiannis Karayiannidis, Knut Åkesson

***Abstract*— The ability to decide and adjust actions according to motion prediction of dynamic obstacles offers a flexible planning scheme and ampler reaction time to avoid potential impact. Prediction-based collision avoidance implies a two-stage decision-making process from motion prediction to action planning. One of the challenges in motion prediction is the movements of objects are usually non-deterministic and governed by multimodal models. Many studies have been made on motion prediction of dynamic obstacles and action planning for mobile robots separately. The objective of this work is to explore their coherence in terms of multiple future predictions by combining a data-driven motion prediction approach with a model-based control strategy. More specifically, we integrate motion prediction from deep learning models, Mixture Density Networks (MDNs) with a Non-linear Model Predictive Control (NMPC) framework. The deep learning models produce the multimodal probability distribution of future positions of dynamic obstacles, which is utilized by the MPC controller as a constraint. We show via simulation that the selected model provides valid predictions of motion in a dynamic environment. The prediction result endows the controller with the capability to avoid dynamic obstacles in advance.**

I. INTRODUCTION

Humans can adjust their actions to avoid collisions based on instantaneous processing of observations and past experiences. On the contrary, a mobile robot cannot achieve collision avoidance without a sophisticated process, which includes information collection, motion prediction, path planning, online trajectory correction, and control. In this work, we consider a setup where mobile robots acquire global information using multiple cameras mounted in the ceiling of a factory or warehouse, as shown in Fig. 1. Top-view camera configurations for mobile robots have previously been presented in [1], [2]. A vision-based omniscient system, consisting of multiple top-view cameras in the ceiling and processing units, provides an extensive global vision to mobile robots and thus enables them to plan and behave in a proactive manner. In this work, we focus on how to handle the information provided by a multi-camera setup in a deep learning fashion to achieve motion prediction of different dynamic obstacles and guide a model predictive controller for online trajectory correction.

Path planning algorithms for mobile robots operating in static environments are quite mature and use grid-based or graph-based maps [3], [4], while planning in case of dynamic obstacles, such as humans, is still an active research topic due to the high complexity and uncertainty of the movement and intention of obstacles.

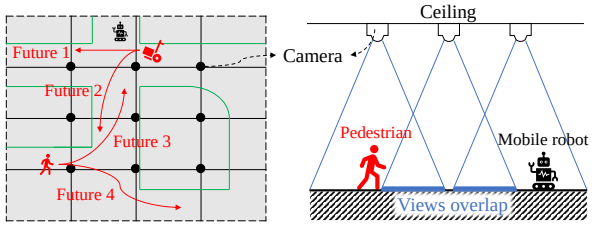


Fig. 1: *Research background: A top-view vision system with multiple cameras with an overlapping field-of-view ensuring no blind areas. The vision information from cameras is used to predict motions of dynamic obstacles such as pedestrians.*

Motion prediction of dynamic obstacles is important for collision avoidance of mobile robots. In [5], a thorough overview of motion prediction depicts the development of relevant algorithms. Most studies focus on the prediction of a single hypothesis or probability distribution that is unstable or not comprehensive enough. Recently, deep learning methods [6], [7], [8] have been explored to overcome the rising complexity of motion prediction tasks in the dynamic environment. Multiple hypothesis prediction [9] uses a special meta loss for Convolutional Neural Networks (CNNs) to learn multiple futures. In [6], the authors combined the meta loss and MDNs [10] to overcome the limitations of MDNs, which are model collapse and hypotheses degeneration. A related approach is model-based methods, such as building Long Short-Term Memory (LSTM) networks as models for pedestrians [7]. The usage of deep learning makes the prediction of non-deterministic motions of dynamic obstacles more robust and brings the extension of predicting the potential multiple choices of future motions.

Given a prediction of future motions of the uncontrolled agents, the problem is to plan trajectories such that the robot can visit target destinations without any collisions. Existing works either regard dynamic obstacles as temporary static obstacles and execute a path replanning with global path planning algorithms such as the rapidly-exploring random tree algorithm [11], or simplify dynamic obstacles into plain models such as the constant velocity (CV) model with local trajectory planning algorithm such as [12], [13]. In recent years, as the computing power rises prominently, the Model Predictive Control (MPC) method has become feasible for

This work is published at the 17th IEEE International Conference on Automation Science and Engineering, CASE 2021.

*We gratefully acknowledge financial support from Chalmers AI Research Centre (CHAIR) and AB Volvo (Project ViMCoR) and the support from Per-Lage Götvall at Volvo Group Truck Operation.

The authors are with Division of Systems and Control, Electrical Engineering, Chalmers University of Technology, Sweden {zhze, deane, yiannis, knut}@chalmers.se

online trajectory correction [14]. However, a similar deficiency is that motion predictions of dynamic obstacles are based on simple motion models that do not take multiple future motion predictions into account.

In this paper, we propose a predictive dynamic obstacle avoidance method for mobile robots, integrating motion prediction information, where multiple future positions are possible, as MPC constraints to obtain a wider prediction scope for the controller. This method yields a global predictive control behavior in terms of dynamic obstacles by combining Mixture Density Networks (MDNs) for estimation of multiple future positions, with the Model Predictive Control (MPC) approach. We show that without the multiple future predictions some potential collisions cannot be foreseen thus leading to safety issues. Using simulation, we present that by applying our method the mobile robot can avoid collisions with dynamic obstacles which have complex non-deterministic motion patterns. The approach is evaluated against motion models computed using a linear Kalman filter assuming constant velocity.

II. Preliminaries

We start by introducing Mixture Density Networks (MDNs) that use learning-based methods to estimate multiple future possible positions, of the uncontrolled agents, for different time-steps into the future. We then introduce the nonlinear model predictive control methods that we use to generate a trajectory-based on estimated future positions of the uncontrolled agents.

A. Mixture Density Network

Mixture Density Networks (MDNs) [10] aim at establishing the potential bond between the d-dimensional input $\boldsymbol{x} \in \mathbb{R}^d$ and the c-dimensional output $\boldsymbol{y} \in \mathbb{R}^c$ by estimating the output in a form of a probability distribution $p(\boldsymbol{y}|\boldsymbol{x})$. MDNs presume the targeted output is a mixture density model:

$$p(\boldsymbol{y}|\boldsymbol{x}) = \sum_{i=1}^{m} \alpha_i(\boldsymbol{x})\phi_i(\boldsymbol{y}|\boldsymbol{x}) \tag{1}$$

consisting of m components ϕ_i weighted through $\alpha_i(\boldsymbol{x})$. For convenience, the subscript i will be omitted if there is no ambiguity. In [10], each component $\phi(\boldsymbol{y}|\boldsymbol{x})$ is a Gaussian distribution and the overall distribution is a Gaussian Mixture Model (GMM). However, other parameterized probability distributions can also be employed. For example, in [6], the authors implemented MDNs with Laplace distributions.

MDNs obtain the output through two steps as shown in Fig. 2. The Body is essentially a feature extractor. Fig. 2 contains a multi-layer perceptron but it can be replaced by other models that generate a characteristic vector $\boldsymbol{z}$. The Head transfers the characteristic vector to the probabilistic vector $\boldsymbol{p}$ containing all the probabilistic parameters of the selected distribution model such as GMMs. Here, the head is the exact part of MDNs.

$$\phi_{GMM}(\boldsymbol{y}) = \frac{1}{\sqrt{(2\pi)^c|\boldsymbol{\Sigma}|}} \exp\left(-\frac{(\boldsymbol{y}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{y}-\boldsymbol{\mu})}{2}\right) \tag{2}$$

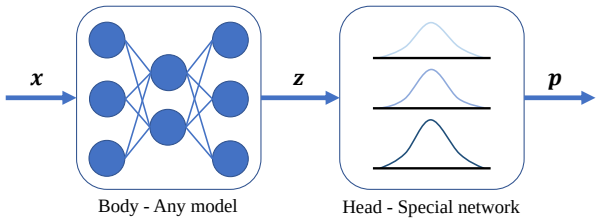


Fig. 2: *The concept of MDNs. The Body takes the input $\boldsymbol{x}$ and generate a characteristic vector $\boldsymbol{z}$. The Head transfers $\boldsymbol{z}$ into the probabilistic form $\boldsymbol{p}$.*

In spite of various types of probability distributions, the Gaussian distribution (2) is one common choice as the component in MDNs. In this work, we only use Gaussian components, hence hereafter MDNs are MDNs with GMMs. In a GMM, ϕ as in (2) is parameterized by the mean value $\boldsymbol{\mu}(\boldsymbol{x})$ and standard deviation $\boldsymbol{\sigma}(\boldsymbol{x})$. The covariance matrix $\boldsymbol{\Sigma}$ is assumed to be a diagonal matrix and represented as a vector $\boldsymbol{\sigma}$ consisting of all the diagonal elements in $\boldsymbol{\Sigma}$. This is not necessarily true in reality but it is a reasonable simplification to reduce the complexity of the model. Together with the weight vector $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_m)^T$, $\boldsymbol{p} = (\boldsymbol{\alpha}^T, \boldsymbol{\mu}^T, \boldsymbol{\sigma}^T)^T$ forms the output of MDNs.

The Head guarantees that the final output satisfying certain conditions via special layers, which produces a valid probability representation. GMMs require three parts: the mean, variance, and weight for each component. There are properties for these probabilistic parameters to hold. The sum of all weights should be one and this can be achieved by applying a Softmax layer [10]. There is no special requirement for the mean value; therefore a regular linear layer is appropriate. As for the variance, since it is a scale parameter, taking the exponential of the characteristic vector is effective.

B. Non-linear Model Predictive Control

Model Predictive Control (MPC) solves constrained optimization problems given a state-space representation of the controlled plant. Non-linear MPC (NMPC), as a variant of MPC, is normally used when the plant is non-linear. Equations (3a), (3b), (3c) are a regular mathematical form of MPC [15], and the hat notation is used to denote the estimated state and control signals:

$$\min_{\hat{\boldsymbol{u}}_0,\ldots,\hat{\boldsymbol{u}}_{N-1}} V_N(\hat{\boldsymbol{s}}, \hat{\boldsymbol{u}}) = \sum_{j=0}^{N-1} l(\hat{\boldsymbol{s}}_j, \hat{\boldsymbol{u}}_j) + V_f(\hat{\boldsymbol{s}}_N) \tag{3a}$$

$$\text{s.t.} \quad \hat{\boldsymbol{s}}_{k+1} = f(\hat{\boldsymbol{s}_k}, \hat{\boldsymbol{u}}_k), \; k = 0, 1, \ldots, N-1 \tag{3b}$$

$$\boldsymbol{g}(\hat{\boldsymbol{s}}, \hat{\boldsymbol{u}}) \leq 0 \tag{3c}$$

where $l(\cdot)$ is the loss function for the state and input in every step, $V_f(\cdot)$ is a final cost for the last state, $f(\cdot)$ is the motion model of the controlled mobile robot, and $\boldsymbol{g}(\cdot)$ models the constraints. The mathematical form shows three superiorities: NMPC deals with non-linear models, it considers all states

and input within the prediction horizon, and it includes constraints. In our case, the *control horizon* is set to be equal to the *prediction horizon*, and will be denoted by N. $\boldsymbol{s}$ is the state vector and $\boldsymbol{u}$ is the control sequence.

In this work, we use a differential drive robot but the non-linear setting allows other types of non-holonomic and holonomic robots, and for control of a fleet of robots that have, for example, distance constraints between them. We use the Proximal Averaged Newton-type method for Optimal Control (PANOC) [16], [17] for solving the NMPC problems, as this method is able to efficiently and in a robust way solve the specified NMPC problems.

III. Approach

Our approach starts with the training of MDNs. The trained neural network produces multimodal predictions in a form of GMMs. We treat the output from the MDN as individual Gaussian distributions and use ellipsoids to represent them. These ellipsoids are then included as constraints in the NMPC formulation.

A. Motion learning and prediction

MDNs are numerically vulnerable under high dimensional input [9], [18], thus it is important to choose the form of input carefully as well as the architecture of the Body, and implement proper preprocessing. To reduce the computation complexity, and reserve more time for trajectory correction, we select an MLP model as the Body. This means the input to the network is a numerical vector. There are some common options to select elements for the input, such as position, velocity, orientation, etc. Our input is composed of three parts. The leading part is the positions of the target, including the coordinates of current and past κ time instants. The later element is the prediction time offset $T = 1, 2, \ldots, T_{max}$, indicating how far we want to obtain the prediction into the future. The last element is the type of this object that is defined by a non-negative integer. To mitigate the risk of model collapse, as discussed in [18], all values about positions should be normalized. We use another alternative that is adding Batch Normalization (BN) layers in the Body to achieve the same effect. The dimension of the output $\boldsymbol{p}$ is determined by the preset number of components times five (one weight, two means, and two variances). Fig. 3 shows the overall architecture of our MDN.

Furthermore, MDNs are also subject to degenerate predictions [6], [9]. The number of hypotheses is a special hyper-parameter in MDNs and set to be a reasonably sufficient large value and usually larger than the actual number of modalities, or motion modes in our case. Once all modes are occupied, other hypotheses will not be updated anymore, thus degenerate. The degenerate components are regarded as redundant information and abandoned before the next step. Assuming $\tilde{m}$ components stay, the weight vector $\boldsymbol{\alpha}$ needs to be renormalized. Denote the weight vector after renormalization as $\tilde{\boldsymbol{\alpha}} = (\tilde{\alpha}_1, \tilde{\alpha}_2, \ldots, \tilde{\alpha}_{\tilde{m}})^T$. Finally, all position-related outputs need to be restored to the original size before the rescaling if it is normalized ahead for training.

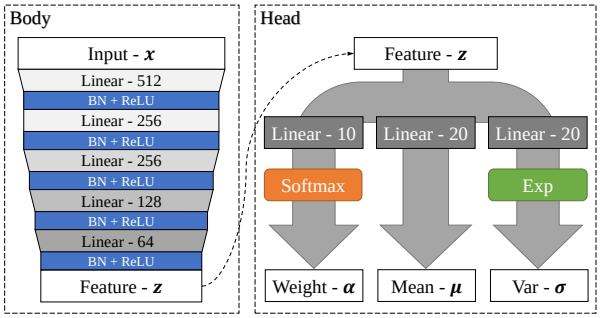


Fig. 3: *The architecture of our MDN. The left part is the body of MDN taking the input vector $\boldsymbol{x}$, which is a MLP consisting of a stack of linear layers followed by the Batch Normalization (BN) layers and ReLU activation. The right part is the head transferring the characteristic vector $\boldsymbol{z}$ from the body into the distribution parameters $\boldsymbol{\alpha}$, $\boldsymbol{\mu}$, and $\boldsymbol{\sigma}$. We use 10 hypotheses. The dimension of the output, p is $10 \cdot 5$.*

B. Prediction Modelling

Through the MDN, the predictions of the future positions of the targeted object are obtained in the form of GMMs. However, it is intricate to formulate the Gaussian mixture density as an MPC constraint due to the irregular contour. We consequently treat the output as several individual Gaussian distributions, and every single distribution can be described by an ellipse with the mean value as the centre while the larger and shorter variances are defined as the major and minor axes, respectively. The weight for each component is used to adjust the extension of the ellipse's border.

To model this information as control constraints, all the ellipses require expansions. The axis parameters of ellipses are scaled from σ to $(2+\tilde{\alpha})\sigma$, where $\tilde{\alpha}$ is the corresponding renormalized weight. This ensures more than 95% confidence in covering the true position. The prediction from MDNs only focuses on the point-mass model of the object without considering its shape. Therefore, every ellipse needs to be enlarged according to the size of the dynamic obstacles which is assumed to be known.

C. Trajectory planning

The modelling for mobile robots adopts the kinematic scheme, defining the state vector $\boldsymbol{s} = (s_x, s_y, \theta)^T \in \mathbb{R}^3$ and the input $\boldsymbol{u} = (v, \omega)^T \in \mathbb{R}^2$:

$$f(\boldsymbol{s}_k, \mathbf{u}_k) = \begin{bmatrix} s_{x,k} + v_k \cos(\theta_k)\Delta t_s \\ s_{y,k} + v_k \sin(\theta_k)\Delta t_s \\ \theta_k + \omega_k \Delta t_s \end{bmatrix} \tag{4}$$

The input uses linear speed $v \in \mathbb{R}$ and angular speed $\omega \in \mathbb{R}$. The state vector $\boldsymbol{s}$ contains the 2D position coordinates (s_x, s_y) and the orientation θ of the target. k is the time step and Δt_s is the sampling time. The cost function [19] consists of three items: the cross-track error J_{cte} weighted by q_{cte}, the speed deviation cost J_v weighted by q_v, and the

acceleration cost J_{acc} weighted by q_{acc}. For the instant k, $J_{cte,k}$ calculate the shortest distance between the predicted state $\hat{\boldsymbol{s}}_k$ and the reference path. $J_{v,k}$ penalizes the deviation of speed from the reference speed. $J_{acc,k}$ penalizes on the difference of input, $\Delta\hat{\boldsymbol{u}} = \hat{\boldsymbol{u}}_k - \hat{\boldsymbol{u}}_{k-1}$.

Let the set $\mathcal{O}_l$ contain all static obstacles, and let r be the safe distance that mobile robots should keep from static obstacles, for example, set r as the maximum distance from the center of the robot to any point on the robot. Let $\mathcal{O}_{d,k}$ be the set of all dynamic obstacles at time-step k. For a dynamic time-step k, this draws an area $\mathcal{D}_k$ in (5) where mobile robots are forbidden to enter. For convenience, we omit subscripts k. A dynamic obstacle $\boldsymbol{o}$ has the form of a ellipse defined by the centre (o_x, o_y), axes (o_w, o_h), and a rotation angle o_α. We assume the predicted distribution has a diagonal covariance matrix, thus the heading o_α can be ignored. So $\boldsymbol{o} = (o_x, o_y, o_w, o_h)^T \in \mathbb{R}^4$.

$$\mathcal{D} = \{(x, y) \mid \forall \boldsymbol{o} \in \mathcal{O}_d, \frac{(x - o_x)^2}{o_w^2} + \frac{(y - o_y)^2}{o_h^2} \leq 1\} \quad (5)$$

The trajectory planning task can be formulated as the following optimization problem. Given $\boldsymbol{s}_0$ as the initial state:

$$\min_{\hat{\boldsymbol{u}}_0, \ldots, \hat{\boldsymbol{u}}_{N-1}} \quad \sum_{k=0}^{N-1} J_{\text{cte},k} + J_{v,k} + J_{\text{acc},k} \quad (6a)$$

$$\text{s.t.} \quad \hat{\boldsymbol{s}}_0 = \boldsymbol{s}_0 \quad (6b)$$

$$||(\hat{s}_{x,k}, \hat{s}_{y,k})^T - (o_x, o_y)^T||_2 \geq r, \ \forall \boldsymbol{o} \in \mathcal{O}_l \quad (6c)$$

$$(\hat{s}_{x,k}, \hat{s}_{y,k}) \notin \mathcal{D}_k \quad (6d)$$

$$\hat{\boldsymbol{s}}_{k+1} = f(\hat{\boldsymbol{s}}_k, \hat{\boldsymbol{u}}_k) \quad (6e)$$

$$\hat{\boldsymbol{u}}_k \in [\boldsymbol{u}_{min}, \boldsymbol{u}_{max}] \quad (6f)$$

$$\Delta\hat{\boldsymbol{u}} \in [\dot{\boldsymbol{u}}_{min}\Delta t_s, \dot{\boldsymbol{u}}_{max}\Delta t_s] \quad (6g)$$

The constants $\boldsymbol{u}_{min}$, $\boldsymbol{u}_{max}$, $\dot{\boldsymbol{u}}_{min}$, and $\dot{\boldsymbol{u}}_{max}$ are the limitations of the input and the derivative of input. The final cost $V_f(s_N)$ is set to 0. The nonlinearity in constraints (6c) and (6d) can be directly handled by the nonlinear model predictive controller.

IV. Results

In this section, we present the synthetic dataset and the training details for the MDN. A modified Mahalanobis distance is introduced for evaluating the performance of MDNs, which is compared with KFs. We conclude this section by showing how motion prediction is used during trajectory planning for two different scenarios. The evaluation code for the motion prediction is available[1].

A. Dataset and training

For training MDNs, we implemented a simulator to generate a synthetic dataset, which is visualized in Fig. 4. We refer to this as the Factory Traffic Dataset (FTD). It facsimiles a factory scene within a 10×10 meters square area, including

[1]https://github.com/Woodenonez/multimodal_motion_prediction

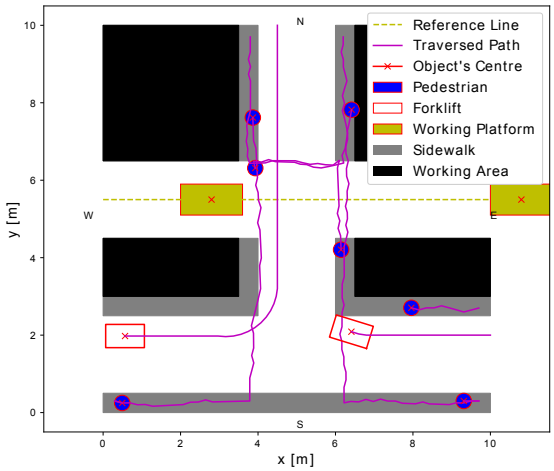


Fig. 4: Visualization of the synthetic factory traffic dataset (FTD).

three basic semantic zones: sidewalks, lanes, and working areas. Three types of entities, pedestrians, forklifts, and mobile working platforms, are simulated. The sizes of objects are based on real-world situations. There is one special lane in the middle representing the assembling line only for mobile working platforms, where platforms move slowly and periodically from the left side to the right. Pedestrians mainly walk within the pavements but may cross the lanes at certain locations. Working areas are not accessible for any objects, while pavements are not drivable for any non-human objects. In the end, the traffic rules are simplified as mobile platforms have the highest priority to pass and pedestrians will always wait for other types of objects. What is more, forklifts decide randomly on which one passes first as long as no collisions. Pedestrians or forklifts may appear at any entrances and aim at one of the other exits randomly.

We run the simulation for 2000 seconds with 0.2 second sampling time, and simulate 1200 trajectories for training. With $\Delta t_s = 0.2s$, $\kappa = 5$, $T_{max} = 20$, and $m = 10$, we have 0.9 million samples for training. In this work, we use the negative log-likelihood (NLL) loss for training [6], [10].

B. Motion prediction performance

We compare our MDN model to a CV based Kalman filter (KF). Fig. 5 shows the advantage of MDNs compared to KFs, which is the multimodality of the output of MDNs. The dynamic obstacle reveals a distinct tendency to turn, and the MDN captures the potential while keeping the possibility that the object keeps moving straight. By contrast, the KF counts on the motion model for predictions thus exhibits slow reactions to changes and only has a single prediction.

A prediction from MDNs is a probability distribution, but the ground truth is a pair of coordinates indicating the future position of the centre of the targeted object. The Mahalanobis distance (MD) [20] is a metric to measure the distance from

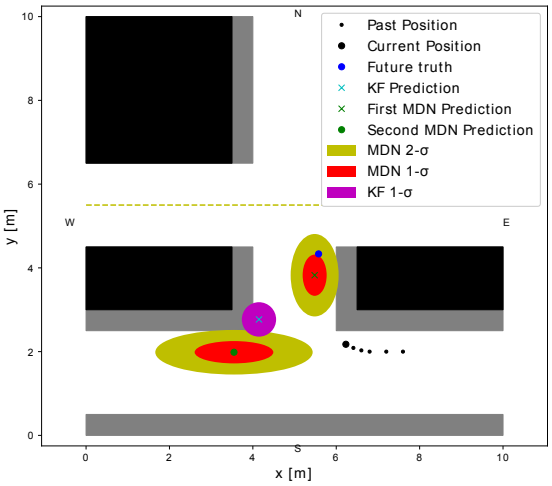


Fig. 5: *Comparison between the predictions from the KF and the MDN when the forklift is about to take a turn. Given the past positions and the current position, the MDN provides two predictions. The first prediction which has the largest weight indicates the potential turning behavior. This is the closest prediction to the future ground truth. The KF only provides one prediction according to the motion model, which deviates from the ground truth severely. The i-σ area is the ellipse area with $i \cdot \boldsymbol{\sigma}$ as its axes.*

a distribution to a point and is defined as:

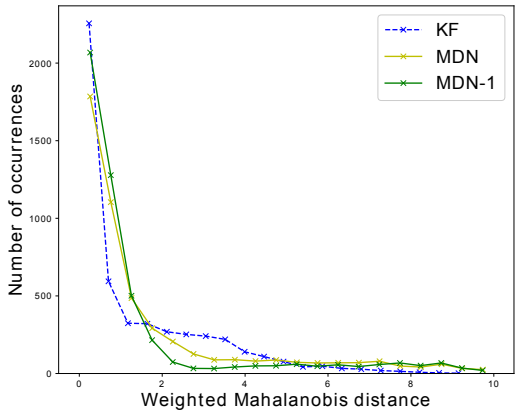
$$D_M(\boldsymbol{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sqrt{(\boldsymbol{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{x}-\boldsymbol{\mu})} \tag{7}$$

where $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ are the mean vector and the covariance matrix. $D_M(\boldsymbol{x})$ provides a measurement between a distribution and a point. To apply this concept for the multimodal case, we propose a weighted MD (WMD) metric, see (8), in which $\tilde{m}$ is the number of components after eliminating the degenerate ones and $\tilde{\alpha}_i$ is the corresponding renormalized weight. Components with weights lower than 10% of the maximal weight are regarded as degenerate.

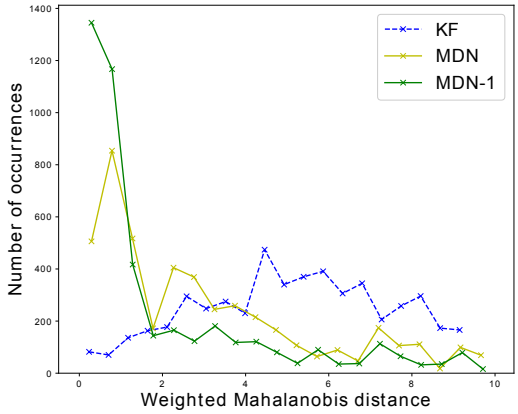
$$D_{WM}(\boldsymbol{x}) = \sum_{i}^{\tilde{m}} \tilde{\alpha}_i D_M(\boldsymbol{x}|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) \tag{8}$$

We evaluate the performance between MDNs and KFs based on the WMD metric using two datasets, *test 1* and *test 2*. Given $T_{max} = 20$, in *test 1*, we generate 5000 samples of pedestrians and forklifts for $T = 1, \ldots, T_{max}$. The second data-set, *test 2*, adopts 5000 forklift samples for $T = T_{max}$. The reason why no mobile platform is included is that they move at constant velocity thus not discriminative for KFs and MDNs. *Test 2* only contains forklift samples with further predictions into the future because these samples reflect the multimodal feature of motions.

The KF evolves according to historical positions and then predicts a future position using a CV model. In this work, we assume and set that the covariance matrix of the KF does not change during prediction steps to avoid the excessive increase. The result in Fig. 6 and table I



(a) *Test-set 1.*



(b) *Test-set 2.*

Fig. 6: *The histograms present WMDs on test data from the corresponding sets for the KF, MDN, and the first component of MDN. This is a truncation from 0 to 10. The first test set contains 5000 samples of pedestrians and forklifts with $T = 1, 2, \ldots, 20$, and the second one picks up 5000 samples of forklifts with $T = 20$.*

shows the average values of the WMD. MDN-1 only takes the most possible prediction with the largest weight from our MDN into account. From this result, the KF has a better score in the first set due to the fact that in most situations the objects move straight. But it already shows that the MDN outperforms the KF within the high WMD section, where the changes of movement directions happen. This is confirmed by implementing the second test with more turning behaviors. The result is reasonable since the KF cannot foresee the potential of the object changing the current motion pattern. The result shows that the most confident prediction of the MDN follows the ground truth well. However, this may not hold when motions of dynamic obstacles are more complicated. For example, if there are several motion modes with similar possibilities to happen, the most confident prediction of the MDN will have a high possibility to be incorrect, thus to consider several possible modes will be necessary during the trajectory planning.

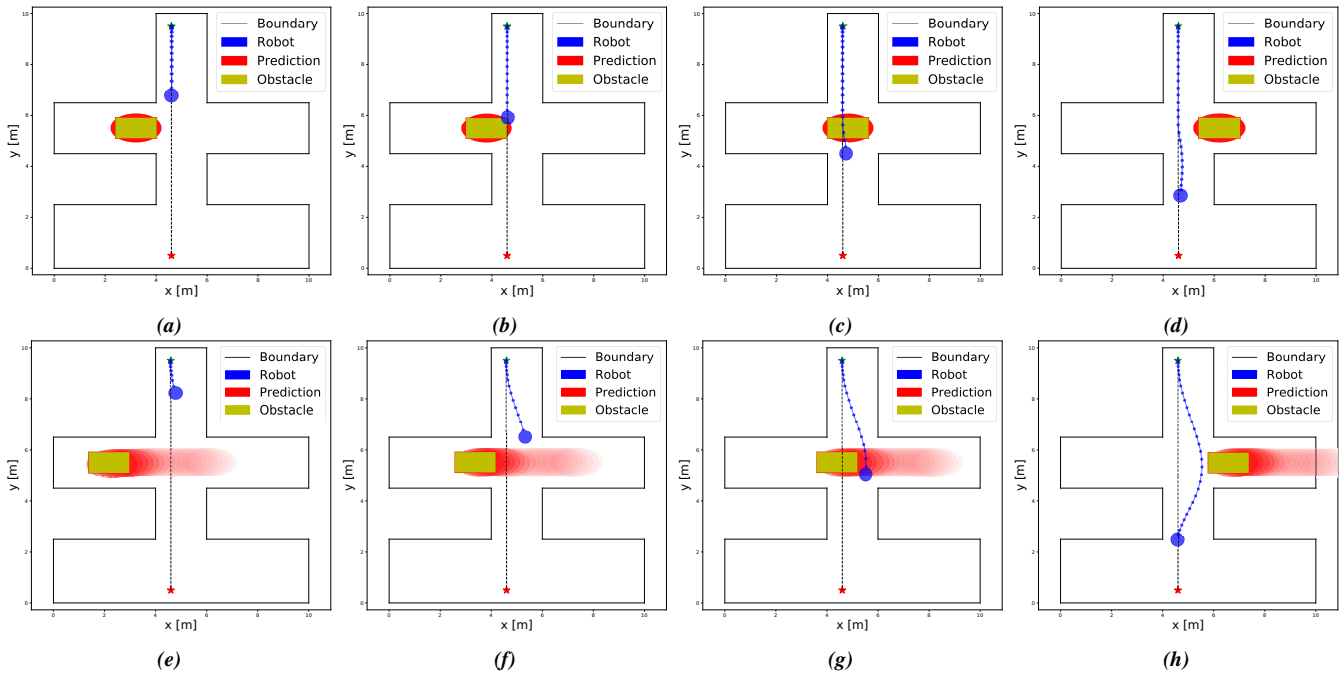


Fig. 7: *The upper half, Fig. 7a, 7b, 7c, and 7d, shows the path adjusted by the NMPC controller without any prediction, which leads to a collision with the mobile platform. The lower half, Fig. 7e, 7f, 7g, and 7h, shows the path adjusted by the NMPC controller with predictions from the MDN, which avoids the collision successfully.*

C. Trajectory planning with multimodal motion predictions

We now address how motion prediction affects trajectory planning. Considering the prediction and control horizon N, we insert the prediction from the MDN to the corresponding horizon step so that the controller plans the trajectory in advance to detour around the dynamic obstacles. We present two planning scenarios to demonstrate the joint outcome of NMPCs and MDNs. In the first scenario, shown in Fig. 7, a mobile robot tries to avoid colliding with an uncontrolled mobile platform going straight with a constant velocity. We compare the result with and without predictions from the MDN on the motion of the platform. In the second scenario, shown in Fig. 8, a mobile robot tries to predict if a pedestrian walking on the pavement will make a turn at a crossing or continue walking forward. In this scenario, we compare the result with predictions of future positions about the pedestrians from the MDN and the KF.

In Fig. 7, 8, the red ellipses are the predictions. The more transparent the ellipse is, the longer in the future the prediction is made for. In both scenarios, the penalty for the cross-track-error q_{cte} is 50, and for the velocity deviation q_v is 20. A video of the controlled behavior is available[2].

TABLE I: *Comparison among the KF, MDN, MDN-1 on the FTD using the WMD metric. Here are the average values of WMD from 5000 samples. The best performance are shown in bold.*

	WMD	
Approach	Test 1	Test 2
Kalman filter	**1.38**	5.11
MDN-1	1.85	**3.62**
MDN	1.94	3.85

1) Plan without and with predictions of dynamic obstacles: Fig. 7 demonstrates the necessity to include predictions of dynamic obstacles for the mobile robot. The working platform moves from the left to the right side of the map. Without any prediction, as shown in Fig. 7a, 7b, 7c, 7d, the robot fails to avoid the collision due to too late actions. In Fig. 7e, 7f, 7g, 7h, the robot avoids the collision with the help of predictions from the MDN. In this scenario, the KF and MDNs achieve the same result, since the mobile platforms follows the CV model.

2) Plan with single-modal and multimodal predictions of dynamic obstacles: Fig. 8 shows the significance of considering multimodality. A pedestrian walking on the sidewalk may take a turn to cross the lane or continue straight forward. In Fig. 8a, 8b, 8c, 8d, the KF cannot foresee that the pedestrian turns, resulting in a collision since the mobile robot does not take the turn into account during trajectory planning. The covariance matrix of the observation noise is set to be an identity matrix, giving a prediction with a larger variance compared with the one from MDN. However, the MDN learned this motion pattern from past trajectories and predicts two modes of the motion of the pedestrian: going straight and turning left, as shown in Fig. 8e, 8f, 8g, 8h. Then the NMPC controller corrects the trajectory according to the turning-left mode of the pedestrian and avoids the collision successfully.

V. Conclusions

We have shown how multimodal motion prediction, based on mixture density networks that learn behaviors of obsta-

[2]https://youtu.be/ficYIO8y04k

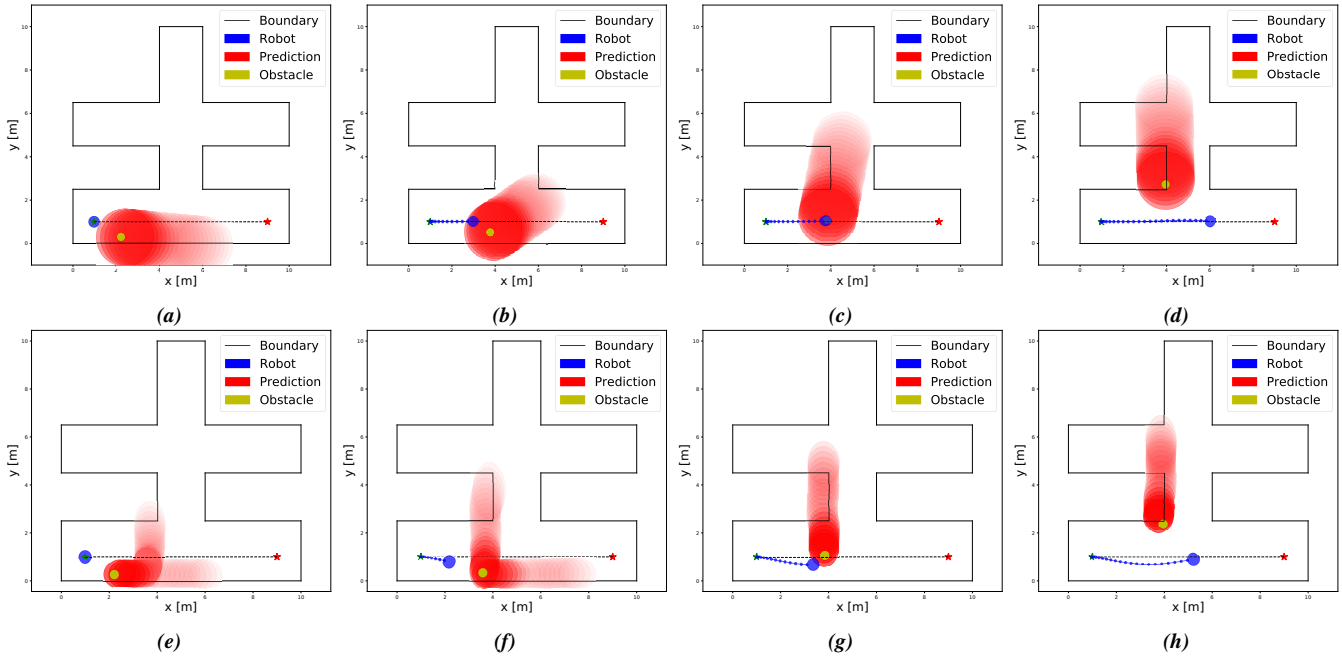


Fig. 8: *The upper half, Fig. 8a, 8b, 8c, and 8d, shows the path adjusted by the NMPC controller with predictions from the KF, which cannot avoid the collision with the pedestrian. The lower half, Fig. 8e, 8f, 8g, and 8h, shows the path adjusted by the NMPC controller with predictions from the MDN, which considers the possibility that the pedestrian may turn and avoid the collision accordingly.*

cles, can be utilized with a model predictive control strategy for trajectory generation to avoid collisions with moving obstacles. This method improves the ability of dynamic obstacle avoidance for mobile robots by the chance of success. The experiments, especially the multimodal prediction scenario, demonstrate the feasibility of our method and the fact that deep-learning-based motion prediction is superior to traditional single-modal predictions such as the Kalman filter by higher prediction accuracy when the test scenario contains more possibilities of the motion of objects.

REFERENCES

[1] M. Scholz, X. Zhang, and J. Franke, "Implementation of an intralogistics routing-service basing on decentralized workspace digitization. applied mechanics and materials," *Applied Mechanics and Materials*, vol. 882, p. 90–95, 2018.

[2] X. Zhang, M. Scholz, S. Reitelshöfer, and J. Franke, "An autonomous robotic system for intralogistics assisted by distributed smart camera network for navigation," in *IEEE Conference on Automation Science and Engineering (CASE)*, 2018, pp. 1224–1229.

[3] E. Masehian and M. R. Amin-Naseri, "A voronoi diagram–visibility graph–potential field compound algorithm for robot path planning," *Journal of Robotic Systems*, vol. 21(6), p. 275–300, 2004.

[4] F. Duchoň, A. Babinec, M. Kajan, P. Beňo, M. Florek, T. Fico, and L. Jurišica, "Path planning with modified a star algorithm for a mobile robot," *Procedia Engineering*, vol. 96, pp. 59–69, 2014.

[5] A. Rudenko, L. Palmieri, M. Herman, K. M. Kitani, D. M. Gavrila, and K. O. Arras, "Human motion trajectory prediction: a survey," *International Journal of Robotics Research*, vol. 39, no. 8, pp. 895–935, 2020.

[6] O. Makansi, E. Ilg, O. Cicek, and T. Brox, "Overcoming limitations of mixture density networks: A sampling and fitting framework for multimodal future prediction," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[7] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese, "Social LSTM: Human trajectory prediction in crowded spaces," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[8] N. Nikhil and B. Tran Morris, "Convolutional neural network for trajectory prediction," in *European Conference on Computer Vision (ECCV)*, September 2018.

[9] C. Rupprecht, I. Laina, R. DiPietro, M. Baust, F. Tombari, N. Navab, and G. D. Hager, "Learning in an uncertain world: Representing ambiguity through multiple hypotheses," in *IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 3611–3620.

[10] C. M. Bishop, "Mixture density networks," 1994, technical report.

[11] D. Connell and H. M. La, "Dynamic path planning and replanning for mobile robots using RRT," in *IEEE Conference on Systems, Man, and Cybernetics (SMC)*, 2017, pp. 1429–1434.

[12] M. Missura and M. Bennewitz, "Predictive collision avoidance for the dynamic window approach," in *International Conference on Robotics and Automation (ICRA)*, 2019, pp. 8620–8626.

[13] C. Cao, P. Trautman, and S. Iba, "Dynamic channel: A planning framework for crowd navigation," in *International Conference on Robotics and Automation (ICRA)*, 2019, pp. 5551–5557.

[14] R. Soloperto, J. Köhler, F. Allgöwer, and M. A. Müller, "Collision avoidance for uncertain nonlinear systems with moving obstacles using robust model predictive control," in *European Control Conference (ECC)*, 2019, pp. 811–817.

[15] J. B. Rawlings, D. Q. Mayne, and M. M. Diehl, *Model Predictive Control: Theory, Computation, and Design (2nd Edition)*. Nob Hill Publishing, LLC, 2020.

[16] L. Stella, A. Themelis, P. Sopasakis, and P. Patrinos, "A simple and efficient algorithm for nonlinear model predictive control," in *IEEE 56th Annual Conference on Decision and Control (CDC)*. IEEE, 2017, pp. 1939–1944.

[17] A. Sathya, P. Sopasakis, R. Van Parys, A. Themelis, G. Pipeleers, and P. Patrinos, "Embedded nonlinear model predictive control for obstacle avoidance using PANOC," in *European control conference (ECC)*. IEEE, 2018, pp. 1523–1528.

[18] L. U. Hjorth and I. T. Nabney, "Regularisation of mixture density networks," in *Ninth International Conference on Artificial Neural Networks ICANN 99. (Conf. Publ. No. 470)*, 1999, pp. 521–526 vol.2.

[19] J. Berlin, G. Hess, A. Karlsson, W. Ljungbergh, Z. Zhang, P.-L. Götvall, and K. Åkesson, "Trajectory generation for mobile robots in a dynamic environment using nonlinear model predictive control," 2021, doi:10.36227/techrxiv.14215862.

[20] R. De Maesschalck, D. Jouan-Rimbaud, and D. Massart, "The Mahalanobis distance," *Chemometrics and Intelligent Laboratory Systems*, vol. 50, no. 1, pp. 1–18, 2000.