



<http://www.diva-portal.org>

## Preprint

This is the submitted version of a paper presented at *2015 IEEE conference on Fuzzy Systems, (FUZZ-IEEE) Istanbul, Turkey, August 2-5, 2015*.

Citation for the original published paper:

Palm, R., Driankov, D. (2015)

Velocity potentials and fuzzy modeling of fluid streamlines for obstacle avoidance of mobile robots.

In: *2015 IEEE conference on Fuzzy Systems (FUZZ-IEEE)* (pp. 1-8). IEEE Press

<http://dx.doi.org/10.1109/FUZZ-IEEE.2015.7337800>

N.B. When citing this work, cite the original published paper.

Permanent link to this version:

<http://urn.kb.se/resolve?urn=urn:nbn:se:oru:diva-47940>

# Velocity potentials and fuzzy modeling of fluid streamlines for obstacle avoidance of mobile robots

Rainer Palm, Senior Member IEEE, and Dimiter Driankov Member IEEE

**Abstract**—The use of the velocity potential of an incompressible fluid is an important and elegant tool for obstacle avoidance of mobile robots. Obstacles are modeled as cylindrical objects - combinations of cylinders can also form super obstacles. Possible trajectories of a vehicle are given by a set of streamlines around the obstacle computed by the velocity potential. Because of the number of streamlines and of data points involved therein, models of sets of streamlines for different sizes of obstacles are created first using dataset models and finally fuzzy models of streamlines. Once an obstacle appears in the sensor cone of the robot the set of streamlines is computed from which that streamline is selected that guarantees a smooth transition from/to the planned trajectory. Collisions with other robots are avoided by a combination of velocity potential and force potential and/or the change of streamlines during operation (lane hopping).

**Keywords**—obstacle avoidance, velocity potential, fuzzy modeling, streamlines

## I. INTRODUCTION

A fast reaction of mobile robots to a sudden appearance of obstacles requires a safe avoidance strategy not only for off-line planned paths but also for autonomous motions. One method is to use the artificial force potential field approach introduced by Khatib in 1985 [1] where artificial attractive and repulsive forces are introduced to force the robot around an obstacle while heading toward a final target at the same time. Optimization techniques like market-based optimization (MBO) and particle swarm optimization (PSO) influencing artificial potential fields have been presented by Palm and Bouguerra [2], [3]. Borenstein [4] introduced the vector field histogram technique, and Michels [5] applied the reinforcement learning method. Another successful method to cope with obstacle avoidance is the *fuzzy logic approach* which has been widely used for mobile robots since the early nineties. Martinez et al described a system of heuristic rules based on interaction of mobile robots and traffic rules [6]. Another rule-based approaches are presented in [7]. A fuzzy obstacle controller using so-called negative-fuzzy rules is reported by Lilly [8]. Stingu and Lewis combined a motion control fuzzy rule base using an occupancy map of the environment similar to an artificial potential field within which the robots interact [9]. A further fuzzy rule-based approach to obstacle avoidance can be found in [10]. A closer look at the problem of path planning and obstacle avoidance leads to a similar case when fluids circumvent obstacles in a smooth and energy saving way. The result is a bundle of trajectories from which one can conclude how an autonomous robot should behave under non-holonomic constraints. This leads to another kind of artificial

potential for obstacle avoidance that was introduced by Khosla [11] who used the *velocity potential* of fluid mechanics to construct stream lines in a working area of a mobile robot moving around obstacles in a very natural way. The velocity potential approach is a method which considers both the path/trajectory planning in the case of a well known scenario including static obstacles and the on-line reaction to unplanned situations like obstacle avoidance in an unknown terrain. Kim and Khosla continued this work with the use of the velocity potential function to avoid obstacles in real time [12]. Further similar research has been published by Li et al [13], Ge et al [14], Waydo and Murray [15], Daily and Bevly [16], Sugiyama [17], and Gingras et al [18]. Most of these approaches use a *point source/point sink* combination for flow construction. This can be of disadvantage in the presence of a combination of tracking velocity vectors and obstacle avoidance vectors.

Therefore in [19] and also in this paper the velocity potential is preferred that uses a uniform flow plus a doublet representing a cylindrical obstacle. In order to handle several separated obstacles the so-called *flow velocity average* is used whereas the *flow potential average* acts to merge several obstacles to one *super obstacle*. The result are streamlines going around the obstacle(s) along which the robot is intended to move.

The contribution of this paper and the motivation to use fuzzy modeling is that because of the high number of streamlines and of data points involved therein, models of sets of streamlines for different sizes of obstacles are created first by dataset models and then by fuzzy models of streamlines. These models are used as "patches" within a predefined trajectory. In the case of an appearance of one or more obstacles in the sensor cone of the robot an appropriate streamline model is selected, from which that streamline is selected, that guarantees a smooth transition from the planned trajectory to the streamline. After having left behind the obstacle, the robot moves back to the original trajectory. In the case of possible collisions between robots that move on crossing streamlines, a change between streamlines during operation (lane hopping) is presented.

The paper is organized as follows. Section II presents the modeling and control of the mobile robot. Section III gives a brief introduction to fluid mechanics. In section IV the generation of *dataset models* and *fuzzy models* is addressed. Section VI deals with the change of streamlines (lane hopping) in cases of conflicts with constraints and static and dynamic obstacles. In Sect. VII simulation results are presented and Sect. VIII ends with conclusion and future work.

Rainer Palm is adjunct professor at the AASS, Dept. of Technology Örebro University SE-70182 Örebro, Sweden, Email: rub.palm@t-online.de

Dimiter Driankov is professor at the AASS, Dept. of Technology Örebro University SE-70182 Örebro, Sweden, Email: dimiter.driankov@oru.se

## II. MODELING AND CONTROL OF THE VEHICLE

### A. Kinematic modeling

In the following a nonholonomic rear-wheel driven vehicle is considered with the kinematics of a car

$$\begin{aligned}\dot{q}_i &= R_i(q_i) \cdot u_i \\ q_i &= (x_i, y_i, \Theta_i, \phi_i)^T \\ R_i(q_i) &= \begin{pmatrix} \cos \Theta_i & 0 \\ \sin \Theta_i & 0 \\ \frac{1}{l_i} \cdot \tan \phi_i & 0 \\ 0 & 1 \end{pmatrix}\end{aligned}\quad (1)$$

where

$q_i \in \mathbb{R}^4$  - state vector

$u_i = (u_{1i}, u_{2i})^T \in \mathbb{R}^2$  - control vector, pushing/steering speed

$x_i = (x_i, y_i)^T \in \mathbb{R}^2$  - position vector of platform  $P_i$

$\Theta_i$  - orientation angle

$\phi_i$  - steering angle

$l_i$  - length of vehicle

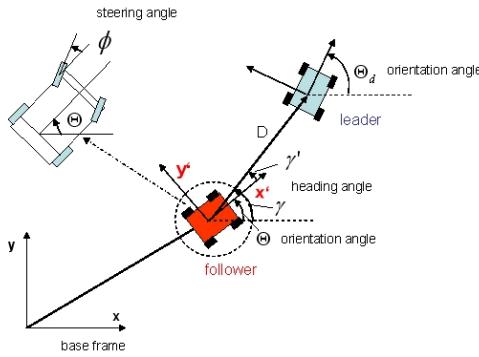


Fig. 1. Leader follower principle

### B. Virtual leader

The trajectory generation is done by a 'virtual' vehicle (the leader) that moves in front of the 'real' vehicle (the follower) (see [20]). The virtual leader acts as trajectory generator for the real platform at every time step, based on starting and end position (target), obstacles to be avoided, other platforms to be taken into account etc (see Fig. 1). The dynamics of the virtual platform is designed as a first order system that automatically avoids abrupt changes in position and orientation

$$\dot{\mathbf{v}}_{vi} = \mathbf{k}_{vi}(\mathbf{v}_{vi} - \mathbf{v}_{di}) \quad (2)$$

$\dot{\mathbf{v}}_{vi} \in \mathbb{R}^2$  - velocity of virtual platform  $P_i$

$\mathbf{v}_{di} \in \mathbb{R}^2$  - desired velocity of virtual platform  $P_i$

$\mathbf{k}_{vi} \in \mathbb{R}^{2 \times 2}$  - damping matrix (diagonal)

$\mathbf{v}_{di}$  is composed of the tracking velocity  $\mathbf{v}_{ti}$  and velocity terms due to artificial potential fields from obstacles and other platforms. The tracking velocity is designed as a control term

$$\mathbf{v}_{ti} = \mathbf{k}_{ti}(\mathbf{x}_i - \mathbf{x}_{ti}) \quad (3)$$

$\mathbf{x}_{ti} = (x_{ti}, y_{ti})$  - position vector of target  $T_i$   
 $\mathbf{k}_{ti} \in \mathbb{R}^{2 \times 2}$  - gain matrix (diagonal)

Among many ways of computing the control vector  $u_i$  for the follower in (1) a local linear gain scheduler is applied under the assumption of a slowly time varying 'leader-follower' system (see [3]).

## III. SOME PRINCIPLES OF FLUID MECHANICS

In the theory of fluids the terms *velocity potential*, *stream function* and *complex potential* are introduced [21]. The so-called *uniform parallel flow* is introduced that corresponds to an undisturbed trajectory along straight lines. The *flow of a doublet* corresponds to a flow around a cylinder. *Superposition of uniform flow and doublet* leads to a model of a uniform flow around a cylindrical obstacle.

### A. Velocity potential, stream function, and complex potential

Consider a flow of an incompressible fluid in the  $x, y$  plane with the velocities  $v_x$  and  $v_y$  in  $x$  and  $y$  directions, respectively. Define a function  $\Phi(x, y)$  - the so-called velocity potential with the property

$$\begin{aligned}v_x &= \frac{\partial \Phi}{\partial x} & v_y &= \frac{\partial \Phi}{\partial y} \\ v_r &= \frac{\partial \Phi}{\partial r} & v_\Theta &= \frac{\partial \Phi}{r \partial \Theta}\end{aligned}\quad (4)$$

Define further a function  $\Psi(x, y)$  - the so-called stream function with

$$\begin{aligned}v_x &= \frac{\partial \Psi}{\partial y} & v_y &= -\frac{\partial \Psi}{\partial x} \\ v_r &= \frac{\partial \Psi}{r \partial \Theta} & v_\Theta &= -\frac{\partial \Psi}{\partial r}\end{aligned}\quad (5)$$

Consider finally a regular function  $w(z)$  of the complex variable  $z = x + iy$

$$w(z) = \Phi + i\Psi \quad (6)$$

$w(z)$  - complex potential,  $\Phi(x, y)$  - velocity potential,  $\Psi(x, y)$  - stream function. Differentiation of (6) yields

$$dw = (v_x - iv_y)(dx + idy) = (v_x - iv_y)dz$$

From this we get the conjugate complex velocity

$$\frac{\partial w}{\partial z} = v_x - iv_y \quad (7)$$

The result is: from the complex potential containing the velocity potential and the stream function one obtains through a differentiation by  $z$  a complex velocity containing the velocities in  $x$  and  $y$  direction which we need for the calculation of trajectories around a cylindrical obstacle.

### B. Superposition of uniform flow and doublet

A uniform parallel flow is characterized by stream lines forming parallel straight lines  $y = \text{const}$ . Furthermore, a doublet flow is a combination of the flows of a source and a sink. The flow around a cylindrical object - an obstacle - is finally computed by a superposition of the uniform flow and the doublet which is a superposition of their complex potentials (see Fig. 2).

$$w(z) = U \cdot z + U \frac{r_0^2}{z - z_0} \quad (8)$$

where  $U$  is the flow,  $r_0$  is the radius of the obstacle  $z = r(\cos\Theta + i\sin\Theta)$  is the complex variable.  $z_0$  is the position of the obstacle in the complex plane,  $\Theta$  is the angle between  $z$  and the imaginary axis (see Fig. 2). The velocity components in polar coordinates are obtained as

$$\begin{aligned} v_r &= U(C_{fact} \cdot \cos\Theta - \frac{2z_{re}r_0^2(z_{re}\cos\Theta + z_{im}\sin\Theta)}{(z_{re}^2 + z_{im}^2)^2}) \\ v_\Theta &= -U(C_{fact} \cdot \sin\Theta + \frac{2z_{re}r_0^2(-z_{re}\sin\Theta + z_{im}\cos\Theta)}{(z_{re}^2 + z_{im}^2)^2}) \end{aligned} \quad (9)$$

where  $z_{re} = r\cos\Theta - x_0$ ,  $z_{im} = r\sin\Theta - y_0$ , and  $C_{fact} = (1 + \frac{r_0^2}{z_{re}^2 + z_{im}^2})$ .

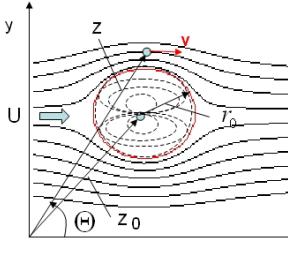


Fig. 2. Flow around a cylinder

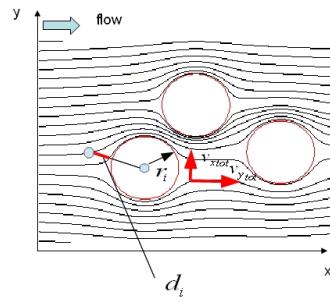


Fig. 3. Flow around 3 cylinders

### C. Superposition of two or more cylinders

For more than one cylinder we obtain from (9), and (11) velocity components  $v_{ri}$  and  $v_{\Theta i}$  in polar coordinates that will be transformed into cartesian coordinates by

$$(v_{xi}, v_{yi})^T = \begin{pmatrix} \cos(\Theta) & -\sin(\Theta) \\ \sin(\Theta) & \cos(\Theta) \end{pmatrix} \cdot (v_{ri}, v_{\Theta i})^T \quad (10)$$

Then, weighting functions  $\mu_i$  for the velocities  $(v_{xi}, v_{yi})$  are introduced depending on the distance of the actual robot position  $d_i$  to the cylinder surfaces [15], [16]

$$\mu_i = \prod_{i \neq j} \frac{d_j}{d_i + d_j} \quad (11)$$

The weighted sum of the velocities  $v_{xi}$  (flow velocity averages) in x-direction and  $v_{yi}$  in y-direction in cartesian coordinates

$$v_{x_{tot}} = \sum_i w_i \cdot v_{xi} \quad v_{y_{tot}} = \sum_i w_i \cdot v_{yi} \quad (12)$$

leads to the streamlines for the multiple obstacle case (Fig.3) where  $w_i = \mu_i / \sum_k \mu_k$  can be interpreted as a fuzzy measure with  $k = 1 \dots N$ ,  $N$  - number of obstacles.

### D. Forming of super obstacles

Cylindrical obstacle models are useful for quite a number of real situations during navigation between obstacles. However cylindrical models might be inappropriate for some kinds of obstacles like walls and objects the length of which is much larger than there width. Another case exists when the vehicle should circumvent a number of obstacles instead of navigating between them. A solution to this problem in the current framework are so-called *super obstacles* [16] which can be formed by the *flow potential averages* in contrast to *flow velocity averages*. *Flow velocity average* is computed by

- a. the velocity potential  $\Phi_i$  of each obstacle and the regarding velocities separately

- b. the total velocity according to (12)

*Flow potential average* is computed by

- a. the stream function  $\Psi_i$  of each obstacle separately.

- b. the total stream function  $\Psi_{tot}$ .

$$\Psi_{tot} = \sum_i w_i \cdot \Psi_i \quad (13)$$

- c. Compute the total velocities ( $v_{r_{tot}}$ ,  $v_{\Theta_{tot}}$ )

$$v_{r_{tot}} = \frac{\partial \Psi_{tot}}{r \partial \Theta} = \sum_i \frac{1}{r} \left[ \frac{\partial w_i}{\partial \Theta} \Psi_i + w_i \frac{\partial \Psi_i}{\partial \Theta} \right] \quad (14)$$

$$v_{\Theta_{tot}} = -\frac{\partial \Psi_{tot}}{\partial r} = -\sum_i \left[ \frac{\partial w_i}{\partial r} \Psi_i + w_i \frac{\partial \Psi_i}{\partial r} \right] \quad (15)$$

and the cartesian velocities ( $v_{x_{tot}}$ ,  $v_{y_{tot}}$ ) according to (10).

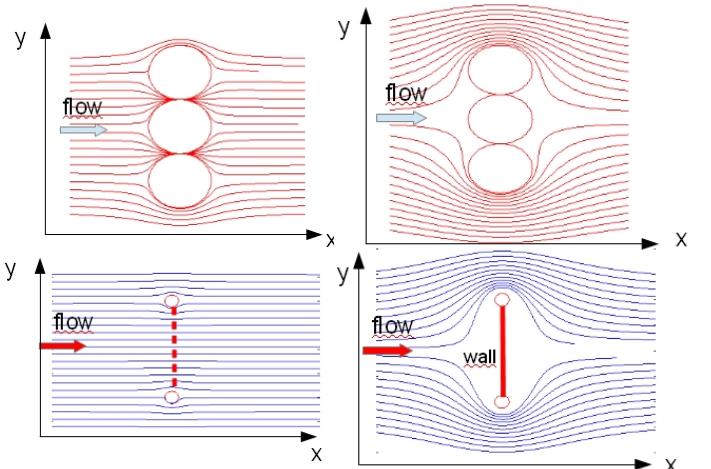


Fig. 4. flow velocity average

Fig. 5. flow potential average

Figures 4 and 5 show two examples of *flow velocity average* and *flow potential average*. By using *flow potential average* the modeling of walls only by placing small cylindrical objects becomes possible.

#### IV. MODELING OF STREAMLINES

In order to make the online computation during obstacle avoidance more efficient, the modeling of a possible obstacle together with the regarding streamlines is done in advance. The use of this model takes place just at the moment of the appearance of the obstacle. The modeling can be described as follows:

1. Define a cylindrical model of an obstacle of well defined size (radius) where the size of the obstacle model be of approximately the same size of the real one.
  2. Define an area in model coordinates  $(\tilde{x}, \tilde{y})$  within which the obstacle model is positioned at  $(\tilde{x}_0, \tilde{y}_0)$  (special case:  $\tilde{y}_0 = 0$ ) and a defined number K of streamlines around the obstacle that needs to be computed. The lengths of the streamlines are defined by time constraints  $t \in (t_{0mod}, t_{endmod})$ .
  3. Generate a separate model for each streamline.
- In the following two kinds of models will be described:
- a) Modeling by a "dataset model" using all available data points
  - b) Modeling by fuzzy time clustering

##### A. Modeling by a dataset model

Let an obstacle appear at time  $t_{0rob}$  in the sensor cone of the robot at the position  $(x_0, y_0)$  in robot coordinates. The use of the obstacle model requires a transformation of the obstacle and streamlines from model coordinates  $(\tilde{x}, \tilde{y})$  into robot coordinates  $(x, y)$  at time  $t_{0rob}$ . Then the transformation of a point of the n-th streamline  $(\tilde{x}_i^n, \tilde{y}_i^n)$  in model coordinates into robot coordinates  $(x_i^n, y_i^n)$  yields (see Fig. 6)

$$x_i^n = \tilde{x}_i^n - (\tilde{x}_0 - x_0) \quad (16)$$

$$y_i^n = \tilde{y}_i^n - (\tilde{y}_0 - y_0) \quad (17)$$

$$n = 1 \dots K$$

The number  $n_{st}$  of the starting streamline of one of the K streamlines is found by  $n_{st} = \text{argmin}(\tilde{y}_1^n - y_0)$ . The number of time steps  $i_{st}$  on the starting streamline is computed by  $i_{st}^{n_{st}} = \text{argmin}(\tilde{x}_i^{n_{st}} - (\tilde{x}_0 - x_0))$ . The robot is supposed to start on streamline  $n_{st}$  at  $t_{0rob} = i_{st}$  and to move along this streamline until the obstacle disappears from the sensor cone at  $t_{endrob}$ . After that the robot moves along the original trajectory.

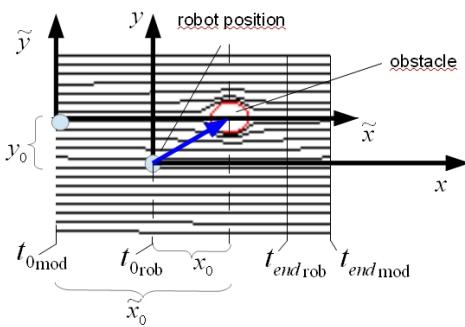


Fig. 6. Modeling of streamlines using all data points in robot frame

The multiple obstacle case will be discussed in the following where the index n is left out for the sake of simplicity.

For a two-obstacle example see Fig. 7:

In this case we compute the streamlines for N obstacles by

$$x_{1i_1} = \tilde{x}_{1i_1} - (\tilde{x}_0 - x_{10}); y_{1i_1} = \tilde{y}_{1i_1} - (\tilde{y}_0 - y_{10}) \quad (18)$$

$$x_{2i_2} = \tilde{x}_{2i_2} - (\tilde{x}_0 - x_{20}); y_{2i_2} = \tilde{y}_{2i_2} - (\tilde{y}_0 - y_{20})$$

...

$$x_{Ni_N} = \tilde{x}_{Ni_N} - (\tilde{x}_0 - x_{N0}); y_{Ni_N} = \tilde{y}_{Ni_N} - (\tilde{y}_0 - y_{N0})$$

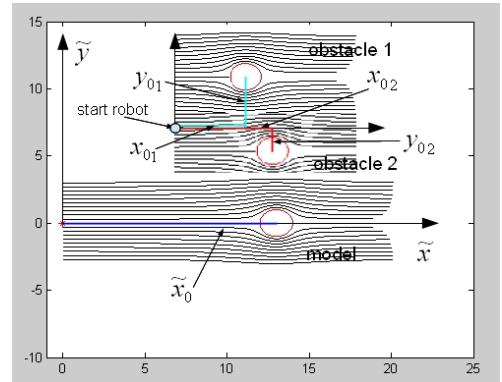


Fig. 7. Modeling of two obstacles

Since a combination of different obstacles is only possible via a summation of weighted velocities we approximate the velocities by

$$\Delta x_{1i_1} = (x_{1i_1} - x_{1i_1-1}); \Delta y_{1i_1} = (y_{1i_1} - y_{1i_1-1}) \quad (19)$$

$$\Delta x_{2i_2} = (x_{2i_2} - x_{2i_2-1}); \Delta y_{2i_2} = (y_{2i_2} - y_{2i_2-1})$$

...

$$\Delta x_{Ni_N} = (x_{Ni_N} - x_{Ni_N-1}); \Delta y_{Ni_N} = (y_{Ni_N} - y_{Ni_N-1})$$

where  $i_1 \in [1, i_{1max}], i_2 \in [1, i_{2max}] \dots i_N \in [1, i_{Nmax}]$

The streamlines are computed by

$$x_{rob}(i_{tot}) = x_{rob}(i_{tot}-1) + \sum_k w_{ki_{tot}} \cdot \Delta x_k(i_{tot}) / \Delta t$$

$$y_{rob}(i_{tot}) = y_{rob}(i_{tot}-1) + \sum_k w_{ki_{tot}} \cdot \Delta y_k(i_{tot}) / \Delta t \quad (20)$$

$$k = 1 \dots N, \quad w_{ki_{tot}} \in [0, 1], \quad i_k \in [1, i_{kmax}] \\ i_{tot} \in [1, \min(i_{kmax})], \quad \Delta t = t_i - t_{i-1}$$

where  $\mathbf{x}_{rob}(1) = (x_{rob}(1), y_{rob}(1))^T$  is the starting point of the robot.

##### B. Modeling by fuzzy time clustering

*1) The clustering principle:* The basis for modeling are the datasets of the streamlines from which models for each individual streamline are developed using fuzzy clustering and Takagi-Sugeno fuzzy modeling ([22], [23]). Here the time instants are considered as model inputs and the streamline

coordinates as model outputs. Let the streamline coordinates be described by a smooth function

$$\mathbf{x}_{rob}(t) = \mathbf{f}_{rob}(t) \quad (21)$$

where  $\mathbf{x}_{rob}(t) = (x_{rob}, y_{rob})^T$ ,  $\mathbf{f}_{rob} \in R^2$ , and  $t \in R^+$ . Linearization of (21) at selected time points  $t_i$  yields

$$\mathbf{x}_{rob}(t) = \mathbf{x}_{rob}(t_i) + \frac{\Delta \mathbf{f}_{rob}(t)}{\Delta t}|_{t_i} \cdot (t - t_i) \quad (22)$$

By means of the local linear models (22) the nonlinear function (21) can be expressed by a Takagi-Sugeno fuzzy model

$$\mathbf{x}_{rob}(t) = \sum_{i=1}^c w_i(t) \cdot (\mathbf{A}_i \cdot t + \mathbf{d}_i) \quad (23)$$

where  $\mathbf{A}_i = \frac{\Delta \mathbf{f}(t)}{\Delta t}|_{t_i} \in R^2$  and  $\mathbf{d}_i = \mathbf{x}_{rob}(t_i) - \frac{\Delta \mathbf{f}(t)}{\Delta t}|_{t_i} \cdot t_i \in R^2$ .

$w_i(t) \in [0, 1]$  is the degree of membership of a time point  $t$  to a cluster with the cluster center  $t_i$ ,  $c$  is the number of clusters, and  $\sum_{i=1}^c w_i(t) = 1$ .

The general clustering and modeling steps are

- Pick an appropriate number of local linear models (data clusters)  $c$
- Find  $c$  cluster centers  $(t_i, x_{rob_i}, y_{rob_i})$ ,  $i = 1 \dots c$ , in the product space of the data triples  $(t, x_{rob}, y_{rob})$  by Fuzzy-c-elliptotype clustering
- Find the corresponding fuzzy regions in the space of input data ( $t$ ) by projection of the clusters of the product space into Gustafson-Kessel clusters (GK) within the input space [24]
- Calculate  $c$  local linear (affine) models (23) using the GK clusters from step 2.

The membership degree  $w_i(t)$  of an input data point  $t$  in an input cluster  $C_i$  is then calculated by

$$w_i(t) = \frac{1}{\sum_{j=1}^c \left( \frac{(t-t_i)^T M_{i,proj}(t-t_i)}{(t-t_j)^T M_{j,proj}(t-t_j)} \right)^{\frac{1}{m_{proj}-1}}} \quad (24)$$

The projected cluster centers  $t_i$  and the induced matrices  $M_{i,proj}$  define the input clusters  $C_i$  ( $i = 1 \dots c$ ). The parameter  $m_{proj} > 1$  represents the fuzziness of an individual cluster.

2) *Fuzzy modeling of streamlines*: The difference between a fuzzy model and the dataset model is that the data triple  $(\mathbf{x}(i), i)$  is computed by a fuzzy approximation instead of being selected from a "look-up table". However, the first step is to generate a "dataset model" by using all data points (see last subsection). On the basis of this dataset model the fuzzy time clustering model for each streamline is generated. Figure 8 shows the modeling results for 3 obstacles and 10 clusters per streamline. The detail in Fig. 9 shows that there are quite large deviations between the data (black) and the approximation (red). In contrast, an extension to 20 clusters shows a much better result (see Figs. 10 and 11).

The advantage of a fuzzy approximation is that it requires a much smaller memory than that of the dataset model, especially for many streamlines. This is because, for the fuzzy model only the cluster centers - e.g. 20 data triples - are stored, in contrast to e.g. 1500 data pairs. However, a

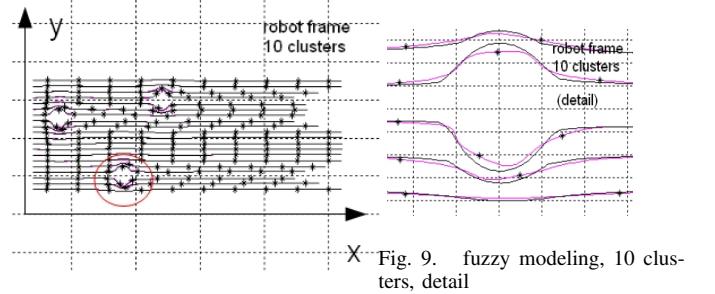


Fig. 8. fuzzy modeling, 10 clusters

Fig. 9. fuzzy modeling, 10 clusters, detail

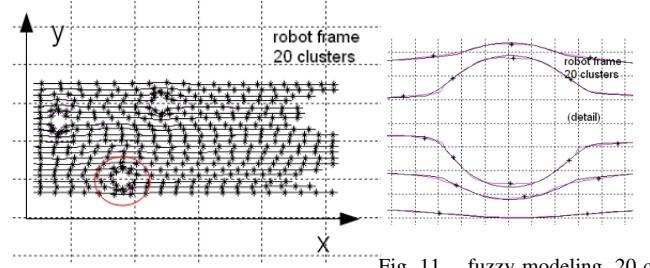


Fig. 10. fuzzy modeling, 20 clusters

Fig. 11. fuzzy modeling, 20 clusters, detail

fuzzy approximation requires the online calculation of a robot position instead of just reading out the next desired position from the dataset model. This disadvantage can be avoided by a combination of both methods: Once the streamline to be activated is selected, then the regarding dataset is computed for using during motion.

## V. STREAMLINE CALCULATIONS IN THE ROBOT AND BASE FRAME

The previous calculations are performed in a coordinate frame corresponding to the local robot frame. In the multi-robot case this concerns every involved robot so that a total view of the whole scenario can only be obtained from the viewpoint of the base frame.

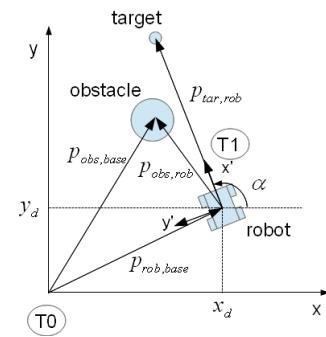


Fig. 12. Relations between frames

Figure 12 and transformation matrix (25) show the relationship between the robot frame T1 and the base frame T0

$$A_{10} = \begin{pmatrix} \cos(\alpha) & -\sin(\alpha) & x_d \\ \sin(\alpha) & \cos(\alpha) & y_d \\ 0 & 0 & 1 \end{pmatrix} \quad (25)$$

Let  $p_{obs,base}$  be the obstacle coordinates in the base frame. Then the following steps to compute the streamline array in base frame coordinates  $(\mathbf{x}_{rob}, 1) = (x_{rob}, y_{rob}, 1)^T$  are

1. Transformation of the obstacle coordinates  $p_{obs,base}$  from the base into the robot frame

$$p_{obs,rob} = A_{10}^{-1} \cdot p_{obs,base} \quad (26)$$

2. Calculate the velocities along the streamlines  $\mathbf{v}_{rob}(k)$ ,  $k$  - discrete time step, from eqs. (9) and (10) in T1 and the corresponding streamline array  $\mathbf{x}_{rob}(k)$  where  $\mathbf{x}_{rob}(k+1) = \mathbf{x}_{rob}(k) + \mathbf{v}_{rob}(k) \cdot \Delta t$  and  $\Delta t = t(k) - t(k-1)$ .
3. Transform the streamline array  $\mathbf{x}_{rob}$  into the base frame T0

$$\mathbf{x}_{base}(k) = A_{10} \cdot \mathbf{x}_{rob}(k) \quad (27)$$

After that, that streamline is selected for the robot to move along which is the closest streamline to the original predefined trajectory. In order to get a smooth connection to the original trajectory appropriate transition filters are used [19].

## VI. CHANGING OF STREAMLINES DURING OPERATION (LANE HOPPING)

### A. Lane hopping using the dataset model

The change of a streamline during operation (lane hopping) becomes feasible if several streamlines are computed in advance. Once a streamline is selected for a platform it may be necessary to change to another lane because of the following reasons:

1. The platform is too close to a static obstacle
2. The actual streamline violates hard/soft constraints
3. The platform is expected to collide with another moving obstacle (platform)

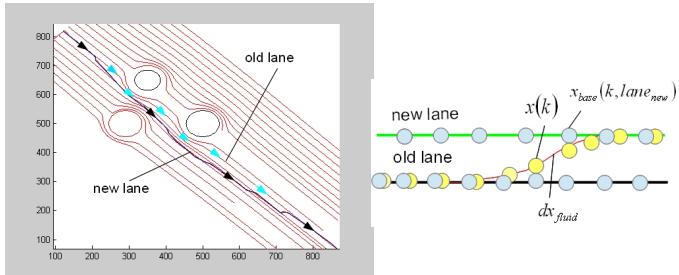


Fig. 14. Principle of lane hopping

Fig. 13. Change of streamline (lane hopping)

"Lane hopping" means the change from the current streamline to another streamline which may be a neighboring streamline but not necessarily. A smooth transition (see Fig. 14) is realized by a filter function either in the robot or world frame.

$$\begin{aligned} \mathbf{d}\mathbf{x}_{fluid}(k+1) &= \\ K_{filt} \cdot (\mathbf{x}_{base}(k+1|lane_{new}) - \mathbf{x}(k)) \\ \mathbf{x}(k+1) &= \mathbf{x}(k) + \mathbf{d}\mathbf{x}_{fluid}(k+1) \end{aligned} \quad (28)$$

where it is assumed that the x-positions in the robot frame  $x_{rob}(k|lane_{old}) \approx x_{rob}(k|lane_{new})$ . If  $x_{rob}(k|lane_{old}) \neq x_{rob}(k|lane_{new})$  then (28) is corrected by

$$\mathbf{d}\mathbf{x}_{fluid}(k+1) = K_{filt} \cdot (\mathbf{x}_{base}(k+\delta|lane_{new}) - \mathbf{x}(k)) \quad (29)$$

$\delta$  is the number of time steps for which

$$x_{rob}(k|lane_{old}) \approx x_{rob}(k+\delta|lane_{new}) \quad (30)$$

For a global (centralized) control of a group of robots it is possible to compute possible collisions of platforms in advance. Let us compare the 5 lanes each of platforms 1 and 2 and calculate the discrete time stamps at their crossings, and the difference between these time stamps. Let for example robot 1 move on lane 5 and robot 2 on lane 2. Lanes 5 and 2 cross at  $t = 367$  for robot 1 (see Fig. 15, matrix K12, blue circle) and for robot 2 at  $t = 369$  (see matrix J12, blue circle). The distance between the two entries is 2 (see Fig. 15, matrix del12) which points to a collision at time  $t \approx 367$ . In order to avoid a collision many different options are possible. We have chosen the following option: robot 1  $\rightarrow$  lane 4, robot 2  $\rightarrow$  lane 1. The result can also be observed in Fig. 15, red circles. The difference (distance) between the time stamps  $t = 316$  for robot 1 and  $t = 366$  for robot 2 is 50 which is sufficient for avoiding a collision.

Time for robot 1 at lane crossing						Delta t between robot 1 and 2 at lane crossing						
Lane	1	2	3	4	5	Robot 2	Lane	1	2	3	4	5
K <sub>12</sub> =	1 302	315	328	346	362		1 106	155	271	163	54	
	2 306	320	332	351	367		2 88	136	253	144	35	
	3 311	324	345	397	443		3 69	118	218	60	81	
	4 316	341	379	454	506		4 50	69	142	49	183	
	5 330	367	419	576	630		5 5	46	231	344		

Time for robot 2 at lane crossing						
J <sub>12</sub> =	408	470	599	509	416	
	394	456	585	495	402	
	380	442	563	457	362	
	366	410	521	405	323	
	335	369	465	345	286	

no collision at t=316      collision at t=367

Fig. 15. Example (dataset), with/without lane hopping

### B. Lane hopping using the time clustering model

In the case of the time clustering model the time points for crossing are computed in the same way as for the dataset model. The only difference is that the cluster centers are equidistant in time and sparsely distributed along the streamline in contrast to the dataset model. The precision of the results depends highly on the number of cluster centers. Figure 17 shows the 3 robot / 3 obstacles scenario with 20 cluster centers for each streamline. Figure 18 illustrates in detail the corresponding pairs of cluster centers for selected streamlines. The numbers (5,2) of the pairs can be read as follows:

robot 1 on lane 5 (time step 350) AND

robot 2 on lane 2 (time step 350)

are the closest possible positions near the (theoretical) crossing point of the streamlines 5 (robot 1) and 2 (robot 2). The time steps are included in the cluster center data triple  $(x, y, t)$ . Figure 16 shows the matrices for robot 1 and 2 and the combination for possible collisions: (5,1), (5,2). An additional safety distance of 50 steps leads to more pairs of a possible collision:

(4,1),(4,2),(5,3),(3,4),(4,4),(1,5),(2,5).

Therefore, the robots are supposed to move from (5,2) to (3,1)

which means that they reach the new crossing point with a safe time distance of 100 time steps.

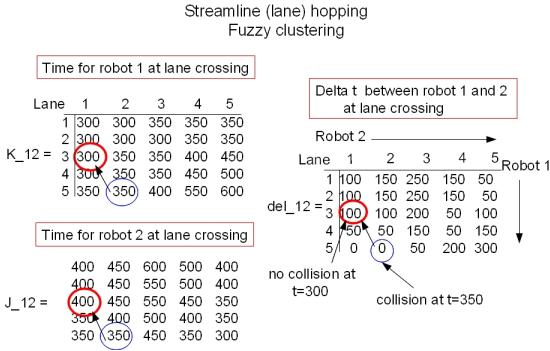


Fig. 16. Example (fuzzy), with/without lane hopping

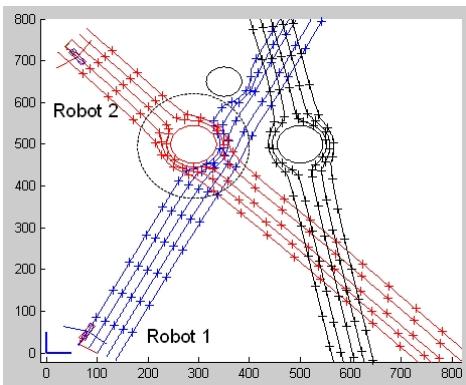


Fig. 17. Lanes with fuzzy clusters

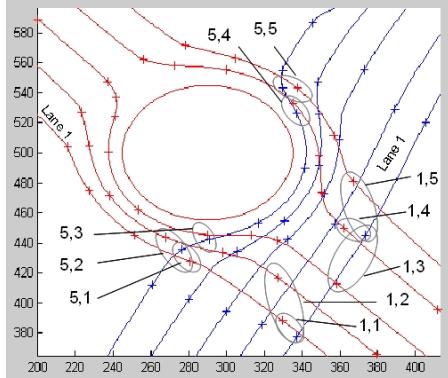


Fig. 18. Fuzzy clusters, detail

## VII. SIMULATION RESULTS

First, a comparison between the artificial force potential (see [1]) and the velocity potential (see Fig. 19) is presented. Here it can be observed that the force potential leads to an unnecessary wide circumvention around the obstacles while the velocity potential results in a better energy saving [19].

In Fig. 20 an example of a fuzzy model is shown, where a robot moves between three obstacles. The model is used for each obstacle separately, to generate a robot trajectory step by step by blending the individual streamlines using weighting

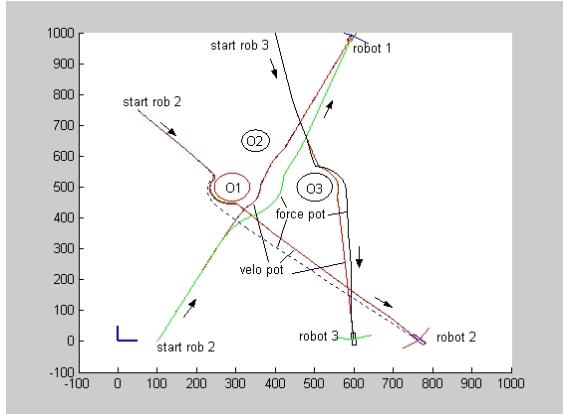


Fig. 19. Comparison force/velocity potential

function (11), (streamlines for obstacle 1 = blue, obstacle 2 = black, obstacle 3 = green). At the lower part of Fig. 20 the streamlines of the fuzzy model together with the cluster centers marked by + are shown.

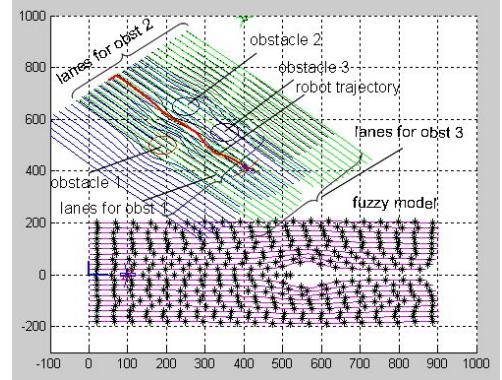


Fig. 20. Blending of streamlines, dataset model

The next simulation results consider lane hopping with dataset modeling and fuzzy modeling. Figure 21 shows the imminent collision of robots 1 and 2. Figure 22 and 23 show the lane hopping for the dataset model and the fuzzy model, respectively. Here we observe a higher built-in safety for the fuzzy case, which is necessary because of the rough calculation of the crossing points.

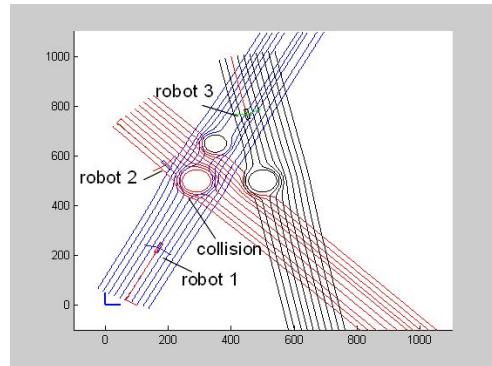


Fig. 21. Simulation example, no lane hopping

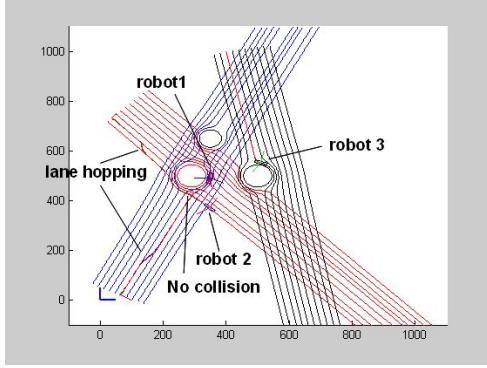


Fig. 22. Simulation example, with lane hopping

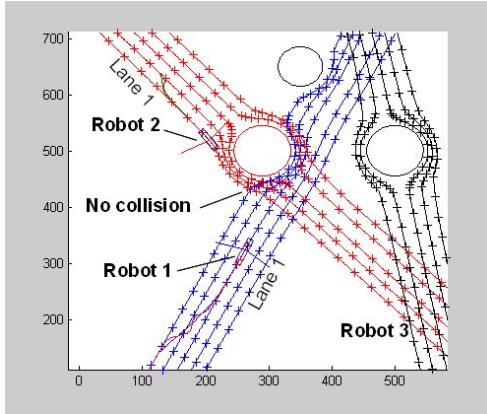


Fig. 23. Simulation example (fuzzy), with lane hopping

## VIII. CONCLUSIONS

The velocity potential of an incompressible fluid is an important tool for obstacle avoidance of mobile robots. Sets of streamlines, based on the velocity potential, are used for the avoidance of static cylindrical obstacles. Several distributed obstacles can be tackled by the so-called *flow-velocity average*. Big non-circular obstacles (super obstacles) can be modeled by the *flow potential average*. The high computational effort of the method is mainly determined by equations (9, 10, 26, 27) to be computed for n streamlines and m time steps at the moment of the detection of an obstacle. Therefore, dataset models and fuzzy models are generated to save memory and to accelerate the online computation of streamlines. To avoid possible collisions between robots that move on crossing streamlines a change between streamlines during operation (lane hopping) is presented. This is done for dataset models and fuzzy models. A future work lies in the improvement of the streamline models and to implement this approach for real applications.

## REFERENCES

- [1] O. Khatib. Real-time Obstacle avoidance for manipulators and mobile robots. *IEEE Int. Conf. On Robotics and Automation, St. Louis, Missouri, 1985*, page 500505, 1985.
- [2] R.Palm and A. Bouguerra. Navigation of mobile robots by potential field methods and market-based optimization. *ECMR 2011*, Oerebro, Sweden., 2011.
- [3] R. Palm and A. Bouguerra. Particle swarm optimization of potential fields for obstacle avoidance. In *Proceeding of RARM 2013, Istanbul, Turkey*. Volume: Scient. coop. Intern. Conf. in elect. and electr. eng., Sept. 2013.
- [4] J. Borenstein and Y. Koren. The vector field histogram - fast obstacle avoidance for mobile robots. *IEEE Trans. on Robotics and Automation, Vol. 7, No 3*, pages 278–288, June 1991.
- [5] J. Michels, A. Saxena, and A. Y. Ng. High speed obstacle avoidance using monocular vision and reinforcement learning. *22nd Int'l Conf on Machine Learning (ICML), Bonn, Germany*, 2005.
- [6] A. Martinez, E. Tunstel, and M. Jamshidi. Fuzzy logic based collision avoidance for a mobile robot. *Robotica, Vol. 12, Part 6*:pp. 521–527, Nov.-Dec. 1994.
- [7] P. G. Zavlangas and S. G. Tzafestas. Motion control for mobile robot obstacle avoidance and navigation: a fuzzy logic-based approach. *Journal Systems Analysis Modelling Simulation, Volume 43, Issue 12*, pages 1625–1637, Dec. 2003.
- [8] John H. Lilly. Evolution of a negative-rule fuzzy obstacle avoidance controller for an autonomous vehicle. *IEEE TFS, Vol. 15, No. 4*, pages 719–839, Aug. 2007.
- [9] P. E. Stigaru and F. L. Lewis. Motion path planning for mobile robots. *Report:University of Texas at Arlington, Automation and Robotics Research Institute.*, <http://arr.utexas.edu/acs/ee5322/lectures/Motion/planning.pdf>:1–7, April 2007.
- [10] Xi Li and Byung-Jae Choi. Obstacle avoidance of mobile robot by fuzzy logic system. In *Proc. of the ISA 2013, ASTL Vol. 21*, pages 244–246, 2013.
- [11] P. K. Khosla and R. Volpe. Superquadric avoidance potentials for obstacle avoidance. In *IEEE Conference on Robotics and Automation, Philadelphia PA*. IEEE, Springer-Verlag London, April 1988.
- [12] Jin-Oh Kim and P. K. Khosla. Real-time obstacle avoidance using harmonic potential functions. *IEEE Trans on Robotics and Automation,,* pages 1–27, 1992.
- [13] Z. X. Li and T. D. Bui. Robot path planning using fluid model. *Journal of Intelligent and Robotic Systems, vol. 21*, pages 29–50, 1998.
- [14] S.S. Ge and Y.J. Cui. Dynamic motion planning for mobile robots using potential field method. *Autonomous Robots, 13*, pages 207 – 222, 2002.
- [15] S. Waydo and R. M. Murray. Vehicle motion planning using stream functions. In *Proc. IEEE Int. Conf. Rob. Autom., volume 2*. IEEE, 2003.
- [16] R. Daily and D. M. Bevly. Harmonic potential field path planning for high speed vehicles. *American Control Conference, 2008, Seattle, Washington, USA*, pages 4609 – 4614, 2008.
- [17] S. Sugiyama, J. Yamada, and T. Yoshikawa. Path planning of a mobile robot for avoiding moving obstacles with improved velocity control by using the hydrodynamic potential. *IEEE/RSJ Intern. Conf. on Intell. Robots and Systems*, page 207222, 2010.
- [18] D. Gingras, E. Dupuis, G. Payre, and J. Lafontaine. Path planning based on fluid mechanics for mobile robots using unstructured terrain models. In *Proceedings ICRA 2010*. ICRA 2010, 2010.
- [19] R. Palm and D. Driankov. Fluid mechanics for path planning and obstacle avoidance of mobile robots. In *Proc. of the 11th Intern. Conf. on Informatics in Control, ICINCO, Automation and Robotics, Wien 09/2014*, pages 231–238. INSTICC.
- [20] N. E. Leonard and E. Fiorelli. Virtual leaders, artificial potentials and coordinated control of groups. *Proc. of the 40th IEEE Conf. on Decision and Control, Orlando, Florida USA*, pages 2968–2973, Dec 2001.
- [21] Y. Nakayama. Introduction to fluid mechanics. *Butterworth-Heinemann, Oxford Auckland Boston*, 1999.
- [22] R. Palm and Ch. Stutz. Open loop dynamic trajectory generator for a fuzzy gain scheduler. *Engineering Applications of Artificial Intelligence, Vol. 16*:213–225, 2003.
- [23] R. Palm and B. Iliev. Learning of grasp behaviors for an artificial hand by time clustering and takagi-sugeno modeling. In *Proceedings FUZZ-IEEE 2006 - IEEE International Conference on Fuzzy Systems, Vancouver, BC, Canada, July 16-21 2006*. IEEE.
- [24] D.E. Gustafson and W.C. Kessel. Fuzzy clustering with a fuzzy covariance matrix. *Proceedings of the 1979 IEEE CDC*, pages 761–766, 1979.