Soft Computing manuscript No.
(will be inserted by the editor)

Fast Marching based Globally Stable Motion Learning

Javier V. Gomez · David Alvarez · Santiago Garrido · Luis Moreno

Received: date / Accepted: date

Abstract In this paper a novel motion learning method is introduced: Fast Marching Learning (FML). While other learning methods are focused on optimising probabilistic functions or fitting dynamical systems, the proposed method consists on the modification of the Fast Marching Square (FM^2) path planning algorithm. Concretely, FM^2 consists of expanding a wave through the environment with a velocity directly proportional to the distance to the closest obstacle. FML modifies these velocities in order to generalise the taught motions and reproduce them. The result is a deterministic, asymptotically globally stable learning method free of spurious attractors and unpredictable behaviours. Along the paper, detailed analysis of the method, its properties and parameters are carried out. Comparison against a state-of-the-art method and experiments with real data are also included.

Keywords Fast Marching · motion learning · path planning · kinesthetic teaching

1 Introduction

The development of human-like robots has led to an increase in the number of degrees of freedom, which makes the planning and control tasks very difficult to be accomplished in real time. Nowadays, it is common to have redundant manipulators carrying out complex tasks in which conventional control over the inverse kinematics is not enough for successfully completing a given motion.

This work is supported by the Spanish Ministry of Science and Innovation under the projects DPI2010-17772 and CSD2009-00067.

RoboticsLab. Carlos III University of Madrid Avda. de la Universidad 30, 28911, Leganes, Madrid, Spain.
Tel.: +34-916248812
E-mail: jvgomez,dasanche,sgarrido,moreno@ing.uc3m.es

The main trend to overcome this problem is the use of learning techniques. Learning algorithms try to identify and generalise the relevant features of a motion in order to be able to reproduce previously given experience, even if the environment has changed or the motion query is not the same as the one taught.

In this paper we focus on programming by demonstration (PbD): the robot is given a set of demonstrations as an input for the learning algorithm. In PbD the demonstrations can be provided either by observing a demonstrator doing a task or by physical guiding of the robot during the task (kinesthetic teaching). While the first method requires the system to handle the re-targeting problem, the kinesthetic teaching method simplifies the problem using the same embodiment for both demonstration and reproduction. At the end, a set of motions (usually given as point-to-point trajectories) is given as input to the learning algorithm.

The motion learning objectives is commonly to execute a specific task as it was previously taught to the robot. Two different task types can be distinguished: 1) to execute a motion in which accuracy is not a critical point but the motion dynamics is, 2) to execute a motion which surely reaches a specific point of the space (the importance of dynamics depends on the problem).

The first task type focuses on teaching the robot how to complete a given task with no specific initial or final points, e. g. ball-in-a-cup game [21] or hitting a table tennis over the net [20]. In these cases, Dynamic Motion Primitives (DMP) [29,23], a set of nonlinear differential equations which creates smooth control policies have become very popular [22]. For learning the primitives, reinforcement learning has played an important role during the last years [15]. A more recent approach based on the motion primitives idea is proposed in [16], where the primitives learning is carried out using incremental kinesthetic teaching by means of Hidden Markov Models (HMM).

The second task type consists of completing a given motion in which there exists a specific goal but the initial states can change. Concretely, it faces the problem of showing the robot how to perform a discrete motion (i.e. point-to-point trajectories). The Stable Estimator of Dynamical Systems (SEDS) approach [13] is able to generalise and reproduce the demonstrations even when spatial and temporal perturbation appear. Calinon goes a step further proposing a control strategy for a robotic manipulator operating in unstructured environments while interacting with human operators [3]. Such situations are starting to be common in manufacturing applications. In fact, dynamical systems have shown to be a powerful alternative to model robot motions [9, 12] and different statistical approaches have been proposed: Gaussian Process Regression [27], Locally Weighted Projection Regression [28] and so on.

Other approaches have been proposed in order to leverage previous robot experience. For instance, obstacle rearrangement for faster replanning in a new environment [19] or the creation of a collision-free paths database so that paths can be reused in the future to speed up the planning process [2]. In these cases, the objective is to reduce the computational time when planning in a

high-dimensional space by previously training the robot with environment-path information. The consequence is that the computed paths will be similar to those given during the training phase. Other example is trajectory prediction [10, 11] which is also able to adapt path initialisations to new environments for optimal planning.

Most of the previous approaches have shown a good performance [1], but their underlying mathematical model is usually based on probabilistic terms, causing the learning to be stochastic. Depending on the problem to solve, this stochasticity may not be a desirable property since with the same demonstrations different solutions are given each time. It may also not converge to a solution, becoming unstable under certain conditions since stochastic optimization algorithms are used to solve the learning. Besides, most of these approaches are based on learning motion control parameters. To include changes in the environment becomes challenging [14].

This paper largely extends our previous work on motion learning [7]. We are introducing the Fast Marching Learning (FML) method: a deterministic, asymptotically globally stable motion learning algorithm designed from a path planning point of view, using the Fast Marching Square (FM^2) [4] algorithm as the underlying path planning method. Concretely, in this paper we improve the formulation of the approach and extend the application of the algorithm to non-static environments. Also, a deep study about its characteristics is included and the analysis of the results is also extended.

The rest of the paper is organised as follows. Next section describes the FM^2 path planning method in which the FML method relies on. The main contributions are detailed in section 3, in which the FML algorithm is detailed and some simulation results are shown. Section 4 includes an in-depth analysis of the proposed method, its characteristics and parameters. Experimental evaluation and comparison against SEDS method are shown in section 5. Finally, section 6 outlines the main conclusions of the paper and the future work.

2 Fast Marching Methods in Path Planning

The Fast Marching Square (FM^2) method is a robust, efficient algorithm to compute safe and smooth trajectories [4]. The powerfulness of this algorithm has been shown during the last years since it has been successfully applied to many different motion planning problems such as robot formations planning, motion learning, roadmap generation, etc. [32, 6]. It has also been applied to underwater autonomous robot path planning [24] or creation of shape features vectors [26] by fast geodesic extraction [25]. Since this method is well described in the literature, we will outline the basis in the following lines.

The FM^2 method consists on applying twice the Fast Marching Method (FMM) proposed by J. A. Sethian [30]. The objective of the FMM is to approximate distances maps. In other words, given a point in a space, it computes the distance from this point to the rest of the points in the space. It provides a fast, approximated solution by simulating a wave front propagation

through non-homogeneous media, in which the propagation speed depends on the current position of the wave front.

Let us assume a 2-dimensional binary grid map. The wave source is given a value $T_0 = 0$. FMM solves $T_{i,j}$ for each grid cell using the discrete Eikonal equation [32]:

$$max\left(\frac{T-T_1}{\triangle x},0\right)^2 + max\left(\frac{T-T_2}{\triangle y},0\right)^2 = \frac{1}{F_{i,j}^2} \tag{1}$$

where $\triangle x$ and $\triangle y$ are the grid spacing in the x and y directions, $F_{i,j}$ is the wave propagation speed for grid cell (i,j) and

$$\begin{aligned} T &= T_{i,j} \\ T_1 &= min(T_{i-1,j}, T_{i+1,j}) \\ T_2 &= min(T_{i,j-1}, T_{i,j+1}) \end{aligned} \tag{2}$$

The first step of FM^2 is to compute the velocitiy map $\mathbf{F}$. Each point of the space is given a relative velocity directly proportional to the distance to the closest obstacle. In order to calculate $\mathbf{F}$ efficiently, the FMM method is applied to the whole workspace, using obstacles as wave sources. In fact, an approximation to the distance transform by applying the FMM is computed [5].

The second step is to apply the FMM from the goal point and expand the wave until it reaches the current robot position (initial point). In this case, the distances map created is interpreted as a time-of-arrival map $\mathbf{T}$, in which every point of the space is assigned a value which represents the time it took to the wave to reach this point from the source point while being restricted to $\mathbf{F}$.

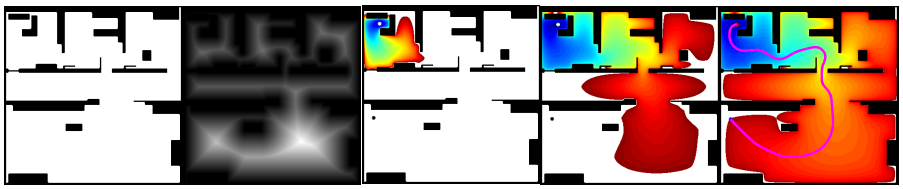
The final path is obtained by applying gradient descent on $\mathbf{T}$ from the initial point until the goal point (global minimum) is reached. Since $\mathbf{T}$ has no local minima, the goal point will be always reached. Figure 1 illustrates the different steps of the proposed algorithm.

The main properties of the paths computed with the FM^2 method are: smoothness, free of local minima and sub-optimal obstacle clearance. Assuming that the robot moves at a relative speed according to the velocitiy map, the provided path is optimal in terms of execution time [32].

Assuming that $\mathbf{F}$ contains relative velocities between 0 and 1, it is possible to trim (saturate) this velocitiy map. With this small modification, the safety and smoothness of the computed paths is still ensured (except for saturation values close to 0), while obtaining trajectories closer to the optimal one in terms of distance. Examples are shown in figure 2.

3 Path Planning Learning with Fast Marching Square

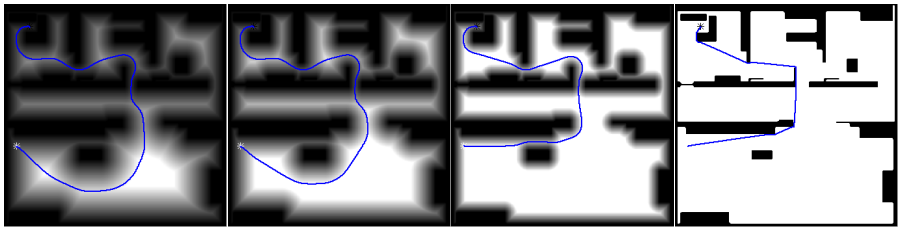
The FM^2 paths tend to go through those places in which the propagation velocity is higher, since it means that the total path can be covered in less



(a) (b) (c) (d) (e)

Fig. 1 Steps of the FM^2 algorithm. a) Initial binary map. b) velocitiy map generated with FMM. c),d) Wave propagation from the goal point. e) Final path shown over the time-of-arrival map.

time. This fact can be exploited by *forcing* the path to take a specific direction if the map **F** is carefully modified. Therefore, the objective is to encode the experience given to the robot by an expert in the velocitiy map. The consequence is that the final paths could be completely different to those given by the standard FM^2 method. However, the main characteristics of the FM^2 method will remain, such smoothness and local-minima-free.



(a) Saturation: 0.75 (b) Saturation: 0.5 (c) Saturation: 0.25 (d) Saturation: 0

Fig. 2 FM^2 saturated variation: modification of the path depending on the saturation value.

This paper is focused on point-to-point demonstrations: taught trajectories are codified as points in the workspace. Therefore, the principal objective of motion learning is to be able to successfully reproduce the taught motion when the robot is asked for a similar plan. The experience is expected to be generalised while improving the motion taught by making it faster, more efficient, smoother, etc.

3.1 Fast Marching Learning Method

The algorithm described next takes as input data gathered during a kinesthetic teaching process. Although it is applicable to any number of dimensions, end-effector's Cartesian coordinates will be used in order to help the reader to understand the methodology. In this case, the re-targeting problem is avoided,

as the dataset (set of end-effector's positions) is recorded in Cartesian coordinates. Every taught path P will contain a set of N three-dimensional points $p(x, y, z)$ sampled with a constant cycle time T. If K paths are taught to the system, the experience E can be codified then as:

$$E = \langle P_1, P_2, ..., P_K \rangle \tag{3}$$

where

$$P_i = \langle p_{i,1}, p_{i,2}, ..., p_{i,N_i} \rangle \tag{4}$$

The environment representation is the same as for FM^2, an n-dimensional cell grid. An empty workspace is assumed. However, obstacles can be directly included in the algorithm as shown in sections 3.2 and 4.1.4. Hence, the steps of the Fast Marching Learning (FML) method are the following:

1. All the points contained in E are labelled as 1 (white) in an workspace with no data (represented in black, value 0). This workspace is denoted as $\mathbf{F_p}$.
2. Apply the dilation operation on $\mathbf{F_p}$ by the size *aoi*, measured in pixels (or voxels), which defines the *area of influence* of the taught data.
   These two steps *divide* the workspace into two different zones: those affected by the previous experience, and those not influenced (where the algorithm will behave as a regular path planner). The *aoi* parameter sets the size of these zones since it is responsible for dilating the initial demonstrations.
3. The FMM is applied as done in the first step of the FM^2 method, so that $\mathbf{F_p}$ is converted into a velocitiy map. All the zero-valued points of $\mathbf{F_p}$ are used as wave sources.
4. By linearly rescaling $\mathbf{F_p}$ in order to be within the bounds defined by $[sat, 1]$ the final velocitiy map $\mathbf{F}$ is obtained. This second parameter *sat* is a saturation which weights the importance of the new data against the rest of the environment.
   At this point, $\mathbf{F_p}$ contains a generalisation of the demonstrations. Those areas with higher value (lighter) represent, in an intuitive manner, the center of the demonstrations.
5. Apply FMM over the entire workspace using as unique wave source point the centroid of the final point of all the trajectories $P_i \in E$, and considering the velocitiy map $\mathbf{F}$.

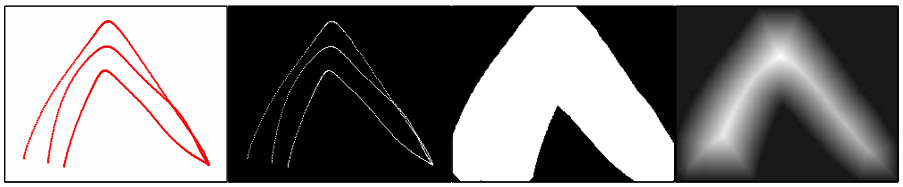
This algorithm is formalised in algorithm 1. DILATE(*map*,*s*) applies the morphological dilation operation on *map* with a structuring element with size *s*, given in pixels (*px*) or voxels (*vx*). Any shape of this structuring element is valid. For the examples of this paper the *ball* shape has been used. FASTMARCHING($\mathbf{F}$,x_S) applies the FMM using the velocitiy map $\mathbf{V}$ and the point *s* as wave source (*s* can be an array with more than one wave source). RESCALE(*map*,*min*,*max*) linearly rescales *map* between *min* and *max* values. Finally, CENTROID(E) takes as input a set of trajectories and output the centroid of the first point of all the demonstrations. Lines 1-11 create the

velocitiy map according to the demonstrations, as shown in figure 3. This way, the second FMM wave (lines 12-13) can travel through areas with no experience but with less priority to those affected by DILATE. In case of new path queries, paths will be *attracted* towards areas with experience since the second FM^2 wave will expand faster as depicted in figure 4.

Algorithm 1 The Fast Marching Learning algorithm

Input: Experience $E = \langle P_1, P_2, ..., P_K \rangle$.
Output: Modified velocitiy map $\mathbf{F}$, reproduction field $\mathbf{T}$.

1: $\mathbf{F_p} \leftarrow \{0\}$;
2: **for** $i = 1$ to K **do**
3: **for** $j = 1$ to N_i **do**
4: $x \leftarrow P_{i,j}$;
5: $\mathbf{F_p}(x) := 1$;
6: **end for**
7: **end for**
8: $\mathbf{F_p} \leftarrow \text{DILATE}(\mathbf{F_p}, aoi)$;
9: $x_S \leftarrow \{\forall x \in \mathbf{F_p} | \mathbf{F_p}(x) = 0\}$;
10: $\mathbf{F_p} \leftarrow \text{FASTMARCHING}(\mathbf{F_p}, x_S)$;
11: $\mathbf{F} \leftarrow \text{RESCALE}(\mathbf{F_p}, sat, 1)$;
12: $x_S \leftarrow \text{CENTROID}(E)$;
13: $\mathbf{T} \leftarrow \text{FASTMARCHING}(\mathbf{F}, x_S)$;
14: **return** $\mathbf{F}, \mathbf{T}$;



(a) Demonstrations (b) Initial $\mathbf{F_p}$ (c) Dilated $\mathbf{F_p}$ (d) $\mathbf{F}$

Fig. 3 Fast Marching Learning steps. $sat = 0.1$ and $aoi = 25px$.

3.2 Including Obstacles in the Workspace

In case that the workspace is not completely free, obstacles are required to be included. Obstacles can either be part of the workspace from the beginning or appear once after demonstrations were done. Thanks to the FM^2 underlying method, both methods can be easily addressed.

Let us assume that the robot has been given an experience E and a velocitiy map $\mathbf{F}$ has been already computed. Let also assume that the initial workspace $\mathbf{W_o}$ is not obstacle-free, or that it was free in the beginning but new obstacles appear. In this case, the velocitiy map of the workspace $\mathbf{F_{o,sat}}$ has to be

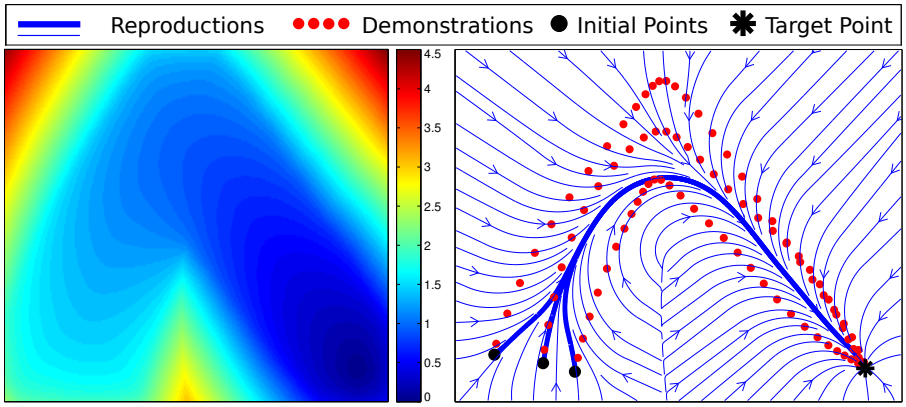


Fig. 4 Left: FM^2 time-of-arrival map $\mathbf{T}$ using the modified $\mathbf{F}$. Right: streamlines (set of possible reproductions) of $\mathbf{T}$ with parameters $sat = 0.1$ and $aoi = 25px$.

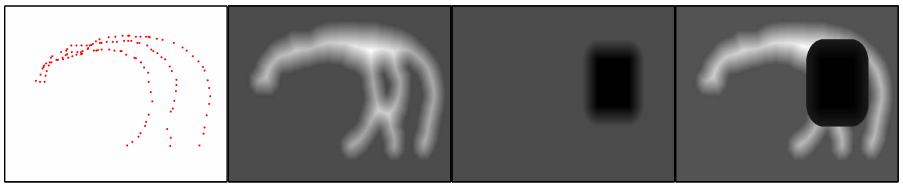
computed by saturating at level *sat*. Finally, in order to compute the final velocitiy map $\mathbf{F}$, the following update step is necessary for all those points in which $\mathbf{F_{o,sat}}$ is not saturated:

$$\mathbf{F} := min(\mathbf{F_{o,sat}}, \mathbf{F}) \quad \forall i \in \mathbf{F_{o,sat}} | \mathbf{F_{o,sat}}(i) < sat \tag{5}$$

This operation has to be repeated any time a new obstacle appears. The $\mathbf{T}$ needs also to be updated by propagating an FMM wave from the target point. It will take into account the demonstrations given to the robot as much as possible while avoiding the new obstacles. However, it would be worthy to recompute $\mathbf{F}$ from scratch with different, more restrictive parameters *sat* and *aoi* because the new obstacles will deviate the reproductions and the behaviour of the solution could change. The algorithm is detailed in figure 5 and its results are shown in figure 6.

4 Analysis of the Fast Marching Learning Method

In this section an in-depth analysis of the FML and its characteristics is carried out.



(a) Demonstrations (b) $\mathbf{F}$ (c) $\mathbf{F_{o,sat}}$ (d) Updated $\mathbf{F}$

Fig. 5 Fast Marching Learning obstacles update steps. $sat = 0.1$ and $aoi = 25px$.

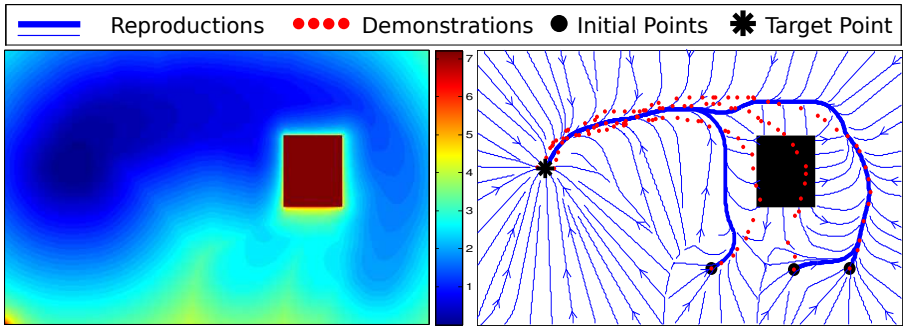


Fig. 6 Left: map of times using the propagation velocities learned. Right: result of the learning method with parameters $sat = 0.1$ and $aoi = 25px$.

4.1 FML Main Characteristics

4.1.1 Duality

Let us suppose that the taught paths have a starting point close a region A of the workspace and the final points are close to any region B of the same workspace. Usually, motion learning algorithms are designed supposing that the new queries (reproductions of the robot) will be in the same direction $A \rightarrow B$. In the case of a query which ask for a plan in the opposite direction, $B \rightarrow A$, the behaviour of other learning algorithms could be unexpected. However, in the case of FML the same motion as taught will be performed but in the opposite direction. Although this property could become a limitation under some circumstances, it allows to predict the behaviour of the robot which is a very desirable property regarding safety.

4.1.2 Determinism

The FMM is a deterministic method. This means that the output will remain always the same if the input does not change. Since FML is based on FMM, and the way $\mathbf{F}$ is computed is deterministic, FML is also deterministic. This is an important factor since the behaviour is easy to be predicted and no spurious behaviours will occur, which is common in probabilistic, optimization-based learning methods.

4.1.3 Behaviour with no experience

In other learning algorithms, if a motion query is done from a point far away from the given experience, the behaviour is often unpredictable. In those cases, the FML method will provide the fastest path from the starting point to the goal point according to the metrics given by the velocitiy map $\mathbf{F}$. Since it has a constant value of sat in those places away from the experience, the fastest

path also means the shortest path to the goal point in terms of distance. This is shown in figure 7.

4.1.4 One-shot Learning

One-shot learning refers to the characteristic of a learning system to be able to successfully reproduce and generalise the experience when only one demonstration has been given [18]. This is a desirable characteristic from the point of view of the final user of the system.

When many demonstrations are given to the FML method, the dilation step of the algorithm brings all these demonstrations together into one area of influence of the learning (depending on the *aoi* parameter). Therefore, many demonstrations act as just one demonstration with a larger *aoi*. If the robot is going to be taught with only one demonstration, then the *aoi* parameter has to be set with a larger value than when many demonstrations are provided.

Figure 8 shows an example of FML one-shot learning in an environment with one obstacle. Notice that in both cases, it is possible that the generated reproduction fields are very similar.

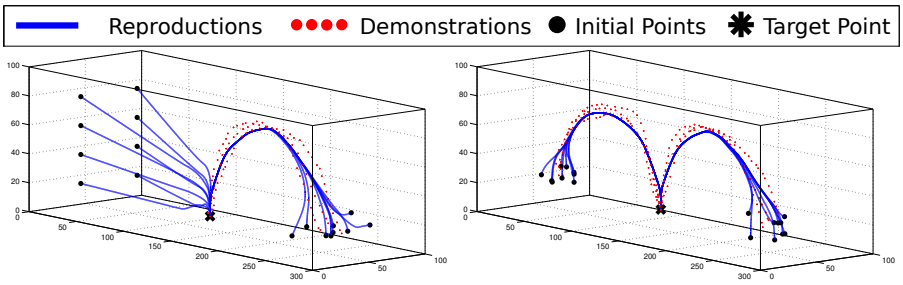
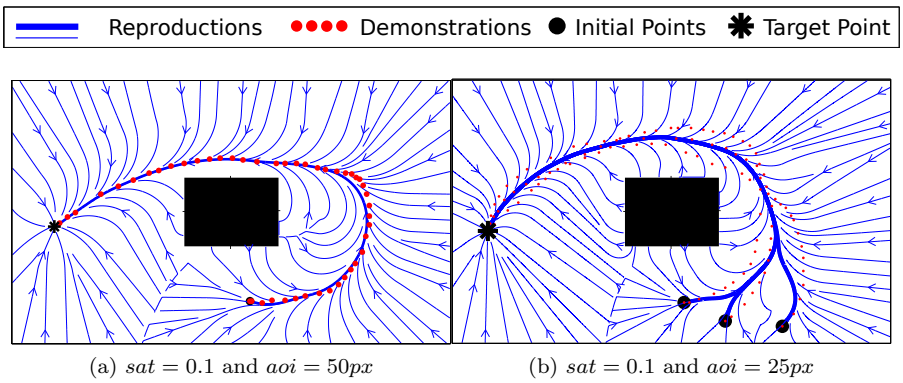


Fig. 7 Behaviour of the Fast Marching Learning algorithm in zones with no experience, and its evolution when new experience is included. $sat = 0.1$ and $aoi = 20vx$.



(a) $sat = 0.1$ and $aoi = 50px$ (b) $sat = 0.1$ and $aoi = 25px$

Fig. 8 One-shot against many demonstrations. Workspace: $300x500px$.

4.2 Stability Analysis

Although the FML method is not based on dynamical systems, we analyse its stability with and analogy to the the Lyapunov Stability theorem [31].This theorem expresses that a function $\dot{x} = f(x)$ is asymptotically stable at the point x_g if a continuous and continuously differentiable Lyapunov function $V(x)$ can be found such that it is always positive, its derivative is always negative and $V(x_g) = \dot{V}(x_g) = 0$.

Let us consider as Lyapunov function the one generated when expanding the second wave of FM^2, $T(x)$, shown in equation (1). This function starts at the goal point of the robot x_g, where the $T(x_g)$ value is 0. Given the fact that this wave expands always with non-negative velocities, the value of $T(x)$ will be higher (positive) as the wave gets farther from x_g. Finally, the derivative of the function is always the same sign since $T(x)$ is free of local minima. Actually, the derivative of equation (1) is always positive as the time is always increasing from the wave source point. However, when running gradient descent from a given point, the path generated will follow the direction in which the time decreases the most. Therefore, in this case the Lyapunov conditions are satisfied to ensure a globally asymptotically stable system. In other words, all the motion reproductions of the FML will converge to the same point as the $T(x)$ function has a unique minimum.

Conceptually, FML is really close to the work presented in [17], where vector fields are created using different Lyapunov function candidates. Qualitatively, their results are close to the SEDS algorithm [13], but prone to have local attractors and unexpected behaviours. However, FML guarantees a globally stable system by numerically computing the Lyapunov function with FMM. This Lyapunov function is modified by parameters *sat* and *aoi*.

4.3 Parameters Analysis

The proposed FML method counts with two parameters whose configuration changes the behaviour of the learning procedure. In this section an intuitive explanation of their influence is given. Also, figure 9 includes the learning results when the same demonstrations but different set of parameters are given.

4.3.1 Saturation, sat

The possible values of the saturation are $sat \in (0, 1]$. This is the value of $\mathbf{F}$ in those places where no experience is given, which represents the propagation velocity of the FMM wave. According to the design of the FML algorithm, the places with experience have a higher propagation velocity. Therefore, the places with experience will be reached earlier by the wave.

The reproductions will only ignore the experience when it takes less time for the wave to reach the target points by propagating through the zones with no experience. Hence, the saturation parameter acts as a weighting factor

between the importance of the experience given to the robot and the rest of the space. Since the objective is for the robot to reproduce the motions that have been taught, this factor is usually close to zero. When an excessively high value is given to this parameter, the reproductions will ignore the experience (top row of figure 9). On the other hand, a extremely low value will result in a very greedy learning, where reproductions will always go to those zones with experience (bottom row of figure 9). Experimentally, good results are given when $sat \in [0.05, 0.1]$.

4.3.2 Area of influence, aoi

This is the size used in the dilation step of the FML, where all the recorded points are enlarged in order to give connectivity to the demonstrations and its spatial surroundings. It is measured in cells (pixels or voxels) and directly depends on the size of the workspace. Experimentally, it has been found that a good value for this parameter is around 5% for multiple demonstrations and 10% for one-shot learning (percentage given over the smallest dimension of the workspace).

This parameter affects the generalisation of the learning. When an excessively low value is given, the algorithm will not generalise and it will follow the taught trajectories strictly (left column of figure 9). In the opposite case, the reproductions will generalise too much and the shape of the demonstrations will be lost (right columns of figure 9).

Special attention is required by the *aoi* parameter when working in dimensions which are not in the same domain. This paper is focused on working in end-effector's Cartesian coordinates, so the three dimensions are in the same spatial domain. However, when using other dimensions, i. e. velocities or angles, the size of this parameter has to be coherent with the desired result.

Figure 10 shows examples of learning results with wrong parameters. In figure 10 a) reproduction queries are demanded from points which are close to the target point. In this case the weight of the experience has to be very high in relation to the rest of the space, so that by decreasing the value of *sat* the learning becomes successful (figure 10 b)). A different example is given in figure 10 c). In this case the excessively high value of *aoi* converts the N-shaped path into a much smoother shape. By decreasing the size of the *aoi* it is possible to keep the shape of the demonstrations, as shown in figure 10 d).

5 Experimental Evaluations

This section includes several demonstrations of the FML performance using real and simulated data. Since the proposed method does not include dynamics, it is not possible to carry out an exhaustive comparison against those learning methods based on dynamical systems. However, a brief comparison against the SEDS method [13] is included. It has been carried out over the handwriting motions described in the SEDS paper: 24 different handwriting

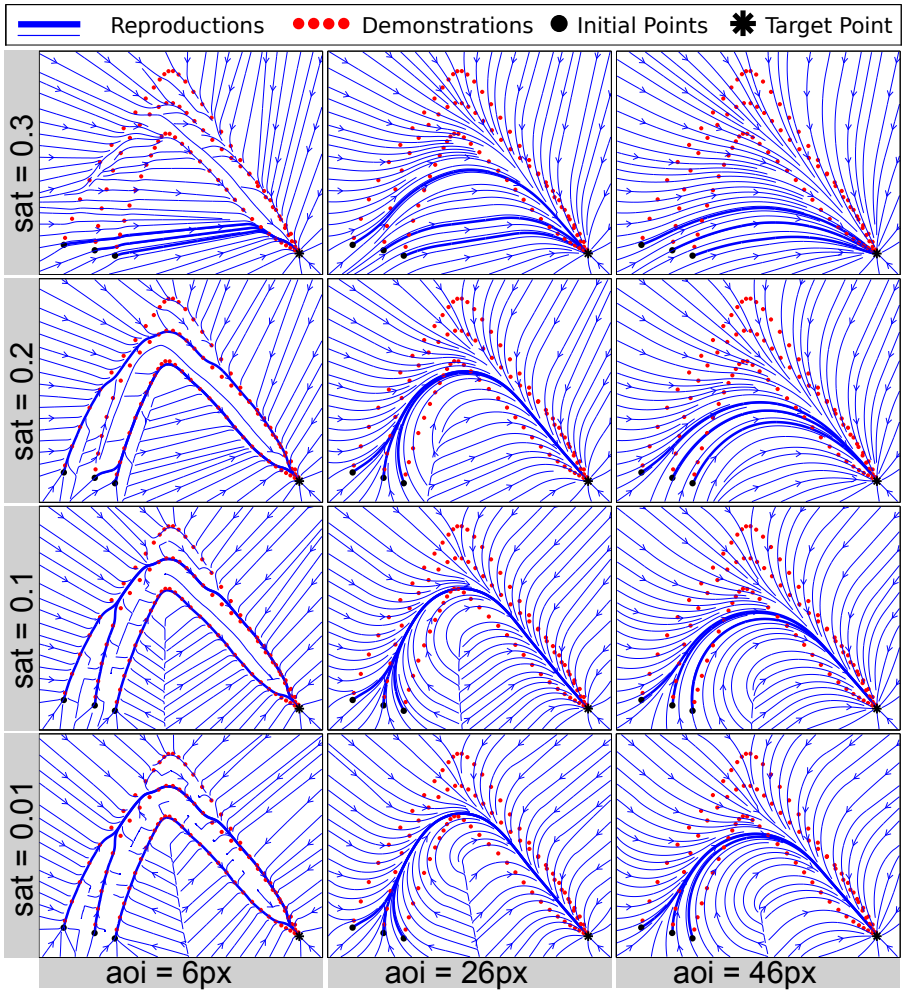


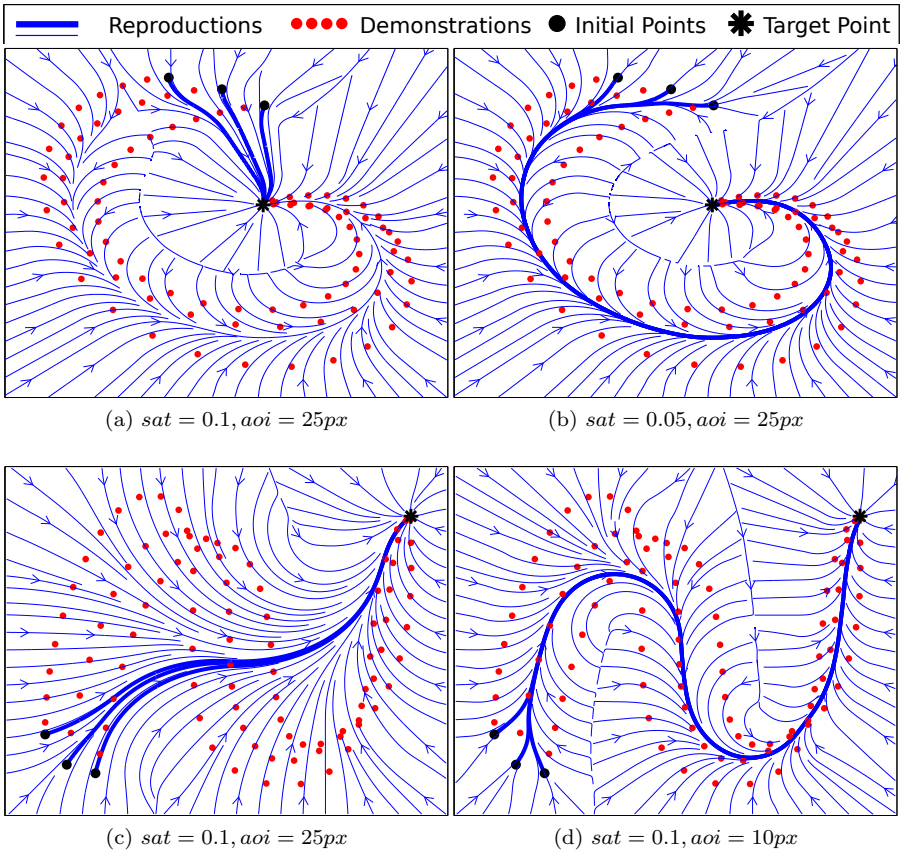
Fig. 9 Analysis of the results using different parameter settings. Workspace: $250x185px$.

motions collected with a Tablet-PC. Each motion is composed by three demonstrations with a uniform sampling time $T = 0.02s$.

Figure 11 shows the results of FML over 9 of the motions in the dataset. The last row of the figure is worthy to mention since it is composed by multi-model motions, in which the motion to learn changes completely depending on the area of the workspace. FML is able to generalise the demonstrations.

Examples in a three-dimensional space are shown in figure 12. The demonstrations given in this case are simulated, manually introduced.

The FML-SEDS comparison, shown in figure 13, focuses on the reproduction field of these methods. Both are running in two dimensions, in a workspace of $250x185px$. The parameter set in SEDS is: 4 Gaussian distributions and a likelihood optimisation method with a maximum of 500 iterations. In FML



(a) $sat = 0.1, aoi = 25px$ (b) $sat = 0.05, aoi = 25px$

(c) $sat = 0.1, aoi = 25px$ (d) $sat = 0.1, aoi = 10px$

Fig. 10 The shape of the trajectories to learn could influence the parameters of the algorithm. Workspace: $250x185px$.

parameters are, $sat = 0.1$ and $aoi = 25px$. These results show that the performance of both algorithms is quite different. While SEDS looks for a complete generalisation of the motion, following the same motion pattern from any point of the space, FML converges to the area with experience in a smooth way, creating motions always similar to the reproductions. However, the behaviour of SEDS in some areas is unexpected and may cause problems when operating in a real robot. This is likely to occur in those places close to the target point but in the opposite direction of the demonstrations or in places far away from the target. Table 1 includes the average and standard deviation of the computation times as an interesting result. However, we acknowledge this is very implementation-dependent.

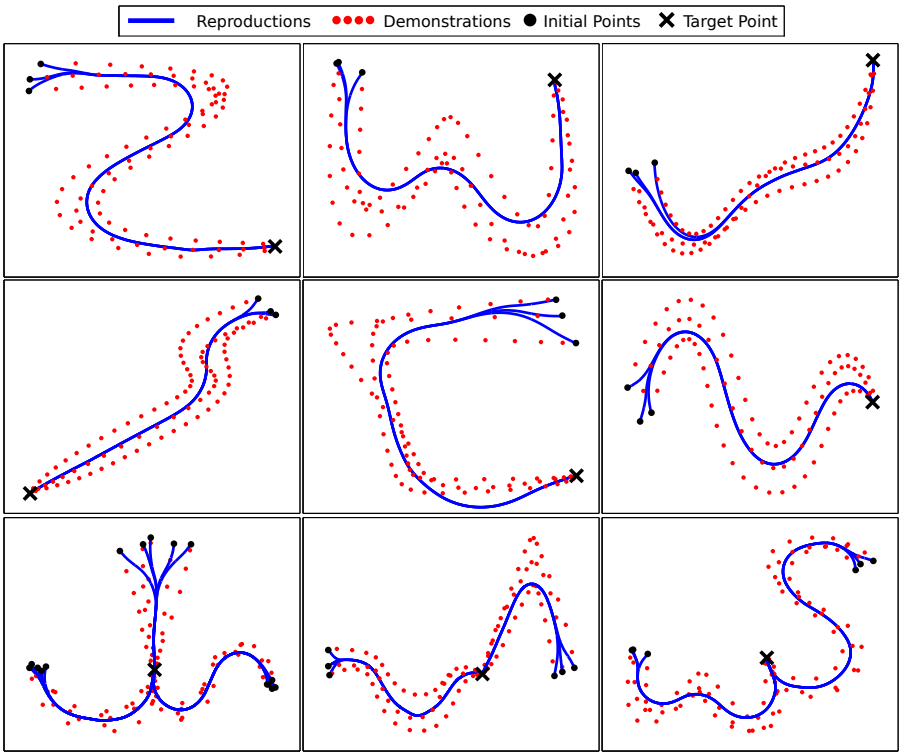


Fig. 11 Results of the Fast Marching Learning algorithm applied to handwriting motions. $sat = 0.1$ and $aoi = 25px$.

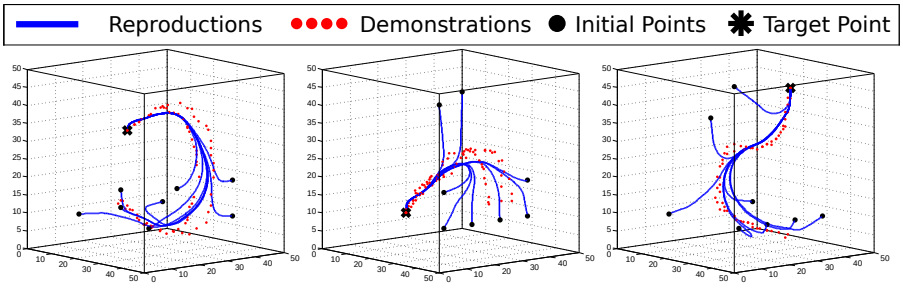


Fig. 12 Results of the Fast Marching Learning algorithm in three dimensions, $sat = 0.05$ and $aoi = 10vx$.

Table 1 Results of SEDS and FML in the handwriting motions dataset.

Times (s)			
FML		SEDS	
μ	σ	μ	σ
0.17	0.12	23.28	15.58

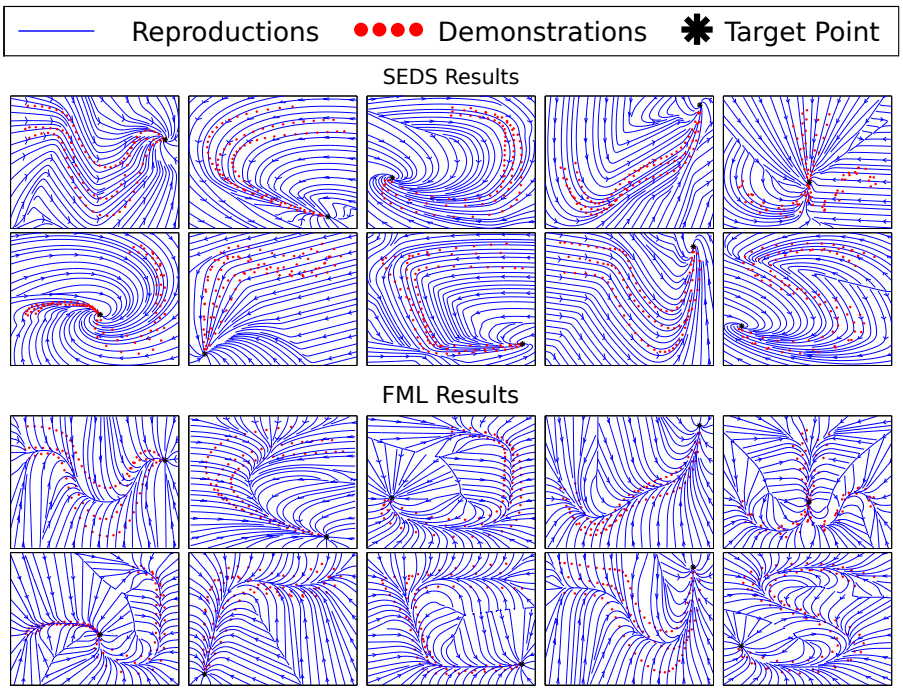


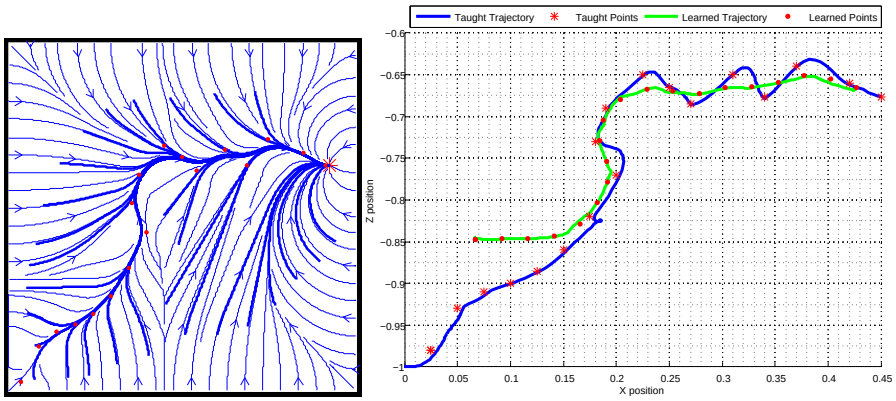
Fig. 13 Qualitative comparison of the learning results of algorithms SEDS and FML in the handwriting motions dataset.

5.1 Experiment on a real platform

In order to prove the feasibility of the proposed learning method, it has been implemented in the mobile manipulator Manfred V2, equipped with a 6 DOF arm. The kinesthetic teaching is done offline, placing the arm in different configurations and recording the Cartesian coordinates of the end-effector. The recorded data is not restricted in time, as only the coordinates are saved.

After this process, the FML algorithm is executed and the robot moves its arms towards the goal point from a different location. For better understanding, the arm is moved in a 2-dimensional plane and only the XZ coordinates of the end-effector are stored. For simplicity, and also test one of the main characteristics of the method, one-shoot learning is carried out. This is, the robot is taught only once since we assume that this is a desirable point by robot's end users.

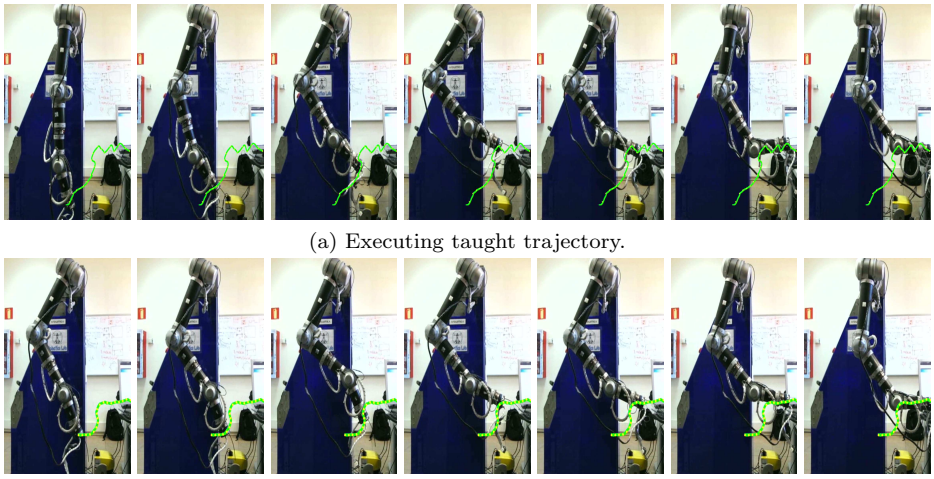
With the data learned, the $T(x)$ function show in figure 14 a) converges to the goal point independently of the starting point of the trajectory and resembling as much as possible to the learned trajectory. Figure 14 b) compares the two trajectories carried out by the robot (with the initial taught data and with the learned data), using $sat = 0.5$ and $aoi = 35$ pixels in a 500x500 pixels region (each pixel corresponds to 1 millimeter). It can be appreciated that the



(a) $T(x)$ function.

(b) Comparison between the taught and reproduced trajectories.

Fig. 14 FML results in a real robot.



(a) Executing taught trajectory.

(b) Executing FML reproduction.

Fig. 15 The robot Manfred V2 is able to successfully follow the FML trajectory which starts in a different location.

reproduction adapts, and even improves, the taught trajectory as it is shorter and smoother.

Finally, figure 15 includes the sequence of the robot Manfred V2 executing both trajectories. Since motion dynamics are not learned, as we aim to low-dimensional spaces, the robot is able to perfectly follow the reference trajectory.

6 Conclusions

Along this paper FML, a novel learning algorithm for robot motions, has been detailed. It has been shown that it can work well with an empty workspace and also with obstacles. It is not based on an optimisation procedure but on a well-studied path planning algorithm, FM^2. This means a completely different point of view of the previous work in motion learning.

A deep analysis on the performance of FML and the dependance on its parameters has been carried out. The main advantages of this method against others are: determinism, asymptotically globally stable, one-shot learning method and low computational time. Besides, experimental results show that FML is reliable and safe. No major problems have been identified.

A brief comparison with a state of the art method has been included. Other metrics could be employed, such as storage size, adaptation to online changes, accuracy reproducing demonstrations, etc. However, the nature of FML regarding the current learning algorithms is very different. Therefore a deeper comparison would not be meaningful. FML does not pretend to improve SEDS, DMP or other methods, but to propose an alternative in which simpler, robust solutions are required. For example, writing motions, door openings, boosting planning from experience (to be addressed in future work), and other problems in which the dynamics are not critical but the final trajectory is.

The main current drawback is that motion dynamics are not codified into the learning. Also, as FML is based on grid cell, the application to more than 3 dimensions can be very expensive in terms of computational time. However, the overall process can be speeded up by using more advanced FMM-like techniques [8].

Therefore, future work will focus on solving both of these problems. The anisotropic Fast Marching Method, together with parallel implementations are promising research lines which can improve the proposed method significantly. It is also worthy to explore optimisation methods in order to automatically set the parameters involved. Finally, studying the influence of the grid resolution in the reproduction field would be very valuable, as it would be possible to optimize the computation-time/reproduction-quality ratio.

References

1. Argall, B.D., Chernova, S., Veloso, M., Browning, B.: A survey of robot learning from demonstration. Robotics and Autonomous Systems **57**(31), 469–483 (2009)
2. Branicky, M., Knepper, R., Kuffner, J.: Path and trajectory diversity: Theory and algorithms. In: IEEE Intl. Conf. on Robotics and Automation, pp. 1359–1364 (2008)
3. Calinon, S., Sardellitti, I., Caldwell, D.G.: Learning-based control strategy for safe human-robot interaction exploiting task and robot redundancies. In: IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, pp. 249–254 (2010)
4. Garrido, S., Moreno, L., Abderrahim, M., Blanco, D.: FM2: A Real-time Sensor-based Feedback Controller for Mobile Robots. International Journal of Robotics and Automation **24**(1), 3169–3192 (2009)

5. Garrido, S., Moreno, L., Blanco, D., Martin, F.: Log of the inverse of the distance transform and fast marching applied to path planning. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 2882–2887 (2006)
6. Gómez, J.V.: Advanced Applications of the Fast Marching Square Planning Method. Master's thesis, Carlos III University (2012)
7. Gómez, J.V., Álvarez, D., Garrido, S., Moreno, L.: Kinesthetic Teaching via Fast Marching Square. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 1305–1310 (2012)
8. Gómez, J.V., Álvarez, D., Garrido, S., Moreno, L.: Fast methods for eikonal equations: an experimental survey. ArXiV **abs/1506.03771** (2015). URL http://arxiv.org/abs/1506.03771
9. Hersch, M., Guenter, F., Calinon, S., Billard, A.: Dynamical system modulation for robot learning via kinesthetic demonstrations. IEEE Trans. on Rob. **24**(6), 1463–1467 (2008)
10. Jetchev, N., Toussant, M.: Trajectory prediction in cluttered voxel environments. In: IEEE Int. Conf. on Robotics and Automation, pp. 2523–2528 (2010)
11. Jetchev, N., Toussant, M.: Fast motion planning from experience: trajectory prediction for speeding up movement generation. Autonomous Robots **34**(1-2), 111–127 (2013)
12. Khansari-Zadeh, S.M., Billard, A.: Imitation learning of globally stable non-linear point-to-point robot motions using nonlinear programming. In: IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems, pp. 2676–2683 (2010)
13. Khansari-Zadeh, S.M., Billard, A.: Learning Stable Nonlinear Dynamical Systems With Gaussian Mixture Models. IEEE Transactions on Robotics **27**(5), 943–957 (2011)
14. Khansari-Zadeh, S.M., Billard, A.: A dynamical system approach to realtime obstacle avoidance. Auton. Robots **32**(4), 433–454 (2012)
15. Kober, J., Bagnell, J.A., Peters, J.: Reinforcement Learning in Robotics: A Survey. Int. J. Rob. Res. **32**(11), 1238–1274 (2013)
16. Lee, A.D., Ott, C.: Incremental Motion Primitive Learning by Physical Coaching Using Impedance Control. In: IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, pp. 4133–4140 (2010)
17. Lemme, A., Neumann, K., Reinhart, R., Steil, J.: Neural learning of vector fields for encoding stable dynamical systems. Neurocomputing **141**, 3–14 (2014)
18. Li, F.F., Fergus, R., Perona, P.: One-Shot Learning of Object Categories. IEEE Trans. Pattern Anal. Mach. Intell. **28**(4), 594–611 (2006)
19. Martin, S., Wright, S., Sheppard, J.: Offline and online evolutionary bi-directional RRT algorithms for efficient replanning in dynamic environments. In: IEEE Int. Conf. on Automation Science and Engineering, pp. 1131–1136 (2008)
20. Mülling, K., Kober, J., Kroemer, O., Peters, J.: Learning to select and generalize striking movements in robot table tennis. Int. J. Rob. Res. **32**(31), 263–279 (2013)
21. Nemec, B., Zorko, M., Zlajpah, L.: Learning of a ball-in-a-cup playing robot. In: Int. Workshop on Robotics in Alpe-Adria-Danube Region, pp. 297–301 (2010)
22. Ning, K., Tamosiunaite, M., Worgotter, F.: A Novel Trajectory Generation Method for Robot Control. J. of Intelligent and Robotics Systems **68**(2), 165–184 (2012)
23. Pastor, P., Hoffmann, H., Asfour, T., Schaal, S.: Learning and generalization of motor skills by learning from demonstration. In: IEEE Intl. Conf. on Robotics and Automation, pp. 1293–1298 (2009)
24. Petres, C., Pailhas, Y., Patron, P., Petillot, Y., Evans, J., Lane, D.: Planning for Autonomous Underwater Vehicles. IEEE Trans. on Rob. **32**(2), 331–341 (2007)
25. Peyré, G., Cohen, L.D.: Heuristically Driven Front Propagation for Fast Geodesic Extraction. Int. J. Comp. Vis. and Biomechanics **1**(1), 55–67 (2008)
26. Rabin, J., Peyré, G., Cohen, L.D.: Geodesic Shape Retrieval via Optimal Mass Transport. In: European Conf. on Comp. Vis., pp. 771–784 (2010)
27. Rasmussen, C., Williams, C.: Gaussian processes for machine learning, 3rd edn. New York, Springer-Verlag (2006)
28. Schaal, S., Atkenson, C., Vijayakumar, S.: Scalable locally weighted statistical techniques for real time robot learning. Applied Intelligence **17**(1), 49–60 (2002)
29. Schaal, S., Peters, J., Nakanishi, J., Ijspeert, A.: Learning Movement Primitives. In: Intl. Symp. Rob. Res. (2004)

30. Sethian, J.A.: A Fast Marching LevelSet Method for Monotonically Advancing Fronts. Proceedings of the National Academy of Science **93**(4), 1591–1595 (1996)
31. Slotine, J., Li, W.: Applied Nonlinear Control. Englewood Cliffs, NJ: Prentice-Hall (1991)
32. Valero-Gómez, A., Gómez, J.V., Garrido, S., Moreno, L.: The Path to Efficiency: Fast Marching Method for Safer, More Efficient Mobile Robot Trajectories. IEEE Robotics and Automation Magazine **20**(4) (2013)