# SVM approximation of value function contours in target hitting problems

L. Chapel, G. Deffuant

Author-produced version of the paper published in Informatics in control, automation and robotics. . Ferrier, J.L. ; Bernard, A. ; Gusikhin, O. ; Madani, K. (eds). 8th International Conference, ICINCO 2011 Noordwijkerhout, the Netherlands, July 28-31, 2011
Available at http://www.springerlink.com, doi : 10.1007/978-3-642-31353-0_3

# SVM approximation of value function contours in target hitting problems

Laetitia Chapel[1] and Guillaume Deffuant[2]

[1] Lab-STICC, Université Européenne de Bretagne, Université de Bretagne Sud, 56017 Vannes Cedex, France laetitia.chapel@univ-ubs.fr
[2] Laboratoire d'Ingénierie pour les Systèmes Complexes, Cemagref, 63172 Aubière Cedex, France guillaume.deffuant@clermont.cemagref.fr

**Abstract.** In a problem of target hitting, the capture basin at cost $c$ is the set of states that can reach the target with a cost lower or equal than $c$, without breaking the viability constraints. The boundary of a $c$-capture basin is the $c$-contour of the problem value function. In this paper, we propose a new algorithm that solves target hitting problems, by iteratively approximating capture basins at successive costs. We show that, by a simple change of variables, minimising a cost may be reduced to the problem of time minimisation, and hence a recursive backward procedure can be set. Two variants of the algorithm are derived, one providing an approximation from inside (the approximation is included in the actual capture basin) and one providing a outer approximation, which allows one to assess the approximation error. We use a machine learning algorithm (as a particular case, we consider Support Vector Machines) trained on points of a grid with boolean labels, and we state the conditions on the machine learning procedure that guarantee the convergence of the approximations towards the actual capture basin when the resolution of the grid decreases to 0. Moreover, we define a control procedure which uses the set of capture basin approximations to drive a point into the target. When using the inner approximation, the procedure guarantees to hit the target, and when the resolution of the grid tends to 0, the controller tends to the optimal one (minimizing the cost to hit the target). We illustrate the method on two simple examples, Zermelo and car on the hill problems.

**Keywords:** Viability theory, capture basin, optimal control, Support Vector Machines.

## 1 Introduction

We consider a dynamical system, described by the evolution of its state variable $x \in \mathbb{R}^n$:

$$x(t)' \in F(x(t)) = \{\varphi(x(t), u(t)) \mid u(t) \in \mathcal{U}\}, \tag{1}$$

where $x(t)$ is the state at time $t$, $F$ is a set-valued map and $\mathcal{U}$ the set of admissible control. We assume that there is a positive cost $\ell(x(t), u(t))$ for taking

control $u$ in state $x$ at time $t$.
We focus on the problem of defining the control function that drives the dynamical system inside a given target compact set $\mathcal{C} \subset \mathcal{K}$ without going out from a given set $\mathcal{K}$ (called the viability constraint set), and that minimises the cost functional

$$\inf_{u \in \mathcal{U}} \int_0^{+\infty} \ell(x(\tau), u(\tau)) \cdot d\tau. \tag{2}$$

This problem, often called the reachability problem, can be addressed by optimal control methods, solving Hamilton-Jacobi-Bellman (HJB) or Isaacs (HJI) equations. Several numerical techniques are available; for example, [11] propose an algorithm that computes an approximation for the backward reachable set of a system using a time dependent HJI partial differential equation, [9] builds the value function of the problem which can be then used to choose the best action at each step when the cost function represents the time.
Reachability problem can also be addressed in the viability theory framework [1]. To apply viability theory to target hitting problems when the cost to minimise is the time elapsed to reach the target, one must add an auxiliary dimension to the system, representing the time. The approach computes an approximation of the envelopes of all $t$-capture basins, the sets of points for which there exists a control function that allows the system to reach the target in a time less than $t$. [7] shows that the boundary of this set is the value function in the dynamical programming perspective. Hence, solving this extended viability problem also provides the minimal time for a state $x$ to reach the target $\mathcal{C}$ while always staying in $\mathcal{K}$ (minimal time function $\vartheta_{\mathcal{C}}^{\mathcal{K}}(x)$ [4]). This function can then be used to define controllers that drive the system into the target. The same approach can be used for cost minimisation; in that case, the extra-dimension represents the cost-to-go to the target [3].
Several numerical algorithms [13, 4] provide an over-approximation of capture basins. [2] implement an algorithm proposed by [14] that computes a discrete under-approximation of continuous minimal time functions (and thus an over-approximation of capture basins), without adding an additional dimension. [8] present an algorithm, based on interval analysis, that provides an inner and outer approximation of the capture basin. In general, capture basin and minimal time function approximation algorithms face the curse of dimensionality, which limits their use to problems of low dimension (in the state and control space).
This paper proposes a new method to solve target hitting problems, inspired by our work on viability kernel approximation [6], that minimise the cost to reach the target. The principle is to approximate iteratively the capture basins at successive costs $c$. To compute cost $c$-capture basin approximation, we use a discrete grid of points covering set $\mathcal{K}$, and label +1 the points for which there exists a control leading the point into the $c - \delta c$-capture basin approximation, and -1 otherwise. Then, we use a machine learning method to compute a continuous boundary between +1 and -1 points of the grid. We state the conditions the learning method should fulfil (they are similar to the one established to approximate viability kernels [6]) in order to prove the convergence toward the actual

capture basins.
We consider two variants of the algorithm: one provides an approximation that converges from outside, and the other from inside. Although no convergence rate is provided, the comparison of the two approximations gives an assessment of the approximation error for a given problem. Moreover, we define a controller that guarantees to reach the target when derived from the inner approximation.
We consider Support Vector Machines (SVMs [16, 15]) as a relevant machine learning technique in this context. Indeed, SVMs provide parsimonious approximations of capture basins, that allow the definition of compact controllers. Moreover, they make possible to use optimisation techniques to find the controls, hence problems with control spaces in more dimensions become tractable. We can also more easily compute the control on several cost steps, which improves the quality of the solution for a given resolution of the grid.
We illustrate our approach with some experiments on two simple examples. Finally, we draw some perspectives.

## 2 Problem definition

We consider a controlled dynamical system in discrete time (Euler approximation), described by the evolution of its state variable $x \in \mathcal{K} \subset \mathbb{R}^n$. We aim at defining the set of controls to apply to the system starting from point $x$ in order to reach the target $\mathcal{C} \subset \mathcal{K}$ :

$$\begin{cases} x(t+dt) = x(t) + \varphi\left(x(t), u(t)\right) \cdot dt, & \text{if } x(t) \notin \mathcal{C} \\ x(t+dt) = x(t), & \text{if } x(t) \in \mathcal{C} \\ u(t) \in \mathcal{U}, \end{cases} \tag{3}$$

where $\varphi$ is a continuous and derivable function of $x$ and $u$. The control $u$ must be chosen at each time step in the set $\mathcal{U}$ of admissible controls. The problem also includes a cost functional $\ell(x(t), u(t)) > 0$ that associates a cost to a trajectory.

The capture basin of the system is the set of states for which there exists at least one series of controls such that the system reaches the target in finite time, without leaving $\mathcal{K}$. Let $G\left(x, (u_1, .., u_n)\right)$ be the point reached when applying successively during $n$ time steps the controls $(u_1, .., u_n)$, starting from point $x$. Let the *minimal time function* (or hitting time function) be the function that associates to a state $x \in \mathcal{K}$ the minimum time to reach $\mathcal{C}$:

$$\vartheta_{\mathcal{C}}^{\mathcal{K}}(x) = \inf \{n | \exists (u_1, .., u_n) \in \mathcal{U}^n \text{ such that } G\left(x, (u_1, .., u_n)\right) \in \mathcal{C} \\ \text{and for } 1 \leq j \leq n, G\left(x, (u_1, .., u_j)\right) \in \mathcal{K}\}. \tag{4}$$

and the *minimal cost function* over the time period $t = \left[0, \vartheta_{\mathcal{C}}^{\mathcal{K}}(x)\right]$ be the function that associates to $x$ the minimal cost to reach $\mathcal{C}$:

$$\varpi_{\mathcal{C}}^{\mathcal{K}}(x) = \min_{u \in \mathcal{U}} \int_0^{\vartheta_{\mathcal{C}}^{\mathcal{K}}(x)} \ell(x(t), u(t)) \cdot dt. \tag{5}$$

This is the value function obtained when solving HJB equations in a dynamic programming approach. It takes values in $\mathbb{R}^+ \cup +\infty$, specifically $\vartheta_{\mathcal{C}}^{\mathcal{K}}(x) = \varpi_{\mathcal{C}}^{\mathcal{K}}(x) = 0$ if $x \in \mathcal{C}$ and $\vartheta_{\mathcal{C}}^{\mathcal{K}}(x) = \varpi_{\mathcal{C}}^{\mathcal{K}}(x) = +\infty$ if no trajectory included in $\mathcal{K}$ can reach $\mathcal{C}$. The *capture basin* of $\mathcal{C}$ viable in $\mathcal{K}$ is then defined as:

$$Capt(\mathcal{K}, \mathcal{C}) = \left\{ x \in \mathcal{K} | \vartheta_{\mathcal{C}}^{\mathcal{K}}(x) < +\infty \right\}, \tag{6}$$

and we can also define the capture basin in finite time $n$:

$$Capt(\mathcal{K}, \mathcal{C}, dt, n) = \left\{ x \in \mathcal{K} | \vartheta_{\mathcal{C}}^{\mathcal{K}}(x) \leq n \right\}. \tag{7}$$

or in finite cost:

$$Capt(\mathcal{K}, \mathcal{C}, dc, n) = \left\{ x \in \mathcal{K} | \varpi_{\mathcal{C}}^{\mathcal{K}}(x) \leq n \right\}. \tag{8}$$

with $dc$ the cost step.

To solve a target hitting problem in the viability perspective, one must consider the following extended dynamical system $(x(t), y(t))$ when $x(t) \notin \mathcal{C}$:

$$\begin{cases} x(t + dt) = x(t) + \varphi\left(x(t), u(t)\right) \cdot dt \\ y(t + dt) = y(t) - \ell(x(t), u(t)) \cdot dt. \end{cases} \tag{9}$$

and $(x(t + dt) = x(t), y(t + dt) = y(t))$ when $x(t) \in \mathcal{C}$. [4] prove that approximating minimal time function comes down to a viability kernel approximation problem of this extended dynamical problem, with $\ell(x(t), u(t) = 1$ and [3] extend the results to any cost function $\ell(x(t), u(t)) > 0$. In a viability problem, one must find the rule of controls for keeping a system indefinitely within a constraint set. [2, 14] give examples of such an application of viability approach to solve a target hitting problem.

[6] proposed an algorithm, based on [13], that uses a machine learning procedure to approximate viability kernels. The main advantage of this algorithm is that it provides continuous approximations that enable to find the controls with standard optimization techniques, and then to tackle problems with control in large dimensional space. The aim of this paper is to adapt [6] to compute directly an approximation of the capture basin limits, without adding the auxiliary dimension, and then to use these approximations to define the sequence of controls.

## 3 Machine learning approximation of value function contours and optimal control

For simplicity, we denote $Capt(\mathcal{K}, \mathcal{C}, dc, n \cdot dc) = Capt^n$. In all the following, continuous sets are denoted by rounded letters and discrete sets in capital letters.

Let's denote $dc$ the cost variation for one state and control at time step $dt$:

$$dc = \ell(x(t), u(t)) \cdot dt \tag{10}$$

and we consider function $G$:

$$G(x,u) = \begin{cases} x + \varphi^*(x,u) \cdot dc & \text{if } x \notin \mathcal{C} \\ x & \text{if } x \in \mathcal{C} \end{cases} \tag{11}$$

where

$$\varphi^*(x,u) = \frac{\varphi(x,u)}{\ell(x,u)}. \tag{12}$$

By a change of variables, minimising a cost may be reduced to the problem of time minimisation ($\ell(x,u) = 1$), except that we consider here the cost step $dc$ instead of time step $dt$.
We suppose that $G$ is $\mu$-Lipschitz with respect to $x$:

$$\forall (x,y) \in \mathcal{K}^2, \forall u \in \mathcal{U}, |G(x,u) - G(y,u)| < \mu |x - y|. \tag{13}$$

We define a grid $K_h$ as a discrete subset of $\mathcal{K}$, such that:

$$\forall x \in \mathcal{K}, \exists x_h \in K_h \text{ such that } |x - x_h| \leq h. \tag{14}$$

Moreover, we define an algebraic distance $d_a(x, \partial\mathcal{E})$ of a point $x$ to the boundary $\partial\mathcal{E}$ of a continuous closed set $\mathcal{E}$, as the distance to the boundary when $x$ is located inside $\mathcal{E}$, and this distance negated when $x$ is located outside $\mathcal{E}$:

$$\text{if } x \in \mathcal{E}, d_a(x, \partial\mathcal{E}) = d(x, \partial\mathcal{E}), \tag{15}$$

$$\text{if } x \notin \mathcal{E}, d_a(x, \partial\mathcal{E}) = -d(x, \partial\mathcal{E}). \tag{16}$$

### 3.1 Capture basin approximation algorithms

In this section, we describe two variants of an algorithm that provides an approximation $\mathcal{C}_h^n$ of the capture basin at cost $n \cdot dc$, one variant approximates the capture basins from outside and the other one from inside. At each step $n$ of the algorithm, we first build a discrete approximation $C_h^n \subset K_h$ of the capture basin $Capt^n$, and then we use a learning procedure $L$ (for instance Support Vector Machines, as shown below) to generalise this discrete set into a continuous one $\mathcal{C}_h^n$:

$$\mathcal{C}_h^n = L(C_h^n) \tag{17}$$

To simplify the writing, we first define:

$$\overline{C_h^n} = \{x_h \in K_h \text{ s.t. } x_h \notin C_h^n\}, \tag{18}$$

$$\overline{\mathcal{C}_h^n} = \{x \in \mathcal{K} \text{ s.t. } x \notin \mathcal{C}_h^n\}. \tag{19}$$

The two variants differ on the conditions for defining the discrete set $C_h^{n+1}$ from $C_h^n$, and on the conditions the learning procedure $L$ must fulfil. For both

variant, we construct an increasing sequence of approximations at cost $n \cdot dc$, by adding the points of the grid for which there exists at least one control that drives the system not too far away (in an algebraic sense – negative distance in the outer case and positive distance in the inner one) from the boundary of the previous approximation. They also both require conditions on the learning procedure, in order to guarantee the convergence toward the actual capture basin when the step of the grid $h$ decreases to 0. In the inner approximation case, the condition is stricter on set $\mathcal{C}_h^n$ and more relaxed on set $\overline{\mathcal{C}_h^n}$, while the outer case requires converse conditions. We now describe in more details both variants and conditions.

**Outer approximation** The algorithm recursively constructs discrete sets $C_h^{n-1} \subseteq C_h^n \subseteq C_h$ and their continuous generalisation $\mathcal{C}_h^n$ as follows:

**Algorithm 1** Outer capture basin approximation algorithm

$n \leftarrow 0$
$C_h^0 \leftarrow \mathcal{C} \cap K_h$
$\mathcal{C}_h^0 \leftarrow \mathcal{C}$
**repeat**
  $n \leftarrow n+1$
  $C_h^n \leftarrow C_h^{n-1} \bigcup \left\{ x_h \in \overline{C_h^{n-1}} \text{ such that } \exists u \in \mathcal{U}, d_a(G(x_h,u), \partial \mathcal{C}_h^{n-1}) \geq -\mu h \right\}$
  $\mathcal{C}_h^n \leftarrow L(C_h^n)$
**until** $C_h^n = C_h^{n-1}$
**return** $\left\{\mathcal{C}_h^i\right\}_{0 \leq i \leq n}$

**Proposition 1.** *If the learning procedure $L$ respects the following conditions:*

$$\forall x \in \overline{\mathcal{C}_h^n}, \exists x_h \in \overline{C_h^n} \text{ such that } |x - x_h| \leq h \tag{20}$$

$$\exists \lambda \geq 1 \text{ such that } \forall h, \forall x \in \mathcal{C}_h^n, \exists x_h \in C_h^n \text{ such that } |x - x_h| \leq \lambda h \tag{21}$$

*then the convergence of the approximation from outside defined on algorithm 1 is guaranteed:*

$$\forall n, Capt^n \subset \mathcal{C}_h^n, \tag{22}$$

$$\mathcal{C}_h^n \to Capt^n \text{ when } h \to 0. \tag{23}$$

*Proof.* The proof involves two parts.
*Part I.* First, let us prove by induction that $\forall h > 0, Capt^n \subset \mathcal{C}_h^n$.
By definition, $Capt^0 = \mathcal{C} = \mathcal{C}_h^0$. Suppose that at step $n$, $Capt^n \subset \mathcal{C}_h^n$. Consider $x \in Capt^{n+1}$. Let us recall that $G(x,u) = x + \varphi(x,u)^* \cdot dc$ when $x \notin \mathcal{C}$.
Defining $B_h(x,d)$ the set of points of $K_h$ such that $|x - x_h| \leq d$, we can easily show that condition (21) can be rewritten as:

$$B_h(x,h) \subset C_h^n \Rightarrow x \in \mathcal{C}_h^n. \tag{24}$$

By definition, we know that there exists $u \in \mathcal{U}$ such that $G(x,u) \in Capt^n$, which implies that for all $x_h \in B_h(x,h)$, $d\left(G(x_h,u), Capt^n\right) \leq \mu h$, because $G$ is $\mu$-Lipschitz.
Moreover, for all $x_h \in B_h(x,h)$, $d(G(x_h,u), \mathcal{C}_h^n) \leq \mu h$, because, by hypothesis, $Capt^n \subset \mathcal{C}_h^n$. Thus $x_h \in C_h^{n+1}$. Therefore, $x \in \mathcal{C}_h^{n+1}$ (because of condition (24)). We can thus conclude $Capt^{n+1} \subset \mathcal{C}_h^{n+1}$.

*Part II.* Now, we prove by induction that for any $n$, $\mathcal{C}_h^n \to Capt^n$ when $h \to 0$.
Suppose now that for a given value $n$, $\mathcal{C}_h^n \to Capt^n$ when $h \to 0$.
Because $Capt^n \subset \mathcal{C}_h^n$, we have:

$$\forall x \in \mathcal{K} \mid x \notin Capt^n, \exists h > 0 \mid x \notin \mathcal{C}_h^n. \tag{25}$$

Now, consider $x \in \mathcal{K}$ such that $x \notin Capt^{n+1}$.
This implies that for all $u \in \mathcal{U}$ such that $d\left(G(x,u), Capt^n\right) > 0$. One can choose $h' > 0$ and $h$ such that for all $u \in \mathcal{U}$, $d\left(G(x,u), Capt^n\right) > h' + \mu\lambda h$.
Condition (20) can be rewritten as:

$$B_h(x, \lambda h) \subset \overline{C_h^n} \Rightarrow x \in \overline{\mathcal{C}_h^n}. \tag{26}$$

In this case, for all $x_h \in B_h(x, \lambda h)$, all $u \in \mathcal{U}$, $d\left(G(x_h,u), Capt^n\right) > h'$, because $G$ is $\mu$-Lipschitz.
Since $\mathcal{C}_h^n \to Capt^n$ when $h \to 0$, there exists $h$, such that, for all $x_h \in B_h(x, \lambda h)$, and all $u \in \mathcal{U}$, $G(x_h,u) \in \overline{\mathcal{C}_h^n}$, hence $x_h \in \overline{\mathcal{C}_h^n}$. Hence, there exists $h$ such that $x \notin \mathcal{C}_h^n$ (because of condition 24).
Therefore $\mathcal{C}_h^{n+1} \to Capt^{n+1}$ when $h \to 0$.

*Conclusion.* $Capt^n \subset \mathcal{C}_h^n$ and $\mathcal{C}_h^n \to Capt^n$ then $\mathcal{C}_h^n$ is an outer approximation of the capture basin at cost $n \cdot dc$, which tends to the actual capture basin when the resolution of the grid $h$ tends to 0. □

**Inner approximation** We consider the following algorithm:

**Algorithm 2** Inner capture basin approximation algorithm

$n \leftarrow 0$
$C_h^0 \leftarrow \mathcal{C} \cap K_h$
$\mathcal{C}_h^0 \leftarrow \mathcal{C}$
**repeat**
  $n \leftarrow n+1$
  $C_h^n = C_h^{n-1} \bigcup \left\{ x_h \in \overline{C_h^{n-1}} \text{ such that } \exists u \in \mathcal{U}, d_a(G(x_h,u), \partial\mathcal{C}_h^n) \geq \mu h \right\}$
  $\mathcal{C}_h^n \leftarrow L(C_h^n)$
**until** $C_h^n = C_h^{n-1}$
**return** $\left\{\mathcal{C}_h^i\right\}_{0 \leq i \leq n}$

**Proposition 2.** *If the learning procedure L respects the following conditions:*

$$\forall x \in \mathcal{C}_h^n, \exists x_h \in C_h^n \text{ such that } |x - x_h| \leq h \tag{27}$$

$$\exists \lambda \geq 1 \text{ such that } \forall h, \forall x \in \overline{\mathcal{C}_h^n}, \exists x_h \in \overline{C_h^n} \text{ such that } |x - x_h| \leq \lambda h \tag{28}$$

*and that the dynamics are such that:*

$$\forall x \in \mathcal{K} \text{ with } d_a\left(x, \partial Capt^n\right) > 0, \exists u \in \mathcal{U} \text{ such that } d_a\left(G(x,u), \partial Capt^{n-1}\right) > 0 \tag{29}$$

*then the convergence of the approximation from inside defined on algorithm 2 is guaranteed:*

$$\forall n, \mathcal{C}_h^n \subset Capt^n, \tag{30}$$

$$\mathcal{C}_h^n \to Capt^n \text{ when } h \to 0. \tag{31}$$

*Proof.* Convergence proof of the algorithm from inside requires an additional condition on the dynamics (eq. (29)): a point $x$ of the interior of capture basin at cost $n \cdot dc$, should be such that there exists $y \in G(x)$ belonging to the interior of capture basin at cost $(n-1) \cdot dc$ (and not on $\partial Capt^{n-1}$).
*Part I.* We begin to show by induction that $\mathcal{C}_h^n \subset Capt^n$.
Suppose that $\mathcal{C}_h^n \subset Capt^n$ and consider $x \in \mathcal{C}_h^{n+1}$.
Because of condition (27):

$$\exists x_h \in C_h^{n+1} \text{ such that } |x - x_h| < h.$$

By definition of $C_h^{n+1}$:

$$\exists u \in U \text{ such that } d_a\left(G(x_h, u), \mathcal{C}_h^n\right) > \mu h.$$

By hypothesis of induction $\mathcal{C}_h^n \subset Capt^n$, hence : $d_a\left(G(x_h, u), Capt^n\right) > \mu h$. By hypothesis on $G$, $|G(x_h, u) - G(x, u)| < \mu |x_h - x|$, hence $d_a\left(G(x, u), Capt^n\right) > 0$. Therefore $x \in Capt^{n+1}$. Thus $\mathcal{C}_h^{n+1} \subset Capt^{n+1}$.

*Part II.* We prove by induction that, when $h \to 0$, $\mathcal{C}_h^n \to Capt^n$ .
Suppose that $\mathcal{C}_h^n \to Capt^n$ when $h \to 0$.
Because $\mathcal{C}_h^n \subset Capt^n$, we have:

$$\forall x \in Capt^n \mid d_a(x, \partial Capt^n) > 0, \exists h > 0 \mid x \in \mathcal{C}_h^n.$$

We use the rewriting of condition (28):

$$B_h(x, \lambda h) \cap C_h^n \Rightarrow x \in \mathcal{C}_h^n. \tag{32}$$

Consider $x \in Capt^{n+1}$ such that $d_a(x, \partial Capt^{n+1}) > 0$ . One can choose $h$ such that $d_a(x, \partial Capt^{n+1}) > (\mu + \lambda)h$. With such a choice, for each $x_h \in B_h(x, \lambda h)$, $d_a(x_h, \partial Capt^{n+1}) > \mu h$, hence, there exists $u \in \mathcal{U}$ such that $d_a\left(G(x_h, u), Capt^n\right)) > 0$ (because $G$ is $\mu$-Lipschitz).
By induction hypothesis, there exists $h$ such that $d_a\left(G(x_h, u), \mathcal{C}_h^n\right) > \mu h$, hence

$x_h \in C_h^{n+1}$. Taking the smallest value of $h$ this is true for all $x_h \in B_h(x, \lambda h)$. Therefore $x \in \mathcal{C}_h^{n+1}$ (because of condition (32)).
Therefore $\mathcal{C}_h^{n+1} \to Capt^{n+1}$ when $h \to 0$.

*Conclusion.* $\mathcal{C}_h^n \subset Capt^n$ and $\mathcal{C}_h^n \to Capt^n$ then $\mathcal{C}_h^n$ is an inner approximation of the capture basin in finite cost $n \cdot dc$, which tends to the actual capture basin when the resolution of the grid tends to 0. □

### 3.2 Optimal Control

The aim of the optimal controller is to choose a control function that reaches the target in minimal cost, without breaking the viability constraints. The idea is to choose the controls which drive the system to cross $\mathcal{C}_h^n$ boundaries in a descending order.

**Algorithm 3** Optimal controller

**Require:** $x(0) \in \mathcal{K}$ and $\notin \mathcal{C}$
**Require:** $n$ such that $x(0) \in \mathcal{C}_h^n$ and $\notin \mathcal{C}_h^{n-1}$.
  **for** $i = 1$ **to** $n$ **do**
    compute $u(i)^*$ such that $G(x(i-1), u(i)^*) \in \mathcal{C}_h^{n-i}$
    $x(i) = G(x(i-1), u(i)^*)$
  **end for**
  **return** $\{u^*(i)\}_{1 \leq i \leq n}$

**Proposition 3.** *The procedure described in algorithm 3 converges towards the control policy minimizing the hitting cost, when h and dt tend to 0.*

*Proof.* By construction, if $x(i) \in \mathcal{C}_h^{n-i}$ and $x(i) \notin \mathcal{C}_h^{n-i-1}$, there exists a control value $u^*(i+1)$ such that $x(i+1) = G(x(i), u^*(i+1)) \in \mathcal{C}_h^{n-i-1}$ (see proof of the convergence of the inner algorithm, part I.). Therefore, the procedure leads to the target in $n+1$ cost steps, i.e. with a cost $(n+1) \cdot dc$. Moreover, by definition, the optimum cost for reaching the target from a point $x$ located on the boundary of capture basin $Capt^n$ is $n \cdot dc$. Hence, the optimum cost for reaching the target from point $x$ such that $x \in Capt^{n+1}$ and $x \notin Capt^n$, with the dynamics defined by $\varphi$, is between $n \cdot dc$ and $(n+1) \cdot dc$. Then, the fact that $\mathcal{C}_h^n$ converges to $Capt^n$ when $h$ tends to 0, ensures that the number of steps needed by the procedure applied to this point $x$ will tend to $(n+1) \cdot dc$. When $dc$ tends to 0, the difference with the optimum cost to reach the target, which is smaller than $dc$, tends to 0. □

# 4 Experiments

## 4.1 SVM as a learning procedure

We use Support Vector Machines [16, 15] as the learning procedure $L$ to define capture basin approximations $\mathcal{C}_h^n = L(C_h^n)$. At each iteration, we construct a learning set: let $S_h^n$ be a set of couples $(x_h, y_h)$, where $x_h \in K_h$ and $y_h = +1$ if $x_h \in C_h^n$ and $-1$ otherwise. Running a SVM on learning sets $S_h^n$ provides a separating function $f_h^n$ between points of different labels and hence, allows us to define a continuous approximation $\mathcal{C}_h^n$ of $C_h^n$ as follows:

$$\mathcal{C}_h^n = \{x \in K \text{ such that } f_h^n(x) \geq 0\}. \tag{33}$$

Points $x$ of the boundary $\partial\mathcal{C}_h^n$ are those such that $f_h^n(x) = 0$. The fulfilment of the conditions guaranteeing convergence is discussed in [6] and the same arguments hold in both variants of the algorithm.

In the following examples, we use libSVM [5] to train the SVMs. As we did in [6], we use the SVM function as a proxy for the distance to the boundary, in order to simplify the computations.

## 4.2 Zermelo Problem

The state $(x(t), y(t))$ of the system represents the position of a boat in a river. There are two controls: the thrust $u$ and the direction $\theta$ of the boat. The system in discrete time defined by a time interval $dt$ can be written as follows:

$$\begin{cases} x(t+dt) = x(t) + (1 - 0.1y(t)^2 + u\cos\theta) \cdot dt \\ y(t+dt) = y(t) + (u\sin\theta) \cdot dt, \end{cases} \tag{34}$$

where $u \in [0; 0.44]$ and $\theta \in [0; 2\pi]$. The boat must remain in a given interval $\mathcal{K} = [-6; 2] \times [-2; 2]$, and reach a round island $\mathcal{C} = B(0; 0.44)$ in minimal time. We suppose that the boat must reach the island before time $T = 7$.
For this simple system, it is possible to derive analytically the capture basin, hence we can compare the approximations given by the two algorithms with the actual capture basin. Figure 1 compares some results obtained with the outer and inner approximation. In any cases, the quality of the approximation can be assessed by comparing both approximations: by construction, the contour of the actual capture basin is surrounded by inner and outer approximations. A example of a optimal trajectory defined with the optimal controller is also presented: with the inner approximation, the trajectory will enable the boat to reach the target, while it is not guaranteed in the outer case.

## 4.3 Mountain car

****** DO STHG LIKE ADDING A COST FOR THE CONSUMPTION OF THE CAR ******

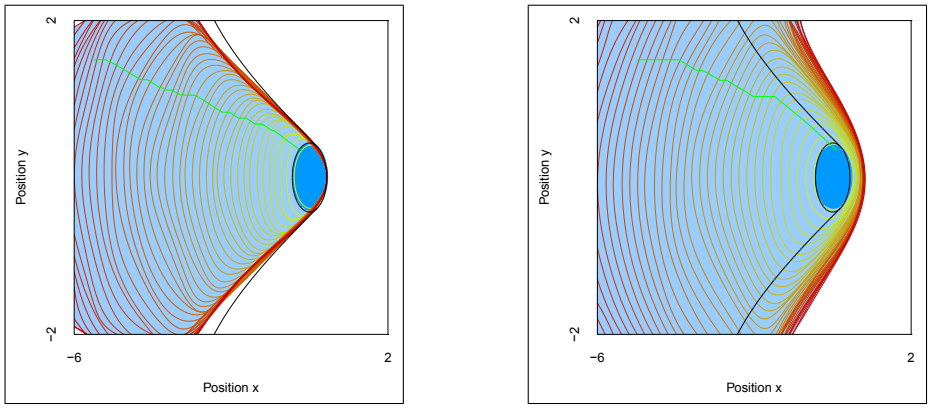


**Fig. 1.** Approximation from inside (left) and from outside (right) for Zermelo problem. The horizontal axis represents the position $x$ and the vertical one position $y$. $\mathcal{K}$ is the rectangle. The capture basin is represented in blue. The black thick line limits the boundary of the actual capture basin. The level lines represent approximation of the contours of the capture basins for successive time steps. The grid contains 41 points by dimension. The optimisation is made on 4 time steps, with $dt = 0.05$. Each figure presents an example of trajectory (in green) using the SVM optimal controller.

We consider the well-known car on the hill problem. The state is two-dimensional (position and velocity) and the system can be controlled with a continuous one-dimensional control (thrust). For a description of the dynamics and the state space constraints, one can refers to [12]. The aim of the car on the hill system is to keep the car inside a given set of constraints, and to reach a target (the top of the hill) as fast as possible. The interesting characteristics of the problem is that the car has limited power, the acceleration can not overcome the gravitational force and hence the car may have to first go away from the solution before reverse up the hill. Figure 2 shows the approximation of the contours of the value function using outer and inner variants of the algorithm, with an example of optimal trajectories.

## 5 DISCUSSION

We proposed an algorithm that approximates capture basins and contours of value function, using a classification procedure, in two variants (inner and outer). The inner approximation can be used to define a optimal controller that guarantees to find a series of controls that allows the system to reach the target. SVMs appear as a particularly relevant classification procedure for this approach, because they provide parsimonious representations of the capture basins and enable to use optimization techniques to compute the controls. This latter point is particularly important to deal with high dimensional control space. The parsimonious property may allow to define compact and fast controller, even for

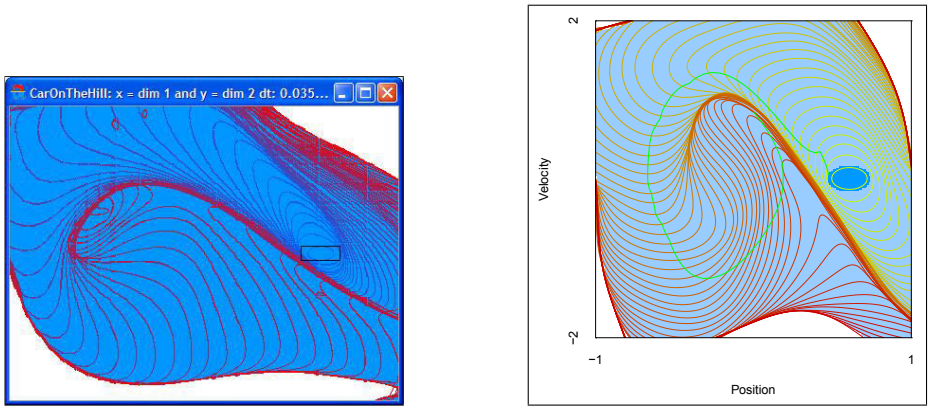


**Fig. 2.** Inner (left) and outer (right) approximation for the car on the hill problem. The grid contains 51 points by dimension. The optimisation is made on 2 time steps. An example of an optimal trajectory is depicted in green.

high dimensional state space. However, although we generally manage to find parameters in which the result respect the conditions of convergence, this is not guaranteed. Therefore, considering other learning algorithms that would be even more appropriate seems a relevant research direction.
For now, the method proposed here can only be used to solve problems with deterministic dynamics. A second direction of research is to investigate the behaviour of the optimal controller when there is some uncertainty on the state or the control.

## References

1. Aubin, J.P.: Viability theory. Birkhäuser (1991)
2. Bayen, A., E., C., & Tomlin, C.: Guaranteed overapproximations of unsafe sets for continuous and hybrid systems: Solving the hamilton-jacobi equation using viability techniques. Lecture Notes In Computer Science 2289, 90–104 (2002)
3. Cardaliaguet, P., Quincampoix, M., Saint-Pierre, P.: Optimal times for constrained nonlinear control problems without local optimality. Applied Mathematics & Optimization 36, 21–42 (1997)
4. Cardaliaguet, P., Quincampoix, M., Saint-Pierre, P.: Set-Valued Numerical Analysis for Optimal control and Differential Games. Annals of the International Society of Dynamic Games (1998)
5. Chang, C.C., Lin, C.J.: Libsvm: a library for support vector machines (2001)
6. Deffuant, G., Chapel, L., Martin, S.: Approximating viability kernels with support vector machines. IEEE transactions on automatic control 52(5), 933–937 (2007)
7. Frankowska, H.: Optimal trajectories associated with a solution of the contingent hamilton-jacobi equation. Applied Mathematics and Optimization 19(1), 291–311 (1989)

8. Lhommeau, M., Jaulin, L., Hardouin, L.: Inner and outer approximation of capture basin using interval analysis. In: 4th International Conference on Informatics in Control, Automation and Robotics (ICINCO 2007) (2007)
9. Lygeros, J.: On reachability and minimum cost optimal control. Automatica 40, 917–927 (2004)
10. Martin, S.: The cost of restoration as a way of defining resilience: a viability approach applied to a model of lake eutrophication. Ecology and Society 9(2) (2004)
11. Mitchell, I., Bayen, A., Tomlin, C.: A time-dependent Hamilton-Jacobi formulation for reachable sets for continuous dynamic games. IEEE Transactions on Automatic Control 50(7), 947–957 (2005)
12. Moore, A., Atkeson, C.: The parti-game algorithm for variable resolution reinforcement learning in multidimensional state-spaces. Machine Learning 21, 199–233 (1995)
13. Saint-Pierre, P.: Approximation of the viability kernel. Applied Mathematics & Optimization 29(2), 187–209 (1994)
14. Saint-Pierre, P.: Approche ensembliste des systèmes dynamiques, regards qualitatifs et quantitatifs. Matapli, Société de Mathématiques appliquées et Industrielles p. 66 (2001)
15. Scholkopf, B., Smola, A.: Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond. MIT Press, Cambridge, MA, USA (2002)
16. Vapnik, V.: The nature of statistical learning theory. Springer Verlag (1995)