European Conference on Complex Systems. Warwick 2009.

Geometric Analysis of a Capture Basin

Application to cheese ripening process

Salma Mesmoudi · Isabelle Alvarez · Sophie Martin · Mariette Sicard · Pierre-Henri Wuillemin

Abstract This paper addresses the issue of analysing the results given by viability sets, which represent all the viable states of a dynamical system. Viability sets and capture basin are a relatively new method to study complex dynamical systems, focusing on the preservation of some properties of the system (constraints in the state space) rather than on the possible stable states and equilibria. The viability set delimits the area of the state space where there is always a list of controls that allows the system to verify indefinitely the constraints. A viability set represents a huge amount of information about the system it studies. We propose here a geometric study of viability sets. This study is applied to the viability tube (the sequence of viability set over time) of a food process.

Keywords Geometric Analysis · Viability · Food processing

1 Introduction

Sets play an increasing role in complex system analysis. Actually, considering a controlled dynamical system facing viability constraints, the viability set [1] gathers all states from which there exists at least one control function that allows the contraints to be satisfied undefinitely or for a given time horizon. Thus, computing such a viability set enables to answer common questions such that "what are the system configurations that allow its persistence"

S. Mesmoudi
LIP6, 104 av. Kennedy F-75016 Paris
E-mail: salma.mesmoudi@decision.lip6.fr

I. Alvarez
LIP6, 104 av. Kennedy F-75016 Paris France
Cemagref LISC, F-63172 Aubiere France
E-mail: isabelle.alvarez@lip6.fr

S. Martin
Cemagref LISC, F-63172 Aubiere France
E-mail: sophie.martin@cemagref.fr

M. Sicard
UMR782 GMPA, AgroParisTech, INRA, F-78850 Thiverval-Grignon, France

P.H. Wuillemin
LIP6, 104 av. Kennedy F-75016 Paris France

and ”which controls have to be applied to remain in the viability set”. When moreover objectives are pursued, the capture basin set gathers all states from which there exists at least one control function that allows reaching these objectives without violating the constraints. Thus, computing such a capture basin set enables to answer common questions as far as complex controlled systems are concerned such that ”what are the system configurations that allow its persistence until its goal achievement” and ”which controls have to be applied to really reach these goals”.

For instance, thanks to viability or capture basin set computation, Aubin et al.[2] studied the conditions of French pension system durability ; Bonneuil and Müllers (1997) [3] the conditions of several species coexistence ; Béné et al. (2001) [4] the configurations to avoid to prevent irreversible non-renewable resources overexploitation ; Mullon et al. (2004) [5] the fishing quotas to lay down to prevent some species extinction. In complex technical process field, viability or capture basin sets are used to study the control of highway traffic [6] or air traffic [7] and also for the food process of cheese ripening [8].

Viability and capture basin sets are subsets of the state space. They are computed thanks to viability algorithms (Saint-Pierre 1994 [9], Deffuant et al. 2007 [10]). In Saint-Pierre's algorithm, these sets are approximated by points on a grid, in Deffuant et al. 's one, their boundary is approximated by a support vector machine. The sensibility towards some dynamics parameters of a particular viability set was studied by Bonneuil and Saint-Pierre (2000)[11].
Beyond sensibility analysis, the shape study of viability and capture basin sets could provide useful information about the robustness to disturbance or uncertainties of a particular state or configuration according to the possibility of maintaining viability or reaching the pursued goal. Such a geometric study could determine if a state is close to the set boundary, and also the directions at risk. At a trajectory level, it would allow to define several robustness indicators, depending on the risk perception of the operator. At the viability set level, it could highlight areas where the process is more sensitive to perturbation, and consequently, where it should be monitored carefully.
In this paper, we show that such a geometric analysis of viability or capture basin sets is based on the computation of both the distance to the set boundary and the projection onto this boundary. Optimal algorithms for the Euclidean distance transform (EDT) in arbitrary dimension have been developed for morphological mathematics and image analysis purpose. We adapt one of them [12] to propose methods and algorithms to carry out the geometric analysis of viability or capture basin sets. In particular, we can evaluate the resilience of a controlled dynamical system, that is its capacity to maintain given properties ; Martin (2004) [13] proposed to evaluate resilience of some system properties as the inverse cost to return to the viability set associated with these properties, here, we base the resilience definition on the distance to the viability set boundary along the possible trajectories. We demonstrate the usefulness of our methods and algorithms by applying them to the viability sets provided by cheese ripening analysis mentioned above.

This paper is organized as follow: Section 2 presents the geometric method used to analyse these sets : the local and global indicators, especially the definition of the resilience of a trajectory based on the distance to the viability set boundary. It also presents the algorithm to compute this distance and the projection onto the boundary. Section 3 develops the coding and study of this algorithm, in order to show its strength and limits. Section 4 shows the result of the geometric study for the cheese ripening process after a brief description of the

problem of management of a complex process of cheese ripening, and the viability tube that was built to represent the information available on the process.

2 Geometric Analysis of a viability set

2.1 Geometric indicators

Each state in the viability set is a viable state, which means that there exists at least one sequence of controls that enables the system to stay in the constraint set indefinitively. However, the situation of the viable states can be very different from one another.

For example, Figure 1 shows the viability set for the eutrophication lake problem. Clear-water (oligotrophic) lake can become turbid (or in a eutrophic state) with an excess of phosphorus. A simple model (see [13] for more details) can be used to determine whether a lake can remain in an oligotrophic state. L is the amount of phosphorus inputs and P the phosphorus concentration in the lake. Agriculture requires a minimum value for L. The oligotrophic state requires a maximum value for P. Regulation laws are constraints on $\frac{dL}{dt}$. The viability kernel is the subset of the (L,P) plane that gathers all states (L,P) such that there exists at least one regulation law that allows the oligotrophic state to be maintained. In this example, it is easy to see that, although points A and B are both viable, the state of the lake at point B is more critical than at point A, because it is closer to the boundary of the viability set.

In the same line of reasoning, trajectories inside the viability set can also be very different from one another. Figure 1 shows two trajectories stemming from point A. These trajectories stay for ever in the viability set. But trajectory $u1$, although always viable, comes very close to the boundary at some points. On the contrary, trajectory $u2$ stays far from the boundary.

Geometric indicators simply focus on these particular aspects.

2.1.1 Local indicators

A viability set can be seen as a two-classes classification system: Viable states are inside the viability set. The decision boundary (the boundary of the inverse image of the class "viable" in the input space) is the boundary of the viability set. Therefore works on explanation of the result in classification systems can be usefully adapted to viability set. In particular, the distance to the decision boundary and other geometric concepts can be used to give relevant information about the situation of a particular state or case (see [14])

Distance and projection. The distance to the viability set boundary Γ is a usefull indicator of the robustness of a state to pertubation or uncertainties on the state variables. If a perturbation is applied to the system in state x, as long as its size is smaller than $d(x,\Gamma)$, the new state of the system will still be inside the viability set. Consequently, the distance to the boundary can be seen as a measure of the robustness of a state to perturbation, error of measurement or uncertainties.

We note $p(x)$ a point of the boundary for which the distance $d(x,\Gamma)$ is reached (for the Euclidean distance it is the orthogonal projection). It gives the direction and size of the smallest perturbation (the sensitive move) to apply at state x to move the system outside the viability set.

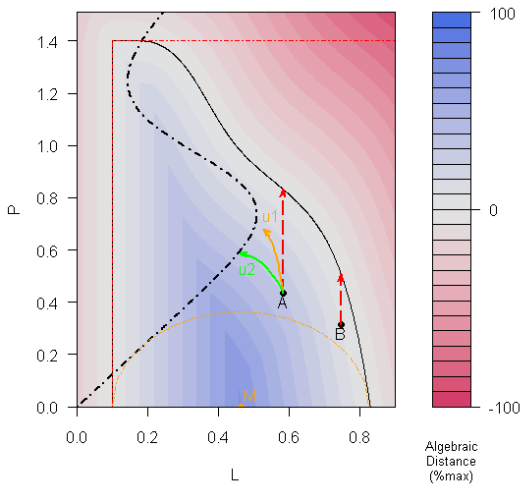

Fig. 1: Viability set of the lake eutrophication problem in the (L,P) (phosphorus input, total phosphorus) plane ([13]), and distance contour lines. The boundary of the viability set is delimited by a black plain line. The Maximal maximal ball is centered on point M. The constraints set is delimited with red dashed lines. A and B are example of viables points. $u1$ and $u2$ are possible trajectories stemming from A.

Skeleton. The skeleton S of a viability set V is the set of the centers of all maximal open balls in A (see [15] for a study of its topological properties). Maximal open balls are topological balls which are included in the viability set V and are maximal (they are not included in any other ball inside the viability set). Therefore the states of the viability set that belong to the skeleton are locally the farthest states from the boundary of the viability set. They could be useful to propose directions of move (with respect to the dynamics) in order to increase the robustness.

As viable states with at least two projection points constitute a dense subset of the skeleton, it is interesting to compute this subset of the skeleton.

2.1.2 Resilience of a trajectory

In ecology, the resilience measures the capacity of a dynamical system to maintain given properties. Here the distance to the viability set boundary gives a geometric definition of resilience inside the viability set. Since the distance map gives the resilience at each point, it is possible to use this information to define resilience indicators at the level of a trajectory. Several definitions can be proposed, depending on the risk perception of the manager (it is well known in decision psychology studies, like in [16], that the risk of adverse decision can be considered in many different ways).

For example, the most risk-adverse indicator is the min over the trajectory.

Definition 1 Resilience of a trajectory.
Let $u : t \longmapsto u(t)$ be a trajectory in the viability set V. We note Γ_V its boundary. The resilience value of u is $r(u) = \min_t\{d(u(t),\Gamma_V)\}$.

Other definitions could be considered, for example, with the mean value of the distance instead of the minimum.

2.1.3 Global quality indicator of a viability set

Some geometric indicators give usefull information about the feasibility of controlling the system to verify the constraints indefinitely.

Relative volume. The size of the viability set compared with the size of the constraint set is a first basic indicator. Since the viability set V is a subset of the constraint set K, it is the fraction of the hypervolumes of both sets: $\frac{vol(V)}{vol(K)}$.

But this relative volume can gives misleading information about the structure of the viability set, even when the viability set is a simply connected set (which means roughly that it has no holes). The same volume can enclose rather different shapes, for instance in a dimension d space, a hypercube with side $2r$ and a hyperrectangle with one side size $2\alpha r$ and all other side $\frac{2r}{(\alpha)^{\frac{1}{d-1}}}$ with $\alpha >> 1$. In the first case, the distance to the boundary of the hypercube reaches its maximum at the center with a value of r. In the second case, the points of the hyperrectangle cannot be farther from the boundary than $\frac{r}{(\alpha)^{\frac{1}{d-1}}}$.

Maximal maximal ball. The maximal maximal ball is the maximal ball with the largest radius among maximal balls. Its center is the farthest point of the viability set from the decision boundary. The largest the radius of the maximal maximal ball is, the more robust is the system to perturbation: among all the closed sets with the same inner volume, the maximum value of the distance to the boundary is reached for the closed ball, as it is illustrated on Figure 2.

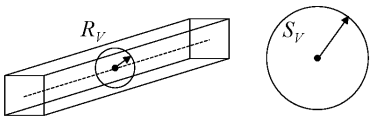

Fig. 2: Maximum (Euclidean) distance to the boundary for a hyperrectangle R_V and a hypersphere S_V of same volume V.

A relatively large maximal maximal ball, as in Figure 1 suggests that it should be easy to find control sequences that guarantee that the viability constraints are verified over time.

2.2 Projection algorithm

2.2.1 Distance transform map algorithm

When the decision boundary Γ is described by an analytical formula, or by a set of constraints, it is possible to compute the exact value of $d(x,\Gamma)$ for all points of the input space, with efficient algorithms. It is the case for axis-parallel decision trees, where the distance is computed in $O(d.n)$, with n the number of leaves and d the dimension of the space (see [14]).

When the decision boundary is described with an implicit formula or in extension, as it is the case with viability sets, we propose to approximate the distance to the decision boundary in a discrete space. A unit hypercube in the state space E is discretized onto a N^d points d-dimensional grid G in $\mathbb{Z}^d$. The discretized boundary $\hat{\Gamma} \subset G$ is defined as follows: let $\hat{x} \in G$ and let x be its corresponding point in E,

$$\hat{x} \in \hat{\Gamma} \text{ if and only if } d(x,\Gamma) \leq \frac{\sqrt{d}}{N-1} \tag{1}$$

($\sqrt{d}$ is the diagonal length of the unit hypercube and $(N-1)$ the rescaling coefficient).

Basically, the distance is computed for the points of the grid. This distance is assigned to the nearest points, so the distance map is piecewise constant. Better approximation could be used, taking into account several points of the grid to assign a value at some point.

Optimal algorithms for the Euclidean distance transform (EDT) in arbitrary dimension have been developed for morphological mathematics and image analysis purpose (see [12] for references). In particular, algorithm from [17] computes the exact EDT in linear time with the number of points where the distance is computed, which is optimal.

Of course these algorithms are exponential with the dimension, but as it is shown in next section, it is possible to use them easily up to dimension 7.

Algorithm 1 ***EuclideanDistanceToSet***$(G,\hat{f})$
Data*: A grid G of N^d points in $\mathbb{Z}^d$, $\hat{f}$ a mapping from G to $\{0,1\}$. $\hat{\Gamma} = \{x/\ \hat{f}(x) = 0\}$*
Result*: The Euclidean distance g to $\hat{\Gamma}$ for each point of G.*
1. ***Step 1***
2. for $(x_2,...,x_d) \in [0,N-1]^{d-1}$ *do* {
3. begin ***FirstAxisDistance***
`Distance calcul following the first axis`
4. for $i \in [0,N-1]$ *do* {
5. $g(i,x_2,.,x_d) \leftarrow \min_{0\leq l<N}\{|i-l|\ /\ f(l,x_2,.,x_d) = 0\}$ }
(with $\min(\emptyset) = \infty$*)*
6. end ***FirstAxisDistance***
7. ***Step 2***
8. $g_1(x_1,...,x_d) \leftarrow g(x_1,...,x_d)^2$
9. for $k \in [2,d]$ *do* {
10. begin ***AdditionalAxisSqDistance***
`assignment following the parabolas envelope`
11. for $(x_1,.x_{k-1},x_{k+1},.x_d) \in [0,N-1]^{d-1}$ *do* {
12. for $i \in [0,N-1]$ *do* {
13. $g_k(x_1,.x_{k-1},i,x_{k+1},.x_d) \leftarrow$
$\min_{0\leq l<N}\{(i-l)^2 + g_{k-1}(x_1,.,l,.,x_d)\}$ }}
14. end ***AdditionalAxisSqDistance*** }
15. return $g(x) = \sqrt{g_d(x_1,...,x_d)}$

FirstAxisDistance procedure is very simple and in linear time in the number of grid points, actually $2N^d$. AdditionalAxisSqDistance is much more complicated and requires explanation. It is detailed in algorithm (2). It is in $O(N^d)$, so the complete algorithm is in $O(d.N^d)$. The basic idea, coming from [18], and detailed in [17] for dimension 2, consists in considering at step k the column of distance computed at step $k-1$. If we note:
$(X,i) = (x_1,...,i,x_{k+1},.x_d)$, we can see that at step k the minimum (square) distance at point (X,i) is at most $g_{k-1}(X,i)$. Thanks to Pythagore, it is also at most $g_{k-1}(X,l) + (i-l)^2$ for $0 <$

$l < N$, and the min value gives the result. But the computation of all these values is suboptimal. On the other hand, considering the set F of parabolas $\left\{F_l^X(i) = g_{k-1}(X,l) + (i-l)^2\right\}$ $(0 \leq l < N)$, the result of AdditionalAxisSqDistance procedure is given by the lower envelope of F. The key point is the fact that two parabolas in F intersect at most once in $\mathbb{Z}^2$, and that the intersection is very easy to compute: F_l^X intersect F_j^X when $2(l-j)$ divides $(l^2 - j^2 + g_{k-1}(X,l) - g_{k-1}(X,j))$. This is the reason why it is optimal: The computation of the envelope for a given X is reduced to a matrix searching problem, which complexity is linear in this totally monotone case (see [18],[19]), so the overall complexity is in $O(d.N^d)$. It is also the reason why it can be modified for other distances, such as the sup norm, as long as this key point is maintained.

Algorithm 2 ***AdditionalAxisSqDistance***(g_{k-1})
Data*: The square Euclidean distance g_{k-1} computed on the $k-1$ first axes for each point of G*
Result*: The square Euclidean distance g_k computed on the k first axes for each point of G.*
1. for $X = (x_1, .x_{k-1}, x_{k+1}, .x_d) \in [0, N-1]^{d-1}$ *do* {
2. $q \leftarrow 0; s[0] \leftarrow 0; t[0] \leftarrow 0$
`s is the stack of parabolas`
`t the stack of parabolas intersections`
3. for $i \in [0, N-1]$ *do* {
4. while $q > 0$ *and* $F^X_{s[q]}(t[q]) > F^X_i(t[q])$ *do* {
5. $q \leftarrow q-1$ } `the present parabola dominates`
7. if $q = 0$ *then* {
8. $q \leftarrow 0; s[0] \leftarrow i$ }
`present parabola replaces all`
9. else {
10. $w \leftarrow 1 + \arg(intersection(F^X_{s[q]}, F^X_i))$
11. if $w < N$ *then* {
`present parabola replaces all (or is recruited) from` $q+1$
12. $q \leftarrow q+1$*;* $s[q] \leftarrow i$*;* $t[q] \leftarrow w$ }}}
13. for i from $N-1$ *to* 0 *do* {
14. $g^X_k(i) \leftarrow F^X_{s[q]}(i)$
`assignment following the parabolas envelope`
15. if $i = t[q]$ *then* {
16. $q \leftarrow q-1$ }}}
17. return g_k

2.2.2 Projection onto the decision boundary

The objective is to compute the projection onto the decision boundary for each point of the grid. With the projection point, it is possible to give the smallest perturbation that will lead the input state outside the viability set.

Points of $\hat{\Gamma}$ are labeled. When an intermediate value of the (square) distance is computed or modified, the label of the original point of $\hat{\Gamma}$ is also transfered. This induces a change in both procedures of Algorithm 1. FirstAxisDistance procedure returns the label of the nearest point along the first axis together with the distance. Line 5 of algorithm 1 becomes:
5. { $j \leftarrow \text{argmin}_{0 \leq l < N}\{|i-l| \,/\, f(l, x_2, ., x_d) = 0\}$;
$g(i, x_2, ., x_d) \leftarrow |i-l|$;

$p(i,x_2,..,x_d) \leftarrow \text{label}(j,x_2,..,x_d)$ }
AdditionalAxisSqDistance procedure transfers the label of the nearest point when the square distance is updated. Line 13 of algorithm 1 becomes:
13. { $j \leftarrow \text{argmin}_{0\leq l<N}\{(i-l)^2 + g_{k-1}(x_1,..,l,..,x_d)\}$;
$g_k(x_1,..x_{k-1},i,x_{k+1},..x_d) \leftarrow (j-l)^2 + g_{k-1}(x_1,..,j,..,x_d)$;
$p(x_1,..x_{k-1},i,x_{k+1},..x_d) \leftarrow p(x_1,..x_{k-1},j,x_{k+1},..x_d)$}
With more details in algorithm 2 we have the following modifications: The Data line includes p, and the assignment (lines 14, 16) modifies p:
14. { $g_k^X(i) \leftarrow F^X_{s[q]}(i)$
$p_k^X(i) \leftarrow p_k^X(s[q])$}
16. { $q \leftarrow q-1$
$p_k^X(i) \leftarrow p_k^X(i), p_k^X(s[q])$}
This modified algorithm, noted NearestPointToSet, gives the nearest point $\hat{P}$ of $\hat{\Gamma}$ for each point of the grid G. This point is used to provide an approximation of the projection onto Γ.

The complexity is in $O(d.N^d)$. Moreover, the algorithm can be easily parallelized for up to N^{d-1} processors.

3 Coding and study of the projection algorithm

The coding of the algorithm requires the use of at least 2 matrices: the first matrix serves to calculate the distances, the second matrix is for the projections. In order to use the same algorithm for all kind of viability sets in different state spaces, in particular with different dimension, we chose to use a Graphical Universal Model "aGRuM" library [1].

3.1 A Graphical Universal Model library

aGrUM is a *C++* library designed for easily building applications using graphical models such as Bayesian networks, influence diagrams, decision trees, GAI networks or Markov decision processes. It is written to provide the basic building blocks to perform the following tasks :

- graphical model learning / elicitation,
- inference within the graphical model,
- planification.

The aGRum part that interests us in this work is the branch *MultiDimImplementation*. In fact, A $MultiDimImplemenetation* < T\_DATA > *$ is the abstract base class for all multidimensional implementation of container of T_DATA. Its purpose is to implement basic algorithms with no regard to how the storage is done (tree or matrix or ...).
Thus, the use of aGRuM allows us to have an optimized tool to manage:

- the addition or suppression of the dimension
- cross multidimensional structure

So, we obtain a code optimization: to cover a multidimensional structure we use a single loop. Also, Thanks to the librairy aGRum, the generation of the various tested forms (hyperrectangle, hyperplan) was easy.

[1] http://agrum.lip6.fr

Table 1: nomber of points for each dimension : Search espace and hypercube

search space	5	7	10	15	17	20	25	30	35	40	50	60	70	80
hypercube 3 dim	-	-	-	-	-	10	-	15	-	20	25	30	35	40
hypercube 4 dim	-	-	5	7	-	10	13	15	17	-	-	-	-	-
hypercube 5 dim	3	5	5	7	8	-	-	-	-	-	-	-	-	-

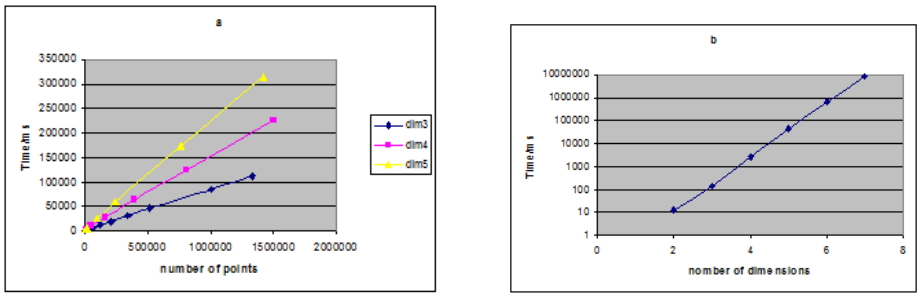

Fig. 3: Evolution of execution time according to: the points number by matrix (a), dimensions number (b)

3.2 The theoretical complexity of the algorithm :

We have shown in paragraphe 2.2. that the algorithm complexity is in $O(d.N^d)$. To confirm these theoretical results we applied the algorithm to different matrices where the bondary is represented by an hypercube.

This experiment is realized for 3 values of the dimension(see Figure 3(a)). To increase the size of the research space we have to vary the number of points for each dimension from 20 to 80 points whene we are in three dimensions, between 10 to 35 with 4 dimensions and 5 to 17 for 5 dimensions(see Table 1). The dimensions of the hypercube also vary according to the size of the research space.For each value of the dimension, the algorithm is linear in the number of points for which the distance and projections are computed.

The maximal number of point that the algorithm is able to treat in an example of hyper-ectangle (15 points for one side and 10 points for the other sides) is equal to 148035889 (6 dimensions, 23 points for each dimension) with a time of execution of 40201.175 s.

In the same way, to study the complexity of the algorithm with regard to the number of dimensions, we apply it to the example of a hyperplane with every time a different number of dimensions d. The number of dimensions varies from 2 to 7 (see Figure 3(b)) and for each dimension we have 11 points. The computer we used for these tests is a *Pentium(R)* 4 with *CPU* of 3.60 *GHz*. We stopped tests with $d = 7$ because beyond this limit ($d = 8$) the number of points is over 214 million consequently the capacity in memory of this computer is widely surpassed. In the Figure 3 (b), we represent the time evolution (ms) execution of the algorithm with a logarithmic scale according to the number of dimensions. As we can notice it, this evolution is exponential with comparison with the number of dimensions. All these results confirm a complexity of $O(d.N^d)$.

4 Application to the cheese ripening viability tube

4.1 A viability tube to represent the cheese ripening process

The cheese mass loss during ripening process was modeled by Helias (2007).The cheese mass loss is linked to the evaporation phenomena and the carbon loss through respiration of microorganisms. The evaporation increases with lower relative humidity in the ripening chamber and higher temperature on the cheese surface. This temperature changes with the ripening chamber temperature, evaporation phenomena and respiration of microorganisms ([8]).

4.2 Geometric analysis of the ripening process

The viability algorithm was used to compute the viability tube of the ripening cheese process. We then applied the distance and projection algorithm to this viability tube.

- Visualization of the distance maps of the ripening process
- Global indicators for the viability set
- Study of robustness of two experimental trajectories with the geometrical model.

4.2.1 Complete tube

The data we used in this work are organised on 11 days. The viability set of the 12th day is defined by the target itself. The dimension of the state space is 3. Each state is defined by the surface temperature (T°) of the cheese, the concentration of the micro-organisms (breathing in $g/m^2/jour$) and the mass of the cheese in g.

Complete tube robustness

to better show the successive viability sets, we eliminated the temperature dimension (the smallest one) and we juxtaposed the days (see Figure 4) from the 1st day to the 11th one. All the shown points are the viable points, thus we are sure to be inside the viability set. The colours show the distance of a state with regard to the bondary. The more red the colour is the more robust the state is (far from boundary), on the contrary, the more blue the colour is, the less robust the state is (closer to the boundary). Some boundaries of the viability set, that are normal to axes, represent the boundary of the exploration set. They are not considered as a true boundary for the computation of the distance map (the states that are not considered, across the hyperplanes, are supposed to be viable). The first day viability set is reduced to a one-dimension set, since the breathing dimension is not to be considered (the microorganisms have not germinated yet, so they can't breathe). So, we consider that all the proposed masses are viable because it's the starting day. From the 2nd day and until the 5th day (see Figure 4), we can notice that the most robust states are localised on the right corner of the matrix, however, the other robust states are added during these 5 days in the form of a beam (yellow color) which extends on towards the centre of the matrix. It is confirmed by the results of the percentage of the viable states with regard to the totality of states (see Figure 5). Indeed, for these 5 days, the percentage of the viable states increases from day to day, this percentage moves on from 1 % the first day to 89 % the 5th day. Also this increase

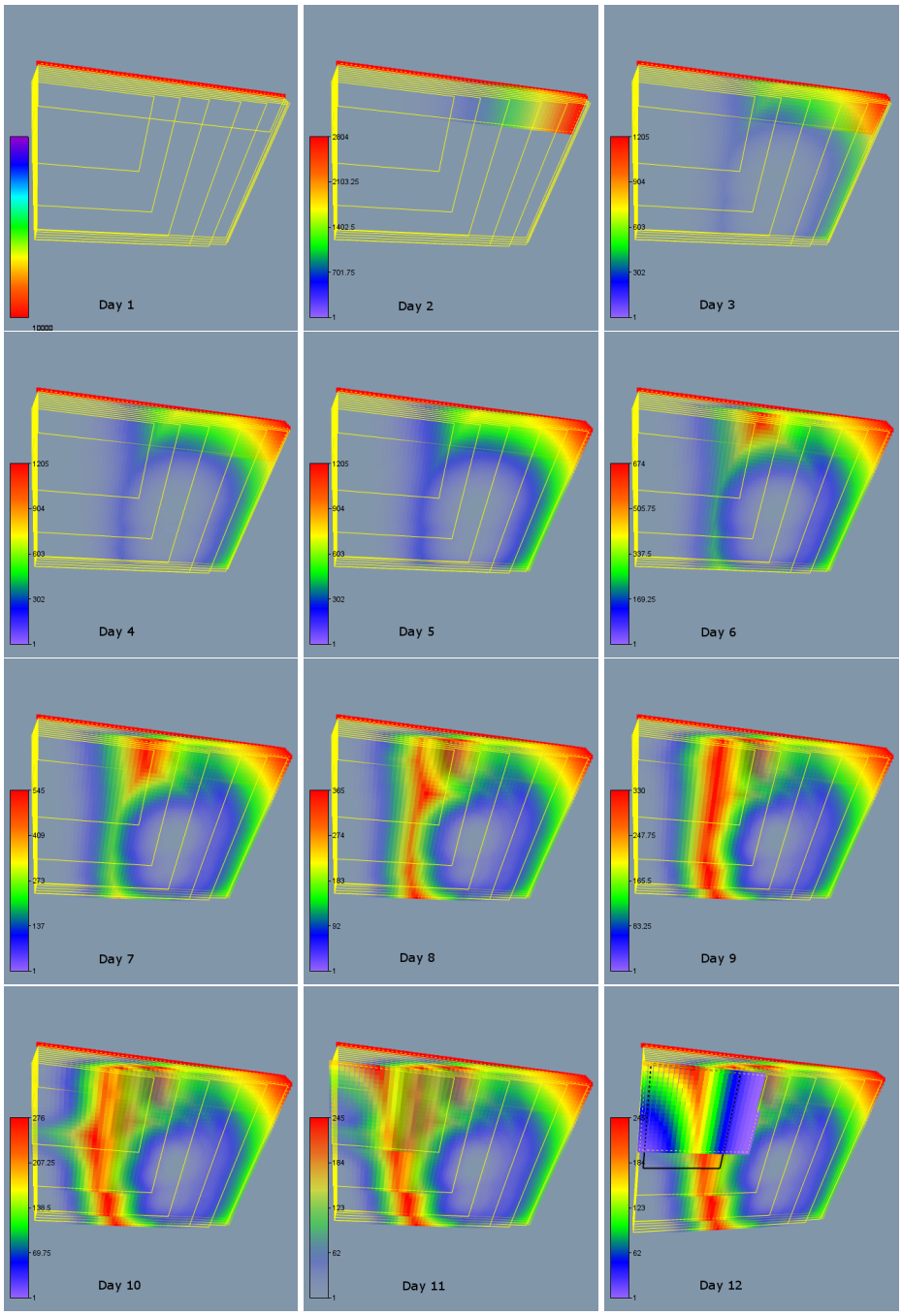

Fig. 4: Square distance map of the viability tube (12 viability sets corresponding to the different days of the process).

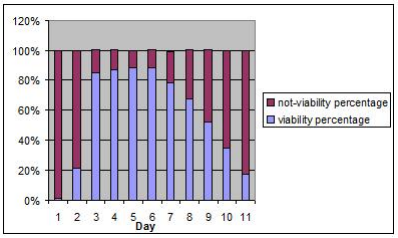

Fig. 5: Ratio of the number of viable states

of the viable states percentage confirms that during these 5 first days the process is robust and numerous trajectories are possible. During the 6th day, the most robust states concentrate in the middle of the matrix without that the percentage of viability changes considerably (88 %). This change of geometrical localization of the most robust states can be explained by the beginning (see table 1) or the sudden increase of the breath (see table 2). Consequently we can say that during the first 5 days the determining factor is the mass of the cheese and from the 5th day another factor becomes so important: microorganisms breathing. Besides, from the 6th day the process begins converging towards the target. This convergence is represented by a concentration of robust states in the middle of the matrix in a vertical beam translating a loss of the mass and an increase of the breathing of the microorganisms. The viability percentage decreases from day to day and this decrease can attain in certain days 17%. The procedure is less and less robust: few trajectories are possible. At the end of this process, the convergence towards the target is represented by a gap of the robust states towards the left corner of the matrix. This gap is due to the loss of the mass until it reaches the weight specified in the constraints of the target. At this stage (11th day), the percentage of the viable points reaches its lowest level: 17 %. Concerning the target(12th day, black bounding box on the Figure 4), its caracteristics(a breath included between 23 and 50, and a mass between 250g and 270g), gather states that are considered as the most robust by the geometrical modelling. So we can notice that certain values of the breath do not exist any more at the level of the 11th day (the black frame exceeds the limit of the 11th day).

Trajectories robustness

We used the geometric study to give robustness information on two trajectories. The first one (see table 1) is generally followed in real factories. It is organised over 11 days. The second one (see table 2) was obtained by the GMPA laboratory of INRA.

For each day, the distance map gives the distance to the viability set boundary. For each of the trajectories, we note the state that is reached following the corresponding trajectory. A viable state is characterized by its temperature (T°), its mass and its breath. The distance to the viability set boundary is registrered, with the radius of the maximal maximal ball for this day. Distances in these tables are the square root of distances represented in Figure 4. According to the table 1, we can notice that with regard to the calculation of the distances on the bordary the states of the first trajectory remain more or less taken away from this bordary and are located in the maximal maximal ball calculated for every day, so as 10th and 11th days when both states approach dangerously the bondary. Their distances are 9 and 6 of the bondary respectively. The second trajectory is organized in only 8 days; Its 8 states are characterized by an average distance to the boundary. Thanks to the calculation of the maximal ball radius for every matrix, we are capable of proposing trajectories where

every state is far as possible from the viability set boundary and approaches as much as possible of the skeleton calculated from the 2-projections states. These trajectories will have the characteristic to resist perturbation.

Table 2: Nominal trajectory

Days	W	T°	B	d	mmbr
1	284	285	0	21.21	31.4
2	280	284	0	23.6	34.19
3	278	285	0	22.5	32.01
4	276	285	0	21.56	32.01
5	274	285	0	20.64	32.01
6	271	285	5	17.20	25.96
7	270	285	13	16.03	23.34
8	267	285	24	13.04	19.10
9	264	285	35	11.79	18.16
10	260	285	37	9	16.61
11	255	285	33	6	15.65

Table 3: experimental trajectory

W	T°	B	d	mmbr
284	285	0	21.21	31.4
280	285	0	23.43	34.19
279	286	1	23	32.01
278	287	4	22.30	32.01
276	287	12	18.50	32.01
274	286	21	14.50	25.96
272	286	37	13.30	23.34
269	282	37	15.65	19.10
–	–	–	–	–
–	–	–	–	–
–	–	–	–	–

5 Conclusion

The geometrical modelling of viability set allowed us to determine the robust states of the process and also to identify the phases of the dynamic where several trajectories are possible and on the contrary the phases of constriction where few trajectories are possible. Thanks to this geometrical modelling and to the calculation of projections of states on the viability set boundary, the algorithm is capable of producing the skeleton of the tube. This skeleton will allow us in future work to propose more robust trajectories.
Thanks to the complexity mastered by the algorithm, we also intend to introduce the controls as state variables in order to take into account the uncertainty on the control parameters as well.

References

1. Aubin, J.: Viability Theory. Birkhauser, Basel (1991)
2. Aubin, J.P., Bonneuil, N., Maurin, F., Saint-Pierre, P.: Viability of pay-as-you-go systems. Journal of Evolutionary Economics **11** (2001) 555–571
3. Bonneuil, N., Mllers, K.: Viable populations in prey-predator system. Journal of Mathematical Biology **35** (1997) 261–293
4. Bn, C., Doyen, L., Gabay, D.: A viability analysis for a bio-economic model. Ecological Economics **36** (2001) 385–396
5. Mullon, C., Cury, P., Shannon, L.: Viability model of trophic interactions in marine ecosystems. Natural Resources Modelling **17**(1) (2004) 27–58
6. Bayen, A., Claudel, C., Saint-Pierre, P.: Viability-Based Computations of Solutions to the Hamilton-Jacobi-Bellman Equation. Lecture Notes in Computer Sciences **4416** (2007) 645–649
7. Aubin, J.P., Martin, S.: Travel Time Tubes Regulating Transportation Traffic. Contemporary Mathematics (to appear)
8. Sicard, M., Perrot, N., Baudrit, C., Reuillon, R., Bourgine, P., Alvarez, I., Martin, S.: The viability theory to control complex food processes. In: Submitted to the European Conference on Complex Systems. (2009)
9. Saint-Pierre, P.: Approximation of viability kernel. Applied Mathematics and Optimization **29** (1994) 187–209

10. Deffuant, G., Chapel, L., Martin, S.: Approximating viability kernels with Support Vector Machines. IEEE Transactions on automatic control **52**(5) (2007) 933–937
11. Bonneuil, N., Saint-Pierre, P.: Protected polymorphism in the two-locus haploid model with unpredictable fittnesses. Journal of Mathematical Biology **40** (2000) 251–277
12. Coeurjolly, D., A., M.: Optimal Separable Algorithms to Compute the Reverse Euclidean Distance Transformation and Discrete Medial Axis in Arbitrary Dimension. IEEE Transactions on Pattern Analysis and Machine Intelligence **29**(3) (March 2007) 437–448
13. 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)
14. Alvarez, I.: Explaining the result of a decision tree to the end-user. In: Proc. of the 16th European Conference on Artificial Intelligence, IOS Press (2004) 411–415
15. Matheron, G. Image Analysis and Mathematical Morphology. In: Examples of topological properties of skeletons. Academic Press (1988) 217–257
16. Plous, S.: The Psychology of Judgment and Decision Making. McGraw-Hill (1993)
17. Meijster, A., Roerdink, J., Hesselink, W.: A General Algorithm for Computing Distance Transforms in Linear Time. Morphology and Its Applications to Image and Signal Processing (2000) 331–340
18. Hirata, T.: A Unified Linear-Time Algorithm for Computing Distance Maps. Information Processing Letters **58**(3) (1996) 129–133
19. Aggarwal, A., Klowe, M., Moran, S., Shor, P., Wilber, R.: Geometric applications of a matrix-searching algorithm. Algorithmica **2** (1987) 195–208