Acta Cybernetica **00** (0000) 1–18.

Boundary approach to characterize the inner and outer approximation of the image of a disk [*]

Maël GODARD [ab], Luc JAULIN [ac], and Damien MASSÉ[de]

Abstract

Calculating directly the inner and outer approximation of the image of a set by a function can be challenging. Then, it is sometimes prefered to compute the image of the boundary of the set instead. However, boundary-based methods are subject to the apparition of fake boundaries in the image set. As they add pessimism when characterizing the inner approximation of the image set, this paper introduces the notion of Box Chains to simplify their detection and their suppression. The characterization of the inner and outer approximation of the image set in the case of a function from the unit disk $\mathcal{D}$ to $\mathbb{R}^2$ will be considered, with two examples.

Keywords: Fake boundaries, Box Chains, Boundary approach

1 Introduction

Calculating the image of a set by a function has many applications in robotics [4] : Image of a subpaving [5], estimation of the area covered by a sensor [2], state estimation [7] ...

The methods to calculate the image of a set by a function can be divided into two subclasses. The set-based methods compute directly an outer , and sometimes an inner, approximation of the image of the whole set. They can rely on Interval Analysis for the operations on sets. The boundary-based methods compute instead the image of the boundary of the set, and from this image an inner and outer approximation of the image set can be deduced. These methods are lighter to calculate, but they are subject to the appearance of fake boundaries in the image set.

These fake boundaries are the result of the difference between the boundary of the image set, and the image of the boundary. These fake boundaries are parts of

[*]This work was supported by the Defense Innovation Agency (AID) and the Brittany Region.

[a]Lab-STICC, ROBEX Team, ENSTA Bretagne
[b]E-mail: mael.godard@ensta-bretagne.org, ORCID: 0009-0002-2822-6215
[c]E-mail: luc.jaulin@ensta-bretagne.fr, ORCID: 0000-0002-0938-0615
[d]Lab-STICC, ROBEX Team, UBO
[e]E-mail: damien.masse@univ-brest.fr, ORCID: 0000-0002-4485-8936

DOI: 10.14232/actacyb.?

the image of the boundary that are in facts inside of the image set and should be classified as such. As they add an unwanted pessimism to the estimation of the inner approximation of the image set, these fake boundaries have to be removed.

This problematic has often been adressed [1] [2] [10] as different applications favor different solutions. The method presented in this article aims to be usable in different applications while remaining as computationally light as possible.

To do so, Section 2 presents the notions and notations that will be used for the remainder of the article. Section 3 introduces the notion of Box Chains that will be used in the boundary simplification algorithm of Section 4. For this article two examples from the unit disk $\mathcal{D}$ to $\mathbb{R}^2$ will be considered. Finally Section 5 concludes the paper.

2 Problem presentation

2.1 Notations and definitions

Let $\mathcal{D}$ be the unit disk with $\mathcal{S}^1$, the unit circle, its contour. As computing the image of the whole disk $\mathcal{D}$ can be challenging, it is often prefered to compute the image of its boundary $\mathcal{S}^1$ instead. For this article we will consider a function $\mathbf{f} : \mathcal{D} \to \mathbb{R}^2$. This function satisfies :

- $\mathbf{f}$ is $\mathcal{C}^1$, we will note $\mathbf{J_f}$ its Jacobian function.
- $\mathbf{f}$ has no singularity on $\mathcal{D}$.

This second point can be verified by checking that the determinant of the jacobian of $\mathbf{f}$ is never null. Thanks to these assumptions, cases where the image of the set contains a cusp or a fold, as depicted Figure 1, will not be considered.

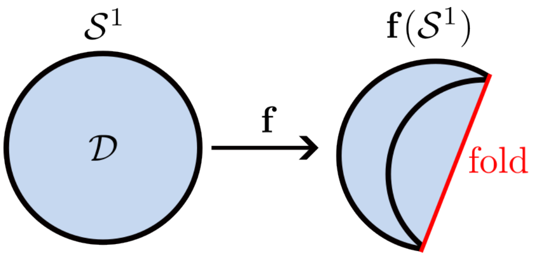
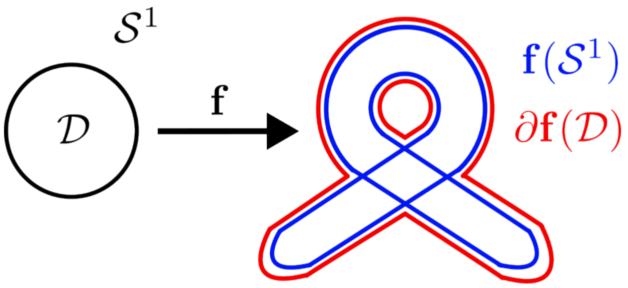


Figure 1: Fold in the image of the set

As depicted in Figure 2, we then have:

$$\partial \mathbf{f}(\mathcal{D}) \subseteq \mathbf{f}(\mathcal{S}^1) \tag{1}$$



$\mathcal{S}^1$
$\mathcal{D}$
$\mathbf{f}$
$\mathbf{f}(\mathcal{S}^1)$
$\partial\mathbf{f}(\mathcal{D})$

Figure 2: Fake boundaries

Calculating the image of $\mathcal{S}^1$ instead of the image of $\mathcal{D}$ makes fake boundaries appear. These fake boundaries are defined by:

$$FB_{\mathbf{f}}(\mathcal{D}) = \mathbf{f}(\mathcal{S}^1) \setminus \partial\mathbf{f}(\mathcal{D}) \tag{2}$$

When calculating the image of a disk, or any other closed outline, we often want to compute the inner and the outer approximation of the image set to verify if they satisfy a given condition (e.g. obstacle avoidance [3] [6]).

Given $\partial\mathbf{f}(\mathcal{D})$ it is possible to compute the inner and the outer approximation of the image set. However using $\mathbf{f}(\mathcal{S}^1)$ will give a more pessimistic result as the neighborhood of $FB_{\mathbf{f}}(\mathcal{D})$ will be considered as part of the boundary.

Remark 1. As $\mathbf{f}$ is continuous, the image of $\mathcal{S}^1$ by $\mathbf{f}$ is a closed outline in $\mathbb{R}^2$.

For practical reasons we define the function displayed Figure 3 $\phi : [0, 2\pi] \to \mathcal{S}^1$ by :

$$\forall t \in [0, 2\pi], \phi(t) = \begin{pmatrix} cos(t) \\ sin(t) \end{pmatrix} \tag{3}$$

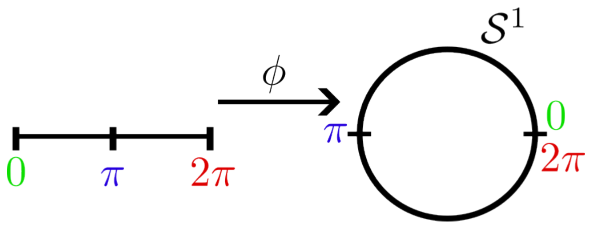


Figure 3: Function ϕ from $[0, 2\pi]$ to $\mathcal{S}^1$

Remark 2. This function ϕ is bijective over $[0, 2\pi[$ and $\phi(0) = \phi(2\pi)$. This means that ϕ is bijective over any strict subset of $[0, 2\pi]$.

For the sake of simplicity, let us denote $\mathbf{g}$ the function $\mathbf{f} \circ \phi$, function from $[0, 2\pi]$ to $\mathbb{R}^2$. We then have:

$$\forall t \in [0, 2\pi], \mathbf{g}(t) = \mathbf{f}(\phi(t)) \tag{4}$$

Considering an interval $[t] \subset [0, 2\pi]$, studying the properties of $\mathbf{g}$ over $[t]$ or the properties of $\mathbf{f}$ over $\phi([t])$ will then give the same result as ϕ is bijective on this interval (see Remark 2). For instance, if $\mathbf{g}$ is injective over $[t]$, then $\mathbf{f}$ is injective over $\phi([t])$. For the remaining of this article the focus will then be on the function $\mathbf{g}$ to detect and remove the fake boundaries in the image of the unit disk $\mathcal{D}$ by $\mathbf{f}$. Note that as for $\mathbf{f}$, $\mathbf{g}$ is a $\mathcal{C}^1$ function and we note $\mathbf{J_g}$ its Jacobian.

As computing the exact image of $[0, 2\pi]$ by $\mathbf{g}$ is not possible, in this article we will consider a cover C of $[0, 2\pi]$ that does not contain $[0, 2\pi]$, and compute $[\mathbf{g}([t])]$ for each interval $[t] \in C$. The result is a set of boxes that contains the true boundary, image of $[0, 2\pi]$ by $\mathbf{g}$. Figure 4 shows the computation of the image of $[0, 2\pi]$ by $\mathbf{g}$ in the case where $\mathbf{f}$ is the identity.

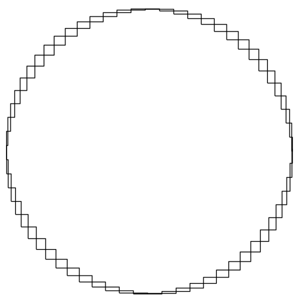


Figure 4: Image of $[0, 2\pi]$ by $\mathbf{g}$ when $\mathbf{f}$ is the identity

2.2 Problem formulation

The objective of this article is to compute the inner and the outer approximation of the image of the unit disk $\mathcal{D}$ by a function $\mathbf{f}$ using a boundary approach. To do so, we limit our study to the cases where the function $\mathbf{f}$ respects the assumption presented in Subsection 2.1.

To do so, and as suggested in Subsection 2.1, the study of the boundary will rely on the function $\mathbf{g} : [0, 2\pi] \to \mathbb{R}^2$ defined by $\mathbf{g} = \mathbf{f} \circ \phi$, ϕ being the function defined by Equation (3).

As we are using a boundary approach, the method presented here is subject to the appearance of fake boundaries. As these fake boundaries add an unwanted pessimism to the computation of the image set, the first step is to detect and remove them.

Section 3 introduces the notion of Box Chains to decompose $[0, 2\pi]$ into subsets where $\mathbf{g}$ is injective in order to facilitate the detection of self-intersections in the boundary. This definition will later be used in the boundary simplification algorithm presented in Section 4. Once the fake boundaries have been removed, the computation of the inner and outer approximation of the image set can be done with less pessimism.

3 Box Chains

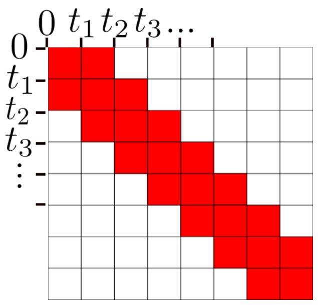
3.1 Neighborhood relation

To define the notion of Box Chain we first need to introduce the relation of neighborhood.

Definition 1. *Let $[t_i] \in \mathbb{IR}$ and $[t_j] \in \mathbb{IR}$ be two intervals. We define the neighborhood relation noted $\mathcal{R}_n$ between $[t_i]$ and $[t_j]$ as :*

$$[t_i]\,\mathcal{R}_n\,[t_j] \Leftrightarrow [t_i] \cap [t_j] \neq \emptyset \tag{5}$$

This relation can be represented for real-value intervals in the t-plane [9] or by its logical matrix as shown Figure 5.



$$\begin{array}{c|cccccccc}
 & 0 & t_1 & t_2 & t_3 & \dots & & & \\
\hline
0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
t_1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
t_2 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\
t_3 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\
\vdots & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\
 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\
 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\
 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1
\end{array}$$

Figure 5: Neighborhood Relation in the t-plane and logical matrix

The t-plane represents real-value intervals $[0, t_1], [t_1, t_2], \dots$ on the abscissa and on the ordinate. The grid is then colored where the corresponding intervals verify the relation, and is left blank otherwise. The resulting grid can be interpreted as a logical matrix, say M, where the blank boxes are null, and the colored boxes are ones. This graphical representation of the relation highlights its properties:

- **Reflexivity** As the main diagonal of the grid has no blank box, i.e. the identity matrix is included in M, it means that the relation is reflexive.

- **Symmetry** As M is symmetric, the relation itself is symmetric.

However we can see that $M^2 \neq M$, meaning that this relation is not transitive.

3.2 Box Chain relation

Definition 2. *Let $[t_i] \in \mathbb{IR}$ and $[t_j] \in \mathbb{IR}$ be two intervals and $\mathbf{g} : \mathbb{R} \to \mathbb{R}^2$. We define the Box Chain relation for $\mathbf{g}$, noted $\mathcal{R}_{BC,\mathbf{g}}$ between $[t_i]$ and $[t_k]$, as :*

$$[t_i]\,\mathcal{R}_{BC,\mathbf{g}}\,[t_k] \iff \exists\,[t_{j_1}],\dots,[t_{j_m}] \in \mathbb{IR}^m, \begin{cases} [t_i]\,\mathcal{R}_n\,[t_{j_1}] \wedge \cdots \wedge [t_{j_m}]\,\mathcal{R}_n\,[t_k] \\ \mathbf{g}_{|[t_i]\cup[t_{j_1}]\cup\ldots\cup[t_{j_m}]\cup[t_k]} \text{ is injective} \end{cases} \tag{6}$$

Equivalently, $[t_i]$ and $[t_k]$ are in Box Chain relation for $\mathbf{g}$ if a trajectory from neighbor to neighbor exists between $[t_i]$ and $[t_k]$ on which $\mathbf{g}$ is injective.

The t-plane representation and the logical matrix of the Box Chain relation depends on both the expression of $\mathbf{g}$ and the chosen $t_1, t_2, \cdots$. However this relation is always **symmetric** as the neighborhood relation and the union of sets are symmetric.

3.3 Box Chain decomposition

As depicted in Figure 2, fake boundaries appear when the considered contour crosses itself. This means that the considered function is not globally injective as two distincts inputs give the same output.

As mentioned in Section 2, the boundary we are dealing with is not a line, but a set of boxes. Finding the self intersections in the contour can then be hard as each box of the contour crosses at least two other boxes as shown Figure 6. This is the result of the continuity of the function $\mathbf{g}$.

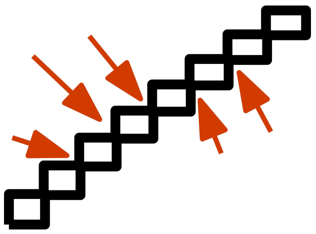


Figure 6: Intersections between the boxes

To make this detection easier, Box Chains can be used to decompose the domain of any function into subsets on which the function is locally injective. Looking for the self-intersections in the boundary will then come down to looking for the intersections between different Box Chains. Algorithm 1 is suggested to perform this decomposition. It takes three inputs :

- The studied function $\mathbf{g}$
- An injectivity criterion h
- A cover of $[0, 2\pi]$, C

The injectivity criterion is defined over the powerset of $[0, 2\pi]$, $\mathcal{P}([0, 2\pi])$. For a given interval in $[0, 2\pi]$ it outputs 1 if $\mathbf{g}$ is injective over this interval, 0 otherwise. This criterion must be sufficient, but does not need to be necessary.

The algorithm takes the element of the cover C one by one and try to group them into Box Chains. As soon as an element can not be added to the Box Chain without loosing the injectivity of $\mathbf{g}$ on it, a new Box Chain is created. Finally the algorithm sorts the intervals of the cover C into a list of Box Chains.

Algorithm 1 Box Chain decomposition.

Input : a function $g : \mathcal{D} \to \mathbb{R}^2$, an injectivity criterion $h : \mathcal{P}([0, 2\pi]) \to \{0, 1\}$ and C a cover of $[0, 2\pi]$

Output : $\mathcal{L}_{BC}$ a list of Box Chains

```
 1: Set the working list $\mathcal{L}_W := C$ and the final list $\mathcal{L}_{BC} := \{\}$
 2: Pop $E$ from $\mathcal{L}_W$
 3: while ( $\mathcal{L}_W \neq \emptyset$ ) do
 4:     Initialize the Box Chain $\mathcal{L}_B := \{E\}$
 5:     while ($\mathcal{L}_W \neq \emptyset$) do
 6:         Pop $E$ from $\mathcal{L}_W$
 7:         if $h(\mathcal{L}_B \cup E)$ and $\mathcal{L}_B \mathcal{R}_n E$ then        injectivity criterion
 8:             Store $E$ in $\mathcal{L}_B$
 9:         else
10:             break
11:         end if
12:     end while
13:     Store $\mathcal{L}_B$ in $\mathcal{L}_{BC}$
14: end while
15: return $\mathcal{L}_{BC}$
```

An example of injectivity criterion could rely on the tangent to the contour. The Jacobian of a function represents this tangent when defined, see Figure 7. Note that as the domain of the studied function $\mathbf{g}$ is $\mathbb{R}$, its Jacobian is a vector.

Theorem 1. *Let $\mathbb{T}$ be a subset of $\mathbb{R}$, $\mathbf{g} : \mathbb{R} \to \mathbb{R}^2$ be a $\mathcal{C}^1$ function and $\mathbf{J_g} : \mathbb{R} \to \mathbb{R}^2$ its Jacobian. If $0_{\mathbb{R}^2} \notin \left[\bigcup_{t \in \mathbb{T}} \mathbf{J_g}(t)\right]$, then $\mathbf{g}$ is injective over $\mathbb{T}$.*

Proof. Let us demonstrate the contrapositive.

If $\mathbf{g}$ is not injective, $\exists(t_1, t_2) \in \mathbb{T}^2$, $t_1 \neq t_2$ so that $\mathbf{g}(t_1) = \mathbf{g}(t_2)$ as shown Figure 7. Let us denote $\forall i \in \{1, 2\}$, $g_i : \mathbb{R} \to \mathbb{R}$ the function that gives the i^{th} component of $\mathbf{g}$. t_1 and t_2 then verify :

$$\forall i \in \{1, 2\}, g_i(t_1) = g_i(t_2) \tag{7}$$

The mean value theorem can then be applied to each of the g_i functions, giving

$$\exists \tau_i \in [t_1, t_2], J_{g,i}(\tau_i) = 0 \tag{8}$$

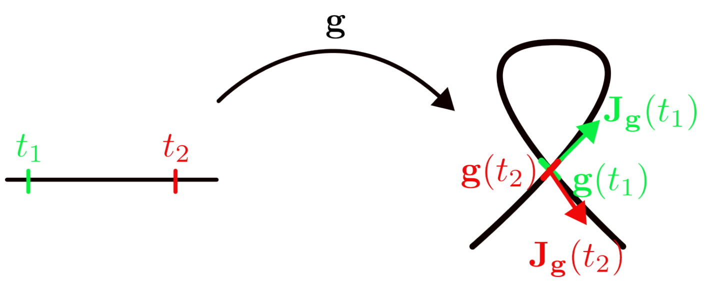


Figure 7: Loop in $\mathbb{R}^2$, $\mathbf{g}$ is not injective

Meaning that

$$\forall i \in \{1,2\}, 0 \in J_{g,i}([t_1,t_2]) \tag{9}$$

Finally,

$$0_{\mathbb{R}^2} \in J_{g,1}([t_1,t_2]) \times J_{g,2}([t_1,t_2]) \tag{10}$$

As $J_{g,1}([t_1,t_2]) \times J_{g,2}([t_1,t_2]) = \left[\bigcup_{t\in[t_1,t_2]} \mathbf{J_g}(t)\right]$ as shown Figure 8, we get that if $\mathbf{g}$ is not injective over $\mathbb{T}$, $0_{\mathbb{R}^2} \in \left[\bigcup_{t\in\mathbb{T}} \mathbf{J_g}(t)\right]$

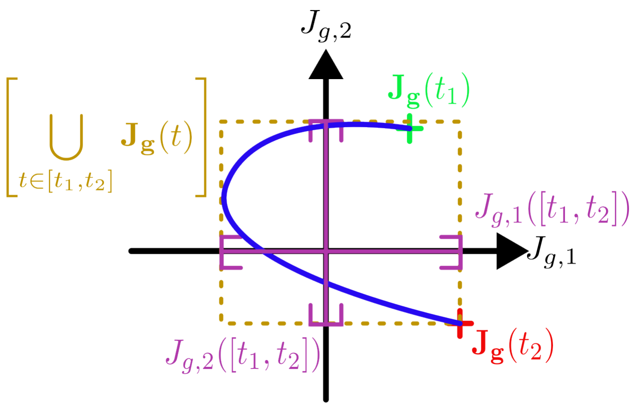


Figure 8: The brown box contains the origin, we can't conclude on the injectivity

□

4 Determination of the inner and outer approximation of the image a disk

To limit the pessimism while determining the inner and outer approximation of the image set, the first step is to remove the fake boundaries. To do so, the Box Chains presented in the previous section will be used to detect the boxes that belong to the self intersections in the boundary. Once these boxes have been set aside, the fake boundary can be removed before the computing of the inner and outer approximation of the image set.

4.1 Boundary Simplification algorithm

In this section, the function $\mathbf{f}$ defined Equation (11) will be considered for the illustrations. However the algorithm presented works with any function as long as the assumptions from Section 2 are satisfied.

$$\forall \mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \in \mathcal{D}, \mathbf{f}(\mathbf{x}) = \begin{pmatrix} x_1^2 - x_2^2 + x_1 \\ 2x_1x_2 + x_2 \end{pmatrix} \tag{11}$$

As explained in Section 2, when studying this function on its boundary $\mathcal{S}^1$, we will work with the function $\mathbf{g} : [0, 2\pi] \to \mathbb{R}^2$ for illustration purposes. It is defined by:

$$\forall t \in [0, 2\pi], \mathbf{g}(t) = \begin{pmatrix} cos(t)^2 - sin(t)^2 + cos(t) \\ 2cos(t)sin(t) + sin(t) \end{pmatrix} \tag{12}$$

If we consider a cover C of $[0, 2\pi]$, Figure 9 shows the image of C by the function $\mathbf{g}$ of Equation (12) with and without fake boundaries. As mentionned earlier, the boundary is here a set of boxes that constitutes an over-approximation of the real boundary.

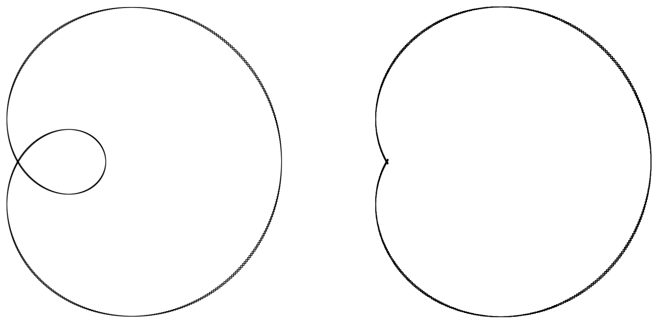


Figure 9: Boundary with (left) and without (right) fake boundaries

The different steps to eliminate the fake boundaries in the image of C by $\mathbf{g}$ are described below.

4.1.1 Step 1: Box Chain decomposition

The first step before removing the fake boundaries is to look for the self-intersections in the boundary. Indeed the boxes that belong to the intersections have to be conserved in the resulting boundary.

The first step is then to decompose the cover C into Box Chains as suggested in Subsection 3.3. This decomposition will make the detection of the fake boundaries easier. To do so Algorithm 1 can be used and Theorem 1 gives an injectivity criterion h:

$$\forall[t] \subset [0, 2\pi], h([t]) = \begin{cases} 0 \; if \; \mathbf{0}_{\mathbb{R}^2} \in [\mathbf{J_g}([t])] \\ 1 \; otherwise \end{cases} \tag{13}$$

C can then be decomposed into Box Chains thanks to Algorithm 1 to get the result Figure 10. As expected the self intersections in the boundary appear between differents Box Chains.

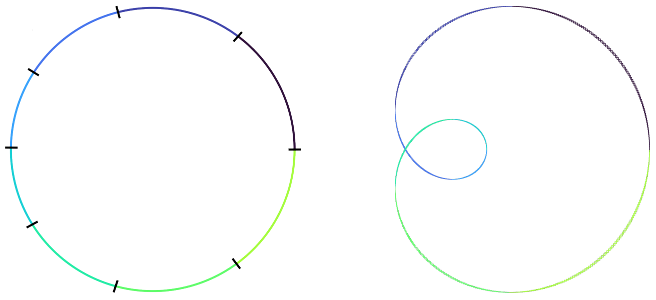


Figure 10: Eight Box Chains in $\mathcal{S}^1$ (left) and their image by $\mathbf{f}$ (right)

4.1.2 Step 2: Finding the self intersections

Once the Box Chain decomposition of C has been done, we can look for intersections between the images of the Box Chains by $\mathbf{g}$. Two cases of intersections can be observed Figure 11.

- If two Box Chains are in neighborhood relation, their images by $\mathbf{g}$ intersect each other at their junction point.
- If fake boundaries exist, the boundary crosses itself and the image of two different Box Chains intersect each other.

Remark 3. The neighborhood relation can be extended to Box Chains : Let there be two Box Chains B_1 and B_2, $B_1 \mathcal{R}_n B_2 \Leftrightarrow \exists([t_1], [t_2]) \in B_1 \times B_2, [t_1]\mathcal{R}_n[t_2]$

The injectivity criterion h can be used to distinguish these two cases. Indeed in the first case the function $\mathbf{g}$ is injective around the junction point between the two Box Chains, which is not the case when the boundary really crosses itself. With this criterion applied the resulting intersections can be computed as visible in red Figure 12.

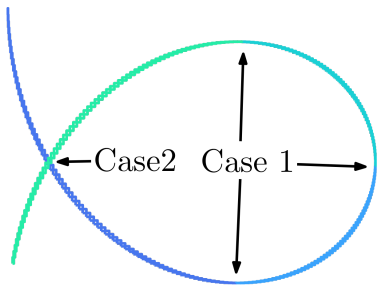


Figure 11: Two cases of intersections between the images of two Box Chains

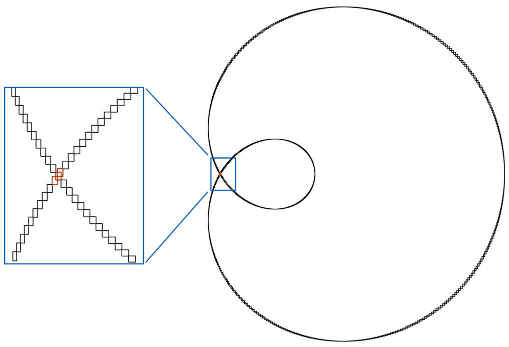


Figure 12: Self intersection in the boundary (in red)

4.1.3 Step 3: Determination of the inner areas

To determine the inner approximation of the image set an initial paving of the set $S = \bigcup_{[t]\in C} [\mathbf{g}([t])]$ is needed, for example with the SIVIA algorithm [4], to see what belongs to the boundary and what does not. The boxes that are not intersecting the boundary can then be divided into connected subsets as shown Figure 13. For this work we will rely on the connected subset decomposition implemented in the Codac library [8].

Then instead of qualifying each box individually as inside or outside the image set, we can qualify the whole connected subset. This means that once a box is proven to be inside, the corresponding connected subset can directly be classified as inside.

A solution to distinguish the boxes that are inside and outside is to look at the normal to the boundary. As depicted Figure 14 the normal of a contour points towards the exterior, meaning that the opposite corner is inside.

Remark 4. For a given interval $[t] \in [0, 2\pi]$ the tangent to the boundary evaluated in $[\mathbf{g}]([t])$ belongs to $[\mathbf{J_g}([t])]$, and rotating $[\mathbf{J_g}([t])]$ by $\pi/2$ (or $-\pi/2$) gives a box containing the normal to the boundary.

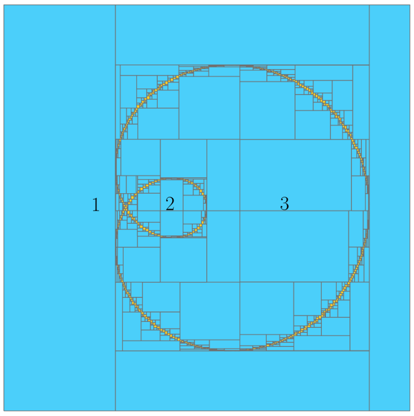


Figure 13: Paving with three connected subsets

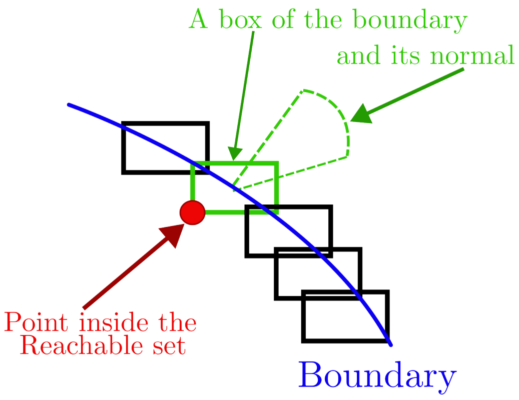


Figure 14: Determination of the inner areas

Applying this to every box of the boundary, except the ones that were previously proven to belong to the self intersections, gives a set of points that are inside. The connected subsets containing at least one of these points can be marked as inside, and the other can be marked as outside. Figure 15 shows an example of output for this step.

4.1.4 Step 4: Suppressing the fake boundaries

As mentioned earlier the normal to the boundary is supposed to point towards the exterior. Figure 16 illustrates the fact that in the case of fake boundaries the normal points towards an area that was classified as inside in the last step.

Suppressing the boxes with a normal pointing towards an inside area allows us to remove the fake boundaries to obtain a less pessimistic approximation of $\partial\mathbf{f}(\mathcal{D})$ as visible the result Figure 17. Note that as the self-intersections in the contour

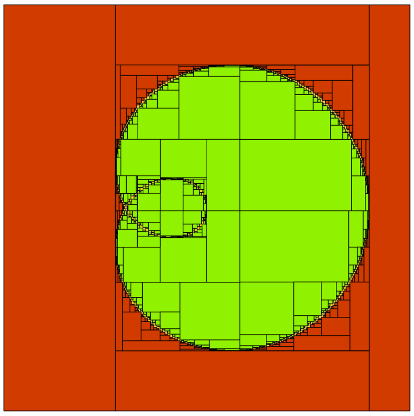


Figure 15: Inner (green) and outer (green + yellow) approximation of the image set with fake boundaries

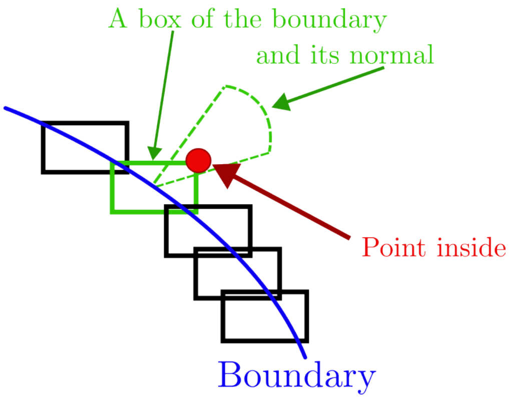


Figure 16: Case of a fake boundary

have been detected in Step 2, it is possible to propagate the information of a box belonging to the fake boundary from neighbor to neighbor until an intersection is reached.

4.2 Inner and outer approximation of the image set

Once the fake boundaries have been removed, Step 3 can be applied again with the remaining boxes to get the inner and outer approximation of the image set without fake boundaries.

To do so if we denote by C_r the set of remaining intervals of the cover C after the boundary simplification, we first proceed to an initial paving of $S = \bigcup_{[t]\in C_r} [\mathbf{g}([t])]$. Then the boxes that are not part of the boundary are sorted into

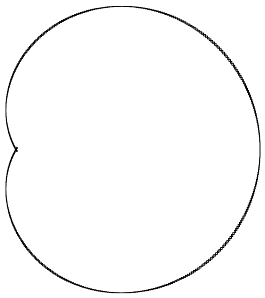


Figure 17: Fake boundaries removed

connected subsets and are finally classified as inside or outside the image set with the criteria illustrated Figure 14.

Finally, we are able to characterize the inner and outer approximation of the image of the unit disk. Figure 18 shows the result obtained with the function $\mathbf{f}$ defined by Equation (11)

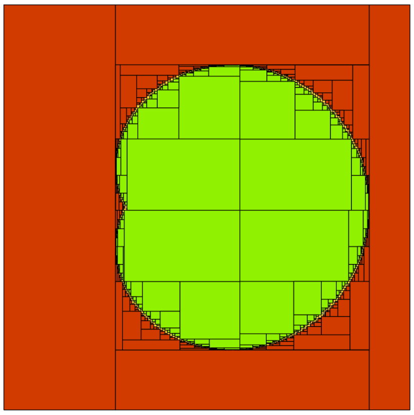


Figure 18: Inner (green) and outer (green + yellow) approximation of the image set without fake boundaries

4.3 Additional example

The algorithm presented here also work in more complex cases. Algorithm 2 gives another example of a function $\mathbf{f} : \mathcal{D} \to \mathbb{R}^2$.

As earlier a function $\mathbf{g} : [0, 2\pi] \to \mathbb{R}^2$ can be defined by $\forall t \in [0, 2\pi], \mathbf{g}(t) = \mathbf{f}(\phi(t))$. Figure 19 shows the image of the unit circle $\mathcal{S}^1$ by this function $\mathbf{f}$. This result is obtained by considering a cover C of $[0, 2\pi]$ and computing $[\mathbf{g}([t])]$ for each interval $[t] \in C$.

With this function, Algorithm 1 gave 15 Box Chains as shown Figure 20.

Algorithm 2 Pseudocode of the implementation of the $\mathbf{f}$ function

```
Define $s(\tau, a, b, c) = a + (b - a) \cdot 0.5 \cdot (1 + \tanh(10 \cdot (\tau - c)))$

Input: $\mathbf{p} = (p_1, p_2)$ a point in the unit disk

$\rho = \sqrt{p_1^2 + p_2^2}$
$\alpha = arctan2(p_2, p_1)$
$\tau = 0.03 + \alpha \cdot \frac{1.01}{2\pi}$
$t_1 = 1 - \cos(2\pi\tau)$
$\mathbf{d} = \begin{pmatrix} 5 - 50 \cdot \cos(5 \cdot t_1) \\ 30 \cdot \sin(5 \cdot t_1) \end{pmatrix}$
$\theta = s(\tau, -3 \cdot \pi/2, -\pi/2, 0) + s(\tau, 0, \pi, 0.5) + s(\tau, 0, \pi, 1)$
$\mathbf{y} = \begin{pmatrix} 5 \cdot t_1 - 5 \cdot \sin(5 \cdot t_1) \\ 2 - 3 \cdot \cos(5 \cdot t_1) \end{pmatrix} + \frac{\rho}{\sqrt{d_1^2 + d_2^2}} \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \mathbf{d}$
Output: $\mathbf{y}$
```

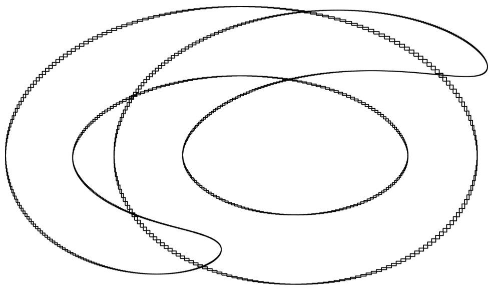


Figure 19: Boundary with fake boundaries

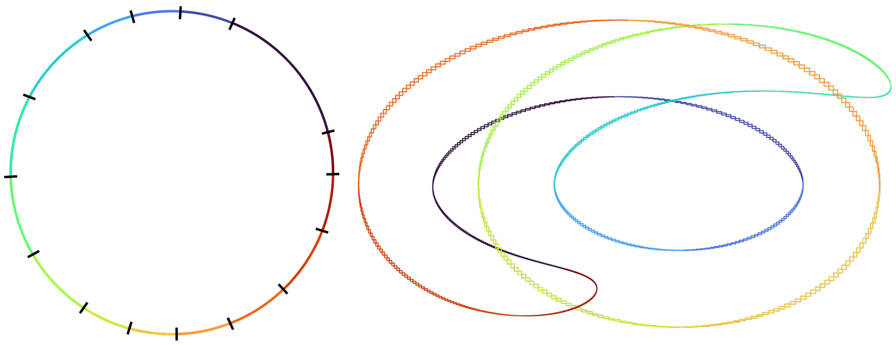


Figure 20: Fifteen Box Chains in $\mathcal{S}^1$ (left) and their image by $\mathbf{g}$ (right)

Thanks to this Box Chain decomposition, we were able to detect the self-intersections in the boundary as shown in red Figure 21.

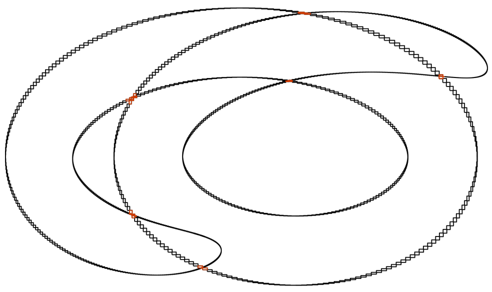


Figure 21: Self-intersections in the boundary (in red)

Finally, the fake boundaries are detected and removed before computing the inner and outer approximation of the image set without fake boundaries as displayed Figures 22a and 22b.

5 Conclusion

This paper presented the problematic of fake boundaries in the computation of the image of a set by a function. A method to remove them in the case of a function from $\mathcal{D}$ to $\mathbb{R}^2$ was presented. The method was finally applied with two examples where it was indeed able to remove the fake boundaries.

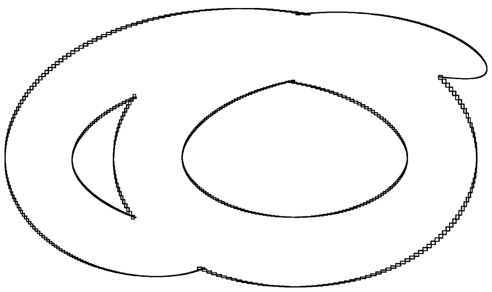
The notion of Box Chains was introduced with the definition of the neighborhood and the Box Chain relations. This Box Chain relation relies on an injectivity criterion and an example for the case of a function from $\mathbb{R}$ to $\mathbb{R}^2$ was proposed. These Box Chains can be used to make the detection of self-intersections in the boundary easier.

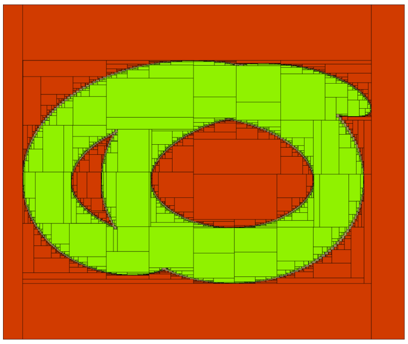
Finally a boundary simplification algorithm was displayed. It relies on Box Chains to facilitate the detection of self intersections in the contour. Under the assumptions presented in Section 4, this algorithm may be extended to higher dimension than displayed in this paper.

References

[1] Brateau, Q., Le Bars, F., and Jaulin, L. Considering adjacent sets for computing the visibility region. *Acta Cybernetica*, 2024.

[2] Costa Vianna, M., Goubault, E., Jaulin, L., and Putot, S. Estimating the coverage measure and the area explored by a line-sweep sensor on the plane. *International Journal of Approximate Reasoning*, 169:109162, 2024. DOI: https://doi.org/10.1016/j.ijar.2024.109162.


22(a) Fake boundaries removed


22(b) Inner (green) and outer (green + yellow) approximation of the image set

[3] Dabadie, C., Kaynama, S., and Tomlin, C. J. A practical reachability-based collision avoidance algorithm for sampled-data systems: Application to ground robots. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4161–4168, 2014. DOI: 10.1109/IROS.2014.6943149.

[4] Jaulin, L., Kieffer, M., Didrit, O., and Walter, E. *Applied Interval Analysis: With Examples in Parameter and State Estimation, Robust Control and Robotics*. Engineering Online Library. Springer-Verlag, London, 2001. DOI: 10.1007/978-1-4471-0249-6.

[5] Kieffer, M., Jaulin, L., Braems, I., and Walter, E. Guaranteed set computation with subpavings. In *SCAN-Interval 2000*, page pp. not specified, 2000.

[6] Malone, N., Chiang, H.-T., Lesser, K., Oishi, M., and Tapia, L. Hybrid dynamic moving obstacle avoidance using a stochastic reachable set-based potential field. *IEEE Transactions on Robotics*, 33(5):1124–1138, 2017. DOI: 10.1109/TRO.2017.2705034.

[7] Rego, B. S., Raffo, G. V., Scott, J. K., and Raimondo, D. M. Guaranteed methods based on constrained zonotopes for set-valued state estimation of nonlinear discrete-time systems. *Automatica*, 111:108614, 2020. DOI: https://doi.org/10.1016/j.automatica.2019.108614.

[8] Rohou, S., Desrochers, B., and Le Bars, F. The Codac library. *Acta Cybernetica*, 2024. DOI: 10.14232/actacyb.302772.

[9] Rohou, S., Jaulin, L., Mihaylova, L., Le Bars, F., and Veres, S. M. *Reliable Robot Localization: A Constraint-Programming Approach Over Dynamical Systems*. Wiley-ISTE, 1 edition, 2020.

[10] Welte, A., Jaulin, L., Ceberio, M., and Kreinovich, V. Avoiding fake boundaries in set interval computing. *Journal of Uncertain Systems*, 11(2):137–148, 2017.