Red: should be corrected
Green: positive comments

Border Propagation: A Novel Approach To Determining Slope Region Decompositions

Determine

Florian Bogner & Alexander Palmrich

1 Abstract

Generally, an abstract should not be an outline of your work ("In section .. we define..."), but summarize it including the most significant statements (usually in the same order as the structure of your work: introduction → methods → results → conclusion)

We introduce the *border propagation* (BP) algorithm, which decomposes a d-dimensional array of scalar values into slope regions. Slope regions as defined in [2] and their prerequisites are defined in the fist two sections. The algorithm is sketched in sections 4-5, linked to the Reeb Graph in section 6 and compared to other decompositions in section 7.

Please take a look at the commented code at our git repository![1]

Just a "style" issue, but exclamation points should be avoided in scientific formal writing

2 Motivating Slope Regions

I like the introducing parts to each section, the reader is prepared well for the following content + it gives a good overview

In this section we try to develop an intuitive understanding of the term *slope region* and its generalization to dimensions higher than 2. The concise definitions of the terms already employed here is reserved for the next section.

Consider an image, either gray-scale or in color. If it is a color image, it can be decomposed into its color channels (red-green-blue), which can individually be read as gray-scale images. We now think of the light intensity (i.e. the image value) of one such gray-scale image as the height of a landscape, yielding a surface in 3D space. The surface will have hills in areas where the image is bright, and will have dales in dark areas. Please refer to figure 1 for a visualization!

Again just style: If you refer to a specific figure → capital F Figure 1

Now our aim is to partition the surface into *regions* (i.e. subsets) in a particular way: We require each region to consist only of a single slope, by which we mean that we can ascend (or descend) from any given point of the region, to any other given point of the region, along a path that runs entirely within the region. Such a decomposition is not

[1] `https://github.com/SirFloIII/MustererkennungLVA`

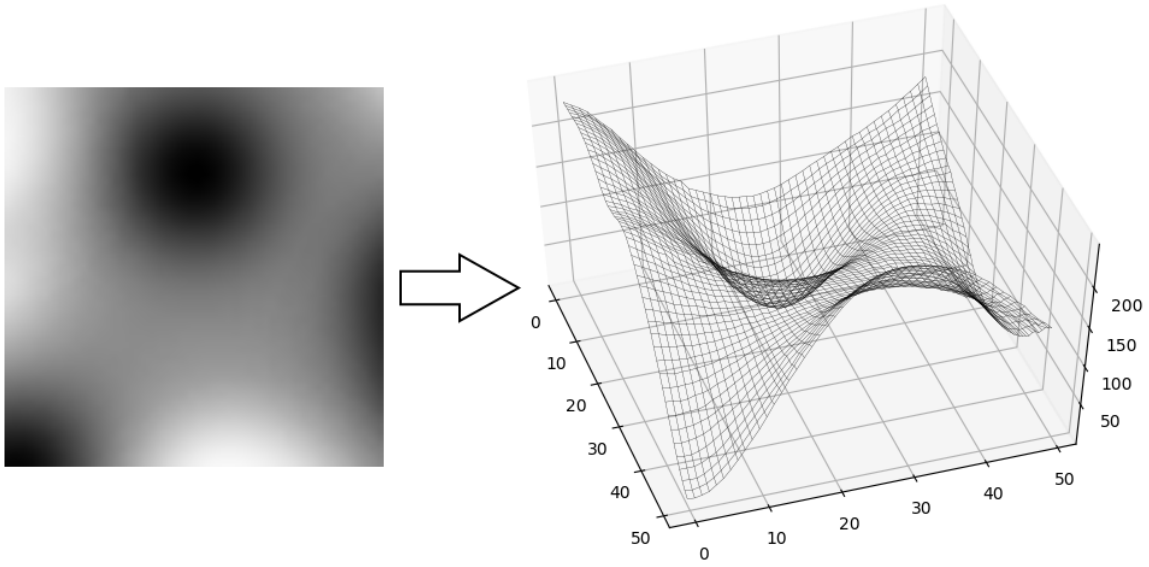

Fig. 1: Gray-scale to Height-map conversion

unique, but we can at least try to get a partition *as coarse as possible*, meaning that we merge slope regions if the resulting subset is still a slope region, and we iterate this until it can be done no more. There might be many different coarsest slope decompositions, but we accept any one of them.

find a different wording

The criterion we used to describe slopes, any two points being connected by either an ascending or a descending path, can easily be used in higher dimensions. Think of a computer tomography scan, which will yield gray-scale data, but not just on a 2D image, but rather on a 3D volume. Now, if you are inclined[2] to visualize a fourth spacial dimension, you can think of that fourth direction as the height of a 3D hyper-surface in 4D space, with the height encoded in the tomography scan. But imagining this is difficult and not required, because our slope region criterion works in any dimension. We want to partition the 3D volume, such that any two points in a region can be connected via an either ascending or descending path within the region. Recall that *ascending* and *descending* refers to the brightness value of the tomography scan as we move in the volume.

By abstracting from image and tomography to a real function defined on some subset of $\mathbb{R}^n$ (think of it as the brightness function!), and by rigorously defining a coarsest slope decomposition, we can lift the concept to arbitrary dimensions in a mathematically concise fashion.

3 Defining Slope Regions

Together with the description this part was easily understandable!

In this and the following chapters we will consider a topological space $(\Omega, \mathcal{T})$ with a continuous function $f : \Omega \to \mathbb{R}$. In practice or for ease of imagination, $(\Omega, \mathcal{T})$ will typically be a rectangle or cuboid subset of $\mathbb{R}^2$ or $\mathbb{R}^3$ equipped with the euclidean topology and f will describe an image or a 3D-Scan.

Definition 3.1. A *path* is a continuous function from the real interval $[a, b]$ (with $a < b$)

into a topological space Ω.

Definition 3.2. Two points $x \neq y$ in a topological space Ω are called *path-connected* if and only if there is a path $\gamma : [a, b] \to \Omega$ with $\gamma(a) = x$ and $\gamma(b) = y$.

Definition 3.3. The set of all points which are path-connected to a point $x \in \Omega$ is called the *(connected) component of* x:

$$[x] := \{y \in \Omega | x \text{ is path-connected to } y\}$$

Any subset of Ω which can be written in this way (for a suitable choice of x) is called a *(connected) component.*

Definition 3.4. A path $\gamma : [a, b] \to \Omega$ is called *monotonic* if and only if

$$\forall s, t \in [a, b] : s < t \Rightarrow f(\gamma(s)) \leq f(\gamma(t)) \vee$$
$$\forall s, t \in [a, b] : s < t \Rightarrow f(\gamma(s)) \geq f(\gamma(t))$$

Definition 3.5. Let $R \subset \Omega$. R is called *slope region* or *monotonically connected* if and only if for all $x, y \in R$ there exists a monotonic path $\gamma : [a, b] \to R$ with $\gamma(a) = x$ and $\gamma(b) = y$.

Definition 3.6. A family of sets $\{A_i \subset \Omega \mid i \in I\}$ is called a *slope region decomposition* if and only if:

- A_i is a slope region for all $i \in I$
- $\forall i, j \in I : i \neq j \Rightarrow A_i \cap A_j = \emptyset$.
- $\bigcup_{i \in I} A_i = \Omega$

Definition 3.7. Consider two slope region decompositions $\mathcal{A} = \{A_i \subset \Omega \mid i \in I\}$ and $\mathcal{B} = \{B_j \subset \Omega \mid j \in J\}$. We call $\mathcal{A}$ *coarser than* $\mathcal{B}$, in Symbols $\mathcal{A} \succeq \mathcal{B}$ if and only if

$$\forall j \in J \; \exists i \in I : B_j \subset A_i.$$

Theorem 3.8. $\succeq$ *is a partial order, i.e. fulfills reflexivity, antisymmetry and transitivity.*

Proof: Straight forward. Antisymmetry follows from the decomposition property. □

Definition 3.9. A slope region decomposition $\mathcal{A}$ is called *maximally coarse* or simply *coarse* if and only if there is no other coarser slope region decomposition.

Theorem 3.10. *Let $A \subset \Omega$ be a path-connected set. A is a slope region if and only if all levelsets of f in A are path-connected, i.e.*

$$\forall c \in \mathbb{R} : f^{-1}(c) \cap A \text{ is path-connected.}$$

Proof: "⇒" via contraposition:

Suppose there exists a $c \in \mathbb{R}$ with $L := f^{-1}(c) \cap A$ not path-connected. We decompose L in its components and pick x and y from different components. Since $f(x) = f(y) = c$ a monotonic path between x and y would have to lie completely in L. However, since x and y are from different components, they cannot be connected by a path in L and therefore cannot be connected with a monotonic path. Therefore, A is not a slope region.

"⇐" via *ironing out* an arbitrary path:

Given $x, y \in A$ we have to find a monotonic path γ. Without loss of generality suppose $f(x) \geq f(y)$. Since A is path-connected, there exists an (not necessarily monotonic) path $\gamma_0 : [a, b] \to A$ from x to y. Using the Rising Sun lemma [1] on the continuous function $f \circ \gamma_0$ we get the *shadow* $S = \bigcup_{i \in I}(a_n, b_n)$ consisting of at most countably many intervals.

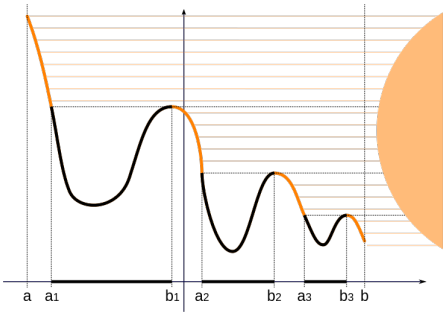

Fig. 2: Rising sun lemma

S consists of the points which contradict the monotony of $f \circ \gamma_0$, thus we want to *iron out* these points.

Let $c_n := f(a_n) = f(b_n)$. Since the levelset of c_n is path-connected, we can connect $\gamma_0(a_n)$ and $\gamma_0(b_n)$ with a level path $\gamma_n^* : [a_n, b_n] \to A$.

Finally, we define:

$$\gamma(\sigma) := \begin{cases} \gamma_n^*(\sigma) & \sigma \in [a_n, b_n] \\ \gamma_0(\sigma) & \text{elsewhere} \end{cases}$$

γ is a monotonic path from x to y, thus A is a slope region.

□

4 Motivating The Border Propagation (BP) Algorithm

Now we will work our way to the central insights on which the border propagation algorithm (~~abbreviated to~~ *BP*) hinges.

Slope regions can be constructed and grown in a straight-forward iterative manner by sweeping through the function values from lowest to highest. This is similar to the intuition employed in Morse theory [3]. Visualize a smooth, compact 2D surface in 3D space. We want to decompose this surface into slope regions. Initially, our decomposition is empty, i.e. there are no slope regions (thus we don't have an actual *decomposition* yet).

Imagine a water level rising from below the surface, up to the point of first contact. Starting at this global minimum, we add a new region, containing only the argmin (i.e. a single point on the 2D image where the minimal value is taken).

Now, there might be many points where the global minimum is taken. This will either be due to a flat region (*plateau*) on the surface, which we want to include into the single existing region, or it will be due to individual dales, which all have their lowest point at the same height. In this second case, we can't put the points into the existing region, because we would not be able to get from one argmin to another via a monotonic path. Instead, we need to add a new region for each individual dale.

Both cases can be dealt with by contracting connected points into their connected component, and creating a new region for each resulting component. This will ensure that plateaus are assigned to a single region.

As the water rises, we can add points to an existing region growing it upwards if they are just outside the region[2]. Otherwise they correspond to a distant local minimum and

[2] Why can we do that? By adding only points which are connected to the region we ensure path-

footnote continues on the next page

have to be dealt with as before, by opening a new region for each point (or rather, each connected component).

With the water rising further still, the regions will grow upwards to a point where they meet. Any such point is a saddle point, and we have to account for it the next time we want to grow any one of the touching regions. The saddle point connects the edges of the regions which meet in it, at the current height of the water level. It might be the case that the not-yet-assigned points (the ones above the water) get separated into multiple connected components, or they might remain connected.

If the points remain connected, then we have to decide for a single one of the involved regions to be allowed to grow upwards from the component. This means that one region effectively inherits the growth directions from the other region(s). The other region(s) lose their potential for expansion and remain frozen in their current state.

This part is quite difficult to understand when you just read it. I think it would be helpful if you include the corresponding subfigures on the right to your explanation

If the unassigned points have multiple components, then we may assign one component to each involved region. The regions will then grow only in the directions determined by the assigned components as the water rises. Any region without an assigned component remains frozen. Should there be more components than regions, then we open up a new region for each surplus component.

An oddity which can occur are self-loops: A region might grow into a "C"-shape, and then proceed to close up into an "O"-shape. This case can be treated similarly as above, the only difference is that the saddle is found by recognizing that the region collides with itself, instead of with another region.

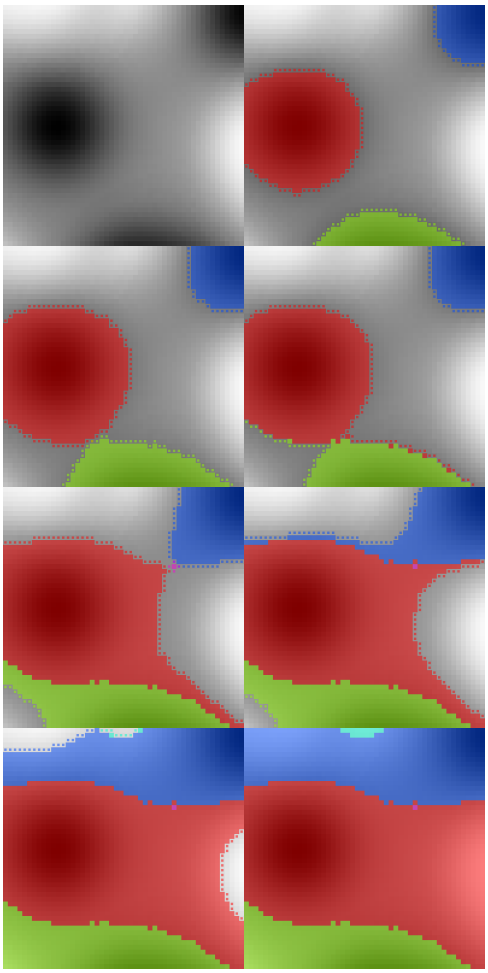

Fig. 3: Evolution of Regions during the algorithm.

Eventually this procedure will arrive at the global maximum, and the entire surface will be divided into regions. Since we proceeded with the necessary care and attention along the way, we ensured that the regions remained slope regions, and we also only created additional regions when we absolutely had to, showing that the resulting composition is maximally coarse.

The same algorithm can be applied in higher dimensions. We deal with iso-hypersurfaces as level sets, but the topological considerations about connectedness remain the same as in the illustrative 2D case.

connectedness, and by growing the region upwards, we can construct ascending paths from old points to new ones. The smoothness of the surface guarantees that while moving at a fixed height, we can reach all points of the region with that height.

5 Details About BP

The somewhat vague description of BP in the previous section assumed a continuous surface. In most applications, however, the data will be provided in a discrete raster format. Some intricacies arise from this discretization, most notably iso-surfaces of a smooth function f will not have a straight-forward representation in the discrete grid obtained from rasterizing f.

As a result, there are multiple ways to define what a *discrete slope region* is supposed to be. Let us first explore, why the most immediate choice for a definition is in fact a bad one. Using the discrete notion of connectedness, allowing discrete paths to move only axially (or at best diagonally) each step, one can define a slope region: R is a slope region iff any two points in R can be connected via a monotonic discrete path contained in R.

There is a simple example to demonstrate that this definition is a bad one: Imagine a plane which is at a slant, such that it isn't parallel to any of the axes. Clearly a sensible slope decomposition would assign the entire plane to a single region. But let's look at what happens after discretization. Due to the slant, the points in successive levelsets or even within a single one will not be adjacent to each other, but rather spaced out. Many regions will be generated, because these points are not considered connected, even though we know that on the continuous surface they are linked by a levelset.

It is precisely this realization that led the authors to the BP algorithm. Every region consists of the points already added to it, but the region also remembers its *halo*, which is a set of unassigned points directly (axially, not diagonally) adjacent to the region points. The halo can be seen as the direction into which the region is allowed to expand. This data structure models the continuous level set connectedness.

The search for the connected components of a level set in the continuous case corresponds to the search of connected components of the halo in the space of all unassigned points. This search is computationally expensive and has to be performed whenever a point is assigned to a region, which is why a very fast test is done first: Only if the halo is disconnected within a small local environment, do we jump into the global test.

Much like connected components need to be assigned to colliding regions in the continuous case, the same has to be done in the discrete world. BP achieves this by distributing halo components among the involved regions. The heuristic chosen by the authors to decide which region gets which component is as follows: The component which allows for expansion towards the biggest chunk of unassigned points gets assigned to the involved region with the lowest minimum. The component allowing for second largest expansion gets assigned to the involved region with the second lowest minimum, etc. The intent is to prefer the construction of large regions which reflect the topology well, and to freeze tiny regions, since they have a good chance of resulting from noise.

By iterating over the levelsets and swapping entire halo components, BP effectively ensures that the borders between regions are on the discretized equivalent of a continuous levelset. This property allows for an interesting characterization related to Reeb Graphs, as shown in the next section.

6 BP And Reeb Graph

Style: before you used lowercase → should be consistent

There is a significant connection between Reeb Graphs and Slope Region Decompositions.

Definition 6.1. Let $(\Omega, \mathcal{T})$ be a topological space, and $f : \Omega \to \mathbb{R}$ continuous. We define an equivalence relation for $x, y \in \Omega$:

$$x \sim y :\Leftrightarrow f(x) = f(y) = c \ \wedge \ x \text{ and } y \text{ are path-connected in } f^{-1}(c)$$

The *Reeb-Graph* of Ω is $G := \Omega/_\sim$

Example 6.2. Let Ω be a torus on its side and f be the distance from the ground. Its Reeb Graph is depicted in Figure 4.

In general, a Reeb Graph consists of vertices and lines between them. Vertices correspond to critical points of f. They are local extrema iff their rank is 1 and saddle points iff their rank is bigger than 2.

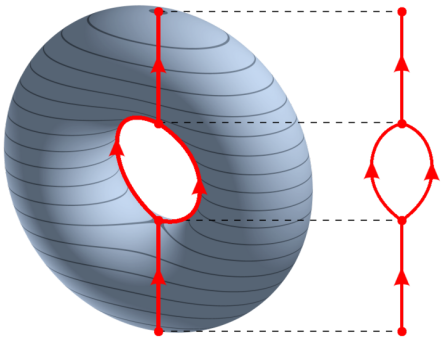

Fig. 4: Reeb Graph of Torus

Theorem 6.3. *Let S be a Slope Region. $S/_\sim$ may intersect no more than one point of G at a given height.*

Proof: *Follows from Theorem 3.10.* □

Conversely:

Theorem 6.4. *Let L be a connected linear subset of G. $S := \bigcup_{l \in L} l$ is a slope region with the additional property that its borders are along levelsets.*

Proof: *Also follows from Theorem 3.10.* □

Thus, decompositions of a Reeb Graph G into connected linear subsets of G are isomorphic to Slope Region Decompositions with borders along levelsets, which are exactly the kind of slope region decompositions which BP produces.

7 BP And Other Segmentation Methods

Are there proper citations? (Not Wikipedia, I was never allowed to include that in a paper) If you include links → footnote

- BP might be considered a level set method.
- Even though the algorithm was motivated using a rising water surface, it is indeed very different from the seemingly similar watershed algorithm both in its mechanics and resulting segmentation. Maybe you could add just a short statement about what's different (but if a long description is needed, then don't)
- Much like maximally stable extremal regions method, regions are grown around extrema in BP. But slope regions need not be extremal (in the sense that either all points in the region are above any point on the border, or all points are below), and neither is their broder maximally stable. In fact, borders of slope regions will change quite drastically by sliding them up or down, as they will change their topology by passing a saddle point.

wording

border

comma

8 Further Potential Development

The result of BP is satisfactory, but the authors assume that improvements can be made in running time. The code was profiled multiple times and has been adapted to run faster with significant gains in many instances.

Wording

Most of the CPU cycles get eaten by the search for connected components (as of writing this document). We have already put efficient preliminary tests into place in order to minimize the calls to these expensive searches. Yet two runtime improvements came to the authors' minds:

- Growing regions not by a random choice from their halo, but from a priority queue, with new halo points being added to the end of the queue. This will reduce cases where self-collisions of regions are triggered, which require a call to the expensive search in order to be handled correctly.

- Making the expensive search itself more efficient: Searching for components with the A* algortihm, and encouraging the search to move along iso-surfaces. This will terminate the search far earlier in cases where there is only one component to be found, potentially reducing the complexity by one order for such calls. This optimization is being implemented as of writing.

Other additional features that the authors considered are :

- An animated render of slope regions being build, for 3D data, showing different regions as point clouds of different colors.

- Generating the Reeb Graph from the slope region decomposition.

- Providing a *tolerance* parameter, which governs how steep a continuous function might get, before an iso-surface is deemed disconnected in the discrete data. This would allow for a trade-off between connectedness which reflects the continuous case, but might yield fragmented regions on the discrete data, and a connectedness which is more geared towards the discrete concept of connectedness, at the cost of an increased number of regions.

Please split this sentence

- Using established data structures that model smooth level sets from discrete data. There might be performance gains in employing such a data structure.

Another interesting experiment would involve generating the Reeb Graph via an external library, dissecting it to generate a slope region decomposition, and then pitting the resulting algorithm against BP. Let the games begin!

9 Authors' Individual Contributions

The BP algorithm as well as this paper are the result of a pattern recognition course held at the Vienna University Of Technology from Oct 2019 till Jan 2020. Professor Kropatsch introduced us to the concept of slope regions and posed the challenge to compute them in high dimensions (> 2). Initially both authors were tasked with small building blocks for this endeavor:

Alex Palmrich was to develop and implement a recognizer for saddle points, for which he coded a framework to perform efficient local regression of arbitrary degree on grid-sampled functions in arbitrary dimension. The gradient of the regression function was used to determine singular points, while the eigenvalues λ_i of the hessian matrix provided a classification of singular points into minima (all $\lambda_i > 0$), valleys (some $\lambda_i = 0$, the rest positive), degenerate points (all $\lambda_i = 0$), saddles (some λ_i positive, some negative), ridges (some $\lambda_i = 0$, the rest negative) and maxima (all $\lambda_i < 0$). This approach proved difficult to use in practice, as there were many parameters to choose, and the recognizer proved very sensitive to noise despite the smoothing resulting from regression over a big local environment. As such this fairly finished project was dropped in favor of developing BP.

wording

Florian Bogner was designing an algorithm similar to BP that assumed information about critical points as a given. It was supposed to use Palmrich's recognizer in practice. However, as the recognizer wasn't perfect, it never reached satisfactory results and was also shelved, and energy was directed to the joint development of BP.

Most of the theory behind BP as well as the actual code was conceived in direct interaction between both authors. Only some tidbits might be attributed to a single one of them, such as Bogner's idea to swap halo components between regions, his proof of theorem 3.10, or his code to generate an interactive display of the algorithm at work using pygame. Palmrich can be credited with the use of the halo data structure, the idea of moving along iso-surfaces for a speedup during the search of connected components, and the *Motivating ...* chapters in this paper.

References

[1] Karl Grill. Maß- und Wahrscheinlichkeitstheorie, 2018.

[2] Walter G Kropatsch, Rocio M Casablanca, Darshan Batavia, and Rocio Gonzalez-Diaz. On the space between critical points. In *International Conference on Discrete Geometry for Computer Imagery*, pages 115–126. Springer, 2019.

[3] Yukio Matsumoto. *An introduction to Morse theory*. Iwanami series in modern mathematics. American Math. Soc., Providence, RI, 2002.

Nice work!! I like your way of writing and explaining things also by including visual examples!