Saarland University
Faculty of Natural Sciences and Technology I
Department of Computer Science

Masterthesis

Clustering with Reaction-Diffusion equations

Mykola Byelytskyy

supervised by
Prof. Dr. Joachim Weickert

Reviewers:
Prof. Dr. Joachim Weickert
Dr. Matthias Augustin

May 5, 2017

Abstract

In this work I propose a diffusion-reaction equation based model for semi-automatic image clustering. Development of this model was inspired from the biology. The model uses the same principles as competitive Lottka-Volterra equation system [6], which describes in population dynamic a struggle between different species for restricted resources.

The proposed model consists of two parts: the diffusion part and the reaction part. Main purpose of the diffusion part is a transport function. The reaction part is used as a segmentation mechanic.

Interaction between the diffusion and reaction parts plays the main role providing a possibility to control the result using different initial configurations. This interaction is a central point of the proposed thesis.

The big advantage of proposed method is its interactivity and a possibility to use different "physical" processes for different feature maps. The main disadvantage is its complexity and significant amount of parameters, that nevertheless can be adapted for a particular image domain.

Acknowledgements

I want to thank Prof. Dr. Joachim Weickert for providing me with really interesting topic and the degree of freedom that I had during my work on it.

I also thank Dr. Matthias Augustin for his kind agreement to be the second reviewer of this thesis.

Contents

List of Figures

1 Introduction.

1.1 Image segmentation: motivation and applications. Important properties.

"Image segmentation" is a very important part of many visual computing algorithms. Computer vision, medical image analysis, automatic quality control, text recognition etc. need segmentation as a pre-processing part of an algorithm to locate their objects of interest for further analysis. Thus, the quality of segmentation often affects the quality of the whole algorithm. This is the reason why it is important to have a good and fast segmentation algorithm that passes to the particular visual computing area.

But what means a "good" segmentation algorithm? Which criterias define the quality of image segmentation? It depends on particular image domain and particular goal of segmentation.

For example, for pedestrian detection in a self-driving car robustness against varying light conditions and noise like rain or fog as well as correct detection / separation of partially hidden objects would be very important. The segmentation algorithm should be adapted to such conditions. Furthermore, the algorithm should be full automatic and fast enough to be used in real-time conditions.

On the other hand in some medical application, like tumor detection on ultrasound images the lightning conditions are stable, the algorithm should not be used in real time and could be semi-automatic (operator can mark a region where the tumor is located). But the algorithm should be robust against noise (ultrasound images are normally very noisy) and be accurate enough to make possible a correct measurement of the tumor properties. In this case the criteria of a good segmentation algorithm would be different from those in the first case.

So as we seen with these two examples for different conditions the algorithms should have different characteristics. That's why there are no universal segmentation technique. For each problem, that requires a segmentation, a suitable algorithm should be chosen and sometimes also adapted for the specific conditions. This was my motivation before the development of the proposed model. I was trying to make it flexible and adaptable on working conditions.

The next part contains examples of some basic and modern segmentation techniques with different characteristics and different application area.

1.2 Examples of image segmentation techniques

As was said in the previous section, there are many different image segmentations algorithms that could be good in some conditions but can be useless in other situations. Despite the simplicity of some algorithms described below they could be nevertheless useful for some application areas.

Thresholding algorithms.

This class of algorithms uses a threshold to decide to which segment a point belongs. The simplest example: binarization based on grey value thresholding. This algorithm compares grey value of each pixel with some threshold value and depending on if it is bigger than the threshold or not decides to which segment this pixel belongs. The advantage of such algorithm is its simplicity. Disadvantage - it does not take into account the spatial context, thus the result is normally very poor for complex images. The other drawback is the necessity of choosing a good threshold value. There are few algorithms that can do it automatically (Otsu's method, quantile method etc.) but this does not solve the general inaccuracy problem of the algorithm.

Region growing / region merging methods.

The main idea of this class of algorithms is to merge a point with neighbour points based on some threshold or other criteria. Region growing is semi-automatic and needs a seed within the object, that has to be separated from background. The region merging does not need any seeds and it works automatically in this sense (just merge the neighbour pixels till it is possible). These algorithms use spatial context, what increases their basic accuracy. Region growing method is interactive and this makes it attractive in some applications.

But, this set of algorithms can be used for very specific applications only. In general the quality of segmentation with this algorithms is relative poor.

Active contour model based algorithms.

This set of image segmentation techniques uses common framework called active contour model. Intuitively, the active contour is a curve that can move across the image and change its form according to some sum of internal and external forces. Internal forces resist the contour deformation and external forces attract it towards some object boundary. The resulting balance of the forces defines the segmentation.

This is a popular state-of-the-art model. We will use one of its representatives [4] for comparison with the model proposed in this work.

1.3 Organisation

The proposed thesis organised as follows. I will start with description of my initial task and basic steps of the proposed model development in Section 2. Section 3 introduces the whole model and contains deeper analysis of different theoretical and practical aspects of the proposed approach. Section 4 describes the main implementation steps of the algorithm. Section 5 compares the proposed approach with Chan-Vese segmentation. Some practical results of the algorithm application are shown in Section 6. Section 7 contains conclusion and description of future work.

2 Reaction-diffusion equations. Ideas from biology.

2.1 Initial task.

My initial task was to develop an image clustering algorithm using some sort of struggle between different phases. As an example the following system of diffusion equations was proposed:

$$\frac{\partial u_1}{\partial t} = \operatorname{div}(D_1(u_1, u_2) \cdot \operatorname{grad}(u_1))$$
$$\frac{\partial u_2}{\partial t} = \operatorname{div}(D_2(u_1, u_2) \cdot \operatorname{grad}(u_2))$$

where

$$D_1 = \begin{cases} 1 & \text{if } u_1 > u_2 \\ 0 & \text{otherwise} \end{cases}$$

$$D_2 = \begin{cases} 1 & \text{if } u_1 < u_2 \\ 0 & \text{otherwise} \end{cases}$$

But in the case of variable diffusion tensor we have to deal with some sort of "negative diffusion", if we need to crowd out one phase from a region, where the other phase should be placed (if the wrong phase "arrived" before the right phase). Such negative diffusion could lead to singularities.

Because of this reason I was seeking some mechanics that would make possible for a phase to suppress other phases in a region, where it should be placed. Since I wanted to use diffusion process as a transport, one way to achieve my goal was to use a "reaction" function f. In this case the diffusion process becomes non-homogeneous.

$$\frac{\partial u}{\partial t} = \operatorname{div}(D_1(u) \cdot \operatorname{grad}(u)) + f(u, x, y)$$

The next step was to find some flexible but intuitive model, that from one side would have enough degree of freedom to play with and from the other side would correspond to the main idea of my thesis. After some searching I came up with a diffusion-reaction models that describe for example dynamics of biological systems consisting of different species struggling for restricted resources or supporting each other.

My intuition behind using such models can be described as follows. We can imagine our struggling phases as populations of different species. Image features could represent different environment conditions for these species such as amount of food. For one sort of these species the "best" food will correspond to the image feature of an object and it should try to occupy and defend this region (it tends not to move in a "foreign" region). For the other species the best food will be the background. But since backgrounds could be very complex it should also try to occupy new territories (try to move in a "foreign region"). During the struggle one species should occupy and defend the object while the other species should occupy the background and try to "attack" the object. As a result of this struggle we should have some clustering that could separate the object from the background. The "defender" should be strong and fast within the object area but weak and slow outside of the object. The attacker should have more homogeneous distribution of "attacking" force and be relative fast but should lose within the object area.

Available resources in each point should be limited. This is very important because, if we have some complex non-homogeneous object we can normally choose only its dominated feature for the defender. But the object could have in this case regions where the feature value is small. If the defender will occupy most outer parts of such object this will prevent the attacker due to limited resources from infiltration in the object area and let the defender slowly occupy the regions with weaker feature value.

2.2 Competitive Lotka-Volterra equations with anisotropic diffusion term.

By seeking a suitable model my focus was on the following properties: limited resources, possibility to model the attacker-defender pattern.

For this purposes I find the competitive Lotka-Volterra model (for more information about the model see [6]) applicable, since it describes the struggle between different species for limited resources and simple enough to work with.

Basically the model consists of time derivative and logistic term:

$$\frac{\partial u_1}{\partial t} = \gamma_1 \cdot u_1 \cdot (1 - \alpha_{1,1} \cdot u_1 - \alpha_{1,2} \cdot u_2)$$
$$\frac{\partial u_2}{\partial t} = \gamma_2 \cdot u_2 \cdot (1 - \alpha_{2,1} \cdot u_1 - \alpha_{2,2} \cdot u_2)$$

Here the γ_i is a growth rate for species i and $\alpha_{i,j}$ is an element of community matrix. Coefficients of this matrix reflect influence of a species j on a species i. This influence can be, depending on the sign of the coefficient, positive or negative . By positive influence one species supports the other species in its growth. By negative influence we will have a suppression instead of support.

For better understanding of the intuition behind these coefficients, we can consider one isolated specie.

$$\frac{\partial u}{\partial t} = \gamma \cdot u \cdot (1 - \alpha \cdot u)$$

In this case the coefficient α represents the capacity of environment. It defines the maximum possible amount of species in a point (as soon as $\alpha \cdot u = 1$, $\frac{\partial u}{\partial t} = 0$ and there is no growth in this point).

The only dynamic in this equation is the struggle between species in each point without migration in space. But since we need a transport, a diffusion term that corresponds to movement of the species in space should be added.

$$\frac{\partial u_i}{\partial t} = \mathrm{div}(D_i \cdot \mathrm{grad}(u_i)) + \gamma_i \cdot u_i \cdot (1 - A \cdot u) \tag{1}$$

where A is the community matrix. Now the species can move in space according to diffusion tensor D. The equation (1) is a diffusion-reaction equation and $\gamma_i \cdot u_i \cdot (1 - A \cdot u)$ is the reaction term.

2.3 Geometrical interpretation

We have now two different processes that influence spreading of our phases. Diffusion controls the movement of the phases in space and the reaction term defines the interaction between them in each point. To decide how to build the diffusion tensor and how to define the community matrix I will use the following intuition. Main purpose of diffusion in the equation system is the transport function. It should bring the phase in the region, that could belong to some segment (according to distribution of an image feature) and keep it in this region. The main goal of the reaction part is to "take the final decision" in labeling of the region.

So intuitively the diffusion tensor should be constructed in the way, that it gains the mobility in the regions with bigger feature value and suppresses it in other regions. If there should be some sort of anisotropy, it have to direct the phase along the object borders but not across them. Correspondent image feature should locate the object for the defender phase and should provide the possibility to spread over the background for the attacker phase. This could be for example some texture or shape feature.

For the reaction part it would make sense to use a feature that has high contrast to allow for precise detection of the object's borders but maybe lower homogeneity and localisation (there are could be elements of background with the same feature value). If the diffusion part takes care of object locating, the reaction part should provide a good and contrast separation of the object from background. Such image features like color difference could be used for this purpose.

In the rest part of this subsection I would like to make a closer look on how the diffusion and reaction terms work together and how do they influence the final result.

Without any reaction term and non-zero diffusivity the diffusion process will act like this. Let's say we have some amount of a phase in some point of otherwise empty space. After starting the diffusion process this phase will dissipate in surrounding space according to the diffusion tensor. Since the total amount of phase stays constant (we have no reaction term) this phase will be after some time spreaded around with much lower density.

The reaction term is a sort of source or think (depending on local conditions) for phases in each point. If the community matrix (and diffusion tensor) is equal zero, then

$$\frac{\partial u_i}{\partial t} = \gamma_i \cdot u_i \tag{2}$$

the velocity of phase increasing/decreasing is proportional to the phase amount. In this case we will have just a classical exponential growth. Let's now add the diagonal components of the community matrix:

$$\frac{\partial u_i}{\partial t} = \gamma_i \cdot u_i \cdot (1 - u_i) \tag{3}$$

In this case, when the value of the phase becomes equal 1 the function stops growing in this point. The time derivative is equal zero so there are no further changes in this point. But if we now add a non-zero diffusion term, we can have two different situations: either because of a positive flux from outside the equilibrium will be shifted towards bigger value (> 1) of u_i at this point or vice versa - towards smaller value. In the first case the time derivative will be negative and the function starts to decrease, in the second case the time derivative will be positive and the function will grow again. If the diffusion process is constant in time then, depending on how strong the inflow or outflow is, the value of u_i will stabilize itself in some point greater or smaller than 1, correspondingly. The stabilisation condition in each point:

$$-\operatorname{div}(D_i \cdot \operatorname{grad}(u_i)) = \gamma_i \cdot u_i \cdot (1 - u_i) \tag{4}$$

Let's now investigate the influence of the community matrix coefficients. If we would have two phases without diffusion and all diagonal components of the community matrix are equal 1, then:

$$\begin{aligned}\frac{\partial u_1}{\partial t} &= \gamma_1 \cdot u_1 \cdot (1 - u_1 - a_{1,2} \cdot u_2) \\ \frac{\partial u_2}{\partial t} &= \gamma_2 \cdot u_2 \cdot (1 - a_{2,1} \cdot u_1 - u_2)\end{aligned} \tag{5}$$

the static solution (excluding trivial solutions) would be for $a_{1,2} \cdot a_{2,1} \neq 1$:

$$\begin{aligned}u_1 &= \frac{a_{1,2} - 1}{a_{1,2} \cdot a_{2,1} - 1} \\ u_2 &= \frac{a_{2,1} - 1}{a_{1,2} \cdot a_{2,1} - 1}\end{aligned} \tag{6}$$

It is clear, that for $0 < a_{1,2} < 1,\ 0 < a_{2,1} < 1$ we have the following conditions:

$$\begin{aligned}a_{1,2} > a_{2,1} &=> u_1 < u_2 \\ a_{1,2} < a_{2,1} &=> u_1 > u_2 \\ a_{1,2} = a_{2,1} &=> u_1 = u_2\end{aligned} \tag{7}$$

So as was mentioned above the coefficients of the community matrix control interaction between different phases. Without any diffusion the phase that has stronger negative influence on the other phase will have bigger value in this particular point. If the negative influence of a phase is equal 1 then the other phase will be completely exterminated in this region. The influence of the other phase could not be equal 1 at the same time in the same region.

But a diffusion process could shift the equilibrium. I'll try to show on a simple example how this change can look like.

Let's say we have a static state (all processes are in equilibrium) in the presence of diffusion for the 1st phase. This state can be described with the following equation system:

$$\begin{aligned}-\mathrm{div}(D_1 \cdot \mathrm{grad}(u_1)) &= \gamma_1 \cdot u_1 \cdot (1 - u_1 - a_{1,2} \cdot u_2) \\ 0 &= 1 - a_{2,1} \cdot u_1 - u_2\end{aligned} \tag{8}$$

In this case the diffusion should be constant in time $(\mathrm{div}(D_i \cdot \mathrm{grad}(u_i)) = d(x_0, y_0)$. The solution in each point should satisfy:

$$\begin{aligned} -d(x_0,y_0) - \gamma_1 \cdot u_1(x_0,y_0) \cdot (1 - u_1(x_0,y_0) - a_{1,2} \cdot u_2(x_0,y_0)) = 0 \\ 1 - a_{2,1} \cdot u_1(x_0,y_0) - u_2(x_0,y_0) = 0 \end{aligned} \tag{9}$$

If we solve this algebraic system, we will get the following expression:

$$\begin{aligned} D = \sqrt{4 \cdot d(x_0,y_0) \cdot (1 - a_{1,2} \cdot a_{2,1}) + \gamma_1 \cdot (a_{1,2}^2 - 2 \cdot a_{1,2} + 1)} \\ u_1(x_0,y_0) = \frac{\sqrt{\gamma_1} \cdot (1 - a_{1,2}) + D}{2 \cdot \sqrt{\gamma_1}(1 - a_{1,2} \cdot a_{2,1})} \\ u_2(x_0,y_0) = \frac{\sqrt{\gamma_1} \cdot (2 - a_{2,1} - a_{1,2} \cdot a_{2,1}) - a_{2,1} \cdot D}{2 \cdot \sqrt{\gamma_1}(1 - a_{1,2} \cdot a_{2,1})} \end{aligned} \tag{10}$$

With increasing of $d(x_0, y_0)$ the value of D increases and consequently u_1 increases and u_2 decreases. If $d(x_0, y_0)$ decreases, we will get the opposite case.

So the positive diffusion shifts the equilibrium towards the increasing of the corresponding phase amount. This can compensate the suppression due to negative influence of the other phase.

We can look at this process from the "biological" point of view. If our phases correspond to some species then we can say that the bigger negative influence of a specie B to the specie A corresponds to faster "eating" of the other species resources (the resources are limited with 1 for each species). Or, in other words, the specie B eats more resources of specie A than the specie A eats from the resources of specie B. If now we have an extra flow of species A from outside, it will eat more resources of species B which will decrease the amount of specie B, which will decrease the amount of resources that specie B eats at specie A. So the total amount of specie A will be bigger.

It follows from the eq. (6) that if one specie has the negative influence equal 1 and the negative influence of the other specie is smaller than one, then the first specie will totally eliminate the second specie. But this can be changed due to diffusion.

I will try now to illustrate with a few characteristic examples how the processes described above could be used in sense of image segmentation.

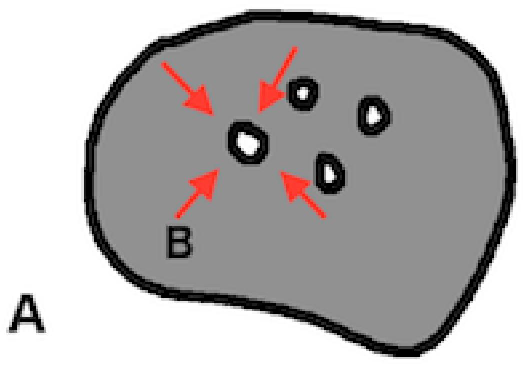


Figure 1: A small region within bigger region.

Let's say we have two regions A and B (Fig 1). Inside the region A the phase A has strong negative influence on the phase B and as a result the phase A dominates in this region. In the region B the phase B has advantage and dominates there. Further, the region B contains small clusters, where phase A has advantage and would dominate without diffusion. But if the diffusion strong enough, these small clusters could be nevertheless occupied by the phase B due to positive flux of the phase B in these small regions.

On the next figure (Fig. 2) an object with complex border is shown. We will focus now on the two thin elements of the border.

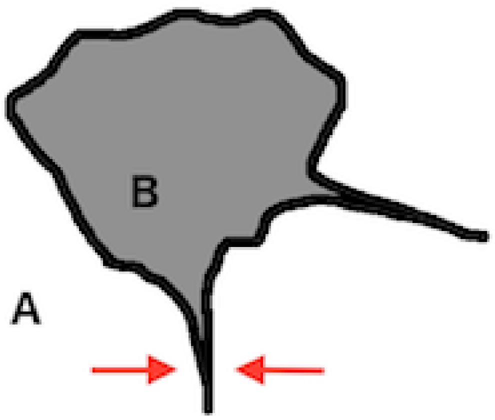


Figure 2: A complex border with thin elements.

The region B should be completely occupied by the phase B. But because of diffusion from the region A the solution could be partially shifted towards phase A within these thin regions. It would look like a smoothing of fine border elements.

This problem could be partially solved by using strong anisotropic diffusion along the borders and suppressed diffusion across the border. But even in this case some very thin parts could be occupied by the "wrong" phase.

The next example shows a sort of a "barrier" (see Fig 3), where negative influence of phase B on the other phase is equal 1 (6). Let's say phase A starts in the blue point and phase B in the red point. In this case if the phase B occupies the marked region before the phase A will reach it, the inner region will be closed for the phase A and will be slowly occupied by the phase B. The phase A will be strongly suppressed when it tries to come across the region B .

Presented example illustrates strong dependancy of the solution dynamic on the start conditions. This allows us to control the diffusion processes manipulating starting conditions.

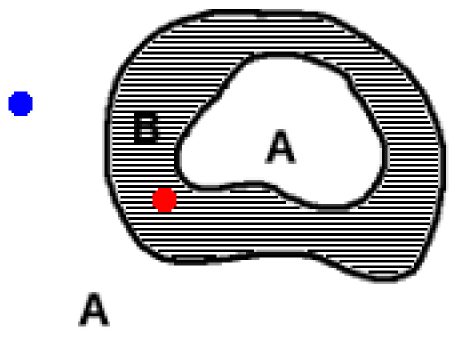


Figure 3: Barrier. Corresponded phase has the negative influence equal to 1 in the region B.

.

2.4 Potential advantages and disadvantages of the method

Potential advantages and disadvantages of the proposed model follow from the properties described above.

Generally the semi automatic clustering requires that the user marks some parts of the object and/or the background to initialize the process. In our case these marked parts should contain the characteristic elements for the used features to successfully locate and separate the object from background. We will take "probes" of this features in seed points defined by the user. The average value of a corresponding feature will be calculated in each seed point and the nomalized difference between the calculated average and the value of this feature in other regions will be considered as a feature map for corresponding seed point. We can then merge this feature maps for a set of object seeds and a set of background seeds, correspondingly. If the chosen probes contain truly characteristic feature for an object, then in the object region the difference feature map value would be lower than in other regions.

In all examples shown in this work the resulted difference feature maps are inverted for the purposes of intuitivity (greater values correspond to stronger similarity between the feature probe and other regions of the image).

So the first main point of the algorithm is its interactivity. The evolution of both phases depends on the starting points and chosen feature probes. The user can identify in this way object of interest and mark the background. If the object contains some parts that have the same feature values as the background, it could be overcome due to special configuration of seed points (see Fig. 3).

The next point is a smoothing of image feature inhomogeneity inside the object region. Natural images have normally variations of feature values all over the image. A simple thresholding methods would have problems with this inhomogeneities inside the object and could create gaps inside the object region. Proposed algorithm due to diffusion effects can compensate these undesirable effects by itself.

One more advantage is the possibility to use different features for the diffusion and for the reaction terms. Let's say we have some very contrast but not very specific feature. Some bright yellow textured object on the green background, for example. If background does not contain yellow or nearby colors, the color itself can be used for both diffusion and reaction parts as a feature. But if we have some yellow segments that belong to the background, it could be a problem to use only color difference as a feature. The algorithm can not distinguish between the object and the parts of background and the only possibility to solve this problem would be some specific selection of seed points. But if the object has for example a unique texture , we can use some texture related feature for diffusion and the color difference for the reaction term. Even if this texture feature is very blurred and do not allow us to accurately define the object boundary, this can still be compensated due to contrast color difference feature for the reaction part.

The time itself can be also considered as an interactivity tool, because depending on the seed points selection the final result can change with time significantly. The user can decide when he wants to stop the process.

First disadvantage is the "blurring" of thin border elements due to diffusion (see 2). But it could be partially solved with anisotropic diffusion.
The main idea of using anisotropic diffusion can be described as following: the thin region could be occupied by wrong phase due to strong diffusion of this phase from outside across the region boundaries. So we should try to minimize this diffusion and gain the inner diffusion of the right phase. The ideas of edge enhancing diffusion can be used for this purpose (for more detailed analysis see Sec. 3.5)

One more drawback is the large amount of parameters and complexity of the model. But some of the parameters can be adapted for particular image domain and simplify application of the model.

Undefined stopping time can be also considered as disadvantage in case if we have to show the result without showing any intermediate dynamic.

3 Segmentation with reaction-diffusion equation.

3.1 Attacker and defender.

As was mentioned before the application of the algorithm will be restricted to separation of an object or objects from the background. Therefore, I will have two types of "species": the attacker and the defender. The main difference between them: the defender is stronger in some restricted interval of image feature and weaker in other parts. The suppression abilities of the attacker are more homogeneous and just proportional to corresponding image feature.

The main purpose of the defender to "defend" the object region whereas the attacker should try to occupy the background and try to occupy the object.

The defender should basically spread within the object, the attacker should move across the background.

The defender seed points should be placed inside the object contour and the attacker on the background. Both seed sets should provide the correspondent feature probes.

3.2 Used features.

For my experiments and to demonstrate the main points of the algorithm I have used the set of features described below. This set can be extended or improved, it is not a constant part of the algorithm.

An example of such improvement could be used here implementation of Delta E. There are more modern and better working versions of this algorithm. But they are also more complex. I have used the simplest version of the feature to concentrate myself on the model itself and not on the used features.

Depending on image domain completely different feature set could be chosen.

Delta E

This feature tries to describe difference between color perception. As was mentioned above there are many implementations of Delta E, some of them (CIEDE2000 [12] for example) are more adapted to the human visual system features than the others. For the purposes of this work I will take the simplest implementation of the feature: euclidean distance in the LAB space ([13]).

$$f(x,y) = \sqrt{(L(x,y) - L_p(x,y))^2 + (a(x,y) - a_p(x,y))^2 + (b(x,y) - b_p(x,y))^2}$$

where $L,\ a,\ b$ are components of the LAB Space, $f(x, y)$ is the value of Delta E.

This feature map and all other feature maps in this work are calculated as follows. There are two types of seed points: object seeds and background seeds. Each seed point is a square with a particular size. For each seed the difference between mean value of the feature in the seed point area and all other points of the image is calculated. This difference is then normalized to the interval [0..1] and inverted to have maximum values in the seed points.

For many natural images the Delta E is not very homogeneous inside the object and can be not very specific for the whole image. Despite this fact in some of my examples it was used for the both diffusion and reaction term and gave a good results. But generally it should be used for the reaction term because it is normally very contrast.

As mentioned above, if we have more then one point for an object or background, their feature maps can be merged. In my experiments the merged image feature was just the maximum of all sub maps.

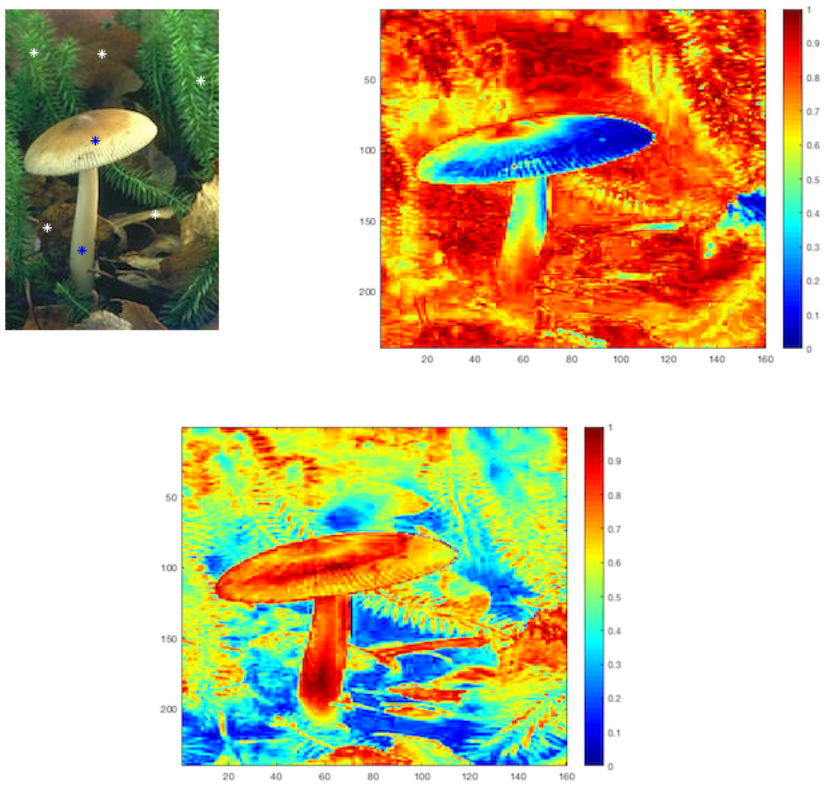


Figure 4: Merged feature maps. From left to right: original image with seed points, background feature map, object feature map.

On the Fig. 4 an example of used Delta E version is shown. There are two "object" points (blue stars inside the mushroom) and five "background" points (white stars). This points mark the characteristic (in the sense of colour) positions for the object and the background. Then the difference between average value within the seed point and all other points of the image is calculated and all object and background feature maps are reduced to two merged feature maps as described above. The resulting feature maps do not clearly mark the object and the background. As we can see the object feature map (bottom image) besides of the object itself contains a lot of small regions, that has the same intensity as the object parts. Sure, we can try to improve the Delta E feature to be more discriminative but this will not help much in general case because in natural images the colour itself often is not very discriminative. A scene could contain parts of background that has the same colour as the object.

Fourier amplitudes.

I will use this feature to analyse simple textures. The transformation will be calculated locally using a window function.

2-D Fourier transformation converts an image from spatial domain to the frequency domain ([18]). Discrete 2D version of the transformation:

$$\widehat{f}(p,q) = \frac{1}{\sqrt{N \cdot M}} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f(m,n) \cdot e^{\frac{-i \cdot 2\pi \cdot p \cdot m}{M}} \cdot e^{\frac{-i \cdot 2\pi \cdot q \cdot n}{N}} \qquad (11)$$
$$(p = 0, ..., M-1; q = 0, ..., N-1)$$

And the corresponding inverse transformation:

$$f(n,m) = \frac{1}{\sqrt{N \cdot M}} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} \widehat{f}(p,q) \cdot e^{\frac{i \cdot 2\pi \cdot p \cdot m}{M}} \cdot e^{\frac{i \cdot 2\pi \cdot q \cdot n}{N}} \qquad (12)$$
$$(m = 0, ..., M-1; n = 0, ..., N-1)$$

As a result of Fourier transformation we will get information about frequencies of which our image consists and their orientation. An example of discrete Fourier transformation is shown on Fig. 5

Using this information we can build image features for analysis and detection of the textures. In this work I use simple natural images like a picture of zebra on homogeneous background. For such contrast textures we can try to use some basic texture features without complex analysis.

In the paper of M. E. Jerniag and F. D'astous [14] regional entropy in the spatial frequency domain as a texture feature is proposed.

$$h' = -\sum_{p=0}^{M-1} \sum_{q=0}^{N-1} P_{p,q} \cdot log(P_{p,q})$$

where $P_{p,q}$ is the probability of a particular frequency (p, q) in the Fourier transformation. The authors propose [14] also to normalize the entropy by the sub image size. I will follow them and use the normalized version to compensate different size of sub images (in my case in the border regions). The normalized version will have the following form:

$$h'_n = \frac{h'}{h'_{max}} = \frac{h'}{log(K)}$$

where K is the amount of the frequencies within the sub image.

The main idea of using the entropy can be interpreted as follows: the entropy shows how unordered the frequencies in the Fourier domain are. The entropy value is maximal if there is no order at all, all frequencies are equally distributed. The entropy is zero if we have only one frequency in the domain - the case of total order. Textures have normally repetitive elements an we intuitively assume that this will give more ordered frequency domain. Background is often less ordered - so it should have bigger entropy (surely the background can also have a sort of texture, but in my experiments the entropy difference was strong enough).

Fig. 5 shows an example of described feature calculation. We have two samples of zebra texture (blue dots on the original image and corresponding the first two pictures in the second row) and two samples of the background. In the third row - corresponding Fourier transformations are shown. It is clear, that the zebra texture is more ordered and will give smaller entropy than the background.

On the Fig.6 middle the result of the complete feature map calculation for the object seeds is shown. This is inverted difference between feature value in seed points and other points of the image. The object is highlighted quite well but the top part of image contains the same difference value as the object itself. This happens because the background is often blurred, what acts as a low-pass filter and decreases the entropy. So the blurred background can have the same amount of entropy as a texture.

To avoid this problem I have added one more feature: the sum of all frequencies in the local foruier transformation (excluding the overall grey level). This feature correlates with the contrast of corresponding sub image.

Both features are normalized and shifted with respect to they variance. The final feature is the euclidean norm of the 2-component feature vector. The whole feature map is the merged difference between averaged feature vector in seed points and all other points of the image.

Fig. 6 illustrates the written above. The most left image is the complete feature map (2-component vector of differences with the object seed points) for the original image shown on the Fig. 5. The second sub image demostrates the difference map of the entropy component. The last sub image - is the difference map of "contrast" component of the feature vector.

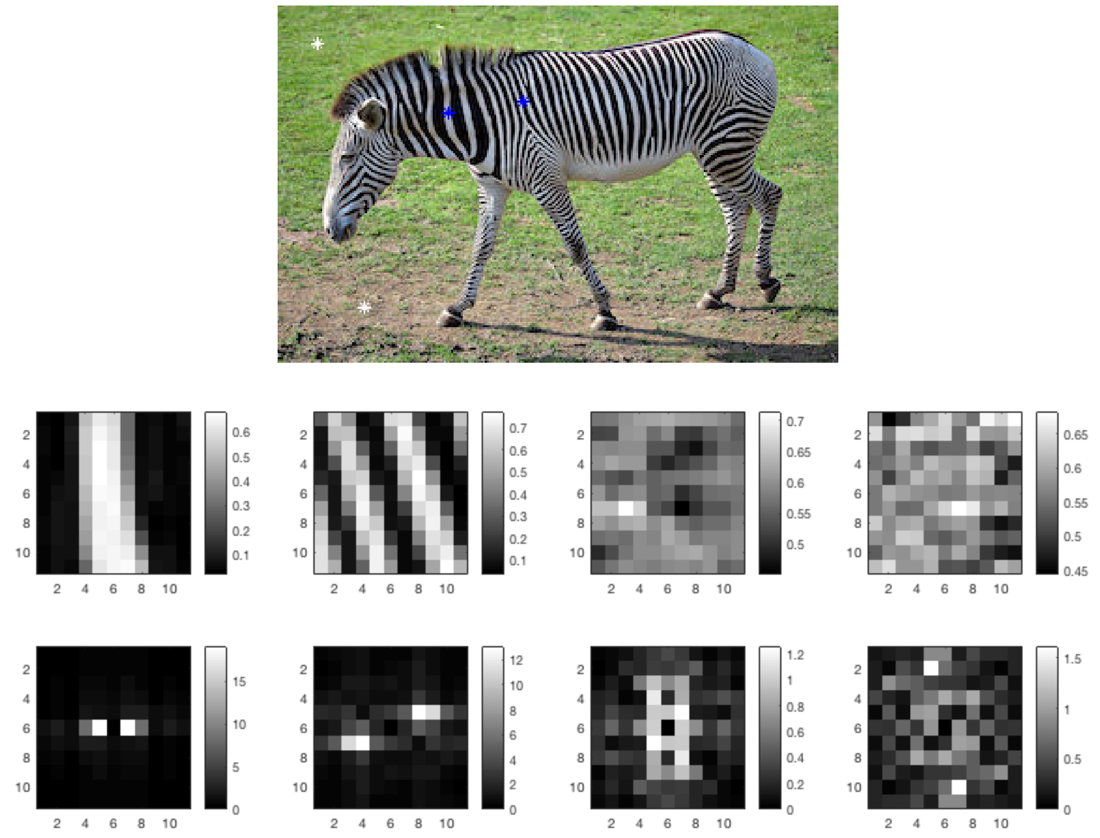


Figure 5: Discrete Fourier transformation example. The first row contains original image. The left two images in the second row - samples of the object texture (blue points on the original image). The next two images - samples of the background. The last row shows magnitudes of discrete Fourier transformation of the correspondent texture samples.

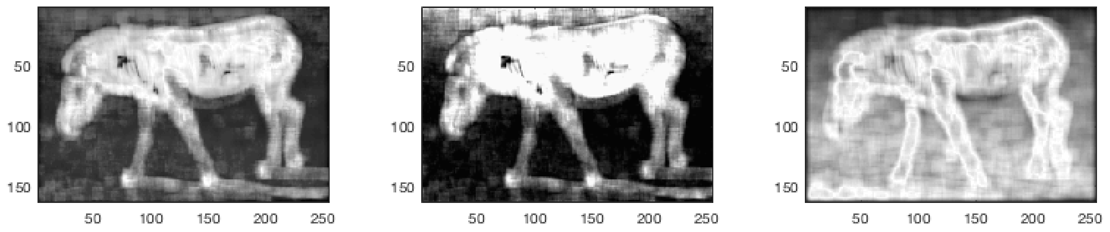


Figure 6: Fourier features. The left most image - combined feature vector. The middle image - entropy and the last image - sum of all frequencies. On can clearly see, that the blurred top part of the image differs from the rest of the background.

Statistical moments.

I'd like to show also how the proposed segmentation algorithm could be used in medical imaging. For example, in computer tomography for grey valued images. It is clear that the Delta E based on grey value difference is often not sufficient for a good clustering of such images. That's why we need some other feature for better targeting objects within this image domain. For my experiments I have taken one of the simplest features - the local central statistical moment of order two (the variance) to show that even this feature could be among Delta E useful in some cases with the proposed algorithm. The experimental image (Fig 7), for which this feature was used, has some very contrast parts (skull bone), so I will take the normalized logarithm of the variance.

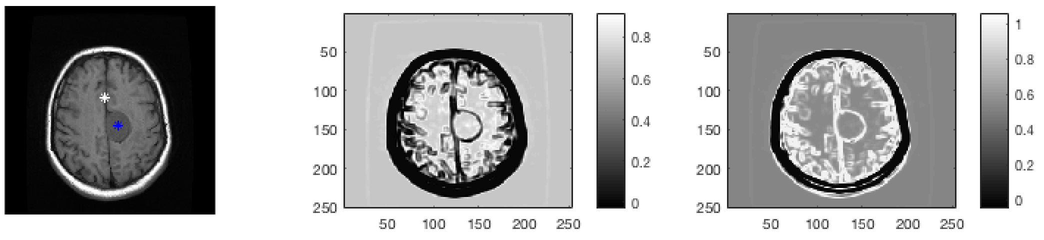


Figure 7: Difference maps of local variance.

Fig. 7 shows an example of the feature application. The original image has an object in the center that we want to mark for further analysis. If we want to use its grey level as a feature we will have a problem with other parts of image, which have the same grey value level. But we can use the fact that other dark parts have increased contrast. So with the proposed feature we can filter these parts out and then use the resulted feature map for the diffusion term to localize the object. It is possible to see on the middle image of the Fig. 7 that the zones with the same grey value level are dark. So the defender will try to avoid them. From the other side the attackers feature map (right image) has increased value in this zones, he will try to occupy them.

3.3 Diffusion tensor.

Anisotropic non-linear diffusion allows us to change the direction of phase spreading. Generally, we want to keep the phase, that should describe the object within the object region and we don't want the background phase to get inside the object's region. If we suppose that the feature map that will be used for construction of the diffusion tensor describes the regions where the object should be, we can use the ideas of edge enhancing diffusion with Perona-Malik diffusivity [3] to lokalize the defender in these regions.

The main idea can be described as follows: we want to allow diffusion along the gradient and suppress it across the gradient. The defender in this case will tend to stay within the object region and for the attacker it will be harder to infiltrate in this region.

The Perona-Malik diffusivity suppresses diffusion in the regions with stronger gradients:

$$g(|\nabla u|^2) = \frac{1}{1 + |\nabla u|^2 \cdot \lambda^2}$$

where λ is the inverted (with respect to the original version) contrast parameter that conrols the level of suppression.

To construct the diffusion tensor we have to construct it's eigenvectors and eigenvalues. The direction of the eigenvectors corresponds to the main direction of the diffusion. The value of eigenvalues corresponds to the diffusivity in the corresponding direction. So as we want to suppress the diffusion across the gradient we will make the first eigenvector parallel to the gradient and suppress the diffusion in this direction with Perona-Malik diffusivity. The second vector will be normal to the previous vector and the diffusivity in this direction will be gained for the regions with gradient. Both eigenvalues will be also proportional to image feature, that is used for the diffusion. Because, for example, we don't want the attacker phase to spread within object's area after it is got across the border.

$$v_1 \parallel \nabla u, v_1 \perp \nabla u$$
$$k_1 = f \cdot g(|\nabla f|^2), k_2 = f + f \cdot (1 - g(|\nabla f|^2))$$

where v_1, v_2 are normalized eigenvectors, k_1, k_2 are correspondent eigenvalues and f is the correspondent feautremap value.

To construct the diffusion tensor from the eigenvalues and eigenvectors we can use the main property of the eigenvalue decomposition.

$$D \cdot v_i = \lambda_i \cdot v_i \;\; or \;\; D \cdot V = V \cdot L$$

where V is a matrix that has the eigenvectors as columns, L is the diagonal matrix of eigenvalues . If V is invertible (eigenvectors are linearly independent) we can write:

$$D = V \cdot L \cdot V^{-1}$$

If further v are normalized, we will get the following equation:

$$D = V \cdot L \cdot V^T$$

For 2x2 matrices with $V = \begin{pmatrix} v_{1,x} & v_{2,x} \\ v_{1,y} & v_{2,y} \end{pmatrix}$ the last equation can be rewritten as follows:

$$D_{xx} = v_{1,x} \cdot v_{1,x} \cdot \lambda_1 + v_{2,x} \cdot v_{2,x} \cdot \lambda_2$$
$$D_{xy} = v_{1,x} \cdot v_{1,y} \cdot \lambda_1 + v_{2,x} \cdot v_{2,y} \cdot \lambda_2$$
$$D_{yy} = v_{1,y} \cdot v_{1,y} \cdot \lambda_1 + v_{2,y} \cdot v_{2,y} \cdot \lambda_2$$

The whole tensor looks like: $D = \begin{pmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{pmatrix}$

3.4 The equation system.

I will use the following notation: f_1 - feature map for the attacker, f_2 - feature map for the defender, $b(f_2)$ - boosting / suppressing function for the defender, u_1 - attacker population and u_2 - defender population. Following all the ideas described int the previous sections, the equations for the attacker (13) and the defender (14) would be:

$$\frac{\partial u_1}{\partial t} = \mathrm{div}(D_1 \cdot \mathrm{grad}(u_1)) + f_1 \cdot u_1 \cdot (1 - u_1 - b(f_2) \cdot u_2) \tag{13}$$

$$\frac{\partial u_2}{\partial t} = \mathrm{div}(D_2 \cdot \mathrm{grad}(u_2)) + f_2 \cdot u_2 \cdot (1 - u_2 - f_1 \cdot u_1) \tag{14}$$

where

$$b(x) = \begin{cases} 1 & \text{if } x > C_{boost} \\ \left(\frac{x}{C_{boost}}\right)^{C_{supp}} & \text{otherwise} \end{cases} \tag{15}$$

Constructing the boundary conditions I want to prevent the diffusion across the image border. So it will be the simple Neumann boundary condition.

$$\frac{\partial u_i}{\partial n} = 0 \tag{16}$$

Where n is the image border normal.

As the initial condition for the defender an the attacker equation a set of object and background seed points correspondingly is used.

Significant difference between the two main equations in the system is their reaction terms. For the first equation I use the reaction term in the following form: $f_1 \cdot u_1 \cdot (1 - u_1 - b(f_2) \cdot u_2$ The first part of this term $f_1 \cdot u_1$ itself controls the growth speed of the phase in each point. The idea: the bigger the feature value is, the faster the function grows. The second part $(1 - u_1 - b(f_2) \cdot u_2)$ controls the suppression the growth due to "limited resources" and presence of the defender. If this term becomes negative what means $u_1 + b(f_2) \cdot u_2 > 1$ and the diffusion part can not compensate it, the growth speed becomes also negative.

Without the boosting function $b(x)$ the abilities of one phase to suppress the other phase are proportional to the feature map. The function can boost or decrease this abilities depending on the local value of the feature.

Example of the function $b(x)$ for $C_{supp} = 4$ and $C_{boost} = 0.7$ is shown below.

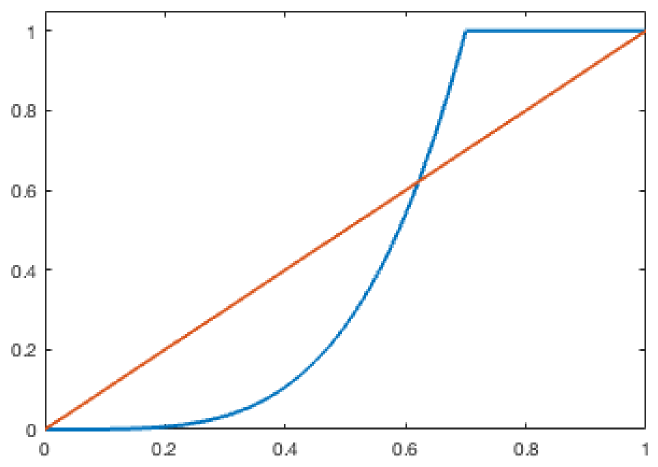


Figure 8: Defender function (blue line). The interval below the red line - suppresing region, above the red line - boosting region.

The idea of this function: in the object region the defender can stronger suppress the attacker and can nearly totally eliminate it if the boosting function is equal to 1. On the background region the attacker is stronger than the defender and can suppress him more effectively. The parameters C_{supp} and C_{boost} will be analysed and explained in the next section.

3.5 Model analysis.

In this section I will analyse the influence of different model parameters and try to give intuition behind them. I hope it should also better clarify the purpose of different equation parts.

The boosting function b(x).

The main idea behind this function is to give the defender an advantage in the region, where the object should be located. With the parameters C_{supp} and C_{boost} we can control the boosting and suppression intervals (See equation (15)).

C_{boost} is a sort of threshold that gives the defender ability to suppress the attacker in a region with feature map value close or greater than C_{boost} (5), (6). This will create a sort of a barrier that complicates for the attacker infiltration inside this region and prevents him from occupation of some parts where it could suppress the defender.

To illustrate this process I will construct a synthetic example. Let's say we have a grey region (the object) with a white background (Fig. 9). The grey region contains some inhomogeneity in form of black and white dots that belongs to the object. Initially the seed points for the defender are placed inside the black regions, the seed points for the attacker are placed in top left and top right corners (Fig. 10 left).

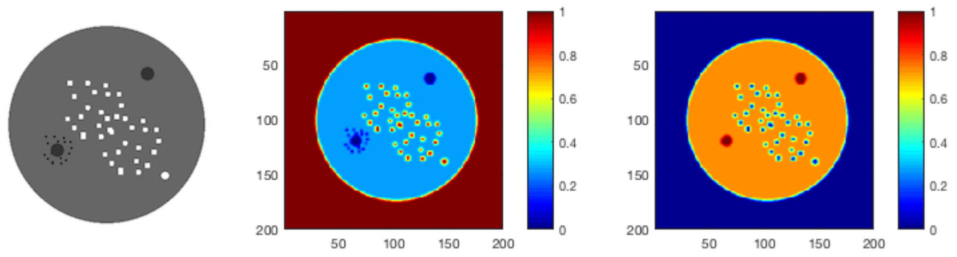


Figure 9: Influence of the defender boosting. Process initialisation. Left: original image. Middle: attacker feature map. Right: defender feature map.

The feature maps for the attacker and the defender are illustrated above (Fig. 9). The attacker has huge advantage in white regions but he can also move in the grey region, the correspondent feature value there is not 0. The defender has advantage inside the grey region, but since it is initialized on the black points, the grey color is not 100% best for him (feature map is not 1). Now if we start the evaluation process in one case without boosting and in the other case with $C_{boost} = 0.5$ we will notice the influence of the parameter. For simplification there is no anisotropy in this example ($\lambda = 0$). The time in both cases is equal 100. The phases are marked with red for attacker and blue for defender (Fig. 10).

In case without boosting the attacker has reached the inner area of the object from both sides before the defender started pushing it back. But due to uncompensated diffusion flow (the defender can not compensate it to 100% in this case) of the attacker phase inside the object it reaches the white points. Inside these white points the attacker grows very fast and starts spreading deeper in the grey region (Fig. 10 middle).

In the case with boosting we can see only one red point inside the grey region. The attacker has reached this point before the defender has pushed it back. Then the attacker was suppressed in all parts of the grey region except this one point. But due to isolation he can not spread to the other points (Fig. 11 right).

This is the main purpose and the main idea of the boosting function. The real images are normally inhomogeneous. But even if the feature map for an object contains some perturbations inside the object region, the defender can occupy the most parts of the region and prevent the attacker to go inside the area.

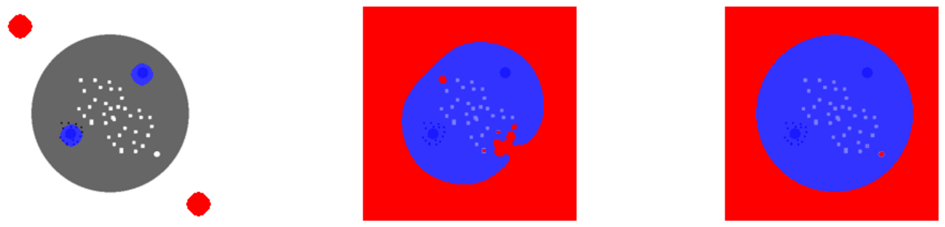


Figure 10: Influence of the defender boosting. Left: start configuration. Middle and right: results without and with boosting correspondingly.

The next parameter is C_{supp}. With increasing of C_{supp} the boosting function decreases faster (see Fig. 12), that means decreasing of defender abilities to suppress the attacker in regions where corresponding feature is smaller than certain threshold. The main idea behind this parameter can be described as follows. If we have images with some contrast object that needs to be separated from the background, we don't need very "selective" defender because the feature map contrast allows the algorithm to distinguish between object parts and background parts. But if we have a complex background with some parts that are close to the object in the sense of feature map value, we have to make the defender more selective. Because otherwise it will occupy this background parts and will make choosing the attacker seed points more complicated (we have to surround the object completely in this case). The C_{supp} parameter should increase the selectivity of the defender. I will illustrate the influence of this parameter with the following example (Fig. 11).

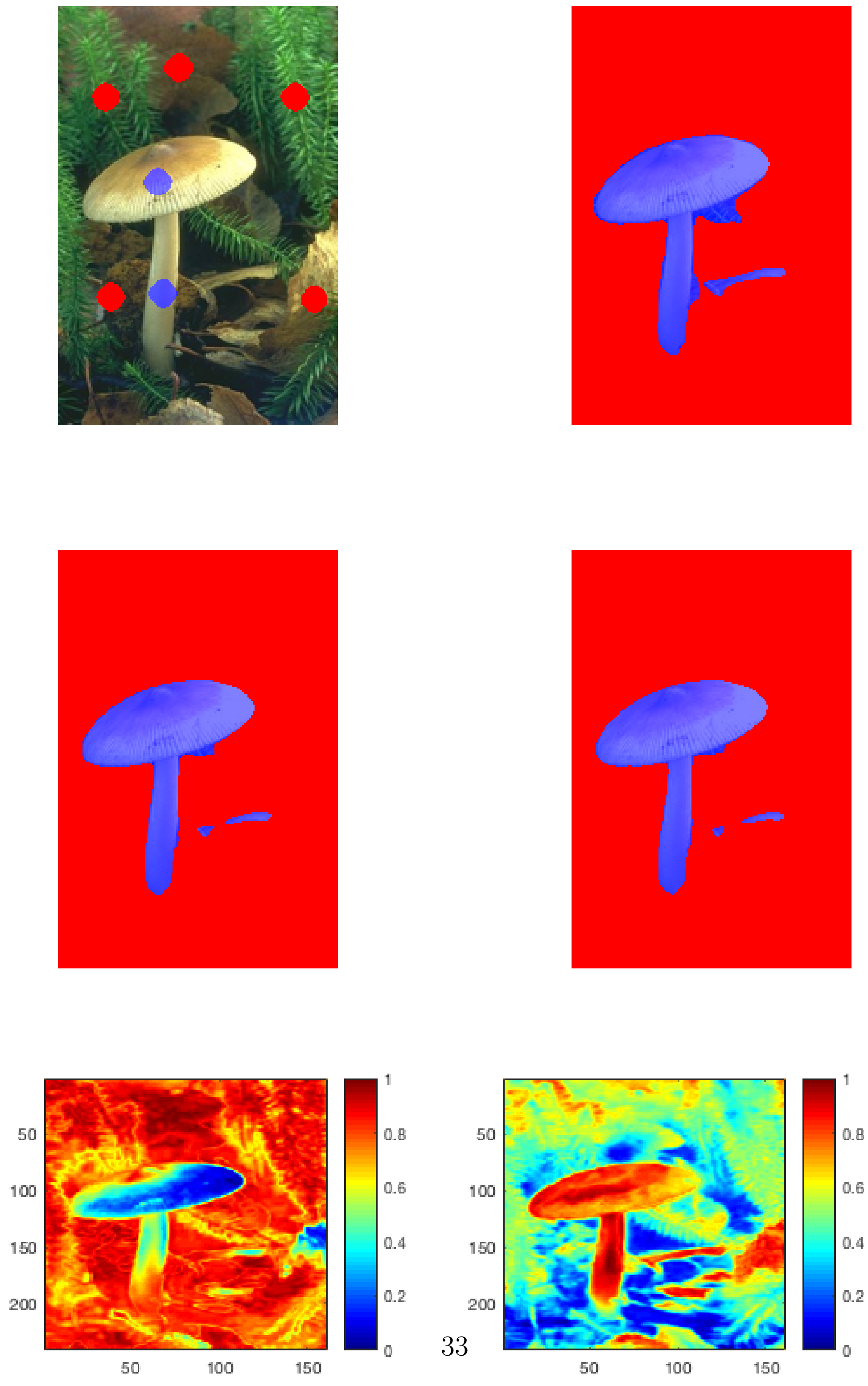


Figure 11: From left to right, top to bottom: original image with seed points, clustering with $C_{supp} = 2$, $C_{supp} = 4$, $C_{supp} = 6$, feature map of the background, feature map of the object. In each case $t = 65$

The two bottom images of the Figure 11 contain feature map (the delta E) for attacker (left) and defender (right). The mushroom is the object that we want to separate from background. As we can see on the bottom right image there are few contrast elements besides the object that have the same feature map level as the object. But there are also some other background elements that lie close to the object feature map. So if we initialize the process as shown on the top left image (blue dots - defender, red dots - attacker) with different C_{supp} we will see the increasing of the defender's selectivity. The bigger this parameter is, the faster the ability of the defender to suppress the attacker out of the boosted region decreases. We can see that in this particular case (and also in the most my experiments) $C_{supp} = 4$ is enough to let only very contrast background parts. Further increasing of the parameter makes no big difference. So in further examples I will use $C_{supp} = 4$

The boosting / suppressing function for all 3 cases:

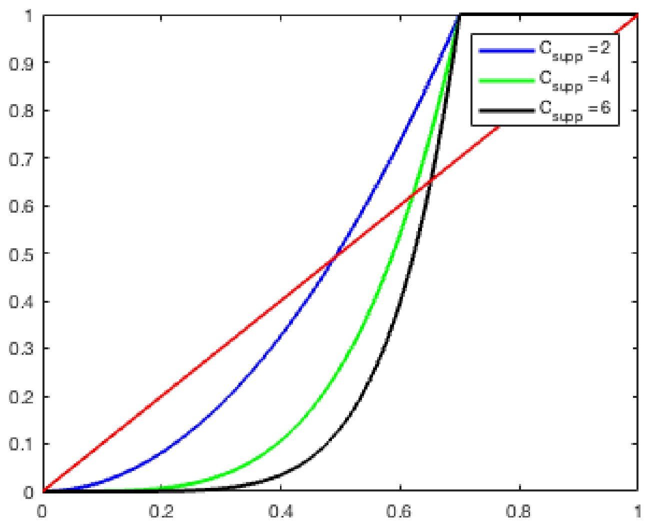


Figure 12: Boosting / suppressing function

There is one more note about the boosting / suppressing function. Even in a region where this function is equal one, the attacker phase can not be always suppressed to 100% because of diffusion. Due to diffusion there be a constant flux of the attacker phase inside the defender region. How deep can the attacker reach inside the region depends on balance of the correspondent feature map values. But if we want to completely prevent the attacker to move inside the object region, the boosting function should be in some cases greater than 1 and should completely compensate the diffusion flux of the attacker.

Anisotropic diffusion.

The main purpose of using the diffusion in this algorithm is the transport function. The defender and attacker phases have to move according some feature maps. In the best case the defender should stay within the object region and the attacker should occupy background but not the object. At the same time real images are often complex and there is no ideal feature map that can highlight the object perfectly. As we have seen above the surrounding space can contain elements with the same shape, color, texture or other properties used for building the feature maps. Surely we can control the initialisation of the diffusion process and, by choosing different seed points we can partially solve described problem. But I was trying to find some mechanics that will even more force the defender to stay in his region and will prevent the attacker stronger from occupation of the object region parts. This would imrpove the quality of the algorithm in case of natural images.

So I came up with the idea to use anisotropic diffusion. I will try to give some more intuition behind it with the following example: let's say in some part of an image we have an object in form of the grey gradient line (Fig. 13 left).

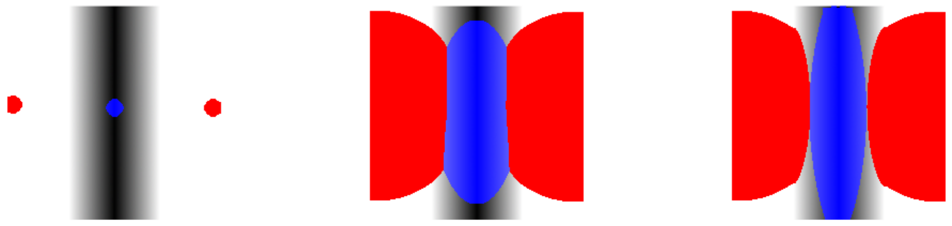


Figure 13: Influence of anisotropy. From left to right: original image, isotropic nonlinear diffusion, anisotropic nonlinear diffusion

Let's say further, that we have initialized the defender in the center of the grey region and placed two seeds of attacker from the left and right sides. Then we start the diffusion process in one case only with feature map dependent non-linearity ($\lambda = 0$) and in the other case with anisotropy ($\lambda = 100$). The time stop point is equal for both cases.

We can clearly see the influence of anisotropy from the result images. The defender phase has moved more quickly in vertical direction and has occupied the most parts of the grey region before it "touched" the attacker phase.

So by using the anisotropy we can improve the spreading of phases. They will tend to move along the gradients, what means along the object boundaries.

The other aspect of using anisotropy is the fine border elements problem described above (see Fig. 2).

Let's consider one more synthetical example to show the influence of anisotropy in this case (see Fig. 14).

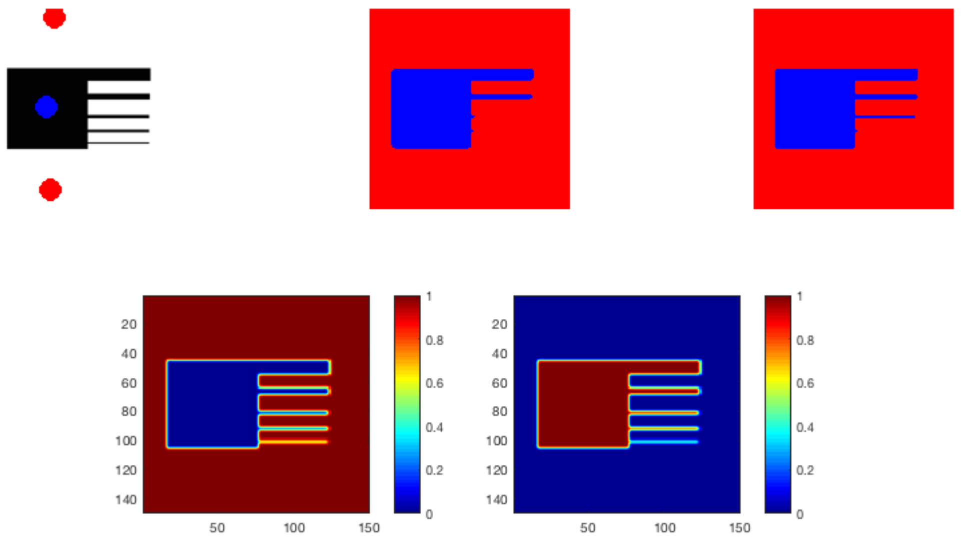


Figure 14: Different anisotropy ratio. Top row from left to right: initial state, evaluation result without anisotropy, evaluation result with anisotropy ($\lambda = 100$). Bottom row: featuremaps for the attacker (left) and defender (right)

On the figure above an object with complex right border is shown. The border consists of thin lines with different width that decreases from top to bottom. The defender seed is placed inside the figure and two attacker seeds are placed near the top and bottom boundary of the image (Fig. 14 top left). In the bottom row the calculated feature maps for the attacker (left) and defender (right) are shown. Considering the correspondent feature maps, we can see that the attacker should prefer everything except inner parts of the object and the defender should prefer the object. If we start the evaluation without anisotropy - only two top thin lines will be occupied by the defender. The other three thinner lines are occupied by the attacker despite the fact that their feature value a much more better for the defender. This happens due to diffusion processes, which have shifted the equilibrium in these regions.

If we now restart the process with the same parameters except the anisotropy contrast ($\lambda = 100$) we will see that one more border element is now occupied with the defender. This is the result of weaker diffusion across the border and stronger diffusion along the border. On the Fig. 15 main direction of the defender phase diffusion for a thin border element from the example above is shown.

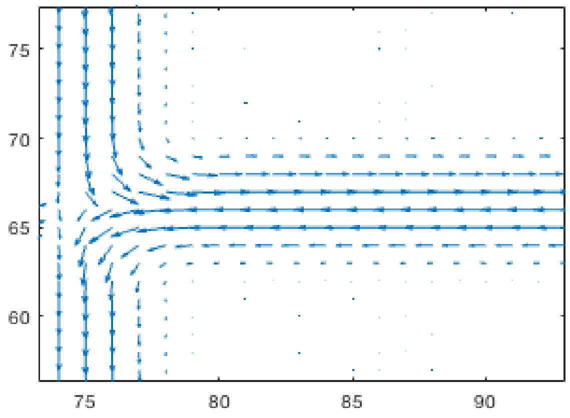


Figure 15: Main direction of anisotropic diffusion

For the attacker we would see the same picture outside the object, the diffusion is directed along the border.

Interactivity.

The last thing I want to mention in this section is the interactivity of the algorithm. It is clear, that because of effects described above the final result depends on the seed points configuration. If we will surround some object with attacker seed points, then the defender will have less chances to get in the environment around and so on.

3.6 Mutiple features.

For all examples above I have used the same feature map for the diffusion part as for the reaction part. This works quite good if we have some contrast feature map that highlights the object and does not contain background elements that have the same feature map value level. But in the opposite case we would have a problem separating the object and the background.

We can try to solve this problem using different feature maps for the diffusion and reaction parts. The idea is: if some feature has a good contrast but intersects with some background elements we can try to find one more feature, that could have weaker contrast but would better localize the object. This second feature can be used for diffusion part and the first feature for the reaction part. Due to diffusion localization the defender will act within the object region and the contrast of the reaction part feature should gain the accuracy of the resulting cluster boundaries.

I will illustrate this idea with the following example.

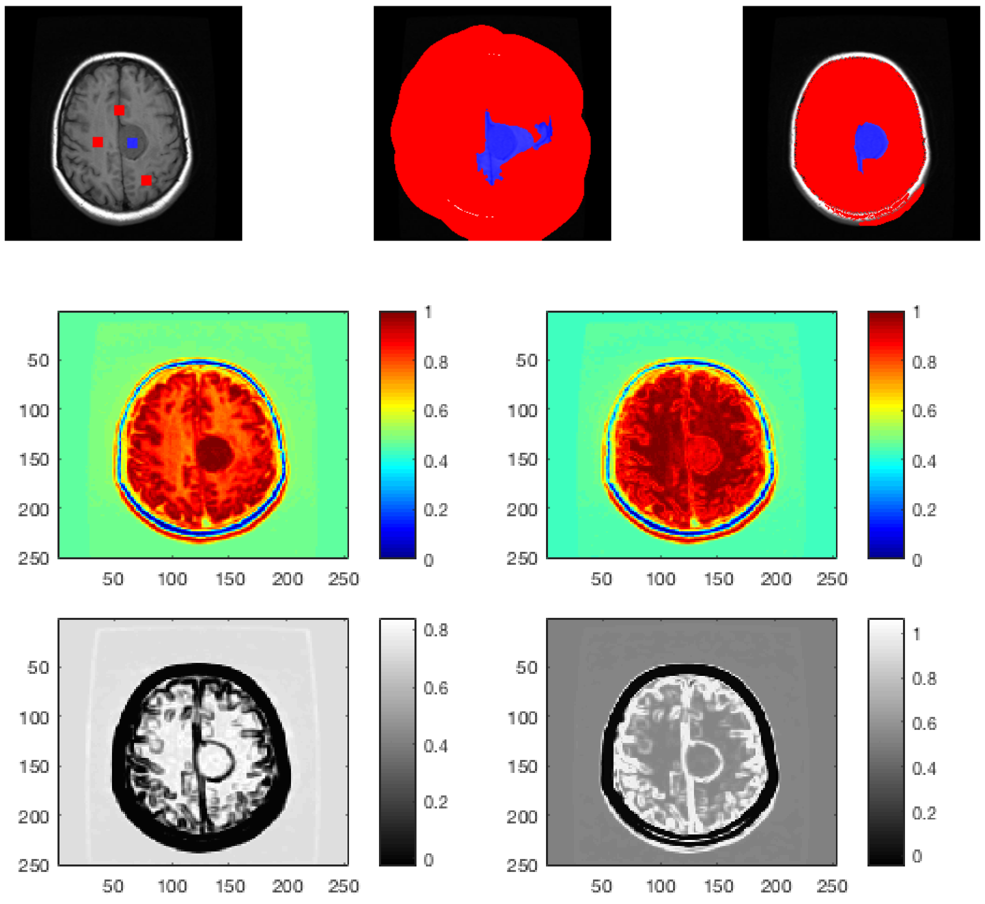


Figure 16: Mutiple features. From left to right, top to bottom: original image with seed points, clustering result using delta E only, clustering result using central moment for the diffusion part, delta E for the defender, delta E for attacker, central moment for the defender, central moment for the attacker.

Let's say we want to analyse dynamic of a tumor growth using computer tomogram of a brain (see Fig. 16). For this purpose we have to separate the tumor from other parts of the brain. If we will use only the Delta E, we will have many regions on the image with the same grey value as a tumor itself (second row left). Surely, in this case due to image symmetry we can surround the tumor with the attacker seeds and clusterize it with Delta E only. But I want to show the other way, that can be used in more complex situation.

The tumor itself have relative homogeneous grey value distribution. But the parts of the brain with the same grey value have stronger contrast as the tumor. So, if we will use a feature, that detects contrast, we can separate those regions from the tumor and build the correspondent feature maps for the diffusion part of attacker and defender. The last row of the Figure (16) contains examples of this feature maps.

We can see that on the defender diffusion feature map (left image) the contrast brain parts have much lower value than the tumor itself. For the attacker (right image) the situation is the opposite. So, the defender will avoid this regions, at the same time the attacker will try to occupy them. The top middle and top right images show the result of clustering without and with the using different features. The diffusion time is equal in both cases ($t = 80$). We can clearly see that in the last case the result is much better and corresponds to the reality.

If we need it, we could even use in such cases some synthetic feature map (drawn by hand for example.)

One more illustration of this idea is shown on the Fig. 17. The object has textured skin, so our color-related feature is very inhomogeneous. Furthermore, the bottom part of the image contains many elements with the same feature value. We can try to clusterize this image using only delta E and choosing specific seed points. But due to texture properties, the attacker will tend to get inside the object occupying zebra stripes near the object boundary.

Using a texture feature (based on Fourier amplitudes in our case, see Sec. 3.2) will simplify the process. The defender will be better localized inside the object and the attacker outside. The result of the clustering is shown below. The separated object looks sharp and contains no artifacts despite the not perfect features.

We can see also a small differences between the clustering with and without the anisotropy. On the isotropic version the front legs are thinner and the middle part of upper border of the object looks unevenly. But with this initial configuration of seed points the difference is insignificant.

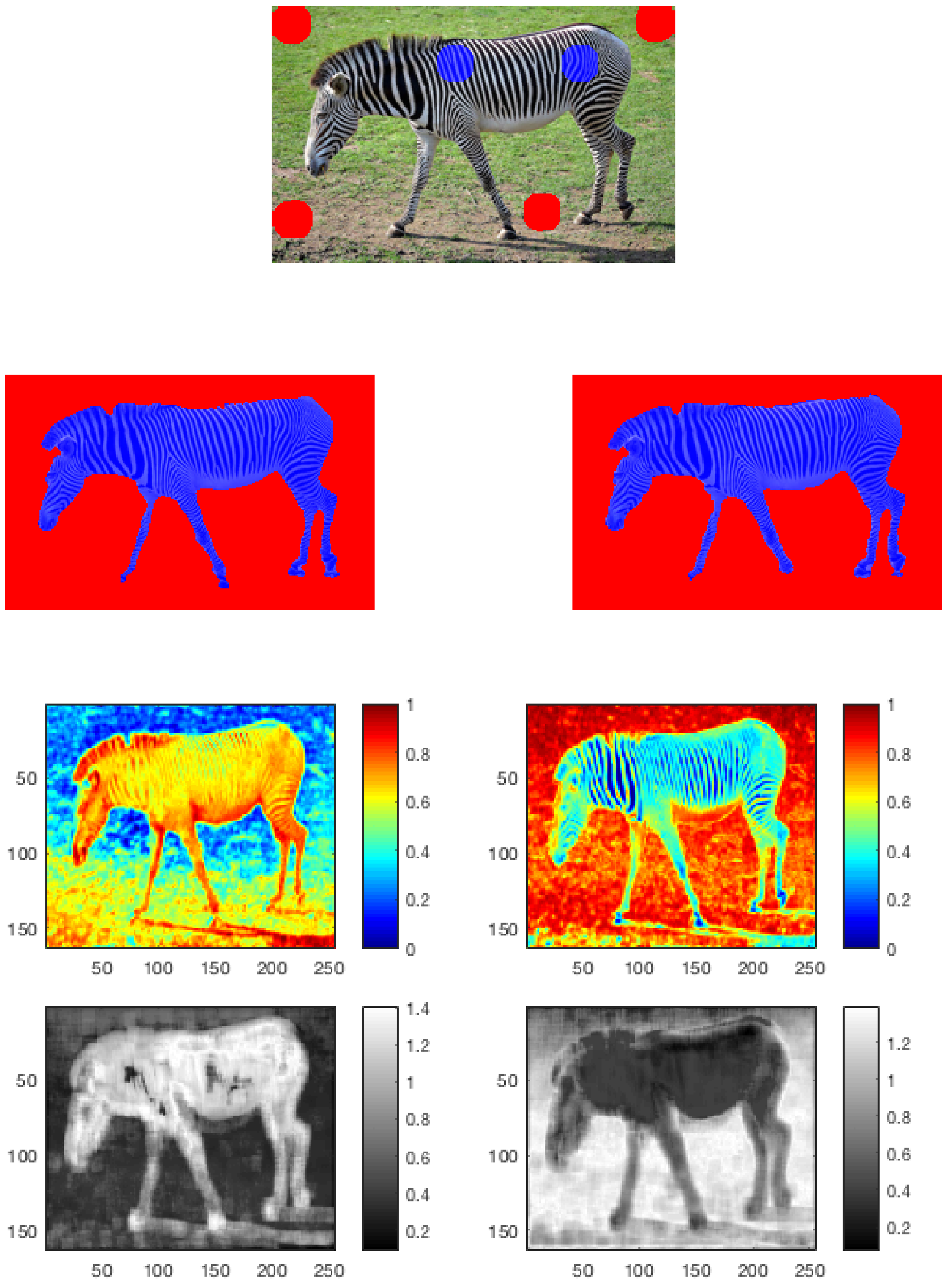


Figure 17: Mutiple features. From left to right, top to bottom: original image with seed points, isotropic clustering result using multiple features , anisotropic clustering result using multiple features, defender and attacker delta E features, defender and attacker texture features

3.7 Robustness against noise.

To test if the algorithm robust against the noise, I have generated gaussian noise directly over the calculated feature maps. I did not test the feature maps itself on the robustness against the noise because this master thesis is concentrated on the model. The feature maps were used only for experiment and demonstration purposes, and do not play a major role themselves in the context of this work. The results of noise tests are shown below.

As we can see on the Fig. 18 the segmentation result was not changed dramatically because of noise. The top boundary of the object looks worse because the attacker phase has occupied partially the border regions. The reaction term in this regions is sensitive to noise. But inside the object diffusion compensates for the noise influence and we don't see any dramatic differences.

In general this model should perform good for noisy image because of "averaging" property of the diffusion term. Only in border regions, where the reaction term has stronger influence, the noise could play a significant role changing the accuracy of clustering. Anisotropic diffusion should be also in some cases averaged or replaced with isotropic diffusion because of its potential sensitivity to noise.

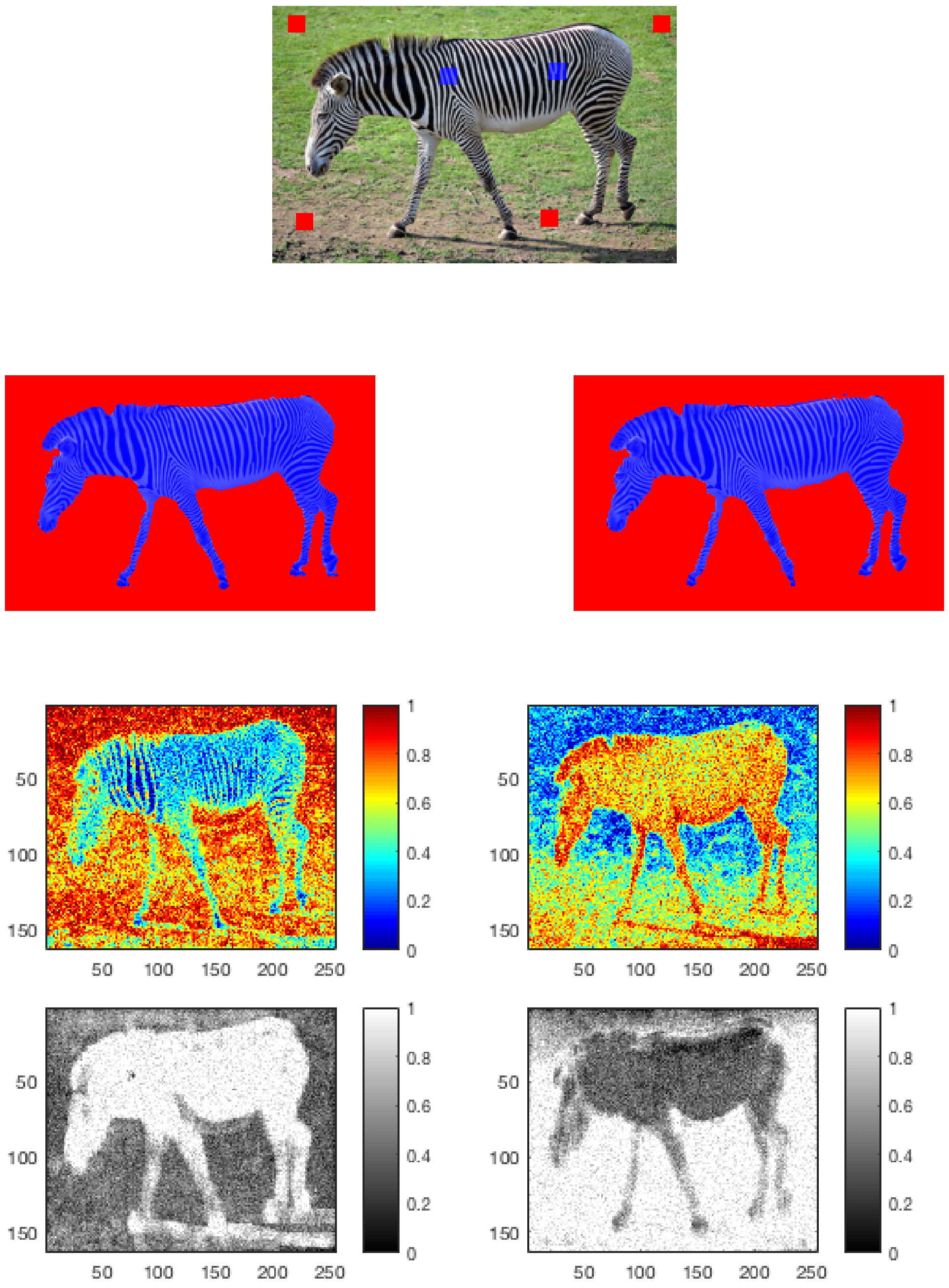


Figure 18: Noise tests. From left to right, top to bottom: original image with seed points, isotropic clustering result using multiple features , anisotropic clustering result using multiple features, defender and attacker delta E features, defender and attacker texture features

3.8 Multiple phases.

In all examples above I used one defender phase and one attacker phase. But this is not a restriction. We can use different defender phases for different objects or different attacker phases for separate parts of the background. We can merge them after stopping the evaluation process.

In the case of multiple phases an example of community matrix A (see eq. 1) is shown below :

$$A = \begin{vmatrix} 1 & b_2(f_2) & f_3 & f_4 & ... \\ b_1(f_1) & 1 & f_3 & f_4 & ... \\ b_1(f_1) & b_2(f_2) & 1 & f_4 & ... \\ b_1(f_1) & b_2(f_2) & f_3 & 1 & ... \\ ... & ... & ... & ... & ... \end{vmatrix}$$

Each defender will have its own boosting function $b_n(f_n)$ and will protect its own region. If two different defenders have intersection in the same region, the result will depend on many factors like: the overall diffusion in the region, presence of other phases, local feature values and so on. This increases significantly complexity of the model and makes it less intuitive. This is the reason why I have used only two phases in all experiments.

4 Implementation notes.

In this section I will briefly describe general scheme of the algorithm and implementation of its main steps.

4.1 Steps of the alogrithm

Generally the algorithm proceeds as follows:

1) User marks seed points for the object and for the background. If the user uses different feature maps for the reaction and diffusion terms, he can choose different seed points for each map.

2) User starts the evaluation.

3) First, the chosen feature maps are calculated. Then, correspondent averaged values of the features in seed points are evaluated. The difference between the averaged values and all other points of the correspondent feature map is calculated. The difference maps are inverted and merged for all seed points using max() function.

4) The 0 iteration of the discrete solution is initialized with seed points.

5) The iteration loop is started. Periodically, intermediate results are shown. Regions where the defender phase value is greater than the attacker phase are marked with blue color. Regions where the attacker phase is greater than the defender phase are marked with red.

6) The algorithm stops when the t_{max} is reached or the user has aborted the evaluation.

4.2 Numerical solution of the equation system

Calculation of diffusion tensor.

To calculate the diffusion tensor, following steps are performed :

1) Blurring the feature maps with gaussian blur to suppress noise (in all described examples $\sigma = 0.5$)

2) Approach, described in sec. 3.3 is applied directly.

Discretisation of the quation.

For numerical solution of the equation system I will use the explicit finite difference schema. It is very simple, does not require complex steps for implementation and in my tests it has performed quite good. In this subsection I will describe the main steps of building this schema. More information about discrete or semi-discrete solution of diffusion equation can be found here [21].

In our equation we have a time and two spatial dimensions, which have to be discretised. Since schema is explicit, the time discretisation is very simple:

$$\frac{u^{t+1} - u^t}{\Delta t} = \operatorname{div}(D \cdot \operatorname{grad}(u^t)) + r(u^t) \tag{17}$$

where $r(u^t)$ is a reaction part (See (13), (14)). All variables u^t in the right part of the equation are already computed so we can directly start to calculate the next time layer. This makes the explicit schema very attractive, because we don't need to worry about complexity of the right part in this sense.

Next step is the approximation of the right part of the equation. The reaction term does not contain any differential operators, so we can evaluate it locally. But the diffusion part needs further simplification and discretisation.

$$\begin{aligned}
\operatorname{div}(D \cdot \operatorname{grad}(u^t)) &= \operatorname{div}\left(\begin{pmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{pmatrix} \cdot \begin{pmatrix} \frac{\partial u}{\partial x} \\ \frac{\partial u}{\partial y} \end{pmatrix}\right) \\
&= \frac{\partial}{\partial x}\left(D_{xx} \cdot \frac{\partial u}{\partial x}\right) + \frac{\partial}{\partial x}\left(D_{xy} \cdot \frac{\partial u}{\partial y}\right) \\
&+ \frac{\partial}{\partial y}\left(D_{xy} \cdot \frac{\partial u}{\partial x}\right) + \frac{\partial}{\partial y}\left(D_{yy} \cdot \frac{\partial u}{\partial y}\right)
\end{aligned} \tag{18}$$

Now we have four terms with derivatives. Second order x- and y-derivatives can be approximated as follows [21]:

$$\begin{aligned}
\frac{\partial}{\partial x}\left(D_{xx} \cdot \frac{\partial u}{\partial x}\right) \approx \frac{1}{\Delta x} \cdot \Bigg(& D_{xx}(x + \frac{1}{2}\Delta x, y) \cdot \frac{u(x + \Delta x, y) - u(x, y)}{\Delta x} \\
&+ D_{xx}(x - \frac{1}{2}\Delta x, y) \cdot \frac{u(x - \Delta x, y) - u(x, y)}{\Delta x}\Bigg)
\end{aligned}$$

Here we approximate the inner first order derivative with standard finite difference. To approximate the second order derivative we compute the finite difference of the first order derivatives multiplied with D_{xx} in a middle point. Now we can write down the whole approximation [21]:

$$\frac{\partial}{\partial x}\left(D_{xx}\cdot\frac{\partial u}{\partial x}\right)\approx\frac{1}{\Delta x}\cdot\Bigg(\frac{D_{xx}(x+\Delta x,y)+D_{xx}(x,y)}{2}\cdot\frac{u(x+\Delta x,y)-u(x,y)}{\Delta x}$$
$$+\frac{D_{xx}(x-\Delta x,y)+D_{xx}(x,y)}{2}\cdot\frac{u(x-\Delta x,y)-u(x,y)}{\Delta x}\Bigg)$$

Symmetrically the same approach can be applied to the second y-derivative [21].

$$\frac{\partial}{\partial y}\left(D_{yy}\cdot\frac{\partial u}{\partial y}\right)\approx\frac{1}{\Delta y}\cdot\Bigg(\frac{D_{yy}(x,y+\Delta y)+D_{yy}(x,y)}{2}\cdot\frac{u(x,y++\Delta y)-u(x,y)}{\Delta y}$$
$$+\frac{D_{yy}(x,y+\Delta y)+D_{yy}(x,y)}{2}\cdot\frac{u(x,y+\Delta y)-u(x,y)}{\Delta y}\Bigg)$$

The standard approximation for the mixed derivatives part could be retrieved using central differences scheme [21]:

$$\frac{\partial f(x,y)}{\partial x}=\frac{f(x+\Delta x,y)-f(x-\Delta x,y)}{2\cdot\Delta x}$$

If we apply it first to the inner derivatives and then to the outer derivatives, we will get the following expressions [21]:

$$\frac{\partial}{\partial x}\left(D_{xy}\cdot\frac{\partial u}{\partial y}\right)\approx\frac{1}{2\cdot\Delta x}\cdot\Bigg(D_{xy}(x+\Delta x,y)\cdot\frac{u(x+\Delta x,y+\Delta y)-u(x+\Delta x,y-\Delta y)}{2\cdot\Delta y}$$
$$+D_{xy}(x-\Delta x,y)\cdot\frac{u(x-\Delta x,y+\Delta y)-u(x-\Delta x,y-\Delta y)}{2\cdot\Delta y}\Bigg)$$

The same procedure could be done for the remained mixed derivative [21]:

$$\frac{\partial}{\partial y}\left(D_{xy}\cdot\frac{\partial u}{\partial x}\right)\approx\frac{1}{2\cdot\Delta y}\cdot\Bigg(D_{xy}(x,y+\Delta y)\cdot\frac{u(x+\Delta x,y+\Delta y)-u(x-\Delta x,y+\Delta y)}{2\cdot\Delta x}$$
$$+D_{xy}(x,y-\Delta y)\cdot\frac{u(x+\Delta x,y-\Delta y)-u(x-\Delta x,y-\Delta y)}{2\cdot\Delta x}\Bigg)$$

Now if we replace our derivatives in the sum (18) with its approximations and then regroup the multipliers for all 8 nearest neighbours of $u(x,y)$ we can build a stencil of the following type [21]:

$\frac{-D_{xy}(x-\Delta x,y)-D_{xy}(x,y+\Delta y)}{4\cdot\Delta x\cdot\Delta y}$	$\frac{D_{yy}(x,y+\Delta y)+D_{yy}(x,y)}{2\cdot\Delta y\cdot\Delta y}$	$\frac{D_{xy}(x+\Delta x,y)+D_{xy}(x,y+\Delta y)}{4\cdot\Delta x\cdot\Delta y}$
$\frac{D_{xx}(x-\Delta x,y)+D_{xx}(x,y)}{2\cdot\Delta x\cdot\Delta x}$	$-\frac{D_{xx}(x+\Delta x,y)+2\cdot D_{xx}(x,y)+D_{xx}(x-\Delta x,y)}{2\cdot\Delta x\cdot\Delta x}$ $-\frac{D_{yy}(x,y+\Delta y)+2\cdot D_{yy}(x,y)+D_{yy}(x,y-\Delta y)}{2\cdot\Delta y\cdot\Delta y}$	$\frac{D_{xx}(x+\Delta x,y)+D_{xx}(x,y)}{2\cdot\Delta x\cdot\Delta x}$
$\frac{D_{xy}(x-\Delta x,y)+D_{xy}(x,y-\Delta y)}{4\cdot\Delta x\cdot\Delta y}$	$\frac{D_{yy}(x,y-\Delta y)+D_{yy}(x,y)}{2\cdot\Delta y\cdot\Delta y}$	$\frac{-D_{xy}(x+\Delta x,y)-D_{xy}(x,y-\Delta y)}{4\cdot\Delta x\cdot\Delta y}$

Each cell in this stencil corresponds to a certain neighbour point. The center is $u(x,y)$, the top left cell is $u(x-\Delta x, y+\Delta y)$ and so on.

Now we have all necessary approximations and can easy get from (17) the direct expression for u^{t+1}.

According to [21] there is no guarantee of the stability for this schema. Nevertheless, authors claim that practically the scheme is quite stable if time step does not exceed 0.25 for $\Delta x = \Delta y = 1$. For my examples I have used the time step equal 0.01 and the spatial step equal to 1. The schema was in most cases stable.

Implementation of the numerical scheme is shown on the listing 1.

Listing 1: Numerical solution of the equation system (14).

```
// u, uold - arrays for storing the last 2 time layers
// for all equations;
// tensor - components of diffusion tensor;
// cmatrix, r - community matrix and growth speed coefficient;

double dx=1, dy=1, dt=0.01, t=1000;
double** tmp;
```

```
int nt = t / dt;

double rxx=dt/(2*dx*dx);
double ryy=dt/(2*dy*dy);
double rxy=dt/(4*dx*dy);

double** dxx = tensor[0];
double** dxy = tensor[1];
double** dyy = tensor[2];

visualise(uold);
//time loop
for(int time = 0; time < nt; time++){
  //equations loop
  for(int i = 0; i < numPoints; i++){
    //x- y-dimensions loop
    for(int j = 1; j < h-1; j++){
      for(int k = 1; k < w - 1; k++){

        //Preparing data for the reaction term
        double sum = 0;
        for(int n = 0; n < numPoints; n++){
          sum += uold[i][j*w + k]*cmatrix[j*w + k][i][n];
        }

        //Calculate the reaction term
        double f = uold[i][j*w + k]*r[i][j*w + k]*(1 + sum);

        //Compute elements of the stencil
        // TL - top left, T - top and so on
        double TL = - rxy*(dxy[i][j-1,k] + dxy[i][j,k+1]);
        double TR = rxy*(dxy[i][j+1,k] + dxy[i][j,k+1]);
        double BL = rxy*(dxy[i][j-1,k] + dxy[i][j,k-1]);
        double BR = - rxy*(dxy[i][j+1,k ] + dxy[i][j,k-1]);

        double T = ryy*(dyy[i][j,k+1] + dyy[i][j,k]);
        double B = ryy*(dyy[i][j,k-1] + dyy[i][j,k]);
        double L = rxx*(dxx[i][j-1,k] + dxx[i][j,k]);
        double R = rxx*(dxx[i][j+1,k] + dxx[i][j,k]);

        double C = -rxx*(dxy[i][j-1,k] + 2*dxy[i][j,k] + dxy[i][j+1,k])
        - ryy*(dyy[i][j,k-1] + 2*dyy[i][j,k] + dyy[i][j,k+1]);

        //Calculate next time layer
        u[i][j,k] = uold[i][j,k] + f * dt  + uold[i][j, k]*C
        + uold[i][j,k+1]*T + uold[i][j,k-1]*B
        + uold[i][j+1,k]*R + uold[i][j-1,k]*L
        + uold[i][j-1,k+1]*TL + uold[i][j+1,k+1]*TR
        + uold[i][j-1,k-1]*BL + uold[i][j+1,k-1]*BR;

        //Negative value correction.
        if(u[i][j,k] < 0){
          u[i][j,k] = 0;
         }
      }
    }

    //Neumann boundary conditions with 0 flux.
    for(int k = 0; k < w; k++){
      u[i][0,k] = u[i][1,k];
    }
    for(int k = 0; k < w; k++){
      u[i][h-1,k] = u[i][h-2,k];
    }
    for(int j = 0; j < h; j++){
      u[i][j,1] = u[i][j,2];
    }
    for(int j = 0; j < h; j++){
      u[i][j,w-1] = u[i][j,w-2];
    }
  }

  //visualize 1 frame from 100
  if(time % 100 == 0){
    visualise(u);
  }

  //swap the layers
  tmp = u;
  u = uold;
  uold = tmp;
}
```

There is a part in the listing 1 marked as "Negative value correction". It is required to correct possible instability connected with negative growth speed. In continuous case the function will stop growing as soon as the growth speed is zero and there is no diffusion. Due to discretisation it could be that the growth speed of a phase becomes negative at the same time as the phase itself becomes negative. In this case the reaction term will start growing infinitely.

4.3 Other functions.

The following functions were used in different parts of the algorithm for processing the feature maps, calculation of diffusion or reaction term.

Feature maps merging

As mentioned above the difference feature maps for particular seed points can be merged using max() function. The following expression shows the logic of this step:

$$f_{merged} = max(f_i \in M) \tag{19}$$

where M is the subset of features which have to be merged.

RGB to LAB transformation. Extracting the color difference.

To compute the delta E feature we need to transform RGB value to the LAB color space. I use RGB-XYZ-LAB transformation with illuminant D65 and the observation angle = 2. One can choose other appropriate implementation of this transformation this would not play a significant role.

For the Delta E itself the euclidean distance of the LAB components is calculated.

Computation of the gradient.

To calculate the gradient the simple finite difference schema was used. For boundary points I have just copied the value of the nearest neighbour in the direction of the correspondent derivative.

Listing 2: Computation of the gradient.

```
// Input: feature - array of a feature
// Output: grad - array with components of a fetaure gradient
// h, w - height and width of the feature array

double** grad = new double*[h*w];

for(int i = 0; i < h; i++)
{
  for(int j = 0; j < w; j++)
  {
    double dfdx = 0;
    if(j+1 < w){
      dfdx = feature[i, j+1] - feature[i,j];
    }else{
      dfdx = grad[i,j-1][0];
    }

    double dfdy = 0;
    if(i+1 < h){
      dfdy = feature[i+1, j] - feature[i,j];
    }else{
      dfdy = grad[i-1,j][1];
    }

    grad[i,j] = new double[2];
    grad[i,j][0] = dfdx;
    grad[i,j][1] = dfdy;
  }
}

return grad;
```

Gaussian blur.

Blurring procedure is also required as a method of preprocessin in different parts of the algorithm. It is used to reduce noice or to average some value.

The algorithm is based on convolution of the initial image with the gaussian kernel. Basically it can be formulated as follows:

$$I'_{i,j} = \frac{1}{C} \cdot \sum_{k=-n}^{n} \sum_{m=-n}^{n} G_{k,m} \cdot I_{i+k,j+m}$$

$$C = \sum_{k=-n}^{n} \sum_{m=-n}^{n} G_{k,m} \quad and \quad G_{k,m} = \frac{1}{2 \cdot \pi \cdot \sigma^2} e^{-\frac{k^2+m^2}{2 \cdot \pi}}$$

The kernel size should be dependent on σ. If it is too small, not all significant weights will be presented in the kernel and the quality of blurring will be worser. Too big kernel will require much more resources and will give insignificant gain of quality compared to the optimal kernel size. For this algorithm I will use blurring kernels with size 4σ x 4σ. Because for gaussian distribution more than 95% of "total weight" lies in the interval -2σ - 2σ.

For boundary points the correspondent renormalized rest of the kernel is used.

Fourier transformation.

For the forward Fourier transformation the direct implementation of (11) is used. The implementation has low performance but calculation of feature maps perfomed once during the initialisation. So this drawback does not play a major role.

4.4 Implementation of Chan-Vese algorithm.

I have made all experiments with Chan-Vese algorithm using Matlab. In my script standard function "activecontour" from the "Image Processing Toolbox" was used. According to Matlab documentation the implementation of the function is based on [4].

The specific script is shown on the listing below.

Listing 3: Chan-Vese implementation.

```
%Input image
I = imread('brain.png');
imshow(I);

%Transform to grey valued image
I = rgb2gray(I);

%Select the initial contour
mask = roipoly;
```

```
%Start the evaluation
maxIterations = 2000;
res = activecontour(I,mask,maxIterations,'Chan-Vese','SmoothFactor',1);

%Show the result
figure, imshow(res);
```

5 Comparison with Chan-Vese approach.

The main advantage of the proposed approach is its interactivity and flexibility. By selecting seed points that characterize the object and the background we can get good results even in case of highly non-homogenous objects. Using different features for diffusion and reaction terms we can avoid problems of close feature values for the object and background parts.

To compare the results of discussed algorithm with Chan-Vese [4] approach I will apply the Chan-Vese to some images used in examples above.

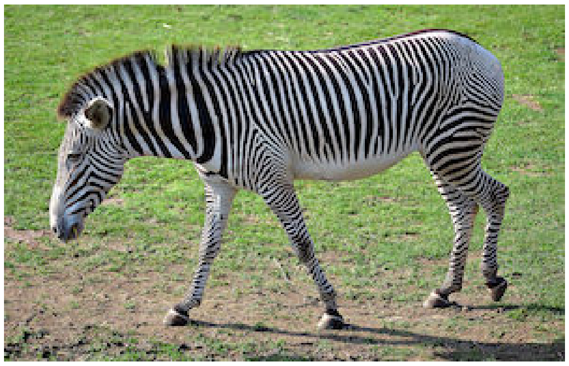


Figure 19: Zebra.

The zebra image contains a contrast texture, that could be a problem for Chan-Vese. We can vary the smoothness of the contour, but there is a certain trade off between fine details and the smoothness. I will illustrate the trade off with the image below.

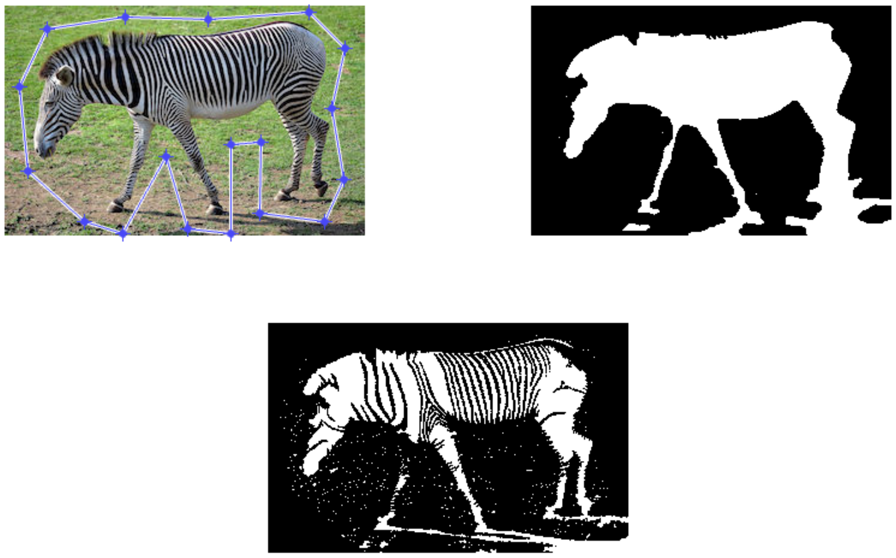


Figure 20: Chan-Vese segmentation. Top left: original image with initial contour. Top right: clustering with high contour smoothness. Bottom: clustering with low contour smoothness.

If the contour is not too smooth, we will get the clear oversegmentation (see Fig. 20 bottom). If the contour is smooth, we will get undersegmentation (rear legs are not separated, Fig. 20 top right). It is not easy to find some good balance between these two effects. Clearly, better contour could help, but if we have multiple fine elements like the mentioned space between rear legs, it would not be very easy to make a good segmentation with Chane-Vese in this case.

The diffusion-reaction approach could help to avoid this problem, provided that the feature used for the reaction term has enough contrast and the diffusion feature has good localization (Fig. 17).

The next example (Fig. 21) has much simpler structure. But if we want to separate the tumor from the rest of the brain, using the Chan-Vese algorithm seems not to be optimal in this case. I have made a few experiments with different parameters, and have not got any appropriable results. This can be explained with the globality of the Chan-Vese algorithm.

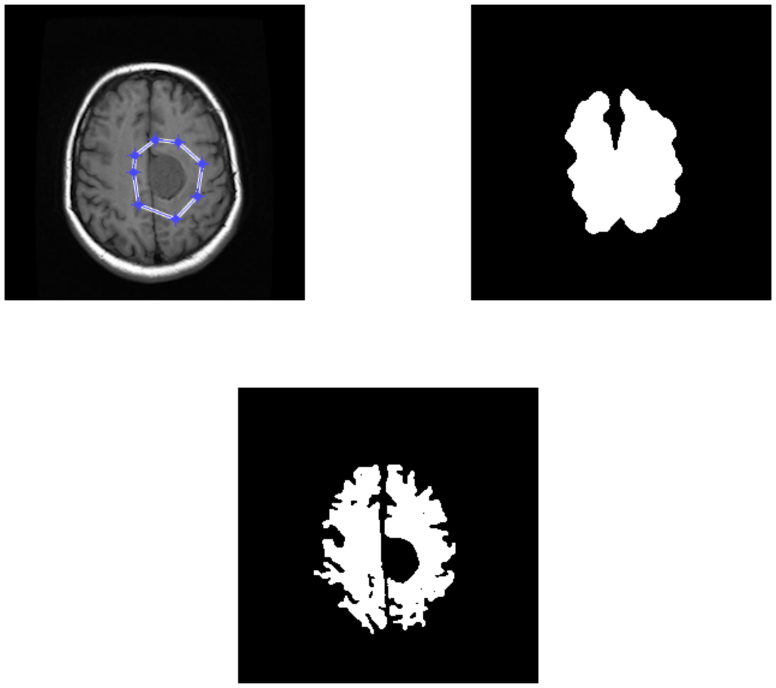
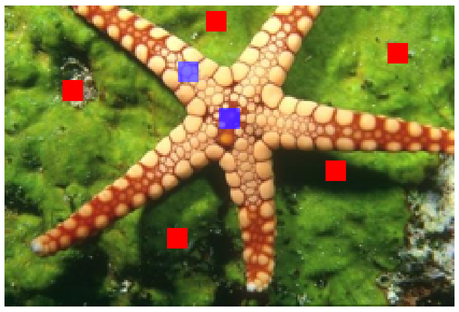
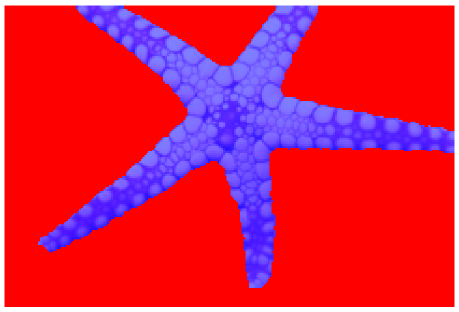


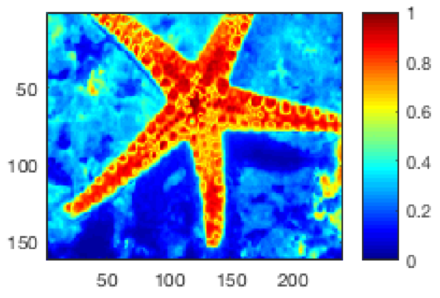
Figure 21: Chan-Vese segmentation. Top left: original image with initialization contour. Top right: clustering result with high smoothing . Bottom: clustering results with low smoothing.

The algorithm proposed in this work has in such case much more flexibility due to its locality. We can use also two different feature maps to gain the quality of the segmentation (see Fig. 16).

6 Some other examples.







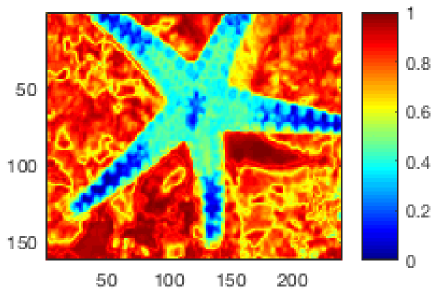


Figure 22: Starfish. Top: the original image. Middle: clustering result. Bottom left: object feature. Bottom right: background feature. $\lambda = 100, C_{boost} = 0.7, C_{supp} = 4, t = 90$, Delta E only

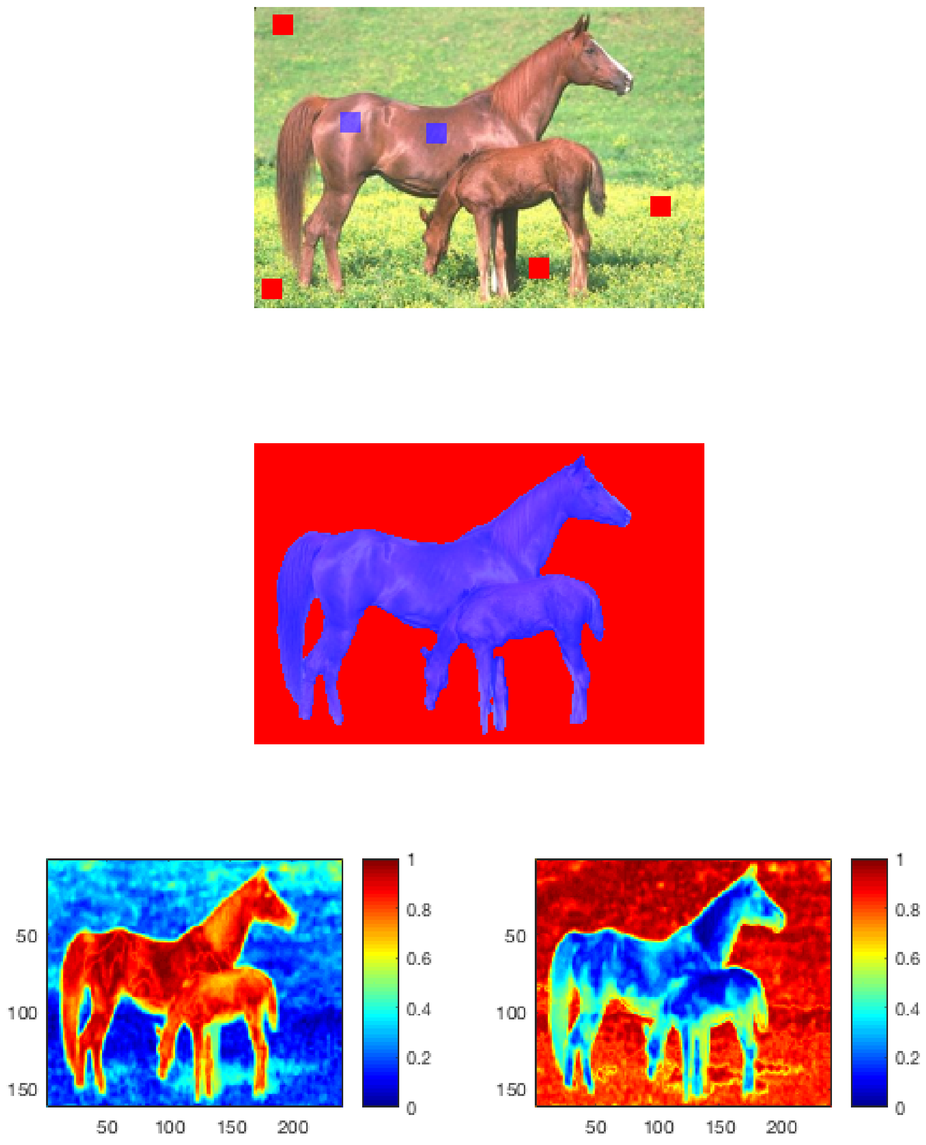
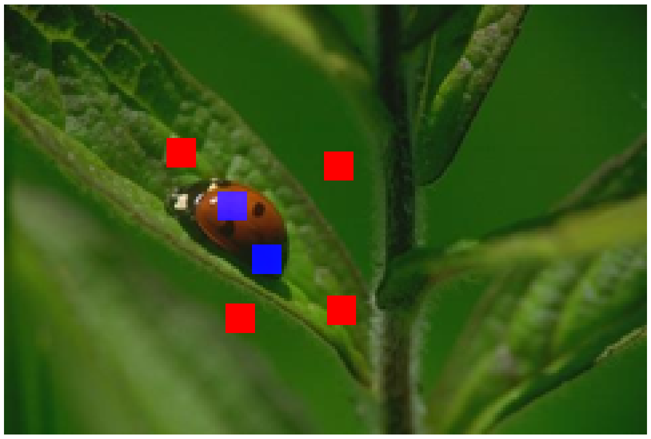
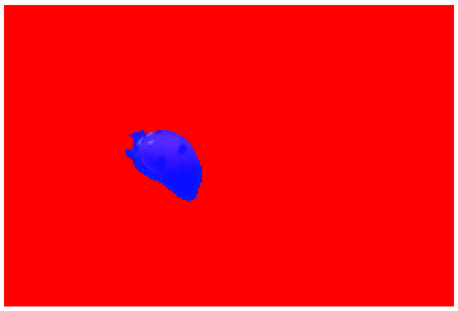


Figure 23: Horses. Top: the original image. Middle: clustering result. Bottom left: object feature. Bottom right: background feature. $\lambda = 100, C_{boost} = 0.7, C_{supp} = 4, t = 100$, Delta E only





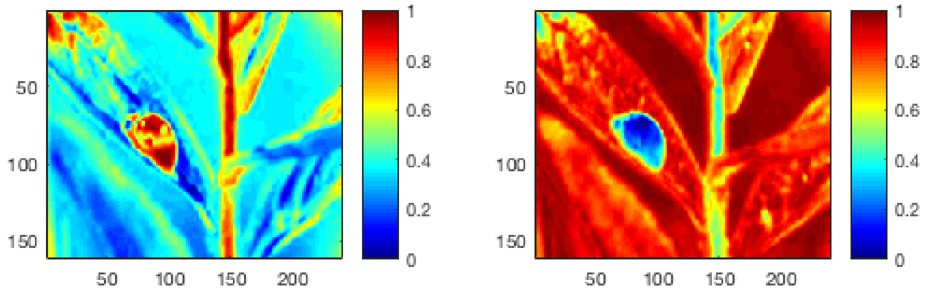


Figure 24: Ladybug. Top: the original image. Middle: clustering result. Bottom left: object feature. Bottom right: background feature. $\lambda = 100, C_{boost} = 0.7, C_{supp} = 4, t = 90$, Delta E only

7 Conclusion and future work .

In the presented work a diffusion-reaction equations based clustering algorithm was designed, implemented, theoretically and practically analysed. Despite the simple image features used for the numerical description of image properties, the algorithm has shown good results with synthetical and real world image tests.

The algorithm is semi-automatic and requires manual initialisation in form of placing a set of seed points to mark the object and the background areas. The result of the segmentation depends strongly on the initialisation conditions and requires a basic understanding of the algorithm principles. Because of this fact a few improvements were proposed to increase the predictability of the algorithm's dynamic and control over the results. The first improvement consists of using anisotropic diffusion instead of isotropic. This can help to avoid fast infiltration of the defender phase to the surrounding space and the attacker phase to the object area. The second improvement is the idea of using different feature maps for the diffusion and reaction parts. It can also help to better localize the correspondent phases within they working area.

Comparied to Chan-Vese algorithm the proposed approach provides greater level of interactivity and flexibility but requires adaptation of bigger amount of parameters, manual initialisation and manual stopping. Despite the significant amount of parameters the proposed algorithm can be adapted to a particular image domain.

The quality and performance of the algorithm could be improved by using better image features and more efficient numerical algorithms.

The model itself should be further simplified and adapted for concrete application fields. For example, the better and more simple forms of the function $b(x)$ could be analysed for this purpose.

Automatisation of the stopping point should be also considered. In this direction the global equilibrium states could be analysed.

I think the proposed clustering method can be used in cases where the interactivity is required. This could be for example a medical imaging or image editing software. The robustness against the noise could be also an important advantage of the described approach.

Image references

The photo of zebra: `https://de.wikipedia.org/wiki/Zebra#/media/File:Equus_grevyi_(aka).jpg`

The image of brain: `http://cn.mathworks.com/matlabcentral/mlc-downloads/downloads/submissions/27022/versions/1/screenshot.jpg`

Other photos used in this work were taken from "Berkeley Segmentation Dataset": `https://www2.eecs.berkeley.edu/Research/Projects/CS/vision/bsds/BSDS300/html/dataset/images.html`

References

[1] Mikal Rousson, Thomas Brox, Rachid Deriche. Active Unsupervised Texture Segmentation on a Diffusion Based Feature Space. RR-4695, INRIA. 2003.

[2] Nonlinear matrix diffusion for optic flow estimation. Thomas Brox, J. Weickert. Pattern Recognition (Proc. DAGM), Springer, LNCS, Vol.2449: 446-453, Sept. 2002.

[3] Weickert, J. "Anisotropic diffusion in image processing", Teuber Verlag, Stuttgart, 1998.

[4] Tony F. Chan, Luminita A. Vese "Active contours without edges" IEEE Trans. Image Processing 2001 Vol. 10, 266-277

[5] Tony F. Chan, Luminita A. Vese "A Multiphase Level Set Framework for Image Segmentation Using the Mumford and Shah Model" International Journal of Computer Vision 50(3), 271-293, 2002

[6] J. D. Murray, "Mathematical Biology: I. An Introduction, Third Edition", Springer-Verlag, Berlin, 1993

[7] Kass M., Witkin A., Terzopoulos. D, "Snakes: Active Contour Models ", International Journal of Computer Vision, 321-331 (1988)

[8] N. Champagnat, P.-E. Jabin and G. Raoul, Convergence to equilibrium in competitive Lotka-Volterra and chemostat systems, C. R. Math. Acad. Sci. Paris, 348 (2010), 1267-1272.

[9] M.W. Hirsch, Systems of differential equations which are competitive or cooperative. III. Competing species. Nonlinearity 1(1), 51-71 (1988).

[10] J. Bigun, G. Granlund and J. Wiklund. Multidimensional orientation estimation with applications to texture analysis and optical flow, in IEEE-PAMI, vol. 13, num. 8, p. 775-790, 1991.

[11] M. Jogenra Kumar, Dr. GVS Raj Kumar, R. Vijay Kumar Reddy "Review on image segmentation Techniques" International Journal of Scientific Research Engineering & Technology (IJSRET), ISSN 2278-0882 Volume 3, Issue 6, September 2014

[12] Luo MR, Cui G, Rigg B. The development of the CIE 2000 colour difference formula: CIEDE2000. Color Res Appl 2001;26:340-350

[13] Robertson, Alan R. (1990). "Historical development of CIE recommended color difference equations". Color Research & Application 15 (3): 167-170

[14] Jernigan ME, D'Astous F. "Entropy-Based Texture Analysis in the Spatial Frequency Domain", IEEE Trans Pattern Anal Mach Intell. 1984 Feb;6(2):237-43.

[15] Tomasz Weglinski, Anna Fabijanska "Survey of modern image segmentation algorithms on CT scans of hydrocephalic brains", Image Processing & Communication, vol. 17, no. 4, pp. 223-230

[16] V. Caselles, R. Kimmel, G. Sapiro, Geodesic active contours. International Journal of Computer Vision, Volume 22, Issue 1, pp. 61-79, 1997.

[17] S. Osher, J.A. Sethian, Fronts propagating with curvature dependent speed: Algorithms based on Hamilton-Jacobi Formulations, Journal of Computational Physics, Vol. 79, pp. 12-49, 1988

[18] Joachim Weickert. "Image Processing and Computer Vision. Lecture slides." Lecture slides. University of Saarland. Summer Term 2016. Online: http://www.mia.uni-saarland.de/Teaching/ipcv16.shtml

[19] Joachim Weickert. "Differential Equations in Image Processing and Computer Vision" Lecture slides. University of Saarland. Winter term 2016/2017. Online: http://www.mia.uni-saarland.de/Teaching/dic16.shtml

[20] Samarsky A. A. "Vvedeniye v chislennye metody" (Introduction to Numerical Methods) M.:Nauka, 1982

[21] Handbook on Computer Vision and Applications, Vol. 2: Signal Processing and Pattern Recognition. Academic Press, San Diego (1999) 423-450

[22] Oliver Labs, Frank-Olaf Schreyer. "Mathematik fuer Informatiker Teil 1,2 und 3" Lecture script. University of Saarland. Winter Term 2012/2013.