Segmentation Using Derformable Models

The main idea is to start from a initial segmentation (either contour or others) and then deform it towards the solution we are looking for. There are several instances of this method that allows to answer to main questions of segmentation:

- finding the best image partition into homogeneous regions
- finding the contours of an object
  Another important distinction is about the representation of this partition (the contour):
- parametric.
- implicit, using level sets. Basically we use a distance metric and the contour is a 0-distance.
  We need to define criteria in order to match them to find the right contour:
- contour info. We want to make the contour to be attracted to the highest gradient of the image.
- Region homogeneity: we look at what is inside the contour and we want that region to be homogeneous. We define criteria of homogeneity inside a contour.
- Regularity (internal constraint)
- balloon force, spatial relations, geometry... (external constraints)

The basic idea for all method in this domain is to express all of the criteria we have in an energy function that we want to minimize in order to find the contour.

Now we look at few approaches:

Parametric active contours

Define a contour as a Parametric representation of a contour and regularity of a contour.
It is called also "snake method" because you can see a contour like a snake that evolves in the image.
Principle: evolution of a curve under internal and external forces (one
object, parametric representation):

$$v(s) = [x(s), y(s)]^t, s \in [0, 1]$$

The x and y are coordinates of this parametric representation
We start from an initial position of this contour and then we make it evolve in order to minimize an energy function, which is defined like this:

$$E_{total} = \int_0^1 (E_{int}(v(s)) + E_{image}(v(s)) + E_{ext}(v(s)))ds$$

The energy here depends on the image (on the gradient of the image). It is composed by:
- image energy: with this we can make contour attracted to the image information (= gradient information on contours). Typically, we can choose it to be the norm of the gradient, in order to make it attract contours towards the high values of the gradient.

$$E_{image} = g(||\nabla f||)$$

- external energy: many pssibilities.
- internal energy: depends only on the shape of the contour but not on the image content, typically used for regularization.

$$E_{int} = \alpha(s)(\frac{dv}{ds})^2 + \beta(s)(\frac{d^2v}{ds^2})^2$$

it depends on regularization. First term is the first derivative of the curve: we are looking on how the points moves along the tangent of the curve; second term is the second derivative of the curve: it corresponds to the curvature. Classical results of geometry: we define the tangent as an entity with norm = 1 and this means that the scalar product of the tangent by its derivative is 0. Which means that the second derivative is orthogonal to the tangent. So it is something that is proportional to the contour, and the coefficient to the norm is the curvature:
if s = curvilinear coordinate, tangent $T = \frac{dv}{ds}$, $||T|| = 1$ and $\frac{dT}{ds} = kN$, k curvature
That explains that this term correspond to the tension (length of the curve and the curvature. Usually Eint is high for regular images, because that means that we want to force our segmentation to be regular, and low if the image presents some irregularities.

The way to solve the Etotal equation is to use le Euler-Lagrange equation:

$$\frac{\partial E}{\partial v} - \frac{d}{ds}(\frac{\partial E}{dv'}) + \frac{d^2}{ds^2}(\frac{\partial E}{\partial v''}) = 0$$

With this we can find the minimum of the E_{total} with respect to the position of the contour.
In our case, the Euler-Lagrange equation is applied like so:

$$-(\alpha v')'(s) + (\beta v'')''(s) + \nabla P(v) = 0$$

$$P(v) = E_{image}(v) + E_{ext}(v) F(v) = -\nabla P(v)$$

(aka, you have terms that depends on the first derivative, terms that depends on the second derivative and the the variation of the energy image and the external energy).
In general alpha and beta are constant but it depends.
Since the digital images are discrete, we have to discretize this solution using finite differences (discretization of the derivatives):

$$V^t = [v_0^t, v_1^t, v_2^t, \ldots\ldots, v_{n-1}^t]t$$

If alpha and beta are constants, we get 5 successive points on every position of the contour:

$$\frac{\beta}{h^2}v_i + 2 - (\frac{\alpha}{h} + 4\frac{\beta}{h^2})v_i + 1 + (2\frac{\alpha}{h} + 4\frac{6\beta}{h^2})v_i - (\frac{\alpha}{h} + 4\frac{\beta}{h^2})v_{i-1} + \frac{\beta}{h^2}v_{i-2} = F(v)$$

The equation above can be expressend in a matricial form:

$$AV = F$$

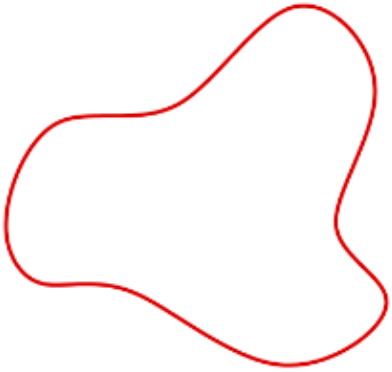
with A as the pentadiagonal matrix that depends only on the coefficient of the energy function, V is the vector that describe the position of the ponts in the contour, F represents all the forces you want (typicall the gradient and some external forces).

Another interpretation of the resolution of the Euler-Lagrange equation is to consider that contours evolve in time and so we can derive a dynamic scheme with inertia in ordeer to avoid to have too large steps (aka this is a different way to consider the iteration):

This is the evolution of contours over iteration (aka over time), weighted by the inertia, which follows the differential equation we have seen before. What it changes is that the A matrix is the same of before, but with some terms on the diagonal which are the inertial parameters. So A could have 0 eigenvalues or very small eigenvalues (at least 2) (eigenvalues could cause some problems during the inversion, so we want to avoid them). With this small numbers of eigenvalues we are sure that the mtrix is invertible and so we don't go towards instability regions. We have to find a compromisation on the parameters, because if we have too many eigenvalues, we could go towards instability regions, while if we have too small number of parameters, we would not go towards the region we want.
The method is starting from an initialization and then applying this scheme iteratively with a stop criteria (for example if after an iteration there are no more chages or too few changes). The initialization is very important, because we do not garantee convergence towards a global minima of the energy function, but just towards a local minima, that depends on the starting point. Also, the initialization indicates which objects we are interested in.

Let's look at some examples:
This matrix A has only 5 non zero terms (in the case that alpha and beta are constants). If the contour are closed, we have a loop (in this case, there is a starting beta and an ending beta). It is a matrix with a lot of zeros so we could have problems during the inversion.

Types of active contours and associated matrices

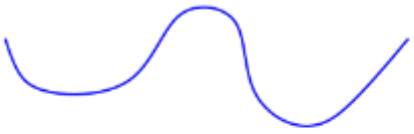


Closed active contour

$$
\begin{bmatrix}
2\alpha+6\beta & -\alpha-4\beta & \beta & 0 & 0 & \beta. & -\alpha-4\beta \\
-\alpha-4\beta & 2\alpha+6\beta & -\alpha-4\beta & \beta & 0 & \dots & \\
\beta & -\alpha-4\beta & 2\alpha+6\beta & -\alpha-4\beta & \beta & & \\
0 & \beta & -\alpha-4\beta & 2\alpha+6\beta & -\alpha-4\beta & & \\
0 & 0 & \beta & -\alpha-4\beta & 2\alpha+6\beta & & \\
\dots & \dots & \dots & & & \dots & \\
-\alpha-4\beta & \beta & 0 & 0 & \beta & -\alpha-4\beta & 2\alpha+6\beta
\end{bmatrix}
$$

If the contour is not close, we still have the same general terms (look at the rows), but there are changes in the corner of the matrix and we could have different expressions in the corner depending wethere the ending points are fixed or free.

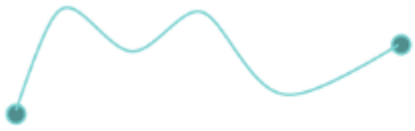
Free end-points



$$\begin{bmatrix}
\beta & -2\ \beta & \beta & 0 & 0 & 0 & 0 \\
-\alpha-2\beta & 2\alpha+5\beta & -\alpha-4\beta & \beta & 0 & \cdots & \\
\beta & -\alpha-4\beta & 2\alpha+6\beta & -\alpha-4\beta & \beta & & \\
0 & \beta & -\alpha-4\beta & 2\alpha+6\beta & -\alpha-4\beta & & \\
0 & 0 & \beta & -\alpha-4\beta & 2\alpha+6\beta & & \\
\cdots & \cdots & \cdots & & & \cdots & \\
 & & & \beta & -\alpha-4\beta & 2\alpha+5\beta & -\alpha-2\beta \\
0 & 0 & 0 & & \beta & -2\ \beta & \beta
\end{bmatrix}$$

(with $v''(0) = v''(1) = v'''(0) = v'''(1) = 0$)

Fixed end-points



$$\begin{bmatrix}
2\alpha+6\beta & -\alpha-4\beta & \beta & 0 & 0 & 0 & 0 \\
-\alpha-4\beta & 2\alpha+6\beta & -\alpha-4\beta & \beta & 0 & \cdots & \\
\beta & -\alpha-4\beta & 2\alpha+6\beta & -\alpha-4\beta & \beta & & \\
0 & \beta & -\alpha-4\beta & 2\alpha+6\beta & -\alpha-4\beta & & \\
0 & 0 & \beta & -\alpha-4\beta & 2\alpha+6\beta & & \\
\cdots & \cdots & \cdots & & & \cdots & \\
 & & & \beta & -\alpha-4\beta & 2\alpha+5\beta & -\alpha-2\beta \\
0 & 0 & 0 & & \beta & -2\ \beta & \beta
\end{bmatrix}$$

Example:

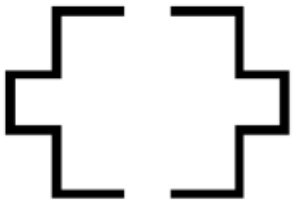
On the left we have the starting point (aka, the initial position of the contour). On the right the final contour after applying the iterative scheme. Here it worked really well because we started from a contour that was very closed from what we wanted as output.

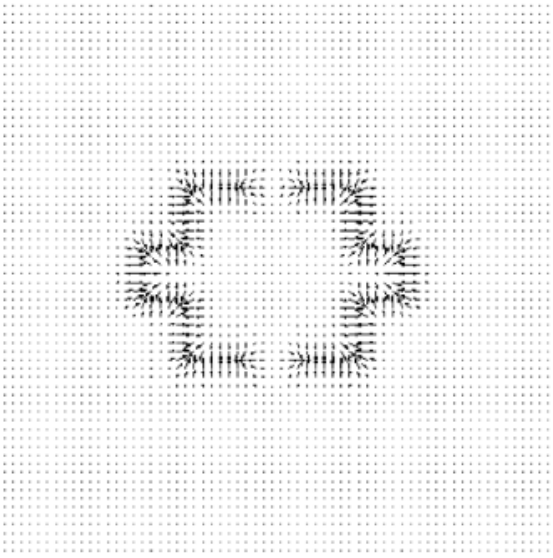
Let's look at the external energies that we can use.

Balloon force

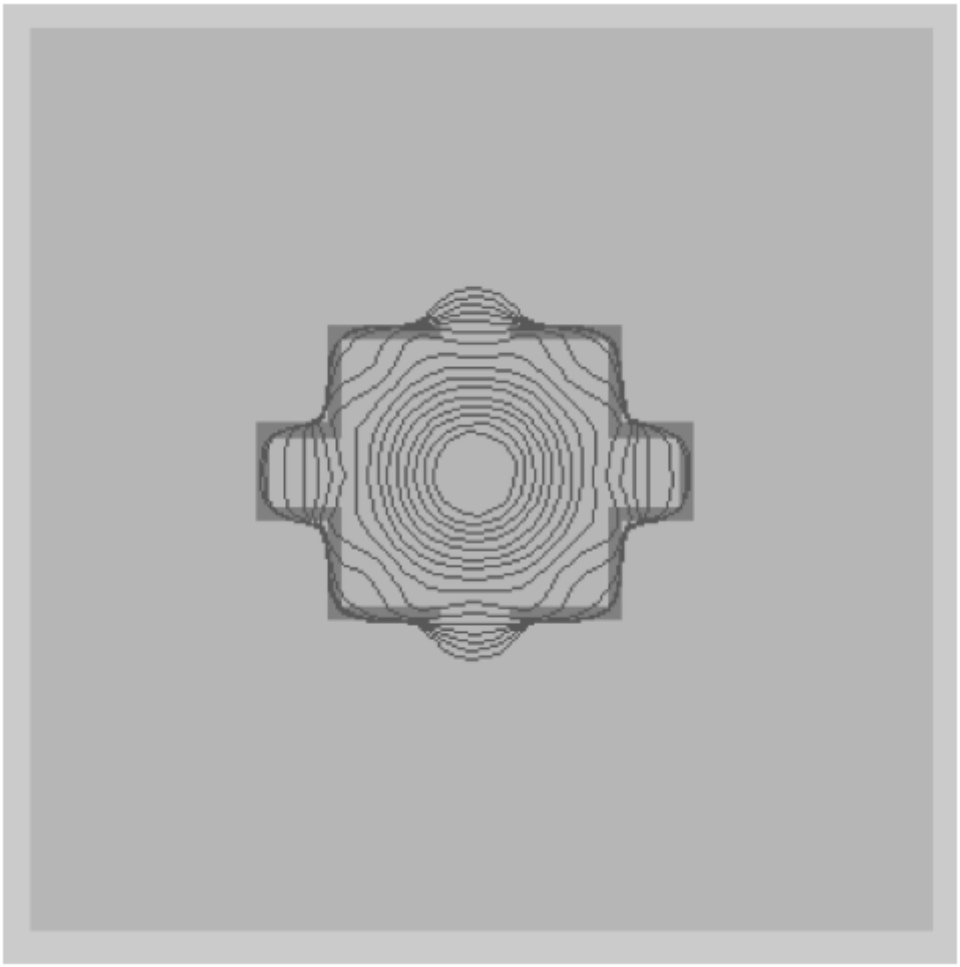
It is called "balloon force", because we can force the contours to increase/decrease the contours like a balloon. In fact, we can have two general problems:

- bad initialization ⇒ no attraction
- no forces ⇒ curve collapses
  One way to solve these problem is to add some external forces suc as the balloon force along the norm of the contour with positive parameters to force the contour to increase, if the starting point was inside the object, or to decrease, if the starting point was outside of the object:
  $k_1 N(s)$
  with $N(s)$ as the unit normal vector at s.
  If we use the balloon force we don't necessarily need the initialization very close to the searched object.
  Example:





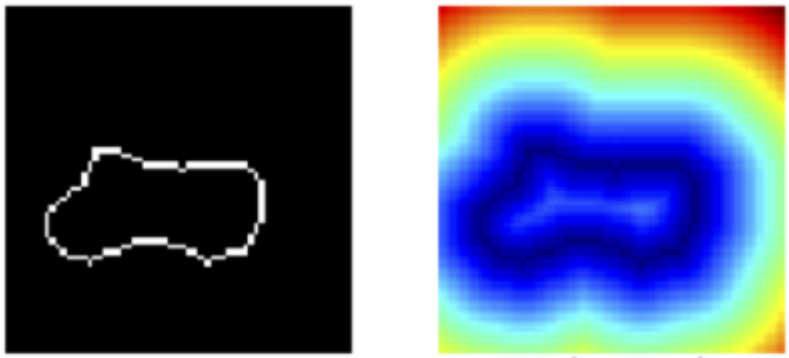
On the right: starting image, on the left: gradient



Application of the minimization of the total energy with balloon force. Note that in the parts of the holes we do not have info for the total energy, but only the balloon force that tells the algorithm how regular should the segmentation be.

Constraint on distance to some edges

We can achieve the precedent result with a distance map: we want to force the contour to be as 0 distance.



$D(x,y)$

Distance map $D(x,y)$ ⇒ potential

$$P_{dist}(x,y) = we^{-D(x,y)}$$

$$F_{ext} = -\nabla P_{dist}$$

Gradient vector flow (Xu et Prince, 1997)

This is an approach that modify the gradient in order to impose some attractions even far from the objects. For instance in the previous distance map image we have the original image that is binary, so if we take any point in the black part that is far from the object, the gradient would be weak and so there won't be any attraction.
To avoid this we can diffuse the gradient al over the image. We can build a modified version of the gradient map in order to minimize the energy defined here:

$$E = \int\int \mu(u_x^2 + u_y^2 + v_x^2 + v_y^2) + |\nabla f(x,y)|^2 |\vec{v} - \nabla f(x,y)|^2 dxdy$$

where f is the contour map. Specifically we want to minimize E for

$$\vec{v}(x,y) = (u(x,y), v(x,y))$$

Note that we want that, looking at the last part of the equation, the term $\vec{v}$ to be closed as possible (in terms of distance) to $\nabla f(x,y)$ when $\nabla f(x,y) \neq 0$ and the first part $\mu(u_x^2 + u_y^2 + v_x^2 + v_y^2)$ is a regularization term that forces the diffused gradient to not be zero outside the contour.
This equation is solved in the following iterative way:

Solving and generalizing GVF

$$\mu\nabla^2 u - (u - f_x)(f_x^2 + f_y^2) = 0$$
$$\mu\nabla^2 v - (v - f_y)(f_x^2 + f_y^2) = 0$$

More general formulation:

$$\frac{\partial v}{\partial t} = g(\|\nabla f\|)\nabla^2 v - h(\|\nabla f\|)(v - \nabla f)$$

$$v(x, y, 0) = \nabla f(x, y)$$

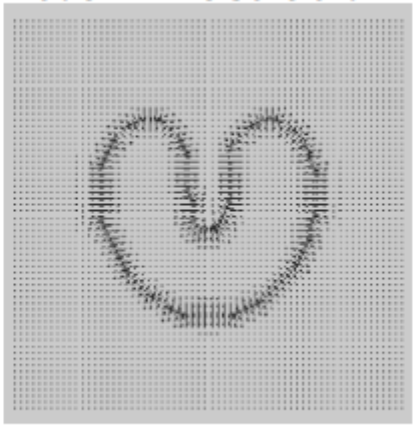
Examples for functions g and h:

- $g(r) = \mu$, $h(r) = r^2$
- $g(r) = \exp(\frac{-r^2}{k})$, $h(r) = 1 - g(r)$

Example application:

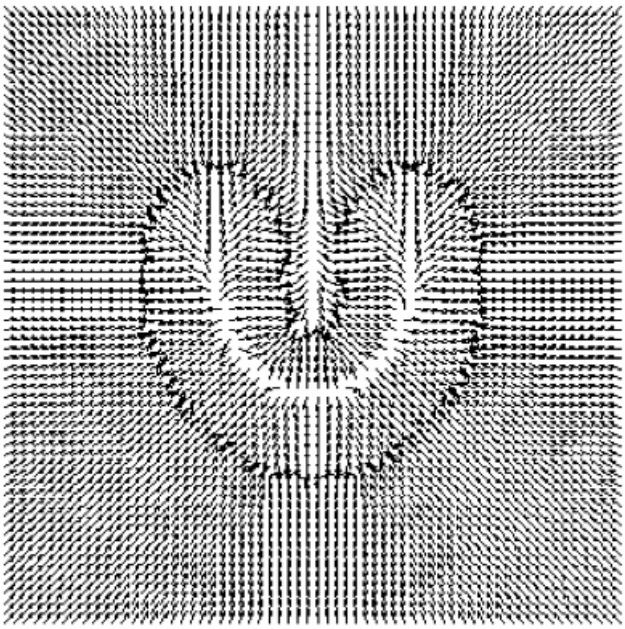
Normal method:

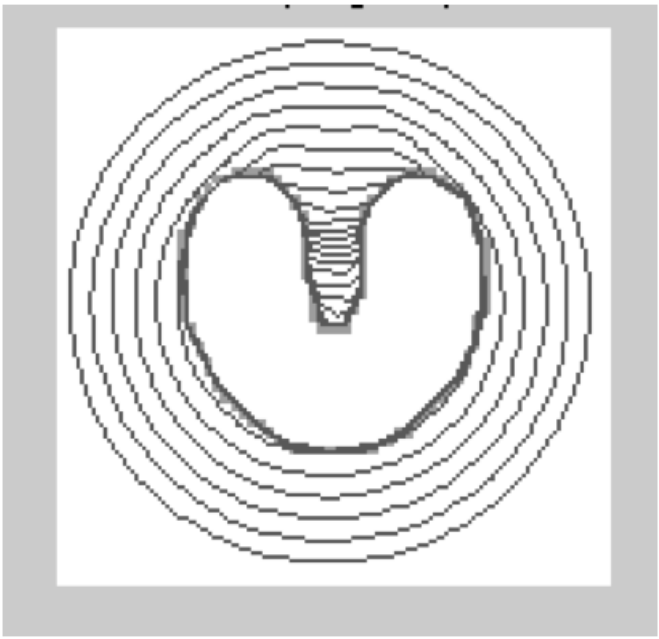
Classical active contour: illustration



We can see from the gradient map that the gradient is diverse from zero only near the true contour of the objects (and this is a problem) because the image is homogeneous inside and outside the object. If we don't use the gradient flow external force, we can see that the contour converges quite well near the true contour, but it fails in the strong concavity.
Gradient flow:

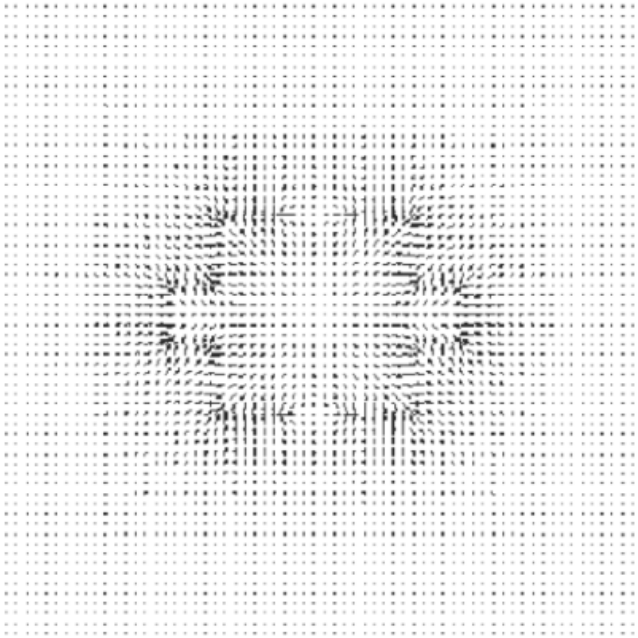
Using GVF:

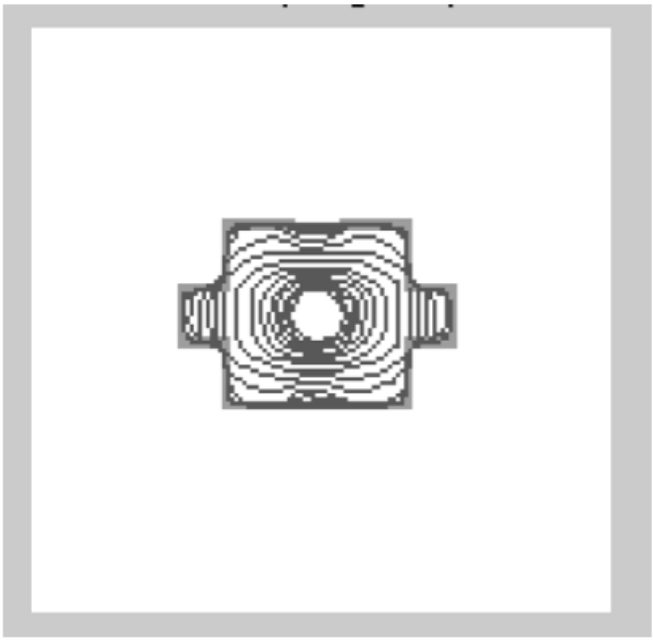




The gradient is not zero in homogeneous regions. In the right image we have an example where the initialization is pretty far from the true contour, but we can achieve really good results. Another example:

Using GVF:





It is even better than the balloon force.

3D parametric deformable models

We can apply the same approach on 3D objects: instead of having a line as a contour, we have a 2D surphace, so we have 2 parameters that describe the contour.

The total energy for 2D contour is defined as:

$$E(v) = \int_\Omega w_{10}\|\frac{\partial v}{\partial r}\|^2 + w_{01}\|\frac{\partial v}{\partial s}\|^2 + w_{20}\|\frac{\partial^2 v}{\partial r^2}\|^2 + w_{02}\|frac\partial^2 v\partial s^2\|^2 + 2w_{11}\|\frac{\partial^2 v}{\partial r \partial s}\|^2 drds + \int_\Omega P(v)d$$

It is composed by:

- first term = first order derivatives: elastic membrane (curvature)
- second term: second order derivatives: thin plate (torsion)
- P: attraction potential
  To solve the equation we have a similar iterative scheme as in 2D

Geodesic active contours (Caselles, 1997)

The idea is to slightly modify the energy function: in fact, in general the first derivative is similar to the second derivative. So in the end our energy function is composed by only 1 first order derivation and the gradient information.

$$J_1(v) = \alpha \int_a^b |v'(s)|^2 ds + \lambda \int_a^b g(|\nabla I(v(s))|)^2 ds$$

Minimizing J1 is equivalent of minimizing J2, defined as follow:

$$J_2(v) = 2\sqrt{\lambda\alpha} \int_a^b |v'(s)| g(|\nabla I(v(s))|) ds$$

The fact that it is called geodesic is because the J_2 can be seen as a metric connected to the image info (not euclidean metric).

We can have an evolution equation, aka the equation of J_2 in the time perspective (as we did before):

$$\frac{\partial v}{\partial t} = g(I)\kappa N - (\nabla g * N)N$$

The first term depends on the curvature, while the second term depends on the gradient.

Since we are using the "geodesic metric", we can compute easily the evolution equation with morphological operators: moving the contours along he norm can be seen as a dilation of what we had in the previous iteration (morphological active contours).

Level sets

We change the representation of the contour to an implicit representation (not a parametric one as before). We want to to define the contour as a 0 level set of some function in a higher dimensional space.
Let $\Gamma(t)$ be a closed hyper-surface (dimension $d-1$)
Let ψ (dimension d) be a function taking values in R with

$$\Gamma(t) = x \in \mathbb{R}^d | \psi(x,t) = 0$$

propagation of Γ (evolution along the normal) $\Leftrightarrow$ propagation of ψ

Instead of making the contour evolve, we make this function ψ evolve and for each iteration the contour would be the 0-level-set of this function.
As before we can use an iterative scheme to evolve ψ and again we can evolve it in the time-perspective as:

$$\frac{\partial \psi}{\partial t} = -F||\nabla \psi||$$

where:

- N is the norm $N = \frac{\nabla \psi}{||\nabla \psi||}$: it is the gradient of ψ normalized
- k is the mean curvature $k = div(\frac{\nabla \psi}{||\nabla \psi||})$: the divergence of the norm, so we have movement along the norm

An example of ψ is the distance map:

Propagation using level sets

- Propagation speed:

$$F = (F_A + F_G)k_I$$

  - F_A expansion or contraction, independent of geometry,
  - F_G geometric properties (curvature),
  - k_I stopping criterion (image).

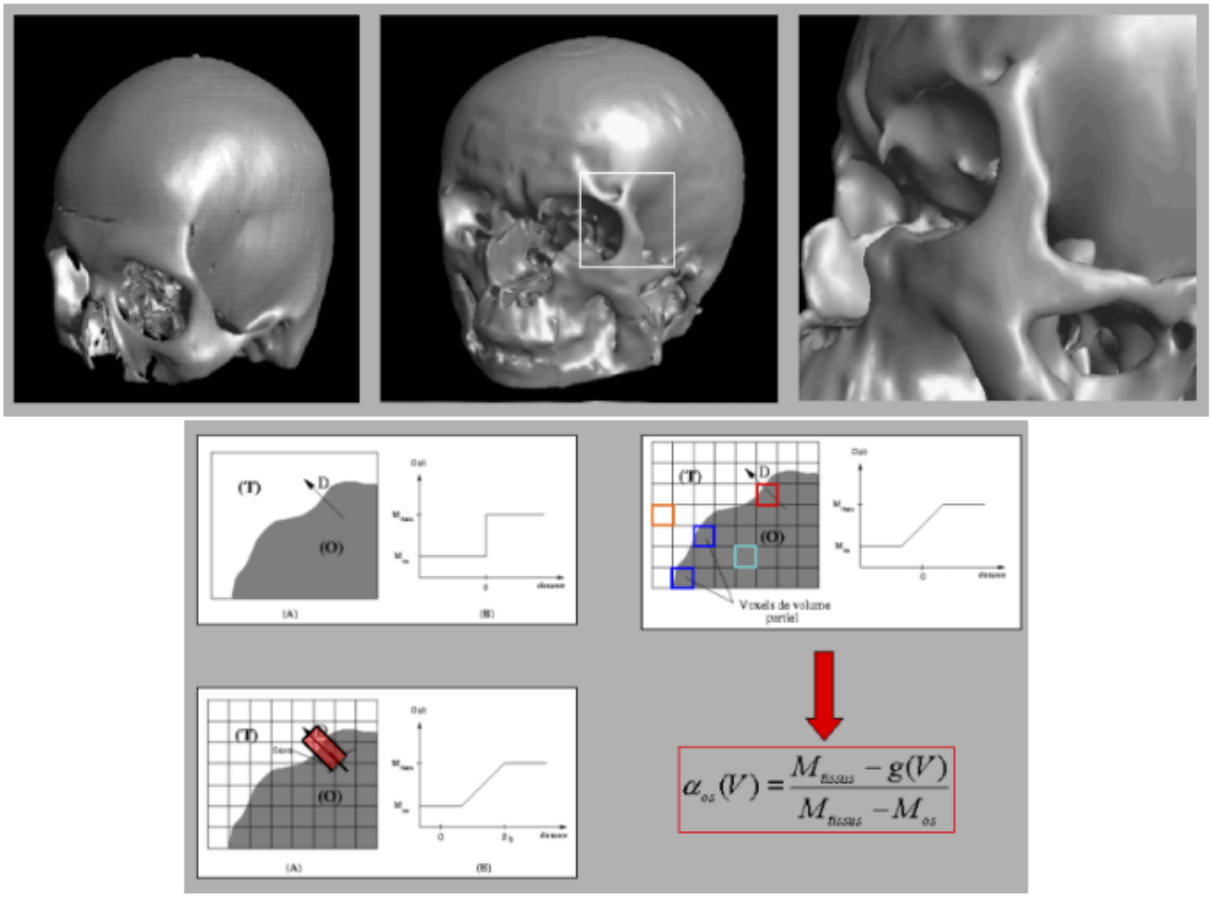
$$F = \frac{1}{1 + ||\nabla G_\sigma * I||^p}(\pm 1 + \epsilon\kappa)$$

- Advantage: potential modification of topology.
- Speeding-up by computation only in a narrow band.

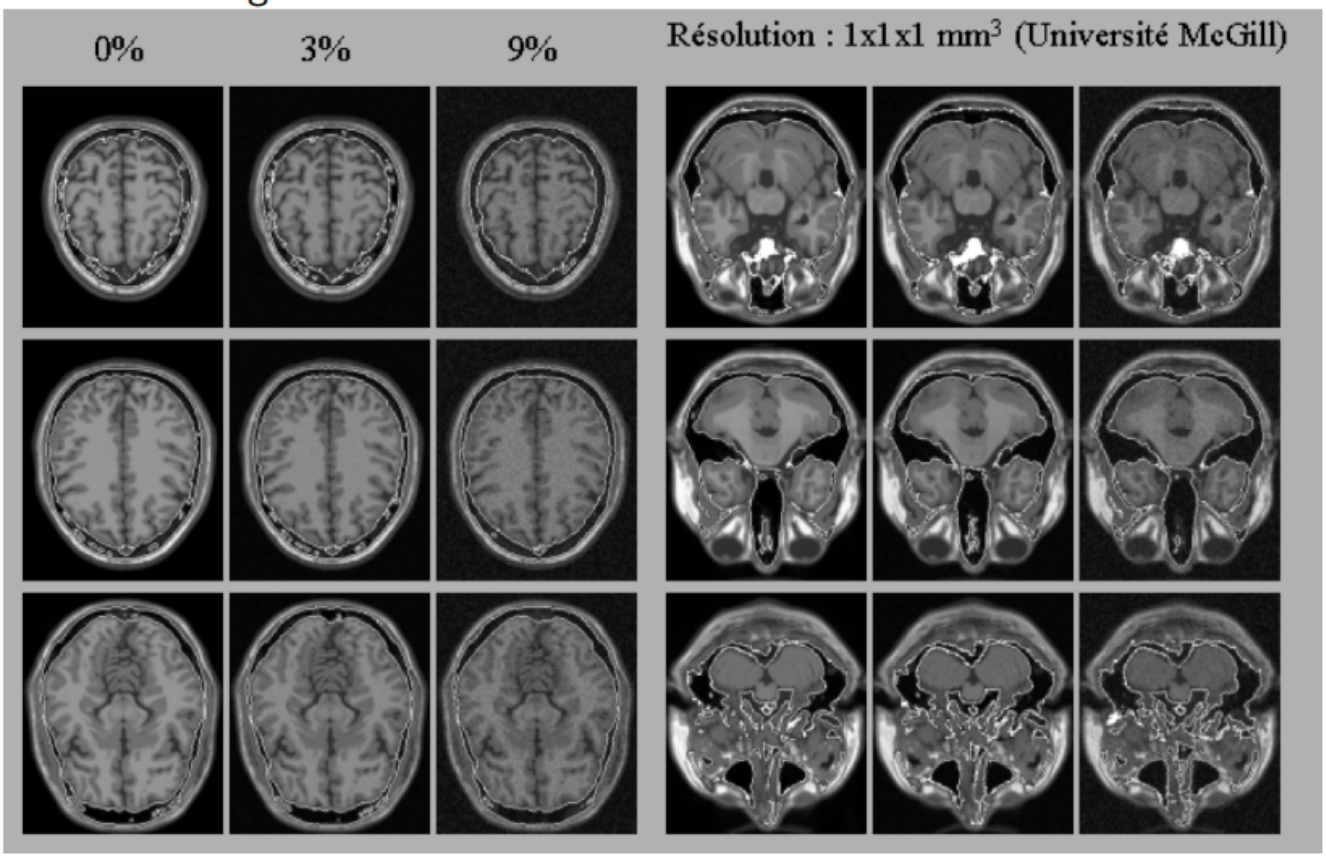
The advantage is that we can always have a 0-level-set, without caring on the changes of the topology of the image. For instance, in the following image, we have the same object, but with different level sets:

With the parametric approach it would have been difficult because we would have to create 2 parametrization on each level-set, while it is completely natural with the level-set approach.

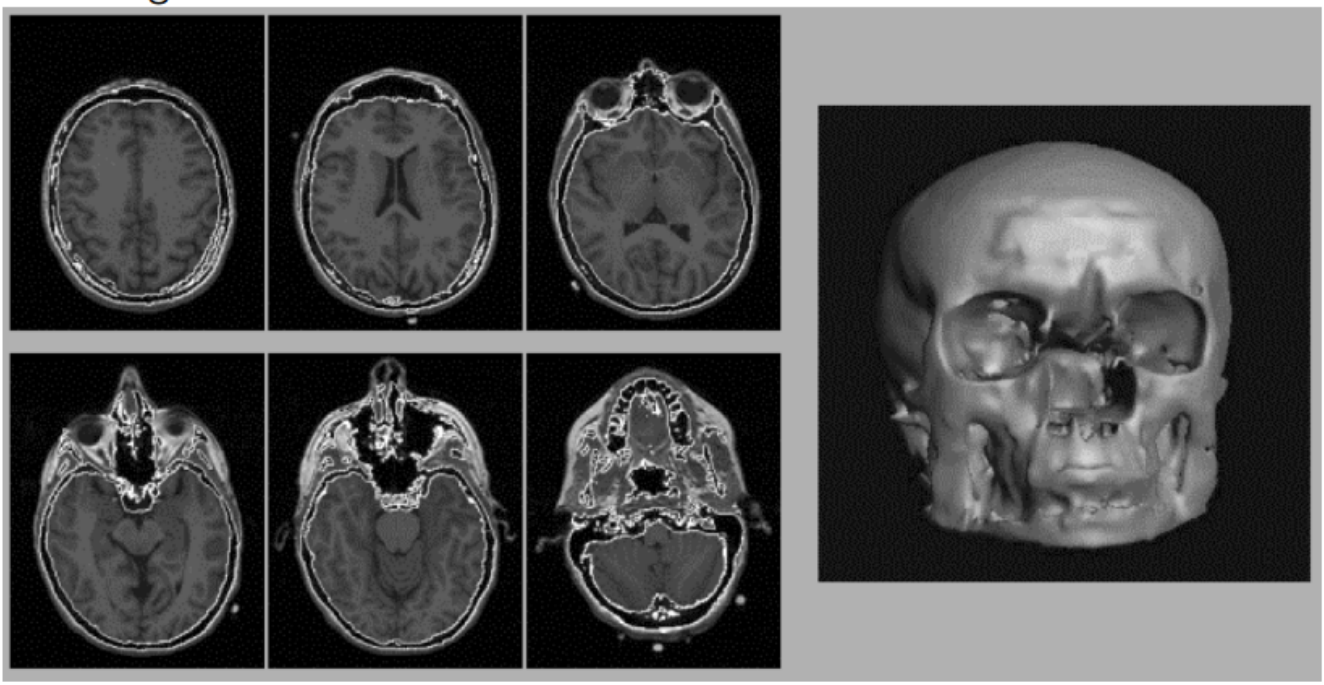
Example: bone segmentation in MRI (H. Rifai)



Simulated images:



Real images:



Evolution

Result

The counter part is that we cannot control the topology of the image.

Region-based approach: Mumford and Shah (1989)

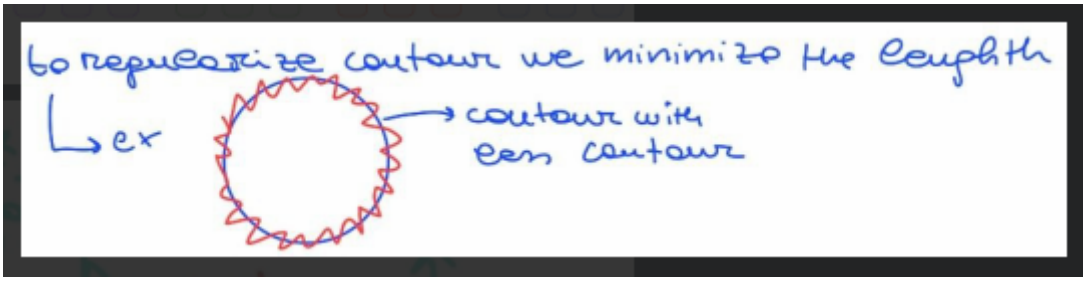
The goal of this approach is tho find differents contours for different objects of the image.
We start by dividing the image in different partitions, using implicit representation, using criteriaon region homogeneity instead of contour informations.

The idea of Mumford and Shah is the model the image to make it approximate the regions by a smooth function. We want to find a partition of the image composed by some regions that are delimited by some contraints and smoothed by a smooth function. The best approximation of this smooth function is the minimum of this energy function:

$$U(\Gamma, g, f) = \lambda \int \int_{I \setminus \Gamma} (f(x,y) - g(x,y))^2 dxdy + \mu \int \int_{I \setminus \Gamma} \|\nabla g(x,y)\|^2 dxdy + \nu \int_{\Gamma} dl$$

It is composed by:

- first term: the g (approximation) should be closed to f (original image). If we ignore this term, we don't have any info on the object so we won't have any contour
- second term: g should be smooth (aka it should have small variations). For that we minimize the norm of g. If we ignore this term, we cannot guarantee that we would obtain a smooth contour.
- third term: controls the length of the contour. This is a way to regularize the contour, because minimizing the contour length is a way of minimizing the curvature of the contour:



The way we use this method for a segmentation task is to say that we want the g to be constant in each region. Then the equation simplifies a lot:
if $g_i = const$ on each R_i then:

$$U_0(\Gamma, f) = \sum_i \lambda_i \int \int_{R_i} (f - g_i)^2 dxdy + \nu \int \Gamma dl$$

$$\Rightarrow gi = \frac{1}{s_i} \int \int R_i f(x,y) dxdy$$

where s_i is the area of R_i
Note that the norm of the g would be 0, so we have only have the first and third terms of the original equation. What we get is a partition of space into homogeneous regions, characterized by their average gray level.

To solve this equation, we want to convert each region that are defined piece-wise, into a function that applies to every other image (so everywhere in the image), which means that we merge the output computation.

Let's consider the case where we have an image with only 2 regions:

- $R_1 = int(\Gamma)$
- $R_2 = ext(\Gamma)$
  with constant values $g1$ and $g2$. The following partition equation would be:

$$U(\Gamma, g, f) = \lambda_1 \int\int_{R_1} (f - g_1)^2 dxdy + \lambda_2 \int\int_{R_2} (f - g_2)^2 dxdy + \nu \int \Gamma dl$$

  To solve this equation, we use this definition of level set:

$$\phi(x, y) \begin{cases} = 0 & \text{on } \Gamma \\ > 0 & \text{in } R_1 = int(\Gamma) \\ < 0 & \text{in } R_2 = ext(\Gamma) \end{cases}$$

where φ is:

- 0 on the contour (defined as $\Gamma(t) = \varphi(t) = 0$)
- positive inside the region
- negative outside the region.
  As we have seen before, we can express φ as an evolution equation over time (mean curvature motion):

$$\frac{\partial \varphi}{\partial t} = |\nabla \varphi| \nabla (\frac{\nabla \varphi}{|\nabla \varphi|})$$

$$\varphi(0) = \varphi_0$$

Note that we can extend this approach it in more than 2 regions by using multiple of this level-set functions.

After we defined the phi, we defined more specifically for our case by:

$$H(z) = \begin{cases} 1 & \text{if } z \geq 0 \\ 0 & \text{if } z < 0 \end{cases} \qquad H'(z) = \delta(z)$$

with,

$$H'(z) = \delta(z)$$

$$\Rightarrow \int_\Gamma dl = \int_I |\nabla H(\varphi)| dxdy = \int_I \delta(\varphi)|\nabla\varphi| dxdy$$

and we re-write the energy equation as:

$$U(\Gamma, g, f) = \lambda_1 \int\int I(f - g_1)^2 H(\varphi) dxdy + \lambda_2 \int\int I(f - g_2)^2 (1 - H(\varphi)) dxdy + \nu \int_I \delta(\varphi)|\nabla\varphi| dxdy$$

What we did is transposing the integral of each singular region into the integral of the whole image:

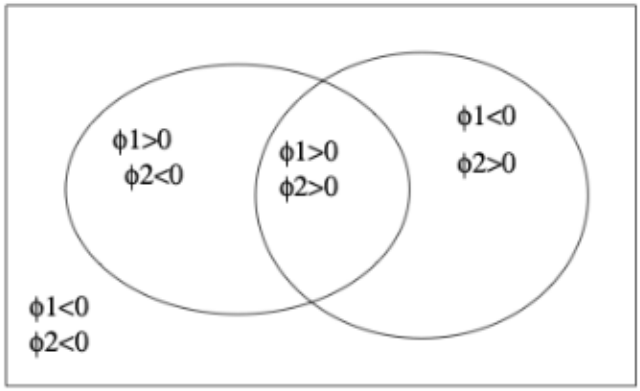
- the first term is $\neq 0$ only for the region R_1
- the second term is $\neq 0$ only for the region R_2
- the length of the contour applies for both of the regions, and it is defined as the norm of the gradient of the φ
  To minimize the energy U means calculating:

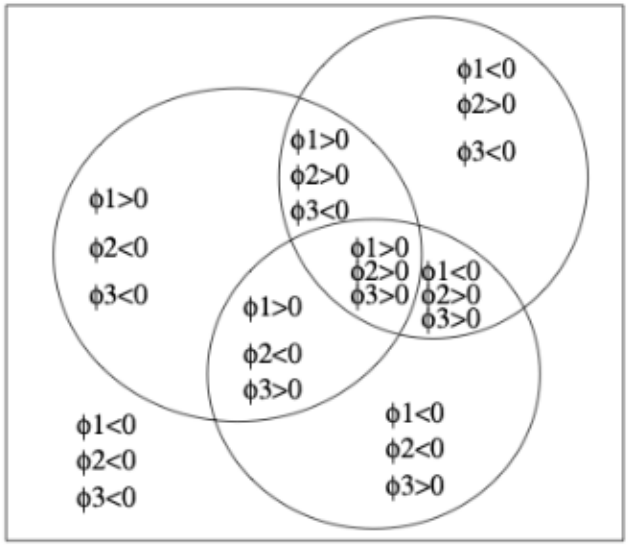
$$g_1 = \frac{\int_I f H(\varphi) dxdy}{\int_I H(\varphi) dxdy}$$

$$g_2 = \frac{\int_I f(1 - H(\varphi)) dxdy}{\int_I (1 - H(\varphi)) dxdy}$$

$$\frac{\partial\varphi}{\partial t} = \delta(\phi)[\nu\nabla(\frac{\nabla\varphi}{|\nabla\varphi|}) - \lambda_1(f - g_1)^2 + \lambda_2(f - g_2)^2]$$

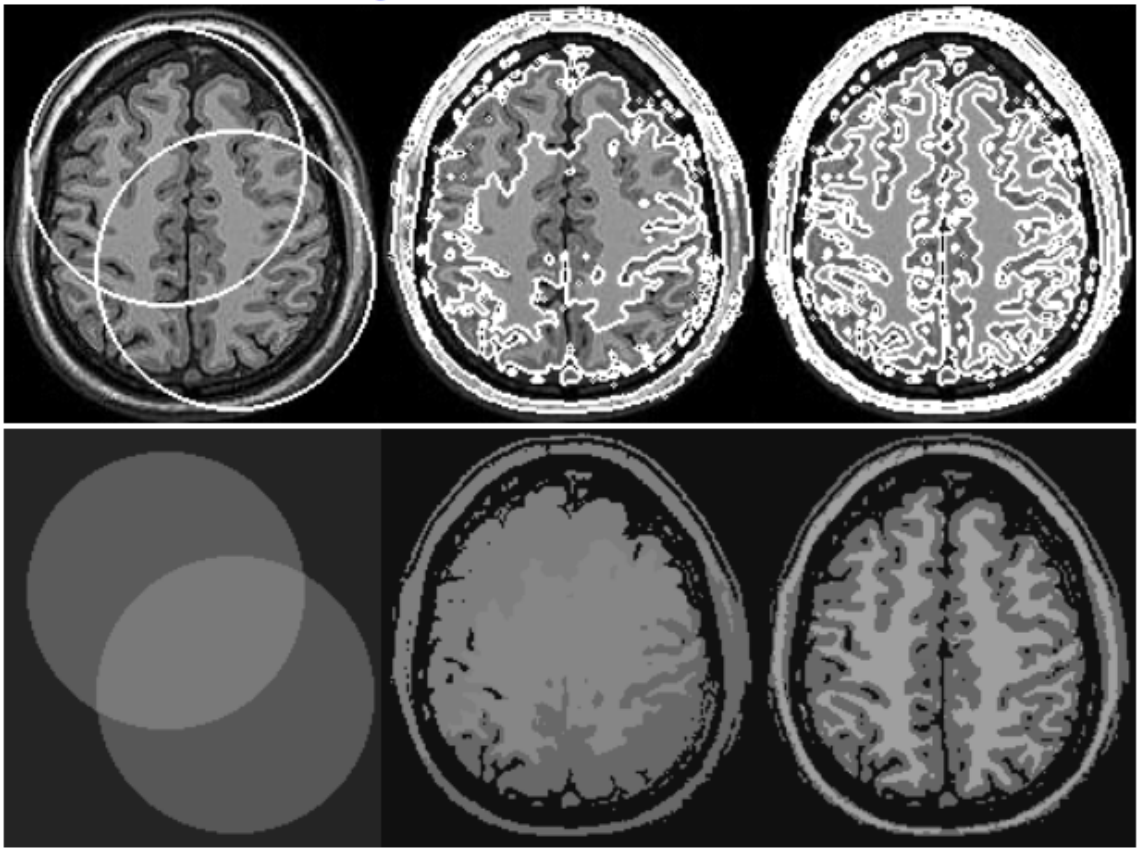
  In practice we get a smooth version of δ and H. It is a kind of competition of the different phis. It is an iterative scheme:
- start from an initial partition
- we make phi evolve, using the U equation
- compute the average values
- make phi evolve again
- .... convergence





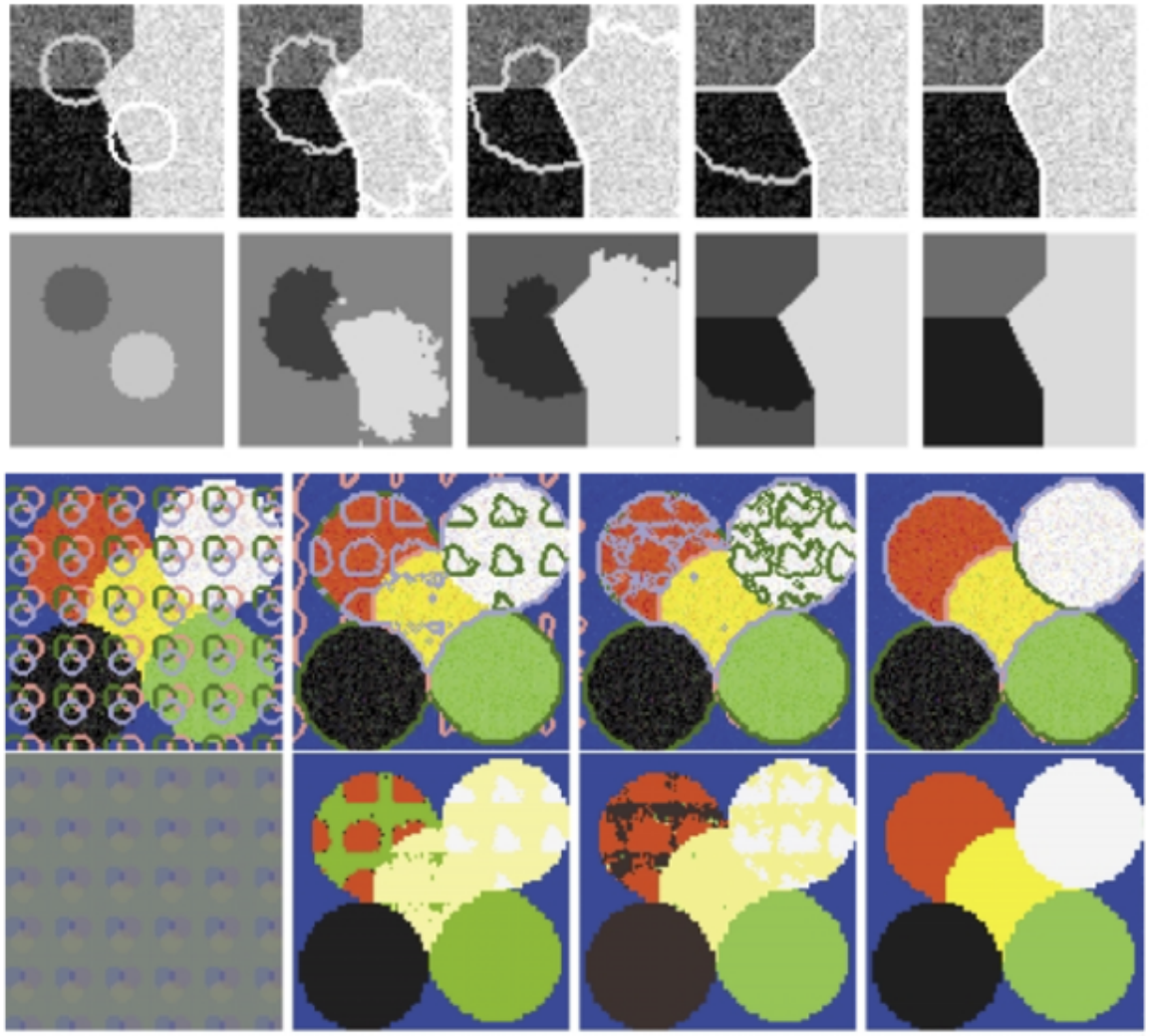
Example:

Segmentation of a MRI image with two level sets



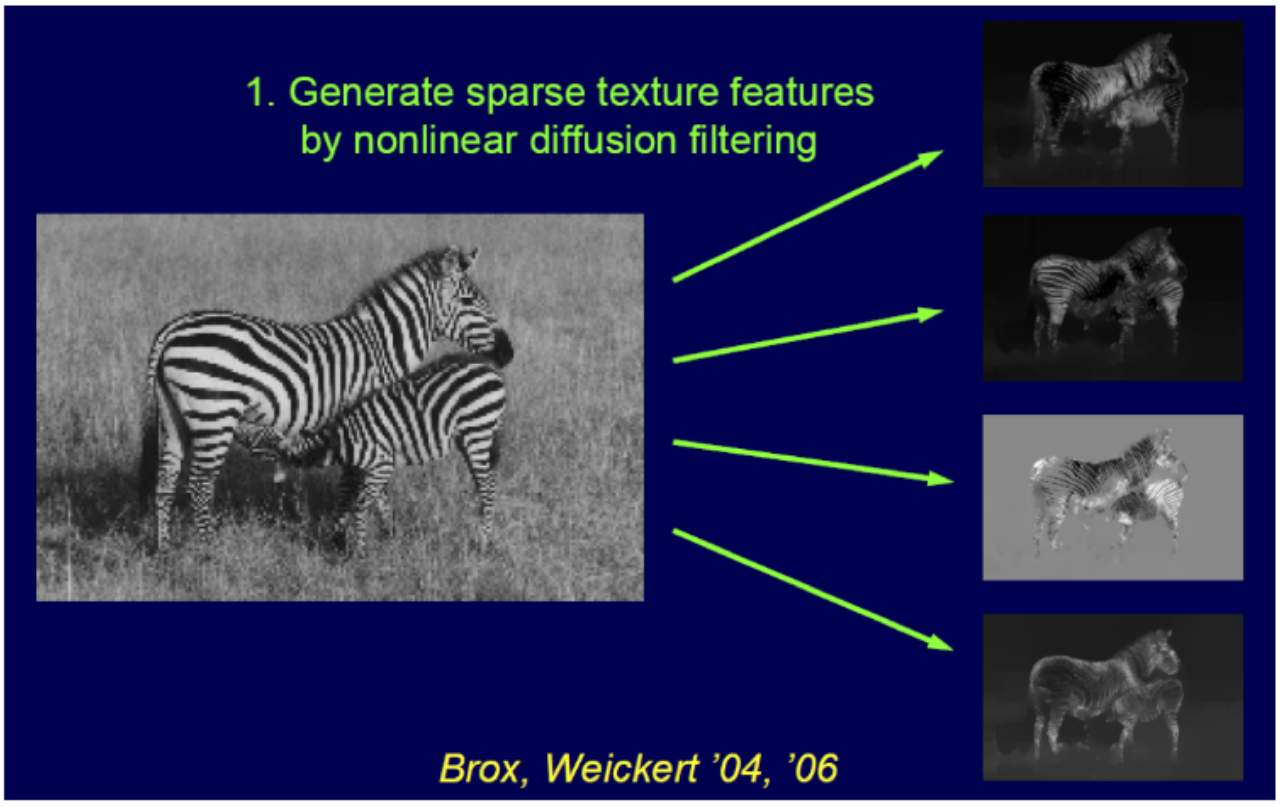
Note that with 2 level sets functions φ we can derive 4 different regions. With 3 level sets functions we can derive 8 regions, ... But we don't get only even number of regions, but also odd number of regions because it can happen that a region disappears.

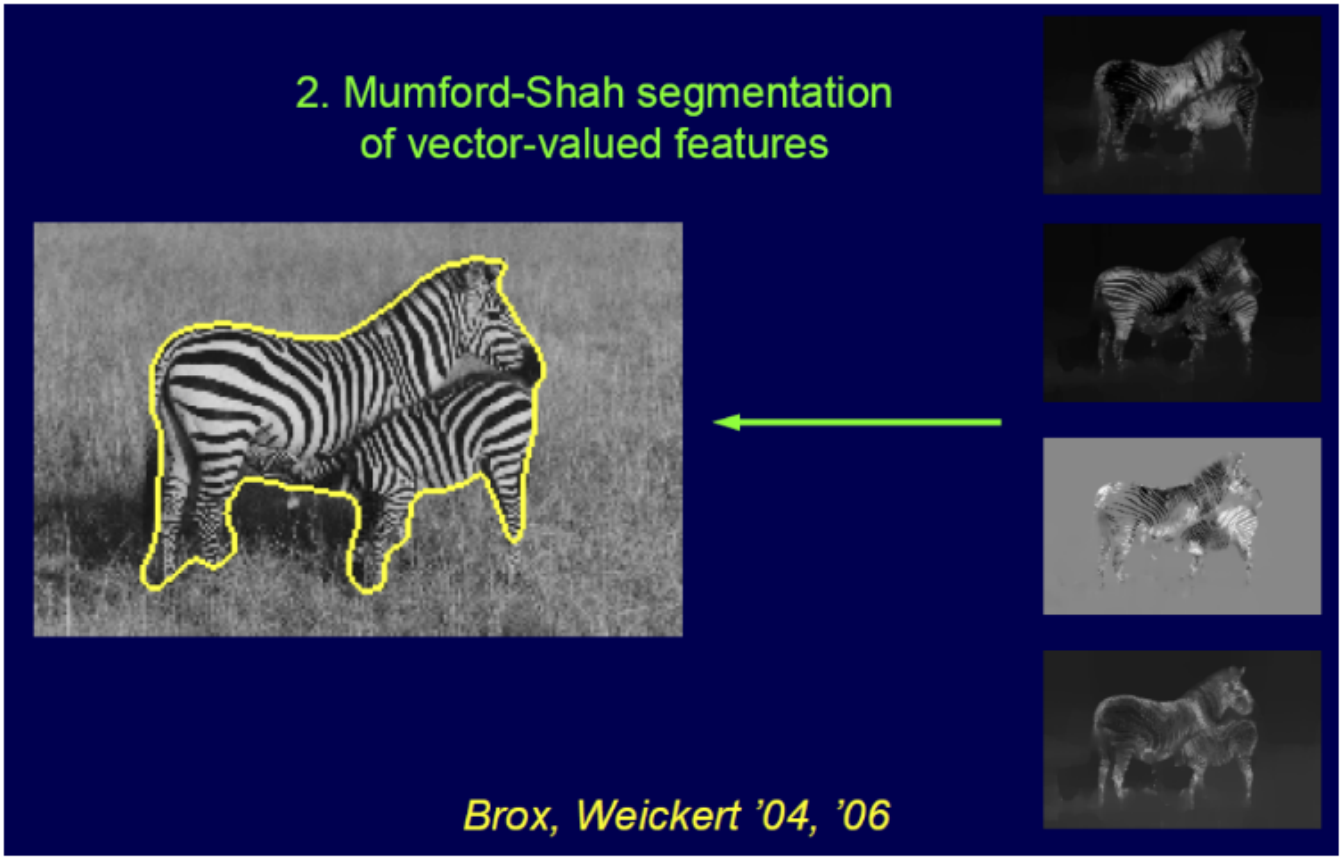
Example with junctions



This shows that we don't need to have just 1 level set function for every objects but we could have multiple phi functions for each region we want to segment. We could need this when we want to segment regions that have really high curvature, so that we can optimize their segmentation by taking the intersection of different phis "contours".

Texture segmentation





Here there is an example where the g_i is not constant. We express this more formally here:

Let's take the example of an image where we have 2 different regions (aka, 2 different textures or 2 different distributions). We want to maximize the probability of this partition given the image informations. Given partition $P(\Omega) = \Omega_e, \Omega_i$, if we use Bayes rules this probability that we want to maximize is proportional to the probability of the product of the likelihood (aka, the probability of the image, given the partition), the prior probability of the partition: $p(I|P(\Omega))p(P(\Omega))$.
We want to compute these 2 terms separately:

- For the prior probability of the partition P, we would modulate it as a regularization constant:

  $$p(P(\Omega)) \propto \nu \, exp(-\nu|C|), \ \nu > 0$$

  So it would be a function of the length of the contour
- For the likelihood, so the probability of the image I given the partition P, we can assume that it can be deomposed as a product over the two different regions. Assuming indipendence over each pixel (strong assumption), we can have a product of a probability of having a certain intensity according to a certain distribution for the region that we have chosen and the same probability of any other regions:

  $$p(I|P(\Omega)) = p(I|\Omega_e)p(I|\Omega_i) = \prod_{x \in \Omega_e} p_e(I(x), \theta_e) \prod_{x \in \Omega_i} p_i(I(x), \theta_i)$$

  The posterial probability is just proportial to this product of the previous two terms:

  $$p(P(\Omega)|I) = \nu \, exp(-\nu|C|) \prod_{x \in \Omega_e} p_e(I(x), \theta_e) \prod_{x \in \Omega_i} p_i(I(x), \theta_i)$$

We can converge the previous probability function as an energy function, we just compute the $-log()$ of each term and sum them:

$$E(C, \theta_e, \theta_i) = E_{reg}(C) + E_e(C, \theta_e) + E_i(C, \theta_i)$$

Where:

$$\begin{cases} E_{reg}(C) = -log\nu + \nu|C| \\ E_e(C, \theta_e) = -\int_{x \in \Omega_e} log\, p_e(I(x), \theta_e) dx \\ E_i(C, \theta_i) = -\int_{x \in \Omega_i} log\, p_i(I(x), \theta_i) dx \end{cases}$$

This energy function depends on the contour (which is the same as before), but also on the distribution of each region (here there are 2 regions).

The solution of this energy functions implies using the level-set method:

- we introduce the level set function:

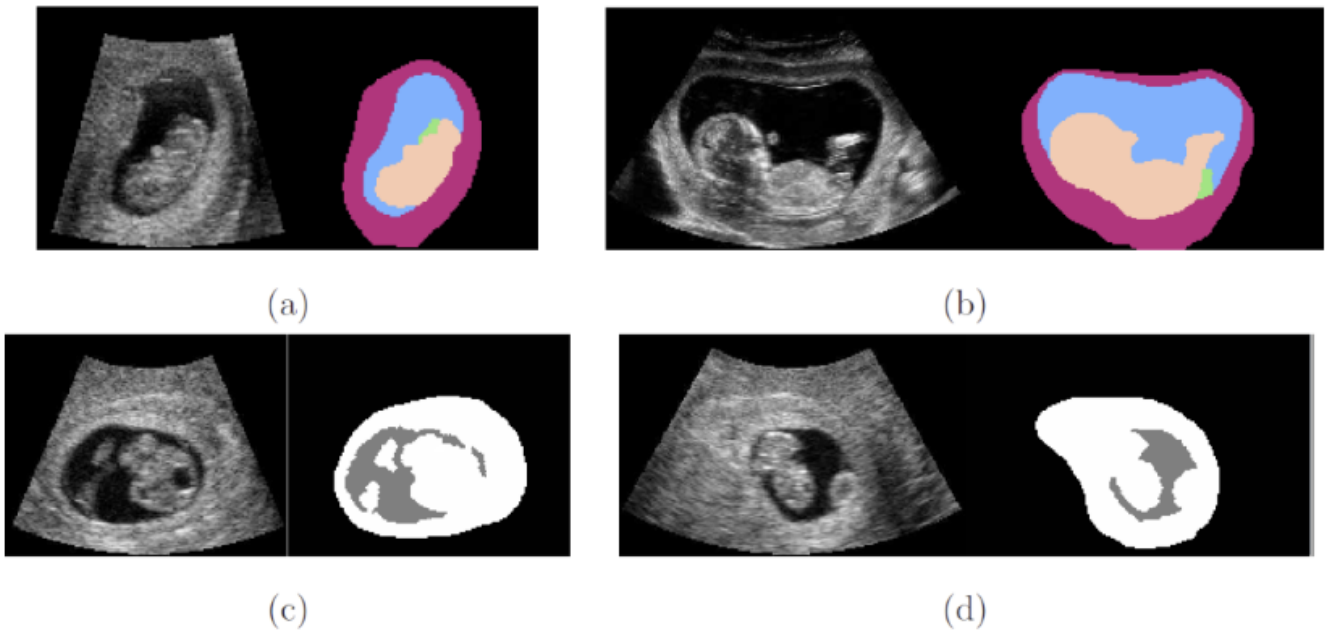
$$\varphi : \Omega \to \mathbb{R} \begin{cases} \varphi(x) > 0 & in \quad \Omega_e \\ \varphi(x) < 0 & in \quad \Omega_i \\ \varphi(x) = 0 & on \quad C \end{cases}$$

- we use φ as the "C" on the previous equations:

$$E(\varphi, \theta_i, \theta_e) = E_{reg}(\varphi) + E_e(\varphi, \theta_e) + E_i(\varphi, \theta_i)$$

$$\begin{cases} E_{reg}(\varphi) = \nu \int_{x \in \Omega} \delta(\varphi(x)) |\nabla \varphi(x)| dx \\ E_e(\varphi, \theta_e) = -\int_{x \in \Omega} H(\varphi(x)) \; log(p_e(I(x), \theta_e)) dx \\ E_i(\varphi, \theta_i) = -\int_{x \in \Omega} (1 - H(\varphi(x))) \; log(p_i(I(x), \theta_i)) dx \end{cases}$$

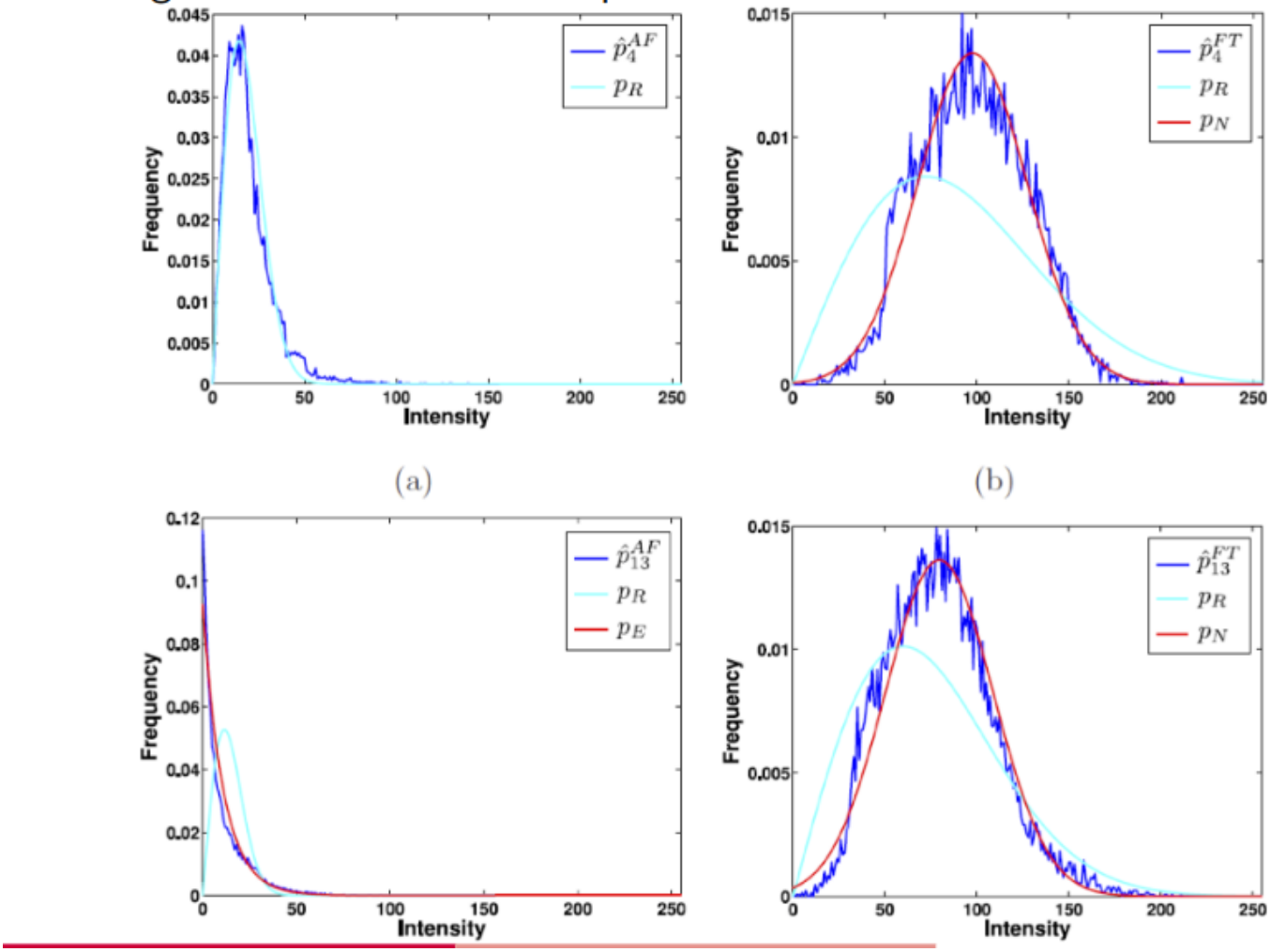
Example in ultrasound imaging (Jérémie Anquez)



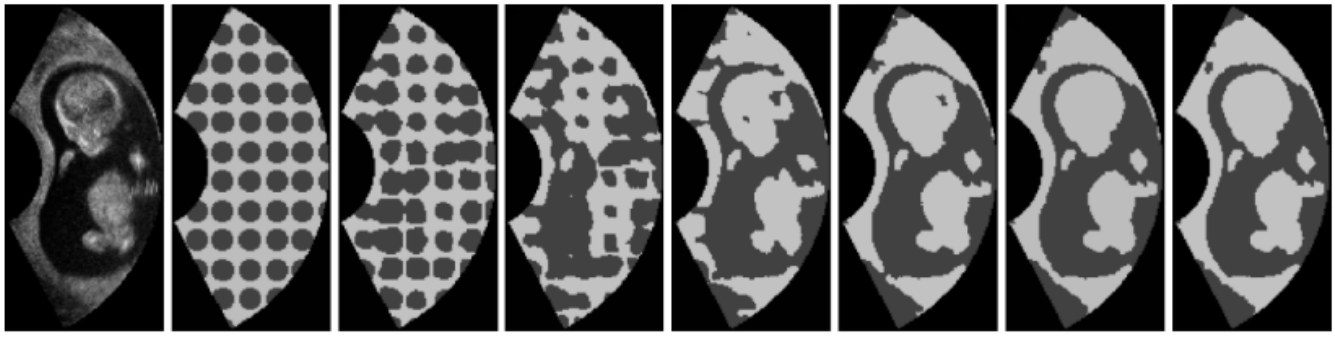
Here the graylevel distribution is different in every region, so we can learn these distribution and apply the algorithm according to each distribution.

We can learn the distribution iteratevely:

Learning distributions and their parameters:



In each region we estimate the best approximate distribution and use it as parameter.



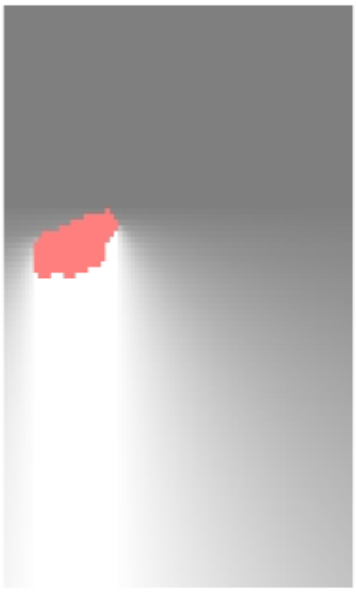
We can add other constraints:

Constraining deformable models by spatial relations (Olivier Colliot et al.)

We can impose that the segmentation is similar to some shapes.

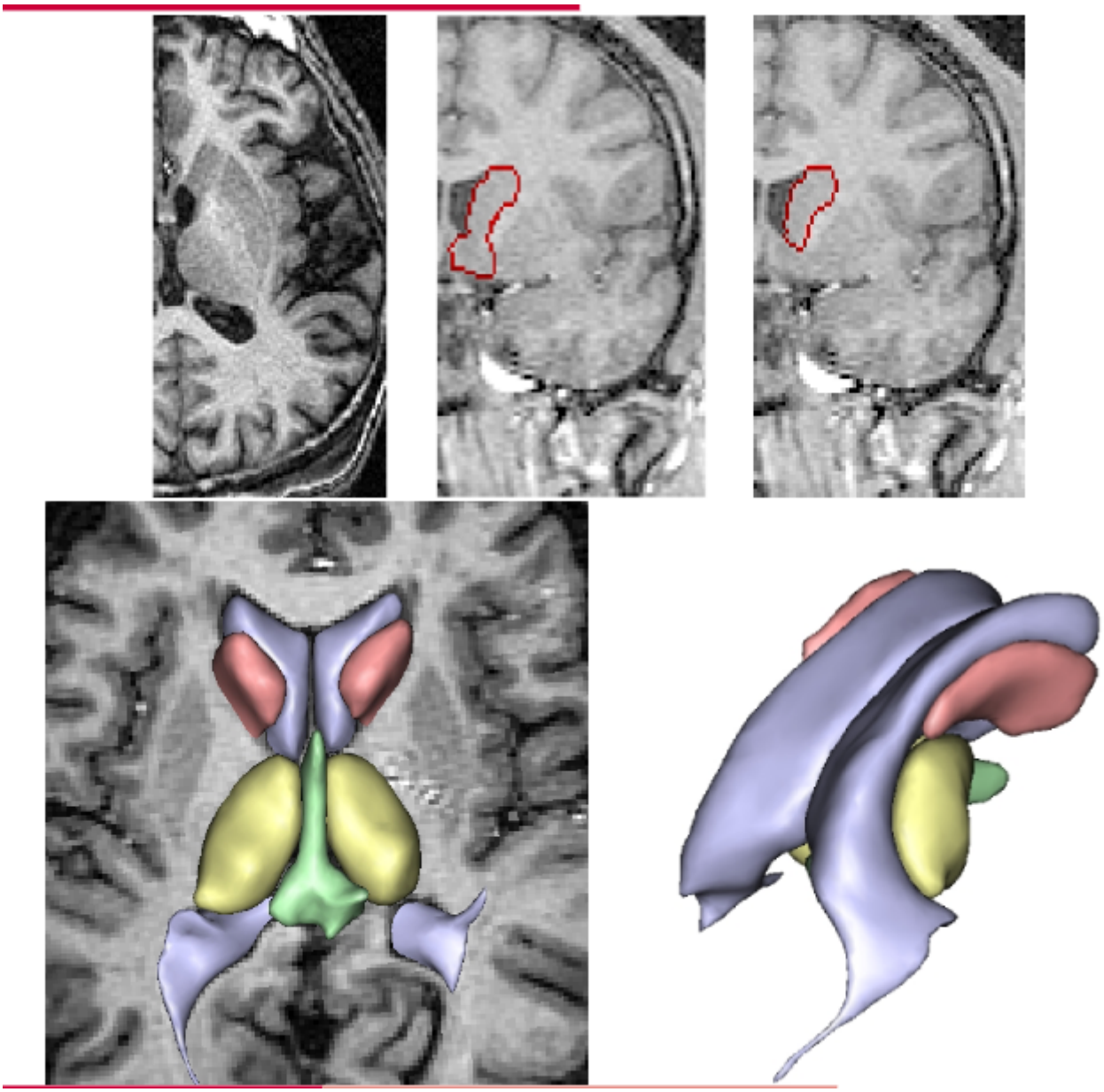
Examples

close to the lateral ventricle



below

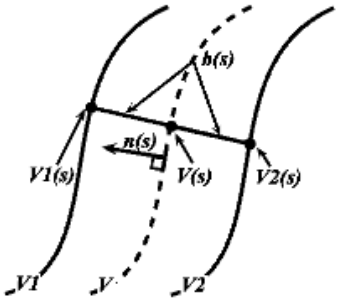
$\Rightarrow$ additional external force (avoids leaking in undesired regions)



For example here the gradient is not strong enough to express a strong contour, so we have a leak. Here we force the contour to have a certain type of shape.

Retina imaging (ISEP and XV-XX)

The idea to segment the vessel is that we want the two line to be parallel. That means that the thickness should not vary too much.



parallelism constraint

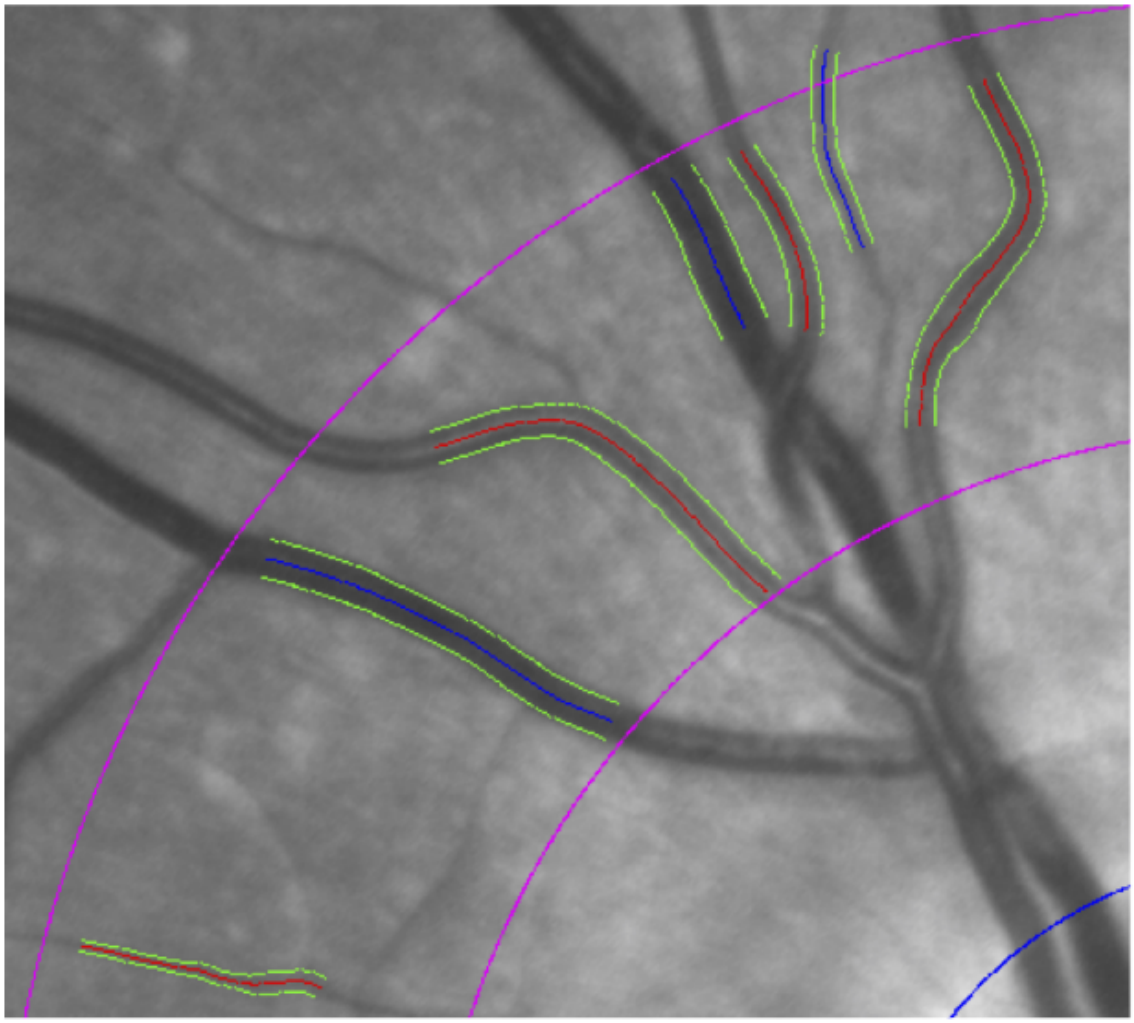
$$E(V, V_1, V_2, b) = E_{Image}(V_1) + E_{Image}(V_2) + E_{Int}(V) + R(V_1, V_2, b)$$

$$E_{Image}(V_i) = \int_0^1 P(V_i(s))ds = -\int_0^1 |\nabla I(V_i(s))|^2 \, ds$$

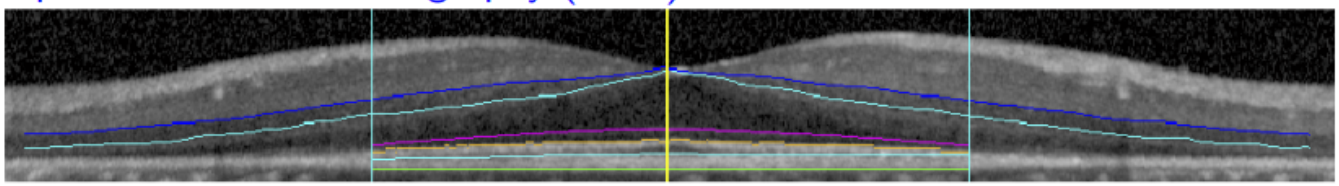
$$E_{Int}(V) = \frac{1}{2}\int \alpha(s)\left|\frac{\partial V(s,t)}{\partial s}\right|^2 + \beta(s)\left|\frac{\partial^2 V(s,t)}{\partial s^2}\right|^2 ds$$

$$R(V_1, V_2, b) = \int_0^1 \varphi(s)(b'(s))^2 ds$$

Eye fundus:



Optical coherence tomography (OCT):



Adaptive optics: