

Herve Lombaert

Research	Projects	Algorithms	School	Presentations	Schedule
----------	----------	------------	--------	---------------	----------

Level set method: Explanation

Here the basics of the level set method is explained, this is not an advanced tutorial. For quite some time, I was afraid of the level set method. I am still not sure why I first got scared. The level set method is just plain easy to understand: there is a surface, it intersects a plane, that gives us a contour and that's it. With image segmentation, the surface is updated with forces derived from the image. In this document, I follow this path: what is the problem, what was done to solve it and where the limitation was, and then the miracle solution with some pretty pictures.

Tracking interfaces

First, let us imagine water falling from the top of a hill. The goal is to track the water front while it's moving down the hill. The figure below shows a map of a V-shaped water fall with water motion shown in red.

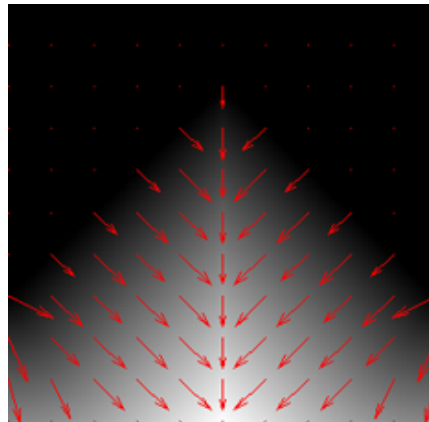


Figure: Elevation map of a V-shaped water fall, black is on top, white is down the fall. Arrows show the Water motion.

The question now is where is the water front at a given time t ?

Explicit contours

One way is to keep track of a few points, advance them in the normal direction to the front (the red arrows), and guess where the front ends up. The figure below shows a step of this technique.

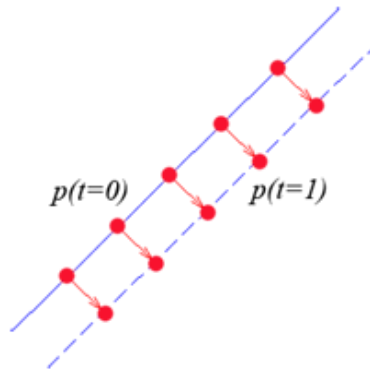


Figure: Evolution of a set of points on the front at time $t=0$, and at time $t=1$

Evolving explicitly the front with a set of points may sound a good solution, but there are a couple of drawbacks that may want you to not use this technique.

With our last example of a V-shaped propagating front, corners can end up in an unknown state. The figure below illustrates such a state after one iteration.

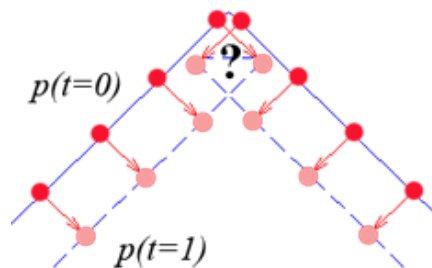
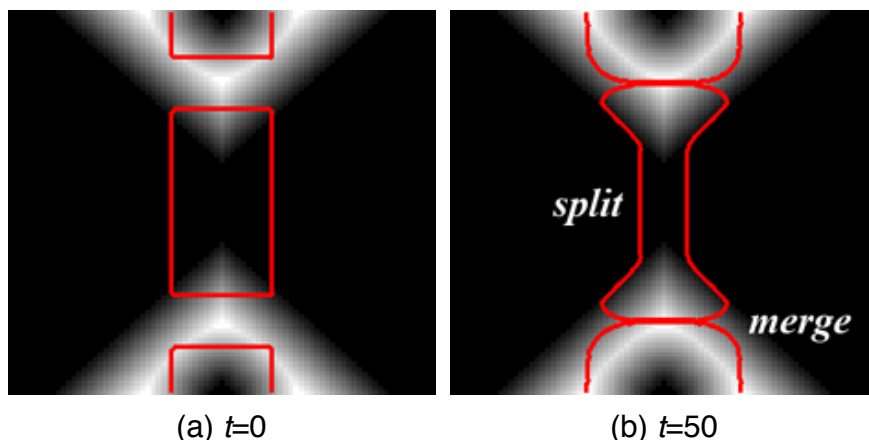


Figure: Front evolution of a corner into an unknown state

Also, if the front is expanding, the initial set of points may not be sufficient enough to define the evolving front. Points should be inserted or removed in the front when it is collapsing, and distance between points should be kept small enough for a smooth front. Such a mechanism could be troublesome during implementation.

Topology change requires further care during implementation (eg. split and merge of fronts). In our hill example, let us now imagine valleys and water moving from the top down into the valleys. The figure below illustrates such a scenario. The challenge is how to insert or remove points during a topology change.



(a) $t=0$

(b) $t=50$

Figure: Two valleys, black is on top, white is down the fall. (a) Initial water fronts on top of a valley, (b) splitting and merging of water fronts

Tracking the front with explicit contours is intuitive but can get troublesome during implementation.

Implicit contours

The idea of evolving a surface (ϕ) instead of a front (C) was proposed, and the front is defined to be all points where the surface has no height ($\phi = 0$). The front is then defined implicitly as the zero level set $\phi = 0$. The figure below shows how the contour is extracted from the evolving surface.

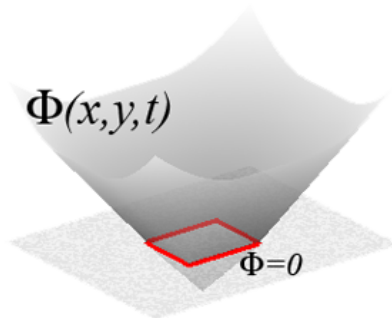
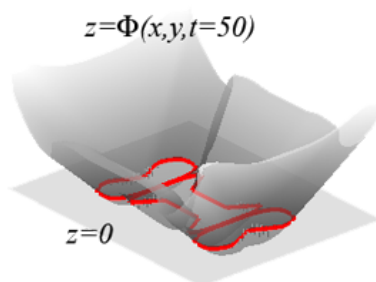
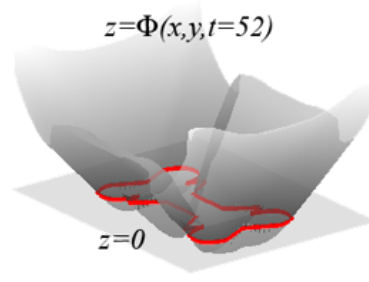


Figure: Here the zero level set of the surface is a square

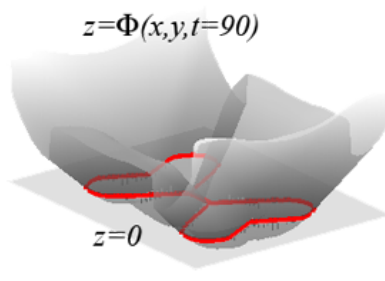
When the surface evolves, cups can appear, they can later narrow, or disappear. The zero level set shows contours splitting and merging as illustrated in the figure below where the plane at $z=0$ represents the map of our valley. The intersection of the surface ϕ with the plane creates the implicit contour. Merging and splitting are here handled naturally by the surface motion.



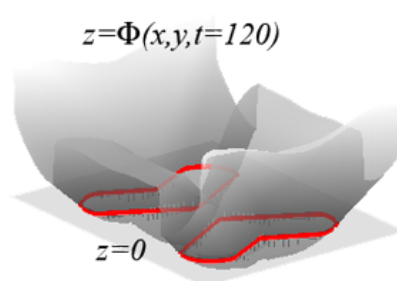
(a) $t=50$, beginning of merging



(b) $t=52$, end of merging



(c) $t=90$, beginning of splitting



(d) $t=120$, end of splitting

Figure: The evolving front in red is known by taking the zero


level set of a surface ϕ

No extra care is needed for topology change. This looks like an interesting idea, the question is now: what is the function ϕ ?

Level set equation

Let us look at the math behind this idea. A point $x = (x, y)$ belonging to a front evolves over time so that $x(t)$ is its position over time. At any time t , for each point $x(t)$ on the front the surface has by definition no height, thus:

$$\phi(x(t), t) = 0$$

The question still remains: what is the function $\phi(x(t), t)$? It can actually be anything we want as long as its zero level set gives us the contour. For example, a previous figure () showed an initial squared contour. The surface height is equal to the distance from (x, y) to the closest point on the contour, so that $\phi(x, y, t=0) = \pm d$, with distance d positive outside the contour, and d negative inside it. So the initial ϕ can actually be any arbitrary function as long as its zero level set matches the initial contour

Given an initial ϕ at $t=0$, it would be possible to know ϕ at any time t with the motion equation $\frac{\partial \phi}{\partial t}$. For that, the chain rule gives us:

$$\begin{aligned} \frac{\partial \phi(x(t), t)}{\partial t} &= 0 \\ \frac{\partial \phi}{\partial x(t)} \frac{\partial x(t)}{\partial t} + \frac{\partial \phi}{\partial t} &= 0 \\ \frac{\partial \phi}{\partial x(t)} x_t + \phi_t &= 0 \end{aligned}$$

Here, recall that $\frac{\partial \phi}{\partial x} = \nabla \phi$. Also, the speed x_t is given by a force F normal to the surface, so $x_t = F(x(t)) n$ where $n = \frac{\nabla \phi}{|\nabla \phi|}$. The previous motion equation can be rewritten with:

$$\begin{aligned}
\phi_t + \nabla \phi \mathbf{x}_t &= 0 \\
\phi_t + \nabla \phi F \mathbf{n} &= 0 \\
\phi_t + F \nabla \phi \frac{\nabla \phi}{|\nabla \phi|} &= 0 \\
\phi_t + F |\nabla \phi| &= 0
\end{aligned}$$

This last equation defines the motion of ϕ . Given ϕ at time $t=0$, and its motion over time, it is now possible to know $\phi(x, y, t)$ at any time t by evolving the initial $\phi(x, y, t=0)$ over time. This answers our initial question, we now know what ϕ is.

An interesting feature that comes with ϕ is that it is possible to get the surface curvature with:

$$\begin{aligned}
\kappa &= \nabla \frac{\nabla \phi}{|\nabla \phi|} \\
&= \frac{\phi_{xx}\phi_y^2 - 2\phi_{xy}\phi_x\phi_y + \phi_{yy}\phi_x^2}{(\phi_x^2 + \phi_y^2)^{3/2}}
\end{aligned}$$

This could be useful to control the front smoothness.

Implementation

In the computer world, images have pixels and functions need to be discretized. That means ϕ_t will be evaluated at pixel (i, j) by $\frac{\phi(i, j, t+\Delta t) - \phi(i, j, t)}{\Delta t}$. The gradient will be evaluated by a finite difference scheme, for example with:

$$\begin{aligned}
\nabla^{+x}(i, j) &= \max [0, \Delta^{-x}\phi(i, j)]^2 + \min [0, \Delta^{+x}\phi(i, j)]^2, \text{ when } F > 0, \text{ or} \\
\nabla^{-x}(i, j) &= \max [0, \Delta^{+x}\phi(i, j)]^2 + \min [0, \Delta^{-x}\phi(i, j)]^2, \text{ when } F < 0.
\end{aligned}$$

Here, $\Delta^{-x}\phi$ or $\Delta^{+x}\phi$ is the left or the right side finite difference for a given point. This is illustrated in the figure below. The gradient is computed differently depending on the front direction and a finite difference scheme takes account of that.

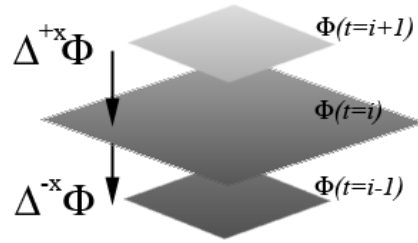


Figure: left or right side finite difference (eg. how to evaluate $|\nabla\phi|$)

The previous motion equation now becomes:

$$\frac{\phi(i, j, t + \Delta t) - \phi(i, j, t)}{\Delta t} + \max[F, 0] \nabla^{+x}(i, j) + \min[F, 0] \nabla^{-x}(i, j) = 0$$

From there, updating the surface $\phi(i, j)$ is done with:

$$\phi(i, j, t + \Delta t) = \phi(i, j, t) - \Delta t [\max[F, 0] \nabla^{+x}(i, j) + \min[F, 0] \nabla^{-x}(i, j)]$$

When computing the curvature, it depends only on the surface ϕ , so central differences can be used. They are:

$$\begin{aligned} \phi_{xx}(i, j) &= (\phi(i+1, j) - \phi(i, j)) - (\phi(i, j) - \phi(i-1, j)) \\ \phi_{yy}(i, j) &= (\phi(i, j+1) - \phi(i, j)) - (\phi(i, j) - \phi(i, j-1)) \\ \phi_{xy}(i, j) &= \frac{1}{4} [(\phi(i+1, j+1) - \phi(i-1, j+1)) - (\phi(i+1, j-1) - \phi(i-1, j-1))] \\ \phi_x &= \frac{1}{2} (\phi(i+1, j) - \phi(i-1, j)) \\ \phi_y &= \frac{1}{2} (\phi(i, j+1) - \phi(i, j-1)) \end{aligned}$$

The curvature is then computed numerically with:

$$\begin{aligned} \kappa(i, j) &= \frac{\phi_{xx}\phi_y^2 - 2\phi_y\phi_x\phi_{xy} + \phi_{yy}\phi_x^2}{(\phi_x^2 + \phi_y^2)^{3/2}} \\ \kappa(i, j)|\nabla\phi(i, j)| &= \frac{\phi_{xx}\phi_y^2 - 2\phi_y\phi_x\phi_{xy} + \phi_{yy}\phi_x^2}{\phi_x^2 + \phi_y^2} \end{aligned}$$

To keep a front smooth, high curvature should be penalized. That means, rather than decreasing ϕ in order to have the zero level set expanding, high curvature will reverse the front motion. Updating $\phi(i, j)$ with the curvature could be:

$$\phi(i, j, t + \Delta t) = \phi(i, j, t) - \Delta t [\max(F, 0) \nabla^{+x}(i, j) + \min(F, 0) \nabla^{-x}(i, j)] + \Delta t [\kappa(i, j)]$$

Results

Given an initial arbitrary ϕ , for example the distance transform of an initial contour, and a numerical scheme for the motion equation $\frac{\partial \phi}{\partial t}$, it is possible to show some examples of contour evolution.

First a simple example of a drop of water expanding ($F=1$ everywhere) with an obstacle on its path (where $F=0$). The water front should be stopped by the obstacle and later be perturbed by the obstacle. This is shown in the figures below.

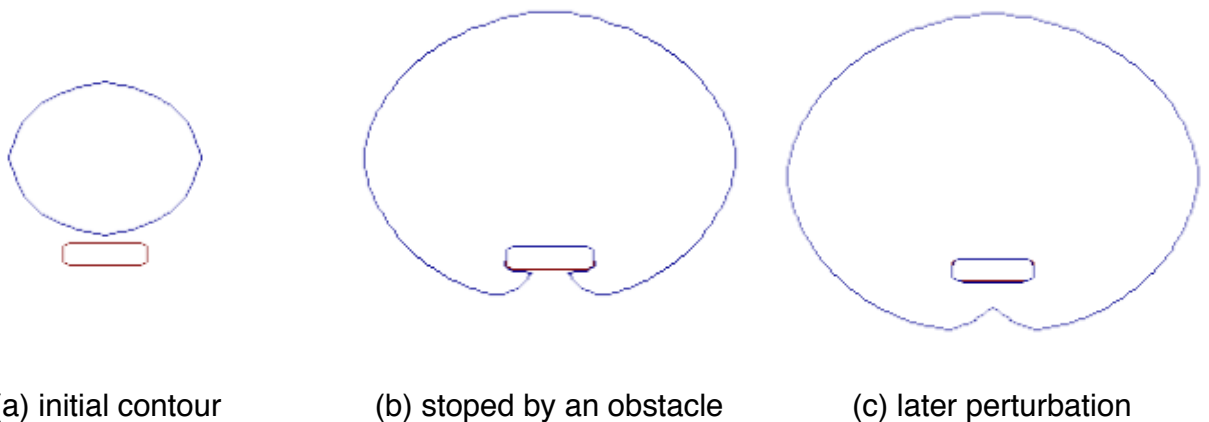


Figure: An initial circle expands where $F=1$ and is perturbed by an obstacle where $F=0$

Then, a slightly more complex form illustrated below. The initial contour is still a circle, and the forces applied to it is positive ($F=1$) inside a form, and it is negative ($F=-1$) outside it. The contour should then be attracted into the form. Also notice the front split between the two regions. Here, the surface ϕ is also shown to explain how this split is formed.

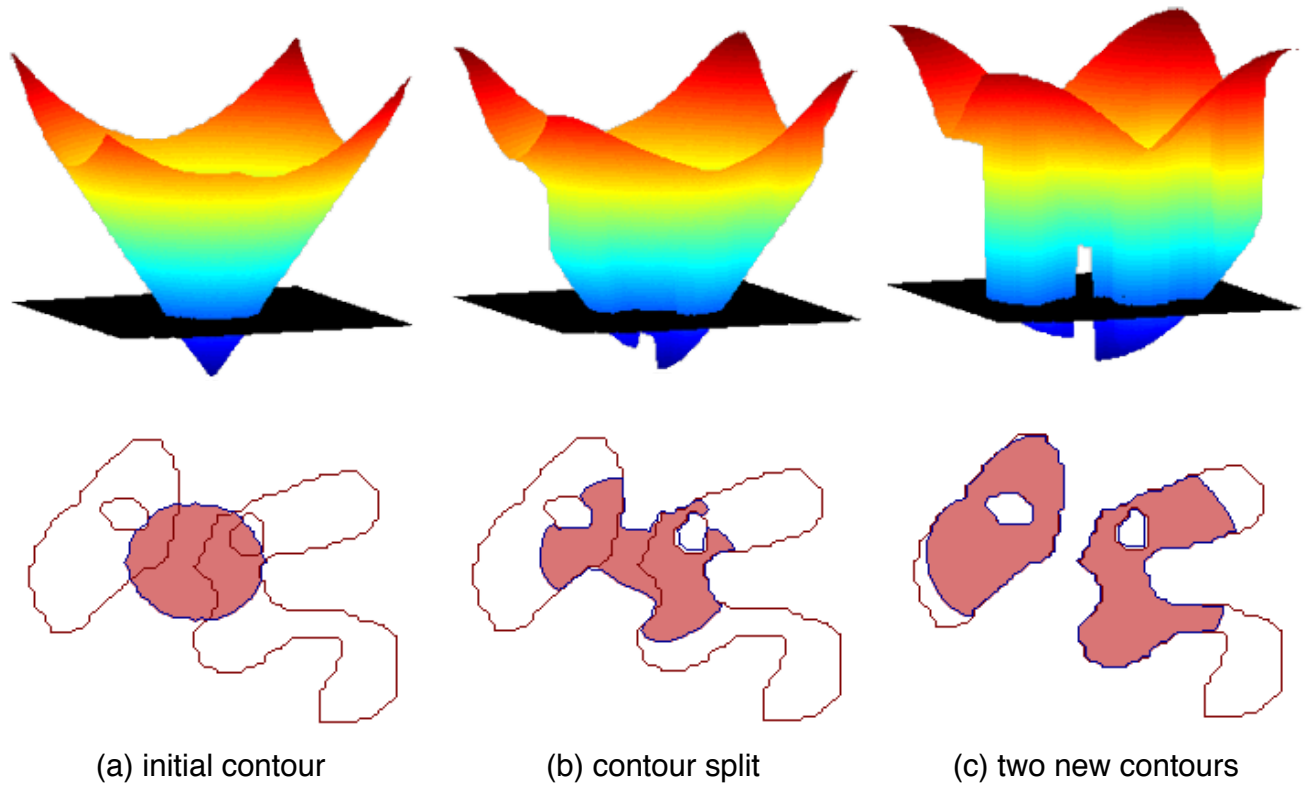


Figure: An initial circle expands inside the form where $F > 0$ and collapses outside the form ($F < 0$), the corresponding ϕ surface is also shown with a plane intersecting at $z=0$

After having shown how constant forces act on ϕ , the next example shows a shape collapsing under its curvature. Here the force is equal to the curvature κ of the surface ϕ .

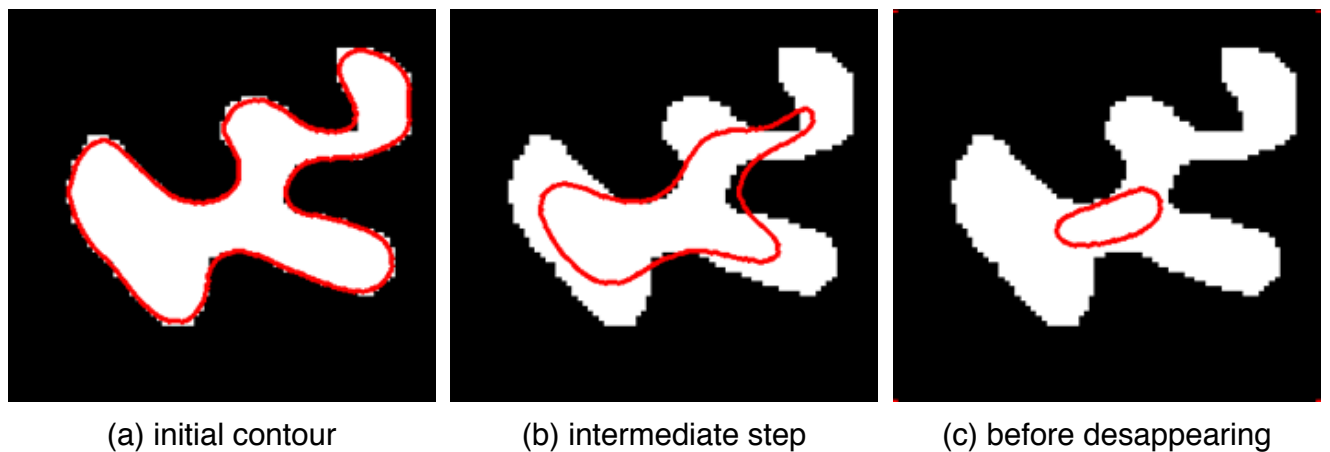


Figure: A form collapsing under its curvature

Now the fun begins with real pictures. Instead of using constant positive or negative forces, with or without curvature, the force is derived from an image. We can think of a contour stopped at the edges of an object. The force should then be high inside the object (we want the curve expanding inside an object), and it should be low close to the edges (we want the curve to stop at the edges). The gradient of an image shows where the edges are.

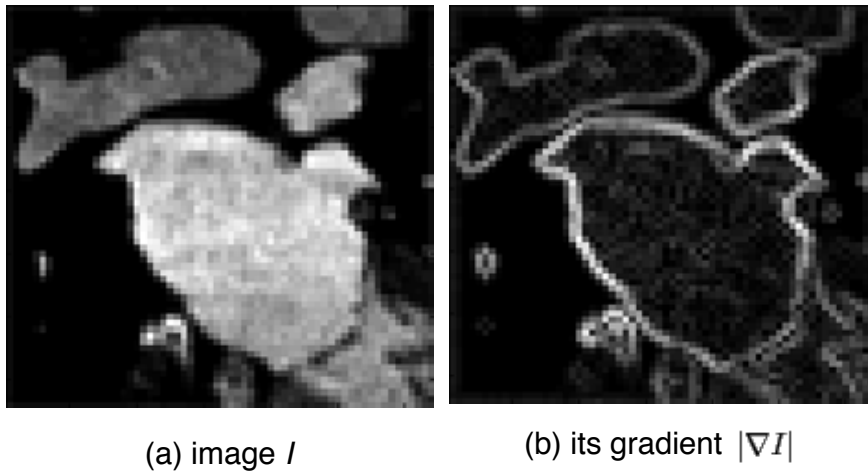


Figure: Image of a heart with its corresponding gradient

The force to be used could be the inverse of the previous gradient image ∇I , or it could be the gaussian of it:

$$F(i, j) = \frac{1}{1 + \lambda |\nabla I(i, j)|}$$

$$F(i, j) = e^{-\frac{|\nabla I(i, j)|^2}{2\sigma^2}}$$

Here λ or σ are parameters controlling how penalizing the edges should be. Below is an evolving contour using the inverse of the gradient image.

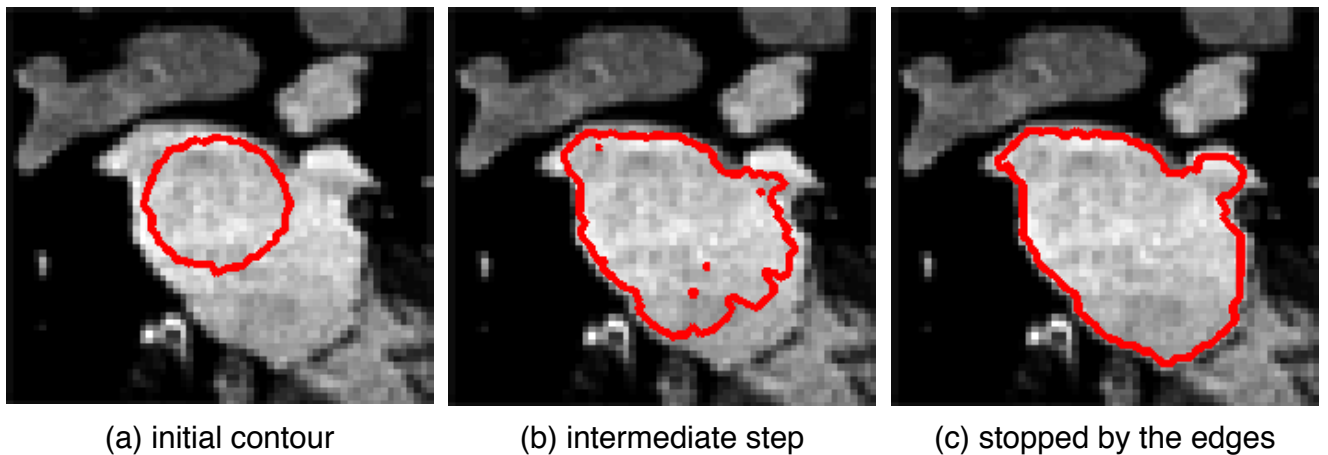


Figure: Contour evolution with a real image

At this point, you should at least understand that level set method is all about evolving a surface instead of the real contour. That makes the beauty of this method. Level sets have applications in many fields. See google for further reading.

Bibliography

1

Osher S., Sethian J.A., *Fronts Propagating with Curvature-Dependent Speed: Algorithms Based on Hamilton-Jacobi Formulations* (Journal of Computational Physics, 79(1), page 12-49, 1988).

2

Kass M., Witkin A., Terzopoulos D., *Snakes - Active Contour Models* (International Journal of Computer Vision, 1(4), page 321-331, 1987)

Herve Lombaert 2006-03-01