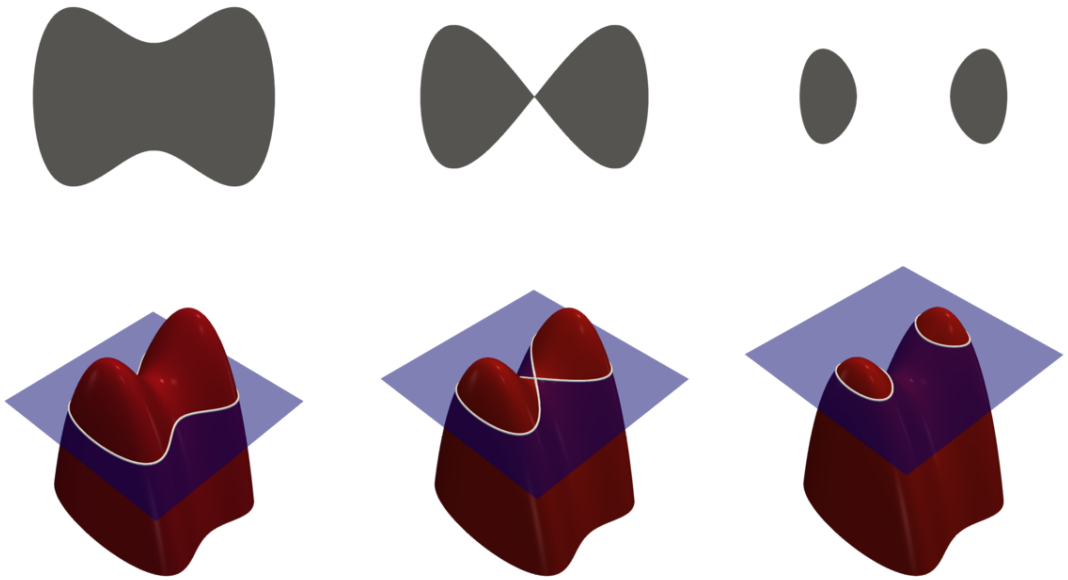
INSA | INSTITUT NATIONAL DES SCIENCES APPLIQUÉES ROUEN NORMANDIE

Image Processing

Active Contour Models



Rand ASSWAD
Applied Mathematics

Submitted to:
Mr. Christian GOUT

2 March 2020

Contents

1 Introduction

Recognising objects and identifying shapes in images is usually an easy task for human, it is however difficult to automate. The field of **computer vision** is concerned with automating such processes. It aims to extract information from images (or video sequences of images) in order to achieve what a human visual system can.

Active contour models (also called **snakes**) is a class of algorithms for finding boundaries of shapes. These methods formulate the problem as an optimisation process while attempting to balance between matching to the image and ensuring the result is smooth.

In the scope of this project, we will explore a few active contour models with the help of *The Numerical Tours of Signal Processing* (Peyré 2011).

The Level-Set Method (LSM) allows manipulating manipulating hypersurfaces in different contexts without having their parametric representation.

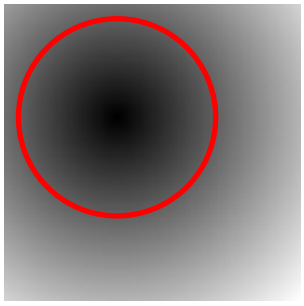
LSM can be used to track changing topologies as well as contour detection in image processing.

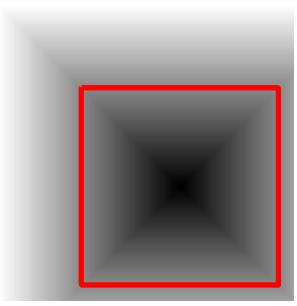
1.1 Basic terminology

- level-set
- phi
- etc

1.2 Exo 1

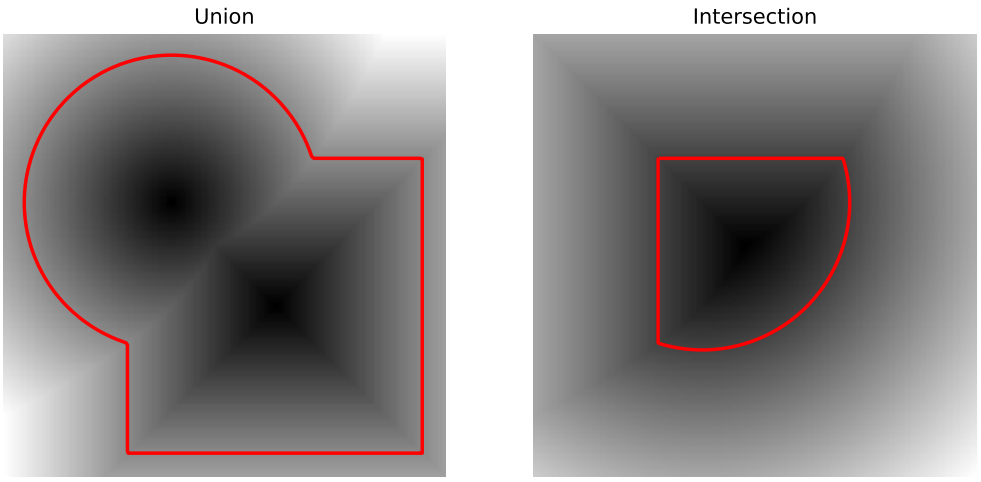
```
n = 200
Y, X = np.meshgrid(np.arange(1,n+1), np.arange(1,n+1))
r = n / 3

c = np.array([r,r]) + 10
phi1 = np.sqrt((X-c[0])**2 + (Y-c[1])**2) - r

c = n - c
phi2 = np.maximum(abs(X-c[0]), abs(Y-c[1])) - r

from nt_toolbox.plot_levelset import *
plt.figure()
plt.subplot(121)
plot_levelset(phi1)
plt.subplot(122)
plot_levelset(phi2)
plt.show()
```





1.3 Exo 2

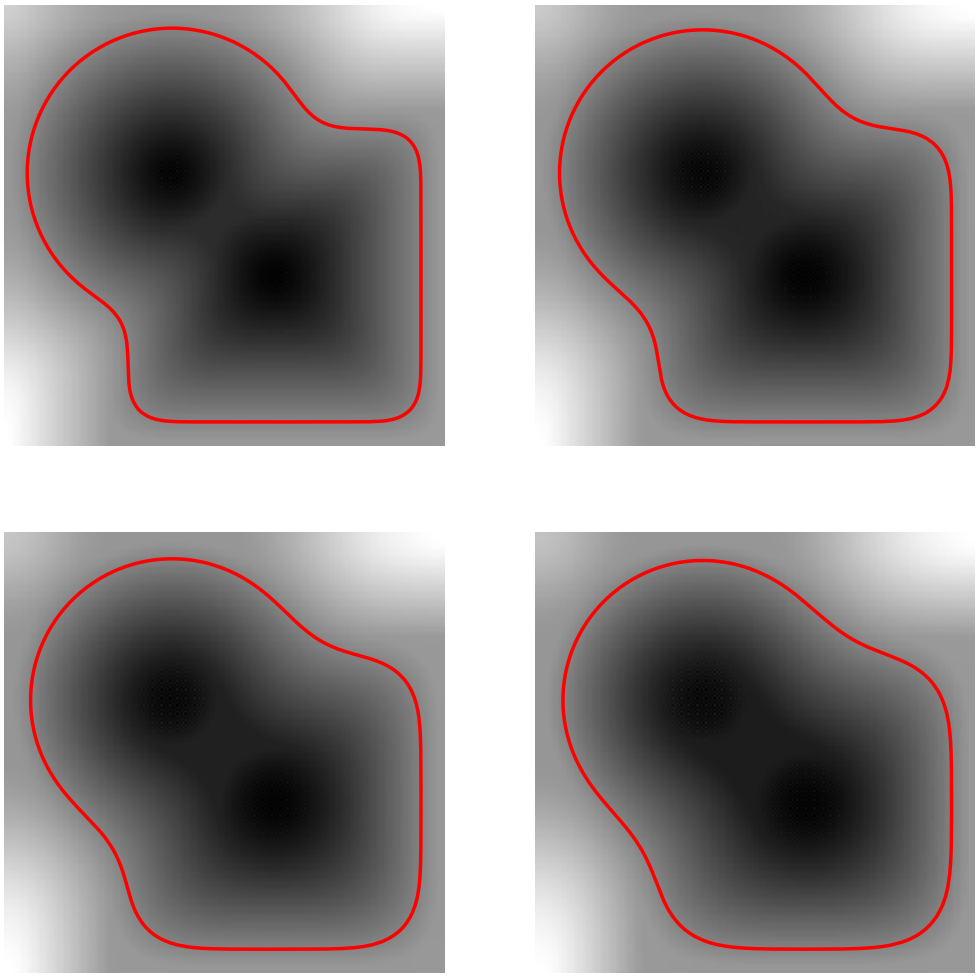
```
plt.figure(figsize = (10,5))
phi0 = np.minimum(phi1, phi2)

plt.subplot(1,2,1)
plot_levelset(phi0)
plt.title("Union")

plt.subplot(1,2,2)
plot_levelset(np.maximum(phi1, phi2))
plt.title("Intersection")

plt.show()
```



1.4 Exo 3

```
from nt_toolbox.grad import *
from nt_toolbox.div import *

Tmax = 200
tau = .5
niter = int(Tmax/tau)


plt.figure(figsize=(10,10))
phi = np.copy(phi0) #initialization
eps = np.finfo(float).eps
k = 0

for i in range(1,niter+1):
    g0 = grad(phi, order=2)
    d = np.maximum(eps*np.ones([n,n]), np.sqrt(np.sum(g0**2, 2)))
    g = g0/np.repeat(d[:,:,np.newaxis], 2, 2)
    K = d*div(g[:,:,0], g[:,:,1], order=2)
    phi = phi + tau*K
```

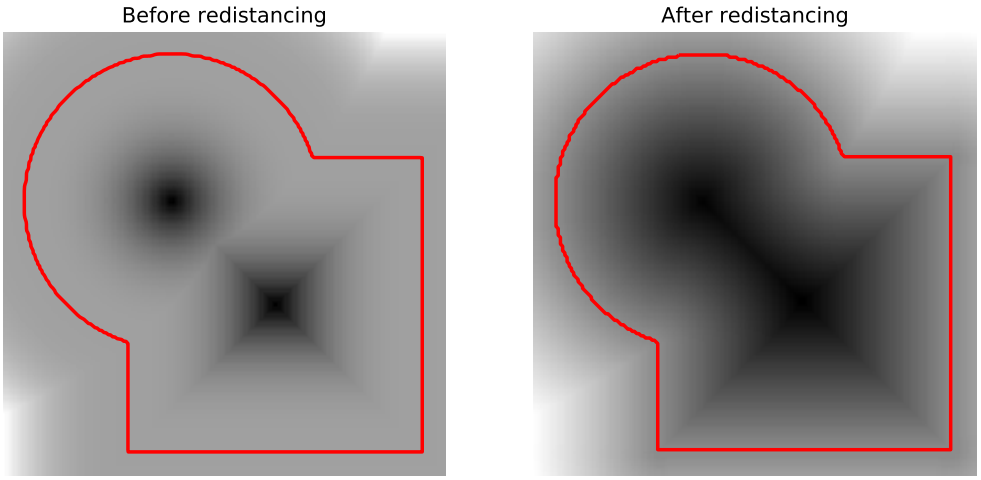
```
    if i % int(niter/4.) == 0:
        k = k + 1
        plt.subplot(2, 2, k)
        plot_levelset(phi)
plt.show()
```



1.5 Exo 4

```
# problem cause
phi = phi0**3
from nt_toolbox.perform_redistancing import *
phi1 = perform_redistancing(phi0)

plt.figure(figsize=(10,5))

plt.subplot(1,2,1)
plot_levelset(phi)
```

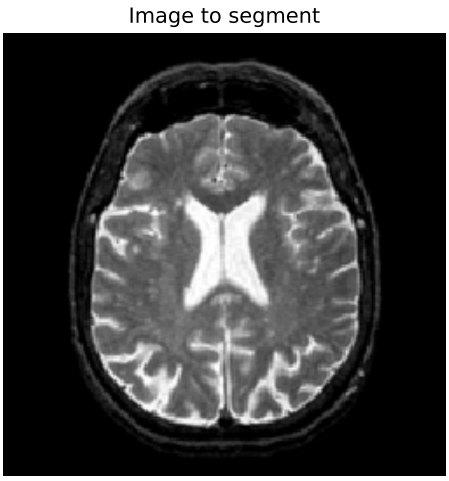
```
plt.title("Before redistancing")

plt.subplot(1,2,2)
plot_levelset(phi1)
plt.title("After redistancing")

plt.show()
```



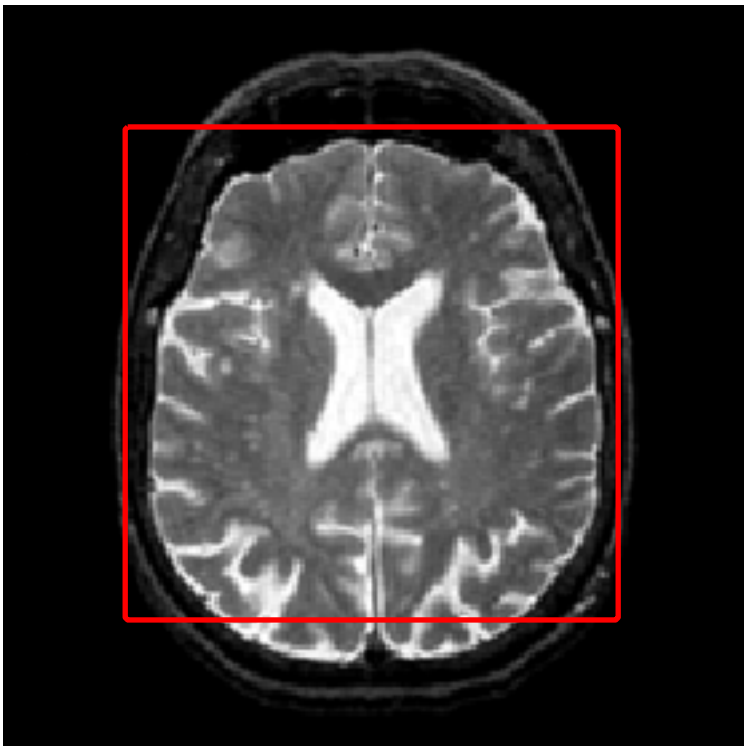

```
n = 200
f0 = rescale(load_image("nt_toolbox/data/cortex.bmp", n))
g = grad(f0, order=2)
d0 = np.sqrt(np.sum(g**2, 2))
a = 5
from nt_toolbox.perform_blurring import *
d = perform_blurring(d0, np.asarray([a]),bound="per")
```

```
[200 200]
```

```
epsilon = 1e-1
W = 1./(epsilon + d)
W = rescale(-d, 0.1, 1)

plt.figure(figsize=(10,5))
imageplot(f0, "Image to segment", [1,2,1])
imageplot(W, "Weight", [1,2,2])
plt.show()
```




```
Y,X = np.meshgrid(np.arange(1,n+1), np.arange(1,n+1))
r = n/3
c = np.asarray([n,n])/2
phi0 = np.maximum(abs(X-c[0]), abs(Y-c[1])) - r
plt.figure(figsize=(5,5))
plot_levelset(phi0, 0, f0)
plt.show()
```

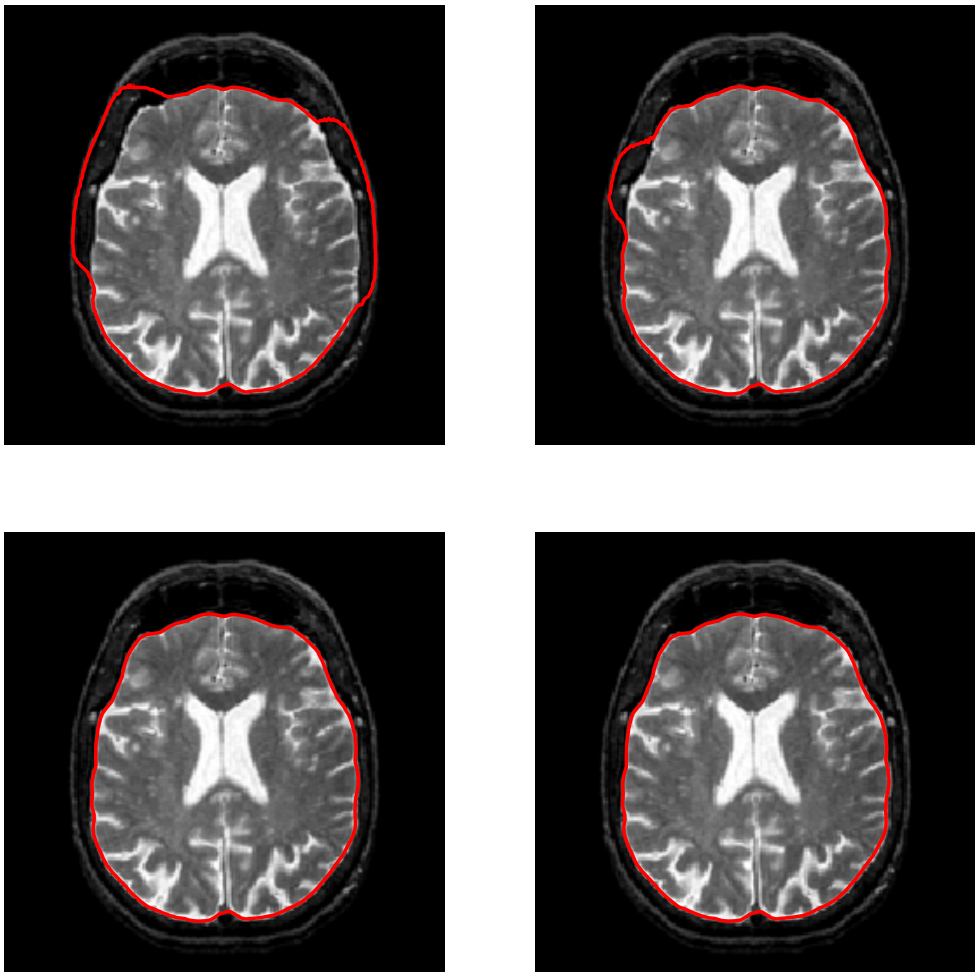


1.6 Exo 5

```
tau = .4
Tmax = 1500
niter = int(Tmax/tau)
phi = np.copy(phi0)
gW = grad(W, order=2)


gD = grad(phi, order=2)
d = np.maximum(eps*np.ones([n,n]), np.sqrt(np.sum(gD**2, 2)))
g = gD/np.repeat(d[:,:,np.newaxis], 2, 2)
G = - W*d*div(g[:,:,0], g[:,:,1], order=2) - np.sum(gW*gD,2)
```

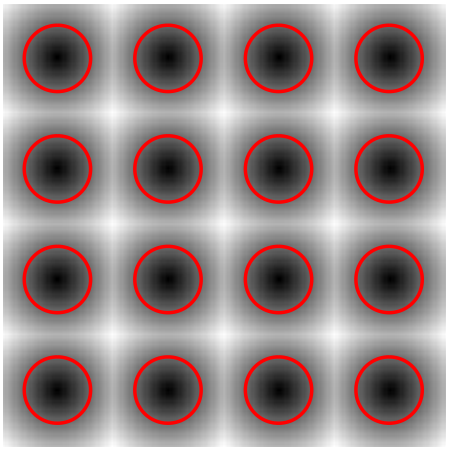
1.7 Exo 6

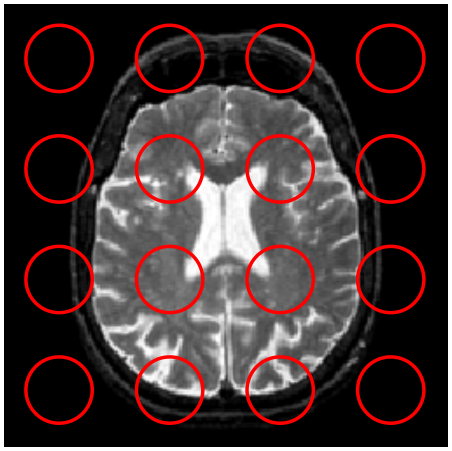
```
plt.figure(figsize=(10,10))
phi = np.copy(phi0)
k = 0
gW = grad(W, order=2)

for i in range(1,niter+1):
    gD = grad(phi, order=2)
    d = np.maximum(eps*np.ones([n,n]), np.sqrt(np.sum(gD**2, 2)))
    g = gD/np.repeat(d[:,:,np.newaxis], 2, 2)
    G = W*d*div(g[:,:,0], g[:,:,1], order=2) + np.sum(gW*gD,2)
    phi = phi + tau*G
    if i % 30 == 0:
        phi = perform_redistancing(phi)
    if i % int(niter/4.) == 0:
        k = k + 1
        plt.subplot(2, 2, k)
        plot_levelset(phi,0,f0)

plt.show()
```



1.8 Exo 7

```
plt.figure(figsize=(10,5))
Y,X = np.meshgrid(np.arange(1,n+1), np.arange(1,n+1))
k = 4 #number of circles
r = .3*n/k
phi0 = np.zeros([n,n])+np.float("inf")

for i in range(1,k+1):
    for j in range(1,k+1):
        c = (np.asarray([i,j]) - 1)*(n/k) + (n/k)*.5
        phi0 = np.minimum(phi0,np.sqrt(abs(X-c[0])**2 + abs(Y-c[1])**2) - r)

plt.subplot(1,2,1)
plot_levelset(phi0,0)
plt.subplot(1,2,2)
plot_levelset(phi0, 0, f0)
plt.show()
```



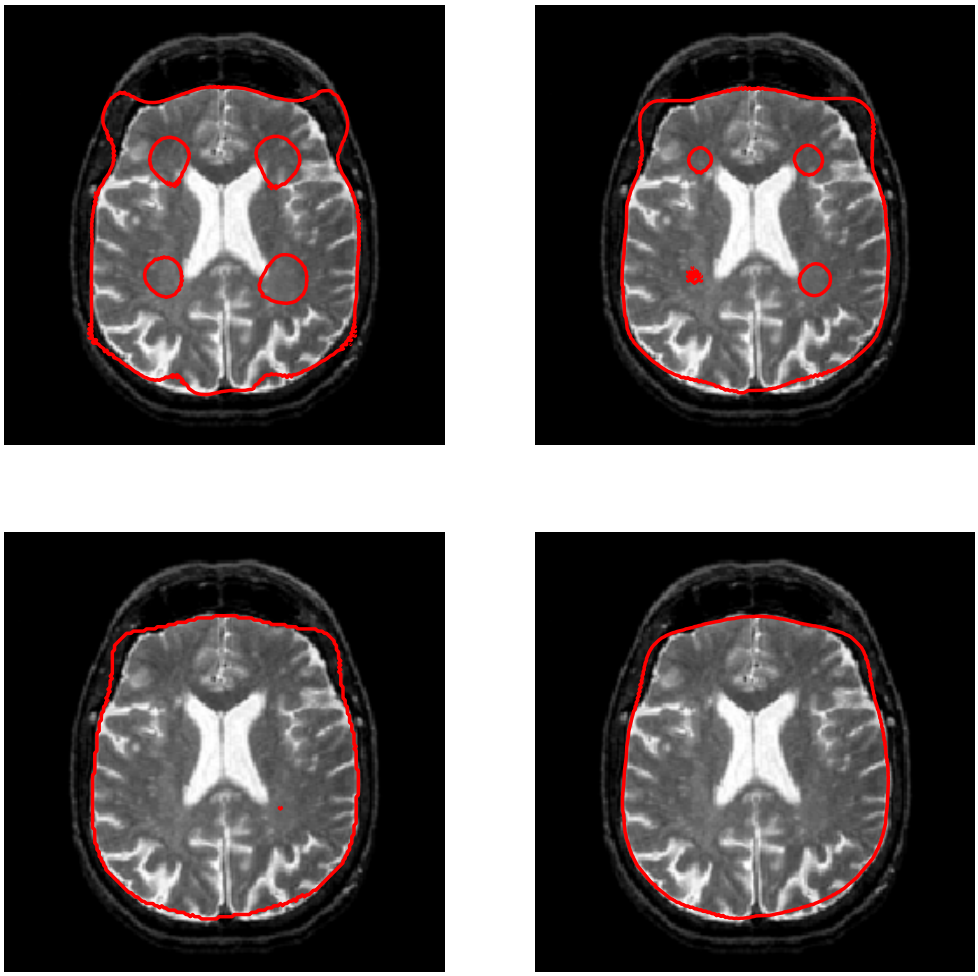


1.9 Exo 8

```
lambd = 2
c1 = .7
c2 = 0
tau = .5

Tmax = 100
niter = int(Tmax/ tau)
phi = np.copy(phi0)

gD = grad(phi, order=2)
d = np.maximum(eps*np.ones([n,n]), np.sqrt(np.sum(gD**2, 2)))
g = gD/np.repeat(d[:,:,np.newaxis], 2, 2)
G = d*div(g[:,:,0], g[:,:,1], order=2) - lambd*(f0-c1)**2 + lambd*(f0-c2)**2
```

1.10 Exo 9

```
plt.figure(figsize=(10,10))
phi = np.copy(phi0)
k = 0

for i in range(1,niter+1):
    gD = grad(phi, order=2)
    d = np.maximum(eps*np.ones([n,n]), np.sqrt(np.sum(gD**2, 2)))
    g = gD/np.repeat(d[:,:,np.newaxis], 2, 2)
    G = d*div(g[:,:,0], g[:,:,1], order=2) - lambd*(f0-c1)**2 + lambd*(f0-c2)**2
    phi = phi + tau*G
    if i % 30 == 0:
        phi = perform_redistancing(phi)
    if i % int(niter/4.) == 0:
        k = k + 1
        plt.subplot(2, 2, k)
        plot_levelset(phi,0,f0)

plt.show()
```



Peyré, Gabriel. 2011. "The Numerical Tours of Signal Processing - Advanced Computational Signal and Image Processing." *IEEE Computing in Science and Engineering* 13 (4): 94–97. https://hal.archives-ouvertes.fr/hal-00519521.