a.) $J(\epsilon) = C(b+\epsilon h)$

$$= \frac{1}{2}\int |I(x-b-\epsilon h) - I'(x)|^2 dx$$

$$\frac{d}{d\epsilon}J(\epsilon)\Big|_{\epsilon=0} = \frac{1}{2}\frac{d}{d\epsilon}\int |I(x-b-\epsilon h) - I'(x)|^2 dx \Big|_{\epsilon=0}$$

$$= \frac{1}{2}\int 2[I(x-b-\epsilon h) - I'(x)] \frac{d}{d\epsilon} I(x-b-\epsilon h)\, dx \Big|_{\epsilon=0}$$

$$= \int [I(x-b-\epsilon h) - I'(x)] \nabla I(x)(-h)\, dx \Big|_{\epsilon=0}$$

$$= -\int [I(x-b) - I'(x)] \nabla I(x)\, h\, dx$$

Notes for (a) and (b):

Here might be some notation errors, but in office hour Frederick confirmed that it's correct.

However, I'm still confused whether the calculus should be eliminated here. When implementing in MATLAB, I, I' and gradient_I are all matrixes. And during each iteration, I think b should be added by a matrix (512*512), rather than a scaler (2*1), in order to be more optimization-wise.

b.) $C(b) = \frac{1}{2}\int |I(x-b) - I'(x)|^2 dx$

$$\nabla C(b) = \frac{1}{2}\int 2[I(x-b) - I'(x)]\, \nabla I(x-b)\, dx$$

$$= \int [I(x-b) - I'(x)] \nabla I(x) \frac{d}{db}(x-b)\, dx$$

$$= -\int [I(x-b) - I'(x)] \nabla I(x)\, dx$$

$$\Rightarrow \nabla C(b)\cdot h = \frac{d}{d\epsilon}J(\epsilon)\Big|_{\epsilon=0}$$

c.) For each iteration in MATLAB:

$$b_{new} = b_{old} - \epsilon \nabla C(b_{old})$$

The gradient part should be modified for vectorization.

Code for gradient descent:

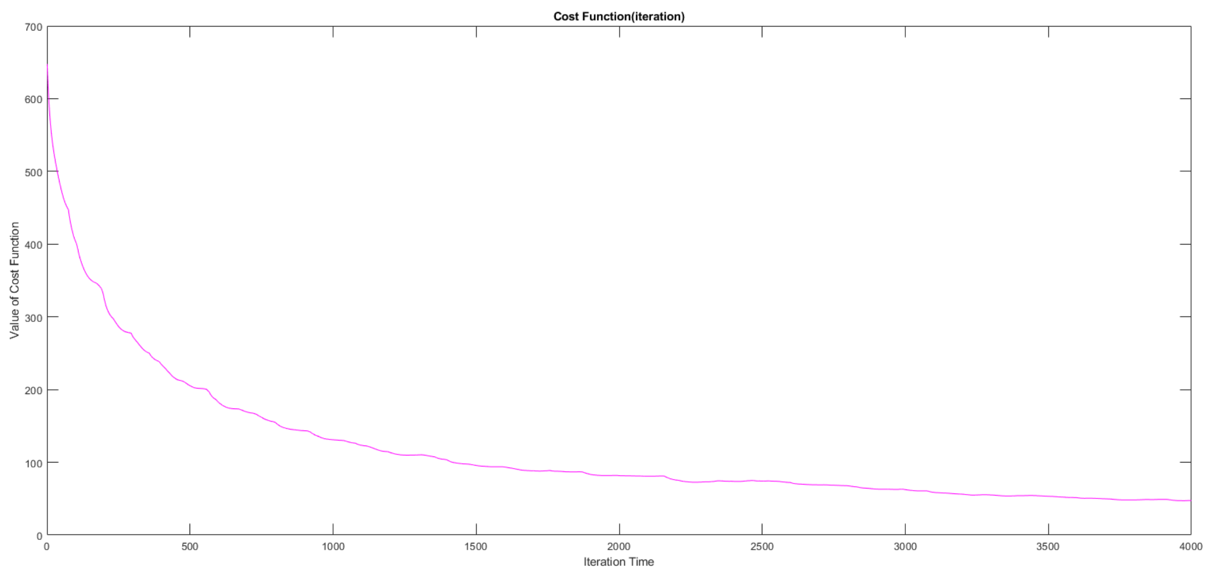
```
[gradIx,gradIy] = gradient(ID);
gradCostx = (J-ID).*gradIx;  %J is the Target image
gradCosty = (J-ID).*gradIy;  %ID is the deformed I
```

d.) e.)

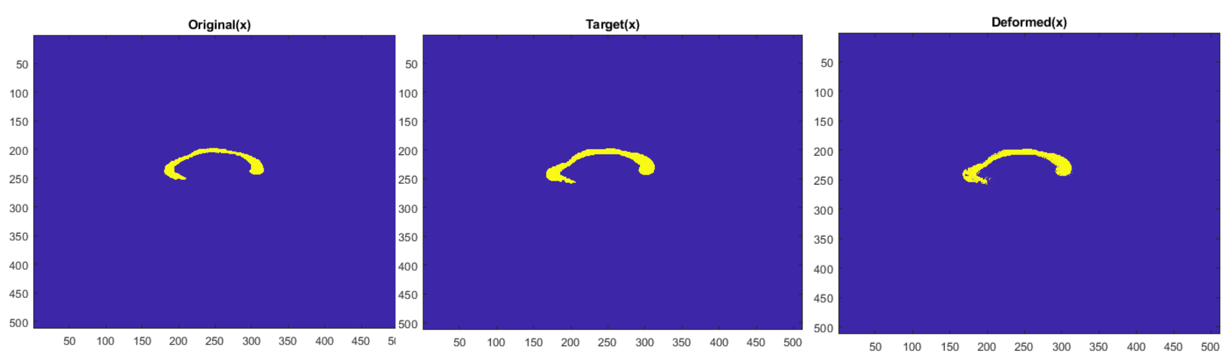
The input:

```
I = double(imread('0001_CC_Con.png') > 0);
J = double(imread('0003_CC_Alz.png') > 0);
epsilon = 0.05;
nIter = 4000;
```

The value of the Cost during iterations:

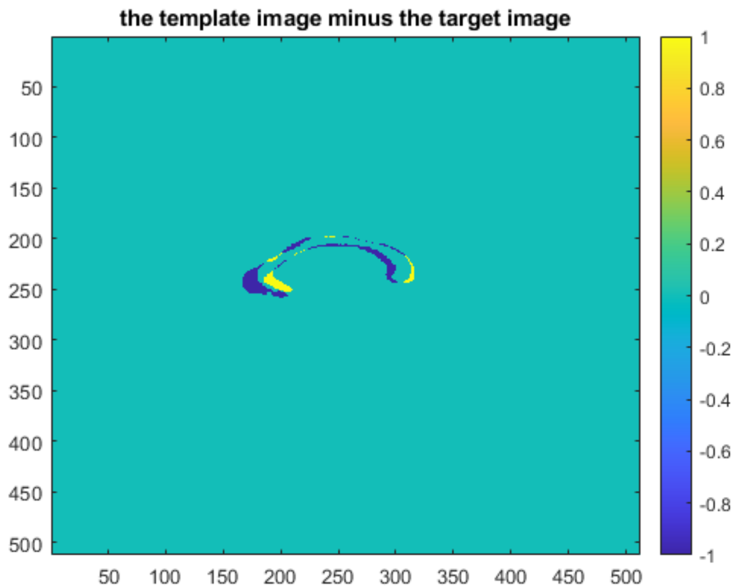


The original/target/deformed image:

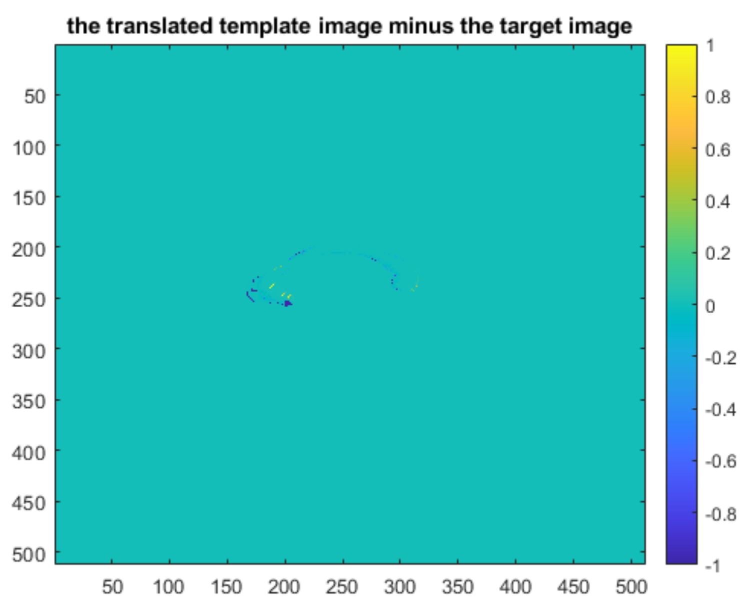


the deformed image is really close to the target but has some noise-like shapes

The template image minus the target image:



The translated template image minus the target image:



I think my algorithm is quite good, but maybe too good?
I noticed that there is a reminder in the homework requirement, which is “Remember that your gradient will be a vector with 2 components”. But I think that will make the ‘b’ too monotonous to fit the template to the target. So I used my algorithm instead.

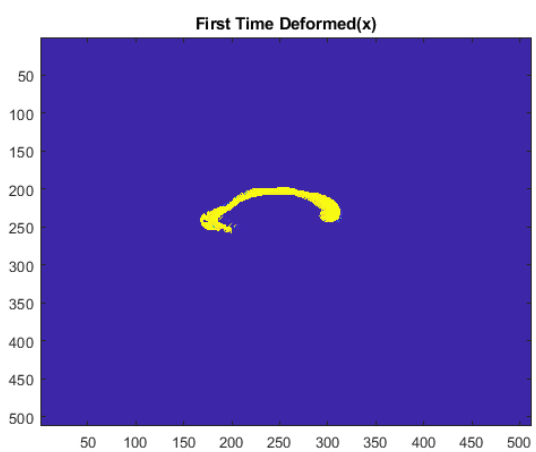
f.)

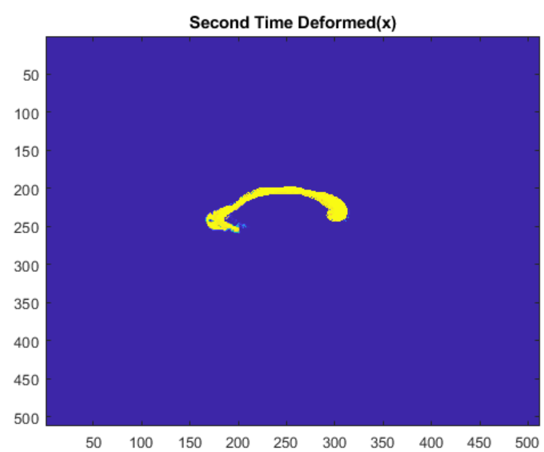
The input:

```
I = IDchenyu; %use my deformed image instead
J = double(imread('0003_CC_Alz.png') > 0);
epsilon = 0.005; %make it much smaller when using slineImage
nIter = 4000;
sigma = 0.01;
alpha = 20;
```

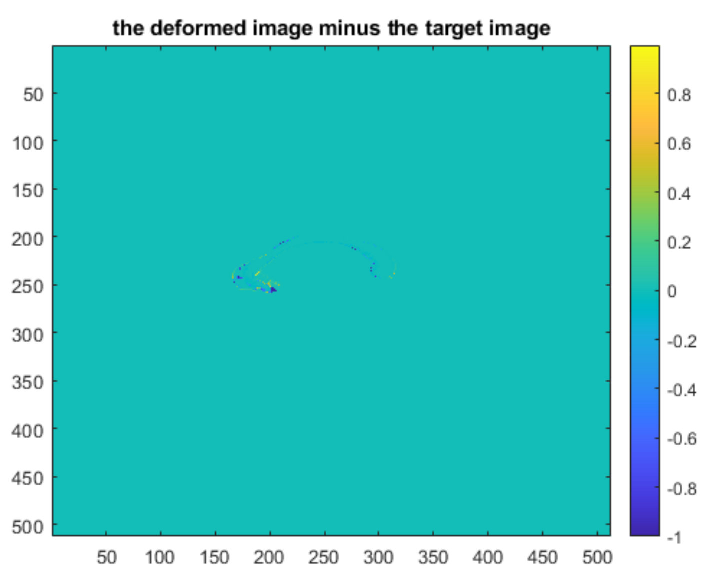
The initial cost: 47.4278

The final cost: 32.9020





Comparison:



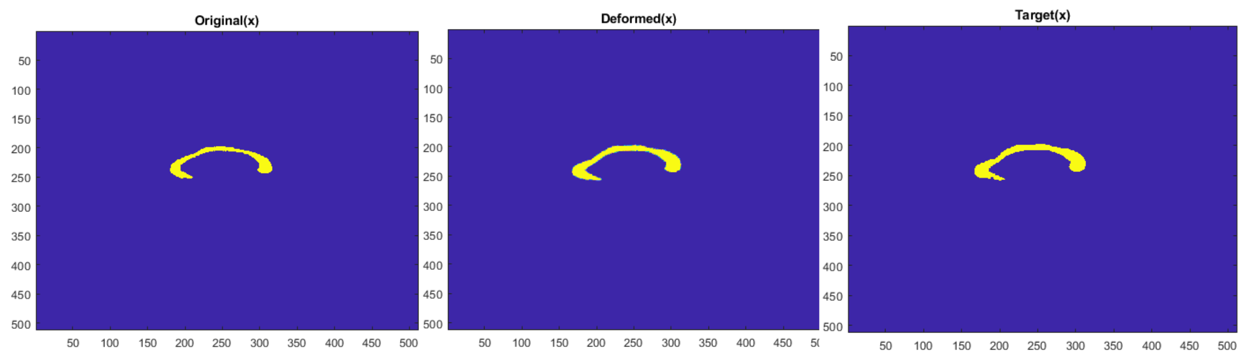
9.) For this question, I run splineImage directly on the template image. And use the output vx and vy to do the Jacobian calculations.

The input:

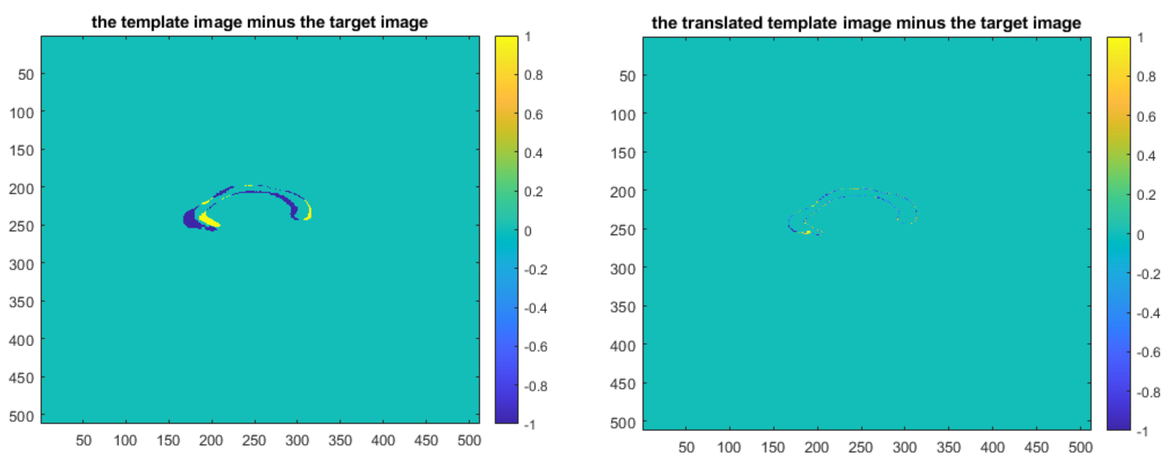
```
I = double(imread('0001_CC_Con.png') > 0);
J = double(imread('0003_CC_Alz.png') > 0);

epsilon = 0.005;
nIter = 4000;
sigma = 0.01;
alpha = 20;
```

The output:



The comparison:



Jacobian calculations :

$$\begin{bmatrix} x \\ y \end{bmatrix} \longrightarrow \begin{bmatrix} x + V_x \\ y + V_y \end{bmatrix}$$

$$D_f = \begin{bmatrix} \frac{d f_1}{d x} & \frac{d f_1}{d y} \\ \frac{d f_2}{d x} & \frac{d f_2}{d y} \end{bmatrix} = \begin{bmatrix} 1 + \frac{d V_x}{d x} & \frac{d V_x}{d y} \\ \frac{d V_y}{d x} & 1 + \frac{d V_y}{d y} \end{bmatrix}$$

$$\text{Determinant} = \left(1 + \frac{d V_x}{d x}\right)\left(1 + \frac{d V_y}{d y}\right) - \frac{d V_y}{d x}\frac{d V_x}{d y}$$

$$= 1 + \frac{d V_x}{d x} + \frac{d V_y}{d y} + \frac{d V_x}{d x}\frac{d V_y}{d y} - \frac{d V_y}{d x}\frac{d V_x}{d y}$$

```
%% Jacobian Calculations
[gradvx_x,gradvx_y] = gradient(vx);
[gradvy_x,gradvy_y] = gradient(vy);
deter = gradvx_x + gradvy_y + gradvx_x.*gradvy_y - gradvy_x .* gradvx_y +1;
```