B spline registration report

Xiaoxiao Zhou

March 1, 2019

1. Basic understanding – B spline
2. Calculation
3. Cubic B spline for registration (2D image currently)
4. B spline registration model design
5. Current result and analysis

1. Basic understanding – B spline

A linear combination of certain basis functions weighted by specific terms.
Personally speaking, B spline is an interpolation method by which we can calculate some locations' values by their neighboring control points. In this way, B spline can be used to do approximation, like curves, values.

2. Calculation

2.1 1-D data

$$b_{j,0}(t) := \begin{cases} 1 & t_j < t < t_{j+1} \\ 0 & \dots \end{cases}$$

$$b_{j,n}(t) := \frac{t - t_j}{t_{j+n} - t_j} b_{j,n-1}(t) + \frac{t_{j+n+1} - t}{t_{j+n+1} - t_{j+1}} b_{j+1,n-1}(t).$$

j – jth control point locaton

n – polynomial degree(0 constant, 1 linear, 2 quadratic, 3 cube)

$b_{j,d}$ (t) – b spline at point t, which is affected by its **n** neighbor: t_j, t_{j+1} , …, t_{j+n}.

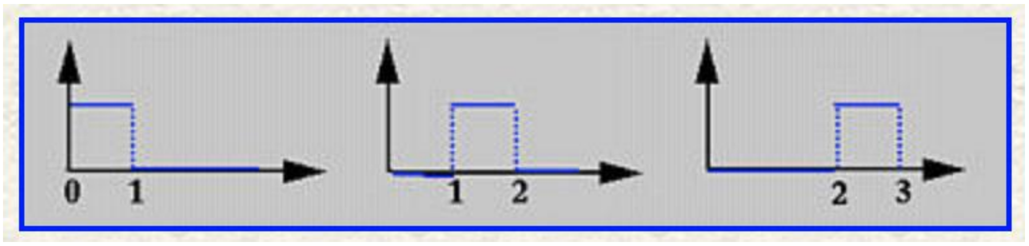
$\frac{t-t_j}{t_{j+n}-t_j}$ – proportion of t in the interval $[t_j, t_{j+n}]$

$\frac{t_{j+n+1}-t}{t_{j+n+1}-t_{j+1}}$ – proportion of t in the interval $[t_{j+1}, t_{j+n+1}]$

Therefore, b spline is only based on relative distance between t and its neighbor points, not relative with location t's corresponding value f(t). If we have uniform distance among control points, points in each interval, which have same relative distance from their control points, their splines will be the same. In this case, we could only calculate one interval's splines, then apply it for every point.

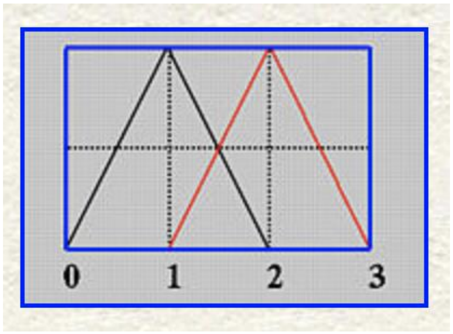
Usually we use **cubic** spline, which have higher degree for complex approximation and it is derivative.

(1) n = 0, $b_{j,0}$, constant, t is affected by 2 points: t_j, t_{j+1}



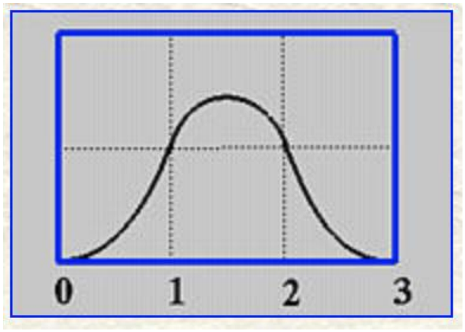
(2) n = 1, $b_{j,1}$, constant, t is affected by 3 points: t_j, t_{j+1}, t_{j+2}

$b_{0,1}(t) = t\, b_{0,0}(t) + (2 - t)\, b_{1,0}(t)$



(3) n = 2, $b_{j,2}$, constant, t is affected by 3 points: t_j, t_{j+1}, t_{j+2}

$$B_2^0(t) = t^2/2$$
$$B_2^1(t) = (-2t^2 + 2t + 1)/2$$
$$B_2^2(t) = (t^2 - 2t + 1)/2$$

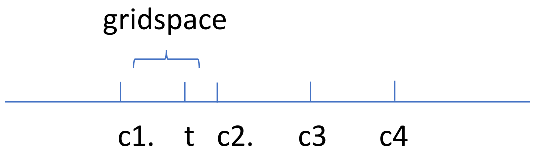


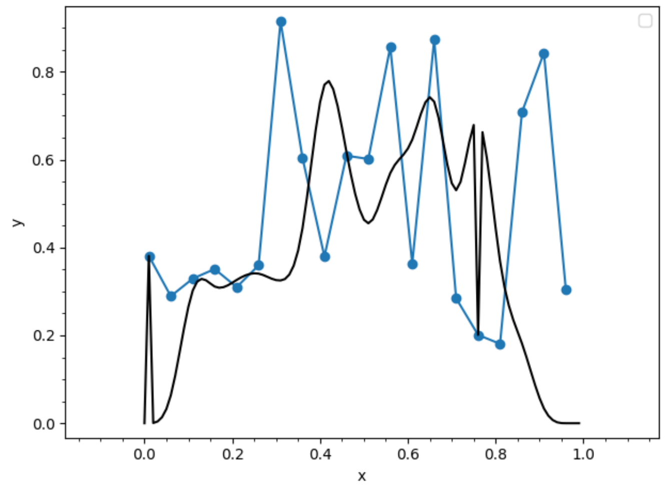
(4) n = 3, $b_{j,3}$, constant, t is affected by 4 points: $t_j, t_{j+1}, t_{j+2}, t_{j+3}$

$$B_3^0(t) = (-t^3 + 3t^2 - 3t + 1)/6$$
$$B_3^1(t) = (3t^3 - 6t^2 + 4)/6$$
$$B_3^2(t) = (-3t^3 + 3t^2 + 3t + 1)/6$$
$$B_3^3(t) = t^3/6$$

2.2 My implementation(cubic spline)

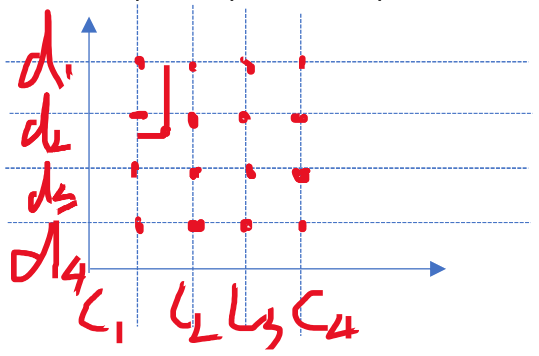
a. 1d signal

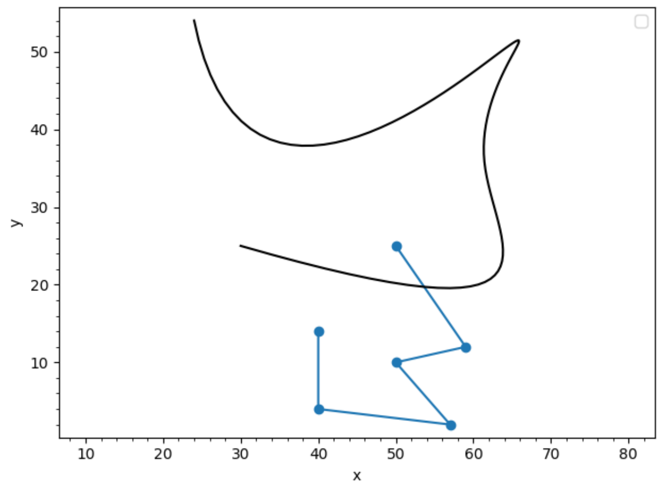




b. 2d

Spline can be calculated separately, on x and y axis.





3. Cubic B spline for registration (2D image currently)

3.1 Background

This registration method is based on deformation field – displacement vector field u. Displacement field is based on linear space relationship, recording every pixel in the moving image I_m displacement between it and its mapped point in fixed image(target) I_t.

3.2 Goal

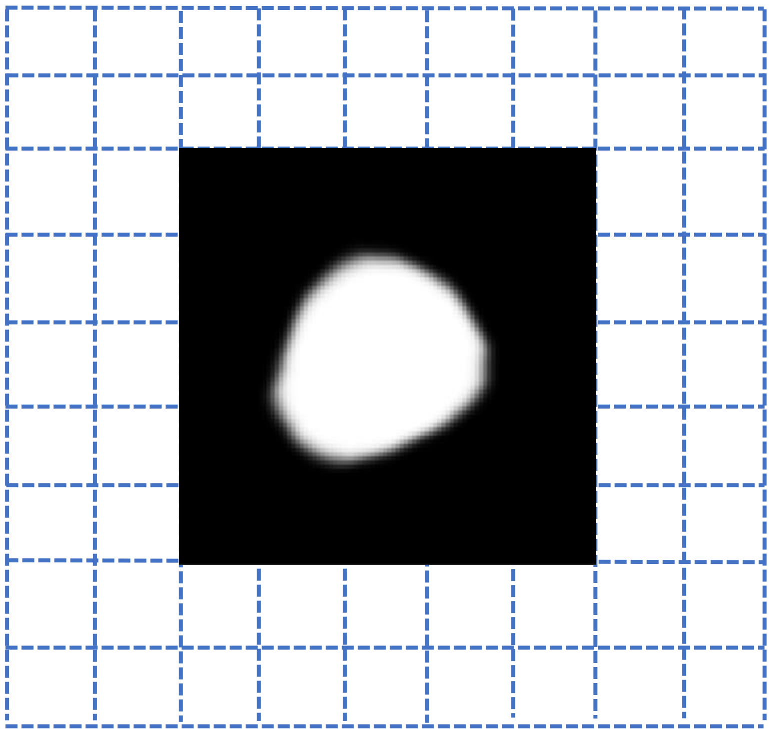
We would like to registered one or more images to a fixed image(target), to make them have SAME space, and be as similar with fixed image as possible. In this model, it is mainly based on displacement field.

3.3 Principle

(1). Control points (grid)

Control points can be chosen either uniformly or not. But for the image, we want every pixel to be surrounded by 4×4 control points. Therefore, if we have an image with size $x \times y$, grid space is $g_x \times g_y$, the grids(only has control points) will be:

$$grids = \lceil \frac{x}{g_x}, \frac{y}{g_y} \rceil + 4$$



(2). B spline for 2d image

This is different with 1d signal, but similar to 2d. Splines in different axis can be calculated separately.

For one pixel (t1, t2), spline at x axis t1:

(1) Find its surrounding control points

$$c(0,0) = \lfloor \frac{t1}{g_x} - 1, \frac{t2}{g_y} - 1 \rfloor$$

$$c(i,j) = \lfloor \frac{t1}{g_x} + i, \frac{t2}{g_y} + j \rfloor \qquad i,j = 0,1,2,3$$

(2) Calculate splines on x axis and y axis separately

X axis: $v_x(t1) = \sum_{m=0}^{d} \sum_{n=0}^{d} B_d^m(u) B_d^n(v) c_x(i,j), \quad i,j = 0,1,2,3$

y axis: $v_y(t2) = \sum_{m=0}^{d} \sum_{n=0}^{d} B_d^m(u) B_d^n(v) c_y(i,j), \quad i,j = 0,1,2,3$

$$u = \frac{t1 - c_x(i,j))}{grid\ space}$$

$$v = \frac{t2 - c_y(i,j))}{grid\ space}$$

d – degree(cubic, d = 3)

c(i, j) – 16 control points' displacement around (t1, t2)

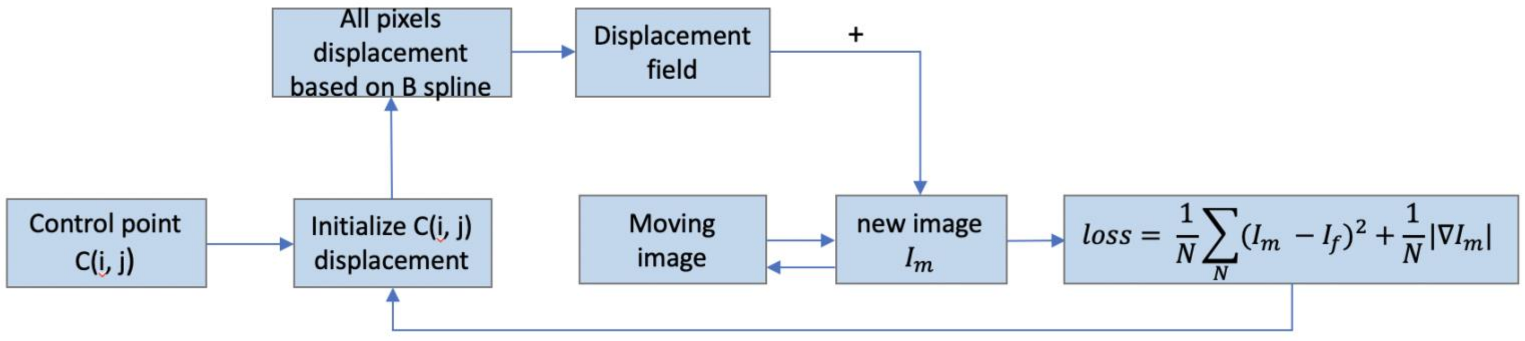
Above equations can be written in matrix.

$$v_x(t1) = \begin{bmatrix} B_d^0(u) \\ B_d^1(u) \\ B_d^2(u) \\ B_d^3(u) \end{bmatrix}^T \begin{bmatrix} c_x(i,j) & \cdots & c_x(i,j+3) \\ \vdots & \ddots & \vdots \\ c_x(i+3,j) & \cdots & c_x(i+3,j+3) \end{bmatrix} \begin{bmatrix} B_d^0(v) \\ B_d^1(v) \\ B_d^2(v) \\ B_d^3(v) \end{bmatrix}$$

$$v_y(t2) = \begin{bmatrix} B_d^0(u) \\ B_d^1(u) \\ B_d^2(u) \\ B_d^3(u) \end{bmatrix}^T \begin{bmatrix} c_y(i,j) & \cdots & c_y(i,j+3) \\ \vdots & \ddots & \vdots \\ c_y(i+3,j) & \cdots & c_y(i+3,j+3) \end{bmatrix} \begin{bmatrix} B_d^0(v) \\ B_d^1(v) \\ B_d^2(v) \\ B_d^3(v) \end{bmatrix}$$

4. B spline registration model design

4.1 Process

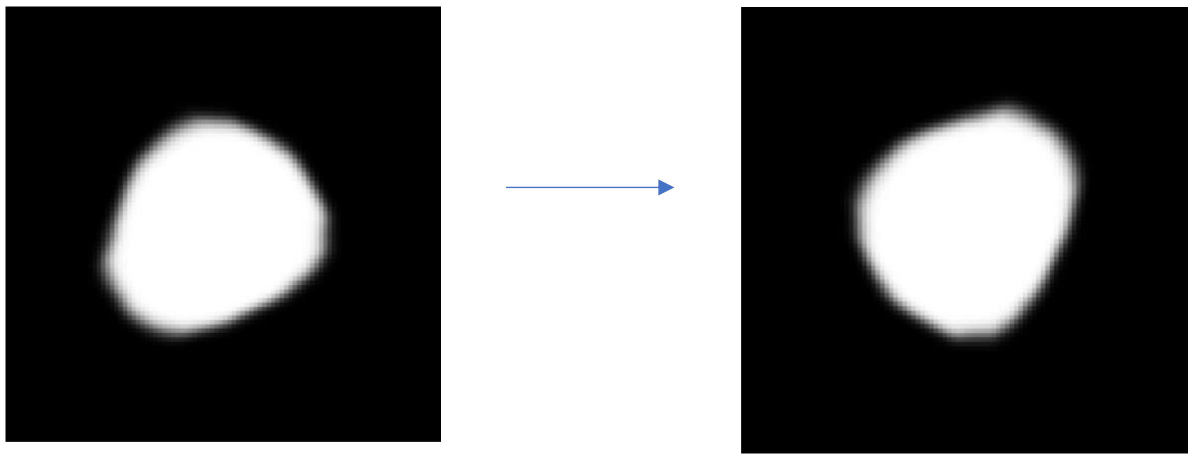


All pixels displacement based on B spline
Displacement field
+
Control point C(i, j)
Initialize C(i, j) displacement
Moving image
new image I_m
$loss = \frac{1}{N}\sum_N (I_m - I_f)^2 + \frac{1}{N}|\nabla I_m|$

Optimize: gradient descent

4.2 Model design

(1). Moving and fixed images: gray [300, 300]

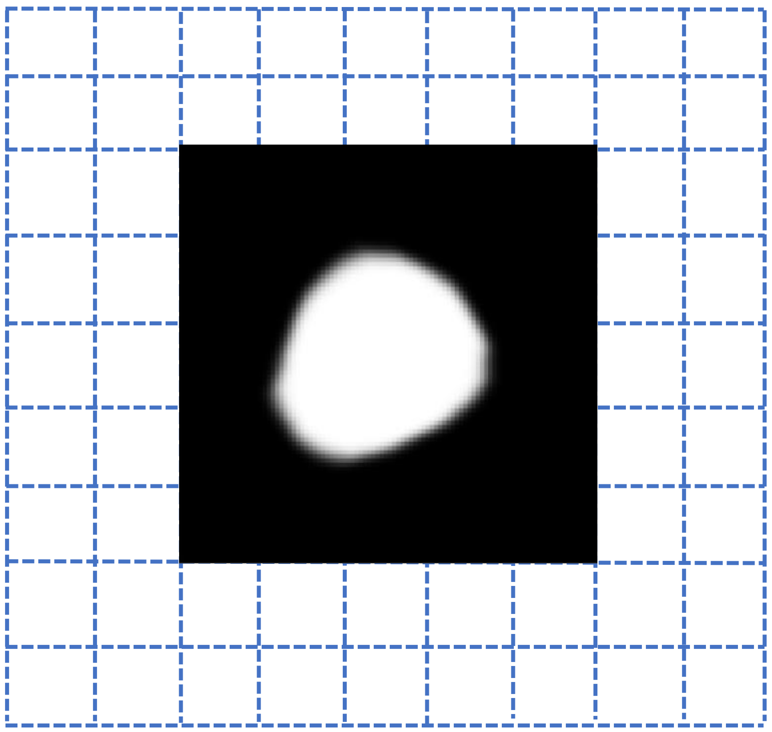


(2). Control points

Grid space = 10

Control points number = 34 (actually 15, but it does not matter)

Displacement field = $[34 \times 10, 34 \times 10]$



(3). B spline computation(backward warping)

a. **Traverse every pixel on the image (a new image for saving registered image every iteration)**

b. **compute every pixel's b spline to approximate its displacement, based on its neighboring 16 control points' displacement**

for one pixel with (t1, t2), its bspline:

$$v_x(t1) = \begin{bmatrix} B_d^0(u) \\ B_d^1(u) \\ B_d^2(u) \\ B_d^3(u) \end{bmatrix}^T \begin{bmatrix} c_x(i,j) & \cdots & c_x(i,j+3) \\ \vdots & \ddots & \vdots \\ c_x(i+3,j) & \cdots & c_x(i+3,j+3) \end{bmatrix} \begin{bmatrix} B_d^0(v) \\ B_d^1(v) \\ B_d^2(v) \\ B_d^3(v) \end{bmatrix}$$

$$v_y(t2) = \begin{bmatrix} B_d^0(u) \\ B_d^1(u) \\ B_d^2(u) \\ B_d^3(u) \end{bmatrix}^T \begin{bmatrix} c_y(i,j) & \cdots & c_y(i,j+3) \\ \vdots & \ddots & \vdots \\ c_y(i+3,j) & \cdots & c_y(i+3,j+3) \end{bmatrix} \begin{bmatrix} B_d^0(v) \\ B_d^1(v) \\ B_d^2(v) \\ B_d^3(v) \end{bmatrix}$$

Therefore, pixel (t1, t2)'s displacement is ($v_x(t1), v_y(t1)$), then update displacement field. After traversing all pixels, a displacement field is formed.

c. Construct new image based on displacement field

Using backward warping method.(from new image back to moving image to find intensity)

d. calculate loss between new and fixed image

$$loss = \frac{1}{N}\sum_{N}((I_m - I_f)^2 + |\nabla I_m|)$$

e. Optimization: gradient descent

Optimization objective: displacement of control points c(i, j)

$$\frac{\partial \sum_N (I_m - I_f)^2}{\partial I_m} = \frac{2}{N}(I_f - I_m)$$

$\frac{\partial \sum_N |\nabla I_m|}{\partial I_m}$ is calculated as below:

$u = Im.$

$$\frac{\partial \int |\nabla Im|}{\partial Im} = \frac{\partial \int |\nabla u|}{\partial u}, \quad |\nabla u| = \langle \nabla u, \nabla u \rangle^{\frac{1}{2}}$$

$\therefore E(u) \to E(u+\varepsilon h)$, only for regularizer term.

$$\frac{\partial E(u+\varepsilon h)}{\partial \varepsilon} = \frac{\partial}{\partial \varepsilon} \int \langle \nabla(u+\varepsilon h), \nabla(u+\varepsilon h) \rangle^{\frac{1}{2}}$$

$$= \int \frac{1}{2} \underset{\text{for } u}{\langle \nabla(u+\varepsilon h), \nabla(u+\varepsilon h)\rangle^{-\frac{1}{2}}} \cdot (\langle \nabla(u+\varepsilon h), \nabla h \rangle) \times 2.$$

$$= \int \langle \nabla u, \nabla u \rangle^{-\frac{1}{2}} \cdot \langle \nabla u, \nabla h \rangle$$

$$= \int \frac{\langle \nabla u, \nabla h \rangle}{|\nabla u|}$$

$$\int \langle \nabla u, \nabla h \rangle = \int \langle \begin{bmatrix} \frac{\partial u}{\partial x} \\ \frac{\partial u}{\partial y} \end{bmatrix}, \begin{bmatrix} \frac{\partial h}{\partial x} \\ \frac{\partial h}{\partial y} \end{bmatrix} \rangle$$

$$\because \int f(x) \cdot g'(x)\, dx = f(x) \cdot g(x) - \int f'(x) \cdot g(x)\, dx.$$

$$\therefore \int \langle \nabla u, \nabla h \rangle = \int - \langle \left[\frac{\partial}{\partial x} \ \frac{\partial}{\partial y}\right] \begin{bmatrix} \frac{\partial u}{\partial x} \\ \frac{\partial u}{\partial y} \end{bmatrix}, h \rangle \quad (\nabla u \cdot h = 0)$$

$$= \int - \langle \underbrace{\frac{\partial^2 u}{\partial x^2}, \frac{\partial^2 u}{\partial y^2}}_{\text{divergence}}, h \rangle = -div(\nabla u)$$

$$\therefore \frac{\partial \int |\nabla Im|}{\partial Im} = \frac{\partial \int |\nabla u|}{u} = \int \frac{-div(\nabla u)}{|\nabla u|}.$$

$$\frac{\partial loss}{\partial I_m} = \frac{2}{N}\sum_N (I_f - I_m) - lamda * \frac{div(\nabla I_m)}{|I_m|}$$

$$\begin{cases} \dfrac{\partial I_m}{\partial x} = \sum_N \dfrac{(I_m(x+1) - I_m(x-1))}{2} \\ \dfrac{\partial I_m}{\partial y} = \sum_N \dfrac{(I_m(y+1) - I_m(y-1))}{2} \end{cases}$$

Because:

X axis: $v_x(t1) = \sum_{m=0}^{d}\sum_{n=0}^{d} B_d^m(u)B_d^n(v)c_x(i,j), \quad i,j = 0,1,2,3$

y axis: $v_y(t2) = \sum_{m=0}^{d}\sum_{n=0}^{d} B_d^m(u)B_d^n(v)c_y(i,j), \quad i,j = 0,1,2,3$

$$\frac{\partial x}{\partial c_x} = \frac{\partial y}{\partial c_y} = \sum_{m=0}^{d}\sum_{n=0}^{d} B_d^m(u)B_d^n(v)$$

As a result, full gradient formula:

$$\frac{\partial loss}{\partial c_x} = \frac{\partial loss}{\partial I_m}\frac{\partial I_m}{\partial x}\frac{\partial x}{\partial c_x} = \left(\frac{2}{N}\sum_N (I_f - I_m) - lamda * \frac{div(\nabla I_m)}{|I_m|}\right)\sum_N \frac{(I_m(x+1) - I_m(x-1))}{2}\sum_{m=0}^{d}\sum_{n=0}^{d} B_d^m(u)B_d^n(v)$$

$$\frac{\partial loss}{\partial c_y} = \frac{\partial loss}{\partial I_m}\frac{\partial I_m}{\partial x}\frac{\partial x}{\partial c_x} = \left(\frac{2}{N}\sum_N (I_f - I_m) - lamda * \frac{div(\nabla I_m)}{|I_m|}\right)\sum_N \frac{(I_m(y+1) - I_m(y-1))}{2}\sum_{m=0}^{d}\sum_{n=0}^{d} B_d^m(u)B_d^n(v)$$

$(\nabla c_x, \nabla c_y) = (\frac{\partial loss}{\partial c_x}, \frac{\partial loss}{\partial c_y})$ is last updated control point’s new displacement, but for the original control point, its displacement to this new point is $(\sum \nabla c_x, \sum \nabla c_y)$

Then update control points’ displacement:

$$displacement\ field = displacement\ field - lr * (\nabla c_x, \nabla c_y)$$

lr – learning rate

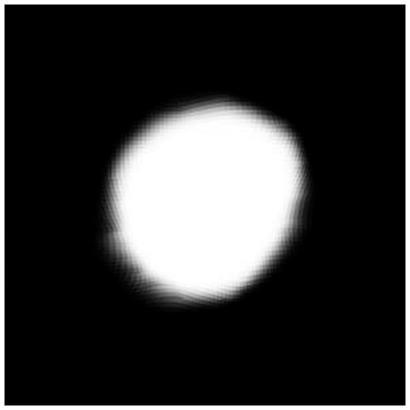
5. Current result and analysis

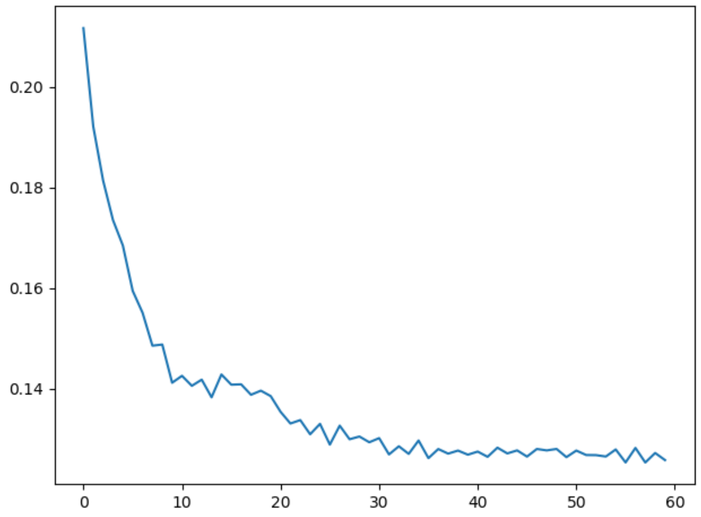
(1). First experiment

Learning rate = 0.001

Lamda = 0.0001

Output:

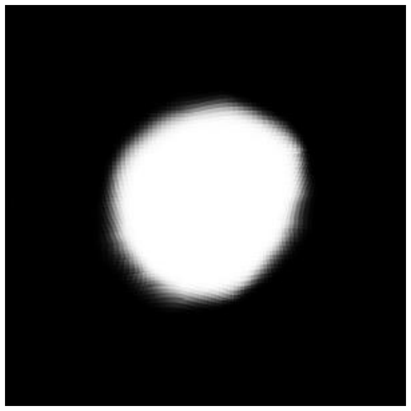


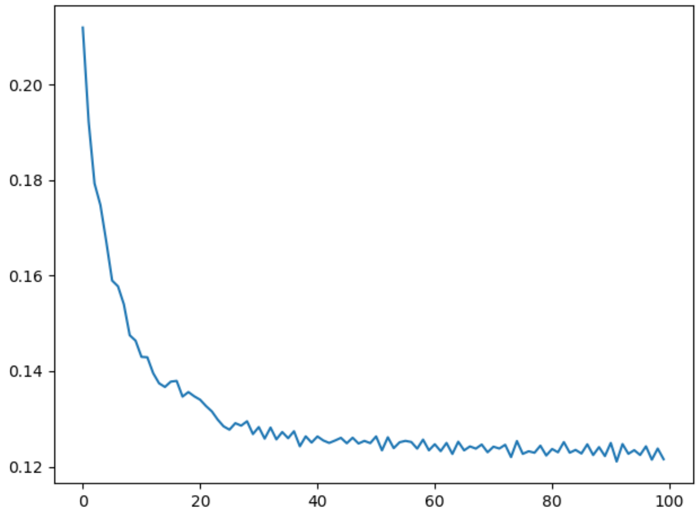


Learning rate = 0.001

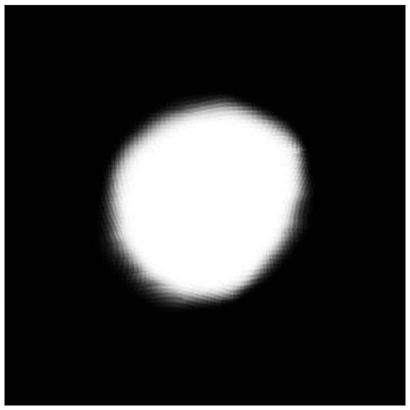
Lamda = 0.00001

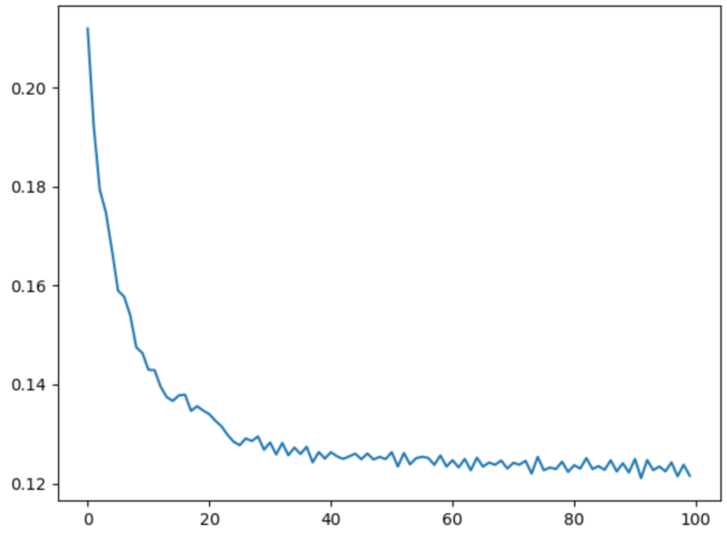
Output:





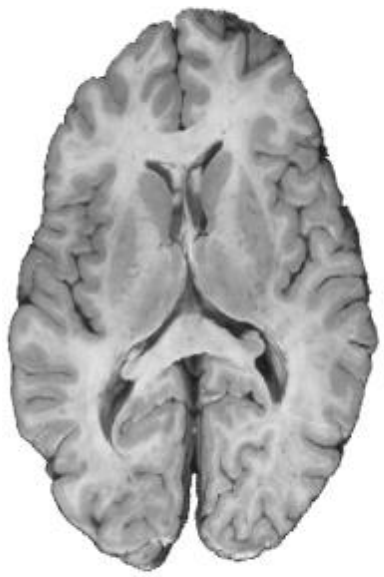
Learning rate = 0.001
Lamda = 0.000005
Output:



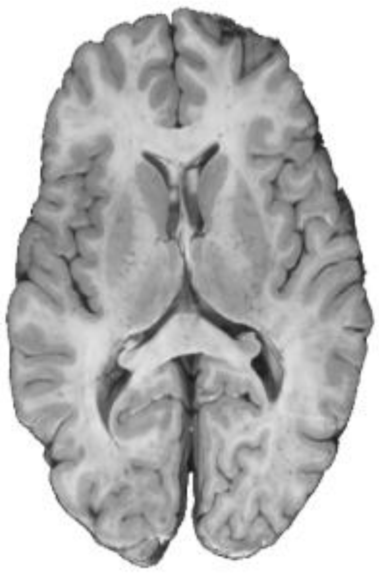


(2). Second experiment

Source:

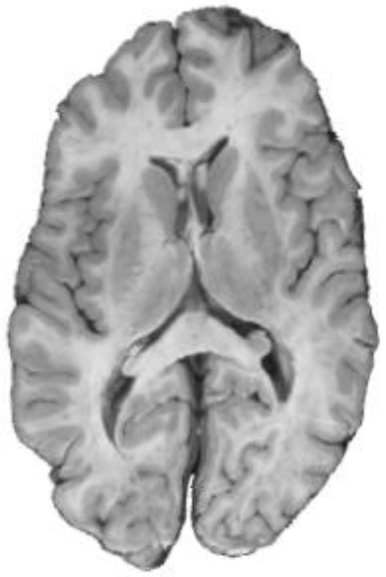


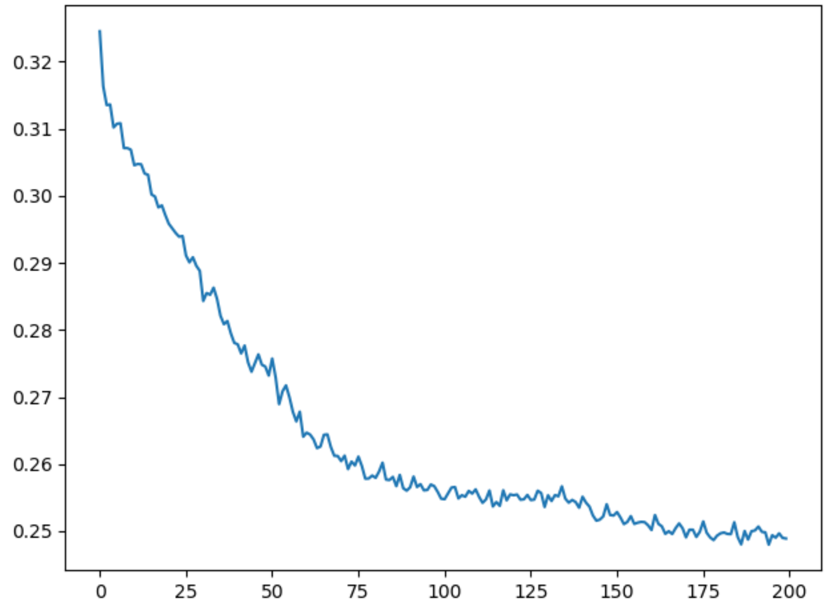
target:



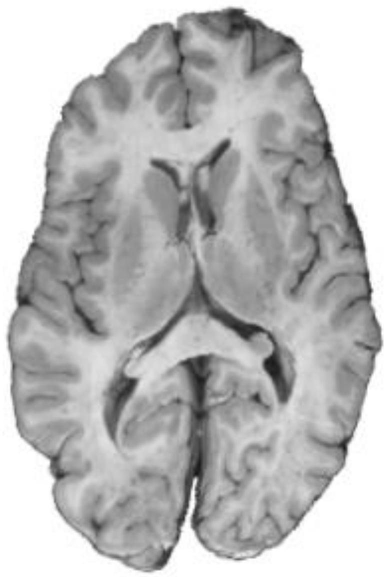
Results:

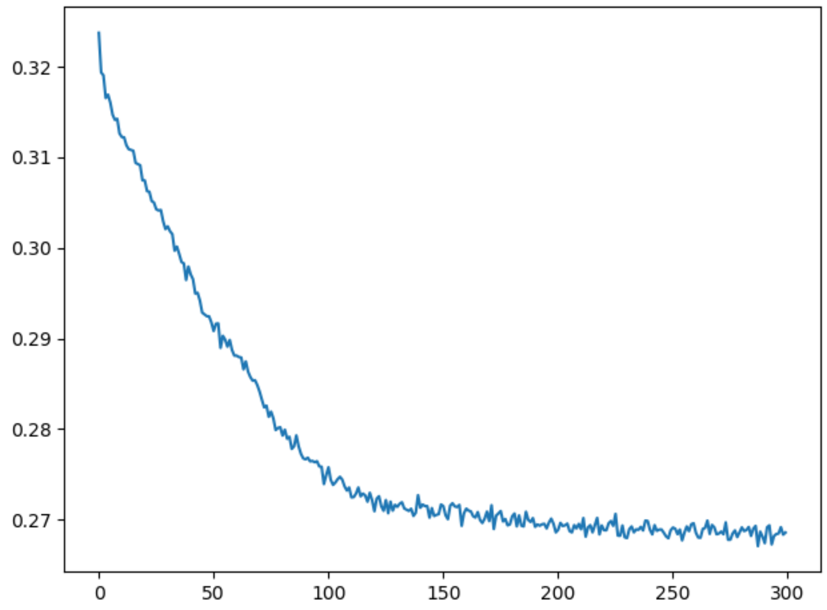
Learning rate = 0.0005
Lamda = 0. 001





Learning rate = 0.001
Lamda = 0. 001





0.32
0.31
0.30
0.29
0.28
0.27
0
50
100
150
200
250
300