%f,g: 2 functions

%n: points of f

%m: points of g

%This function search neiborghood v

function [path,E]=sldp(f,g)

c=10;

n=length(f);

E=zeros(n,n);

E(1,:)=c;

E(:,1)=c;

E(1,1)=0;

v=[1,1;2,1;3,1;4,1;5,1;6,1;1,2;1,3;1,4;1,5;1,6;2,3;3,2;3,4;4,3;2,5;3,5;4,5;5,2;5,3;5,4;5,6;6,5;

1,7;2,7;3,7;4,7;5,7;6,7;7,1;7,2;7,3;7,4;7,5;7,6;1,8;3,8;5,8;7,8;8,7;8,5;8,3;8,1];

for i=2:n;

for j=2:n;

for r=1:size(v,1);

k=i-v(r,1);

l=j-v(r,2);

if (k>0 && l>0)

CandE(r) = E(k,l) + energy(f,g,k,l,i,j);

else

CandE(r)=c;

end

end

[E(i,j),idx] =min(CandE);

path(i,j,1) = i-v(idx,1);

path(i,j,2) = j-v(idx,2);

end

end

%reconstruct gamma

x(1) = n;

y(1) = n;

cnt = 1;

while x(cnt)>1;

x(cnt+1) = path(x(cnt),y(cnt),1);

y(cnt+1) = path(x(cnt),y(cnt),2);

cnt = cnt+1;

end

path=[x',y'];

function cost=energy(f,g,k,l,i,j)

%This is the function to calculate 2-norm distence between two fuction

%between path from (k,l) to (i,j) used in dynamic programming

n=length(f);

m=length(g);

slope=(j-l)/(i-k);

gidx=round((l+((k+1:i)-k).\*slope)/n\*m);

cost=norm(f(k+1:i)-g(gidx))^2/n;

clear all;

close all;

%define functions

n=100;

m=100;

x=[0:n]/n;

y=[0:m]/m;

f=normpdf(x,0.2,0.05)+normpdf(x,0.4,0.05);

g=normpdf(y,0.5,0.05)+normpdf(x,0.7,0.05);

[path, E]=sldp(f,g);

close all;

fig=figure();

set(fig,'Position', [200 200 1600 450]);

subplot(131);plot(f);hold on;plot(g((1:n)\*m/n));title('before registration');

set(gca,'XTickLabel',[],'YTickLabel',[]);

subplot(132);imagesc(E');colormap(gray);hold on;plot(path(:,1),path(:,2));axis xy;

axis equal;set(gca,'XTickLabel',[],'YTickLabel',[]);

title('Warping function');

subplot(133);plot(f);hold on;

plot((1:n),g(round(interp1(path(:,1),path(:,2),1:n)\*m/n)));title('after registered');

set(gca,'XTickLabel',[],'YTickLabel',[]);