Házi feladat

# A médiatechnológia alapjai

### A feladat megfogalmazása

A tanár által a diákok számára rendelkezésre bocsátott, Markov véletlen mezőkön alapuló modell működését leíró programkódok kiegészítése a Metropolis és Gibbs Sampler algoritmusokkal (képszegmentálási feladat Matlab környezetben). A Metropolis, MMD, ICM és Gibbs Sampler algoritmusok összehasonlítása az MRF Segmenter nevű programmal.

### A kiegészített Metropolis kód

function mrf=gmrf\_doMetropolis(handles)

mrf=handles.MRF;

h=mrf.imagesize(1);

w=mrf.imagesize(2);

cnum=mrf.classnum;

betapar=mrf.Beta;

showedImage=uint8(zeros(h,w));

%---------------------------

threshSummadelta=mrf.DeltaUmin;

summa\_deltaE=2\*threshSummadelta;

cycle=0;

t=mrf.T0;

while summa\_deltaE > threshSummadelta

rand('state',sum(100\*clock));

cindArray=min(cnum,1+floor(cnum\*rand(h,w)));

cycle=cycle+1;

summa\_deltaE=0;

for y=1:h

for x=1:w

y1=max(1,y-1);

y2=min(y+1,h);

x1=max(1,x-1);

x2=min(x+1,w);

actClass=mrf.classmask(y,x);

actClassN1=mrf.classmask(y1,x);

actClassN2=mrf.classmask(y2,x);

actClassN3=mrf.classmask(y,x1);

actClassN4=mrf.classmask(y,x2);

actLogProb=mrf.logProbs{actClass}(y,x);

u\_old=actLogProb+ betapar\* ( (actClass~=actClassN1)+(actClass~=actClassN2)+(actClass~=actClassN3)+(actClass~=actClassN4))- betapar\* ( (actClass==actClassN1)+(actClass==actClassN2)+(actClass==actClassN3)+(actClass==actClassN4));

cind=cindArray(y,x);

if cind ~= actClass

newLogProb=mrf.logProbs{cind}(y,x);

u\_new=newLogProb+ betapar\* ( (cind~=actClassN1)+(cind~=actClassN2)+(cind~=actClassN3)+(cind~=actClassN4))-betapar\* ( (cind==actClassN1)+(cind==actClassN2)+(cind==actClassN3)+(cind==actClassN4));

deltaU=u\_new-u\_old;

ksi=rand();

if(ksi==1)

ksi=0;

end;

if (ksi < exp(-deltaU/t))

actClass=cind;

mrf.classmask(y,x)=cind;

summa\_deltaE=summa\_deltaE+abs(deltaU);

end;

end;

showedImage(y,x)=uint8(mrf.gauss\_params{actClass}.mean);

end;

end;

t=t\*mrf.c;

%---------------------------

image(showedImage,'Parent',handles.axes2);

set(handles.axes2,'XTick',[],'YTick',[])

cycle

pause(0.1)

end;

### A kiegészített Gibbs Sampler kód

function mrf=gmrf\_doGibbsSampler(handles)

mrf=handles.MRF;

h=mrf.imagesize(1);

w=mrf.imagesize(2);

cnum=mrf.classnum;

betapar=mrf.Beta;

showedImage=uint8(zeros(h,w));

t=mrf.T0;

threshSummadelta=mrf.DeltaUmin;

cycle=0;

summa\_deltaE=2\*threshSummadelta;

while summa\_deltaE > threshSummadelta

cycle=cycle+1;

summa\_deltaE=0;

rand\_num = rand(1);

for y=1:h

for x=1:w

y1=max(1,y-1);

y2=min(y+1,h);

x1=max(1,x-1);

x2=min(x+1,w);

actClass=mrf.classmask(y,x);

actClassN1=mrf.classmask(y1,x);

actClassN2=mrf.classmask(y2,x);

actClassN3=mrf.classmask(y,x1);

actClassN4=mrf.classmask(y,x2);

actLogProb=mrf.logProbs{actClass}(y,x);

u\_old=actLogProb+ betapar\* ( (actClass~=actClassN1)+(actClass~=actClassN2)+(actClass~=actClassN3)+(actClass~=actClassN4))- betapar\* ( (actClass==actClassN1)+(actClass==actClassN2)+(actClass==actClassN3)+(actClass==actClassN4));

% új változók definiálása

u\_local\_sum = 0;

z = 0;

rand('state',sum(100\*clock));

for cind=1:cnum

if cind ~= actClass

newLogProb=mrf.logProbs{cind}(y,x);

u\_new=newLogProb+ betapar\* ( (cind~=actClassN1)+(cind~=actClassN2)+(cind~=actClassN3)+(cind~=actClassN4))-betapar\* ( (cind==actClassN1)+(cind==actClassN2)+(cind==actClassN3)+(cind==actClassN4));

u\_local = exp(-u\_new/t);

u\_local\_sum = u\_local\_sum + u\_local;

z = u\_local/u\_local\_sum;

if rand\_num < z

actClass=cind;

mrf.classmask(y,x)=cind;

end;

end;

end;

summa\_deltaE=summa\_deltaE+abs(u\_new-u\_old);

showedImage(y,x)=uint8(mrf.gauss\_params{actClass}.mean);

end;

end;

t=t\*mrf.c;

image(showedImage,'Parent',handles.axes2);

set(handles.axes2,'XTick',[],'YTick',[])

cycle

%pause(0.1)

end

### Az algoritmusok összehasonlítása

A tesztkép és tesztprogram segítségével végeztem el az algoritmusok összehasonlítását.

Változtassátok a tesztképet, paramétereket, különösképpen a zajt és a béta paramétert, és írjátok le a tapasztalataitokat, illusztráljátok sok képpel!