CSE 473-573 Fall 2010 Nov 3, 2010

Homework#3 Solution Set

1. RGBConn.m

```
function OutIm = RGBConn(InIm,c_type)
    ctr = 1;
    InImg = InIm;
    [nr nc] = size(InIm);
    while(1)

        InImgR = InImg(:,:,1);
        InImgG = InImg(:,:,2);
        InImgB = InImg(:,:,3);
        InImgGrey = sqrt(InImgR.^2 + InImgG.^2 + InImgB.^2);
        logicalImg = InImgGrey>=0.5;
        bwLogicalImg = bwlabel(logicalImg,c_type);
        if(max(max(bwLogicalImg))==1 ||max(max(bwLogicalImg))==0)
            OutIm(:,:,1) = InImgR;
            OutIm(:,:,2) = InImgG;
            OutIm(:,:,3) = InImgB;
            break;
        end
        B = ones(ctr,ctr);
        ctr = ctr + 5;
        OutImgR = imclose(InImgR,B);
        OutImgG = imclose(InImgG,B);
        OutImgB = imclose(InImgB,B);
        InImg(:,:,1) = OutImgR;
        InImg(:,:,2) = OutImgG;
        InImg(:,:,3) = OutImgB;
    end
end
```

2. Granfn.m

```
function G_psi = GranFn(InIm,B)
    G_psi = zeros(size(InIm),'uint8');
    % create 4-connected NHOOD
    NHOOD4N = zeros(3,3);
    NHOOD4N(2,2) = 1;
    NHOOD4N(1,2) = 1;
    NHOOD4N(3,2) = 1;
    NHOOD4N(2,1) = 1;
    NHOOD4N(2,3) = 1;
    % create NHOOD for opening
    NHOOD = Set2NHOOD(B);
    while(sum(sum(InIm))~=0)
        InIm2 = imopen(logical(InIm),NHOOD);
        G_psi = G_psi + uint8(InIm2);
        InIm = InIm2;
        NHOOD = imdilate(logical(NHOOD),NHOOD4N,'full');
    end
end

function NHOOD = Set2NHOOD(B)
```

```
    maxR = max(abs(B(:,2)));
    maxC = max(abs(B(:,1)));
    numPts = size(B,1);
    NHOOD = zeros(2*maxR + 1,2*maxC + 1);
    c = floor((size(NHOOD) + 1)/2);
    cR = c(1);
    cC = c(2);
    for n = 1 : numPts
        r = cR - B(n,2);
        c = cC + B(n,1);
        NHOOD(r,c) = 1;
    end
end

3. MCCD.m
function CD=MCCD(cc,n)
    sNum=GetSamplingNum(cc,n);
    ccIdx=Chaincode2Matrix(cc);
    chord=zeros(sNum,1);
    for sIdx=1:sNum
       [iStar,jStar]=GetSampleNum(length(cc),ccIdx);
       [iEnd,jEnd]=GetSampleNum(length(cc),ccIdx);
       chord(sIdx)=GetLen(iStar, jStar, iEnd, jEnd);
    end
    [CD,Loc]=hist(chord,n);
end

function mat=Chaincode2Matrix(cc)
    cLen=length(cc);
    ccIdx=zeros(2*cLen,cLen);
    ccIdx(cLen,1)=1;
    iCur=cLen;
    jCur=1;
    for cIdx=1:cLen
        switch cc(cIdx)
            case '1'
                iCur=iCur-1;
                jCur=jCur+1;
            case '2'
                iCur=iCur-1;
            case '3'
                iCur=iCur-1;
                jCur=jCur-1;
            case '4'
                jCur=jCur-1;
            case '5'
                iCur=iCur+1;
                jCur=jCur-1;
            case '6'
                iCur=iCur+1;
            case '7'
                iCur=iCur+1;
                jCur=jCur+1;
            case '0'
                jCur=jCur+1;
            otherwise
                ccIdx=-1;
```

```
                    return;
                end
                ccIdx(iCur,jCur)=cIdx;
            end
            iIdx=find(sum(ccIdx,1)>0);
            jIdx=find(sum(ccIdx,2)>0);
            mat=ccIdx(jIdx(1):jIdx(length(jIdx)),iIdx(1):iIdx(length(iIdx)));
        end

        function sNum=GetSamplingNum(cc,n)
            m=length(cc);
            sNum=n*m;
        end

        function len=GetLen(iStar, jStar, iEnd, jEnd)
            len=sqrt((iStar-iEnd)^2+(jStar-jEnd)^2);
        end

        function [iIdx,jIdx]=GetSampleNum(ccMaxNum,ccIdx)
            samNum=ceil(rand()*ccMaxNum);
            [iIdx,jIdx]=find(ccIdx==samNum);
        end
```

4. Moment_script.m

```
        function muRs=Moment_script
            fileName=input('Please input the name of a bitmap file','s');
            M=input('Please input the maximum desired moment index M');
            N=input('Please input the maximum desired moment index N');
            if (~isnumeric(M)|~isnumeric(N))
                display('Input index M and N should be positive integer!');
                muRs=-1;
                return;
            end
            if (M<0 |N<0)
                display('Input index M and N should be positive integer!');
                muRs=-1;
                return;
            end
            InIm=imread(fileName);
            muRs=zeros(M,N);
        %     m00=GetMoment(InIm, 0, 0);
        %     m11=GetMoment(InIm, 1, 1);
        %     m10=GetMoment(InIm, 1, 0);
        %     m01=GetMoment(InIm, 0, 1);
        %     mu11=m11-m01*m10/m00;
            [xc,yc]=GetXYc(InIm);
            for p=1:M
                for q=1:N
                    muRs(p,q)=GetCentralMoment(InIm, p, q, xc, yc);
                end
            end
        end

        function [xc,yc]=GetXYc(InIm)
            m10=GetMoment(InIm, 1, 0);
```

```
172       m01=GetMoment(InIm, 0, 1);
173       m00=GetMoment(InIm, 0, 0);
174       xc=m10/m00;
175       yc=m01/m00;
176   end
177
178   function rs=GetMoment(InIm, p, q)
179       [iDim,jDim]=size(InIm);
180       iVec=[];
181       jVec=[];
182       for idx=1:iDim
183           iVec=[iVec idx^p];
184       end
185       for jdx=1:jDim
186          jVec=[jVec; jdx^q];
187       end
188       rs=iVec*double(InIm)*jVec;
189   end
190
191   function rs=GetCentralMoment(InIm, p, q, xc, yc)
192       [iDim,jDim]=size(InIm);
193       iVec=[];
194       jVec=[];
195       for idx=1:iDim
196           iVec=[iVec (idx-xc)^p];
197       end
198       for jdx=1:jDim
199          jVec=[jVec; (jdx-yc)^q];
200       end
201       rs=iVec*double(InIm)*jVec;
202   end
```