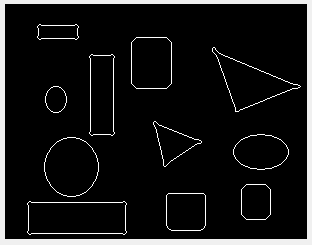
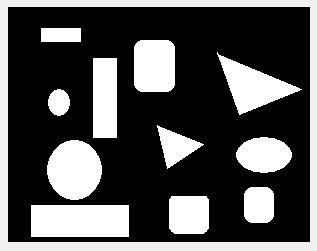
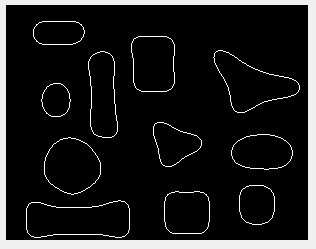
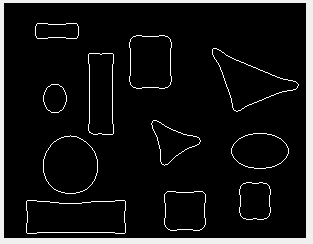
|  |
| --- |
| **YILDIZ TEKNİK ÜNİVERSİTESİ – BİLGİSAYAR MÜHENDİSLİĞİ** |
| **Bilgisayarla Görme 1. Ödevi** |
| **Logfilter, Canny, Connected Component Labeling, Erosion** |

|  |
| --- |
| Melike Nur Mermer - 15501010  28.03.2017 |

**1-**

Log filtresinin sigma katsayısı değiştirilerek “regions.pgm” görüntüsü üzerinde kenar tespiti uygulamaları gerçekleştirilmiştir. Girişte kullanılan görüntü ve her sigma değeri için elde edilen çıkış görüntüleri aşağıda verilmektedir.

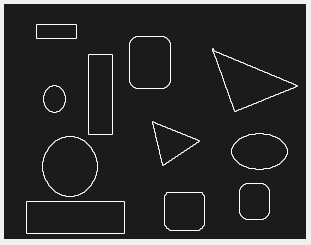
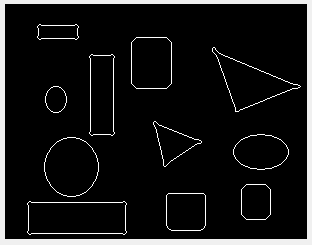
 

giriş sigma=1 sigma=5 sigma=10

Bu görüntülere bakıldığında logfilter ile elde edilen çıkış görüntülerinde köşelerin yuvarlatılmış olduğu görülmektedir. Sigma katsayısı arttıkça bu yuvarlatmanın çapı da büyümektedir, yani köşeleri daha yuvarlak olan çıkışlar oluşmaktadır. Logfilter keskin köşeleri olduğu şekilde göstermemektedir.

**2-**

Canny kenar bulma algoritması “regions.pgm” görüntüsü üzerinde uygulanarak elde edilen çıkışlar log filtresi ile karşılaştırılmıştır. Görüntüden logfilter ve canny ile çıkartılan kenarlar aşağıda verilmektedir.



logfilter canny

İki yöntemle elde edilen çıkışlara bakıldığında üçgen ve dikdörtgen şekillerin köşelerinin canny ile daha keskin şekilde belirlenebildiği görülmektedir. Ayrıca logfilter ile köşeleri yuvarlak olan dörtgen şekillerin yuvarlak köşelerinin düz kenar şeklinde çıktığı görülmektedir. Canny ise bu şekillerde köşeleri gerçeğine daha yakın olarak çıkartabilmektedir.

**3-**

1. Görüntüdeki bağlı bileşenlerin bulunması için “connected component labeling” algoritması gerçeklenmiştir. Algoritmanın 4 komşuluk ve 8 komşuluk için “semboller.pgm” ve “labirent.pgm” görüntüleri üzerinde uygulanması sonucunda elde edilen çıkış görüntüleri dosya ekinde .pgm formatında mevcuttur.
2. “semboller.pgm” 4 komşuluk (doğu-batı-kuzey-güney) yönünde bağlı bileşenlerin sayısı = 20

“semboller.pgm” 8 komşuluk (d-b-k-g-kd-kb-gd-gb) yönünde bağlı bileşenlerin sayısı = 6

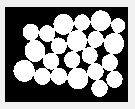
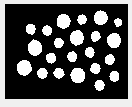
“labirent.pgm” 4 komşuluk (doğu-batı-kuzey-güney) yönünde bağlı bileşenlerin sayısı = 19

“labirent.pgm” 8 komşuluk (d-b-k-g-kd-kb-gd-gb) yönünde bağlı bileşenlerin sayısı = 19

Bu sonuçlara göre “semboller.pgm” görüntüsünde çapraz komşularda dolu pikseller olduğu için 8 yönlü algoritma ile elde edilen bağlı bileşenlerin sayısı daha az bulunmuştur. “labirent.pgm” görüntüsünde ise bileşenlerin çapraz komşularında dolu pikseller bulunmadığı için 4 komşuluk ve 8 komşuluk ile elde edilen bağlı bileşen sayıları arasında fark görülmemektedir.

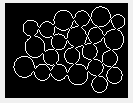
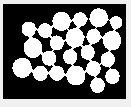
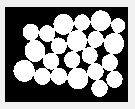
**4-**

1. Birbiri ile temas halinde bulunan metal paraları birbirinden ayırıp sayısını bulabilmek için “coins.pgm” görüntüsü üzerinde aşındırma işlemi uygulanmıştır. Bu uygulamada aşındırma elemanı olarak çapı 4 piksel olan disk şeklinde yapılandırma elemanı kullanılmıştır. Metal paraların birbirinden tam olarak ayrıldığı değer olan 4, deneme yanılma ile bulunmuştur. (Çapı 11 piksel olan elemanla erosion işlemi uygulandığında paraların tamamı silinmektedir.) Girişteki görüntü ve elde edilen çıkış aşağıda verilmektedir. Burada giriş görüntüsünde birbirine yapışık halde bulunan daireler erosion ile birbirinden ayrılmış ve oluşan görüntü üzerinde bir önceki soruda kullanılan algoritma ile toplam kaç tane daire olduğu bulunmuştur. Bağlı bileşenlerin bulunması ile elde edilen etiket sayısı 22 olarak belirlenmektedir.

giriş 4 piksellik disk ile aşındırılmış görüntü

1. Kenar bulma işlemi için giriş görüntüsünden aşındırılmış görüntü çıkartılarak sadece aşındırılmış çizgilerin yani kenarların kaldığı görüntü elde edilebilir. Bu uygulamada ise aşındırma elemanı olarak çapı 1 piksel olan disk şeklinde yapılandırma elemanı kullanılmıştır. Girişteki görüntü ve elde edilen çıkış aşağıda verilmektedir. Bu yöntem kullanılarak bulunan kenarlar aşağıdaki şekilde verilmiştir.



giriş aşındırılmış görüntü giriş-aşındırılmış görüntü

**Program kodları:**

**1-**

clear all;

sigma=5;%sigma 1,5,10 için filtre büyüdükçe iç kenarları gösteremiyor

x=-3\*sigma:3\*sigma;

y=-3\*sigma:3\*sigma;

[a, boyut]=size(x);

for i=1:boyut %2 boyutlu gauss filtreyi oluştur

for j=1:boyut

B(i,j)=(-1)\*((x(i)^2+y(j)^2-2\*(sigma^2))/sigma^4)\*exp(((-1)\*(x(i)^2+y(j)^2))/(2\*(sigma^2)));

end

end

toplam=sum(B(:));

B=B/toplam;%filtrenin içindeki katsayıların toplamı 1 olsun

f=imread('C:\Users\Melike Nur Mermer\Desktop\Odev1\regions.pgm');

g=imfilter(f,B);

[g,t]=edge(g,'zerocross');

imshow(f);

figure

imshow(g);

%mesh(B);

**2-**

clear all;

% Nx1, Nx2 => x ekseni yönünde uygulanacak filtrenin boyutları

% Sigma1, sigma2 azaldıkça daha ince kenarlar

Nx1=10; Sigmax1=0.5; Nx2=10; Sigmax2=0.5; Theta1=pi/2;

% Ny1, Ny2 => y ekseni yönünde uygulanacak filtrenin boyutları

Ny1=10; Sigmay1=0.5; Ny2=10; Sigmay2=0.5; Theta2=0;

alfa=0.1;%threshold parametresi

w=imread('C:\Users\Melike Nur Mermer\Desktop\Odev1\regions.pgm');

w=im2double(w);

% x ekseni yönünde kenarları bulma

filterx=d2dgauss(Nx1,Sigmax1,Nx2,Sigmax2,Theta1);

Ix=imfilter(w,filterx);

% y ekseni yönünde kenarları bulma

filtery=d2dgauss(Ny1,Sigmay1,Ny2,Sigmay2,Theta2);

Iy=imfilter(w,filtery);

% x ve y yönündeki türevleri birleştirme

NVI=sqrt(Ix.\*Ix+Iy.\*Iy);

% eşik seviyesi min ve max piksel arasında alfa değerine göre bulunur

I\_max=max(max(NVI));

I\_min=min(min(NVI));

level=alfa\*(I\_max-I\_min)+I\_min;%eşik seviyesi

Ibw=max(NVI,level.\*ones(size(NVI)));%görüntüde eşik değerinde olan pikselleri al

imshow(Ibw);

% 2 boyutlu gauss fonksiyonunun birinci türevinden kenar bulma filtresini oluşturan fonksiyon

function h = d2dgauss(n1,sigma1,n2,sigma2,theta)

r=[cos(theta) -sin(theta);

sin(theta) cos(theta)];

for i = 1 : n2

for j = 1 : n1

u = r \* [j-(n1+1)/2 i-(n2+1)/2]';

h(i,j) = gauss(u(1),sigma1)\*dgauss(u(2),sigma2);

end

end

h = h / sqrt(sum(sum(abs(h).\*abs(h))));

% 1-boyutlu gauss fonksiyonun birinci türevi

function y = dgauss(x,std)

y = -x \* gauss(x,std) / std^2;

% 1 boyutlu gauss

function y = gauss(x,std)

y = exp(-x^2/(2\*std^2)) / (std\*sqrt(2\*pi));

**3-**

clear all;

grnt='labirent';

bw=imread('C:\Users\Melike Nur Mermer\Desktop\Odev1\labirent.pgm');

bw=im2bw(bw);

bw=imcomplement(bw);

imshow(bw);

[m,n]=size(bw);

bwnnew = [ zeros(1,n) ; bw; zeros(1,n)];%zero-padding

bwnnew = [ zeros(m+2,1) , bwnnew , zeros(m+2,1)];

L=zeros(size(bwnnew));

startLabel=1;

for ir=2:m+1

for ic=2:n+1

curdata=bwnnew(ir,ic);

lc = L(ir,ic);

if((curdata==1)&&(lc==0))

L(ir,ic)=startLabel;

L = findConnectedLabels(L,startLabel,bwnnew,ir,ic,m,n);

startLabel=startLabel+1;

end

end

end

startLabel=startLabel-1;

L = L(2:m+1,2:n+1);

% L=uint8(L);

imshow(L);

filename=strcat(grnt,'\_4yonlu.pgm');

imwrite(L,filename);

%4-yönlü bağlı bileşen bulma fonksiyonu

function [L] = findConnectedLabels(L,startLabel,bwcur,ir,ic,m,n)

a = bwcur(ir+1, ic); % sağdaki piksel

b = bwcur(ir-1, ic); % sol

c = bwcur(ir, ic+1); % alt

d = bwcur(ir, ic-1); % üst

%gezilmiş matristeki pikseller

aa = L(ir+1, ic);

bb = L(ir-1, ic);

cc = L(ir, ic+1);

dd = L(ir, ic-1);

if((a==1)&&(aa==0))

L(ir+1, ic)=startLabel;

[L] = findConnectedLabels(L,startLabel,bwcur,ir+1,ic,m,n);

end

if((b==1)&&(bb==0))

L(ir-1, ic)=startLabel;

[L] = findConnectedLabels(L,startLabel,bwcur,ir-1,ic,m,n);

end

if((c==1)&&(cc==0))

L(ir, ic+1)=startLabel;

[L] = findConnectedLabels(L,startLabel,bwcur,ir,ic+1,m,n);

end

if((d==1)&&(dd==0))

L(ir, ic-1)=startLabel;

[L] = findConnectedLabels(L,startLabel,bwcur,ir,ic-1,m,n);

end

end

%8-yönlü bağlı bileşen bulma fonksiyonu

function [L] = findConnectedLabels8yonlu(L,startLabel,bwcur,ir,ic,m,n)

a = bwcur(ir+1, ic); % sağ

b = bwcur(ir-1, ic); % sol

c = bwcur(ir, ic+1); % alt

d = bwcur(ir, ic-1); % üst

e = bwcur(ir+1,ic+1); % sağ-üst

f = bwcur(ir-1,ic+1); % sol-üst

g = bwcur(ir+1,ic-1); % sağ-alt

h = bwcur(ir-1,ic-1); % sol-alt

%gezilmiş matristeki pikseller

aa = L(ir+1, ic);

bb = L(ir-1, ic);

cc = L(ir, ic+1);

dd = L(ir, ic-1);

ee = L(ir+1,ic+1);

ff = L(ir-1,ic+1);

gg = L(ir+1,ic-1);

hh = L(ir-1,ic-1);

if((a==1)&&(aa==0))

L(ir+1, ic)=startLabel;

[L] = findConnectedLabels8yonlu(L,startLabel,bwcur,ir+1,ic,m,n);

end

if((b==1)&&(bb==0))

L(ir-1, ic)=startLabel;

[L] = findConnectedLabels8yonlu(L,startLabel,bwcur,ir-1,ic,m,n);

end

if((c==1)&&(cc==0))

L(ir, ic+1)=startLabel;

[L] = findConnectedLabels8yonlu(L,startLabel,bwcur,ir,ic+1,m,n);

end

if((d==1)&&(dd==0))

L(ir, ic-1)=startLabel;

[L] = findConnectedLabels8yonlu(L,startLabel,bwcur,ir,ic-1,m,n);

end

if((e==1)&&(ee==0))

L(ir+1, ic+1)=startLabel;

[L] = findConnectedLabels8yonlu(L,startLabel,bwcur,ir+1,ic+1,m,n);

end

if((f==1)&&(ff==0))

L(ir-1, ic+1)=startLabel;

[L] = findConnectedLabels8yonlu(L,startLabel,bwcur,ir-1,ic+1,m,n);

end

if((g==1)&&(gg==0))

L(ir+1, ic-1)=startLabel;

[L] = findConnectedLabels8yonlu(L,startLabel,bwcur,ir+1,ic-1,m,n);

end

if((h==1)&&(hh==0))

L(ir-1, ic-1)=startLabel;

[L] = findConnectedLabels8yonlu(L,startLabel,bwcur,ir-1,ic-1,m,n);

end

end

**4-**

clear all;

f=imread('C:\Users\Melike Nur Mermer\Desktop\Odev1\coins.pgm');

imshow(f);

f=im2bw(f);

se = strel('disk',1);%11 olunca tüm coinler aşınıyor

f1=imerode(f,se);

imshow(f);

figure

% f1=uint8(f1);

imshow(f1);

[m,n]=size(f1);

bwnnew = [ zeros(1,n) ; f1; zeros(1,n)];%zero-padding

bwnnew = [ zeros(m+2,1) , bwnnew , zeros(m+2,1)];

L=zeros(size(bwnnew));

startLabel=1;

for ir=2:m+1

for ic=2:n+1

curdata=bwnnew(ir,ic);

lc = L(ir,ic);

if((curdata==1)&&(lc==0))

L(ir,ic)=startLabel;

L = findConnectedLabels(L,startLabel,bwnnew,ir,ic,m,n);

startLabel=startLabel+1;

end

end

end

startLabel=startLabel-1;

L = L(2:m+1,2:n+1);

f2=f-f1;%kenar bulma için ilk görüntüden aşındırılmış görüntü çıkartılır

figure

imshow(f2);