Edge detection using the Sobel operators

The Sobel operators

In[13]:=
```
IA = ImageAdjust;
```

Defining the horizontal and vertical Sobel operators

In[14]:=
```
sobelh = KroneckerProduct[{1, 2, 1}, {-1, 0, 1}]; sobelh // MatrixForm
```

Out[14]//MatrixForm=

$$\begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix}$$

In[15]:=
```
sobelv = KroneckerProduct[{-1, 0, 1}, {1, 2, 1}]; sobelv // MatrixForm
```

Out[15]//MatrixForm=

$$\begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{pmatrix}$$

Applying the Sobel operators separately

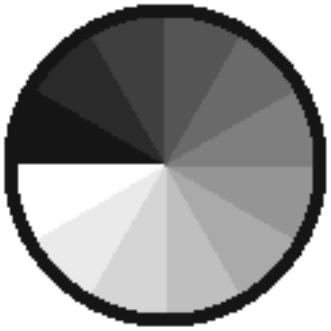
In[16]:=
```
sobel[A_]:=
Module[{h,v},
AA=ArrayPad[A,{1,1},"Extrapolated"];
h=ListConvolve[sobelh,AA];
v=ListConvolve[sobelv,AA];
{h,v}
]
```

A test image

In[17]:=
```
circle7 = Import["~/LEHRE/Wavelets-All/WTBV-15/CWT-Edges/circle7.m"];
```

In[18]:=
```
img = Image[circle7, ImageSize → Small]
```

Out[18]=

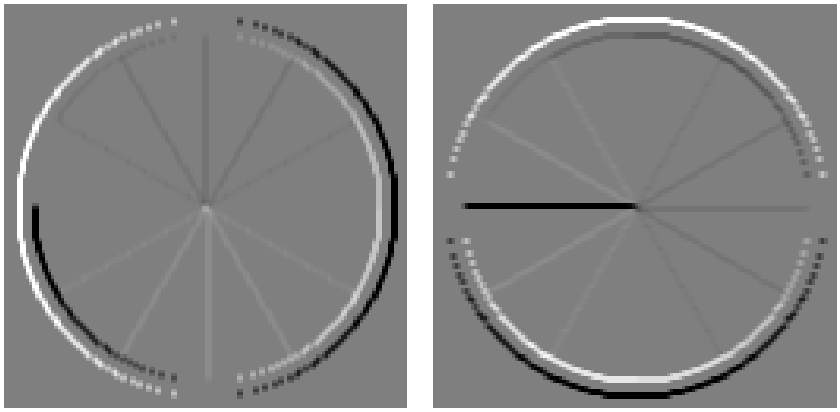


In[19]:=
```
{Dx, Dy} = sobel[circle7];
```

Applying the horizontal and the vertical Sobel filters separately

In[20]:=
```
GraphicsRow[{IA[Image[Dx]], IA[Image[Dy]]}, ImageSize → Scaled[0.7]]
```

Out[20]=



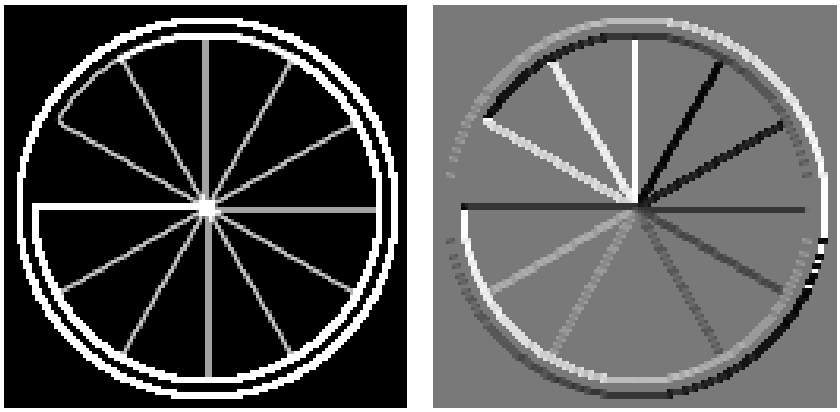
The size and direction of the approximate gradients obtained from the previous data

In[21]:=
```
W = Dx + I * Dy;
```

In[22]:=
```
{R, S} = {Abs[W], Arg[W]};
```

In[23]:=
```
GraphicsRow[{IA[Image[R], {1, 0.7}], IA[Image[S]]}, ImageSize → Scaled[0.7]]
```

Out[23]=

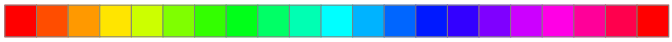


Encoding the gradient

Hue colors

In[24]:=
```
Graphics[Table[{Hue[s], EdgeForm[Gray], Rectangle[{20 s, 0}]}, {s, 0, 1, 1/20}]]
```

Out[24]=



In[25]:=
```
Graphics[Table[{Hue[s], EdgeForm[Gray], Rectangle[{10 s, 0}]}, {s, 0, 2, 1/10}]]
```

Out[25]=

Visualizing the gradient

In[26]:=
```
grd[f_, x_, y_] := Sqrt[Total[ (∇{x,y} f) ^ 2]]
```

In[27]:=
```
f[x_, y_] = Sin[x ^ 2 * y]
```

Out[27]=
$$\text{Sin}\left[x^2\, y\right]$$

In[28]:=
```
g[x_, y_] = grd[f[x, y], x, y]
```

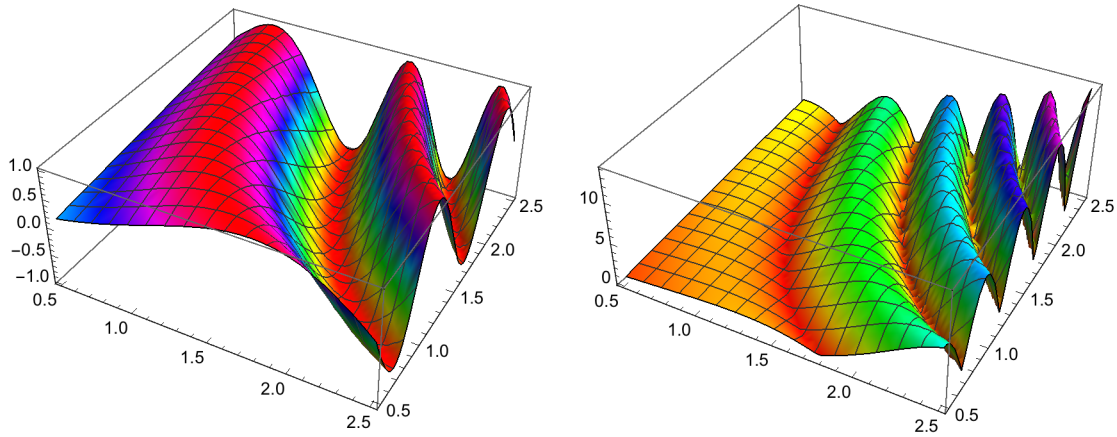
Out[28]=
$$\sqrt{\left(x^4 \,\text{Cos}\left[x^2\, y\right]^2 + 4\, x^2\, y^2 \,\text{Cos}\left[x^2\, y\right]^2\right)}$$

In[29]:=
```
pf = Plot3D[f[x, y], {x, 0.5, 2.5}, {y, 0.5, 2.5},
    ColorFunction -> Function[{x, y, z}, Hue[z]], ColorFunctionScaling → True];
```

In[30]:=
```
pg = Plot3D[g[x, y], {x, 0.5, 2.5}, {y, 0.5, 2.5},
    ColorFunction -> Function[{x, y, z}, Hue[z]], ColorFunctionScaling → True];
```

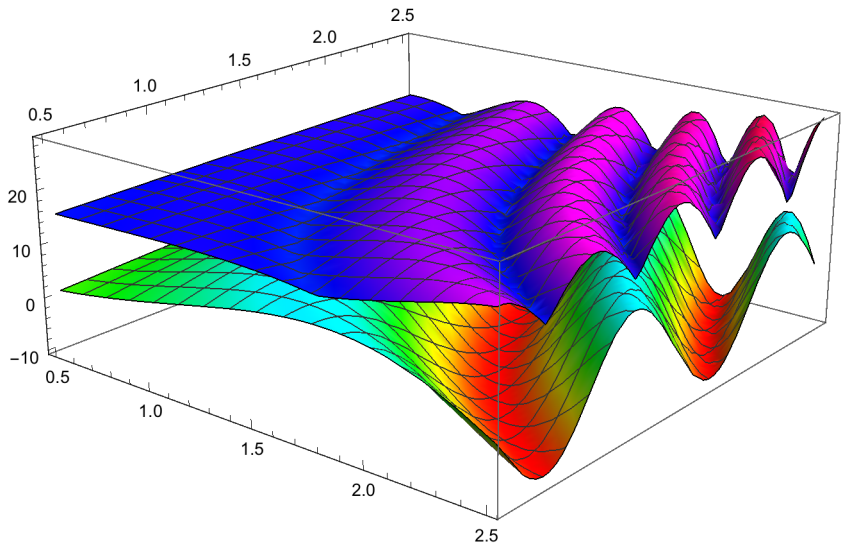
In[31]:=
```
GraphicsRow[{pf, pg}, ImageSize → Large]
```

Out[31]=



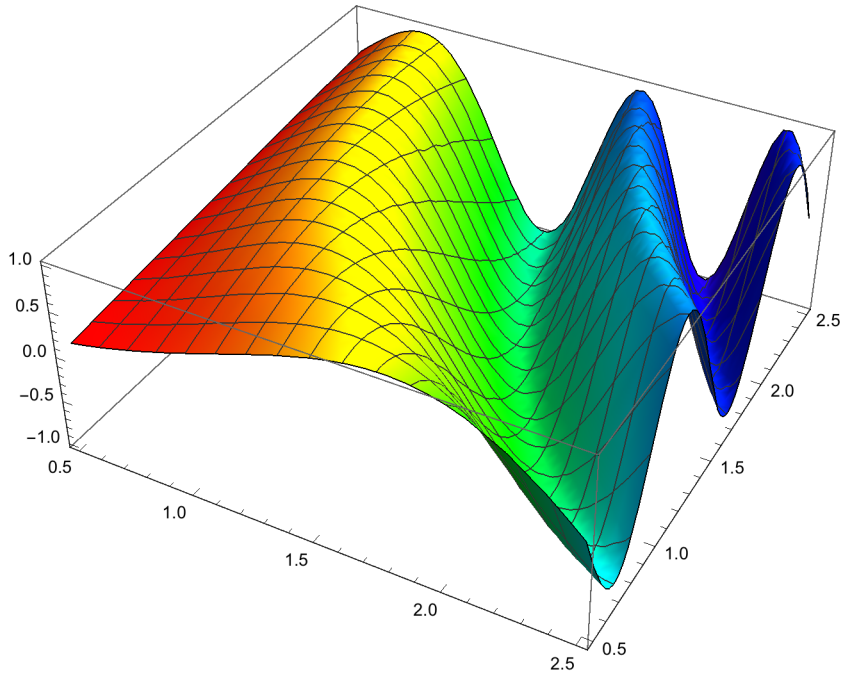
In[32]:=
```
Plot3D[{10 f[x, y], g[x, y] + 15}, {x, 0.5, 2.5}, {y, 0.5, 2.5},
 ColorFunction -> Function[{x, y, z}, Hue[z]],
 ColorFunctionScaling → True, ImageSize → Scaled[0.7]]
```

Out[32]=



In[33]:=
```
Plot3D[f[x, y], {x, 0.5, 2.5}, {y, 0.5, 2.5},
 ColorFunction -> Function[{x, y, z}, Hue[g[x, y] / 2]],
 ColorFunctionScaling → True, ImageSize → Scaled[0.7]]
```

Out[33]=



Discretizing directions

In[34]:=
```
ddir2vec[a_]:=Module[{aa},
aa=Mod[a,2 Pi,-Pi];
Which[
Abs[aa]≤Pi/8,{1,0},
Pi/8<aa≤3 Pi/8,{1,1},
3 Pi/8<aa≤5 Pi/8,{0,1},
5 Pi/8<aa≤7 Pi/8,{-1,1},
-7 Pi/8<aa<-5 Pi/8,{-1,-1},
-5 Pi/8<aa≤-3 Pi/8,{0,-1},
-3 Pi/8 <aa ≤-Pi/8,{1,-1},
Abs[aa]≥ 7 Pi/8,{-1,0},
True,{0,0}]
]
```

Direction vectors for angles which are multiples of 36 degrees

In[35]:=
```
Transpose[Table[{k * Pi / 5, ddir2vec[k * Pi / 5] // MatrixForm}, {k, 0, 9}]] //
 MatrixForm
```

Out[35]//MatrixForm=

$$\begin{pmatrix} 0 & \frac{\pi}{5} & \frac{2\pi}{5} & \frac{3\pi}{5} & \frac{4\pi}{5} & \pi & \frac{6\pi}{5} & \frac{7\pi}{5} & \frac{8\pi}{5} & \frac{9\pi}{5} \\ \begin{pmatrix} 1 \\ 0 \end{pmatrix} & \begin{pmatrix} 1 \\ 1 \end{pmatrix} & \begin{pmatrix} 0 \\ 1 \end{pmatrix} & \begin{pmatrix} 0 \\ 1 \end{pmatrix} & \begin{pmatrix} -1 \\ 1 \end{pmatrix} & \begin{pmatrix} -1 \\ 0 \end{pmatrix} & \begin{pmatrix} -1 \\ -1 \end{pmatrix} & \begin{pmatrix} 0 \\ -1 \end{pmatrix} & \begin{pmatrix} 0 \\ -1 \end{pmatrix} & \begin{pmatrix} 1 \\ -1 \end{pmatrix} \end{pmatrix}$$

In[36]:=
```
dirlist = {{1, 0}, {1, 1}, {0, 1}, {-1, 1}, {-1, 0}, {-1, -1}, {0, -1}, {1, -1}}
```

Out[36]= {{1, 0}, {1, 1}, {0, 1}, {-1, 1}, {-1, 0}, {-1, -1}, {0, -1}, {1, -1}}

In[37]:=
```
color =
 <|Join[Table[dirlist[[i]] → Hue[(i - 1) / 8], {i, 1, 8}], {{0, 0} → White}]|>
```

Out[37]= ⟨| {1, 0} → ■, {1, 1} → ■, {0, 1} → ■, {-1, 1} → ■,
{-1, 0} → ■, {-1, -1} → ■, {0, -1} → ■, {1, -1} → ■, {0, 0} → □ |⟩

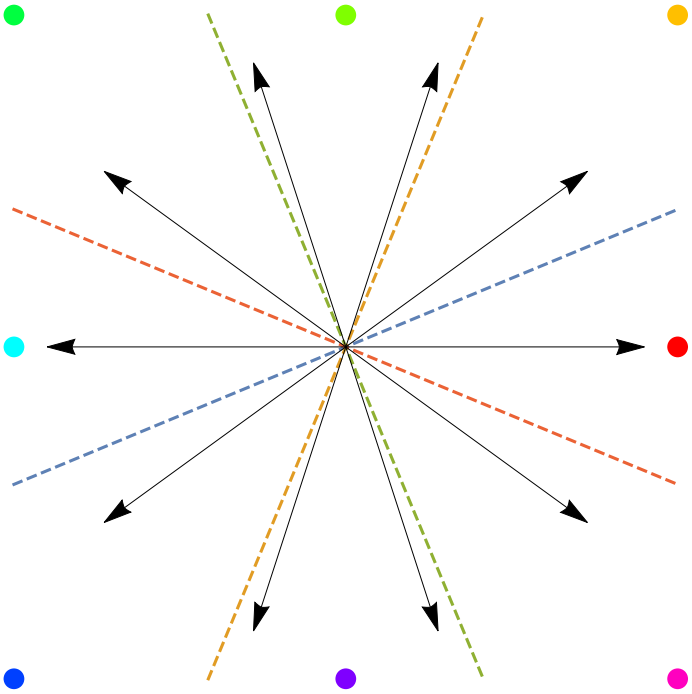
In[38]:=
```
p0 = Graphics[
    Map[{PointSize[0.03], color[#], Point[#]}
      &, dirlist] ];
```

In[39]:=
```
p1 = Plot[
    {x * Tan[Pi / 8], x * Tan[3 Pi / 8], x * Tan[5 Pi / 8], x * Tan[7 Pi / 8]}, {x, -1, 1},
    AspectRatio → 1, PlotRange → {-1, 1}, PlotStyle -> Dashed];
```

In[40]:=
```
p2 =
  Graphics[Table[Arrow[{{0, 0}, 0.9 * {Cos[k * Pi / 5], Sin[k * Pi / 5]}}], {k, 0, 9}]];
```

In[41]:=
```
Show[{p0, p1, p2}, Background → White]
```

Out[41]=



Computing absolute values and discretized directions of the gradients of an array (discretized image) using the Sobel operators

In[42]:=

```
ggrad[A_]:=
Module[{AA,val,dir,m,n,h,v,hh,vv,dv},
{m,n}=Dimensions[A];
AA=ArrayPad[A,{1,1},"Extrapolated"];
h=ListConvolve[sobelh,AA];
v=ListConvolve[sobelv,AA];
val=Table[0,{x,1,m},{y,1,n}];
dir=Table[0,{x,1,m},{y,1,n}];
Do[
{hh,vv}={h[[x,y]],v[[x,y]]};
val[[x,y]]=N[Sqrt[hh^2+vv^2]];
dir[[x,y]]=
If[
hh==0&&vv==0,
{0,0},
dv=-ddir2vec[N[ArcTan[hh,vv]]];
{dv[[1]],dv[[2]]}],
{x,1,m},{y,1,n}];
{val,dir}
]
```

Plotting the gradients of an array as a vector field

In[43]:=

```
ggradplot[A_]:=
Module[{AA,val,dir,p1,p2,p3,q,m,n,h,v,hh,vv,dv},
{m,n}=Dimensions[A];
AA=ArrayPad[A,{1,1},"Extrapolated"];
h=ListConvolve[sobelh,AA];
v=ListConvolve[sobelv,AA];
p1=ListVectorPlot[
Table[Table[{h[[x,y]],v[[x,y]]},{x,1,m}],{y,1,n}],VectorPoints→All];
val=Table[Table[0,{y,1,n}],{x,1,m}];
dir=Table[Table[0,{y,1,n}],{x,1,m}];
Do[
{hh,vv}={h[[x,y]],v[[x,y]]};
val[[x,y]]=N[Sqrt[hh^2+vv^2]];
dir[[x,y]]=
If[
hh==0&&vv==0,
{0,0},
dv=ddir2vec[N[ArcTan[hh,vv]]];
-{ dv[[1]],dv[[2]]}],
{x,1,m},{y,1,n}];
p2=ArrayPlot[val];
q=3*val*dir;
p3 =ListVectorPlot[ Transpose[q],VectorPoints→All];
p3
]
```

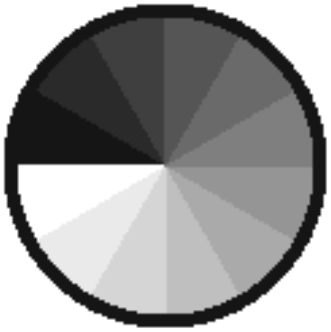
The gradient encoding

In[44]:=
```
color
```

Out[44]=
```
⟨|{1, 0} → ■, {1, 1} → ■, {0, 1} → ■, {-1, 1} → ■,
 {-1, 0} → ■, {-1, -1} → ■, {0, -1} → ■, {1, -1} → ■, {0, 0} → □|⟩
```

In[45]:=
```
img
```

Out[45]=

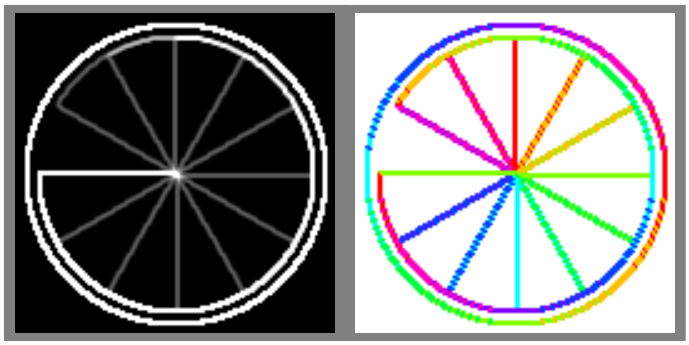


In[46]:=
```
imggrad = ggrad[circle7];
```

The gradient given as (value,direction)-pairs with discretized directions

In[47]:=
```
GraphicsRow[
 {Image[imggrad[[1]]], Image[(imggrad[[2]] /. color)]}, Background → Gray]
```

Out[47]=



Examples for gradients

First example

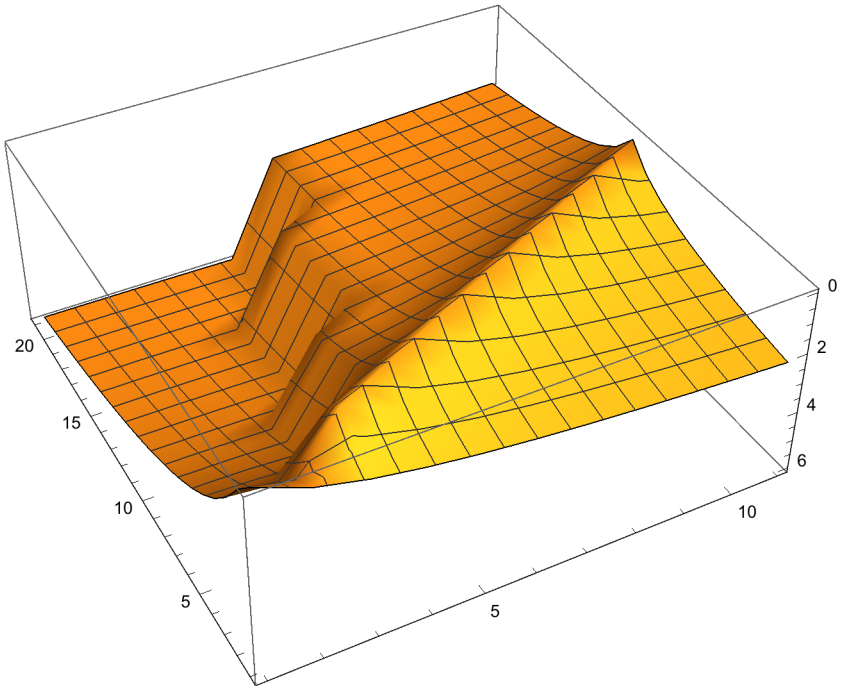
In[48]:=
```
A=Table[
Piecewise[
{{Log[1+Abs[x^2+y^2]],x^2<=y},
{Log[1+Abs[x-y]],x^2>y}}],
{x,0,10},{y,0,20}]; A//MatrixForm
```

Out[48]//MatrixForm=

0	Log[2]	Log[5]	Log[10]	Log[17]	Log[26]	Log[37]	Log[50]	Log[65]	Log[8
Log[2]	Log[3]	Log[6]	Log[11]	Log[18]	Log[27]	Log[38]	Log[51]	Log[66]	Log[8
Log[3]	Log[2]	0	Log[2]	Log[21]	Log[30]	Log[41]	Log[54]	Log[69]	Log[8
Log[4]	Log[3]	Log[2]	0	Log[2]	Log[3]	Log[4]	Log[5]	Log[6]	Log[9
Log[5]	Log[4]	Log[3]	Log[2]	0	Log[2]	Log[3]	Log[4]	Log[5]	Log[6
Log[6]	Log[5]	Log[4]	Log[3]	Log[2]	0	Log[2]	Log[3]	Log[4]	Log[5
Log[7]	Log[6]	Log[5]	Log[4]	Log[3]	Log[2]	0	Log[2]	Log[3]	Log[4
Log[8]	Log[7]	Log[6]	Log[5]	Log[4]	Log[3]	Log[2]	0	Log[2]	Log[3
Log[9]	Log[8]	Log[7]	Log[6]	Log[5]	Log[4]	Log[3]	Log[2]	0	Log[2
Log[10]	Log[9]	Log[8]	Log[7]	Log[6]	Log[5]	Log[4]	Log[3]	Log[2]	0
Log[11]	Log[10]	Log[9]	Log[8]	Log[7]	Log[6]	Log[5]	Log[4]	Log[3]	Log[2

In[49]:=
```
ListPlot3D[A, ImageSize → Scaled[0.7]]
```

Out[49]=



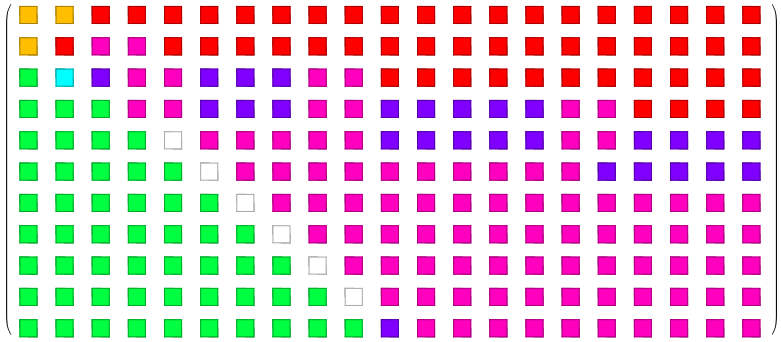
In[50]:=
```
ggrad[A][[1]] // MatrixForm
```

Out[50]//MatrixForm=

7.84207	7.26777	6.66637	4.97011	3.85393	3.12709	2.62509	2.25977	1.98272	1.76
4.91314	2.75581	6.40468	7.94479	5.55832	3.00179	2.54943	2.21088	1.94942	1.74
3.09985	1.83706	4.5995	10.823	11.6516	9.26919	9.3163	9.50643	8.24755	4.69
3.15805	4.23661	4.0876	2.75193	10.0723	11.5159	10.7892	10.6062	12.4537	12.5
2.4472	2.99851	4.23661	4.0876	0.	4.0876	4.23661	2.99851	5.97247	9.95
2.00332	2.34509	2.99851	4.23661	4.0876	0.	4.0876	4.23661	2.99851	2.34
1.69778	1.93195	2.34509	2.99851	4.23661	4.0876	0.	4.0876	4.23661	2.99
1.47398	1.64494	1.93195	2.34509	2.99851	4.23661	4.0876	0.	4.0876	4.23
1.30274	1.43324	1.64494	1.93195	2.34509	2.99851	4.23661	4.0876	0.	4.08
1.16738	1.27035	1.43324	1.64494	1.93195	2.34509	2.99851	4.23661	4.0876	0.
1.07831	1.16738	1.30274	1.47398	1.69778	2.00332	2.4472	3.15805	4.53088	4.91

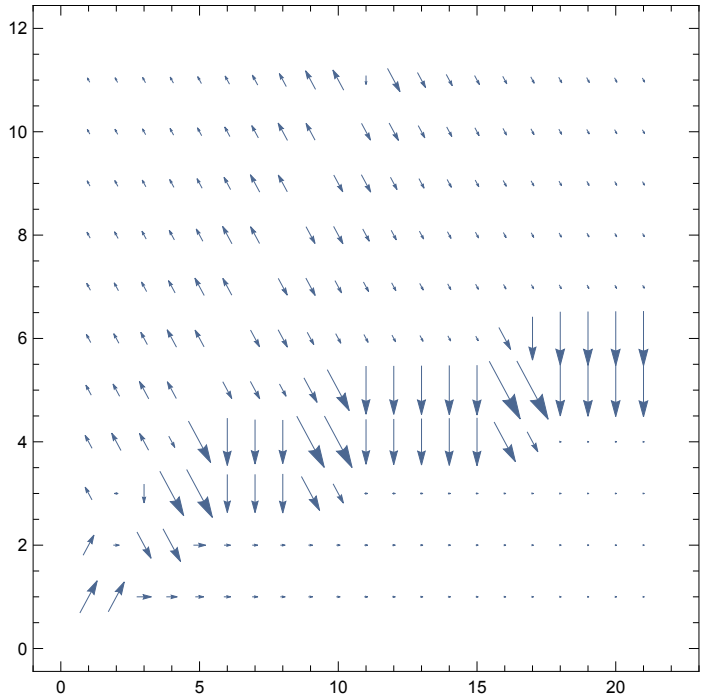
In[51]:=
```
ggrad[A][[2]] /. color // MatrixForm
```

Out[51]//MatrixForm=



In[52]:=
```
Show[ggradplot[A]]
```

Out[52]=

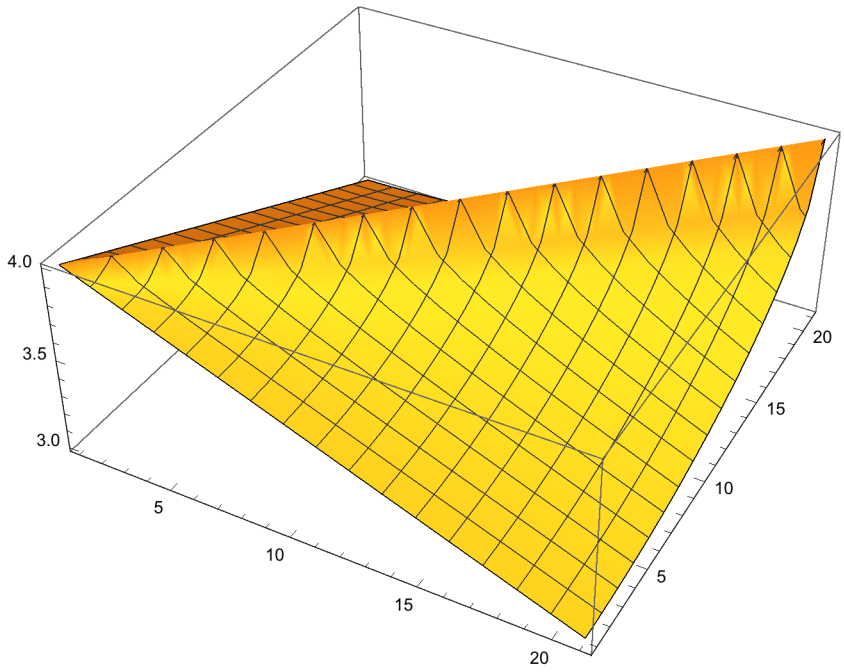


An example image with a sharp edge

In[53]:=
```
fvals1 = Table[
    4 - Sqrt[Abs[x^2 - y^2]], {x, 0, 1, 1/20}, {y, 0, 1, 1/20}];
```

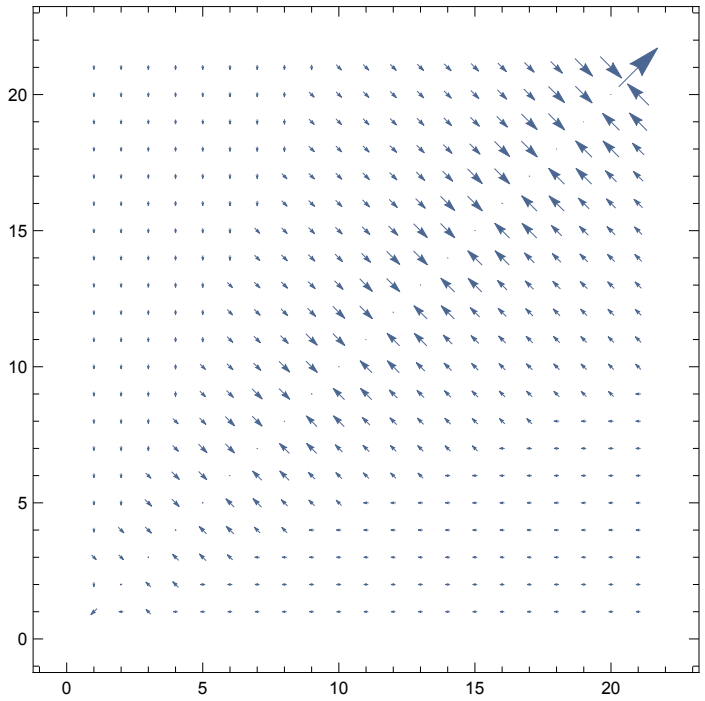
In[54]:=
```
ListPlot3D[fvals1, ImageSize → Scaled[0.7]]
```

Out[54]=



In[55]:=
```
ggradplot[fvals1]
```

Out[55]=

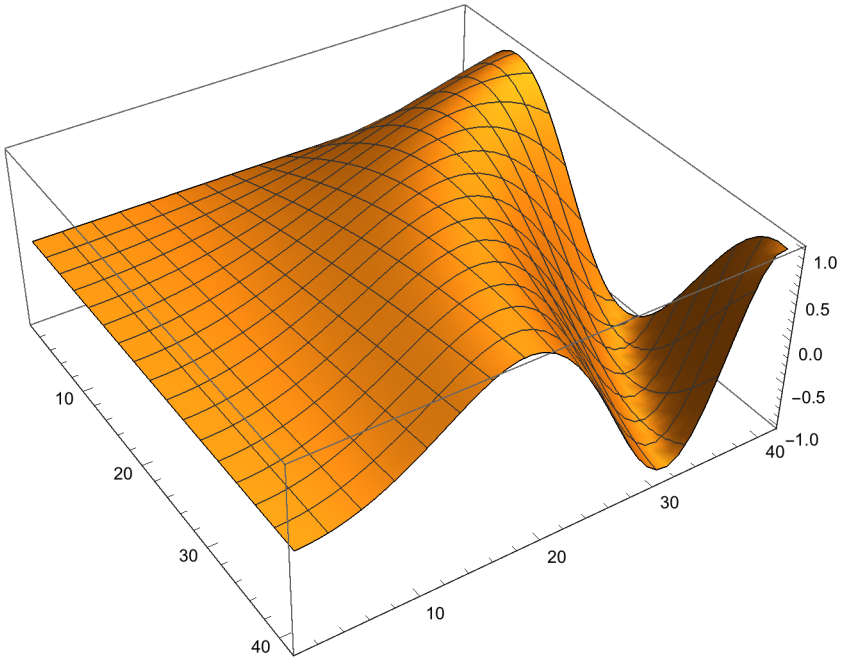


Another example image

In[56]:=
```
fvals2 = Table[
   Sin[x^2 * y], {x, 0, 2, 1 / 20}, {y, 0, 2, 1 / 20}];
```

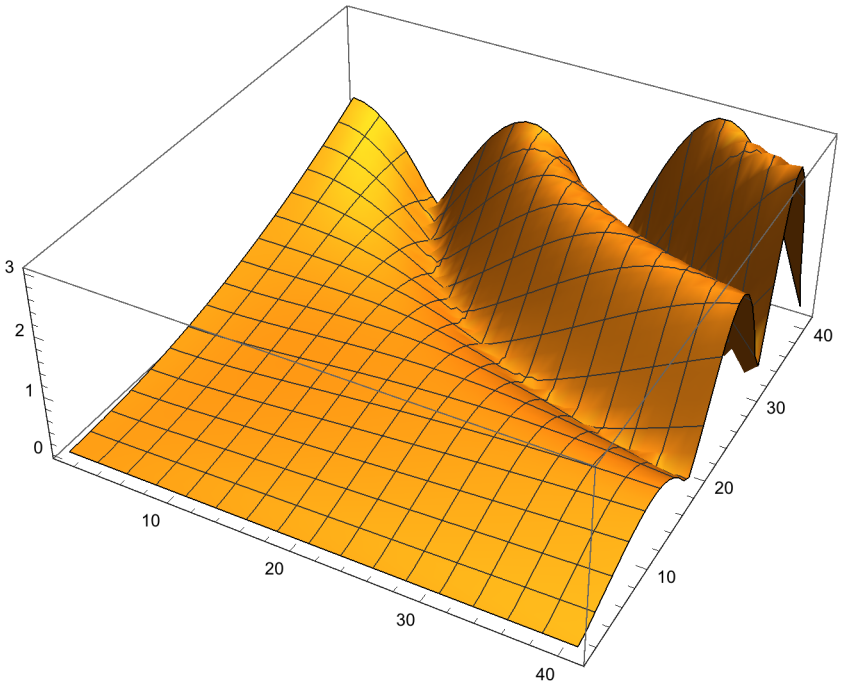
In[57]:=
```
ListPlot3D[fvals2, ImageSize → Scaled[0.7]]
```

Out[57]=



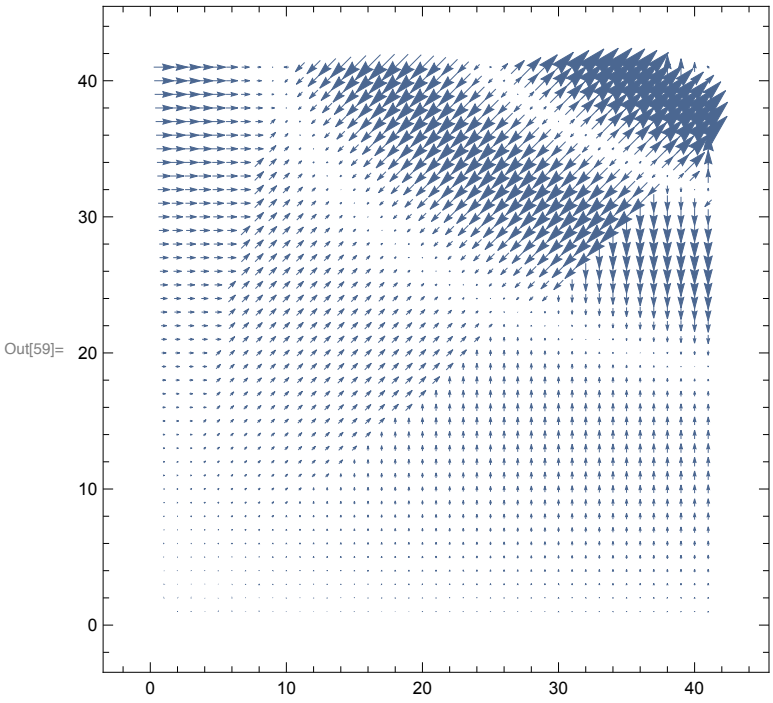
In[58]:=
```
ListPlot3D[ggrad[fvals2][[1]], ImageSize → Scaled[0.7]]
```

Out[58]=



In[59]:=
```
ggradplot[fvals2]
```

Out[59]=



Canny's algorithm

Identifying edge vertices of level λ in an array

In[60]:=
```
edges[A_,level_]:=
Module[{m,n,val,dir,max,VVal,p,x1,y1,x2,y2,EC,v,v1,v2,x,y},
{m,n}=Dimensions[A];
{val,dir}=ggrad[A];
max=Max[val];
VVal=ArrayPad[val,{1,1},"Extrapolated"];
EC=Table[0,{x,1,m},{y,1,n}];
Do[
{p={x+1,y+1};
{x1,y1}=p+dir[[x,y]];
{x2,y2}=p-dir[[x,y]];
v=VVal[[x,y]];
v1=VVal[[x1,y1]];
v2=VVal[[x2,y2]];
If[
v≥Max[v1,v2]&&v≥level max,
EC[[x,y]]=v,
EC[[x,y]]=0]},
{x,1,m},{y,1,n}];
EC
]
```

Canny's algorithm using two levels and a prescribed number of iterations

In[61]:=

```
canny[A_,low_,high_,iter_]:=
Module[{k,m,n,max,Alow,AAlow,Ahigh,AAhigh,Diff},
{m,n}=Dimensions[A];
max=Max[A];
Alow=Map[ If[#<low max,0,1]&,A,{2}];
AAlow=ArrayPad[Alow,{1,1},"Extrapolated"];
Ahigh =Map[ If[#<high max,0,1]&,A,{2}];
AAhigh=ArrayPad[Ahigh,{1,1},"Extrapolated"];
Print["strong edge vertices: ",Total[Flatten[Ahigh]]];
Print["weak edge vertices: ",Total[Flatten[Alow]]];
Print["further edge vertices: "];
For[k=1,k≤iter,k++,
Diff=AAlow-AAhigh;
Do[
If[
Diff[[x,y]]==1,
Diff[[x,y]]=
Max[Take[AAhigh,{x-1,x+1},{y-1,y+1}]]
],
{x,2,m+1},{y,2,n+1}
];
Print["round ",k," : ",Total[Flatten[Diff]]];
AAhigh=AAhigh+Diff
 ];
Take[AAhigh,{2,m+1},{2,n+1}]
]
```

In[62]:=

```
cannydemo[A_,low_,high_,iter_]:=
Module[{k,m,n,max,Alow,AAlow,Ahigh,AAhigh,Diff,out},
{m,n}=Dimensions[A];
max=Max[A];
Alow=Map[ If[#<low max,0,1]&,A,{2}];
AAlow=ArrayPad[Alow,{1,1} ];
Ahigh =Map[ If[#<high max,0,1]&,A,{2}];
AAhigh=ArrayPad[Ahigh,{1,1} ];
Print["strong edge vertices: ",Total[Flatten[Ahigh]]];
Print["weak edge vertices: ",Total[Flatten[Alow]]];
out={Ahigh};
Print["further edge vertices: "];
For[k=1,k≤iter,k++,
Diff=AAlow-AAhigh;
Do[
If[
Diff[[x,y]]==1,
Diff[[x,y]]=
Max[Take[AAhigh,{x-1,x+1},{y-1,y+1}]]
],
{x,2,m+1},{y,2,n+1}
];
Print["round ",k," : ",Total[Flatten[Diff]]];
AAhigh=AAhigh+Diff;
AppendTo[out,Take[AAhigh,{2,m+1},{2,n+1}]]
];
out
]
```

Examples for the Canny algorithm

First example

In[63]:=
```
img1 = Import["~/LEHRE/Wavelets-All/WTBV-10/CWT/camera.jpg"]
```

Out[63]=

In[64]:=
```
img1 = ColorConvert[img1, "Grayscale"]
```

Out[64]=

In[65]:=
```
ImageDimensions[img1]
```

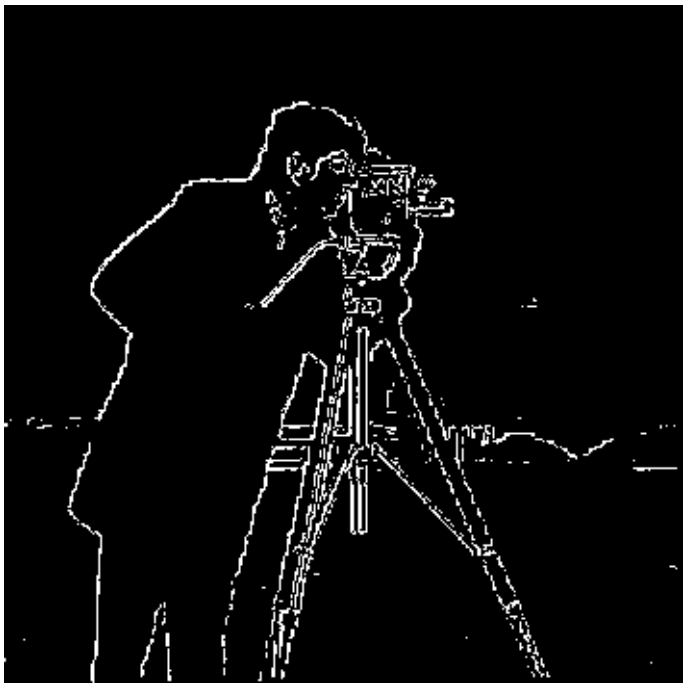
Out[65]=
```
{353, 352}
```

In[66]:=
```
data1 = ImageData[img1];
```

In[67]:=
```
edge1 = edges[data1, 0.15];
```

In[68]:=
```
Image[edge1]
```

Out[68]=



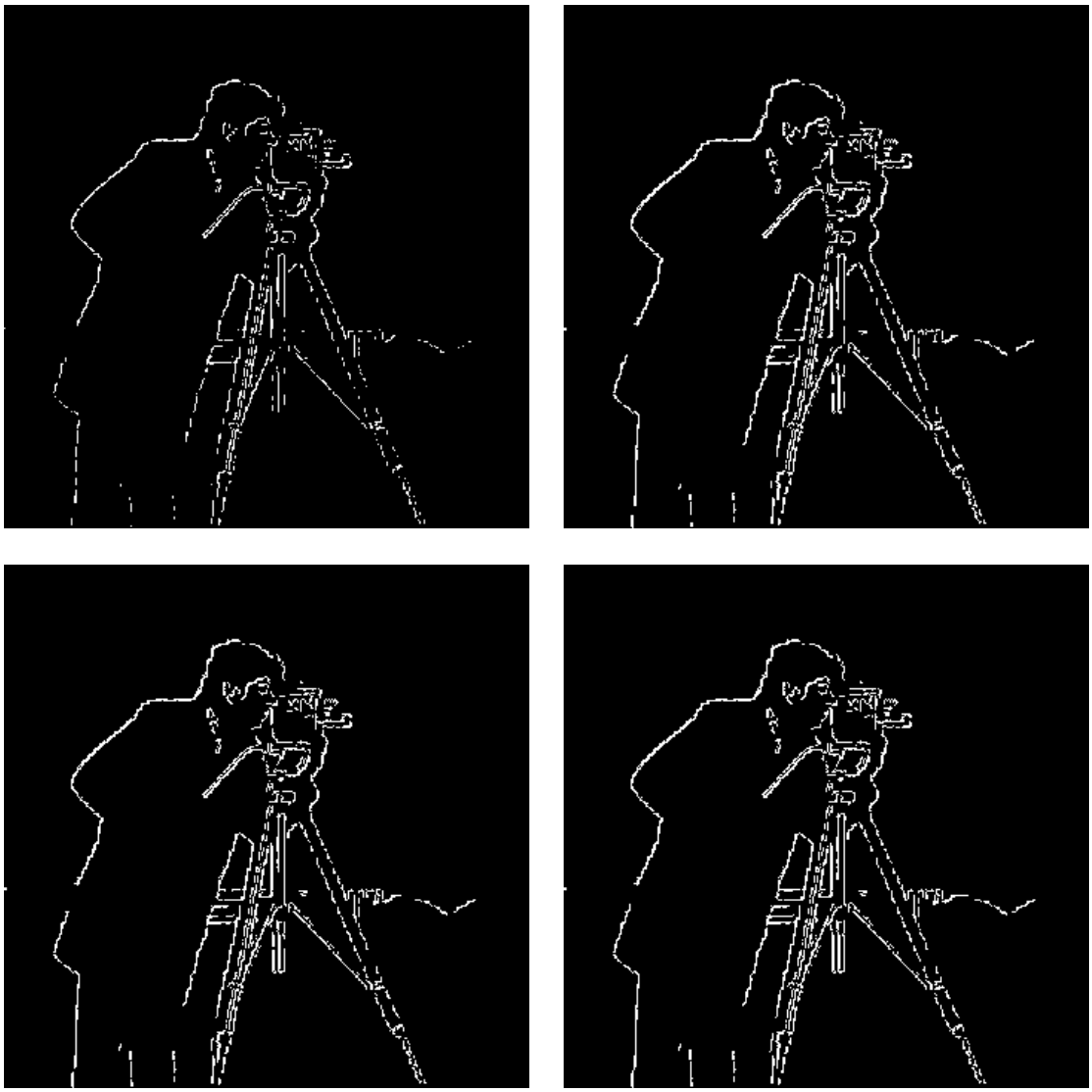
In[69]:=
```
cout1 = cannydemo[edge1, 0.2, 0.5, 10];
```

```
strong edge vertices: 2434
weak edge vertices: 4492
further edge vertices:
round 1 : 1101
round 2 : 229
round 3 : 112
round 4 : 62
round 5 : 39
round 6 : 18
round 7 : 10
round 8 : 6
round 9 : 5
round 10 : 5
```

In[70]:=
```
Animate[Image[cout1[[k]]], {k, 1, Length[cout1], 1}, AnimationRunning → False]
```

In[71]:=
```
GraphicsGrid[{{Image[cout1[[1]]], Image[cout1[[3]]]},
  {Image[cout1[[5]]], Image[cout1[[10]]]}}, ImageSize → Large]
```

Out[71]=



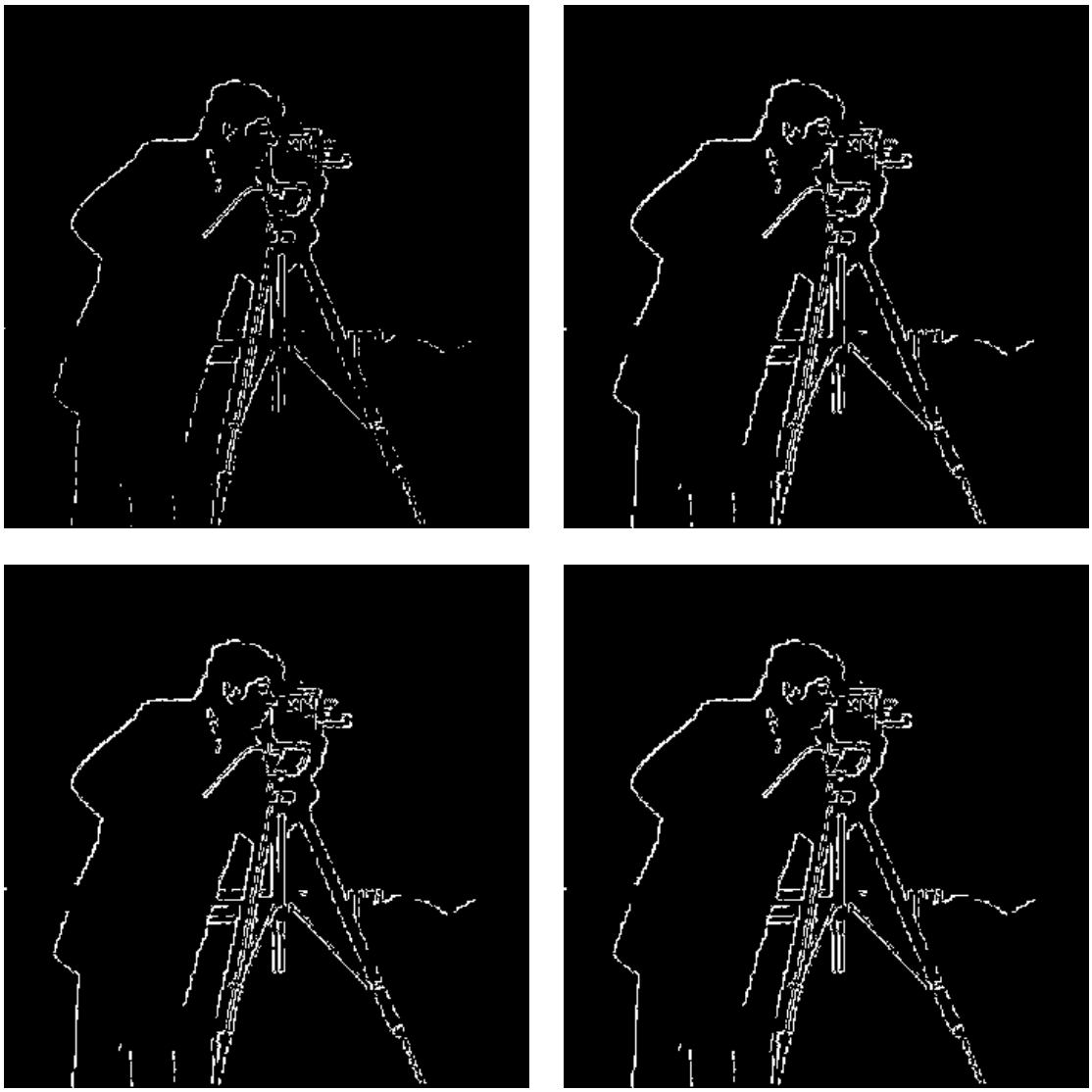
In[72]:=
```
cout2 = cannydemo[edge1, 0.3, 0.5, 10];
```

```
strong edge vertices: 2434
weak edge vertices: 4492
further edge vertices:
round 1 : 1101
round 2 : 229
round 3 : 112
round 4 : 62
round 5 : 39
round 6 : 18
round 7 : 10
round 8 : 6
round 9 : 5
round 10 : 5
```

In[73]:=
```
Animate[Image[cout2[[k]]], {k, 1, Length[cout2], 1}, AnimationRunning → False]
```

In[74]:=
```
GraphicsGrid[{{Image[cout2[[1]]], Image[cout2[[3]]]},
  {Image[cout2[[5]]], Image[cout2[[10]]]}}, ImageSize → Large]
```

Out[74]=



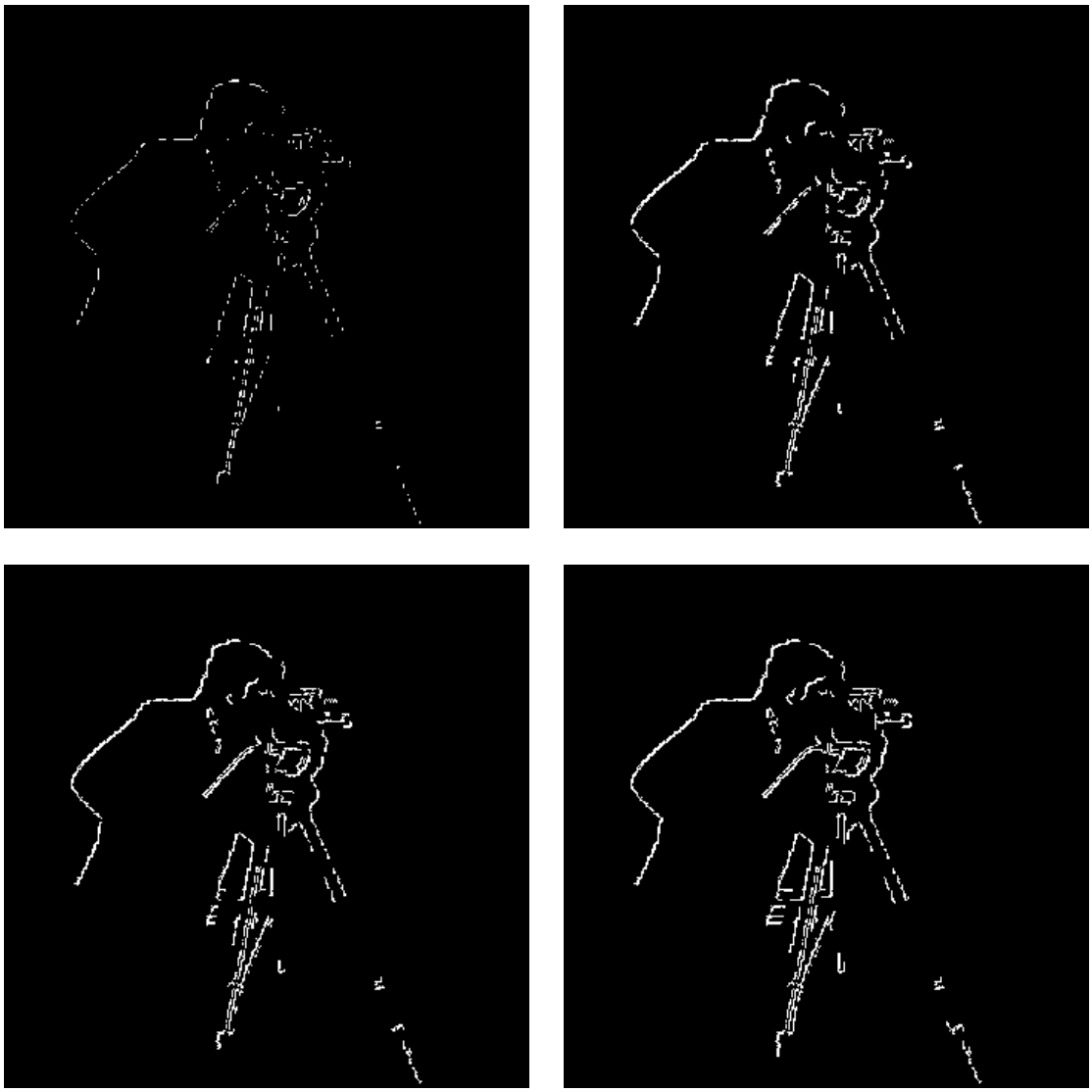
In[75]:=
```
cout3 = cannydemo[edge1, 0.15, 0.7, 10];
```

```
strong edge vertices: 684
weak edge vertices: 4492
further edge vertices:
round 1 : 823
round 2 : 367
round 3 : 193
round 4 : 122
round 5 : 97
round 6 : 60
round 7 : 48
round 8 : 41
round 9 : 41
round 10 : 33
```

In[76]:=
```
Animate[Image[cout3[[k]]], {k, 1, Length[cout3], 1}, AnimationRunning → False]
```

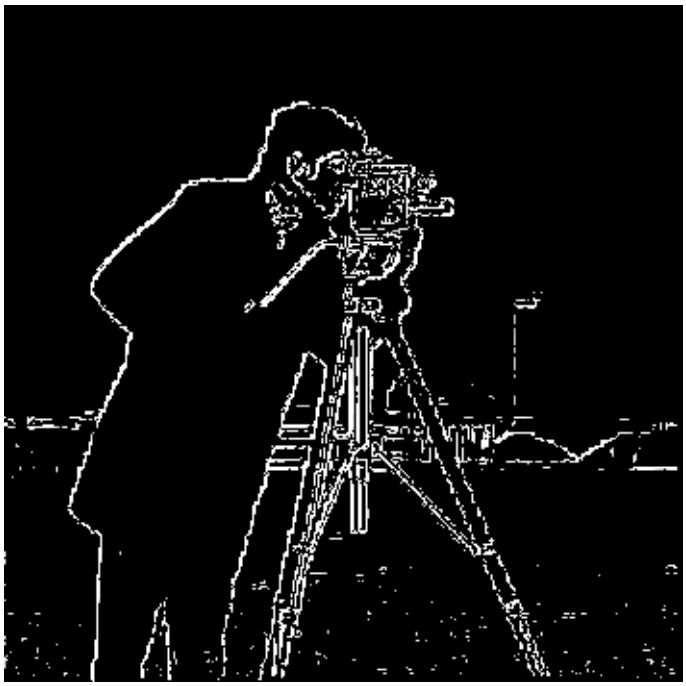
In[77]:=
```
GraphicsGrid[{{Image[cout3[[1]]], Image[cout3[[3]]]},
  {Image[cout3[[5]]], Image[cout3[[10]]]}}, ImageSize → Large]
```

Out[77]=



In[78]:=
```
edge1a = edges[data1, 0.1]; Image[edge1a]
```

Out[78]=



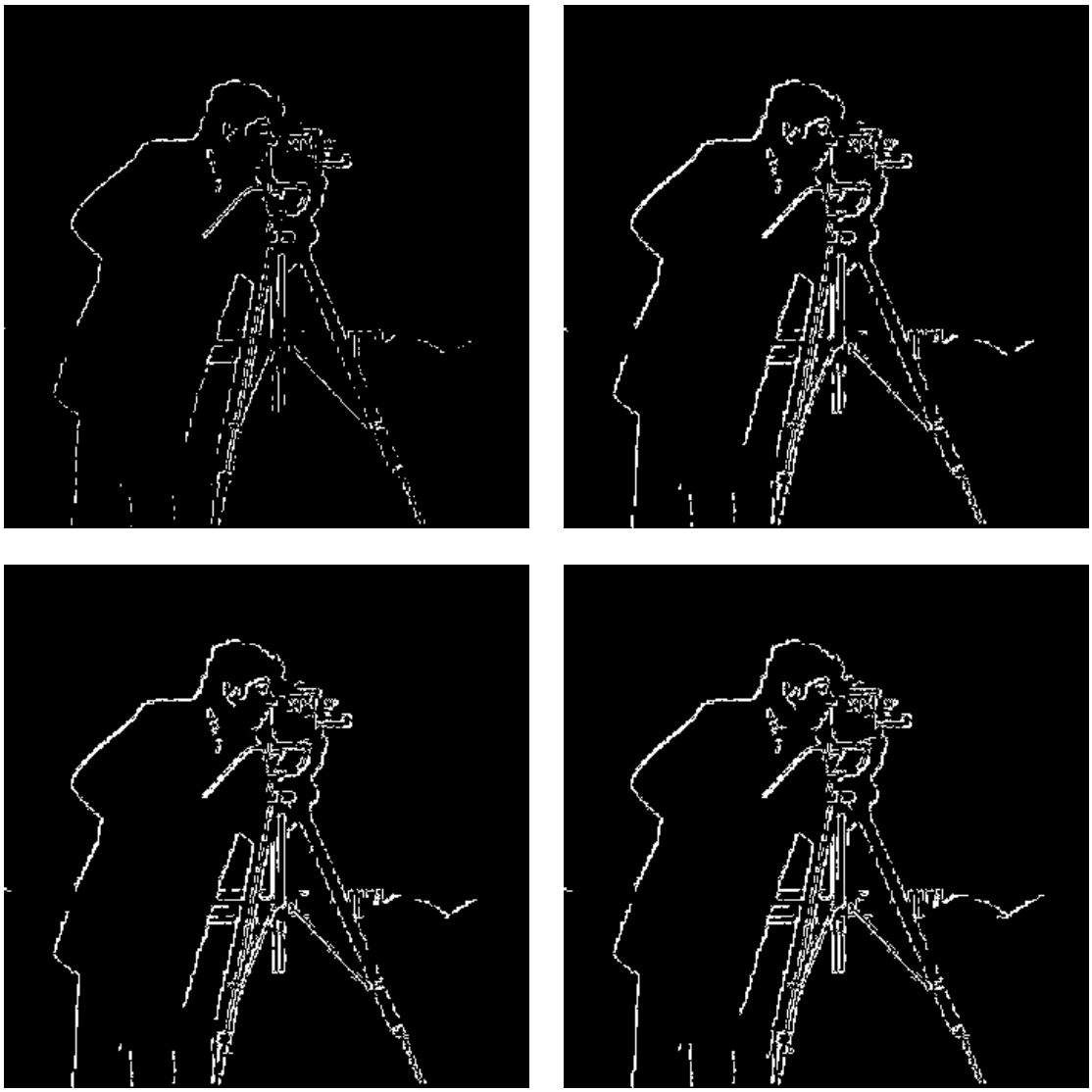
In[79]:=
```
cout4 = cannydemo[edge1a, 0.1, 0.5, 10];
```

```
strong edge vertices: 2434

weak edge vertices: 6236

further edge vertices:

round 1 : 1453

round 2 : 409

round 3 : 203

round 4 : 120

round 5 : 85

round 6 : 56

round 7 : 35

round 8 : 27

round 9 : 28

round 10 : 25
```

In[80]:=
```
Animate[Image[cout4[[k]]], {k, 1, Length[cout4], 1}, AnimationRunning → False]
```

In[81]:=
```
GraphicsGrid[{{Image[cout4[[1]]], Image[cout4[[3]]]},
  {Image[cout4[[5]]], Image[cout4[[10]]]}}, ImageSize → Large]
```

Out[81]=



In[82]:=
```
edge1b = edges[data1, 0.07]; Image[edge1b]
```

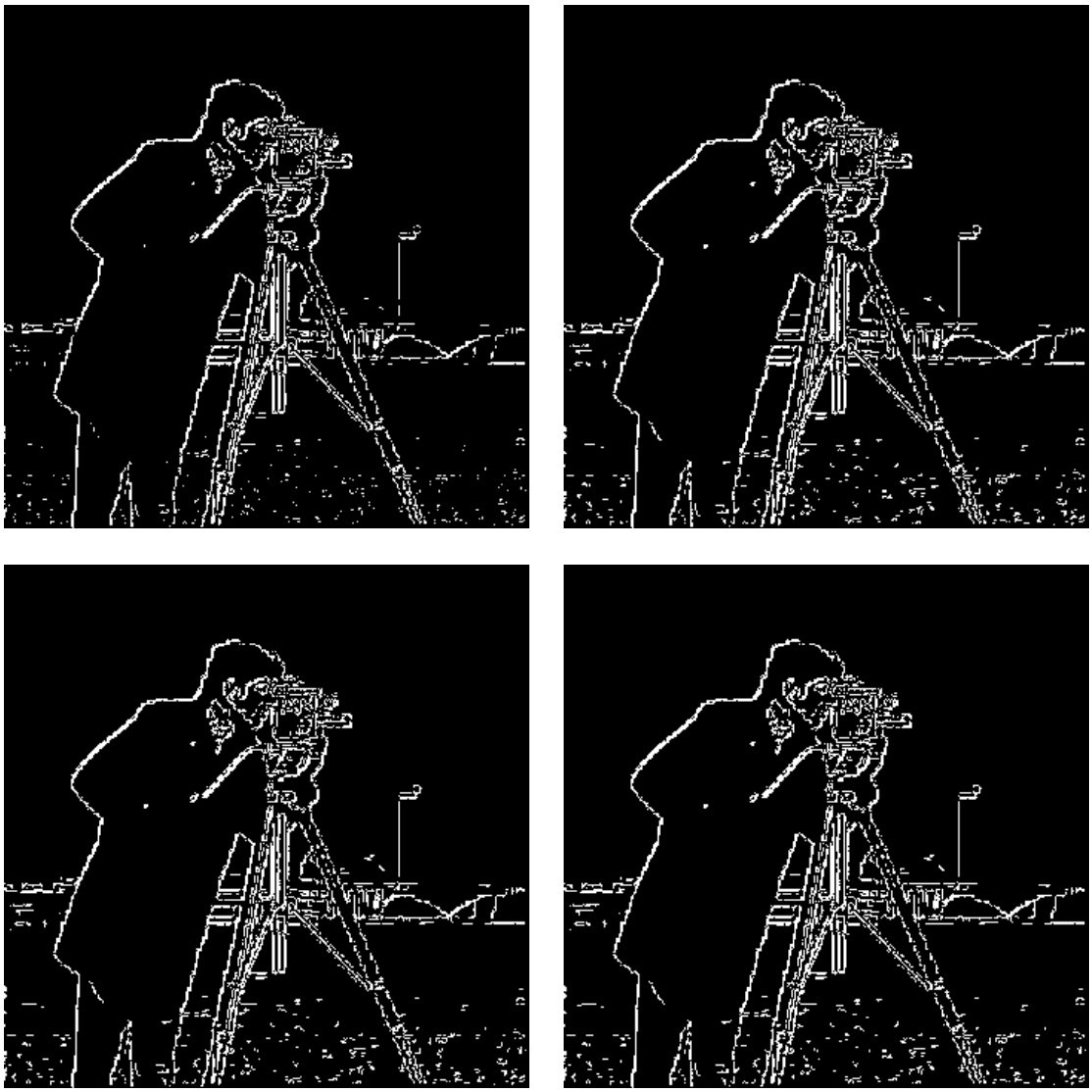
Out[82]=

In[83]:=
```
cout5 = cannydemo[edge1b, 0.07, 0.2, 10];
strong edge vertices: 6615
weak edge vertices: 8461
further edge vertices:
round 1 : 887
round 2 : 128
round 3 : 53
round 4 : 19
round 5 : 9
round 6 : 5
round 7 : 6
round 8 : 2
round 9 : 2
round 10 : 1
```

In[84]:=
```
Animate[Image[cout5[[k]]], {k, 1, Length[cout5], 1}, AnimationRunning → False]
```

In[85]:=
```
GraphicsGrid[{{Image[cout5[[1]]], Image[cout5[[3]]]},
  {Image[cout5[[5]]], Image[cout5[[10]]]}}, ImageSize → Large]
```

Out[85]=



In[86]:=

Second example

In[87]:=
```
img2 = Import["~/Lehre/Wavelets-All/WTBV-10/CWT/maira.jpg"]
```

Out[87]=

In[88]:=
```
img2 = ColorConvert[img2, "Grayscale"]
```

Out[88]=

In[89]:=
```
data2 = ImageData[img2][[301 ;; 700, 301 ;; 700]];
```

In[90]:=
```
Dimensions[data2]
```

Out[90]=
```
{400, 400}
```

In[91]:=
```
Image[data2]
```

Out[91]=

In[92]:=
```
edge2 = edges[data2, 0.25];
```

In[93]:=
```
cout6 = cannydemo[edge2, 0.25, 0.4, 20];
```

```
strong edge vertices: 5444
weak edge vertices: 13217
further edge vertices:
round 1 : 2801
round 2 : 232
round 3 : 126
round 4 : 80
round 5 : 47
round 6 : 42
round 7 : 31
round 8 : 25
round 9 : 25
round 10 : 21
round 11 : 23
round 12 : 15
round 13 : 16
round 14 : 14
round 15 : 14
round 16 : 15
round 17 : 16
round 18 : 13
round 19 : 13
round 20 : 13
```

In[94]:=
```
Animate[Image[cout6[[k]]], {k, 1, Length[cout6], 1}, AnimationRunning → False]
```

In[95]:=
```
GraphicsGrid[{{Image[cout6[[1]]], Image[cout6[[5]]]},
  {Image[cout6[[10]]], Image[cout6[[20]]]}}, ImageSize → Large]
```

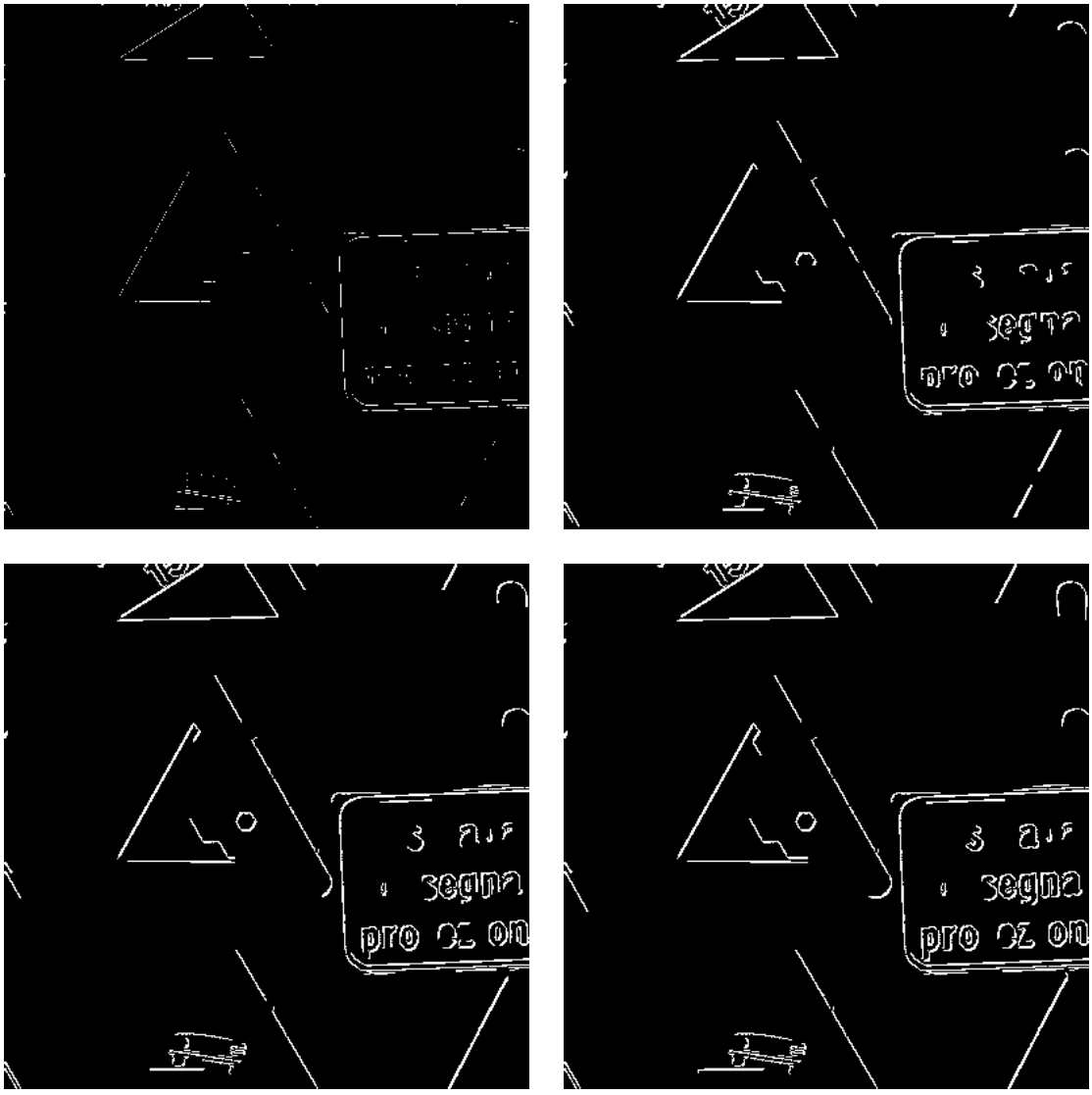
Out[95]=

In[96]:=
```
cout7 = cannydemo[edge2, 0.2, 0.5, 30];
strong edge vertices: 880
weak edge vertices: 15576
further edge vertices:
round 1 : 1132
round 2 : 421
round 3 : 311
round 4 : 259
round 5 : 215
round 6 : 191
round 7 : 148
round 8 : 144
round 9 : 148
round 10 : 133
round 11 : 137
round 12 : 122
round 13 : 112
round 14 : 102
round 15 : 93
round 16 : 89
round 17 : 74
round 18 : 73
round 19 : 57
round 20 : 54
round 21 : 52
round 22 : 47
round 23 : 40
round 24 : 38
round 25 : 38
round 26 : 35
round 27 : 35
round 28 : 31
round 29 : 30
round 30 : 27
```

In[97]:=
```
Animate[Image[cout7[[k]]], {k, 1, Length[cout7], 1}, AnimationRunning → False]
```

In[98]:=
```
GraphicsGrid[{{Image[cout7[[1]]], Image[cout7[[10]]]},
  {Image[cout7[[20]]], Image[cout7[[30]]]}}, ImageSize → Large]
```

Out[98]=



Third example: smoothed version of the second example

In[99]:=
```
img3 = GaussianFilter[img2, 3]
```

Out[99]=

In[100]:=
```
data3 = ImageData[img3][[301 ;; 700, 301 ;; 700]];
```

In[101]:=
```
Image[data3]
```

Out[101]=

In[102]:=
```
edge3 = edges[data3, 0.25];
```

In[103]:=
```
cout8 = cannydemo[edge3, 0.25, 0.6, 50];
```

```
strong edge vertices: 7637
weak edge vertices: 22 882
further edge vertices:
round 1 : 4389
round 2 : 780
round 3 : 312
round 4 : 245
round 5 : 170
round 6 : 134
round 7 : 117
round 8 : 90
round 9 : 71
round 10 : 55
round 11 : 50
round 12 : 45
round 13 : 40
round 14 : 28
```

```
round 15 : 19
round 16 : 20
round 17 : 19
round 18 : 16
round 19 : 21
round 20 : 18
round 21 : 17
round 22 : 16
round 23 : 17
round 24 : 16
round 25 : 15
round 26 : 16
round 27 : 16
round 28 : 16
round 29 : 18
round 30 : 16
round 31 : 15
round 32 : 14
round 33 : 12
round 34 : 13
round 35 : 11
round 36 : 13
round 37 : 13
round 38 : 12
round 39 : 11
round 40 : 10
round 41 : 11
round 42 : 10
round 43 : 12
round 44 : 11
round 45 : 9
round 46 : 10
round 47 : 9
round 48 : 11
round 49 : 9
round 50 : 10
```

In[104]:=
```
Animate[Image[cout8[[k]]], {k, 1, Length[cout8], 1}, AnimationRunning → False]
```

In[105]:=
```
GraphicsGrid[{{Image[cout8[[1]]], Image[cout8[[10]]]},
  {Image[cout8[[20]]], Image[cout8[[30]]]},
  {Image[cout8[[40]]], Image[cout8[[50]]]}}, ImageSize → Large]
```

Out[105]=

Fourth example

In[106]:=
```
img4 = Import["~/Lehre/Wavelets-All/WTBV-10/CWT/zebras.jpg"]
```

Out[106]=

In[107]:=
```
img4=ColorConvert[img4,"GrayScale"]
```

Out[107]=

In[108]:=
```
img4g=GaussianFilter[img4,3]
```

Out[108]=

In[109]:=
```
data4g=ImageData[img4g];
```

In[110]:=
```
edge4=edges[data4g,0.2];
```

In[111]:=
```
cout9=cannydemo[edge4,0.2,0.5,30];
```

```
strong edge vertices: 5518
weak edge vertices: 17 175
further edge vertices:
round 1 : 2501
round 2 : 883
round 3 : 385
round 4 : 318
round 5 : 245
round 6 : 205
round 7 : 188
round 8 : 176
round 9 : 165
round 10 : 147
round 11 : 135
round 12 : 121
round 13 : 109
round 14 : 81
round 15 : 77
round 16 : 64
round 17 : 62
round 18 : 56
round 19 : 47
round 20 : 51
round 21 : 49
round 22 : 46
round 23 : 41
round 24 : 32
round 25 : 38
round 26 : 40
round 27 : 43
round 28 : 30
round 29 : 30
round 30 : 30
```

In[112]:=
```
Animate[Image[cout9[[k]]],{k,1,Length[cout9],1},AnimationRunning->False]
```

In[113]:=
```
GraphicsGrid[{{Image[cout9[[1]]], Image[cout9[[5]]]},
  {Image[cout9[[10]]], Image[cout9[[15]]]},
  {Image[cout9[[20]]], Image[cout9[[30]]]}}, ImageSize → Large]
```

Out[113]=