Illustrating edge detection

Procedures

In[1]:=
```
dir2vec[a_] := Module[{aa},
  (*
  Computing the octant in which vector a={ax,ay} lies
  *)
  If[a == {0, 0}, Return[{1, 0}]];
  aa = Mod[N[ArcTan[a[[1]], a[[2]]] + Pi / 2], Pi, -Pi / 2];
  Which[
   Abs[aa] < Pi / 8, {1, 0},
   Pi / 8 < aa ≤ 3 Pi / 8, {1, 1},
   -3 Pi / 8 < aa ≤ - Pi / 8, {1, -1},
   True, {0, 1}]
 ]
```

In[2]:=
```
grad[A_, hfilter_, vfilter_] :=
 (*
 Computing the arrays for length (val) and
  direction (dir) of gradient vectors by filtering array A with
  filter hfilter and vfilter
 *)
 Module[{AA, val, dir, m, n, Ah, Av, h, v},
  {m, n} = Dimensions[A];
  AA = ArrayPad[A, {1, 1}];
  Ah = ListConvolve[hfilter, AA];
  Av = ListConvolve[vfilter, AA];
  val = Table[0, {m}, {n}];
  dir = Table[0, {m}, {n}];
  Do[
   {h, v} = {Ah[[x + 1, y + 1]], Av[[x + 1, y + 1]]};
   val[[x, y]] = N[Sqrt[h^2 + v^2]];
   dir[[x, y]] = dir2vec[{h, v}],
   {y, 1, n}, {x, 1, m}];
  {val, dir}
 ]
```

In[3]:=
```
WTstep[AX_, AY_, H_, V_, h_, v_] :=
 Module[{Ax, Ay, Wx, Wy},
  If[Dimensions[AX] ≠ Dimensions[AY], Throw["dimensions don't match"]];
  Ax = ListConvolve[H, AX, {1, 1}, 0];
  Ay = ListConvolve[V, AY, {1, 1}, 0];
  Wx = ListConvolve[h, AX, {1, 1}, 0];
  Wy = ListConvolve[v, AY, {1, 1}, 0];
  {Ax, Ay, Wx, Wy}
 ]
```

In[4]:=
```
edges1m[A_, hfilter_, vfilter_, level_, mode_] :=
 (*
 Detecting egde vertices in array A using
    gradient analysis. The output cand is a 0-
   1-array of the same dimension as A
    which shows all candidates.
     Shown ar all vertices which satisfy the edge
    vertex criterion and where the local gradient maximum is ≥
  "level"*(mode gradient) where mode is either "max" or
   "mean"
 *)
 Module[{m, n, val, dir, mm, VVal, Cand,
   x, x1, x2, y, y1, y2, val0, val1, val2},
  {m, n} = Dimensions[A];
  {val, dir} = grad[A, hfilter, vfilter];
  If[mode ≠ "max" && mode ≠ "mean",
    Throw["mode undetermined"]];
  If[mode == "max", mm = Max[Flatten[val]]];
  If[mode == "mean", mm = Mean[Flatten[val]]];
  VVal = ArrayPad[val, {1, 1}];
  Cand = Table[0, {m}, {n}];
  Do[
   {x1, y1} = {x + 1, y + 1} + dir[[x, y]];
   {x2, y2} = {x + 1, y + 1} - dir[[x, y]];
   val0 = VVal[[x + 1, y + 1]];
   val1 = VVal[[x1, y1]];
   val2 = VVal[[x2, y2]];
   If[
    val0 ≥ Max[val1, val2] && val0 ≥ level mm,
    Cand[[x, y]] = 1],
   {x, 1, m}, {y, 1, n}];
  Cand
 ]
```

In[5]:=
```
edges2m[A_, B_, level_, mode_] :=
 (*
 Detecting egde vertices in array A using gradient analysis.
     Input is given as two x- and y-filtered array A and B.
     The output cand is a 0-1-array of the same dimension as A
    which shows all candidates.
     Shown are all vertices which satisfy the edge
    vertex criterion and where the local gradient maximum is ≥
  "level"*(mode gradient) where mode is either "max" or "mean"
 *)
 Module[{m, n, val, dir, mm, VVal, Cand, h, v,
   x, x1, x2, y, y1, y2, val0, val1, val2},
  {m, n} = Dimensions[A];
  If[{m, n} ≠ Dimensions[B],
   Throw["Dimensions don't match"]
  ];
  val = Table[0, {m}, {n}];
  dir = Table[0, {m}, {n}];
  Do[
   {h, v} = {A[[x, y]], B[[x, y]]};
   val[[x, y]] = N[Sqrt[h^2 + v^2]];
   dir[[x, y]] = dir2vec[{h, v}],
   {x, 1, m}, {y, 1, n}
  ];
  If[mode ≠ "max" && mode ≠ "mean",
    Throw["mode undetermined"]];
  If[mode == "max", mm = Max[Flatten[val]]];
  If[mode == "mean", mm = Mean[Flatten[val]]];
  VVal = ArrayPad[val, {1, 1}];
  Cand = Table[0, {m}, {n}];
  Do[
   {x1, y1} = {x + 1, y + 1} + dir[[x, y]];
   {x2, y2} = {x + 1, y + 1} - dir[[x, y]];
   val0 = VVal[[x + 1, y + 1]];
   val1 = VVal[[x1, y1]];
   val2 = VVal[[x2, y2]];
   If[
    val0 ≥ Max[val1, val2] && val0 ≥ level mm,
    Cand[[x, y]] = 1],
   {x, 1, m}, {y, 1, n}
  ];
  Cand
 ]
```

In[6]:=
```
edges3m[A_, B_, low_, high_, mode_] :=
 (*
 Detecting edge vertices using gradient analysis. Input is given as two x-
   and y-filtered data arrays A and B.
     Output is given in two arrays Clow and Chigh of same dimensions as A
    and B. Shown are all vertices which satisfy the edge vertex criterion
    and where the local gradient maximum is ≥ "level"*(mode gradient)
   where mode is either "max" or "mean"
 *)
 Module[{m, n, Val, Dir, mm, VVal, Clow, Chigh, h, v,
   x, x1, x2, y, y1, y2, val, val1, val2},
  {m, n} = Dimensions[A];
  If[{m, n} ≠ Dimensions[B],
   Throw["Dimensions don't match"]
  ];
  Val = Table[0, {m}, {n}];
  Dir = Table[0, {m}, {n}];
  Do[
   {h, v} = {A[[x, y]], B[[x, y]]};
   Val[[x, y]] = N[Sqrt[h^2 + v^2]];
   Dir[[x, y]] = dir2vec[{h, v}],
   {x, 1, m}, {y, 1, n}
  ];
  If[mode ≠ "max" && mode ≠ "mean",
    Throw["mode undetermined"]];
  If[mode == "max", mm = Max[Flatten[Val]]];
  If[mode == "mean", mm = Mean[Flatten[Val]]];
  VVal = ArrayPad[Val, {1, 1}];
  Clow = Table[0, {m}, {n}];
  Chigh = Table[0, {m}, {n}];
  Do[
   {x1, y1} = {x + 1, y + 1} + Dir[[x, y]];
   {x2, y2} = {x + 1, y + 1} - Dir[[x, y]];
   val = VVal[[x + 1, y + 1]];
   val1 = VVal[[x1, y1]];
   val2 = VVal[[x2, y2]];
   If[
    val ≥ Max[val1, val2] && val ≥ low mm,
    Clow[[x, y]] = 1;
    If[val ≥ high mm, Chigh[[x, y]] = 1]],
   {x, 1, m}, {y, 1, n}
  ];
  {Clow, Chigh}
 ]
```

In[7]:=
```
cannym[A_, B_, low_, high_, iter_, mode_] :=
 (*
 Uses edges3 for 2-level edge detection by turning weak
   (low level) edge points into strong (high level) edge vertices
   mode is either "max" or "mean"
 *)
 Module[{k, m, n, Alow, AAlow, Ahigh, AAhigh, Diff},
  {m, n} = Dimensions[A];
  {Alow, Ahigh} = edges3m[A, B, low, high, mode];
  AAlow = ArrayPad[Alow, {1, 1}];
  AAhigh = ArrayPad[Ahigh, {1, 1}];
  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}]
 ]
```

Filter, Images

filters

In[8]:=
```
Clear[H, V, h, v]
```

In[9]:=
```
spread[fil_] := Most[Flatten[Map[{#, 0} &, fil]]]
```

In[17]:=
```
spread[fil_, n_] := Nest[spread[#] &, fil, n]
```

In[10]:=
```
spline1 = {1, 1} / Sqrt[2];
spline2 = Sqrt[2] * {1, 2, 1} / 4;
```

In[12]:=
```
H[k_] := KroneckerProduct[
    spread[spline2, k],
    spread[spline1, k]];
```

In[13]:=
```
V[k_] := KroneckerProduct[
    spread[spline1, k],
    spread[spline2, k]];
```

In[14]:=
```
h[k_] := {spread[{1, -1} / Sqrt[2], k]};
v[k_] := Transpose[h[k]];
```

In[18]:=
```
H[0] // MatrixForm
```

Out[18]//MatrixForm=

$$\begin{pmatrix} \frac{1}{4} & \frac{1}{4} \\ \frac{1}{2} & \frac{1}{2} \\ \frac{1}{4} & \frac{1}{4} \end{pmatrix}$$

In[19]:=
```
V[1] // MatrixForm
```

Out[19]//MatrixForm=

$$\begin{pmatrix} \frac{1}{4} & 0 & \frac{1}{2} & 0 & \frac{1}{4} \\ 0 & 0 & 0 & 0 & 0 \\ \frac{1}{4} & 0 & \frac{1}{2} & 0 & \frac{1}{4} \end{pmatrix}$$

In[20]:=
```
h[1] // MatrixForm
```

Out[20]//MatrixForm=

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

In[21]:=
```
v[2] // MatrixForm
```

Out[21]//MatrixForm=

$$\begin{pmatrix} \frac{1}{\sqrt{2}} \\ 0 \\ 0 \\ 0 \\ -\frac{1}{\sqrt{2}} \end{pmatrix}$$

test image (circle)

In[22]:=
```
Clear[x, y]
```

In[23]:=
```
x[n_] :=
  Table[{1}, {2^n + 1}].{Range[-2^(n - 1), 2^(n - 1)]};
```

In[24]:=
```
y[n_] :=
  Transpose[{Range[-2^(n - 1), 2^(n - 1)]}].{Table[1, {2^n + 1}]};
```

In[25]:=
```
IA = ImageAdjust;
```

In[26]:=
```
XY[n_] := x[n] + I * y[n];
```

In[27]:=
```
circle[n_, inner_, outer_] := Module[{abs, arg, CC, i, j},
  CC = Array[0, {2^n, 2^n}];
  For[i = 1, i ≤ 2^n, i++,
   For[j = 1, j ≤ 2^n, j++,
    {abs, arg} = AbsArg[XY[n][[i, j]]];
    CC[[i, j]] =
     If[abs < outer,
      If[abs < inner,
       Round[(arg + Pi / 12) / (Pi / 6)] + 6, 1
      ], 12];
   ]
  ];
  CC / Max[Abs[Flatten[CC]]]
 ]
```

In[153]:=
```
circle7 = 1 - circle[7, 55, 60];
```

In[28]:=
```
circle7 = Import["circle7.m"];
```

In[29]:=
```
Image[circle7]
```

Out[29]=

In[30]:=
```
AX = circle7; AY = circle7;
```

In[31]:=
```
idim = ImageDimensions[Image[circle7]]
```

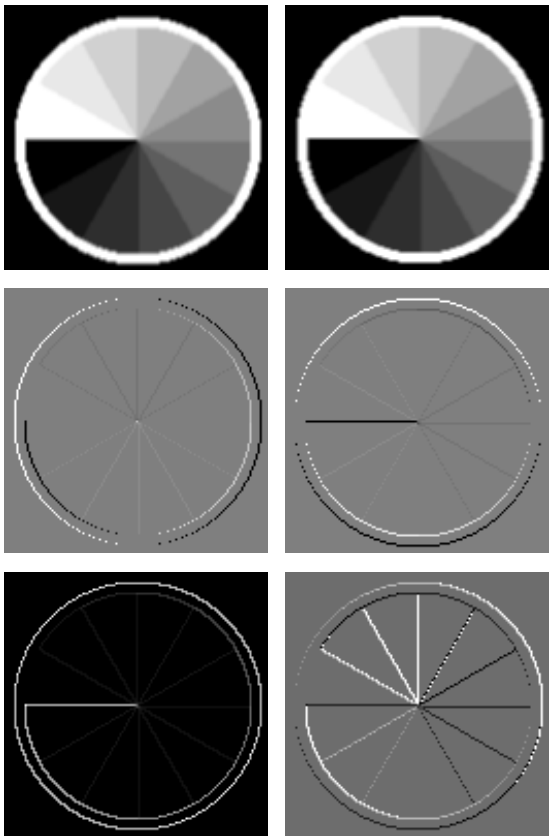
Out[31]=
```
{128, 128}
```

In[32]:=
```
{A1x, A1y, W1x, W1y} = WTstep[AX, AY, H[0], V[0], h[0], v[0]];
```

In[33]:=
```
W1 = W1x + I * W1y;
```

In[34]:=
```
GraphicsGrid[{
   {IA[Image[A1x]], IA[Image[A1y]]},
   {IA[Image[W1x]], IA[Image[W1y]]},
   {IA[Image[Abs[W1]]], IA[Image[Arg[W1]]]}
  }
]
```

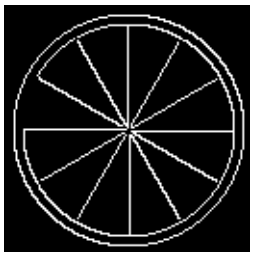
Out[34]=



In[35]:=
```
K1 = edges2m[W1x, W1y, 1, "mean"];
```

In[36]:=
```
Image[K1]
```

Out[36]=

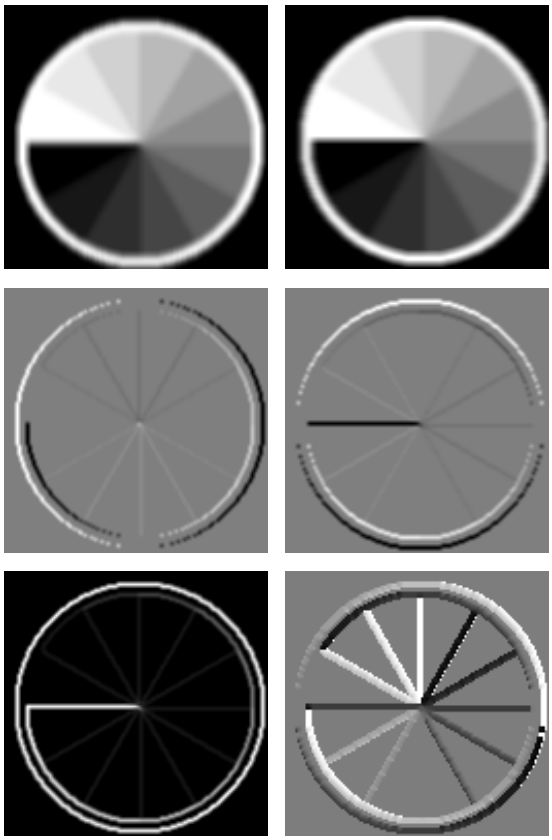


In[37]:=
```
{A2x, A2y, W2x, W2y} = WTstep[A1x, A1y, H[1], V[1], h[1], v[1]];
```

In[38]:=
```
W2 = W2x + I * W2y;
```

In[39]:=
```
GraphicsGrid[{
   {IA[Image[A2x]], IA[Image[A2y]]},
   {IA[Image[W2x]], IA[Image[W2y]]},
   {IA[Image[Abs[W2]]], IA[Image[Arg[W2]]]}
  }
]
```

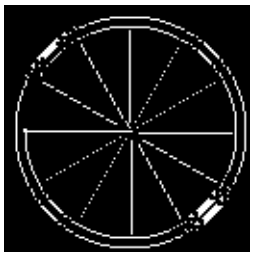
Out[39]=



In[40]:=
```
K2 = edges2m[W2x, W2y, 1, "mean"];
```

In[41]:=
```
IA[Image[K2]]
```

Out[41]=

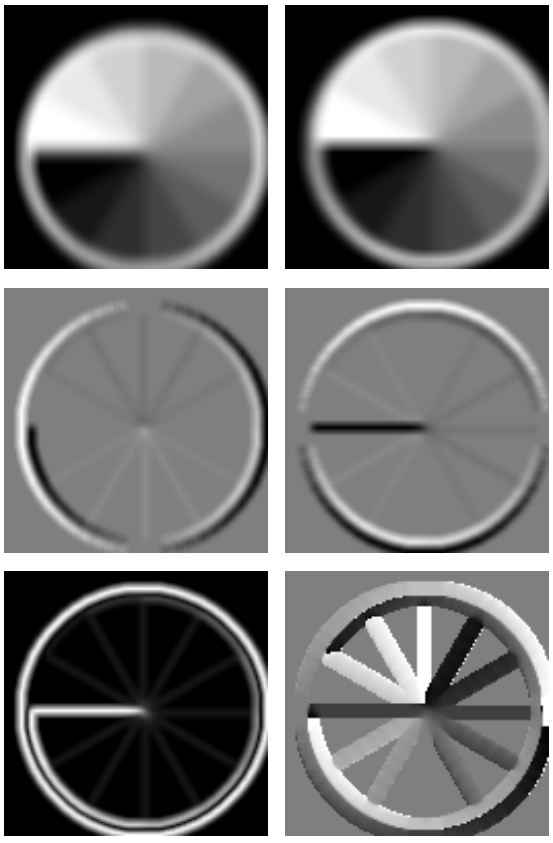


In[42]:=
```
{A3x, A3y, W3x, W3y} = WTstep[A2x, A2y, H[2], V[2], h[2], v[2]];
```

In[43]:=
```
W3 = W3x + I * W3y;
```

In[44]:=
```
GraphicsGrid[{
   {IA[Image[A3x]], IA[Image[A3y]]},
   {IA[Image[W3x]], IA[Image[W3y]]},
   {IA[Image[Abs[W3]]], IA[Image[Arg[W3]]]}
  }
]
```

Out[44]=



In[45]:=
```
K3 = edges2m[W3x, W3y, 0.4, "mean"];
```

In[46]:=
```
IA[Image[K3]]
```

Out[46]=

test image (fence)

In[47]:=
```
fence = Import["~/Desktop/WTBV/WTBV-15/Baeni_Materialien/MatLab/Weide.bmp"]
```

Out[47]=

In[48]:=
```
ImageDimensions[fence]
```

Out[48]=
```
{256, 256}
```

In[49]:=
```
fencedat = ImageData[ColorConvert[fence, "Grayscale"]];
```

In[50]:=
```
AX = fencedat; AY = fencedat;
```

In[51]:=
```
Image[AY]
```

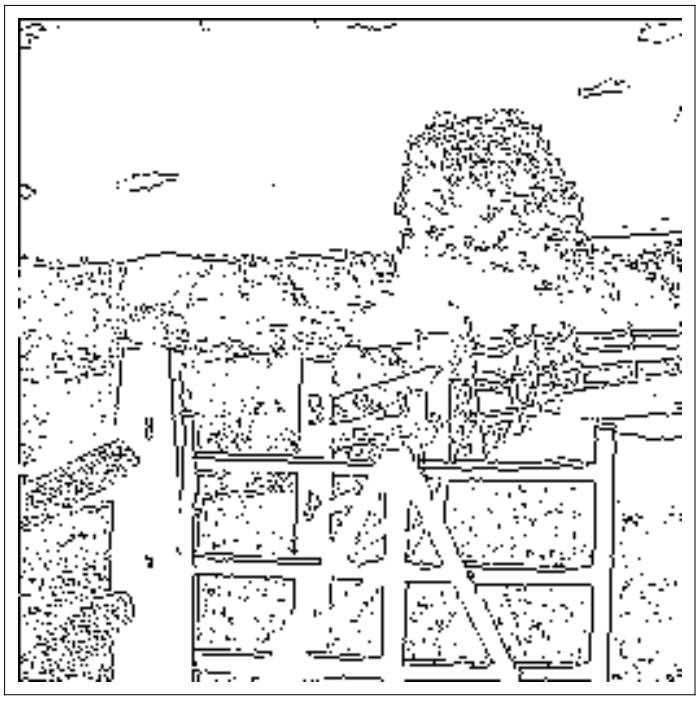
Out[51]=

In[52]:=
```
{A1x, A1y, W1x, W1y} = WTstep[AX, AY, H[0], V[0], h[0], v[0]];
```

In[53]:=
```
fenceout1 = edges2m[W1x, W1y, 0.10, "max"];
```

In[54]:=
```
ArrayPlot[fenceout1]
```

Out[54]=

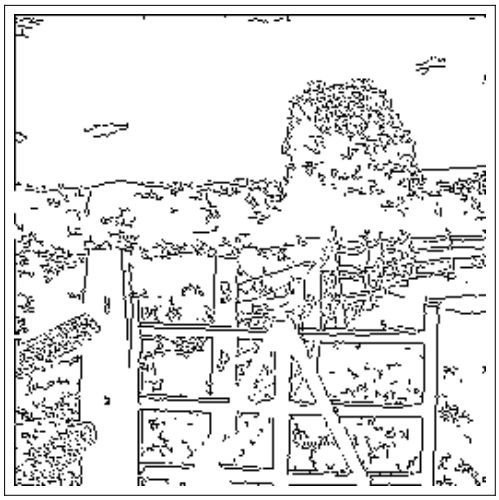


In[55]:=
```
cannym[W1x, W1y, 0.05, 0.15, 10, "max"];
```

```
strong edge vertices: 5125

weak edge vertices: 10514

further edge vertices:

round 1 : 1257

round 2 : 523

round 3 : 315

round 4 : 176

round 5 : 108

round 6 : 69

round 7 : 52

round 8 : 36

round 9 : 31

round 10 : 20
```

In[56]:=
```
ArrayPlot[%]
```

Out[56]=

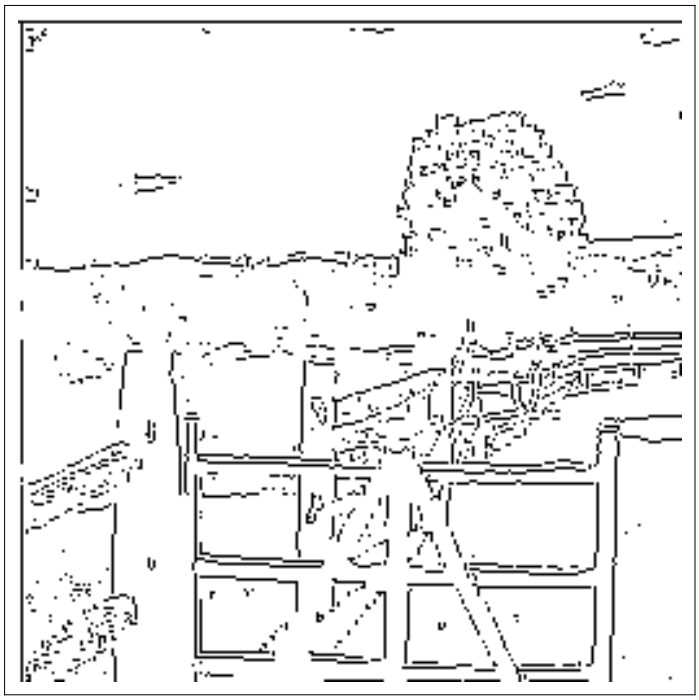


In[57]:=
```
{A2x, A2y, W2x, W2y} = WTstep[A1x, A1y, H[1], V[1], h[1], v[1]];
```

In[58]:=
```
fenceout2 = edges2m[W2x, W2y, 0.15, "max"];
```

In[59]:=
```
ArrayPlot[fenceout2]
```

Out[59]=

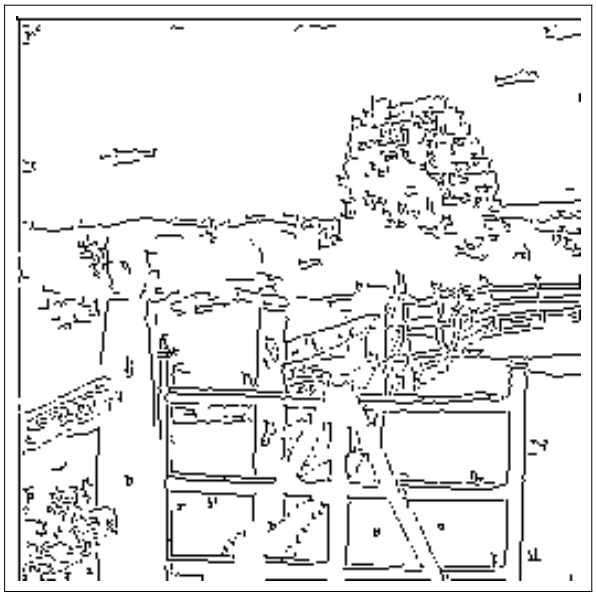


In[60]:=
```
cannym[W2x, W2y, 0.05, 0.15, 10, "max"];
```

```
strong edge vertices: 4857

weak edge vertices: 9403

further edge vertices:

round 1 : 680

round 2 : 298

round 3 : 179

round 4 : 118

round 5 : 63

round 6 : 46

round 7 : 29

round 8 : 25

round 9 : 13

round 10 : 10
```

In[61]:=
```
ArrayPlot[%]
```

Out[61]=

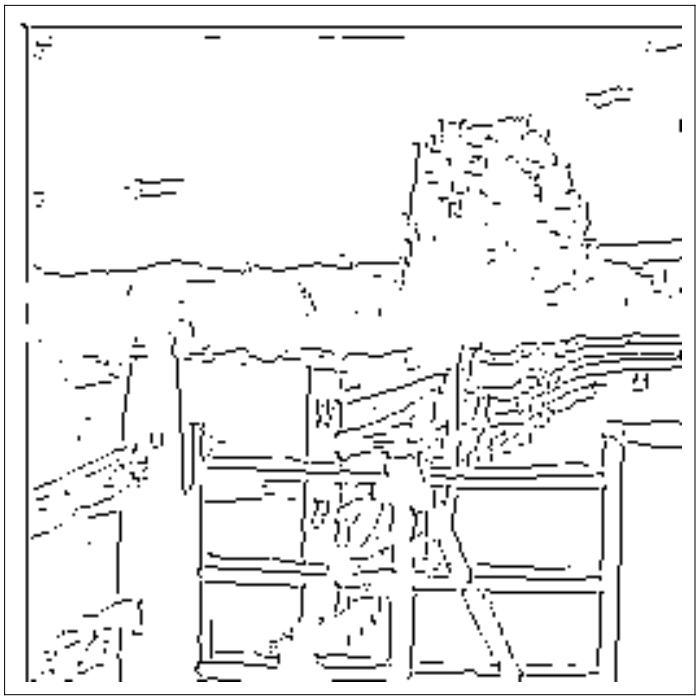


In[62]:=
```
{A3x, A3y, W3x, W3y} = WTstep[A2x, A2y, H[2], V[2], h[2], v[2]];
```

In[64]:=
```
fenceout3 = edges2m[W3x, W3y, 0.12, "max"];
```

In[65]:=
```
ArrayPlot[fenceout3]
```

Out[65]=

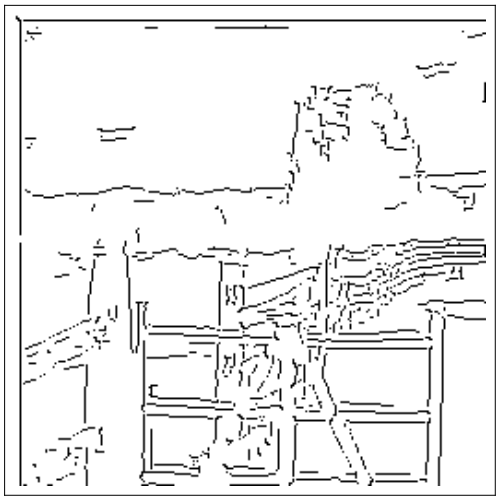


In[66]:=
```
cannym[W3x, W3y, 0.05, 0.15, 10, "max"];
```

```
strong edge vertices: 3865

weak edge vertices: 6448

further edge vertices:

round 1 : 255

round 2 : 110

round 3 : 55

round 4 : 37

round 5 : 25

round 6 : 17

round 7 : 12

round 8 : 6

round 9 : 5

round 10 : 4
```

In[67]:=
```
ArrayPlot[%]
```

Out[67]=

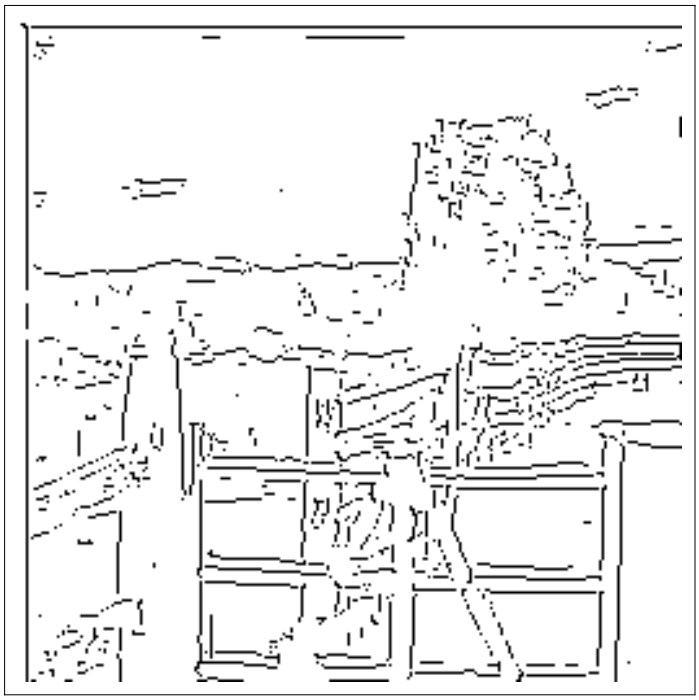


In[68]:=
```
{A4x, A4y, W4x, W4y} = WTstep[A3x, A3y, H[3], V[3], h[3], v[3]];
```

In[69]:=
```
eout4 = edges2m[W3x, W3y, 0.10, "max"];
```

In[70]:=
```
ArrayPlot[%]
```

Out[70]=



In[73]:=
```
cannym[W4x, W4y, 0.05, 0.15, 10, "max"];
```

```
strong edge vertices: 3051

weak edge vertices: 4663

further edge vertices:

round 1 : 92

round 2 : 59

round 3 : 40

round 4 : 25

round 5 : 18

round 6 : 13

round 7 : 11

round 8 : 9

round 9 : 7

round 10 : 6
```

In[74]:=
```
ArrayPlot[%]
```

Out[74]=