Image preprocessing in spatial domain

Sampling theorem, aliasing, interpolation, geometrical transformations

Revision: 1.4, dated: May 25, 2006

Tomáš Svoboda

Czech Technical University, Faculty of Electrical Engineering
Center for Machine Perception, Prague, Czech Republic

svoboda@cmp.felk.cvut.cz

http://cmp.felk.cvut.cz/~svoboda

Reminder — Convolution theorem

The Fourier transform of a convolution is the product of the Fourier transforms.

$$\mathcal{F}\{f(x,y) * h(x,y)\} = F(u,v)H(u,v)$$

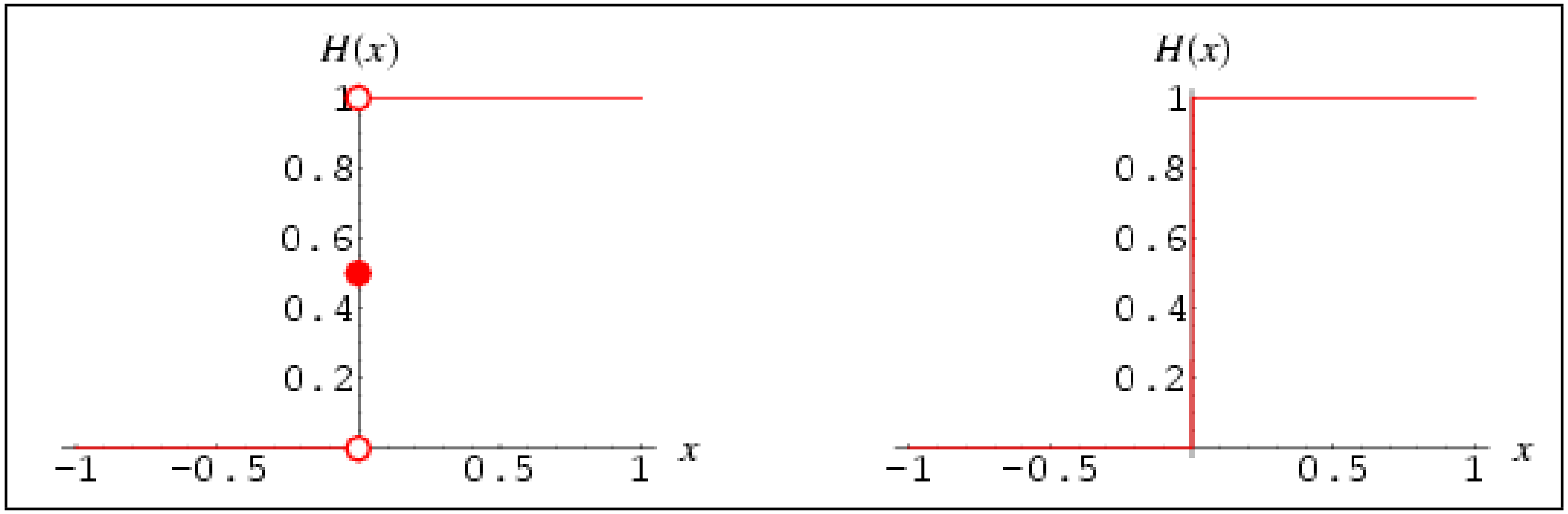
The Fourier transform of a product is the convolution of the Fourier transforms.

$$\mathcal{F}\{f(x,y)h(x,y)\} = F(u,v) * H(u,v)$$

Impulse (delta) function

$$\delta(x) = \frac{\partial H(x)}{\partial x}$$

where $H(x)$ is the Heaviside step function



[1]

Sifting property

$\int_{-\infty}^{\infty} f(x)\delta(x-a)dx = f(a)$

scaling property

$\delta(ax) = \frac{1}{\|a\|}\delta(x)$

---

[1] Eric W. Weisstein. "Heaviside Step Function." From MathWorld–A Wolfram Web Resource. http://mathworld.wolfram.com/HeavisideStepFunction.html

Shah function

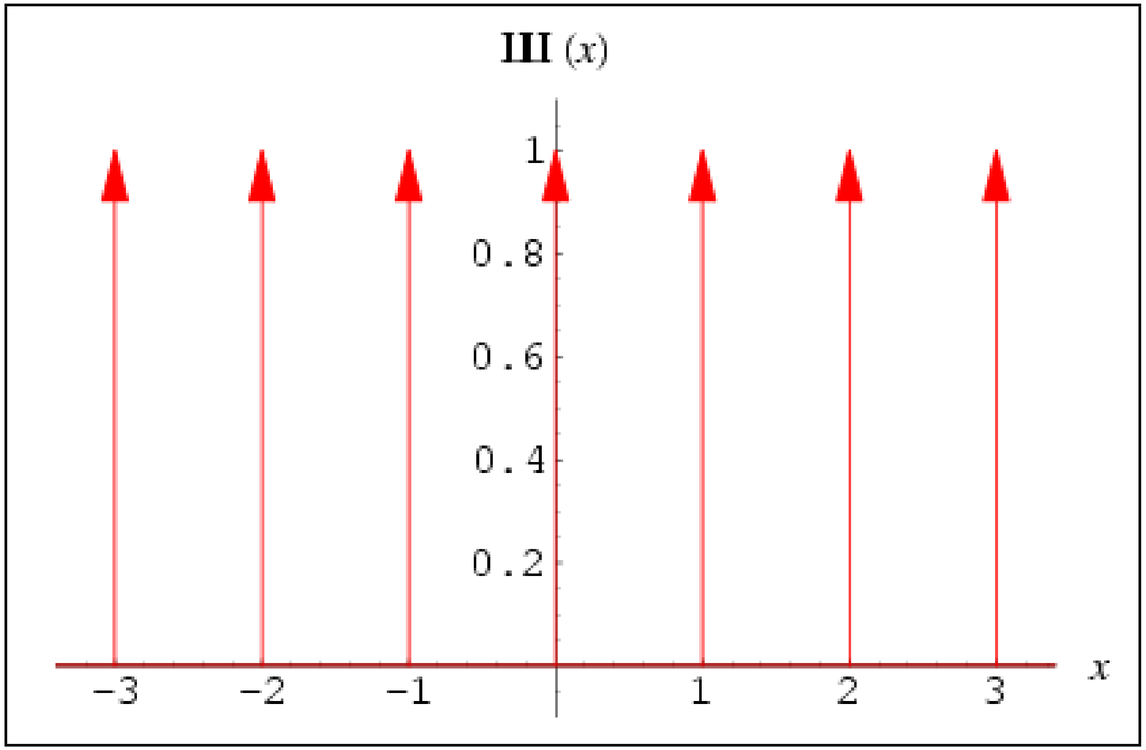
Replication of delta function

$$\text{III}(x) = \sum_{k=-\infty}^{\infty} \delta(x-k)$$

Shah function

Replication of delta function

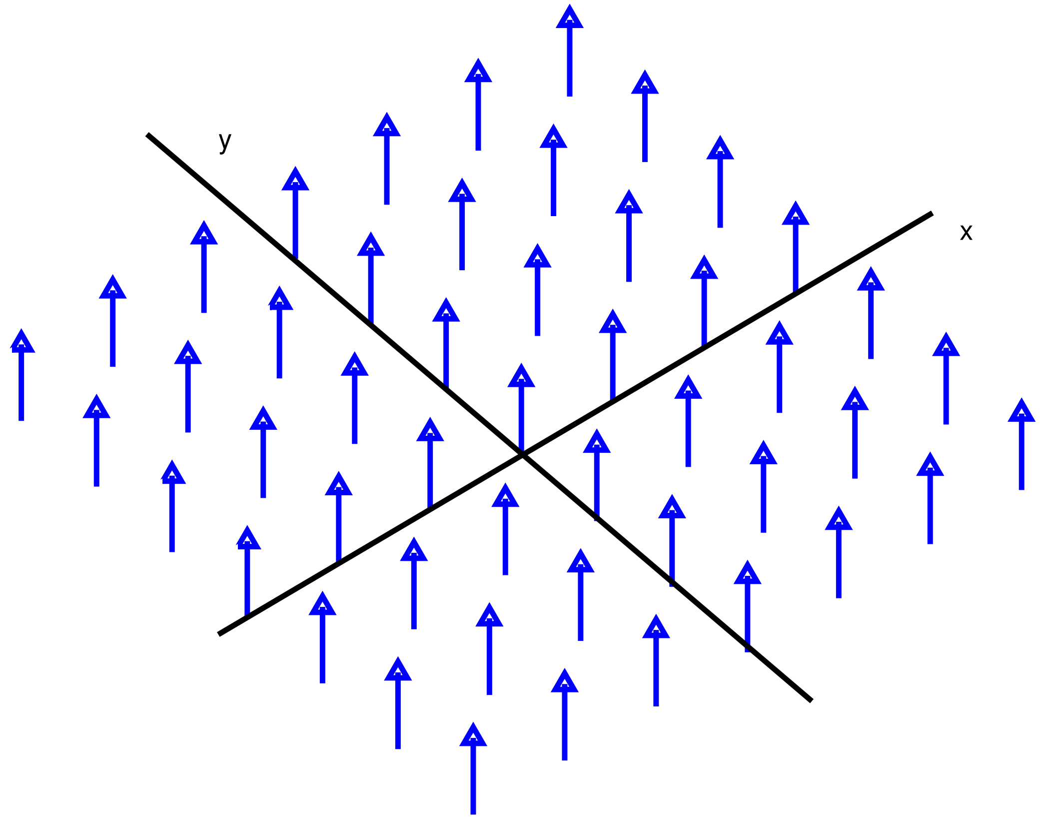
$$\mathrm{III}(x) = \sum_{k=-\infty}^{\infty} \delta(x-k)$$



[2]

[2] Eric W. Weisstein. "Shah Function." From MathWorld–A Wolfram Web Resource. http://mathworld.wolfram.com/ShahFunction.html

2D Shah function — bed and nails



$$\mathrm{III}(x,y) = \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} \delta(x-k, y-l)$$

2D Shah function — properties

For unit intervals

$$\mathrm{III}(x,y) = \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} \delta(x-k, y-l)$$

2D Shah function — properties

For unit intervals

$$\mathrm{III}(x,y) = \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} \delta(x-k, y-l)$$

If we need samples spaced X and Y

$$\mathrm{III}\left(\frac{x}{X}, \frac{y}{Y}\right) = \|XY\| \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} \delta(x-kX, y-lY)$$

because of the scaling property of the delta function

$$\delta(ax) = \frac{1}{\|a\|}\delta(x)$$

2D Shah function — Fourier pair

$$\mathcal{F}\{\mathrm{III}\left(\frac{x}{X},\frac{y}{Y}\right)\} = \|XY\|\,\mathrm{III}\,(Xu, Yv)$$

The Shah function is its own transform but the frequency inverses!

This will have important consequences!

Sampling by using the Shah function

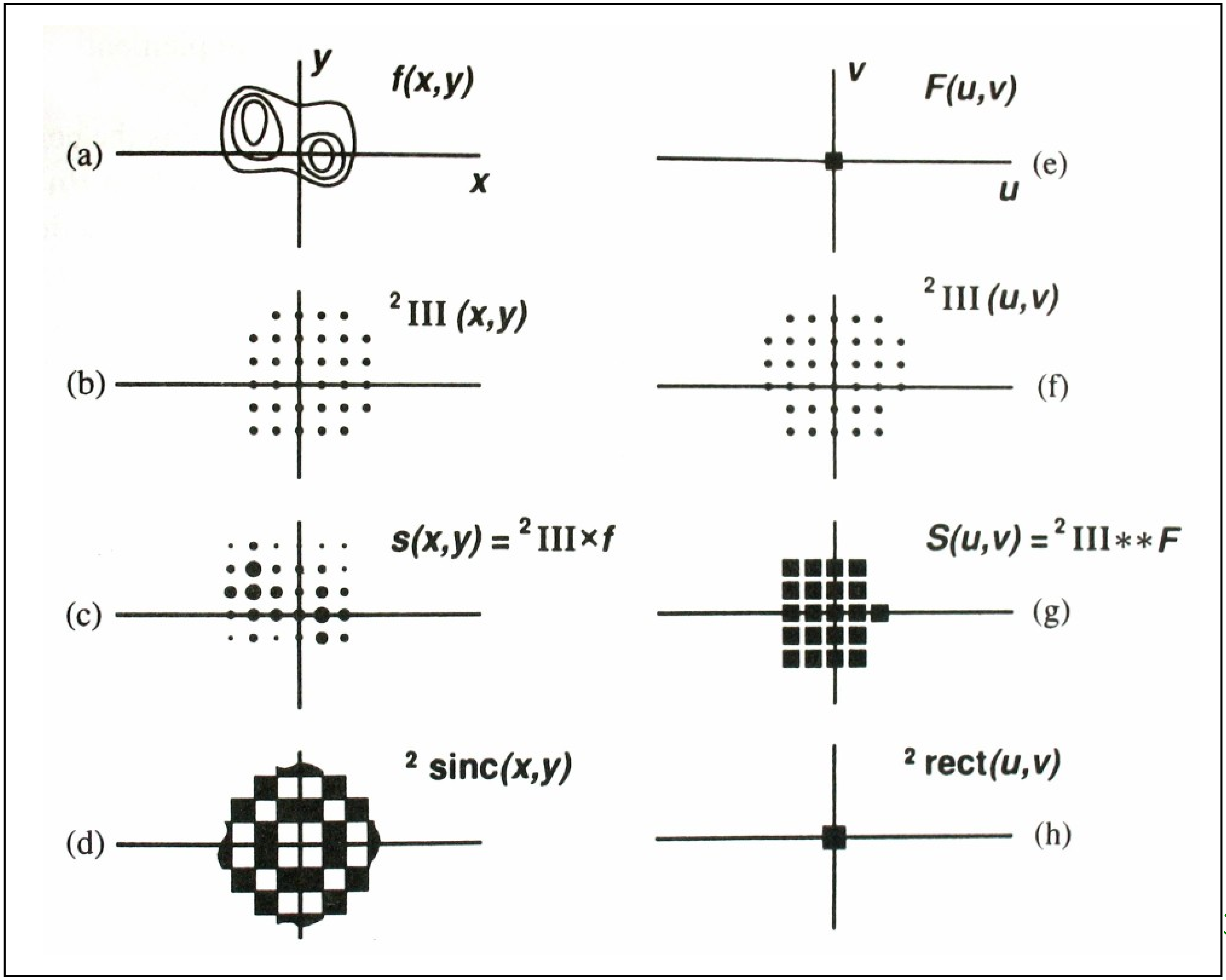
Product of a function with the Shah function.

$$s(x,y) = \mathrm{III}(x,y) f(x,y)$$

We know from convolution theorem that

$$S(u,v) = \mathrm{III}(u,v) * F(u,v)$$

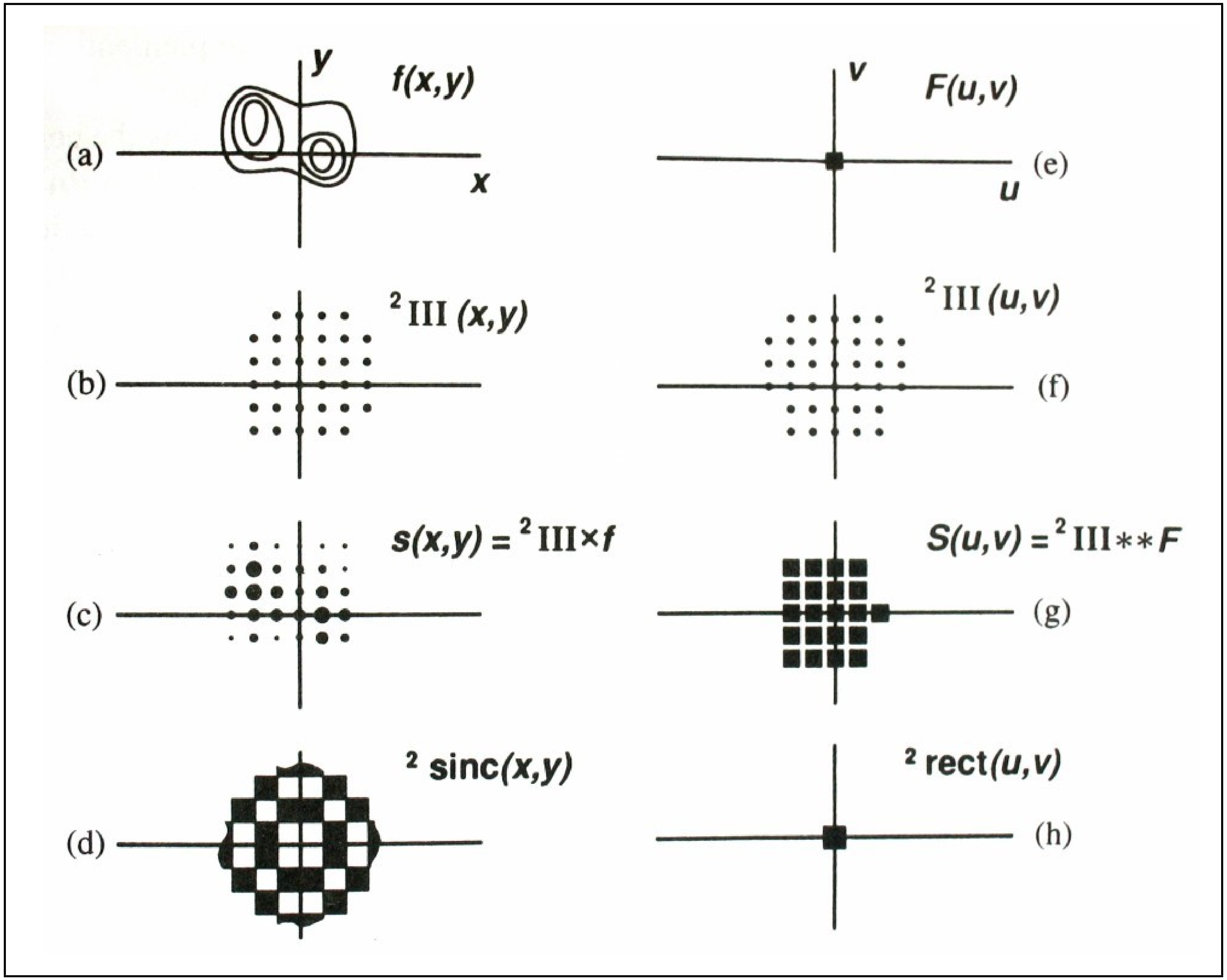
Sampling theorem



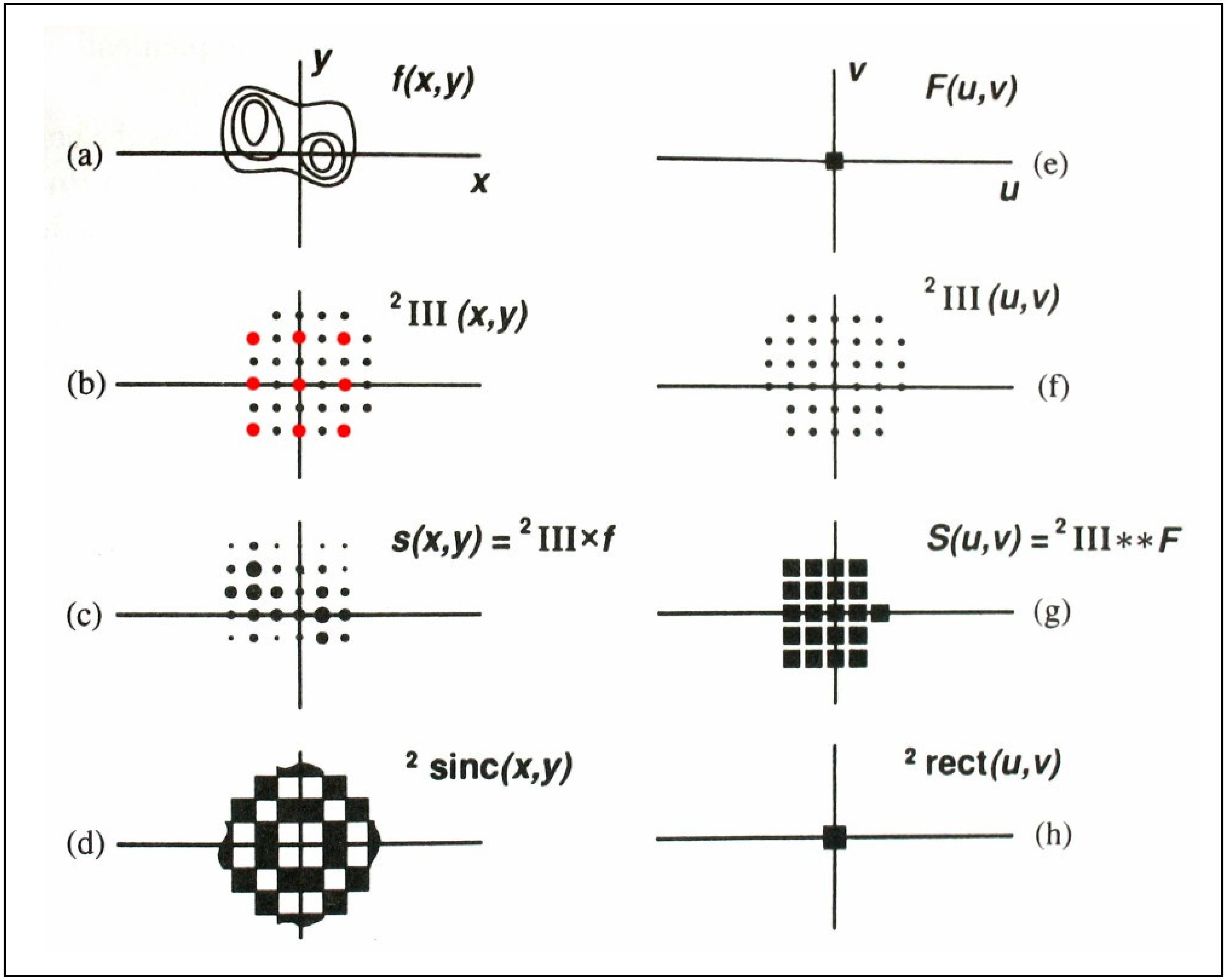
[3]

[3] Image taken from [1]

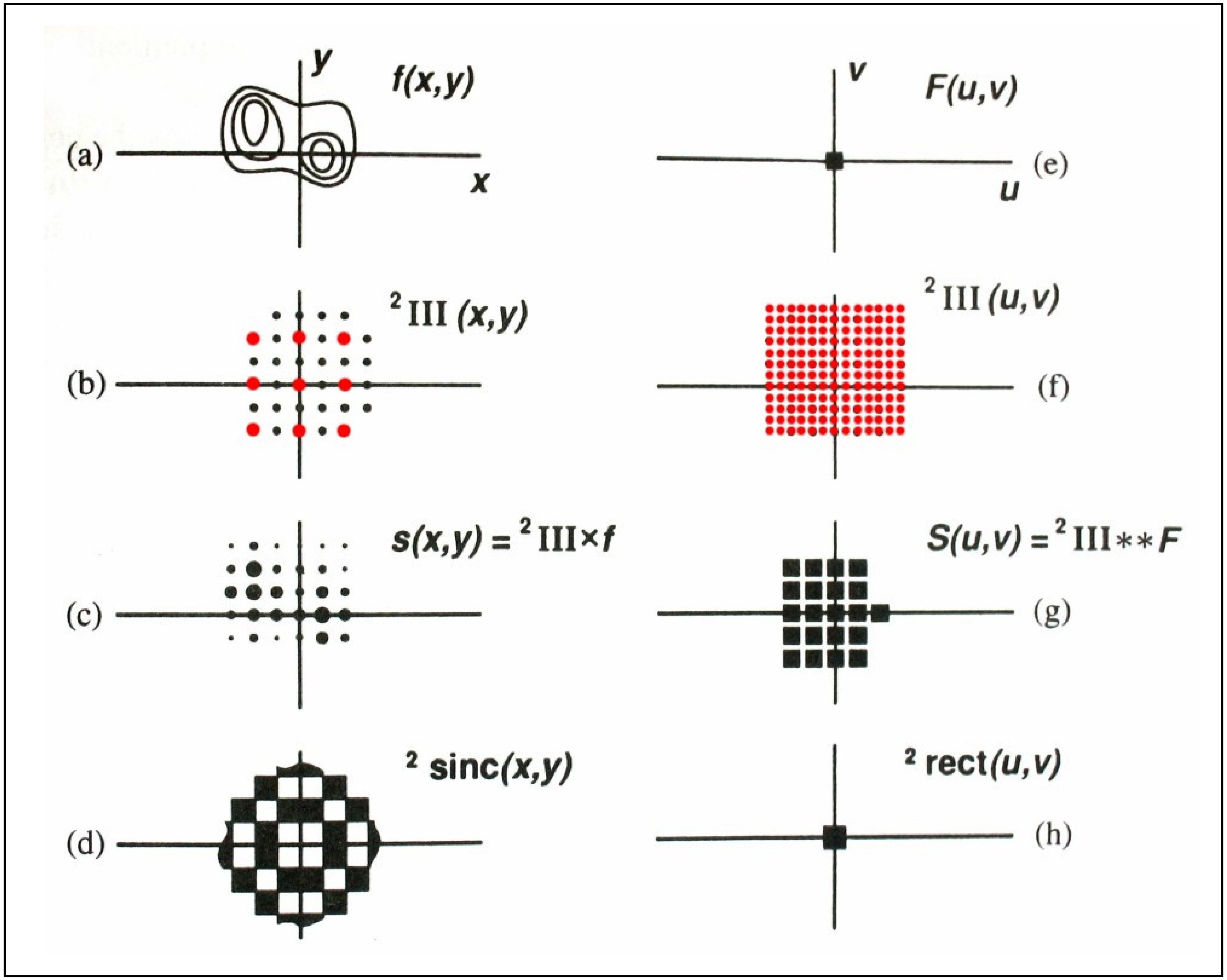
Sampling theorem—Aliasing



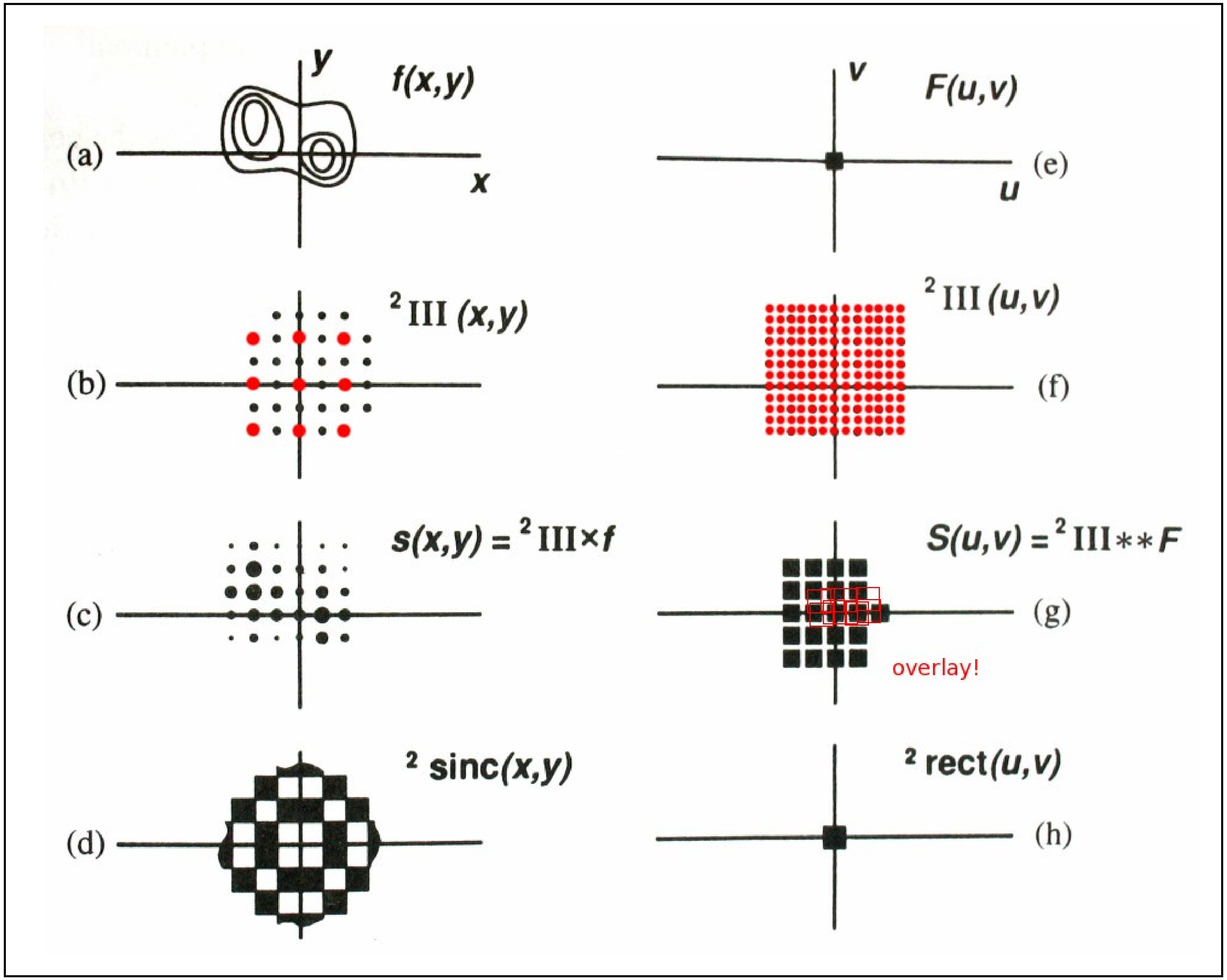
Sampling theorem—Aliasing



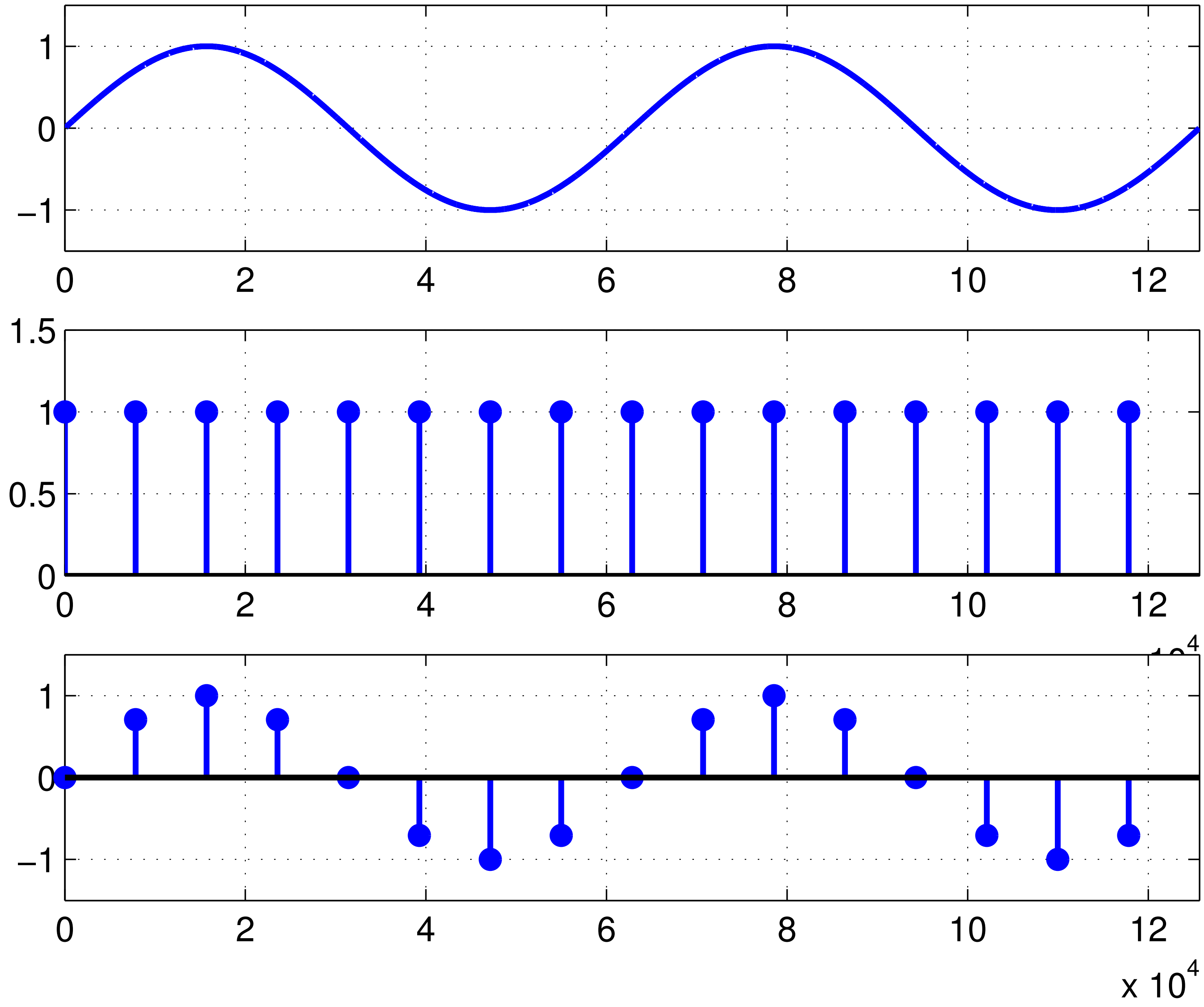
Sampling theorem—Aliasing



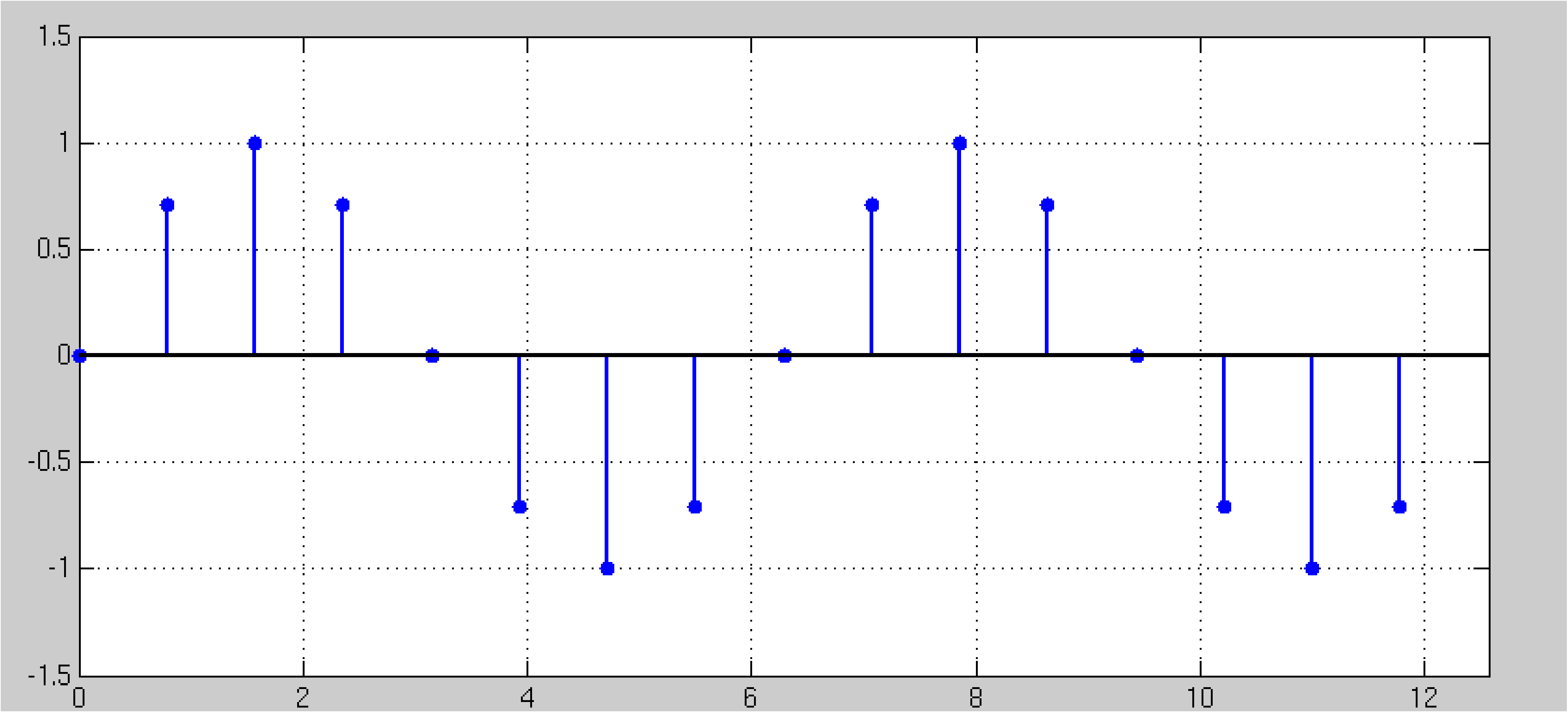
Sampling theorem—Aliasing



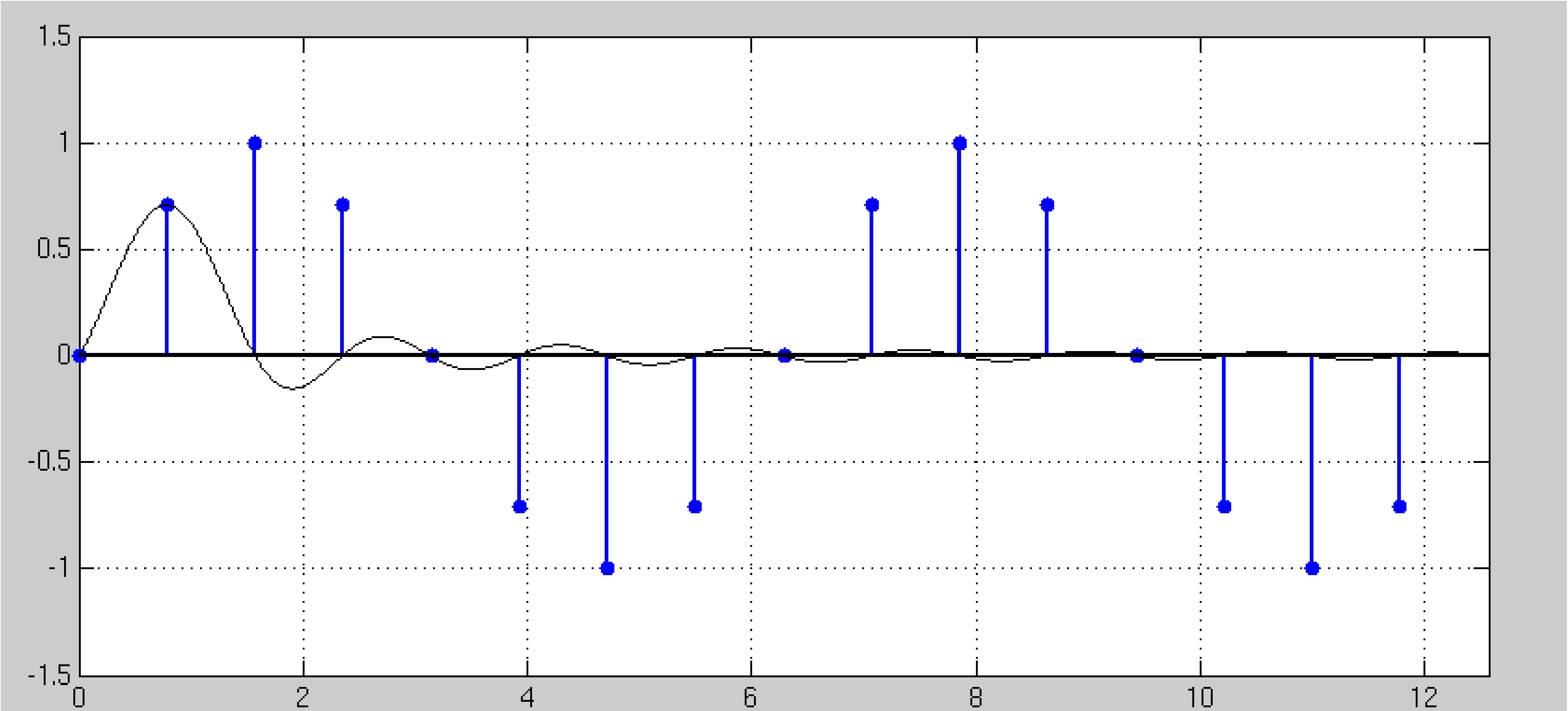
From continuous to discrete and back again



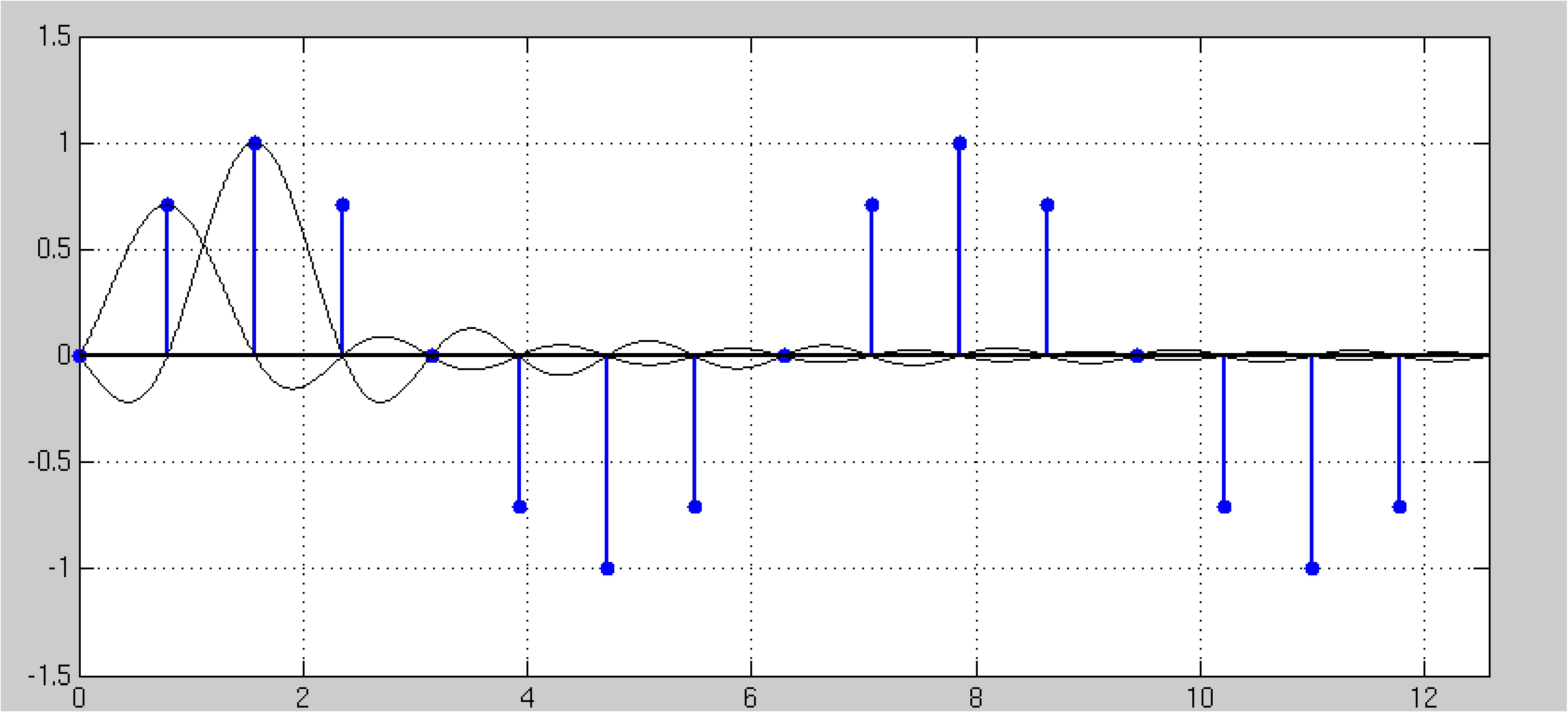
From continuous to discrete and back again



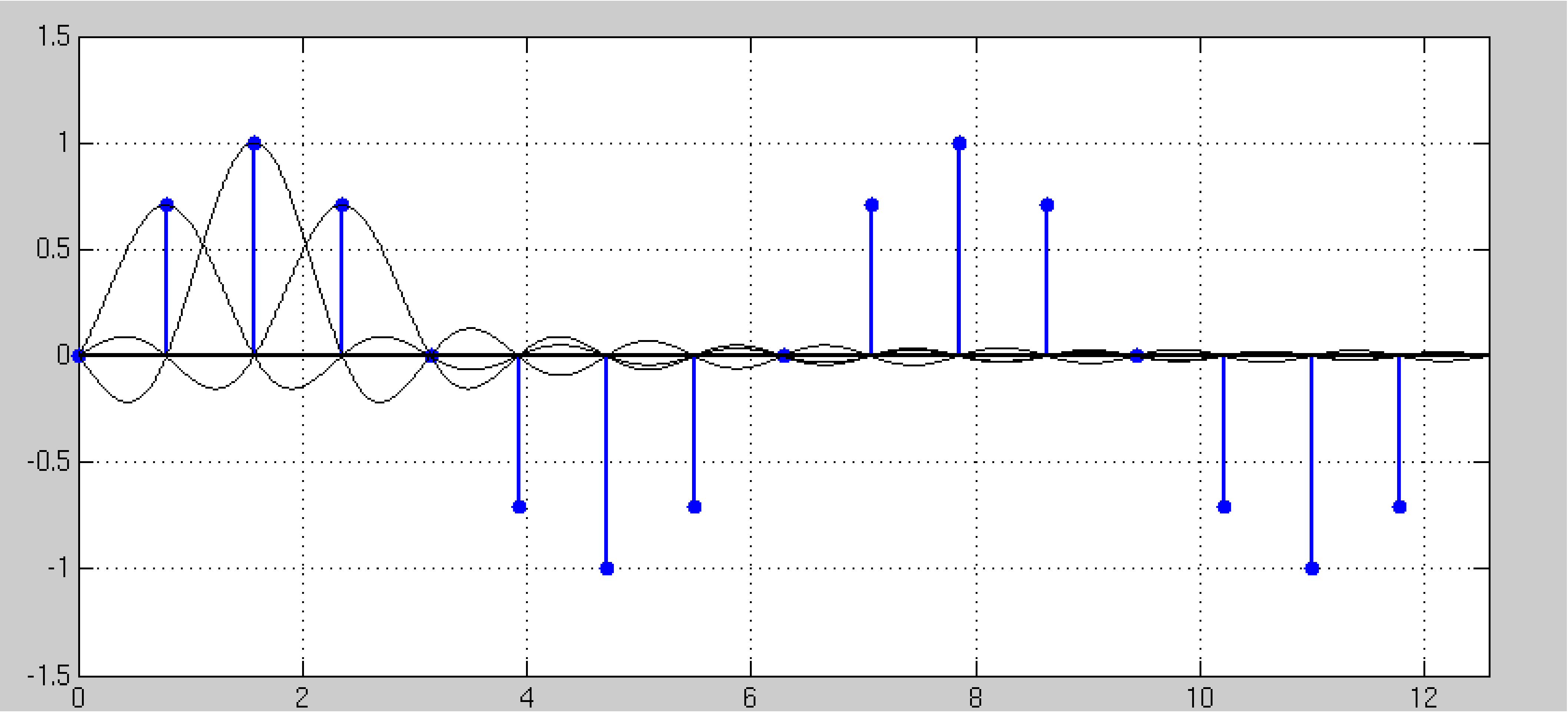
From continuous to discrete and back again



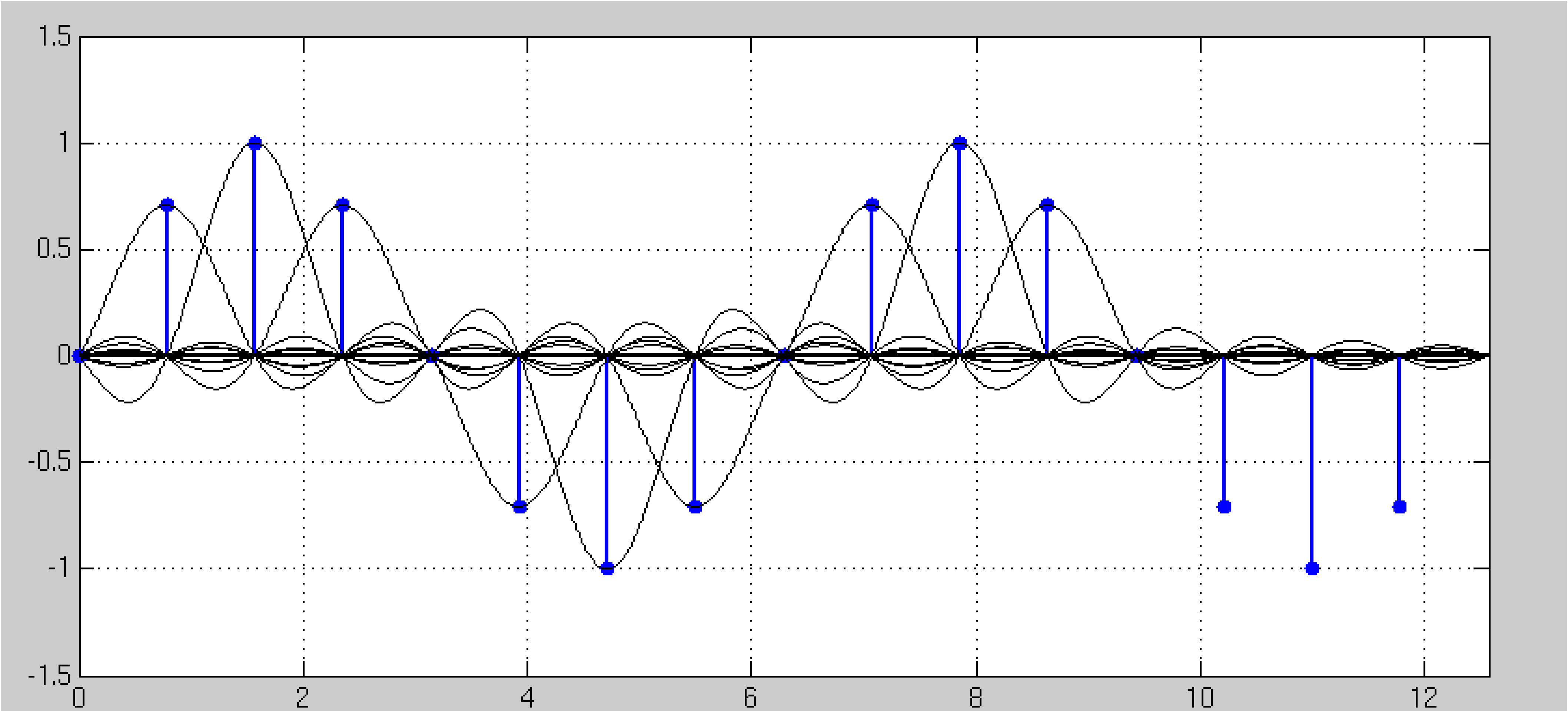
From continuous to discrete and back again



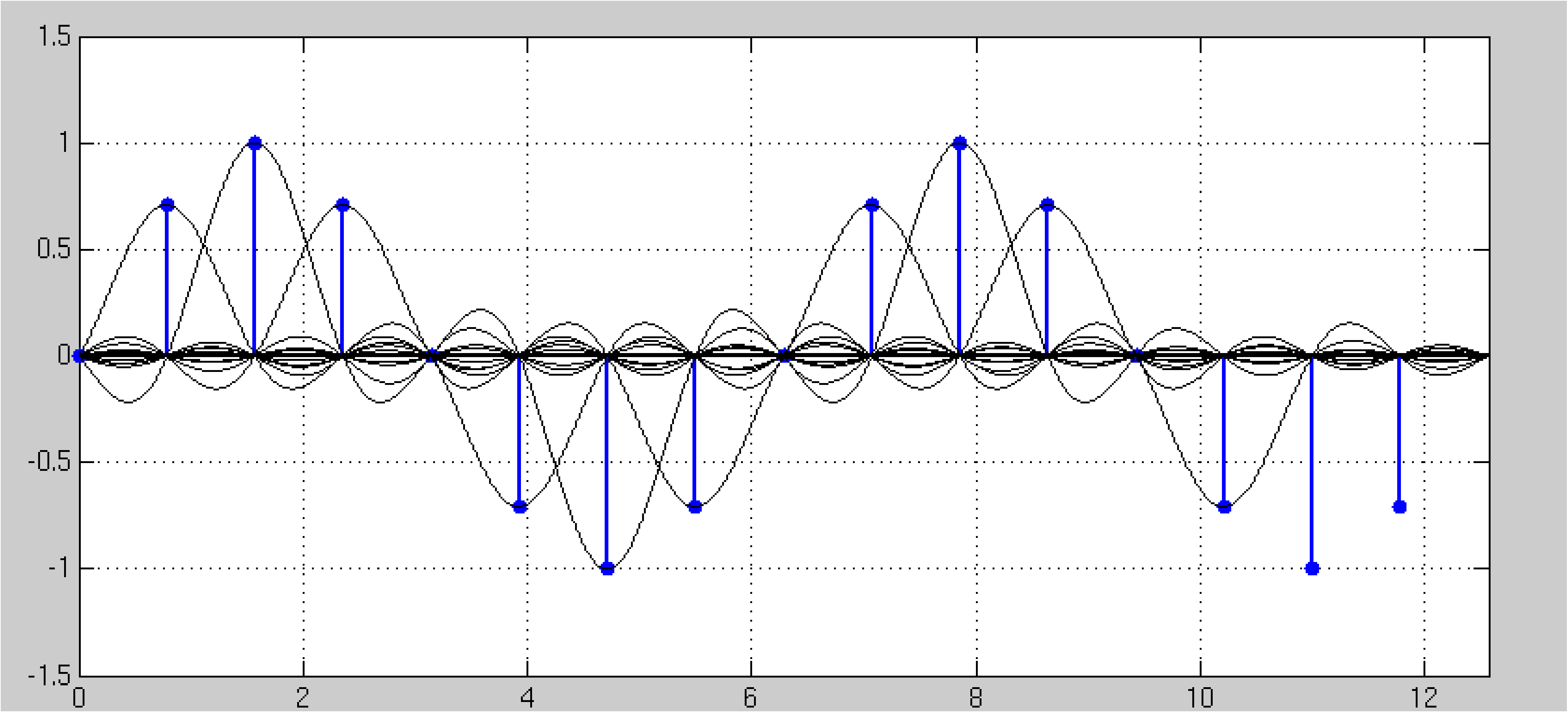
From continuous to discrete and back again



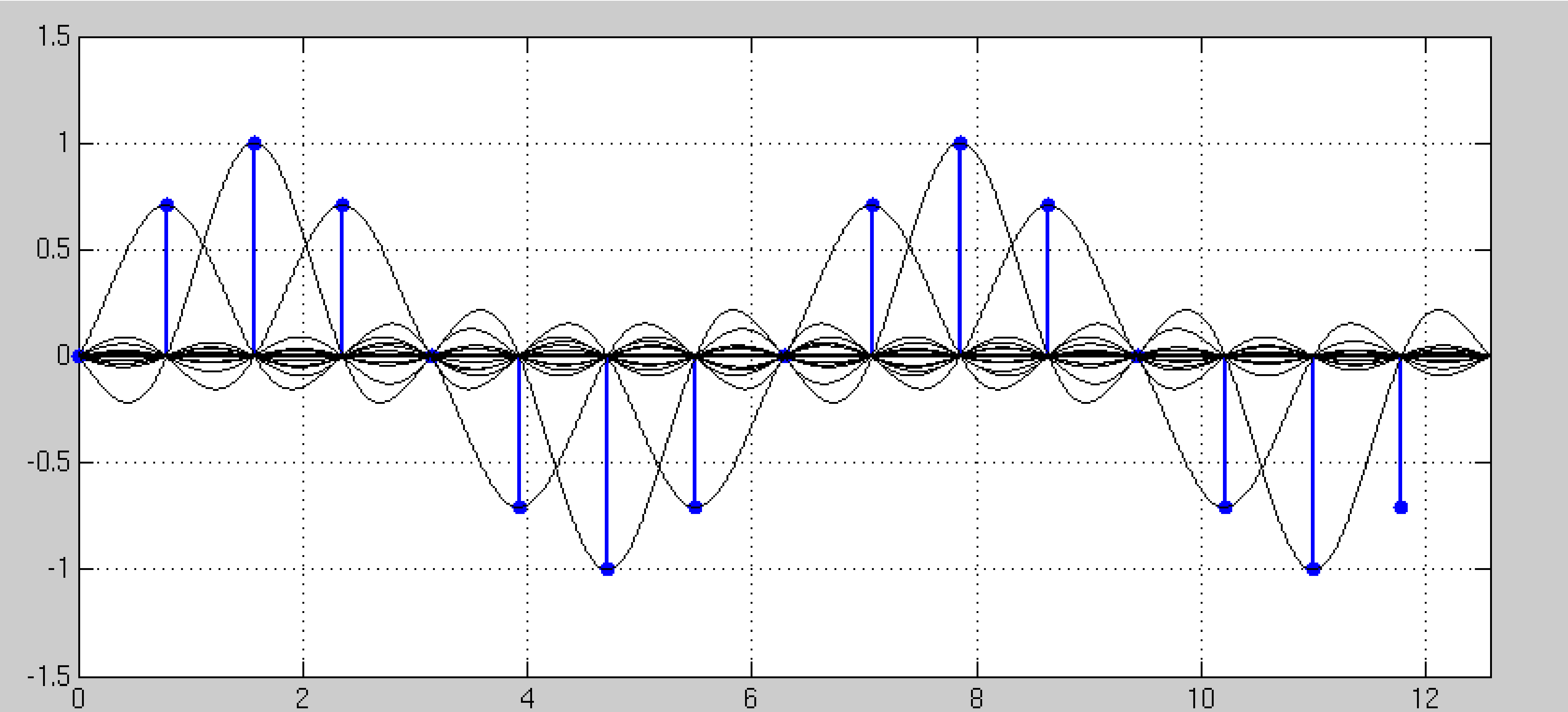
From continuous to discrete and back again



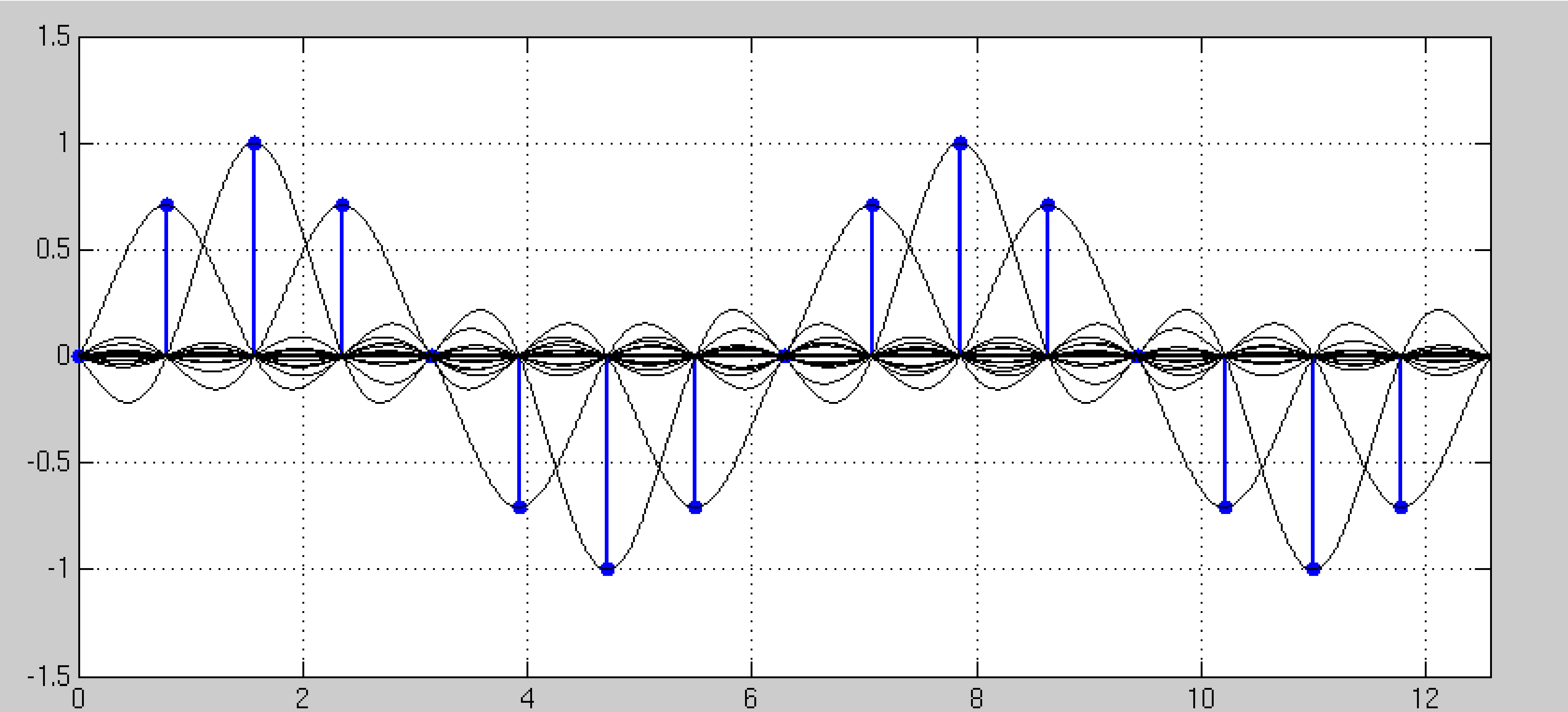
From continuous to discrete and back again



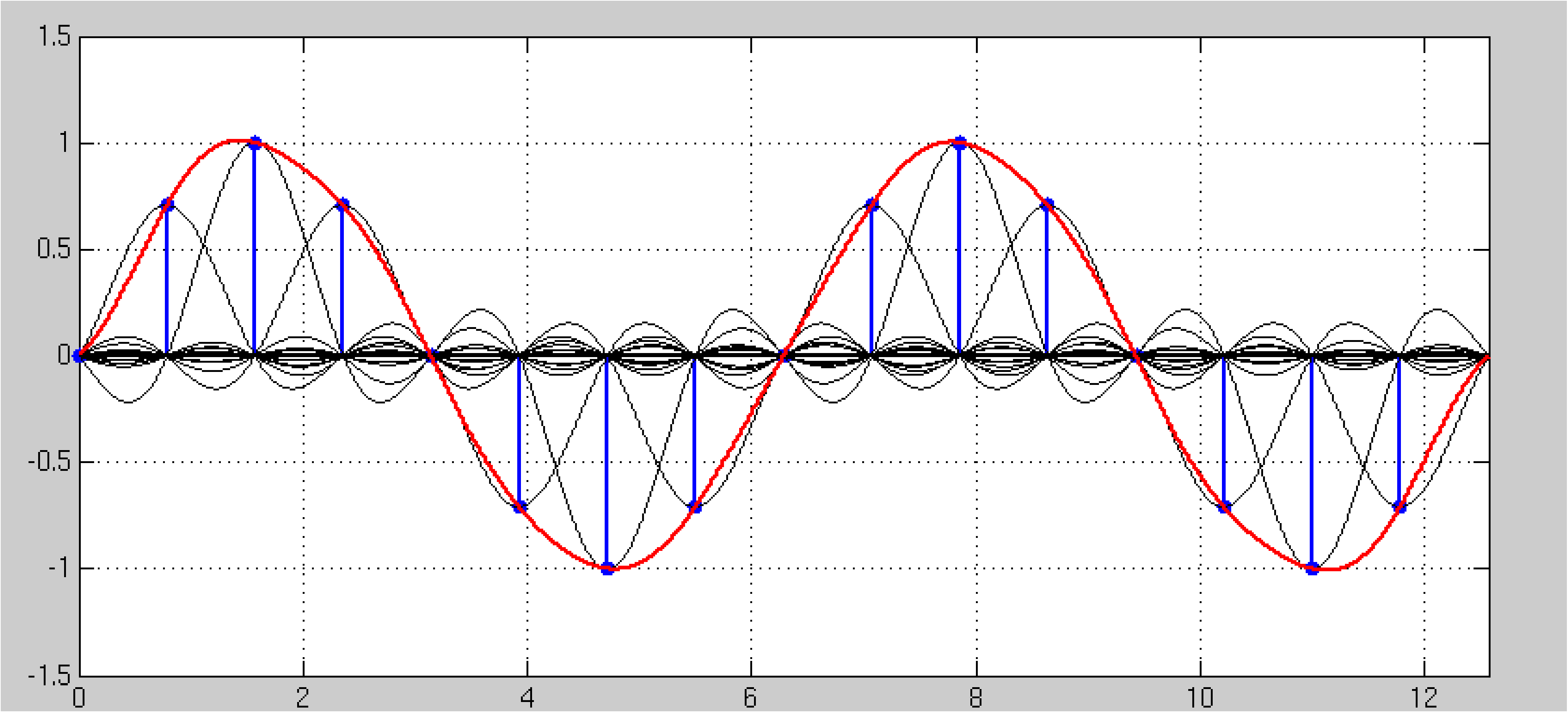
From continuous to discrete and back again



From continuous to discrete and back again

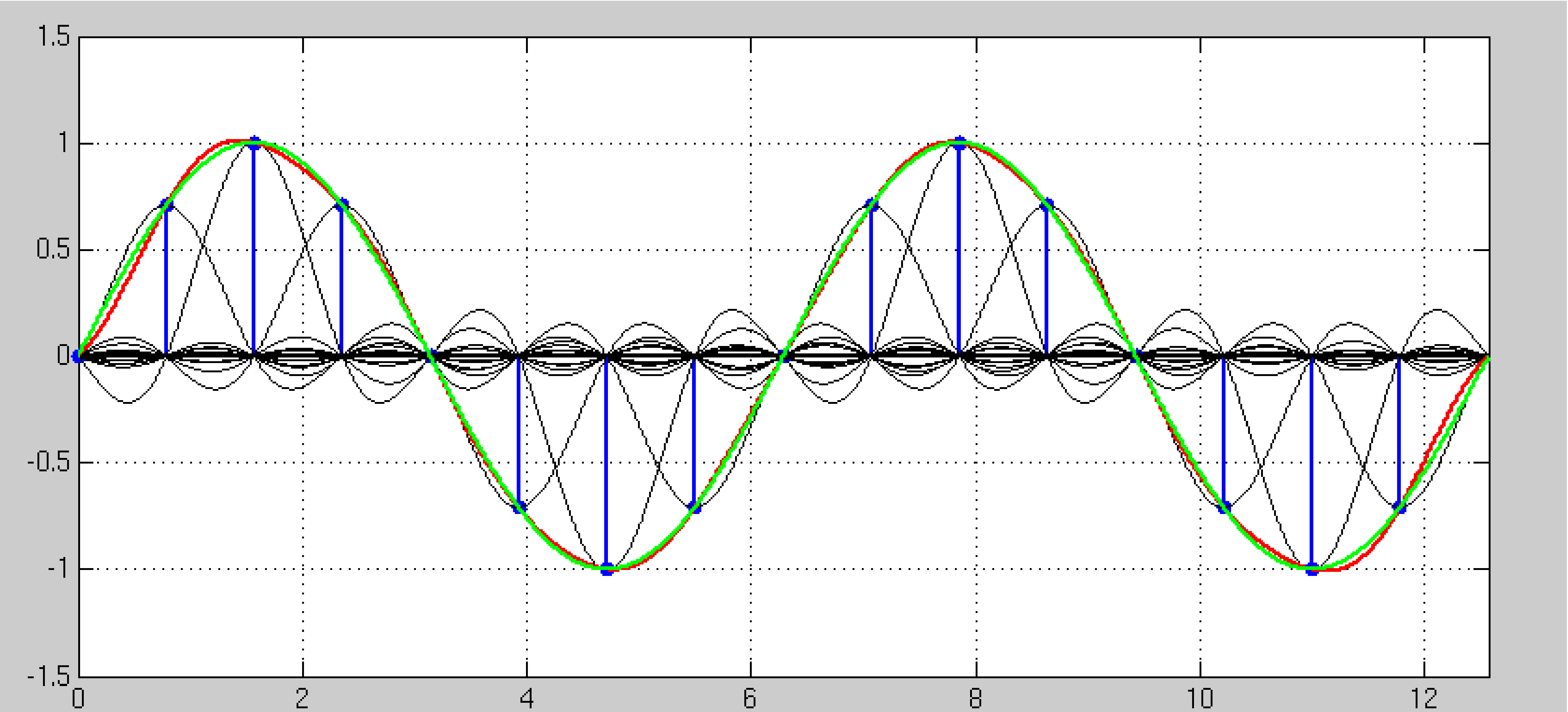


From continuous to discrete and back again



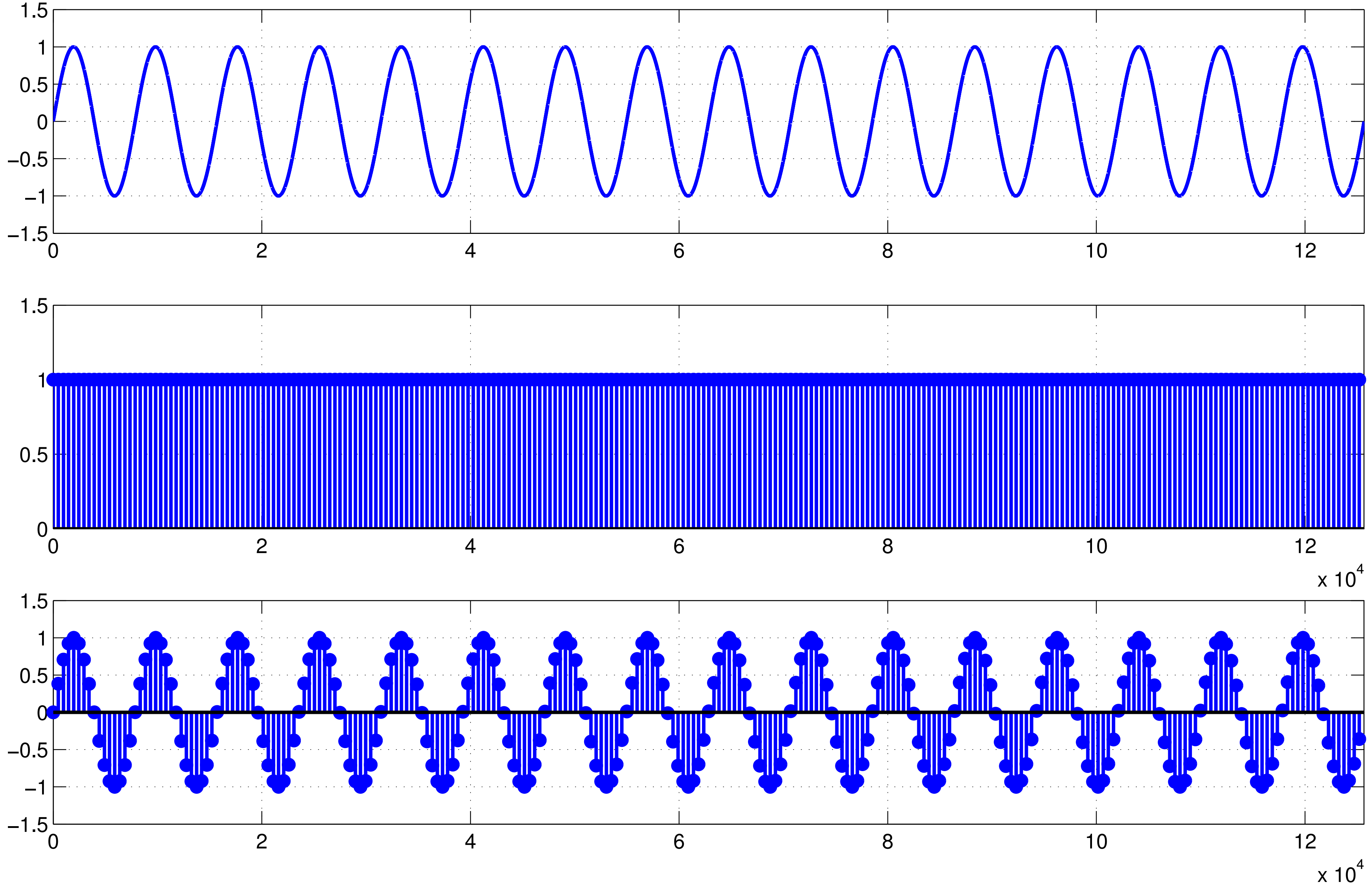
Sum the sinc functions up.

From continuous to discrete and back again

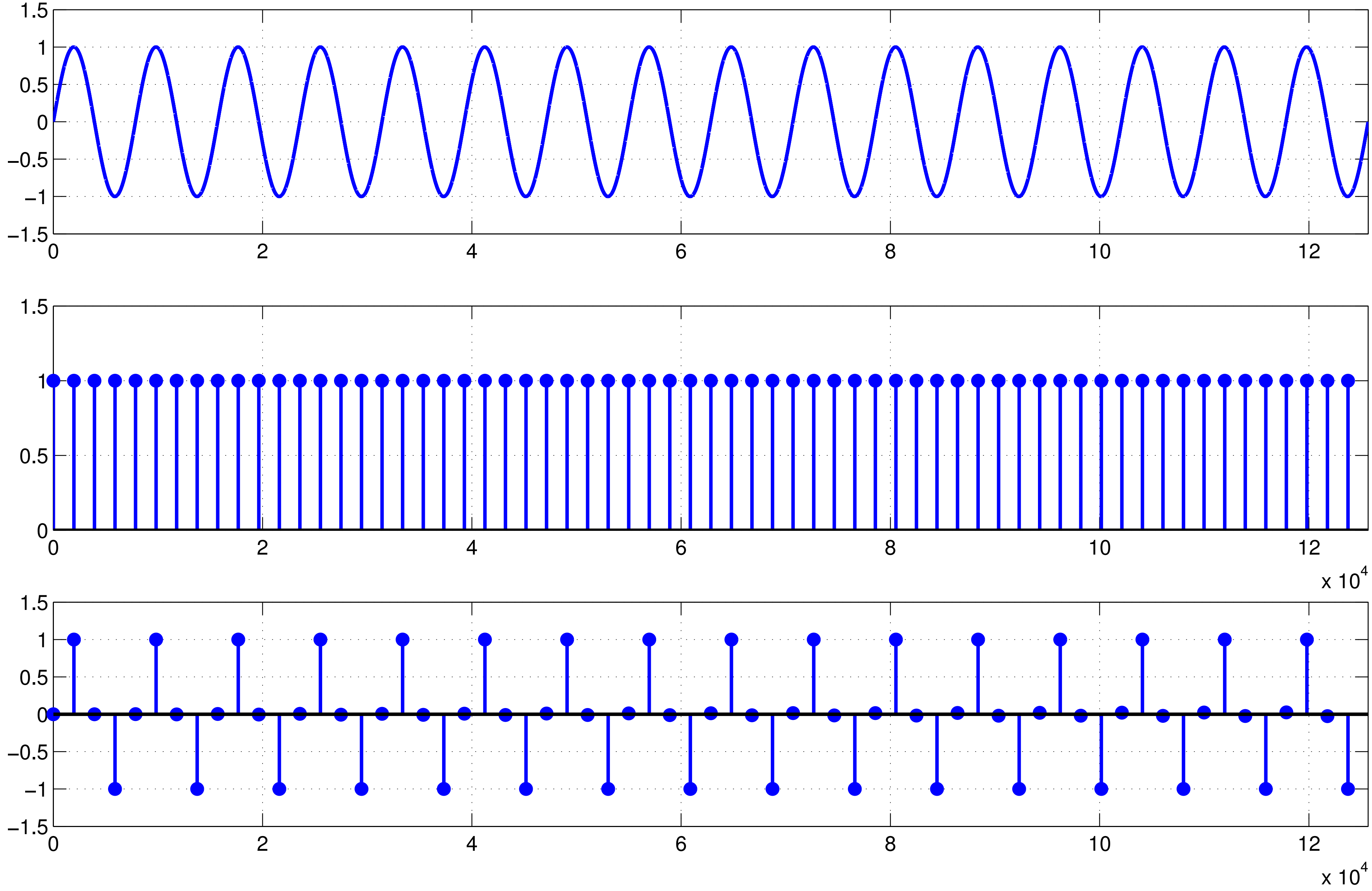


Compare the reconstructed result with the original function.

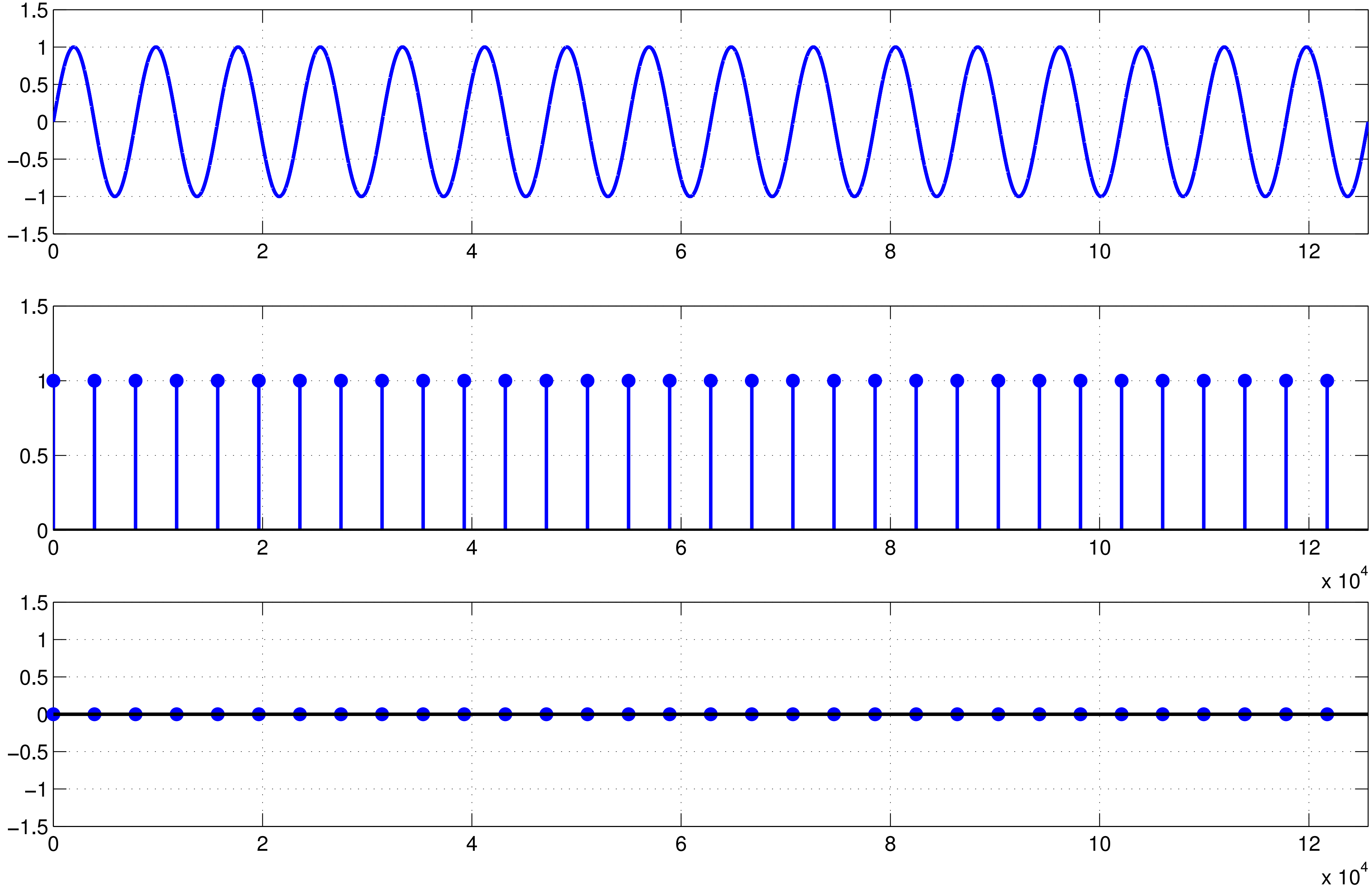
1D sampling $T_{vz} = \frac{T_{max}}{8}$



1D sampling $T_{vz} = \frac{T_{max}}{2}$



1D sampling $T_{vz} = T_{max}$



1D sampling $T_{vz} = 2.1\, T_{max}$

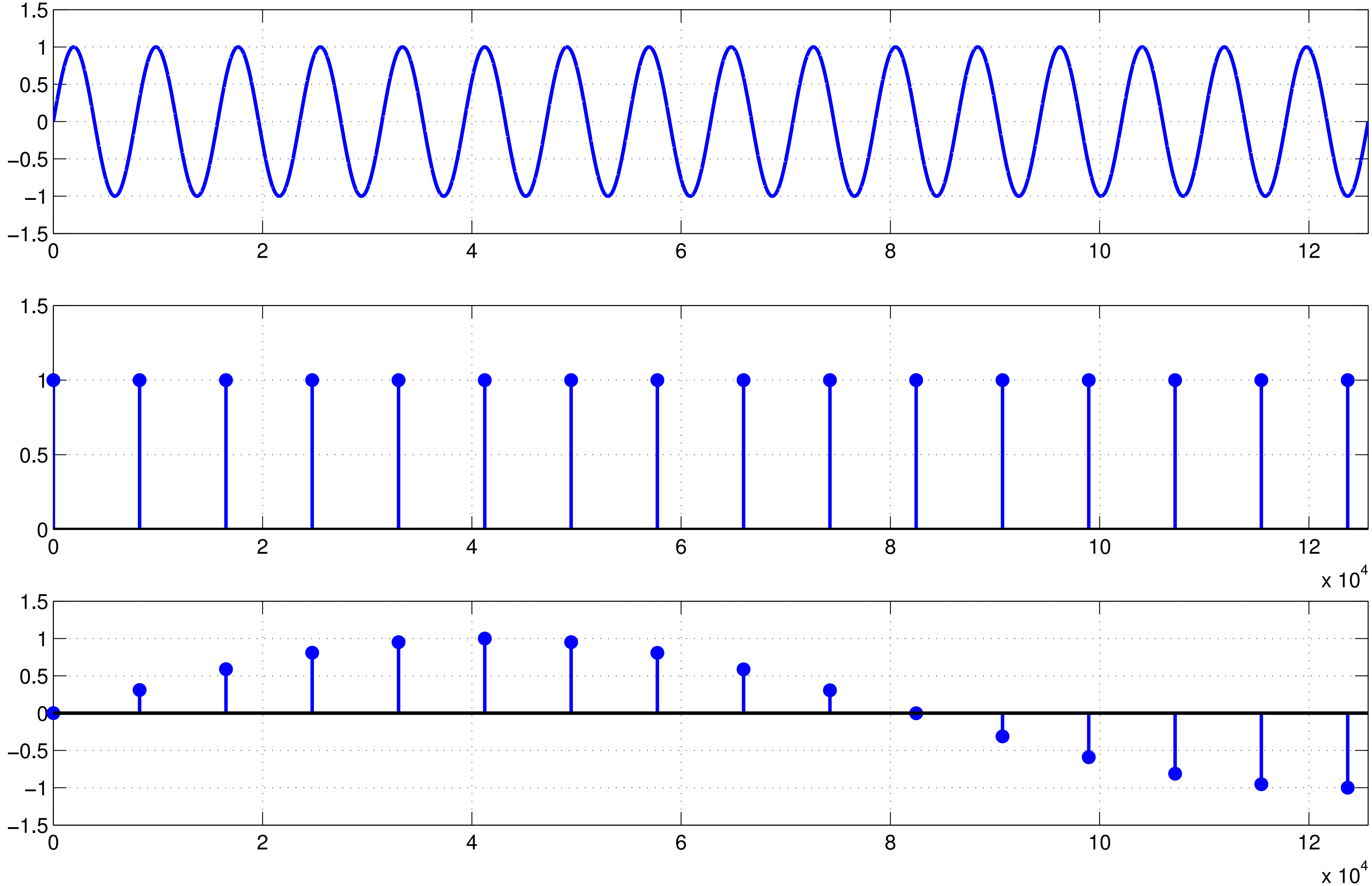
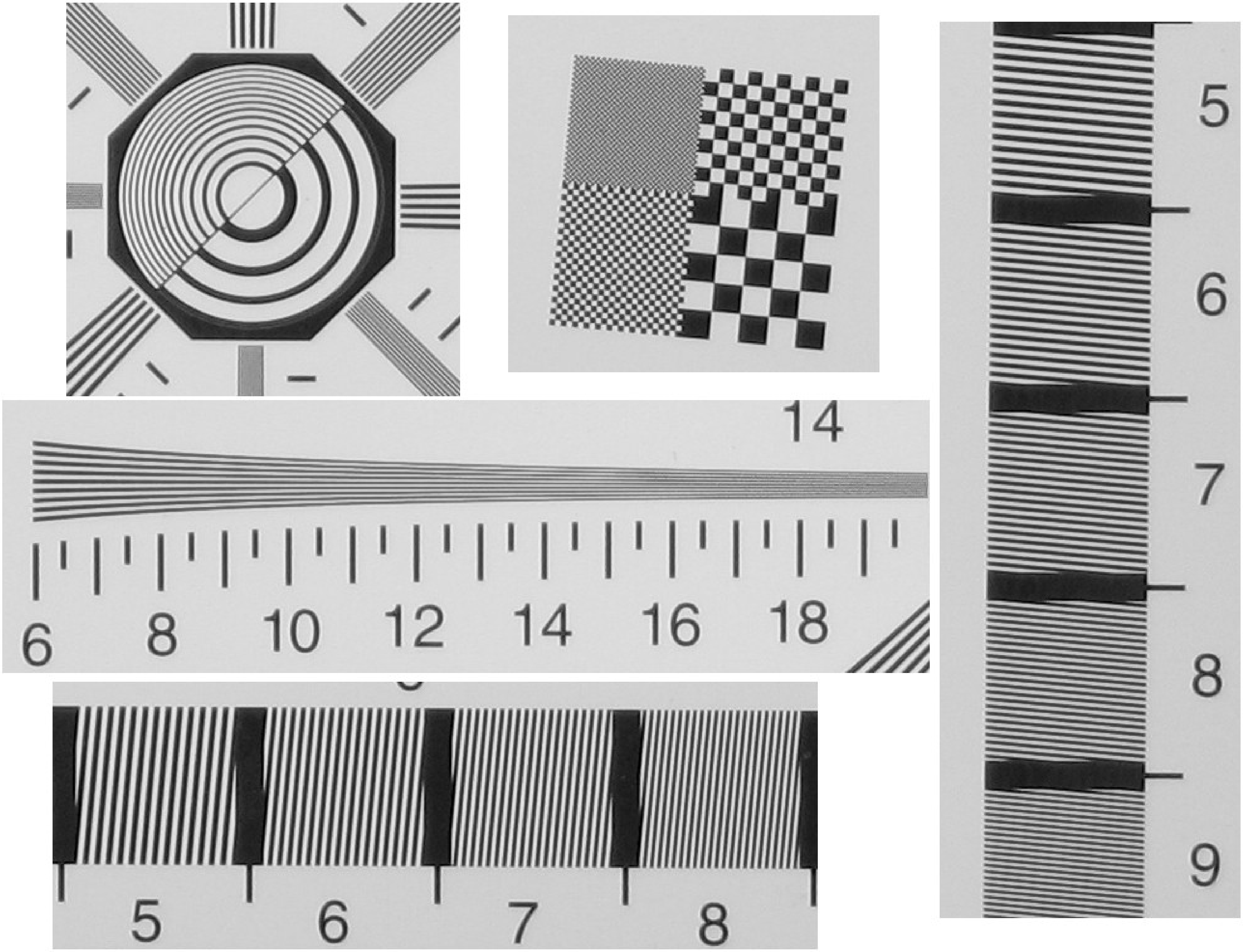
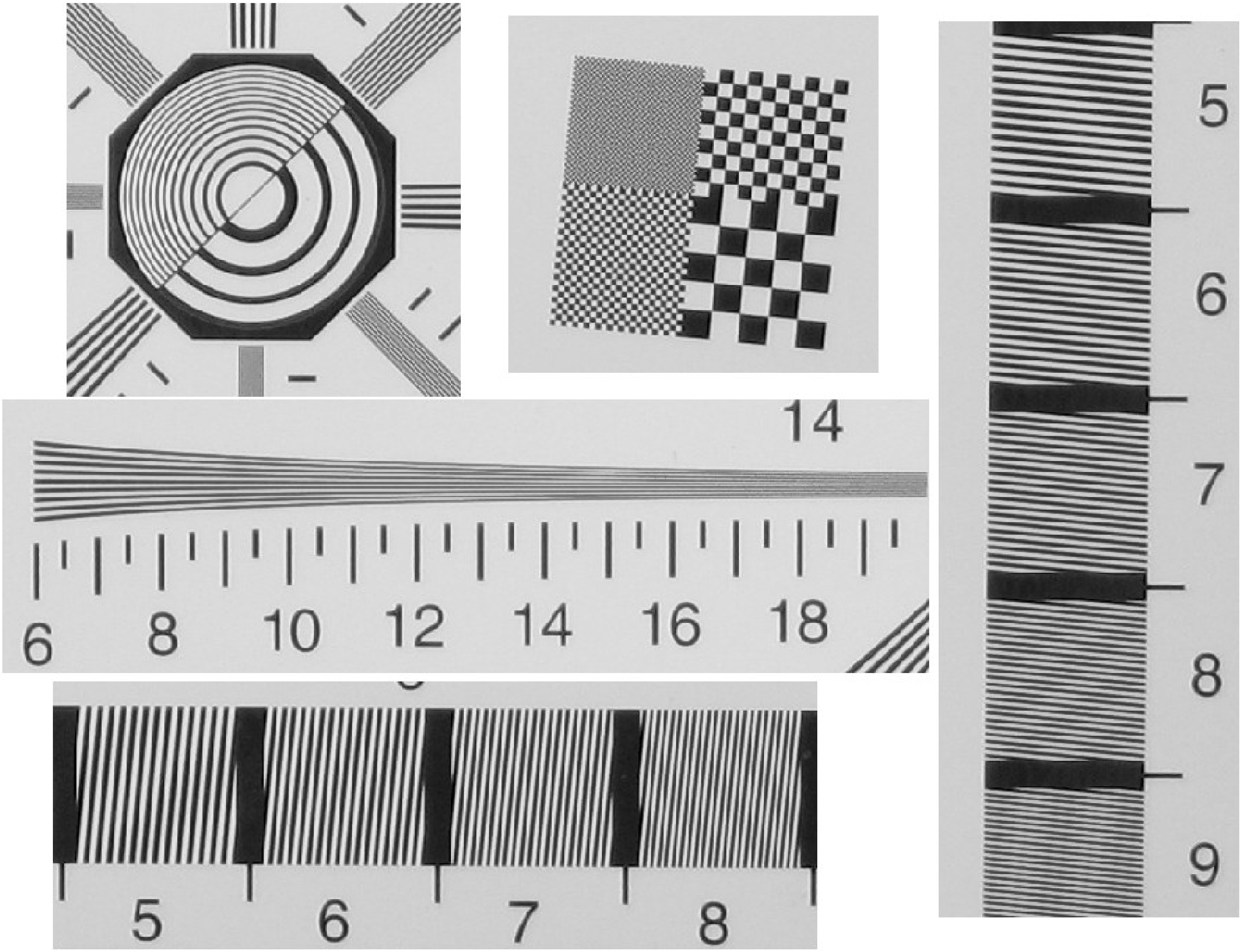


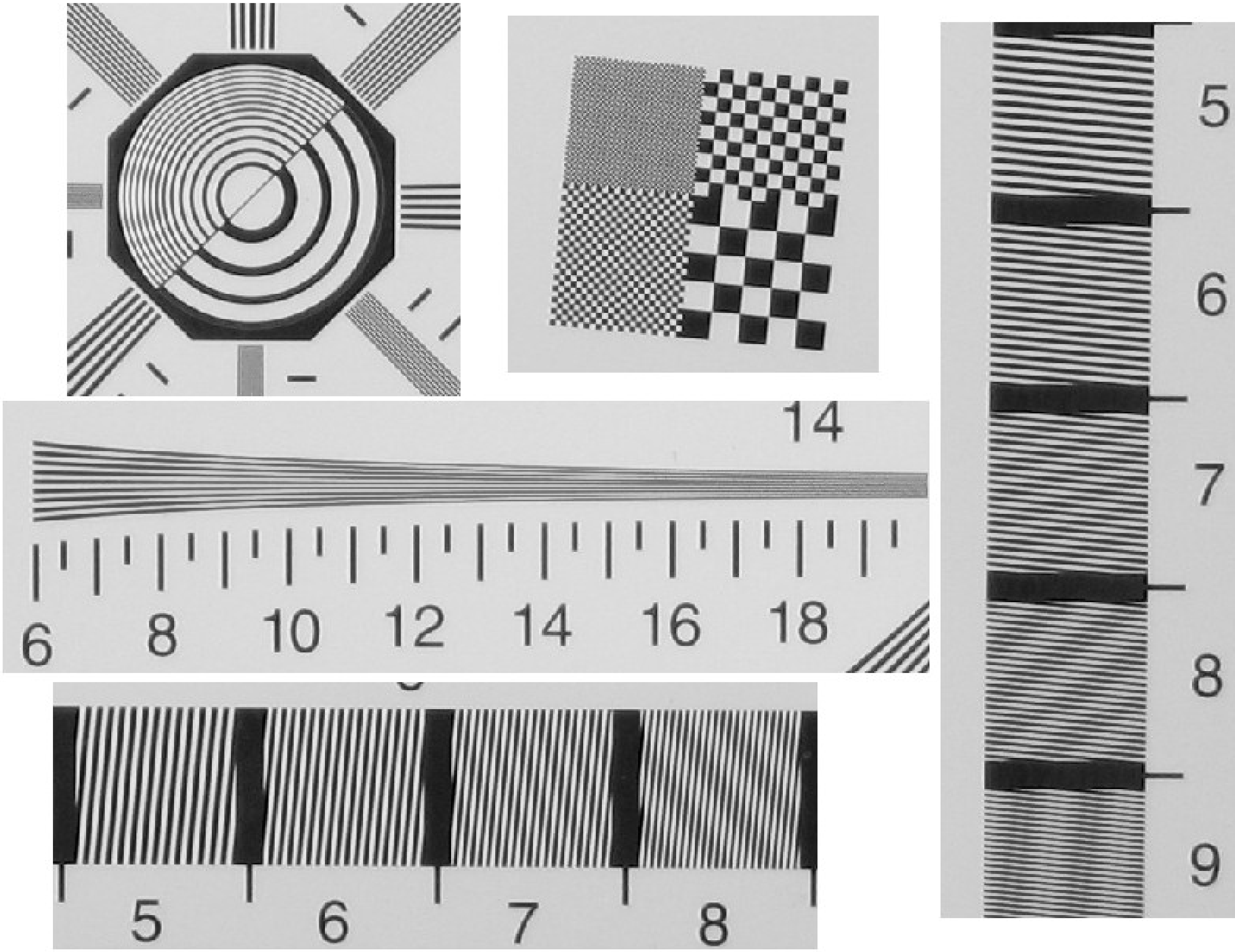
Image aliasing example 100%



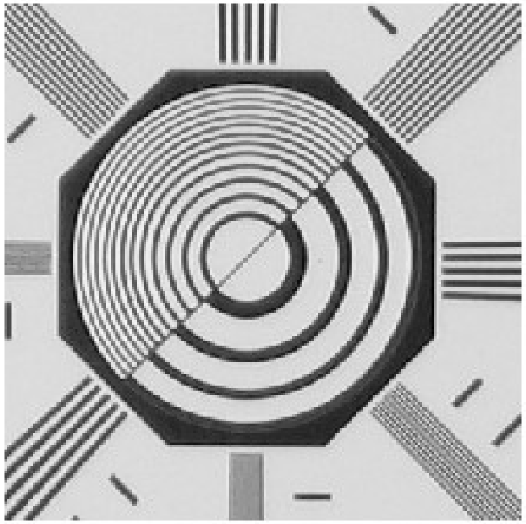
Aliasing example 90%

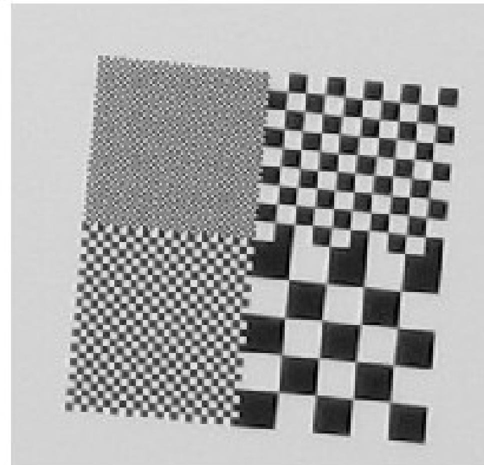


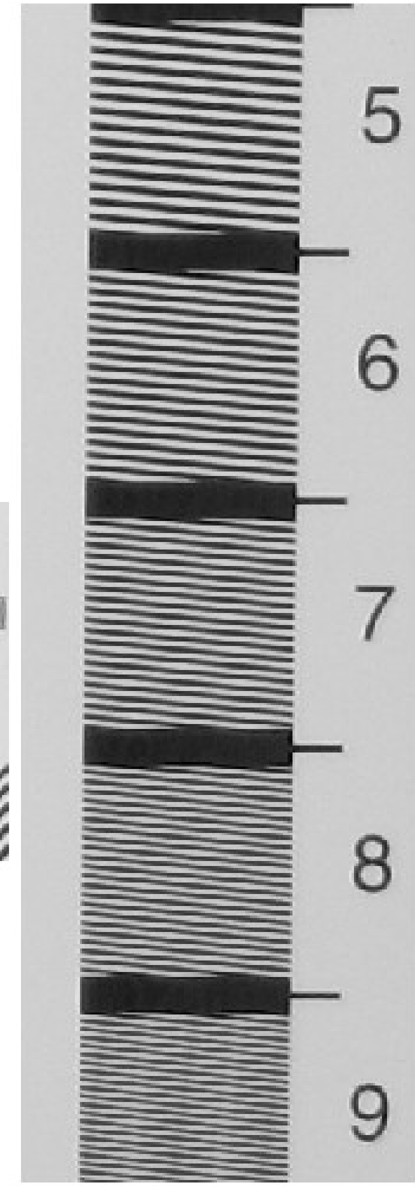
Aliasing example 80%

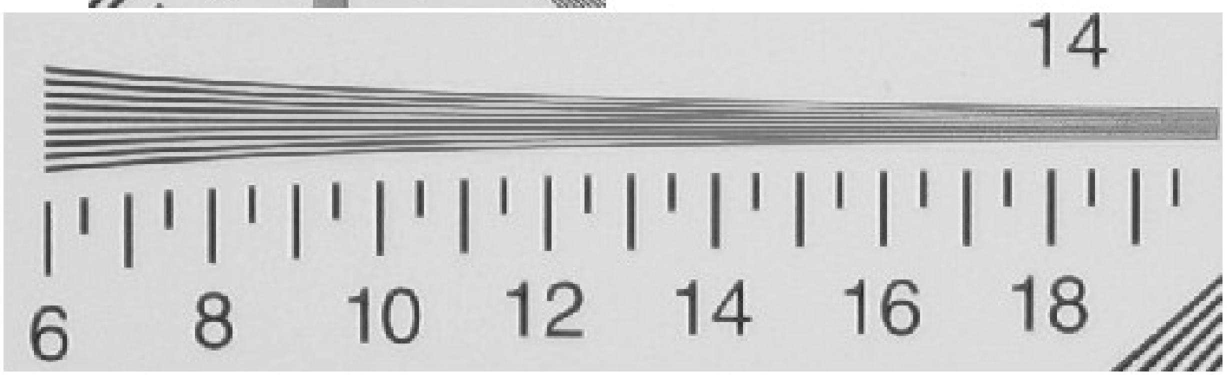


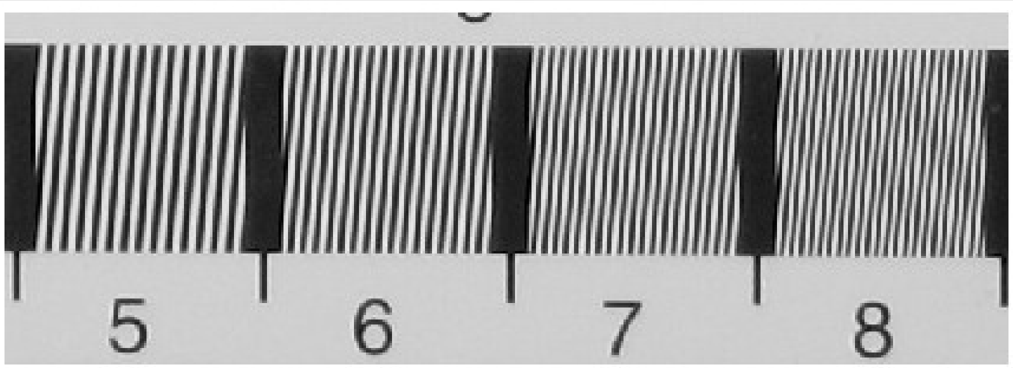
Aliasing example 70%



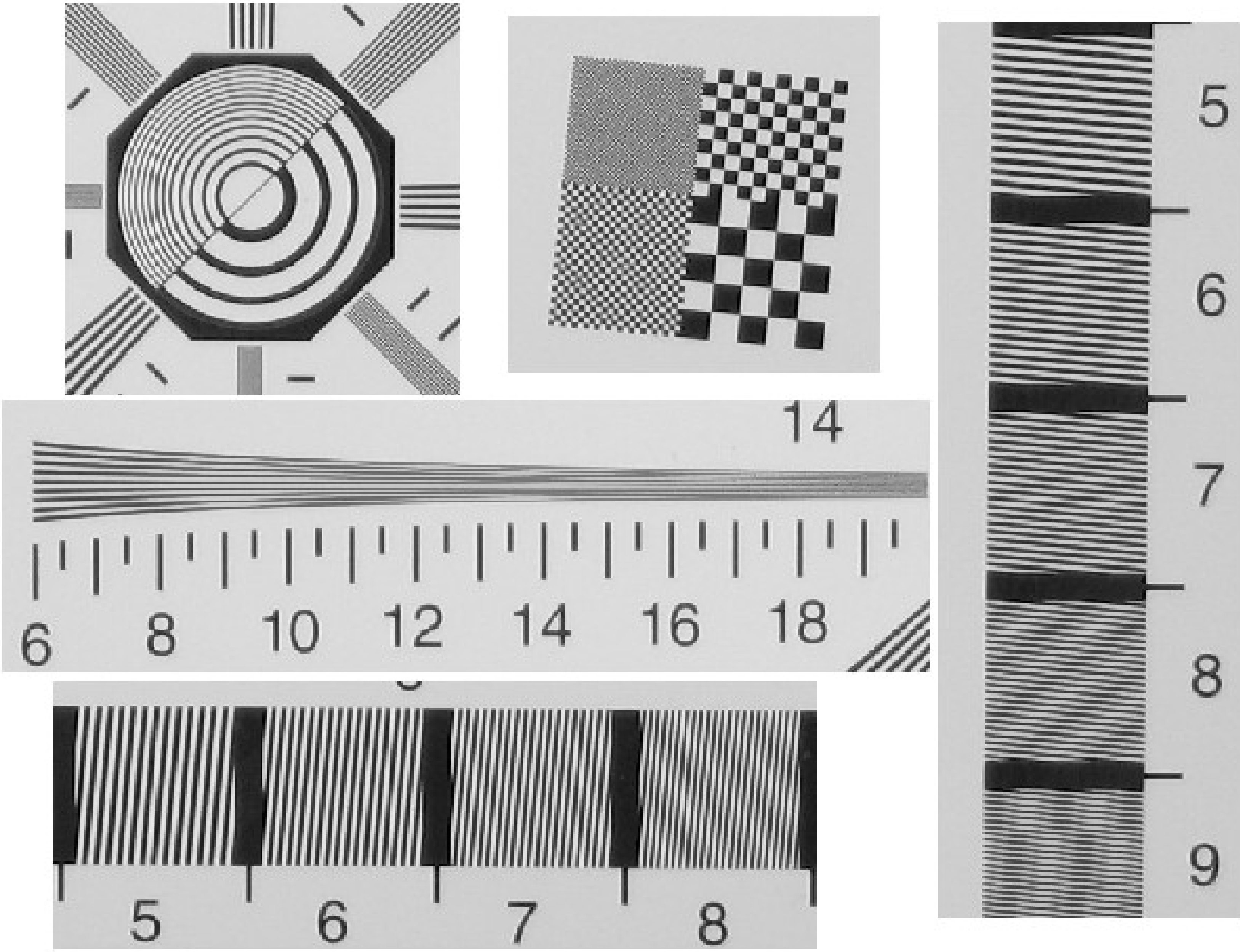




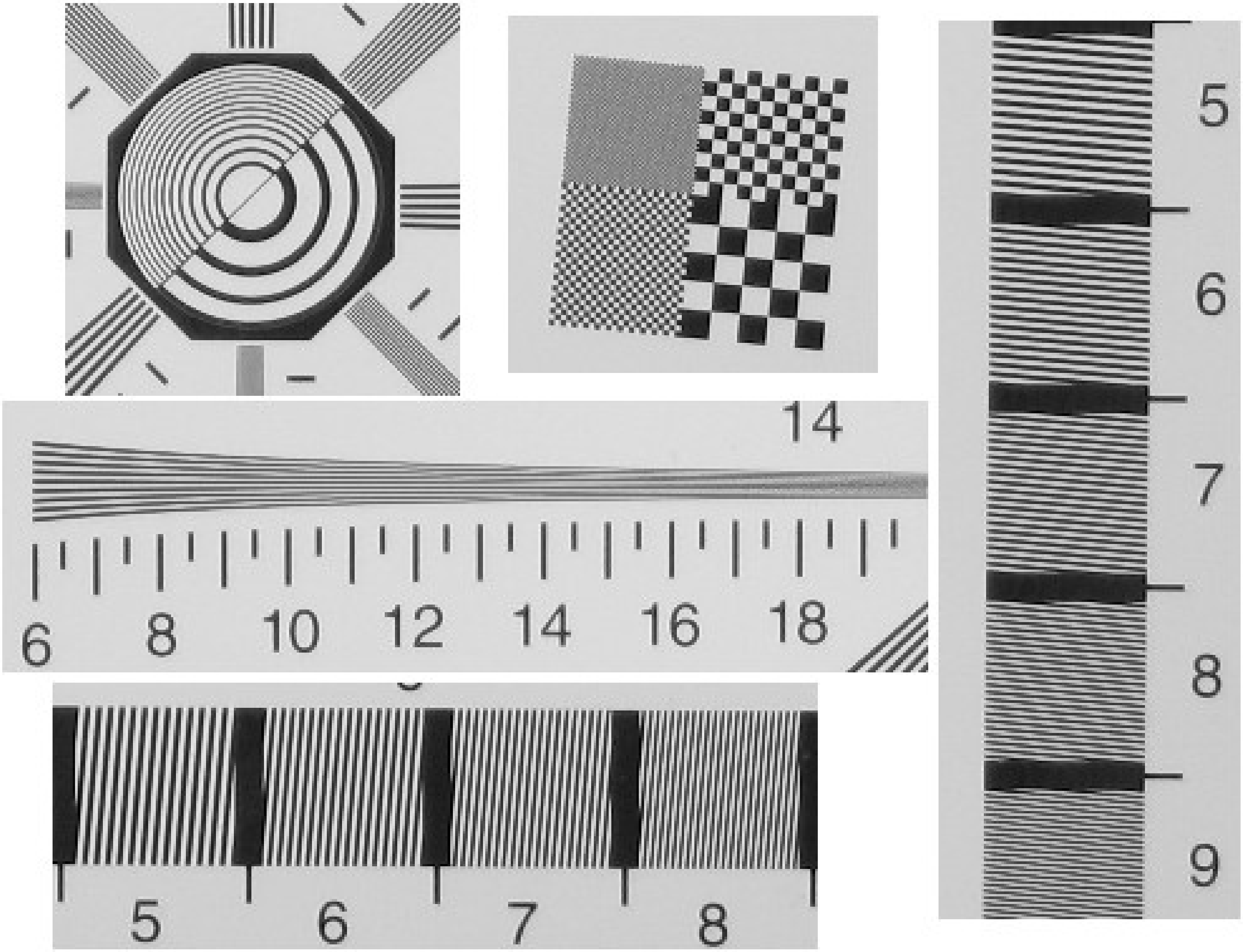




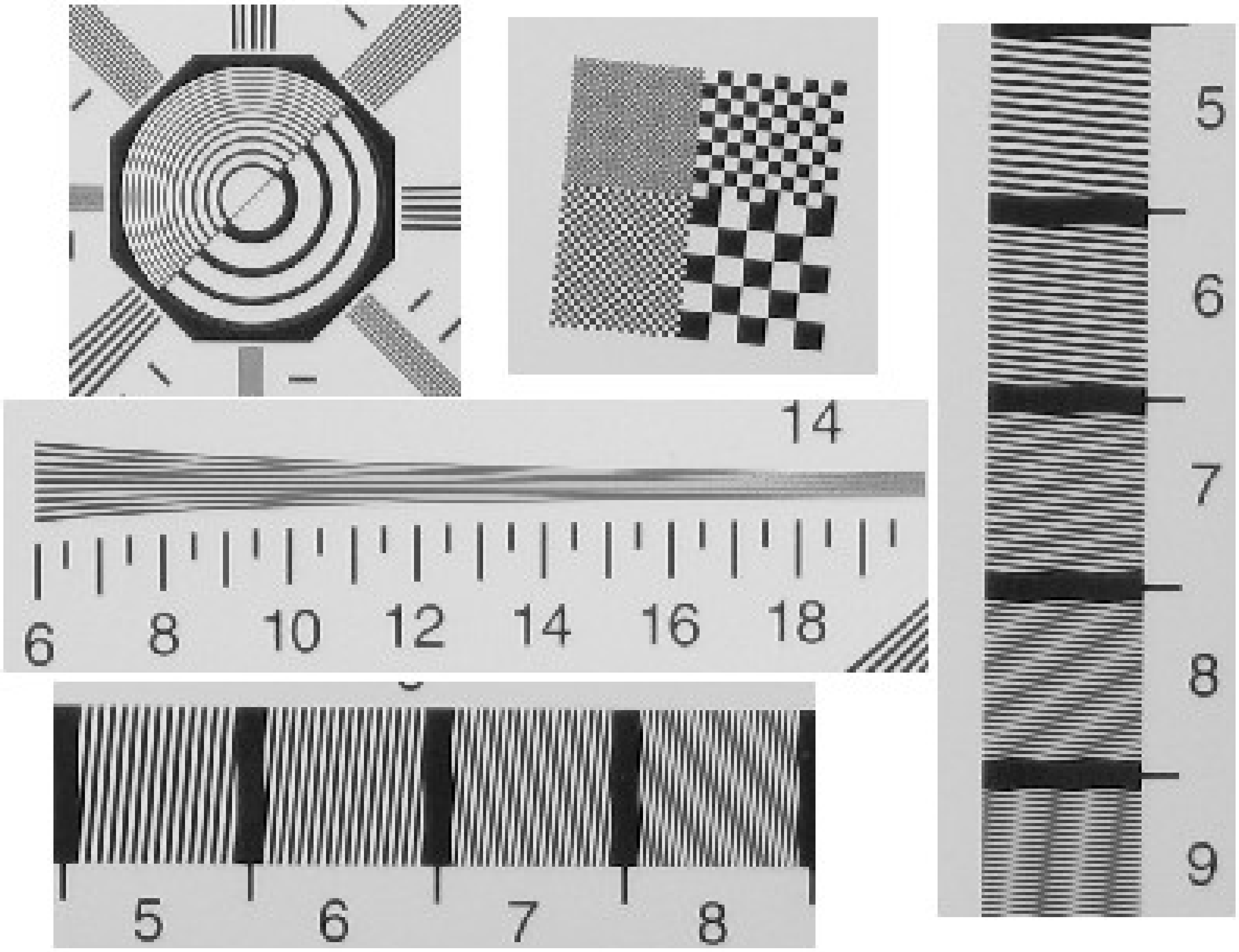
Aliasing example 60%



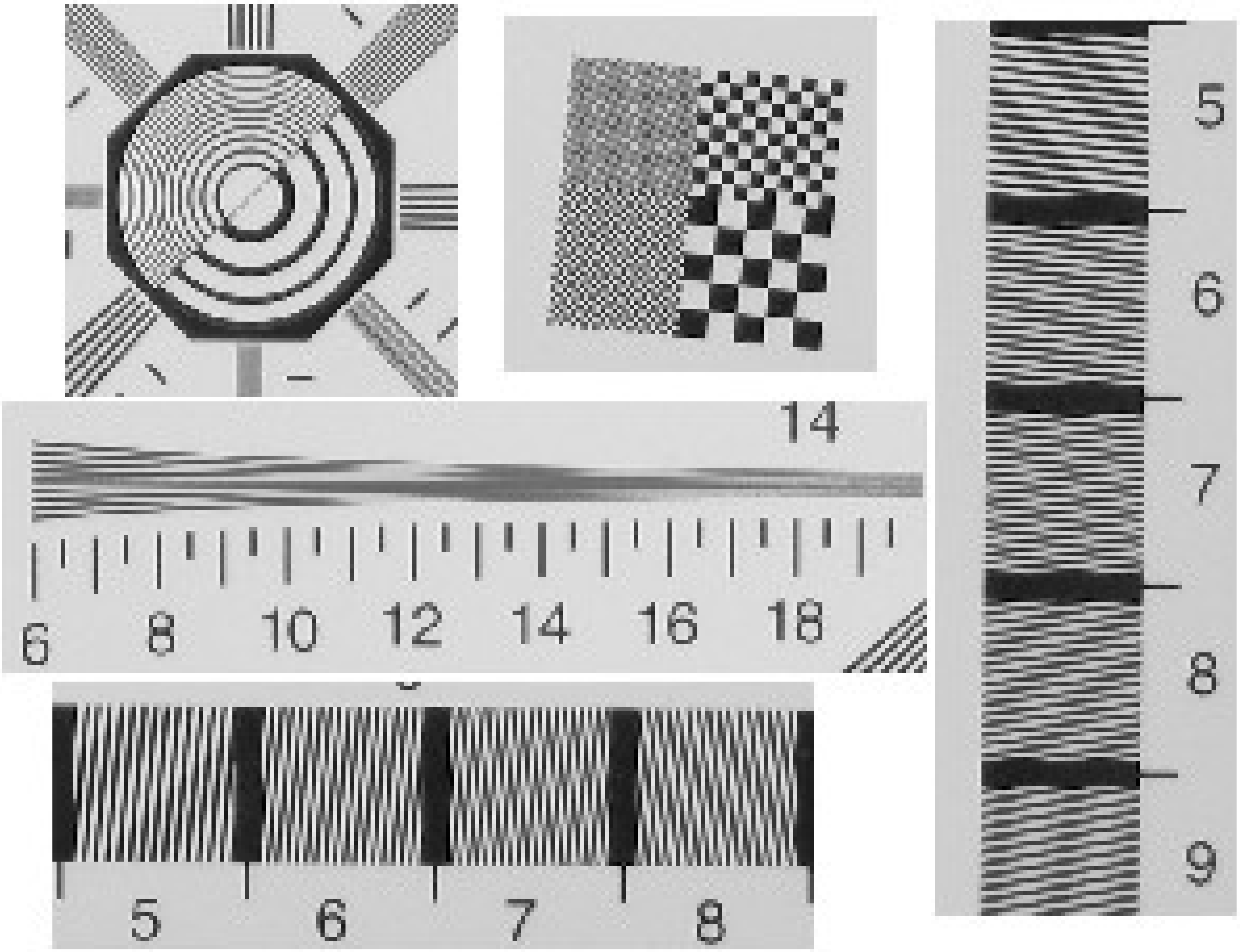
Aliasing example 50%



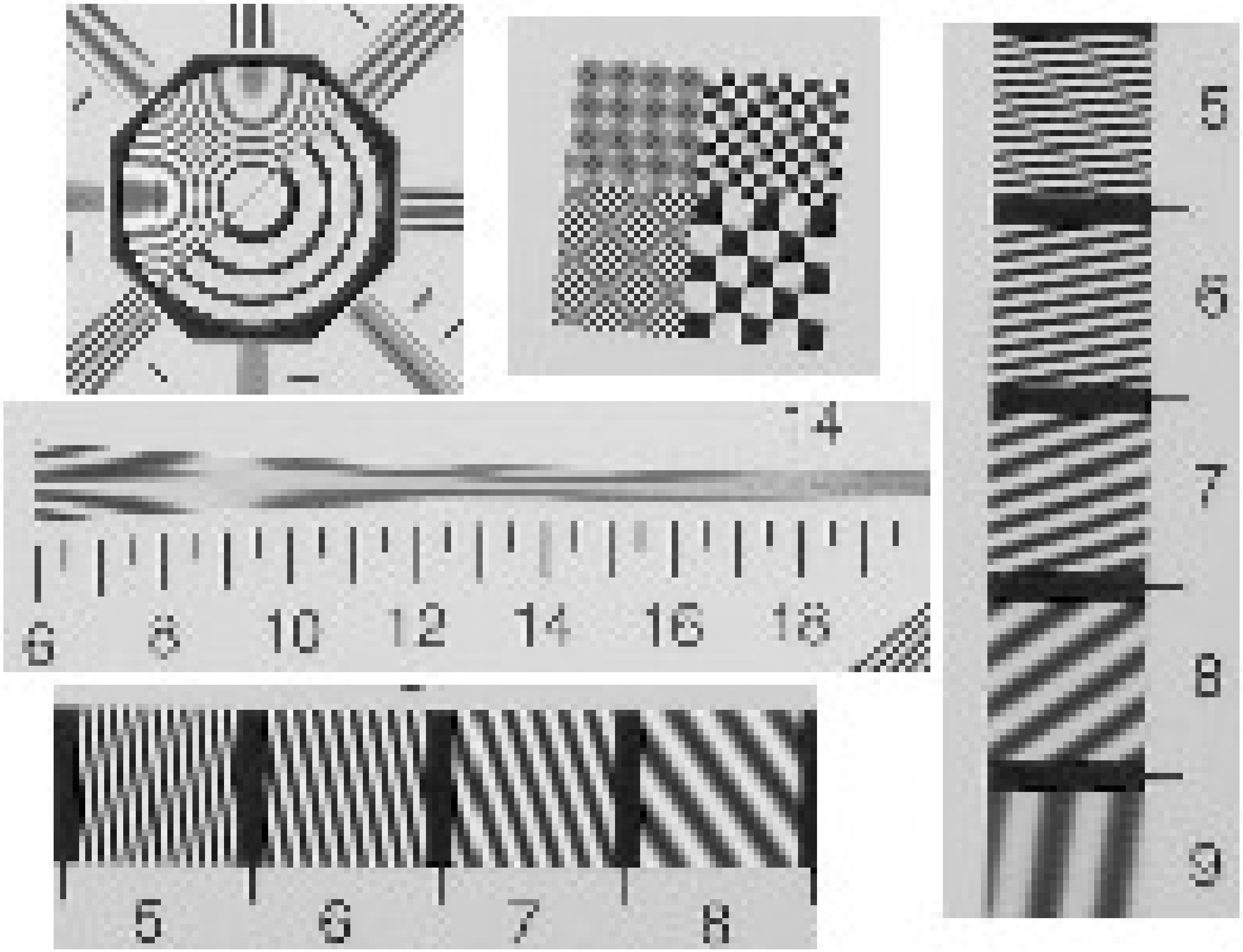
Aliasing example 40%



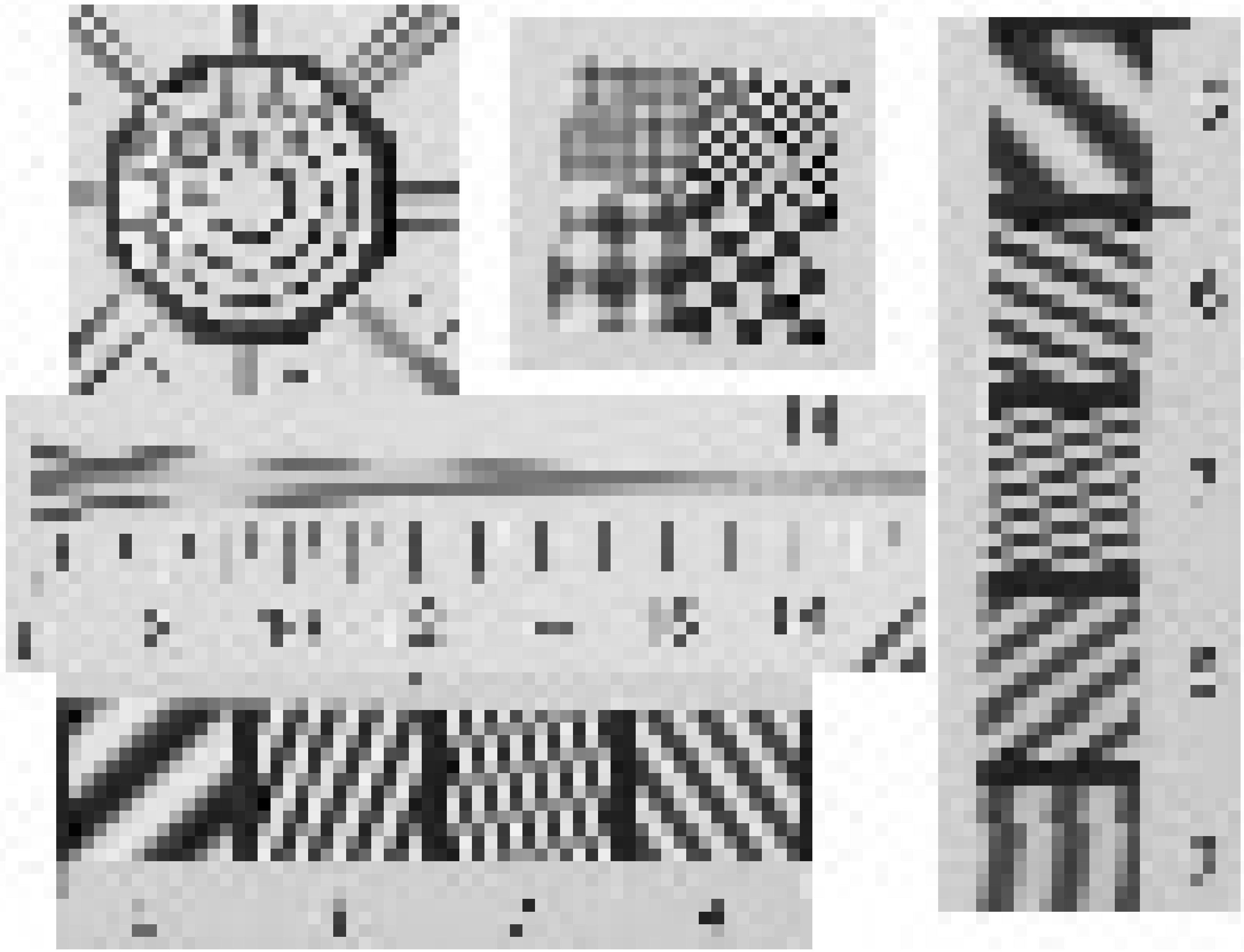
Aliasing example 30%



Aliasing example 20%



Aliasing example 10%



Aliasing — problem in digital photography

Aliasing example 90%

Aliasing example 80%

Aliasing example 70%

Aliasing example 60%

Aliasing example 50%

Aliasing example 40%

Aliasing example 30%

Aliasing example 20%

Aliasing example 10%

What can we do against aliasing?

- Increase sampling frequency $\longrightarrow$ number of pixels (but not only, optics is also important)
- Decrease the frequency $\longrightarrow$ blurring

The same snapshots but the input image blurred (convolved) with 5×5 Gaussian with $\sigma = 2$:

Suppressed Aliasing

Suppressed Aliasing — example 90%

Suppressed Aliasing — example 80%

Suppressed Aliasing — example 70%

Suppressed Aliasing — example 60%

Suppressed Aliasing — example 50%

Suppressed Aliasing — example 40%

Suppressed Aliasing — example 30%

Suppressed Aliasing — example 20%

Suppressed Aliasing — example 10%

Sampling, Aliasing—Revisited

- Sampling is an important issue in images.
- Aliasing is typically not wanted.
- Aliasing is ubiquitous. Toy www example[4]

[4] http://cmp.felk.cvut.cz/cmp/courses/EZS/Demos/Aliasing/

Sampling, Aliasing—Revisited

- Sampling is an important issue in images.
- Aliasing is typically not wanted.
- Aliasing is ubiquitous. Toy www example[4]

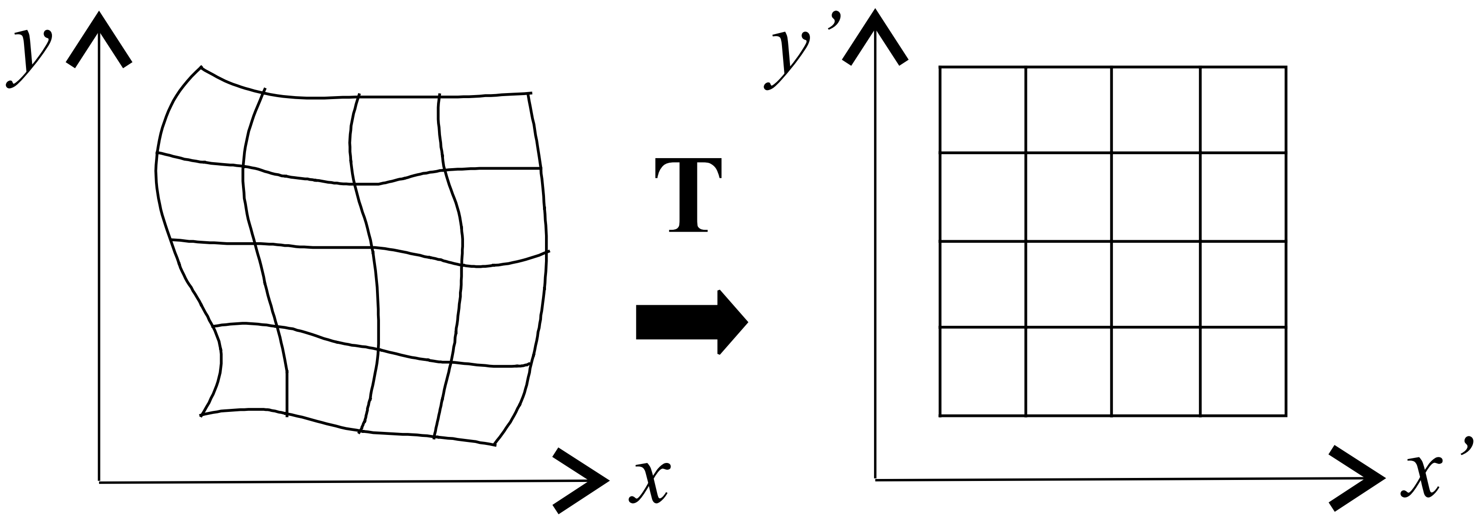
Defeating Aliasing

- Low-pass filter before subsampling
- Subsampling by using image interpolation (will come to that later)

[4] http://cmp.felk.cvut.cz/cmp/courses/EZS/Demos/Aliasing/

Geometrical Transformation

Transformation of spatial coordinates



Geometrical Transformation — What for?

- intentional image transformations (you know where to go)
  - resizing
  - rotation
  - shift
  - warping, texture mapping
- correction of distortions (you know how it should look)
  - projective skew
  - non-linear distortion (fish-eyes)

Techniques shared by Image processing, Computer graphics, even Robotics or Mechanics.

Example of texture mapping

Realization — Rotation and shift

$$
\begin{aligned}
x' &= \cos(\alpha)x + \sin(\alpha)y + t_x \\
y' &= -\sin(\alpha)x + \cos(\alpha)y + t_y
\end{aligned}
$$

More elegant and efficient

$$
\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos(\alpha) & \sin(\alpha) & t_x \\ -\sin(\alpha) & \cos(\alpha) & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}
$$

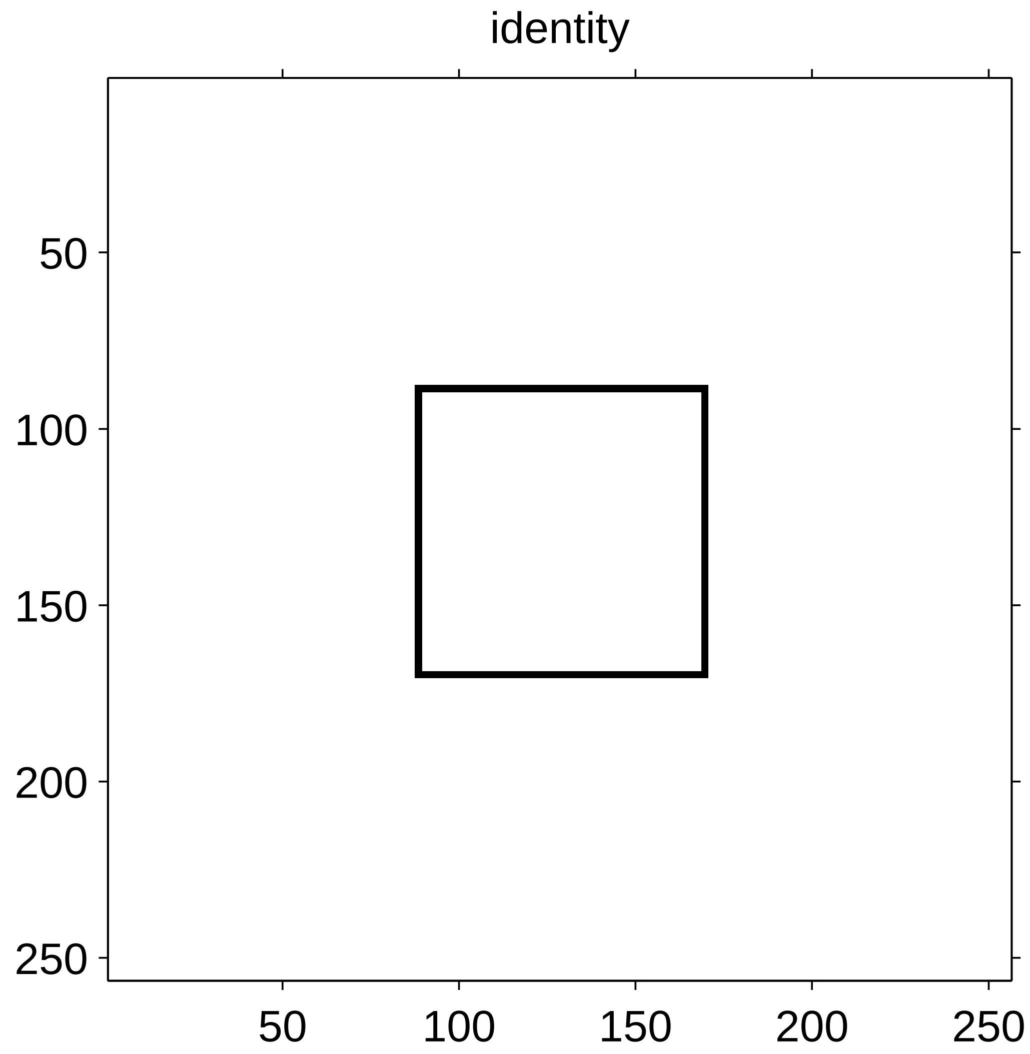
in a matrix form

$$
\mathbf{x}' = T\mathbf{x}
$$

where $\mathbf{x}'$ and $\mathbf{x}$ are homogeneous coordinates: $\mathbf{x} = [\lambda x, \lambda y, \lambda]^T$, $\lambda \neq 0$.

Identity

$$T = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

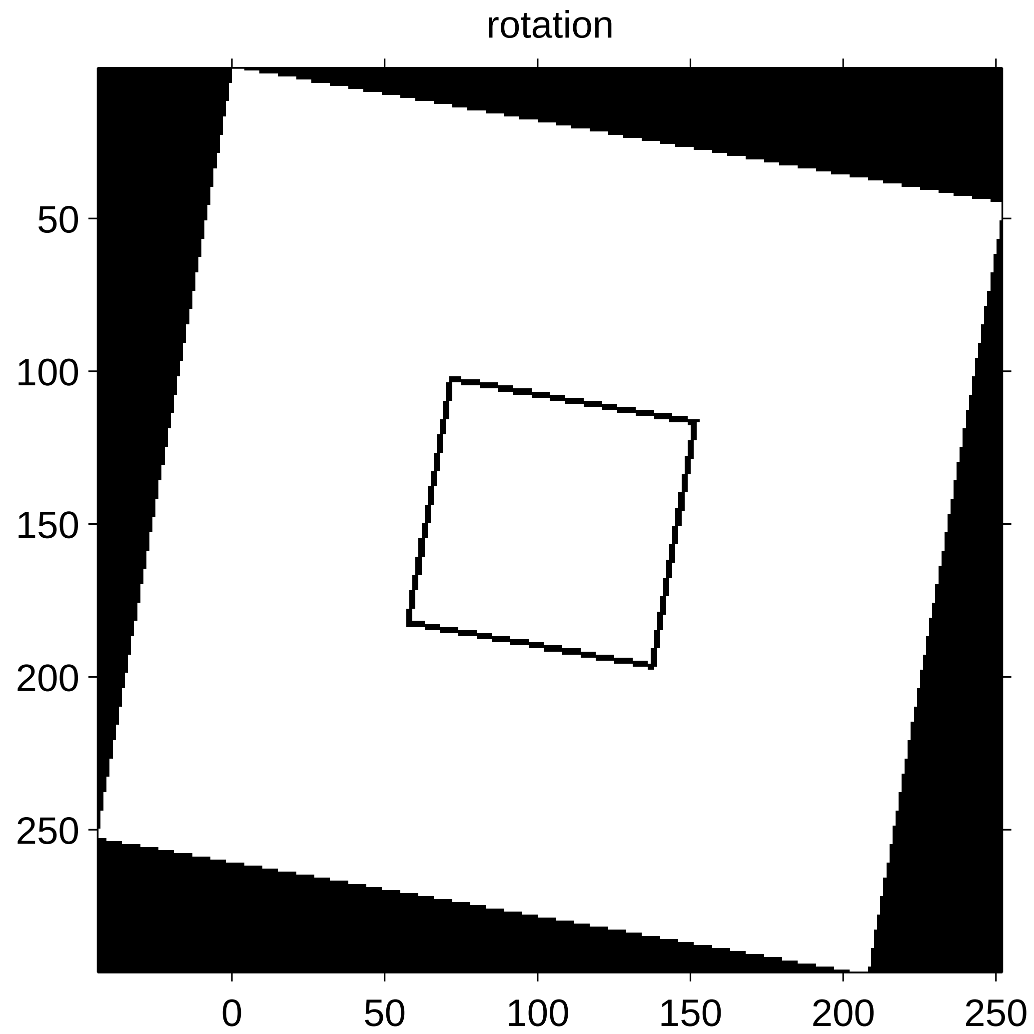


Rotation

$$T = \begin{bmatrix} \cos(\alpha) & \sin(\alpha) & 0 \\ -\sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

for $\alpha = 10^\circ$

$$T = \begin{bmatrix} 0.9848 & 0.1736 & 0 \\ -0.1736 & 0.9848 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

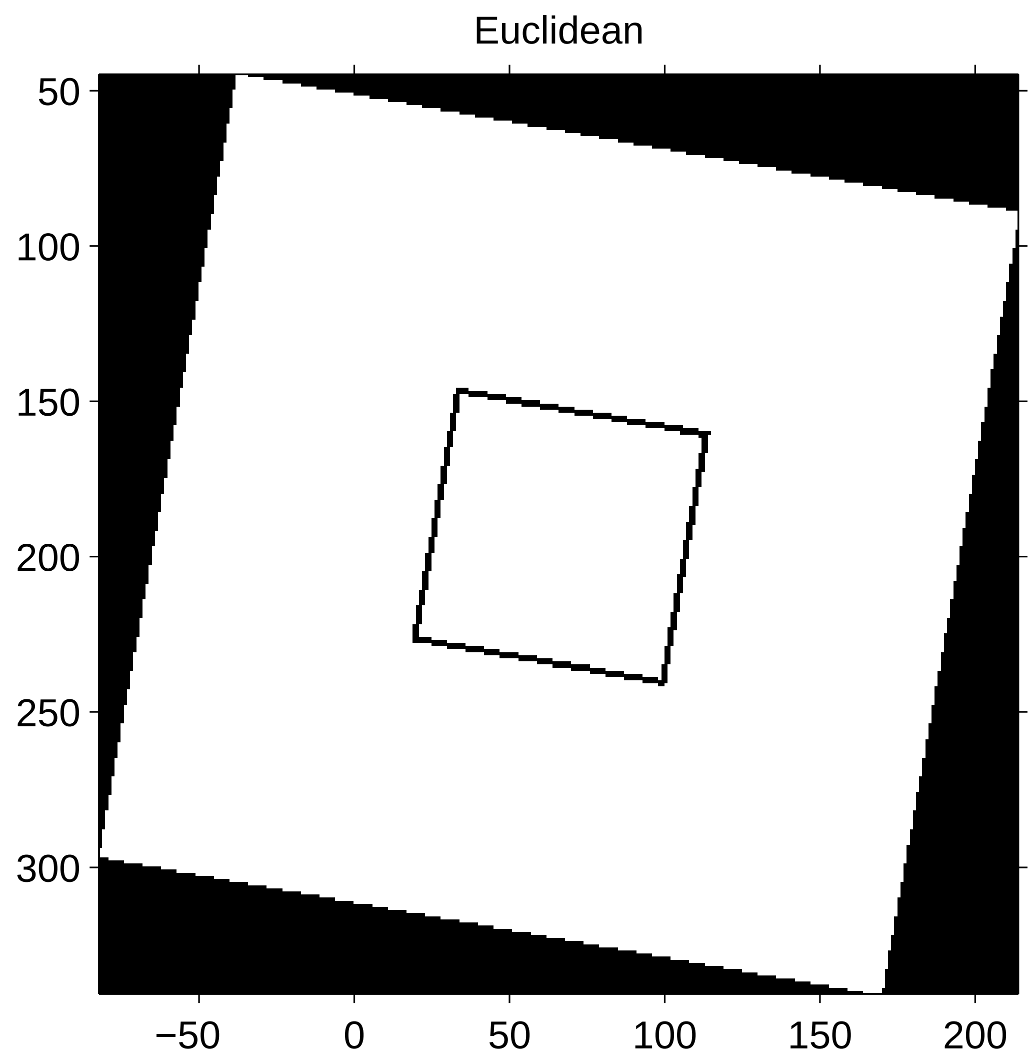


Rotation + translation

$$T = \begin{bmatrix} \cos(\alpha) & \sin(\alpha) & t_x \\ -\sin(\alpha) & \cos(\alpha) & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

for $\alpha = 10^\circ$ and $\mathbf{t} = [-50, 30]^T$

$$T = \begin{bmatrix} 0.9848 & 0.1736 & -50 \\ -0.1736 & 0.9848 & 30 \\ 0 & 0 & 1 \end{bmatrix}$$



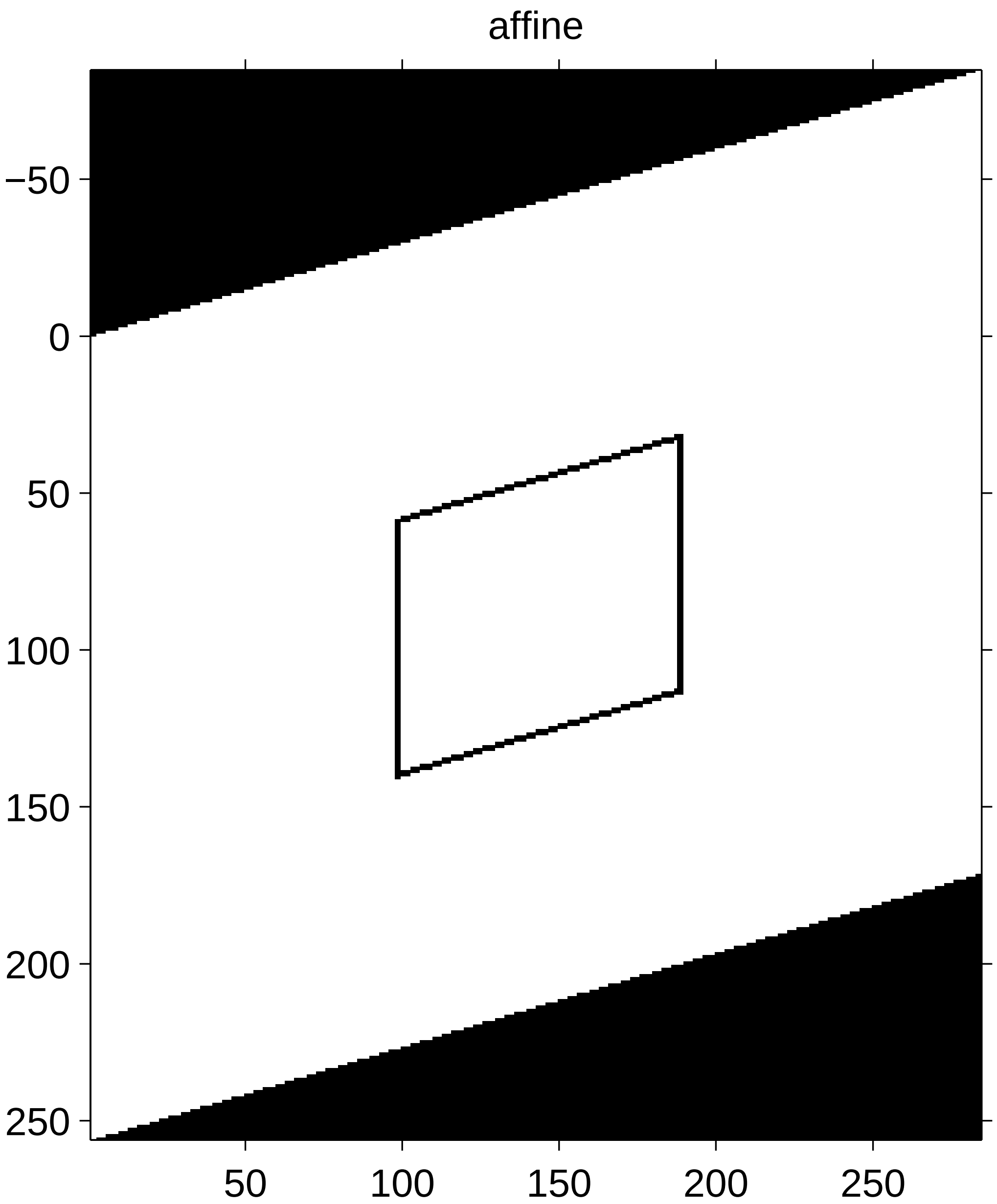
Affine

$$T = \begin{bmatrix} 0.9000 & 0 & 0 \\ 0.3000 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

in general

$$T = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix}$$

6 degrees of freedom



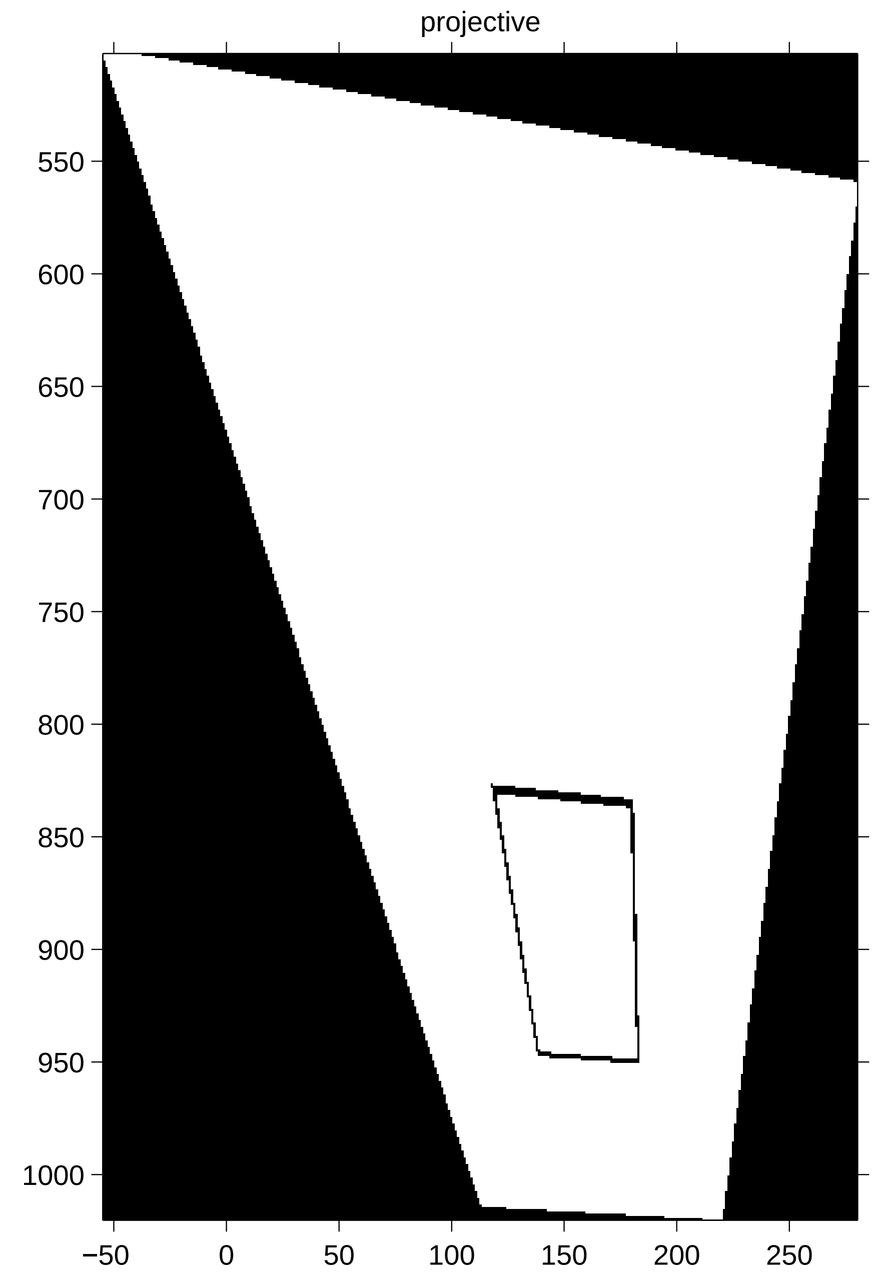
Projective

$$T = \begin{bmatrix} 0.445 & -0.147 & 98.400 \\ -0.018 & 0.099 & -50.000 \\ -0.000 & -0.001 & 1 \end{bmatrix}$$

in general

$$T = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & 1 \end{bmatrix}$$

8 degrees of freedom



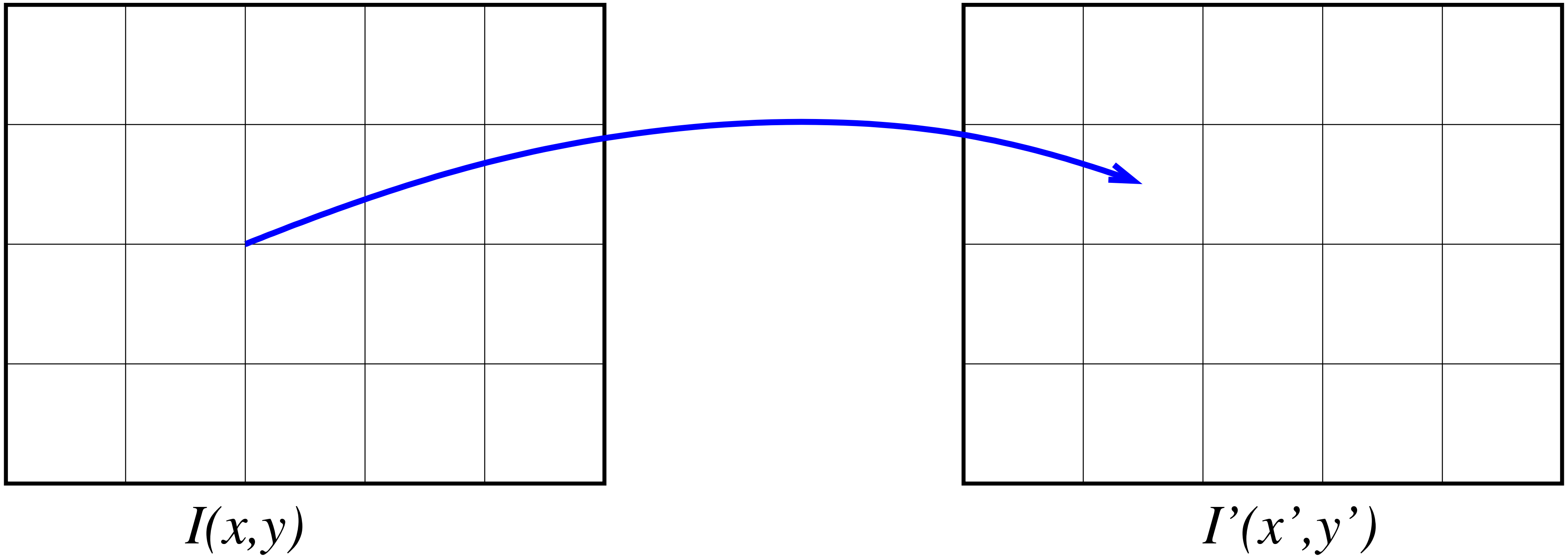
Correction of converging lines I

Correction of converging lines II

Correction of converging lines III

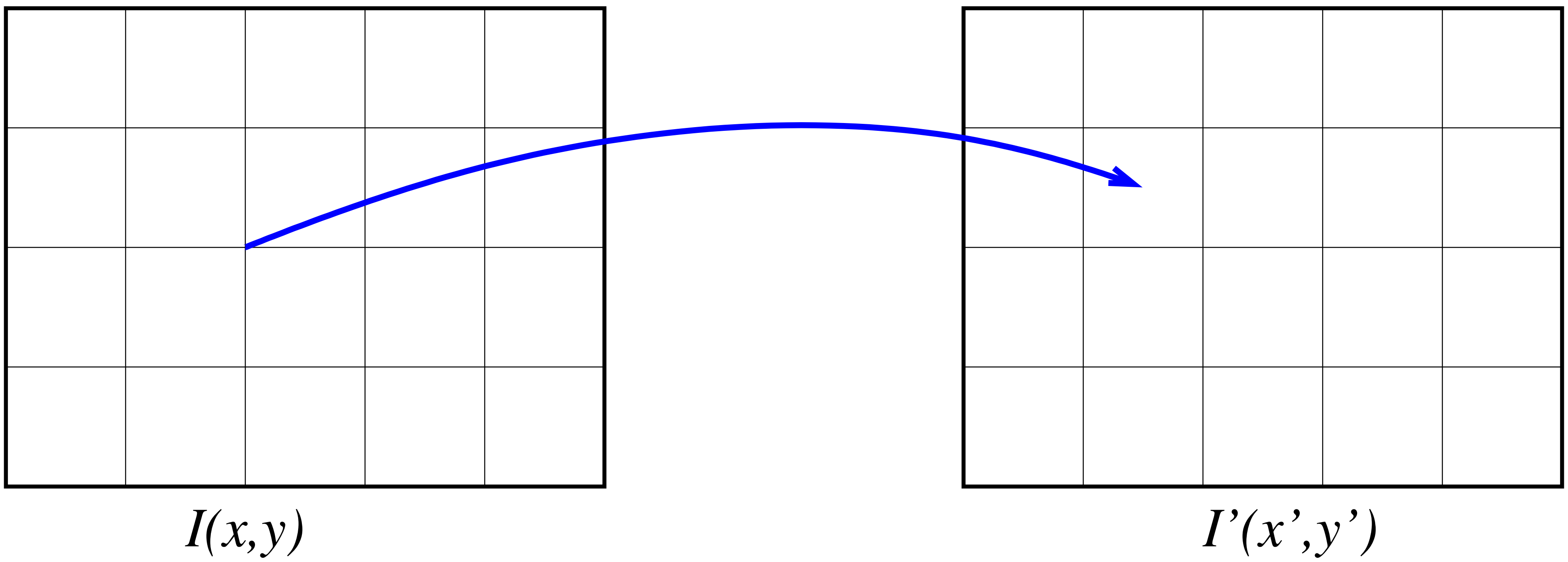
Forward mapping

$\mathbf{x}' = T\mathbf{x}$;



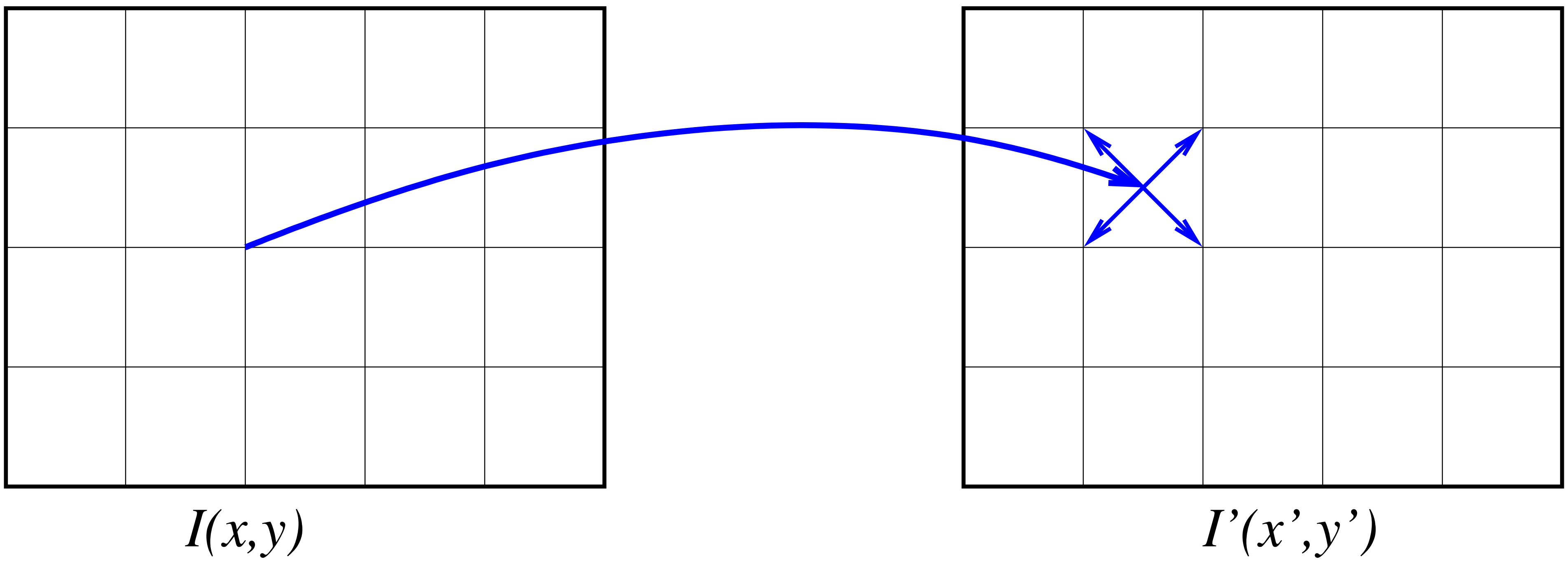
Forward mapping — problems

Maps outside the pixel locations.



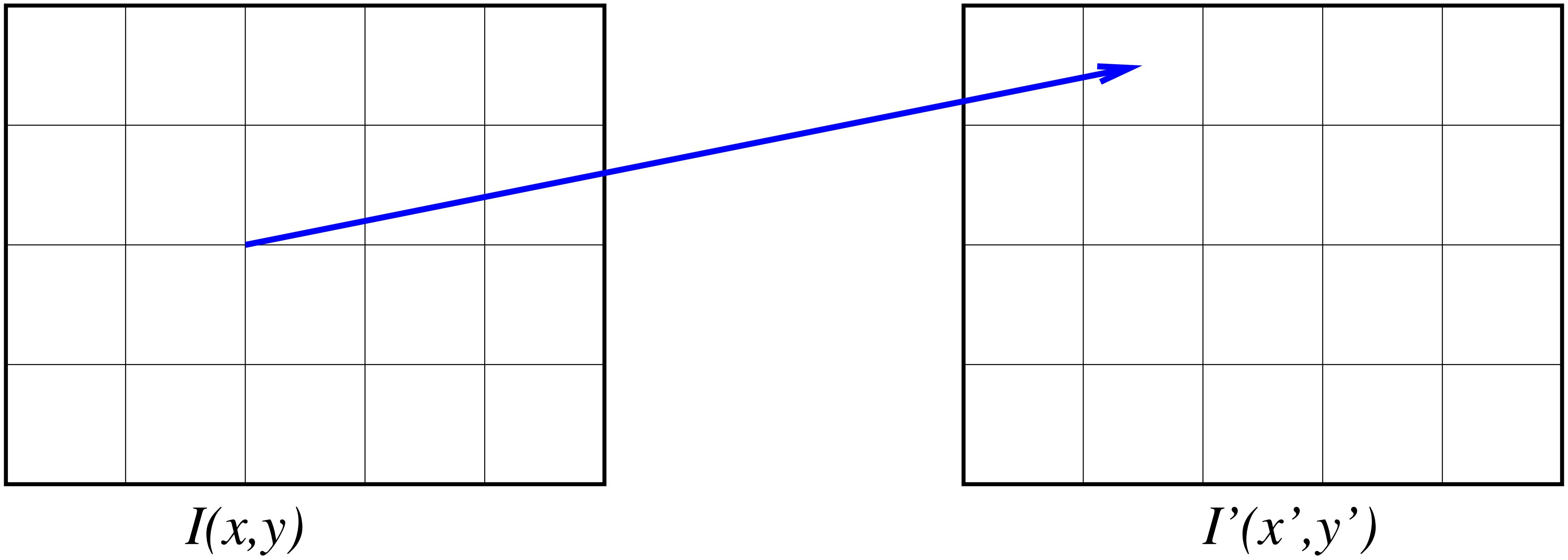
Forward mapping — problems

Solution: Spread out the effect of each pixel



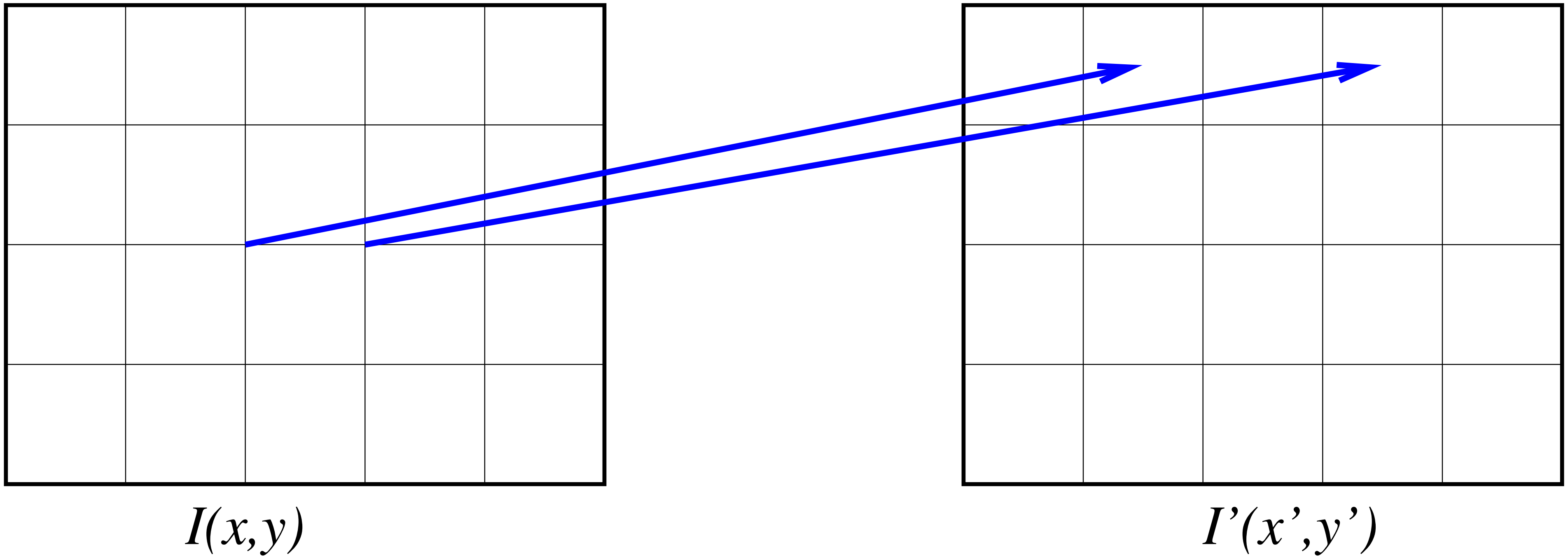
Forward mapping — problems

May produce holes in the output



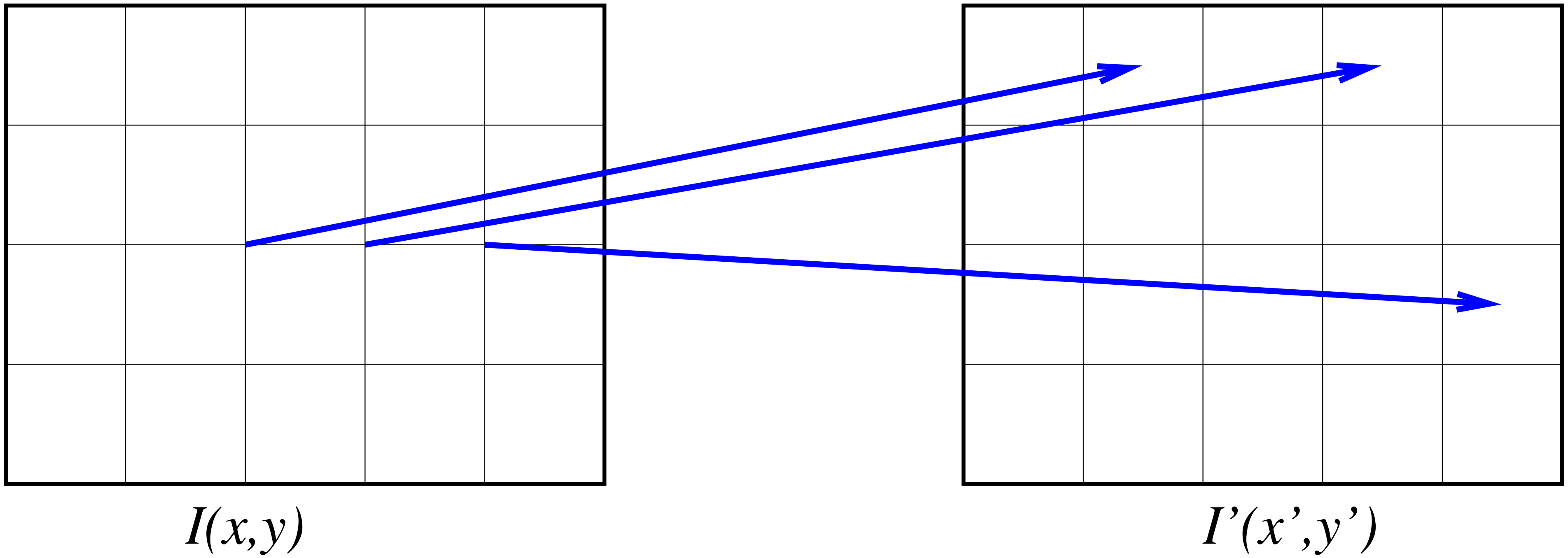
Forward mapping — problems

May produce holes in the output



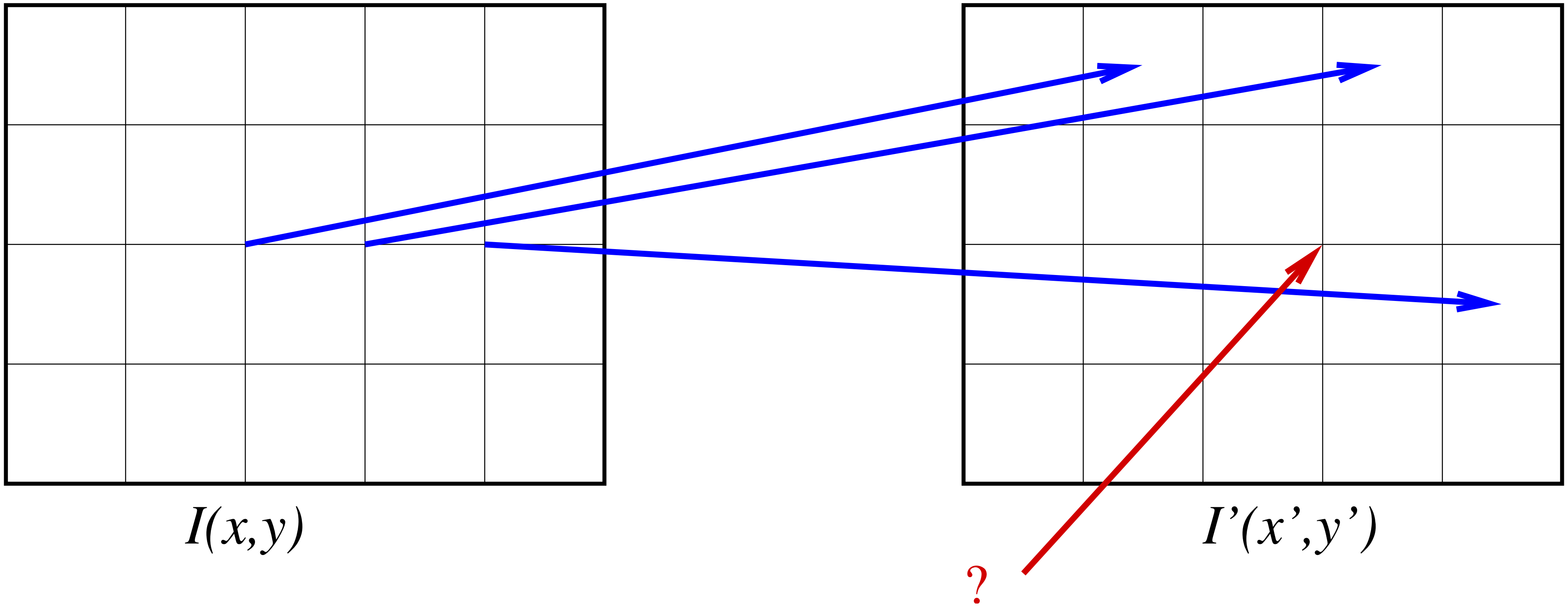
Forward mapping — problems

May produce holes in the output



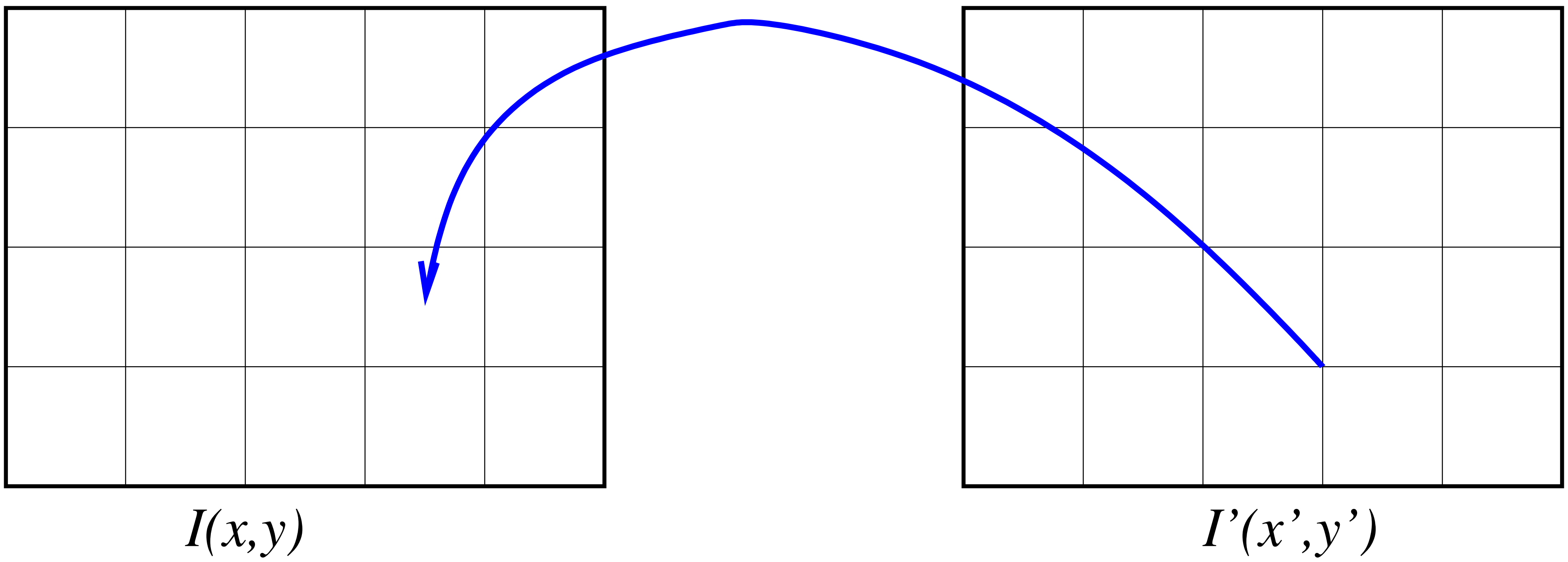
Forward mapping — problems

May produce holes in the output



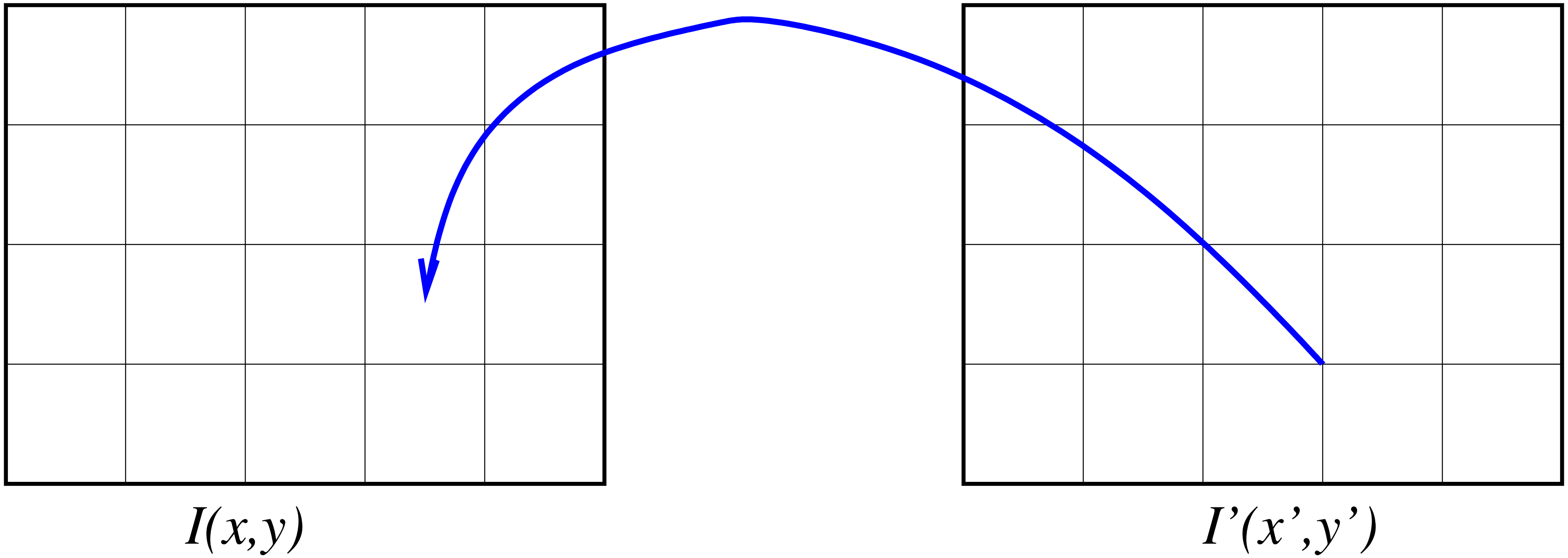
Backward mapping

$\mathbf{x} = T^{-1}\mathbf{x}'$



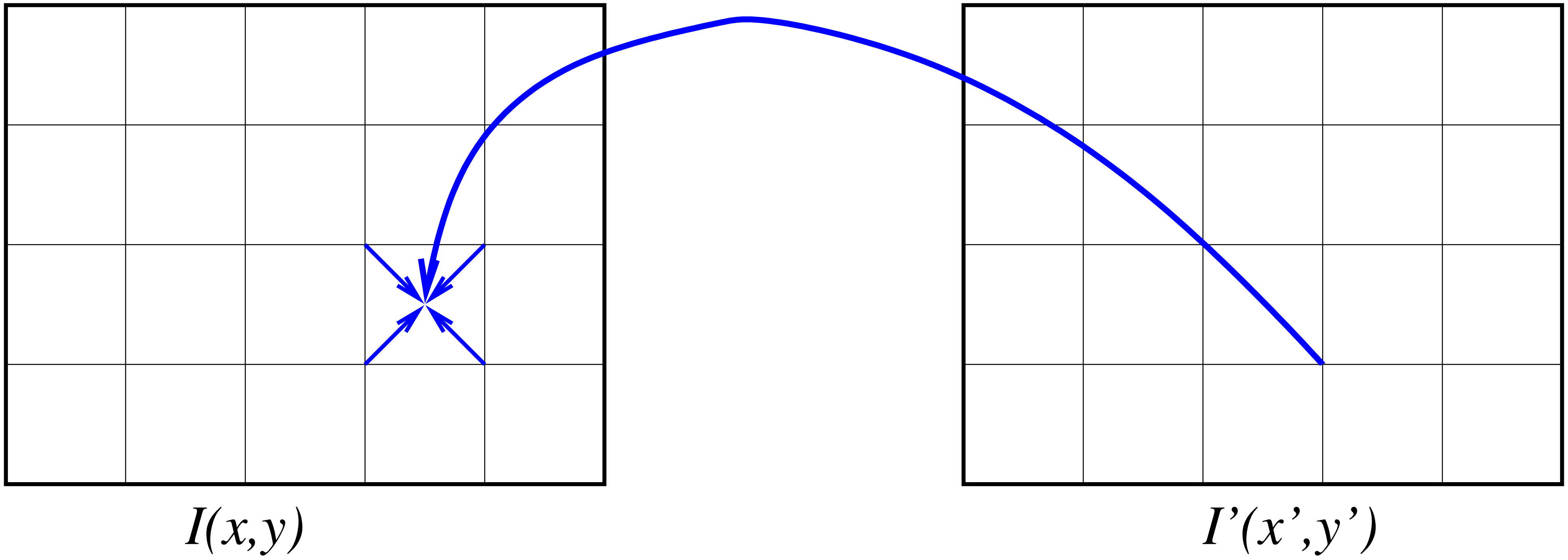
Backward mapping — problems

Does not always map from a pixel



Backward mapping — problems

Solution: Interpolate between pixels



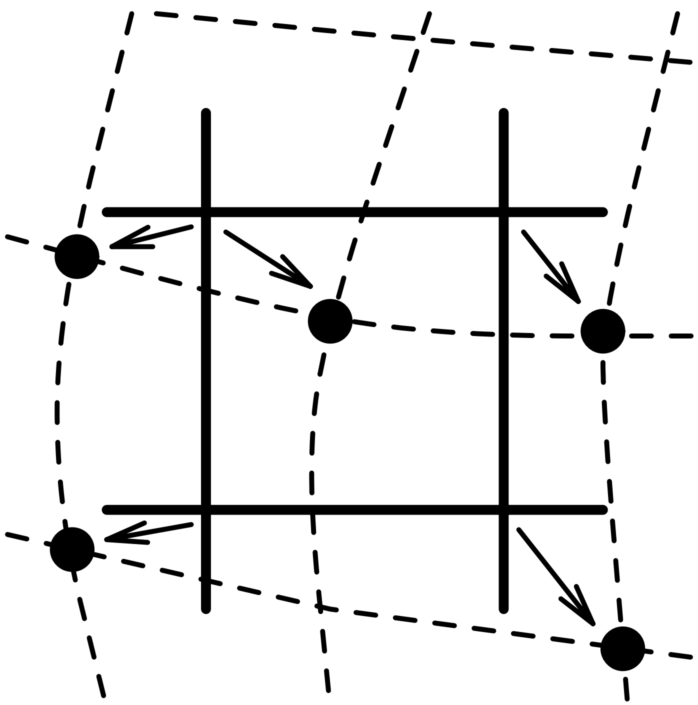
Interpolation as 2D convolution

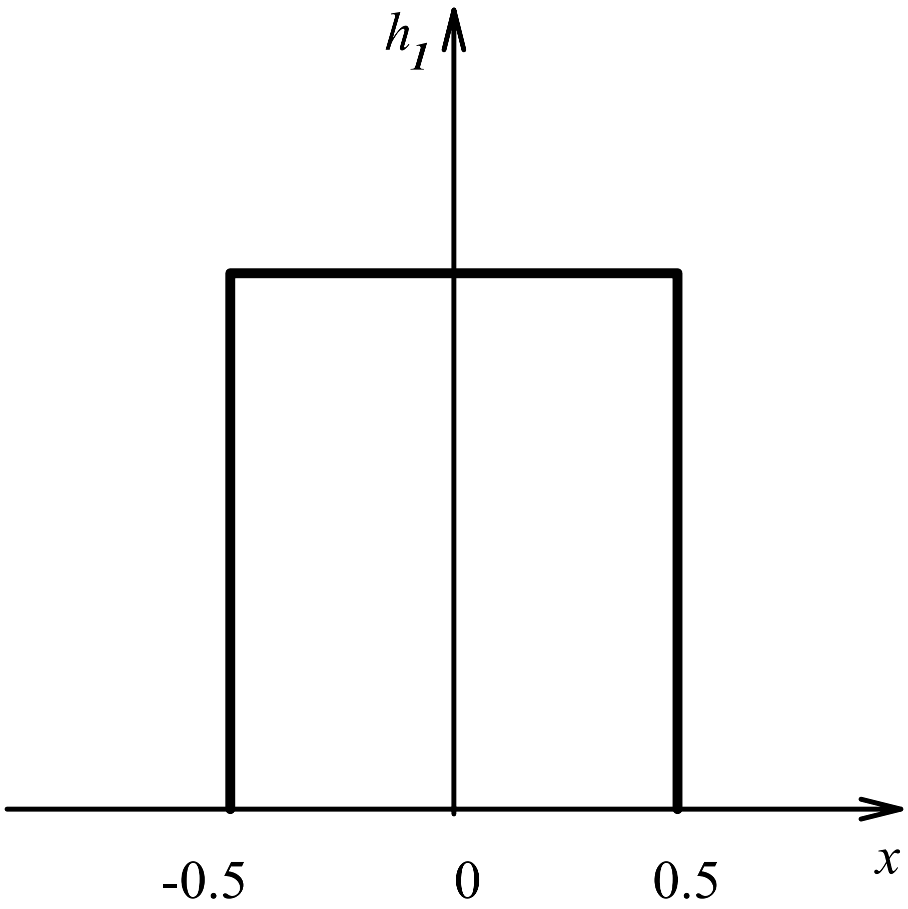
We have sampled (possibly outside the regular grid) function $s(x, y)$ instead of the wanted $f(x, y)$.

We want to find a good approximation by convolution

$$\hat{f}(x, y) = \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} s(k, l) h(x - k, y - l)$$

Interpolation — nearest neighbour



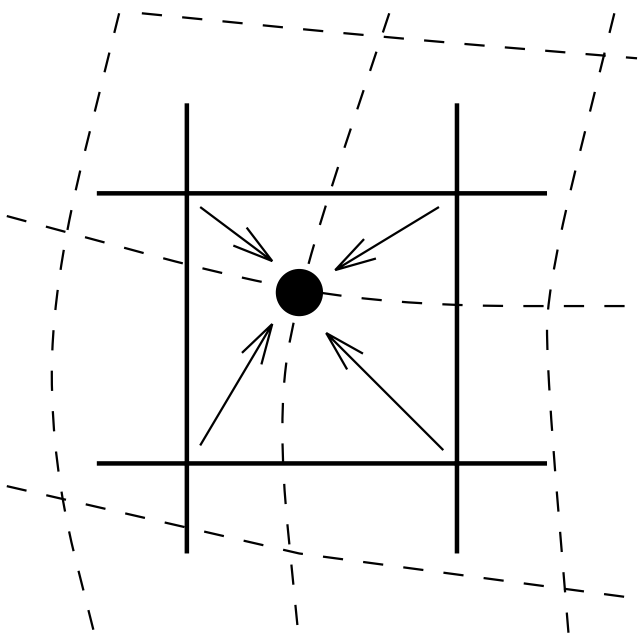


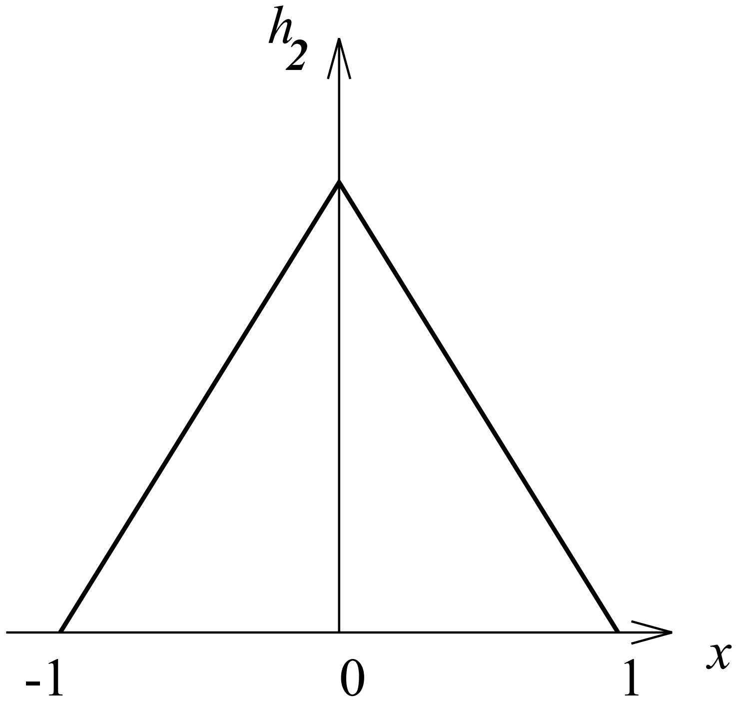
Interpolation — bilinear

$$\begin{aligned}\hat{f}(x,y) = \;& (1-a)(1-b)\,s(l,k) + a(1-b)\,s(l+1,k) \\ & +b(1-a)\,s(l,k+1) + ab\,s(l+1,k+1)\;,\end{aligned}$$

where $\quad l = round\,(x)\;, \quad a = x - l\;,$

$\quad k = round\,(y)\;, \quad b = y - k\;.$

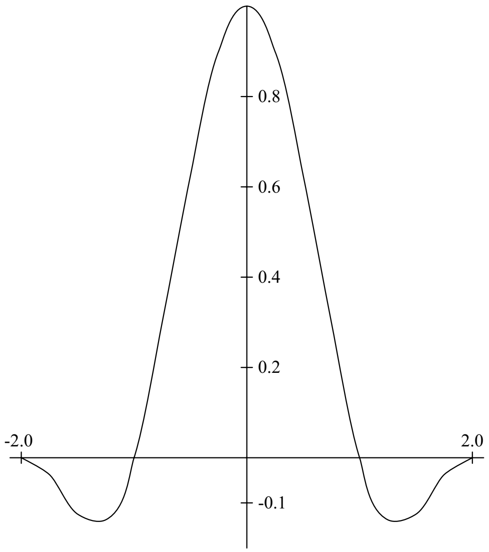




Interpolation — bicubic

Just 1D, for clarity

$$\hat{f} = \begin{cases} 1 - 2|x|^2 + |x|^3 & \text{pro } 0 \le |x| < 1\,, \\ 4 - 8|x| + 5|x|^2 - |x|^3 & \text{pro } 1 \le |x| < 2\,, \\ 0 & \text{elsewhere.} \end{cases}$$

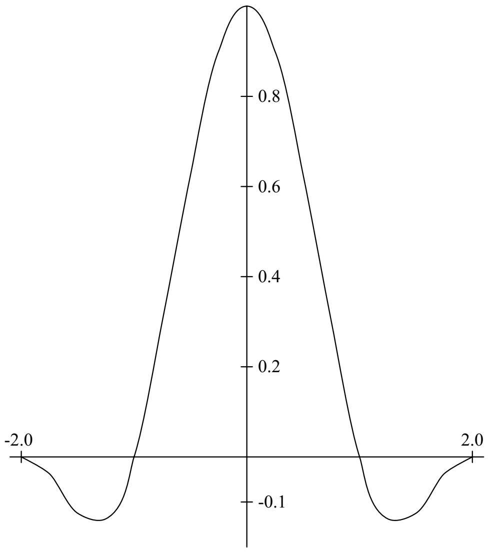


Does not remind you the shape something?

Interpolation — bicubic

Just 1D, for clarity

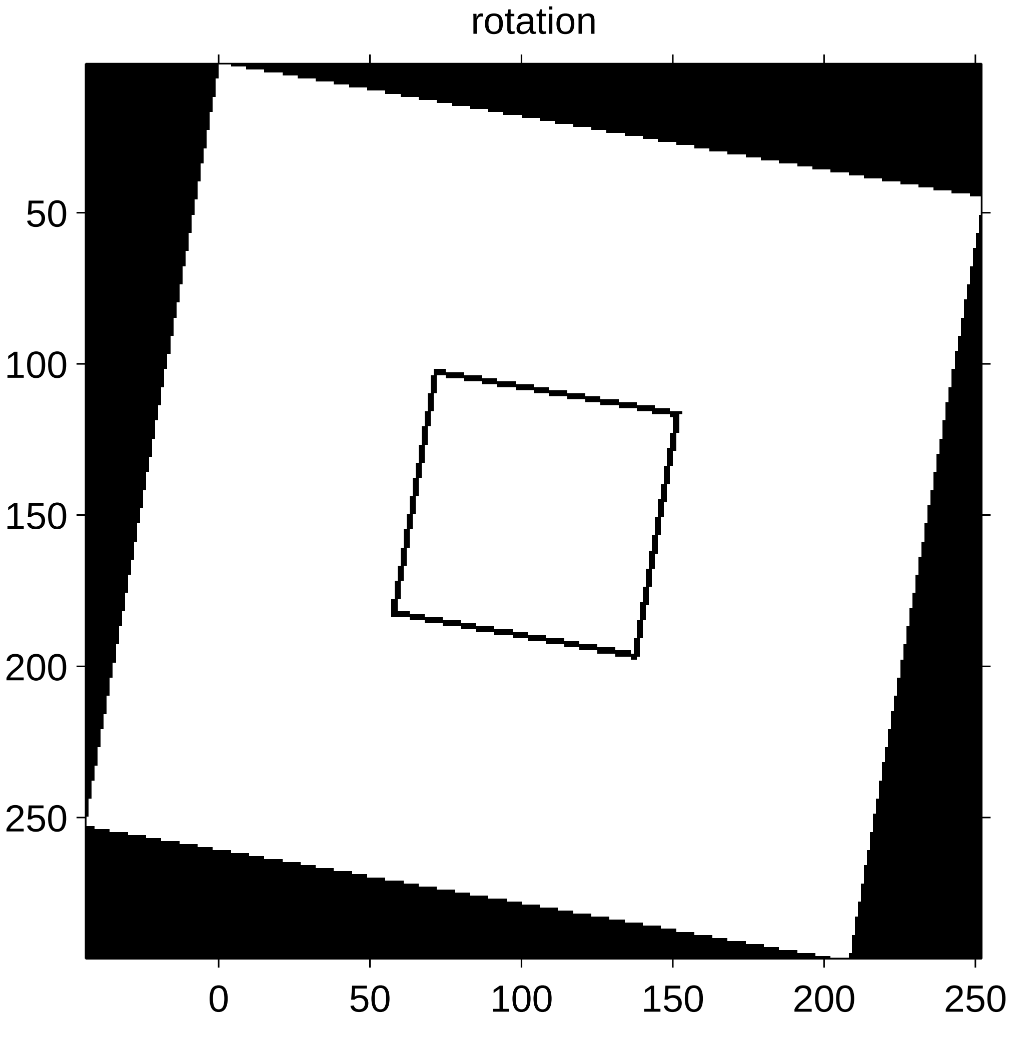
$$\hat{f} = \begin{cases} 1 - 2|x|^2 + |x|^3 & \text{pro } 0 \leq |x| < 1\,, \\ 4 - 8|x| + 5|x|^2 - |x|^3 & \text{pro } 1 \leq |x| < 2\,, \\ 0 & \text{elsewhere.} \end{cases}$$



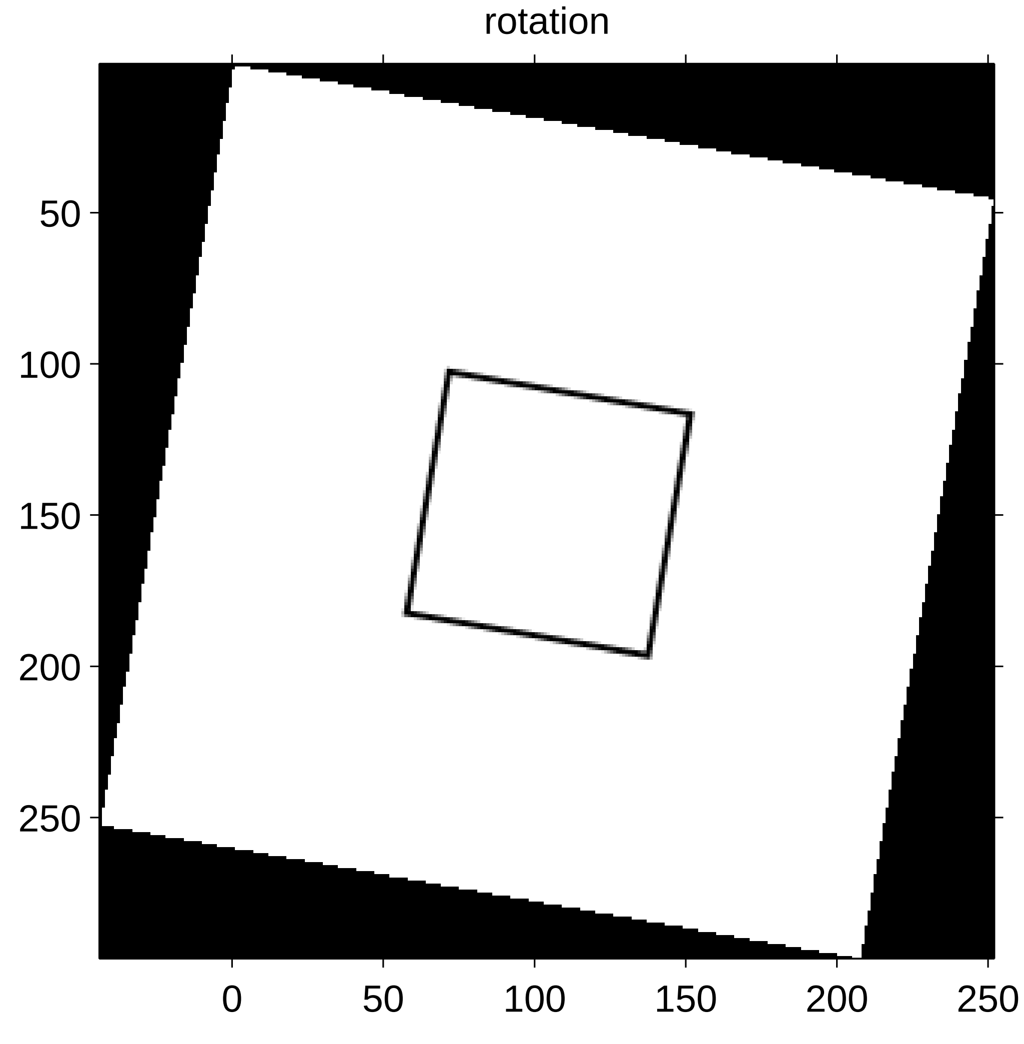
Does not remind you the shape something?

$\mathrm{sinc}(x)$! The ideal reconstructor.

Interpolation — nearest vs. bilinear

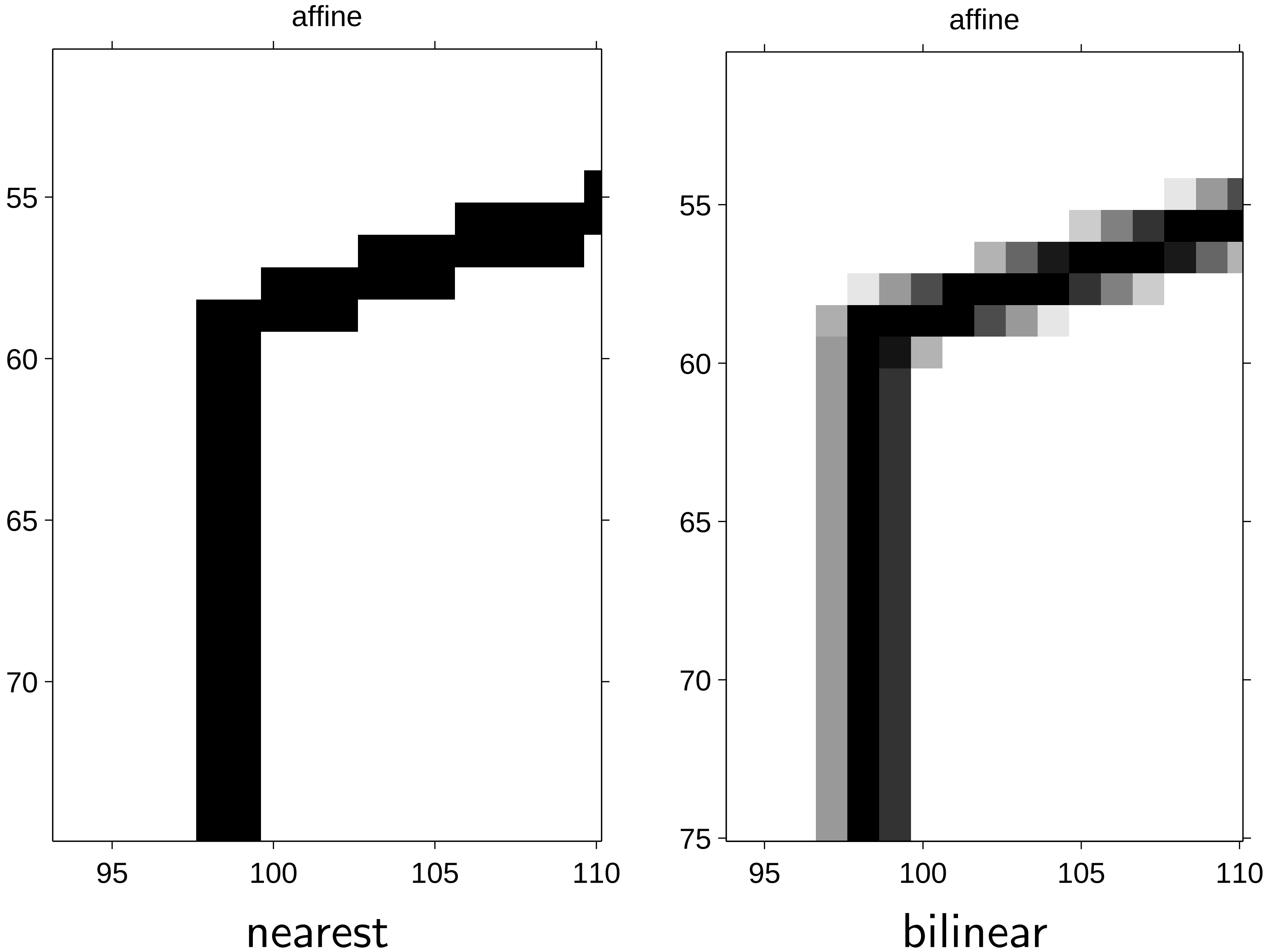


nearest



bilinear

Interpolation — nearest vs. bilinear — close up

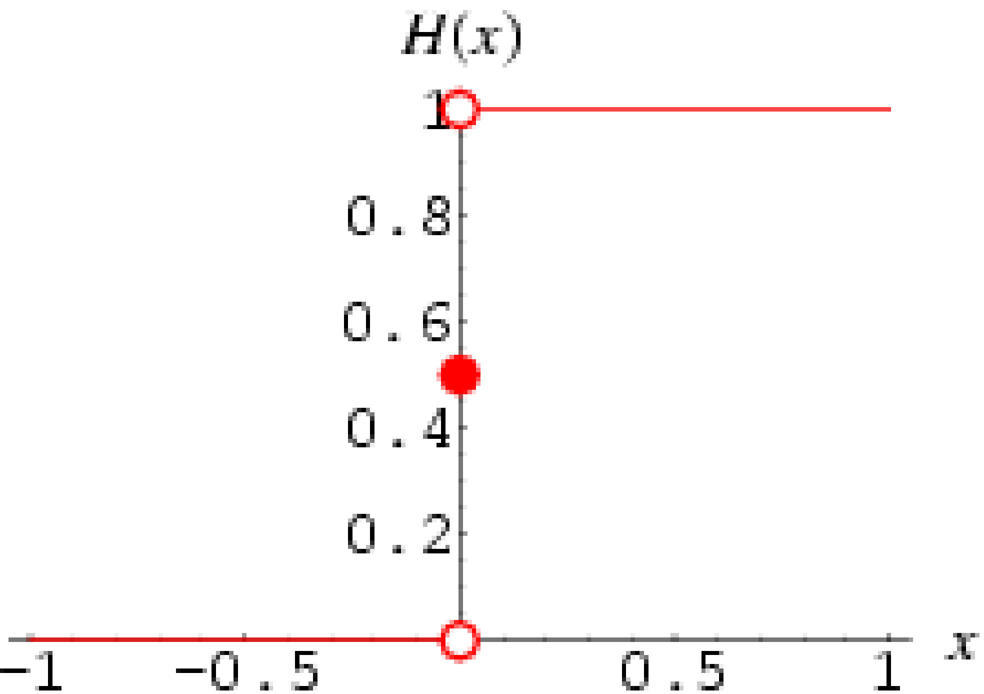


Geometrical Transformation — Summary

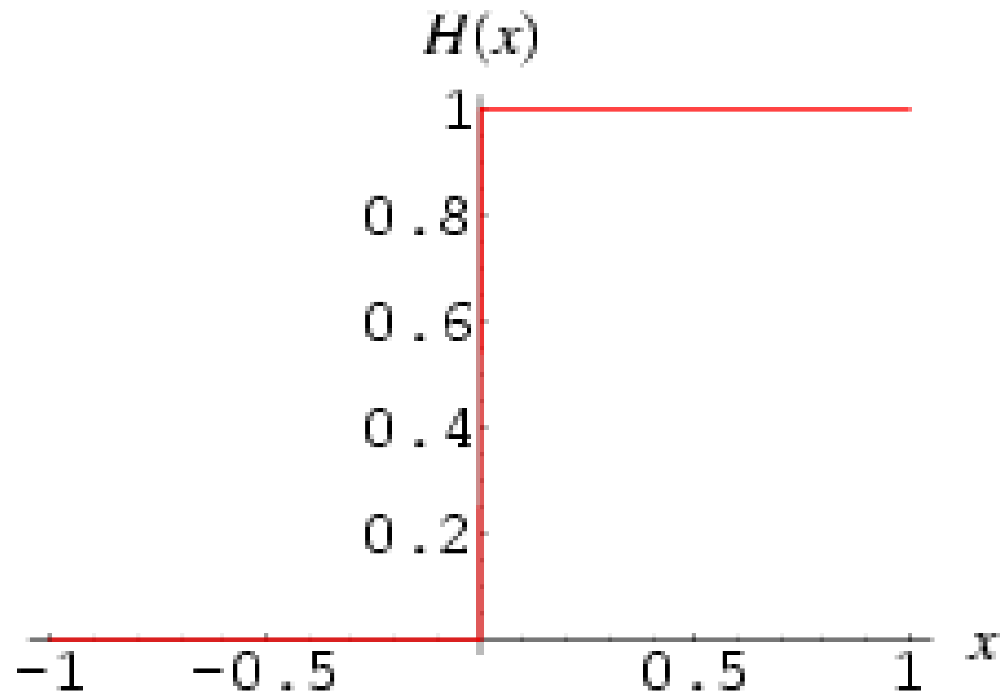
- Closed form solution, all pixels undergo the same transformation
  - rotation, scaling, translation
  - affine, perspective
- General, deformation meshes. Transformation depends on position (warping, morphing)

References

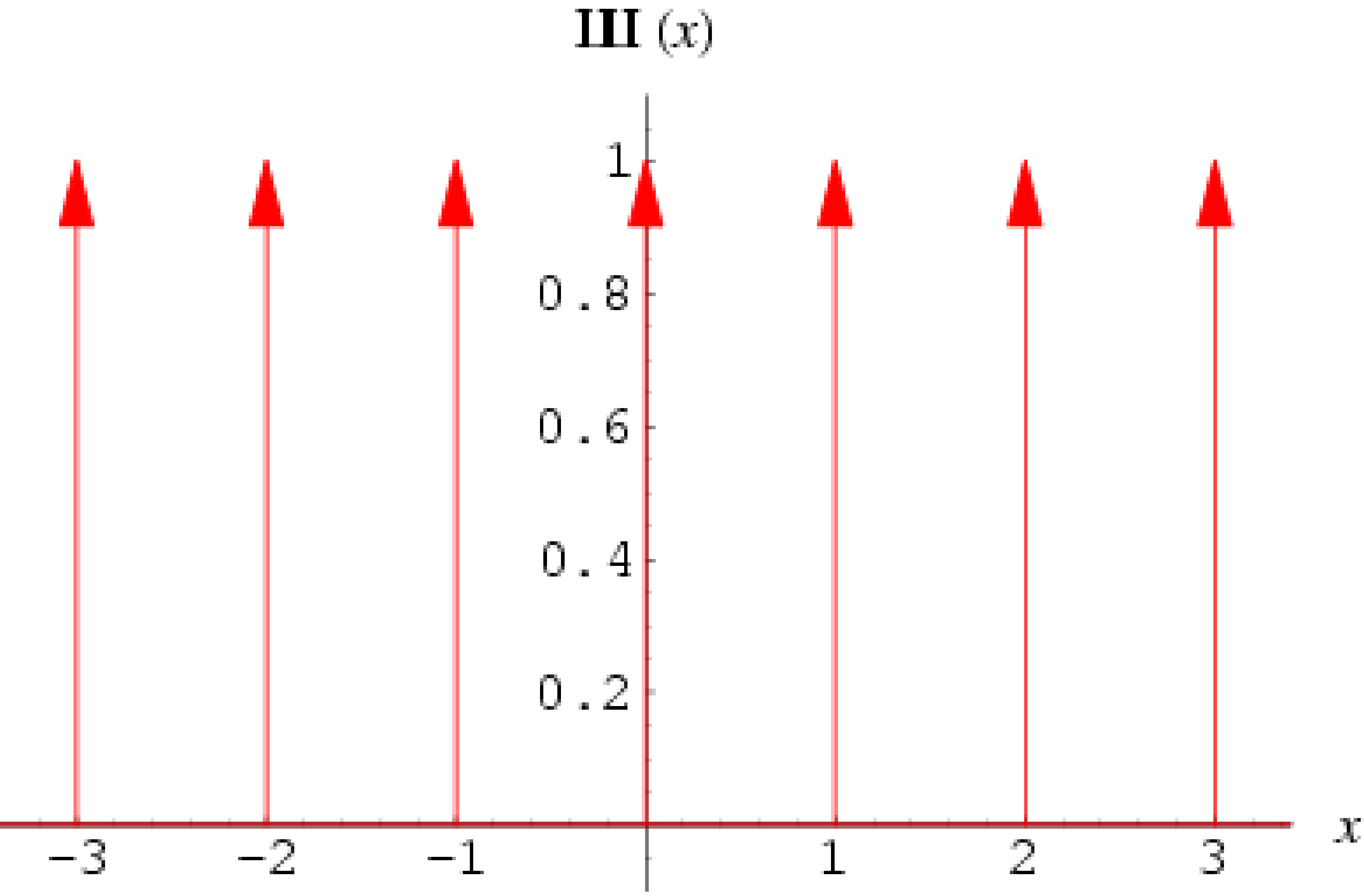
[1] Ronald N. Bracewell. Fourier analysis and imaging. Kluwer Academic/Plenum Publishers, New York, USA, 2003.



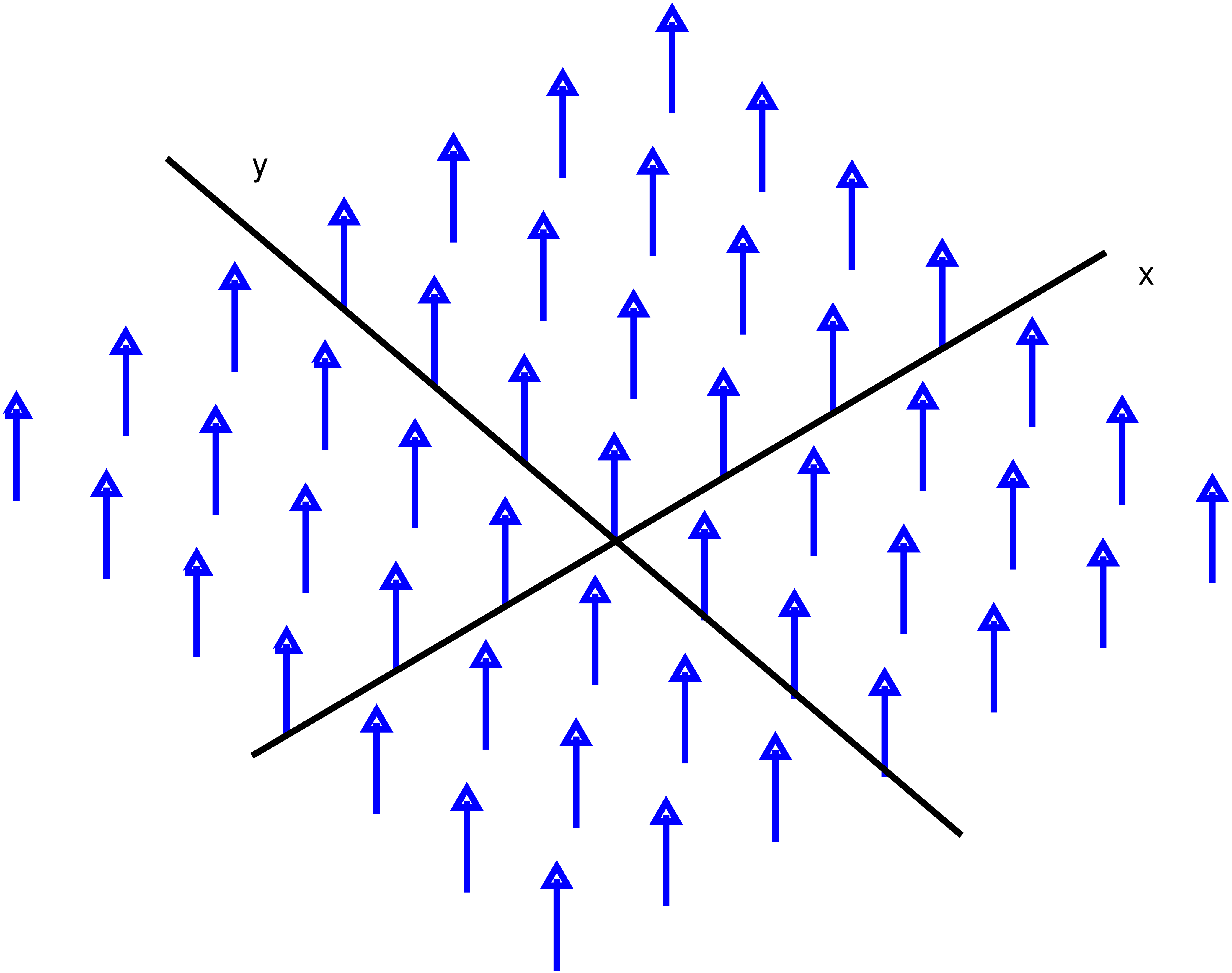
H(x)
1
0.8
0.6
0.4
0.2
x
-1
-0.5
0.5
1

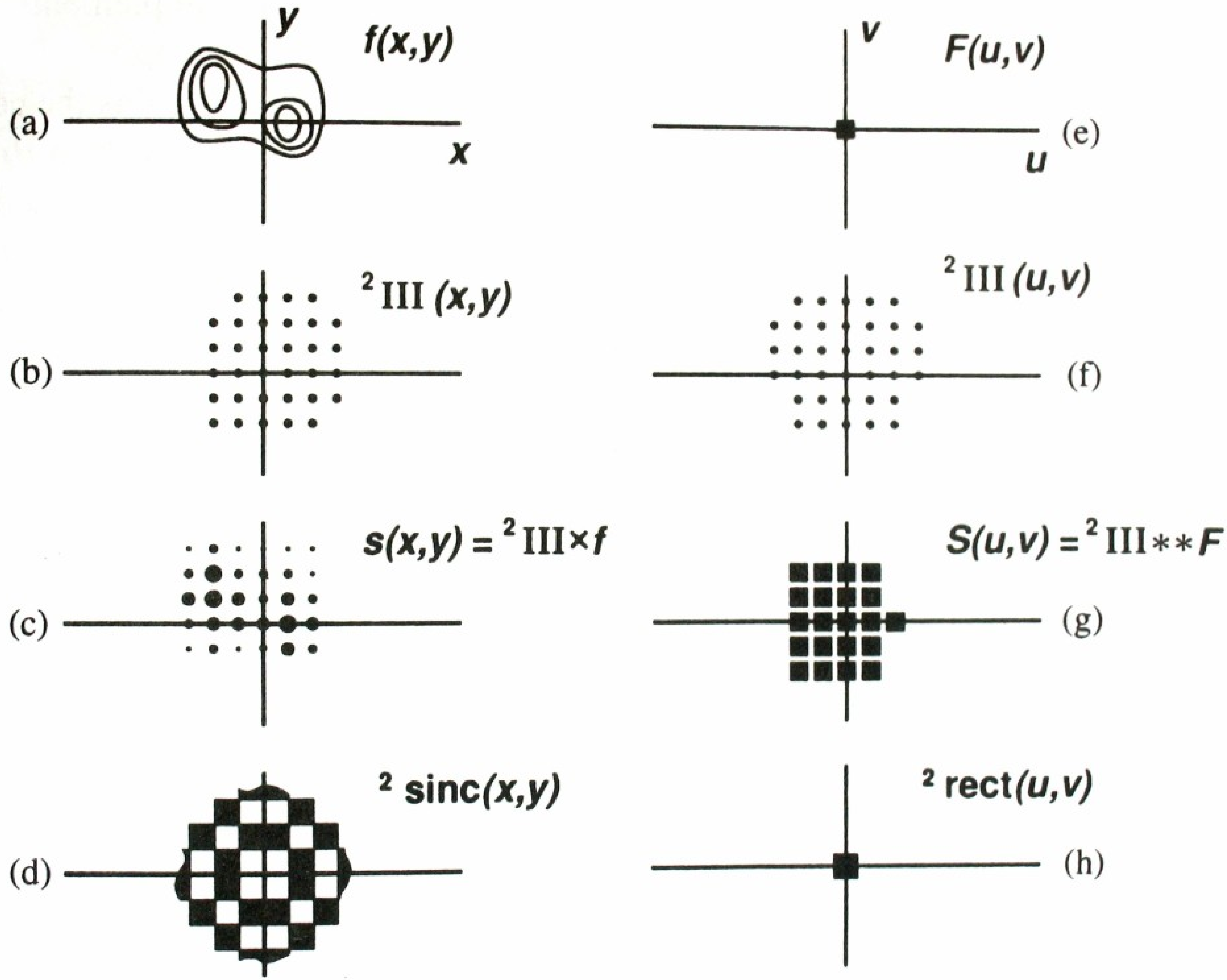


H(x)
1
0.8
0.6
0.4
0.2
x
-1
-0.5
0.5
1

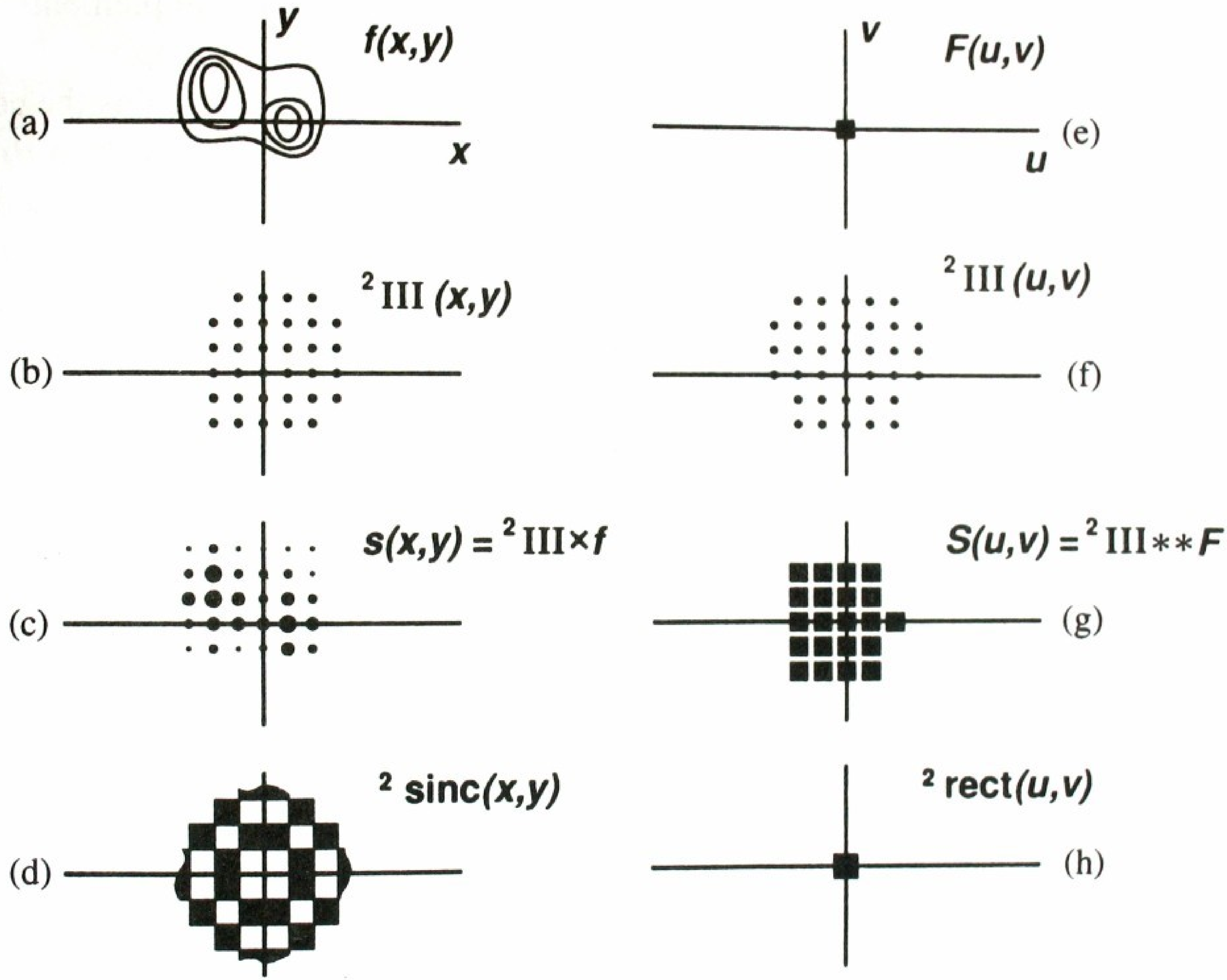


III (x)
1
0.8
0.6
0.4
0.2
x
−3
−2
−1
1
2
3

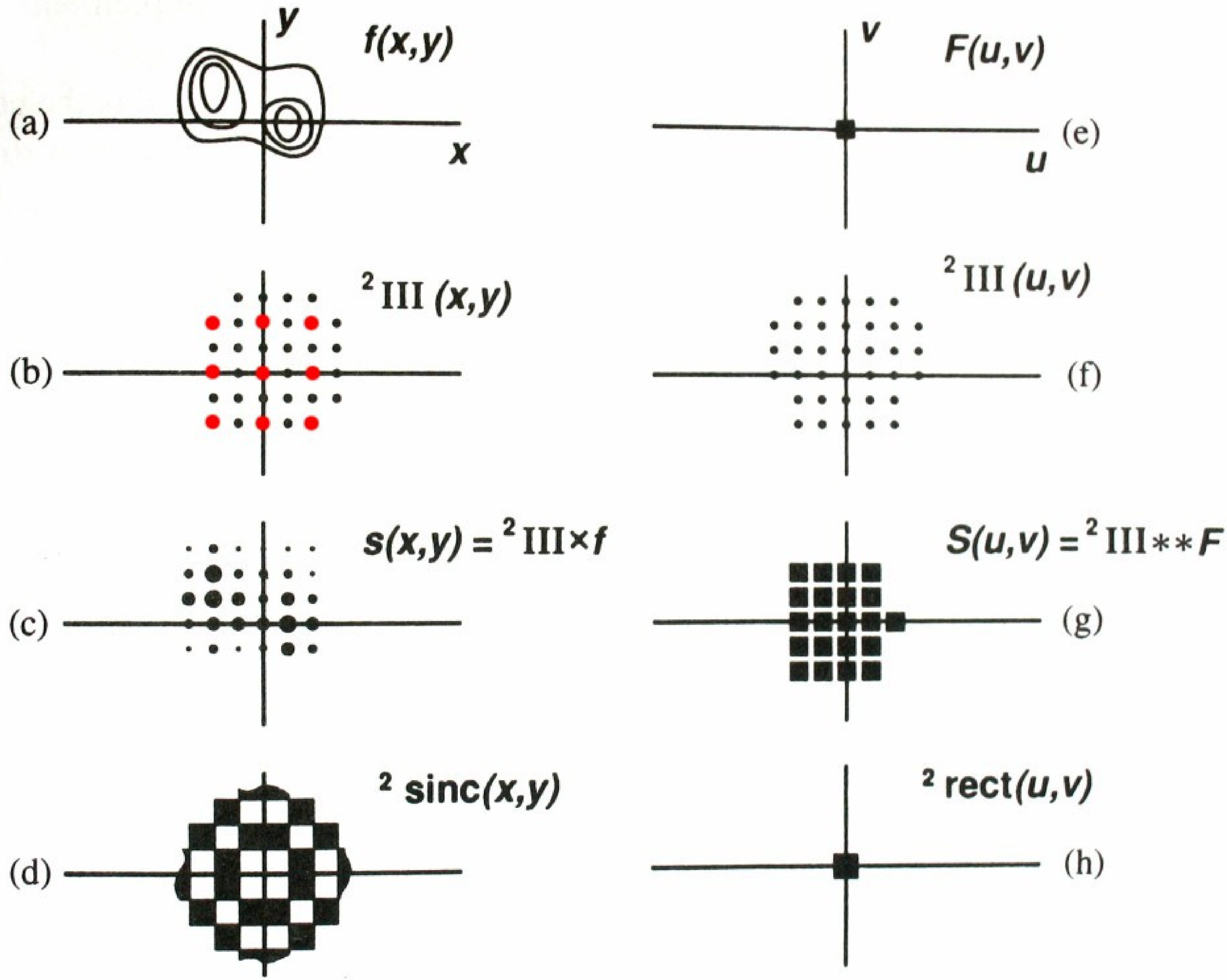

y
x



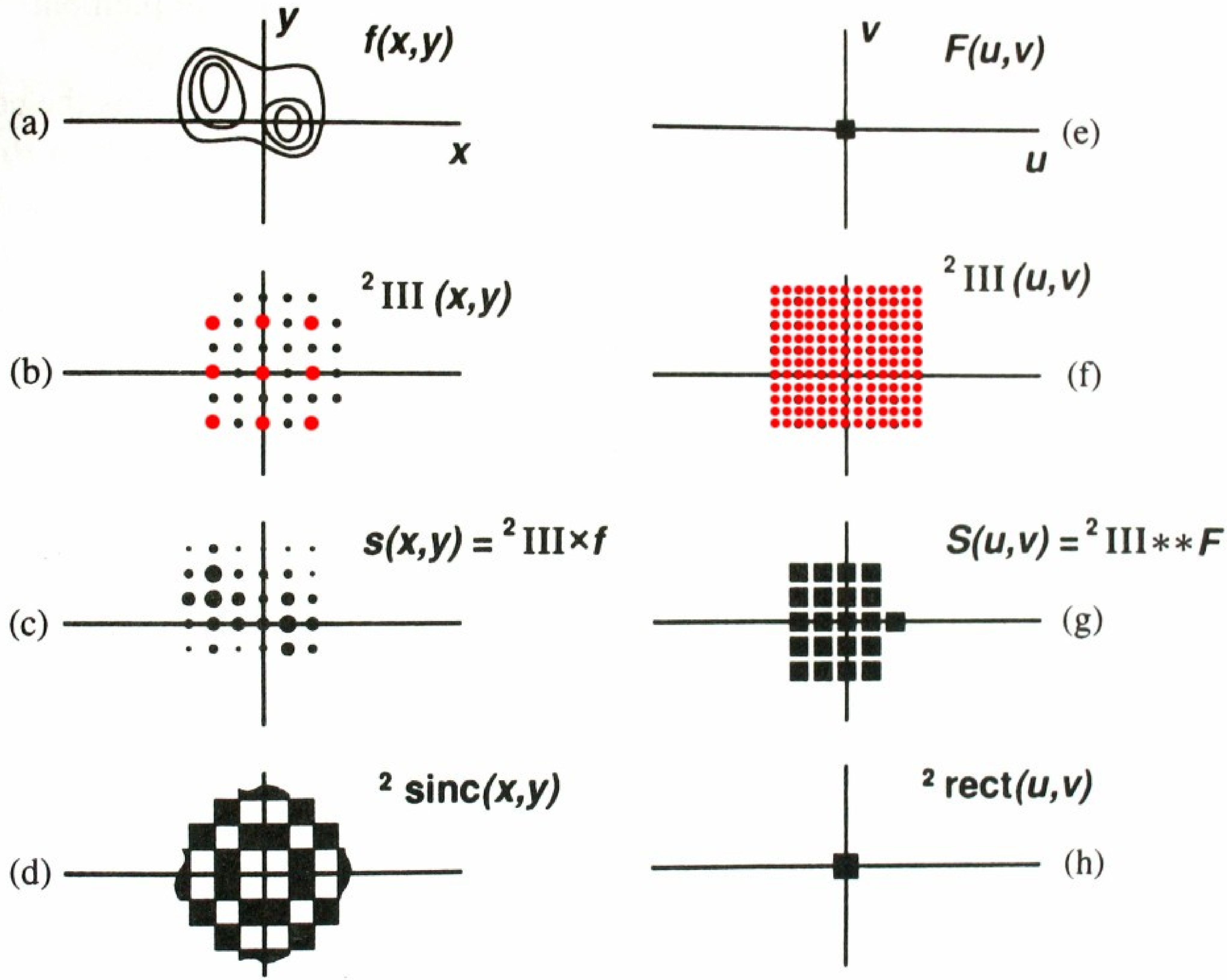
(a)
y
f(x,y)
x
(e)
v
F(u,v)
u
(b)
²III (x,y)
(f)
²III (u,v)
(c)
s(x,y) = ²III × f
(g)
S(u,v) = ²III ** F
(d)
² sinc(x,y)
(h)
² rect(u,v)



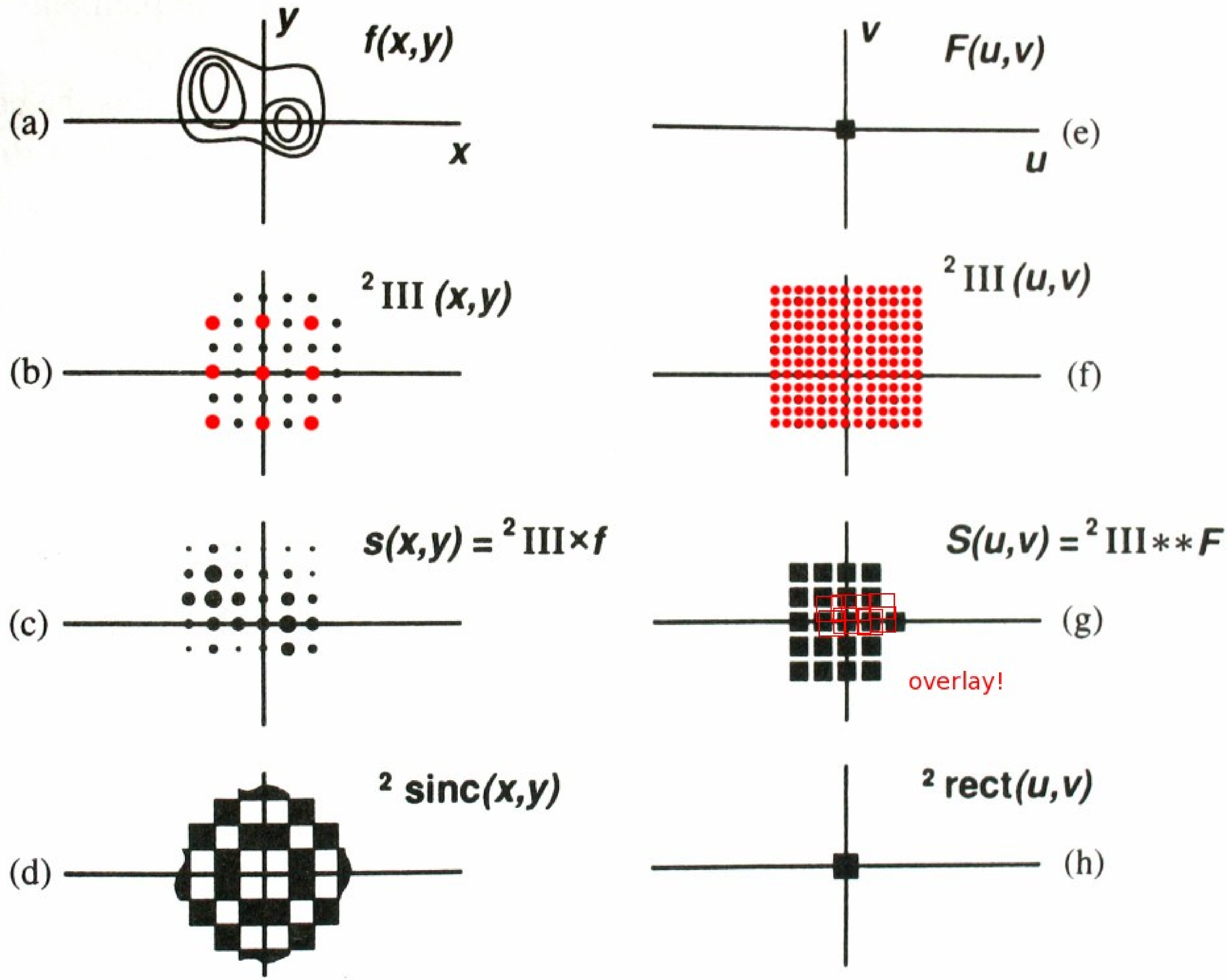
(a)
f(x,y)
x
y
(b)
²III (x,y)
(c)
s(x,y) = ²III×f
(d)
² sinc(x,y)
(e)
F(u,v)
u
v
(f)
²III (u,v)
(g)
S(u,v) = ²III∗∗F
(h)
² rect(u,v)



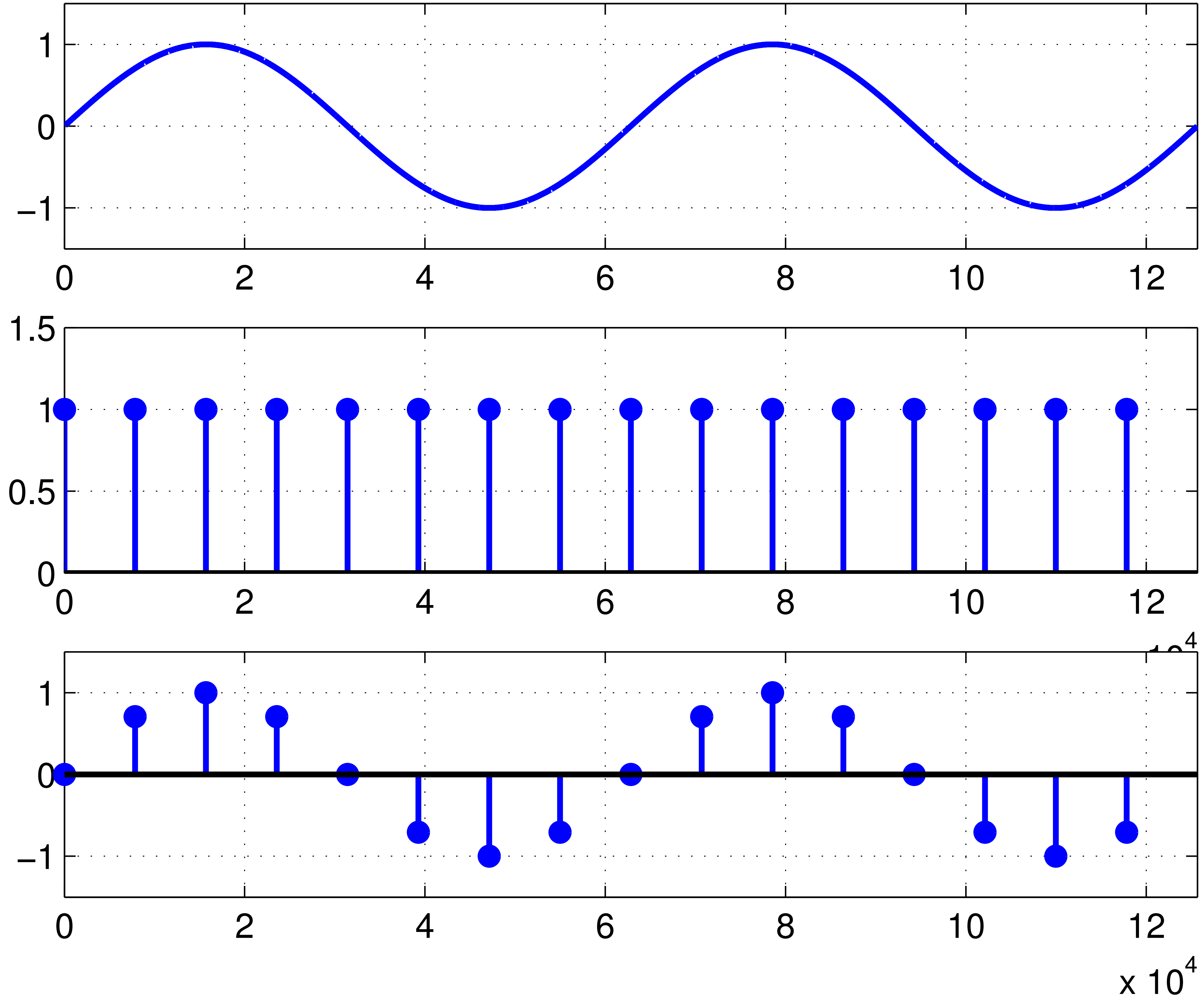
(a)
f(x,y)
y
x
(b)
²III (x,y)
(c)
s(x,y) = ²III×f
(d)
² sinc(x,y)
(e)
F(u,v)
v
u
(f)
²III (u,v)
(g)
S(u,v) = ²III**F
(h)
² rect(u,v)



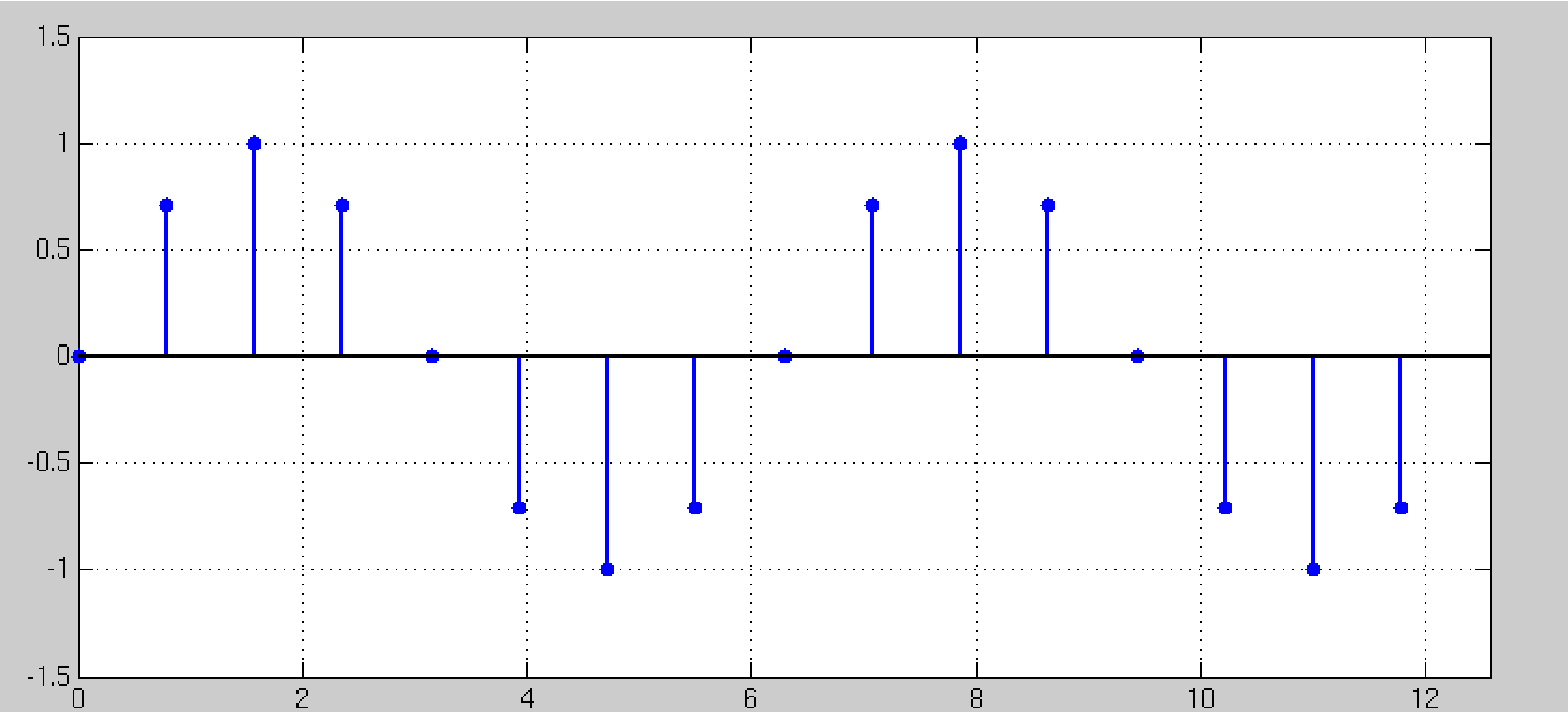
f(x,y)
y
x
(a)
F(u,v)
v
u
(e)
²III (x,y)
(b)
²III (u,v)
(f)
s(x,y) = ²III×f
(c)
S(u,v) = ²III∗∗F
(g)
² sinc(x,y)
(d)
² rect(u,v)
(h)

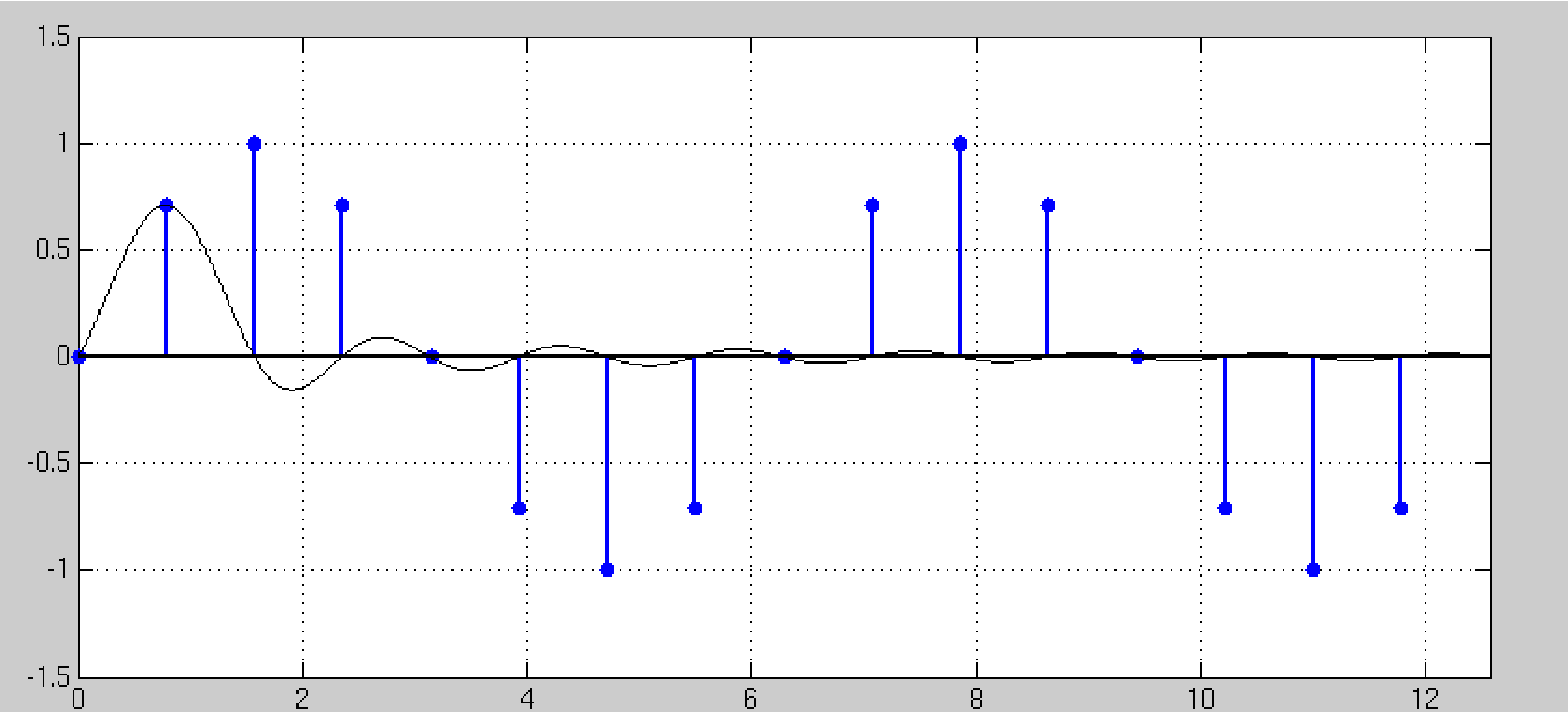


(a)
y
f(x,y)
x
(e)
v
F(u,v)
u
(b)
²III (x,y)
(f)
²III (u,v)
(c)
s(x,y) = ²III×f
(g)
S(u,v) = ²III**F
overlay!
(d)
² sinc(x,y)
(h)
² rect(u,v)

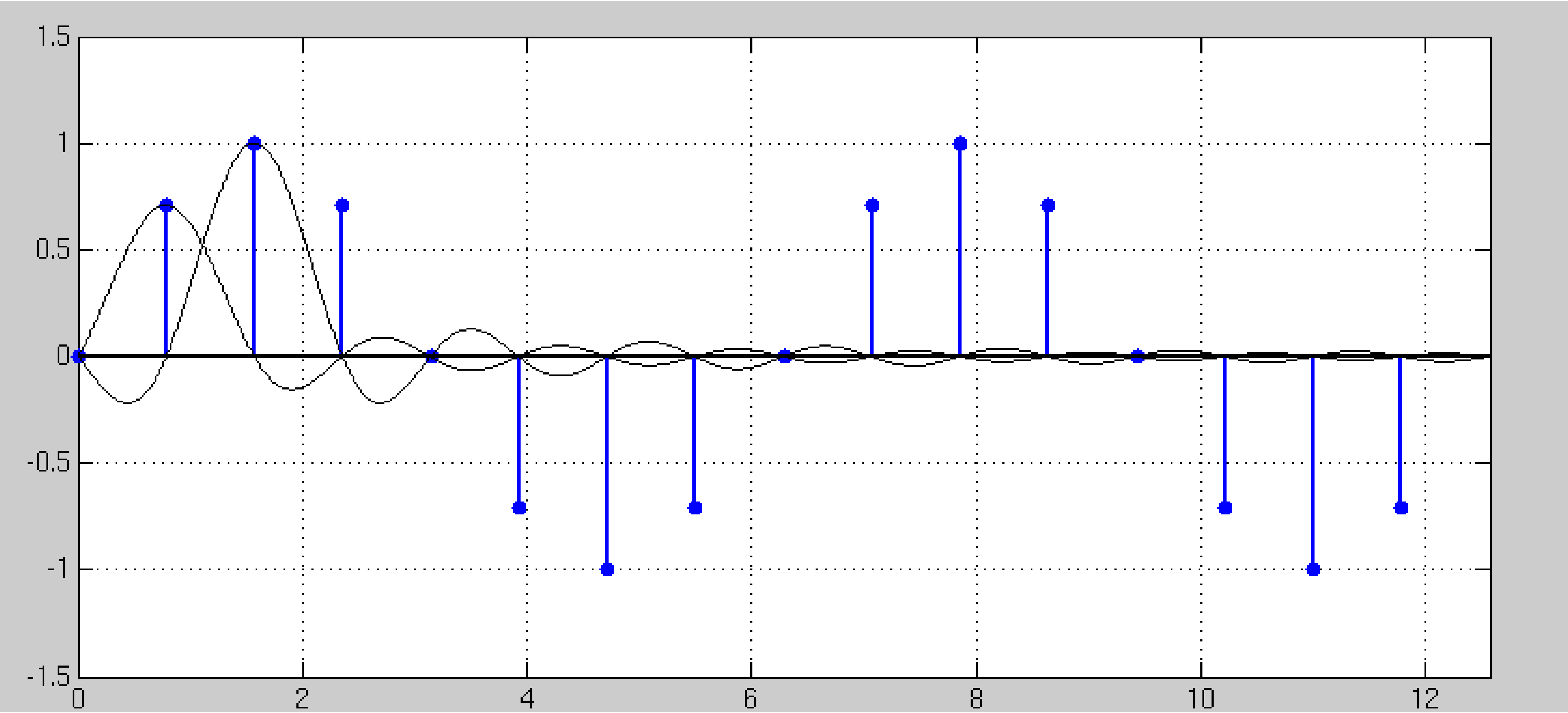


1
0
−1
0
2
4
6
8
10
12
1.5
1
0.5
0
0
2
4
6
8
10
12
1
0
−1
0
2
4
6
8
10
12
x 10^4

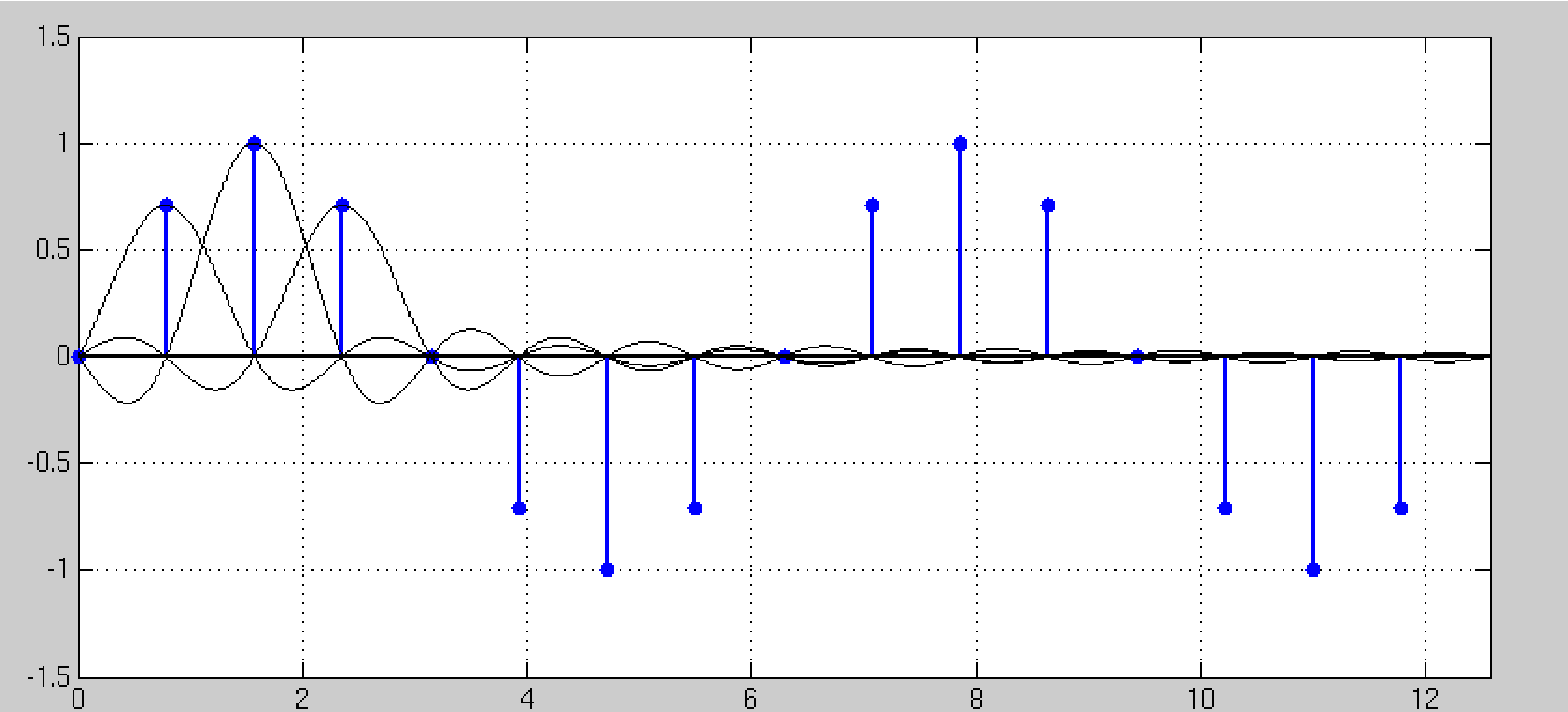

1.5
1
0.5
0
-0.5
-1
-1.5
0
2
4
6
8
10
12



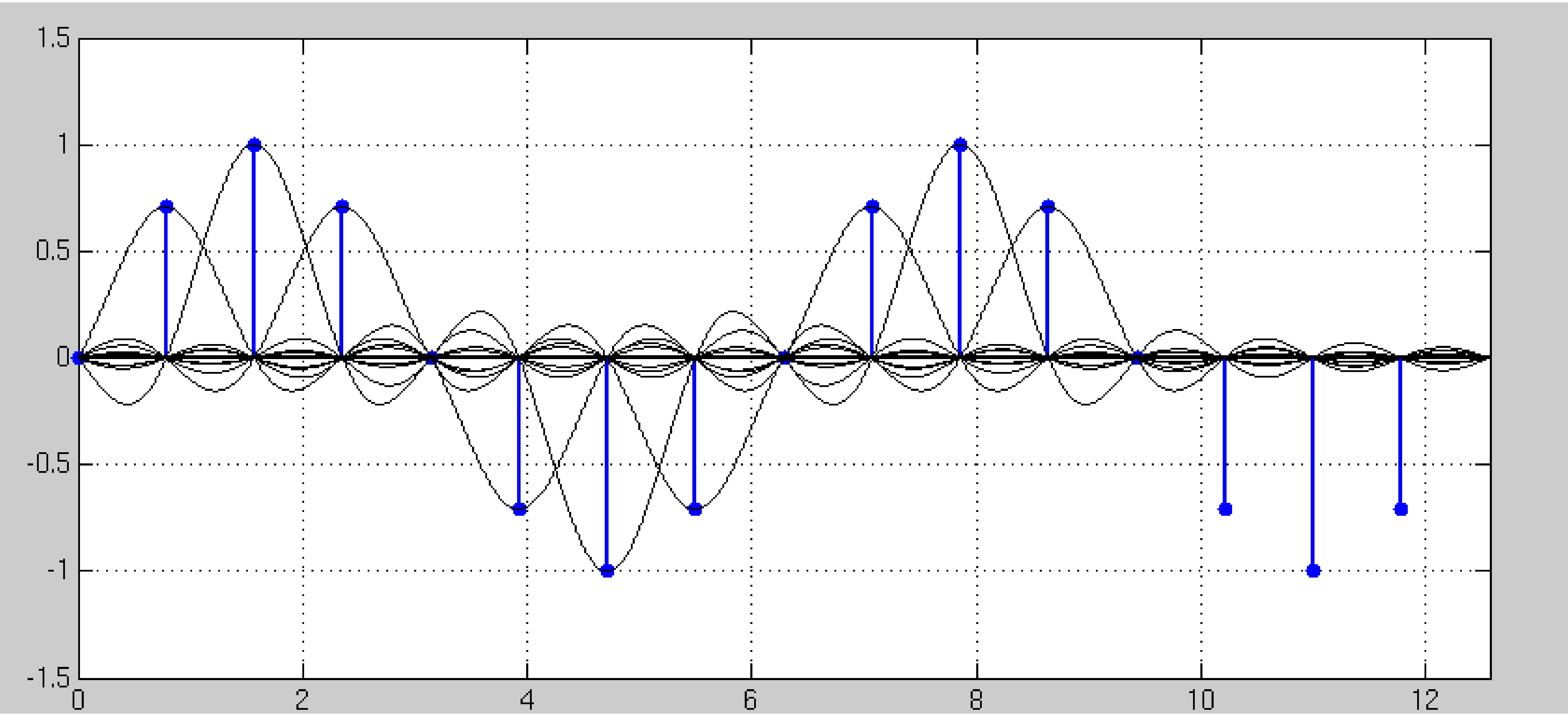
1.5
1
0.5
0
-0.5
-1
-1.5
0
2
4
6
8
10
12



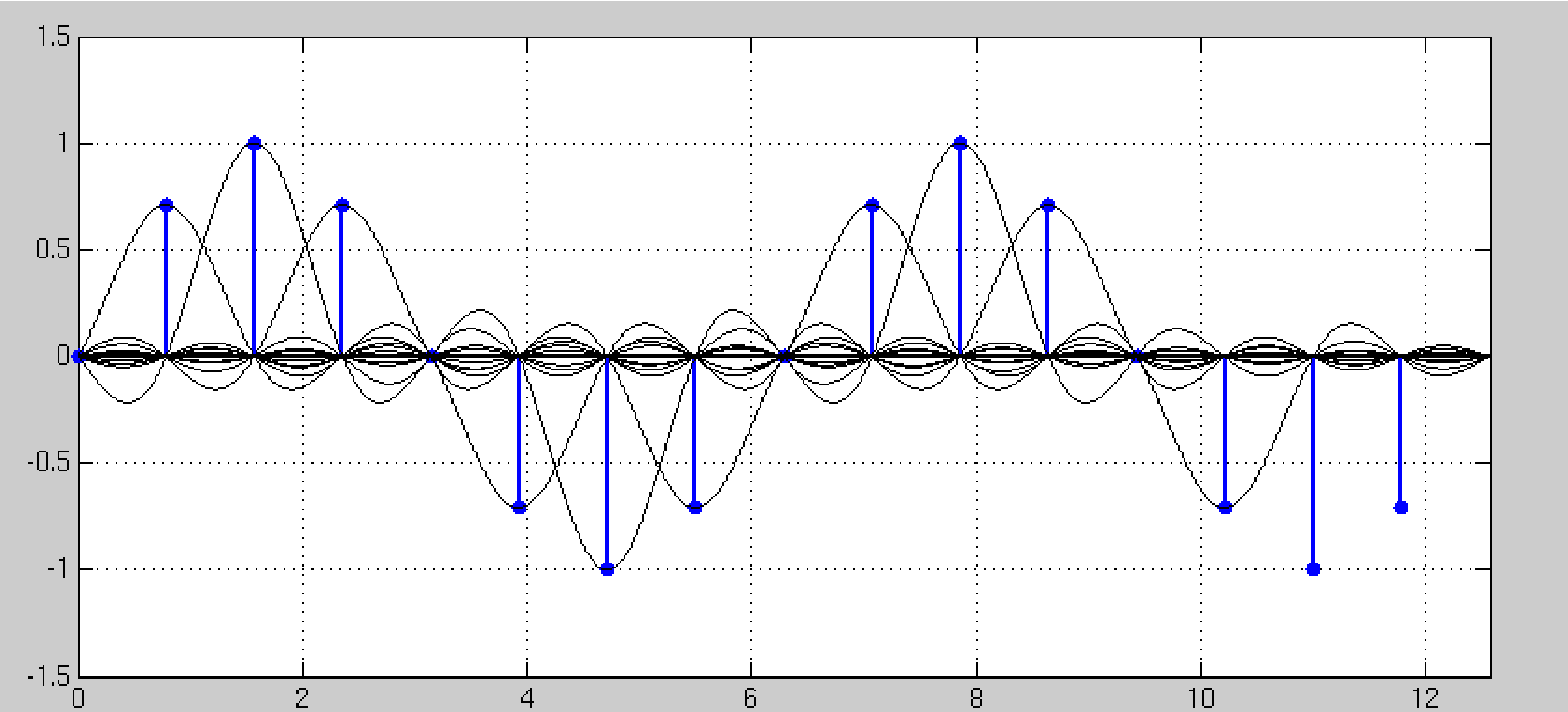
1.5
1
0.5
0
-0.5
-1
-1.5
0
2
4
6
8
10
12

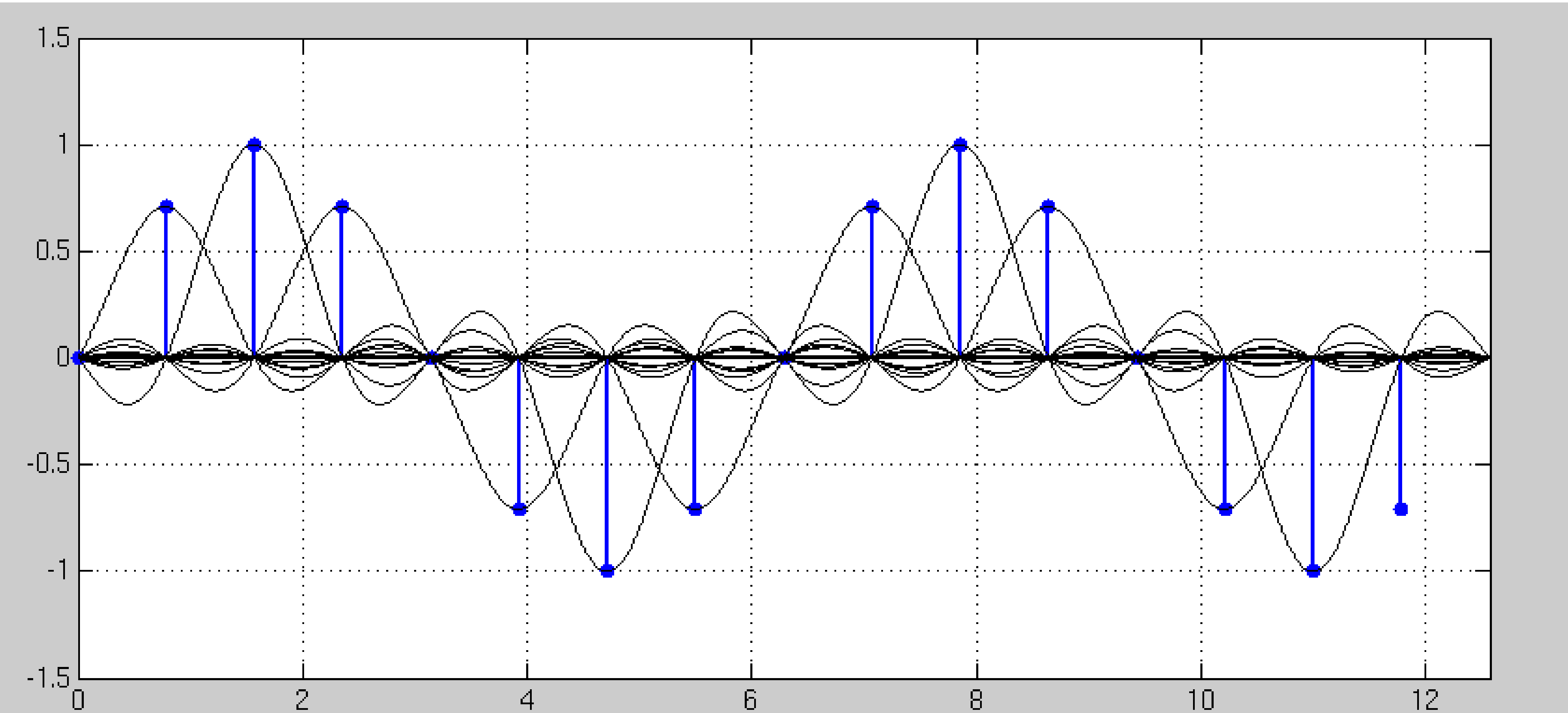


1.5
1
0.5
0
-0.5
-1
-1.5
0
2
4
6
8
10
12

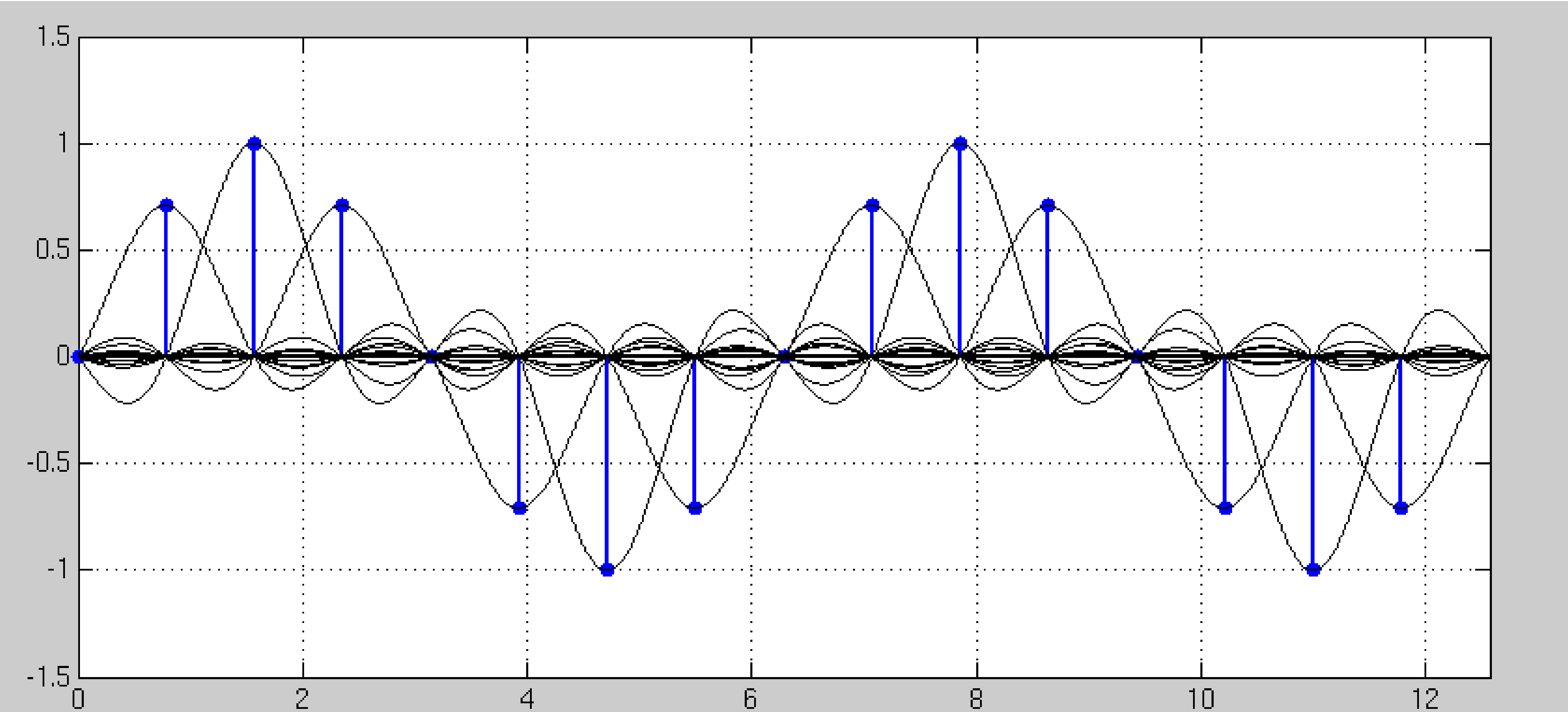


1.5
1
0.5
0
-0.5
-1
-1.5
0
2
4
6
8
10
12

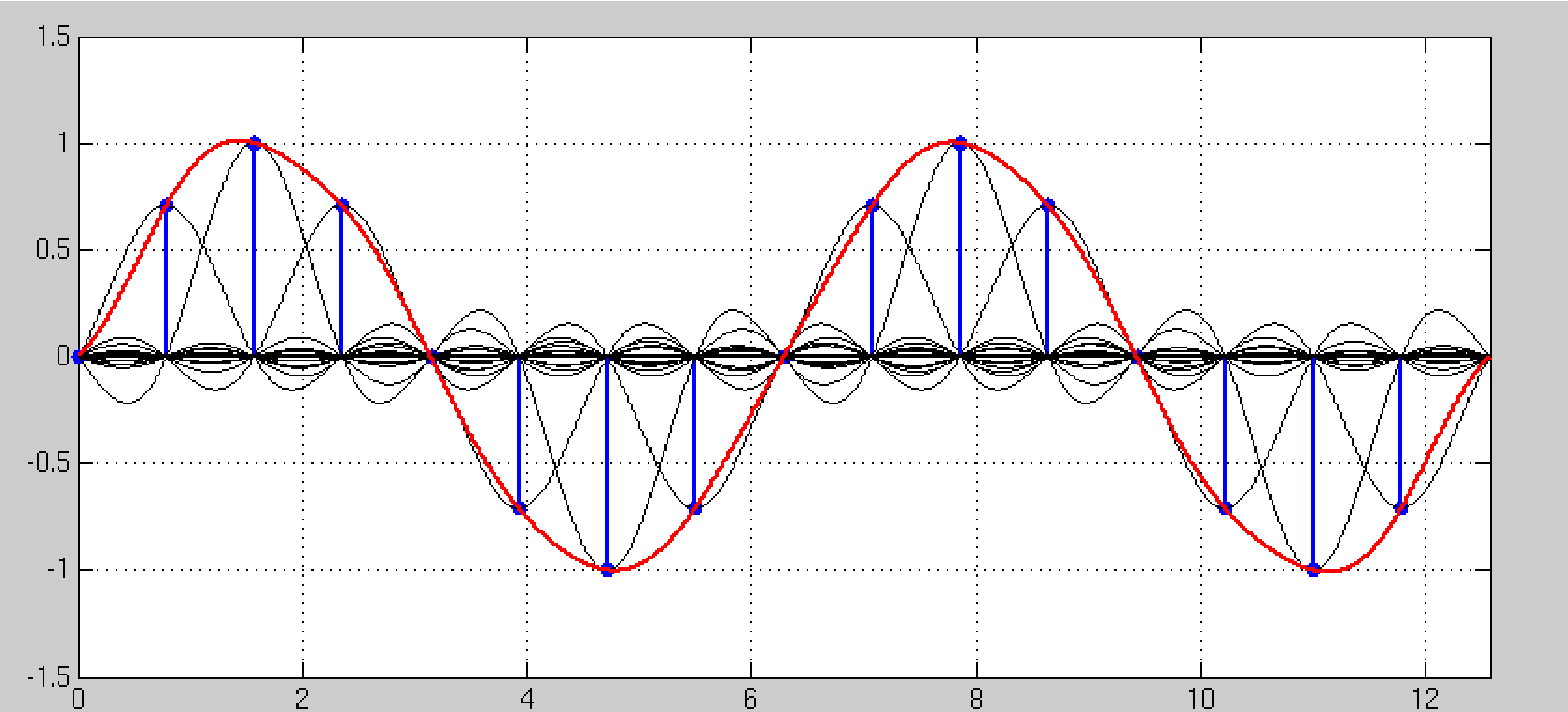

1.5
1
0.5
0
-0.5
-1
-1.5
0
2
4
6
8
10
12

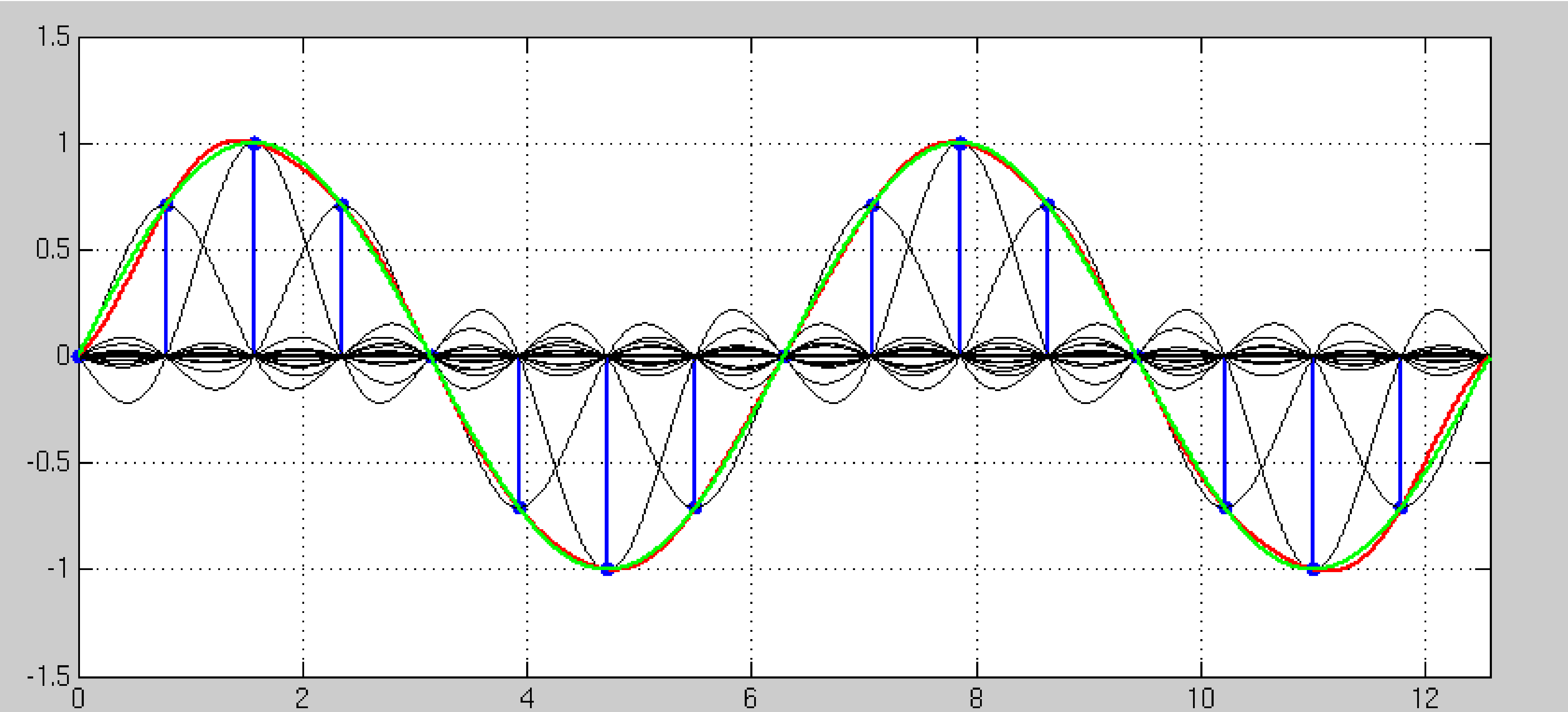


1.5
1
0.5
0
-0.5
-1
-1.5
0
2
4
6
8
10
12

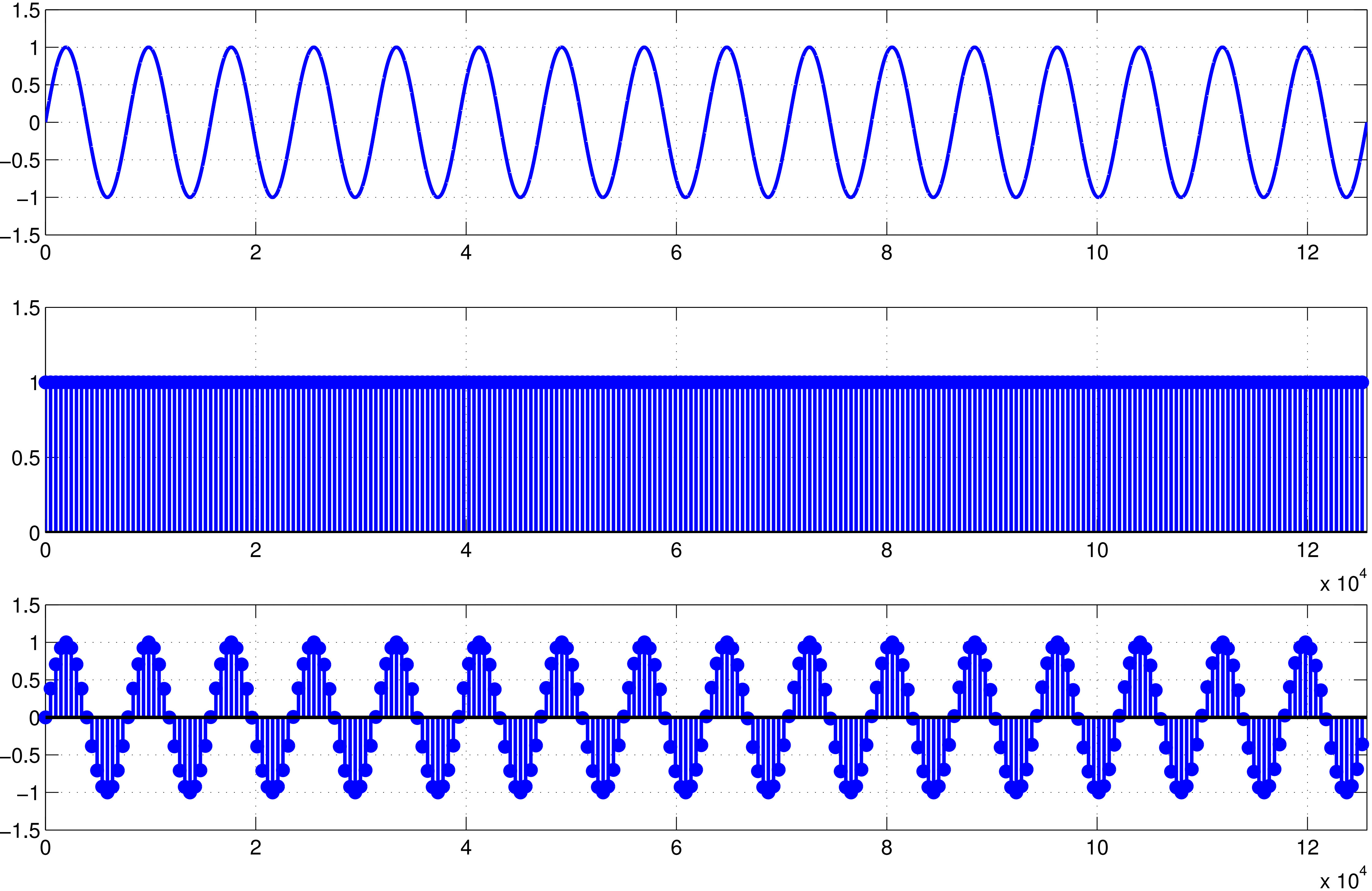


1.5
1
0.5
0
-0.5
-1
-1.5
0
2
4
6
8
10
12

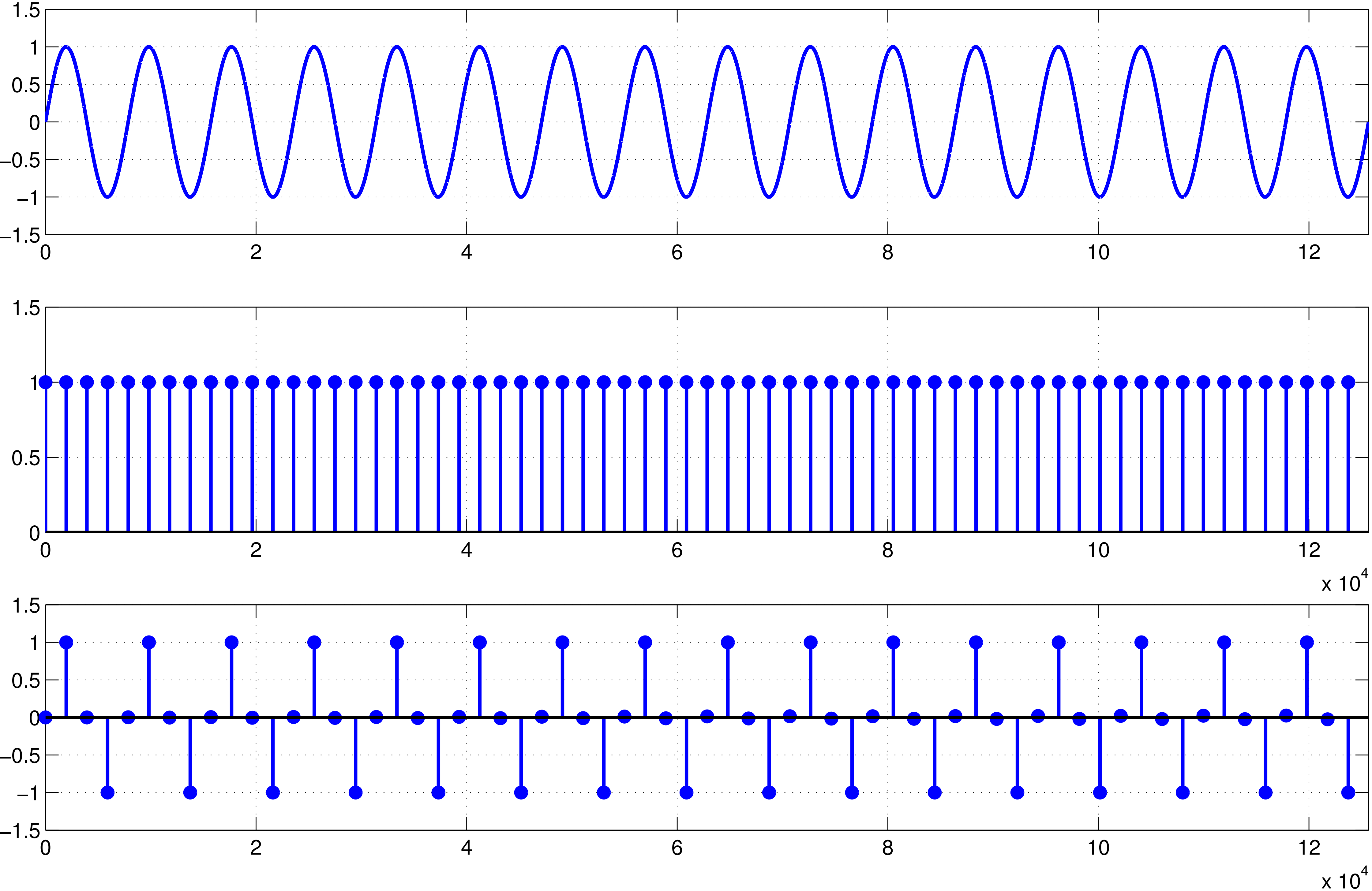

1.5
1
0.5
0
-0.5
-1
-1.5
0
2
4
6
8
10
12

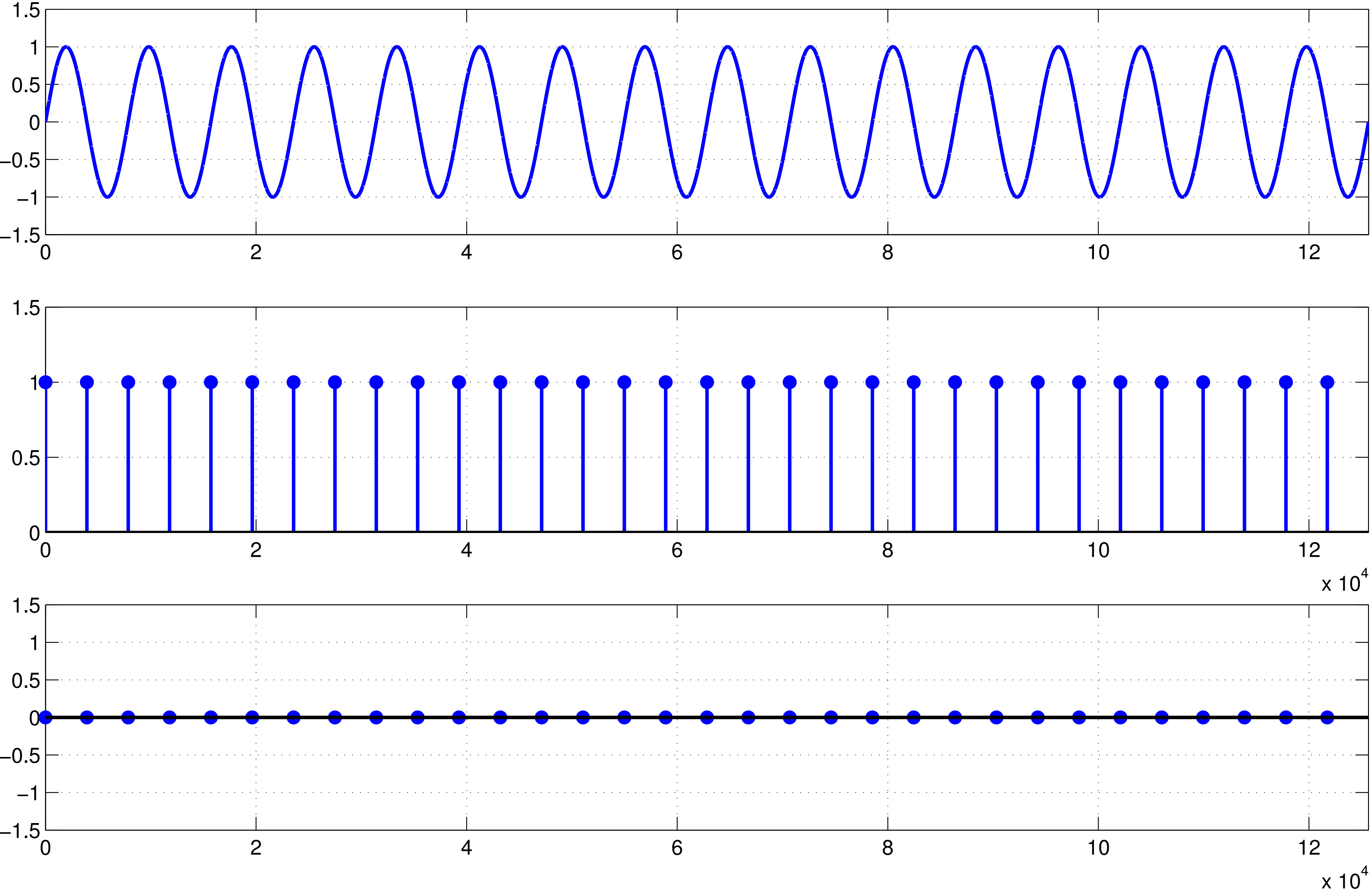


1.5
1
0.5
0
-0.5
-1
-1.5
0
2
4
6
8
10
12

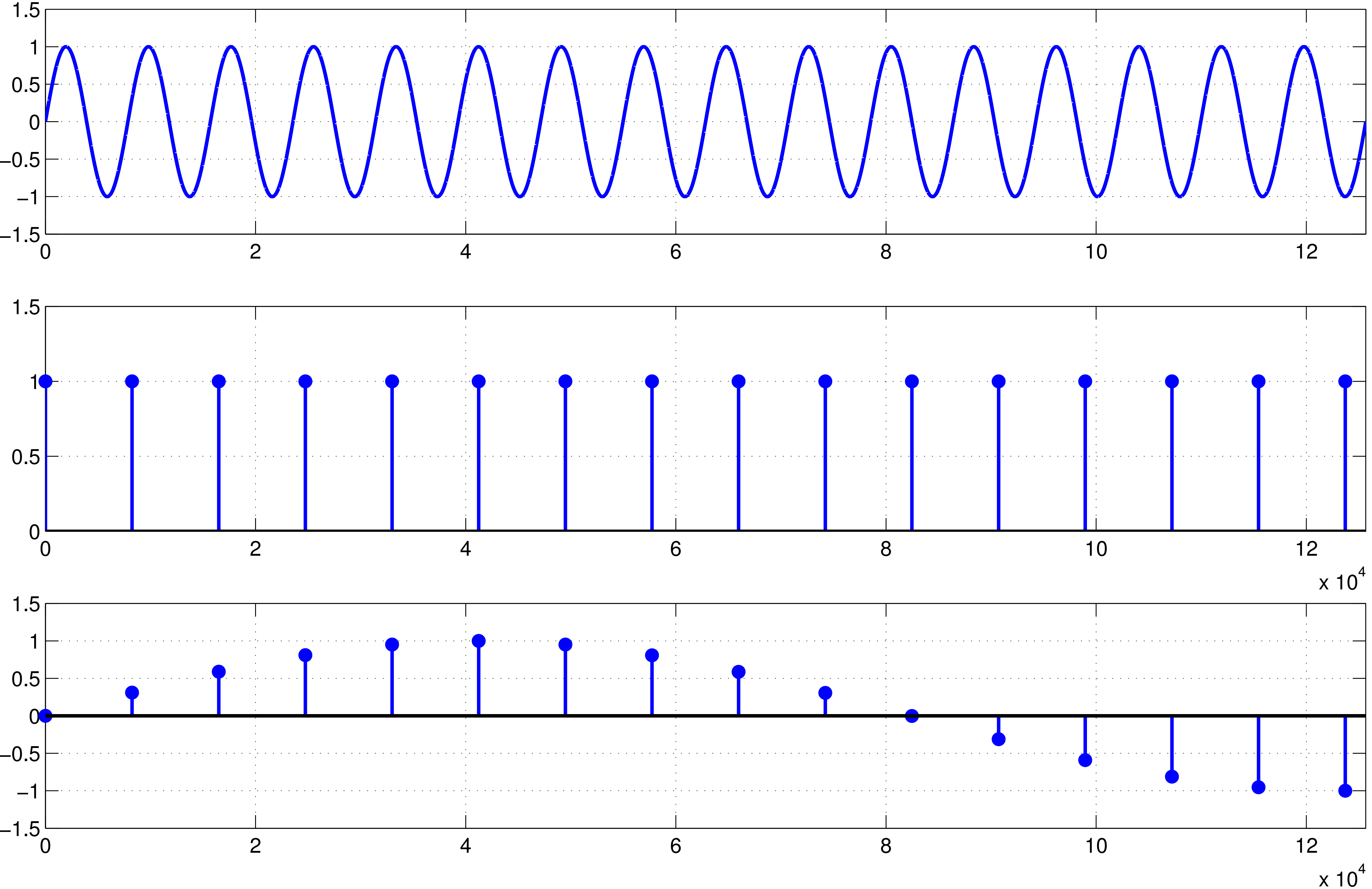


1.5
1
0.5
0
−0.5
−1
−1.5
0
2
4
6
8
10
12
x 10^4
x 10^4

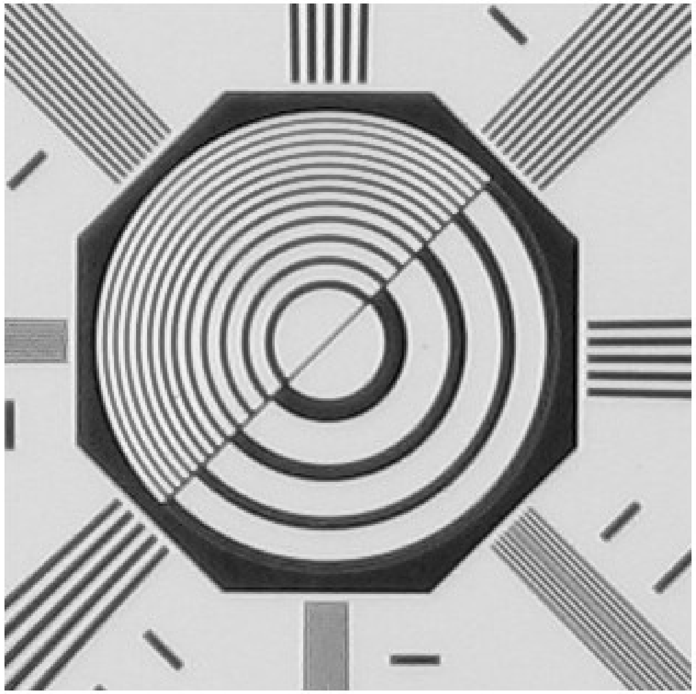

1.5
1
0.5
0
−0.5
−1
−1.5
0
2
4
6
8
10
12
x 10^4

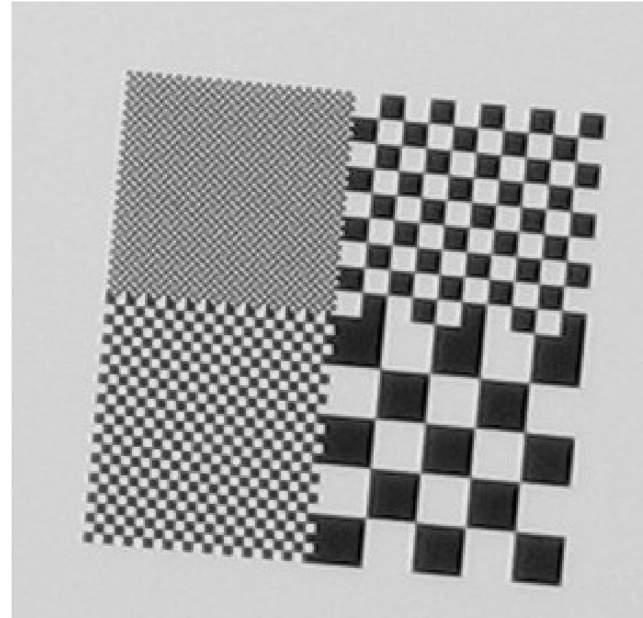


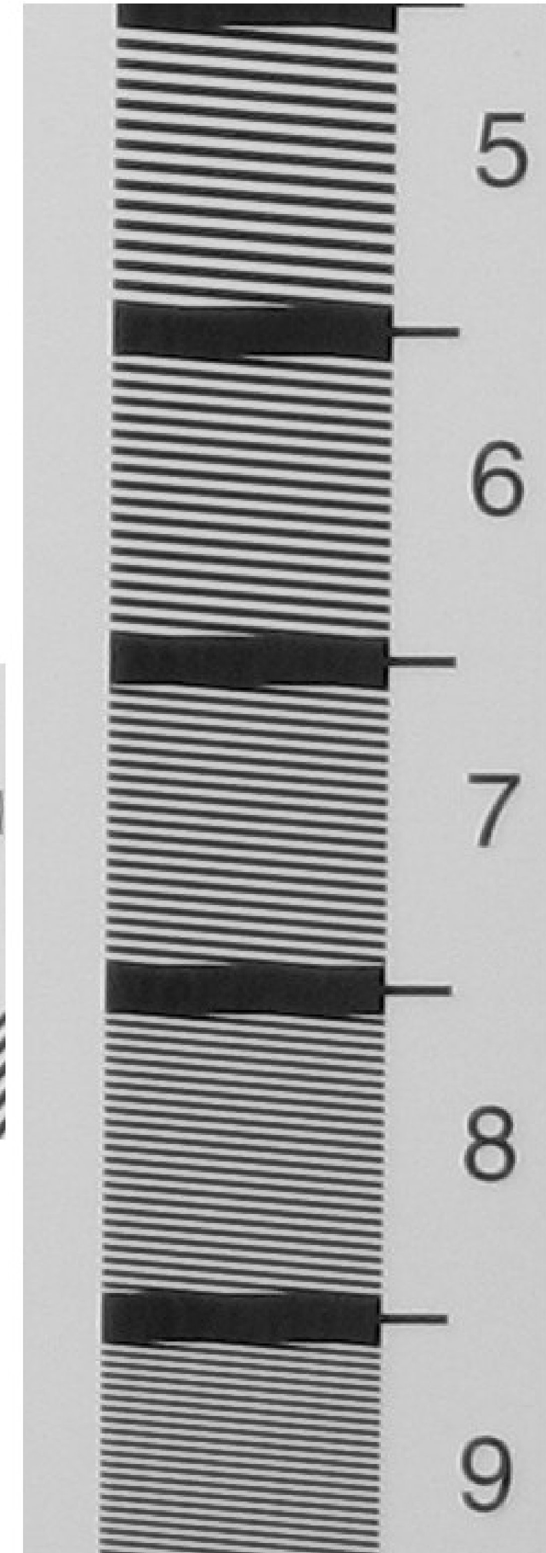
1.5
1
0.5
0
−0.5
−1
−1.5
0
2
4
6
8
10
12
x 10^4
x 10^4

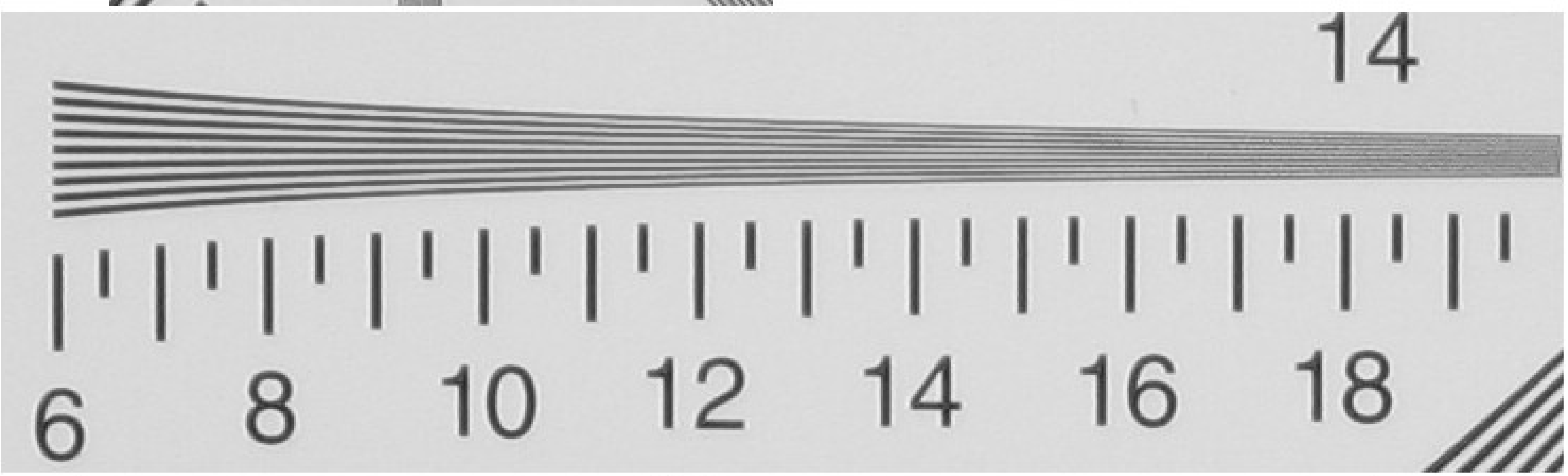


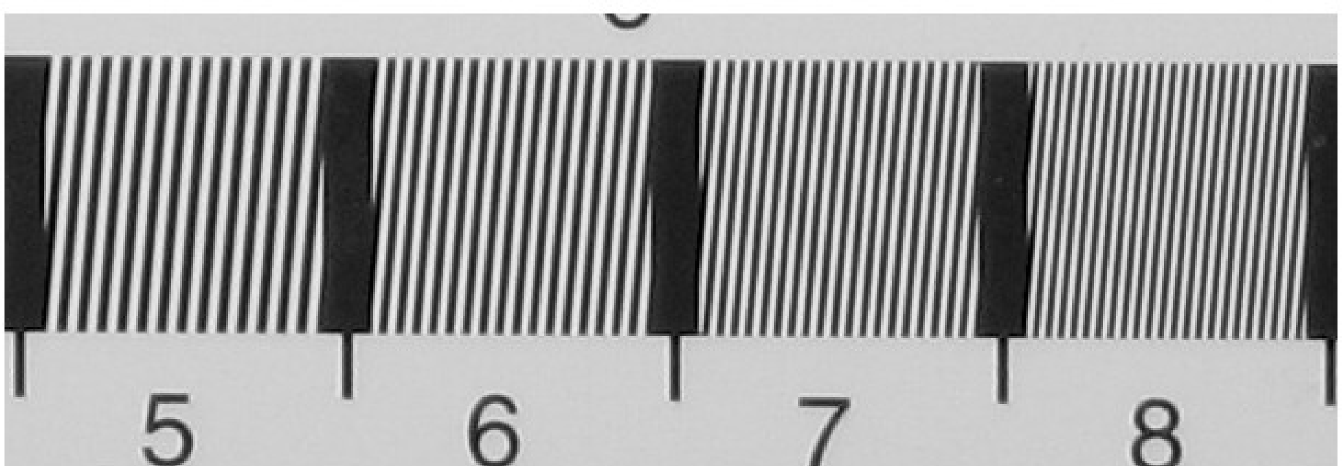
1.5
1
0.5
0
−0.5
−1
−1.5
0
2
4
6
8
10
12
x 10^4

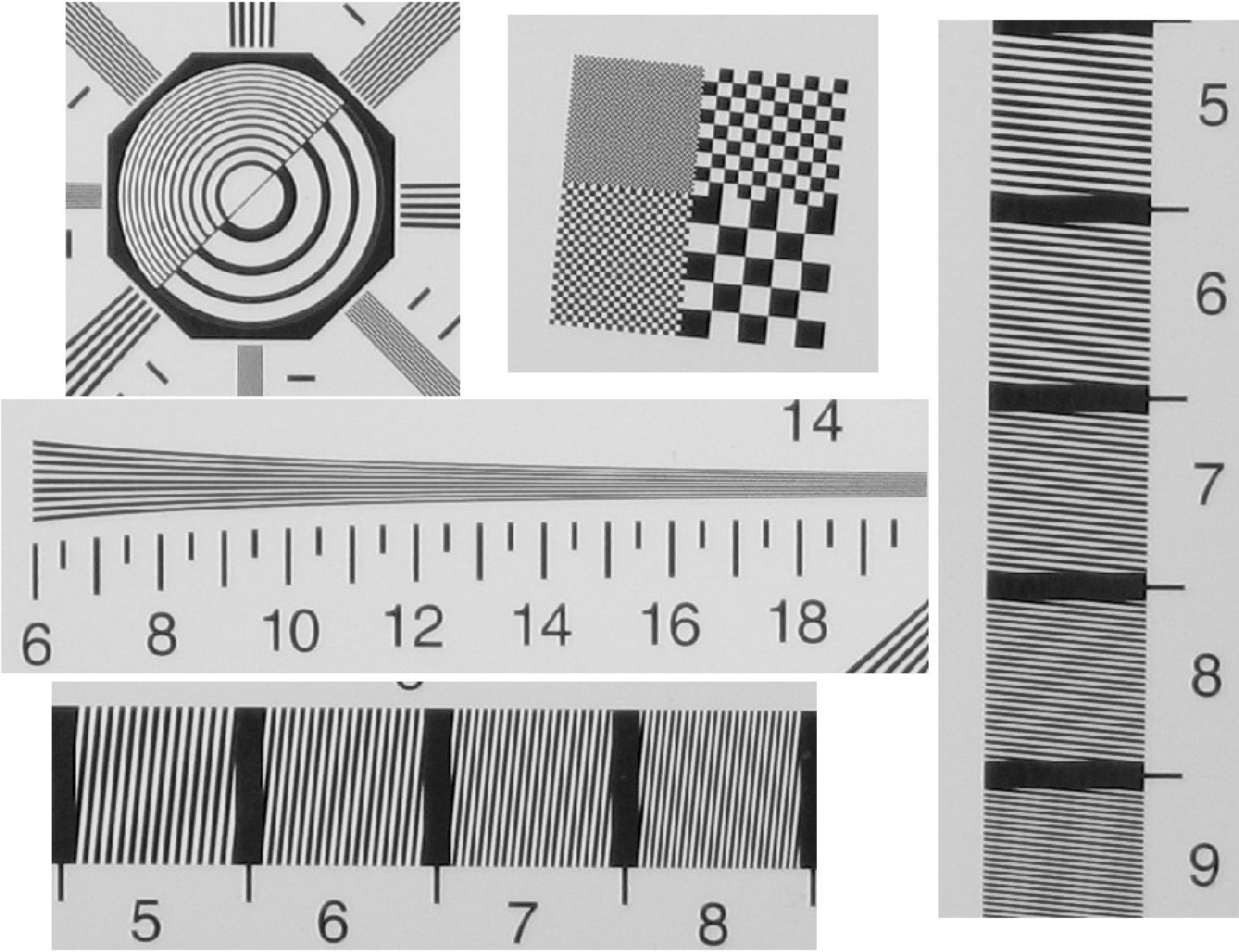




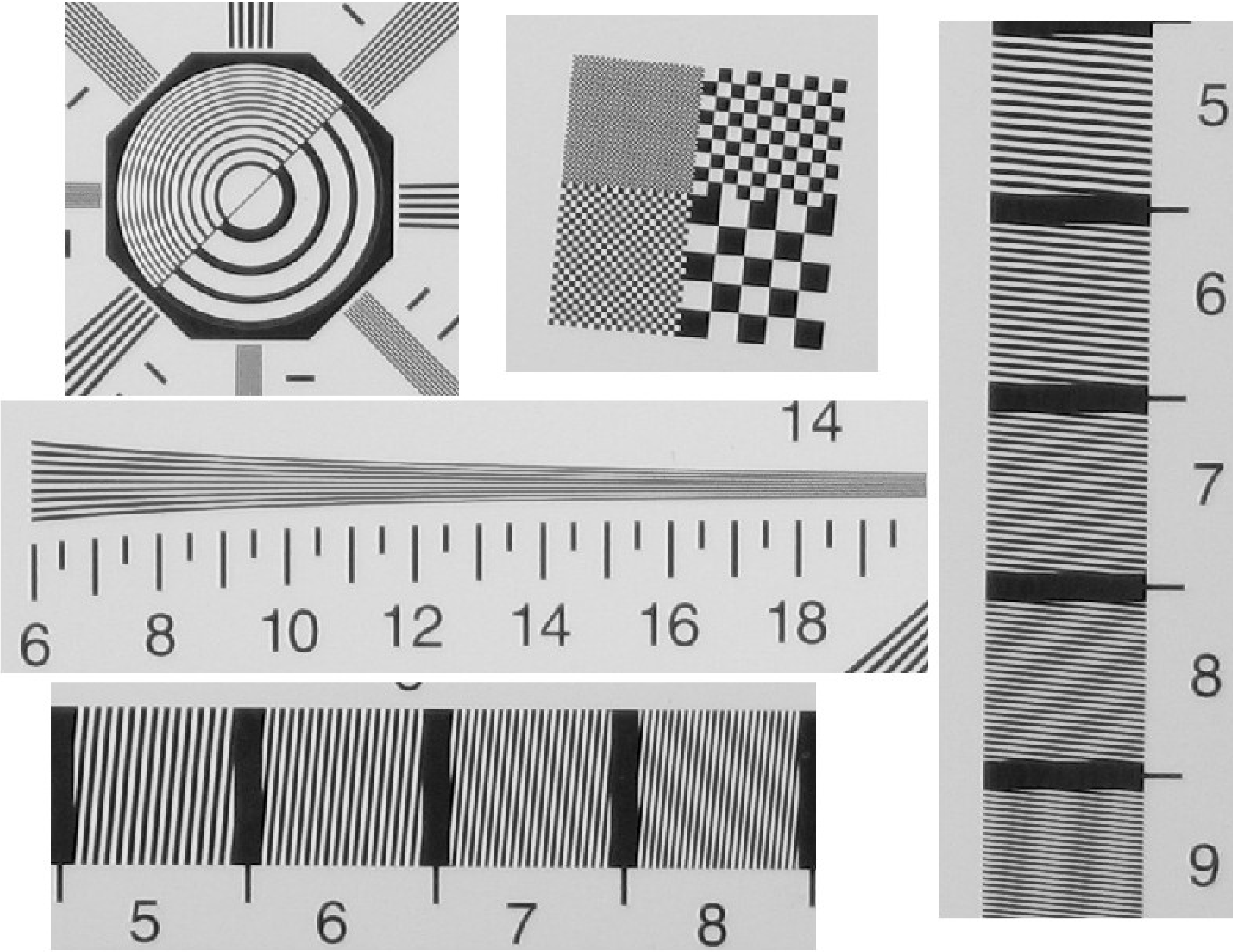

5
6
7
8
9


14
6
8
10
12
14
16
18

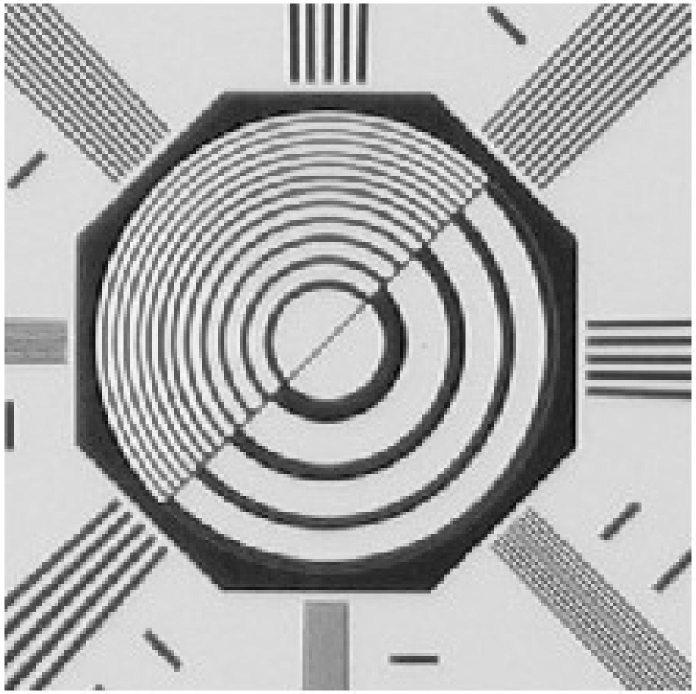

5
6
7
8

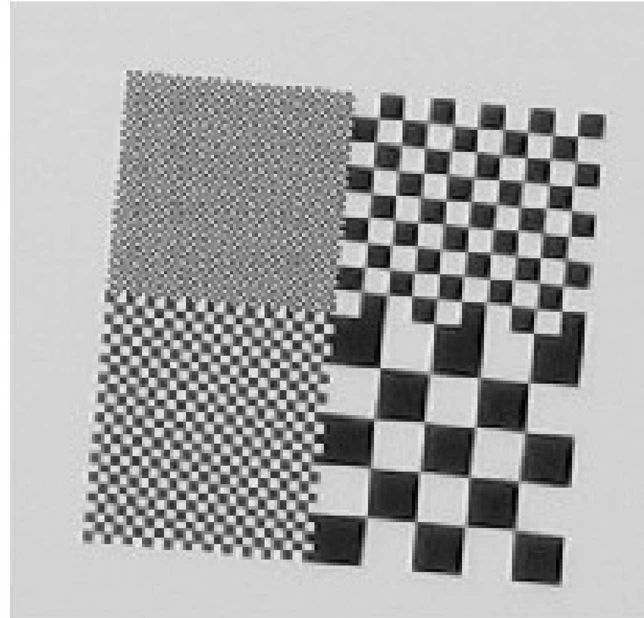


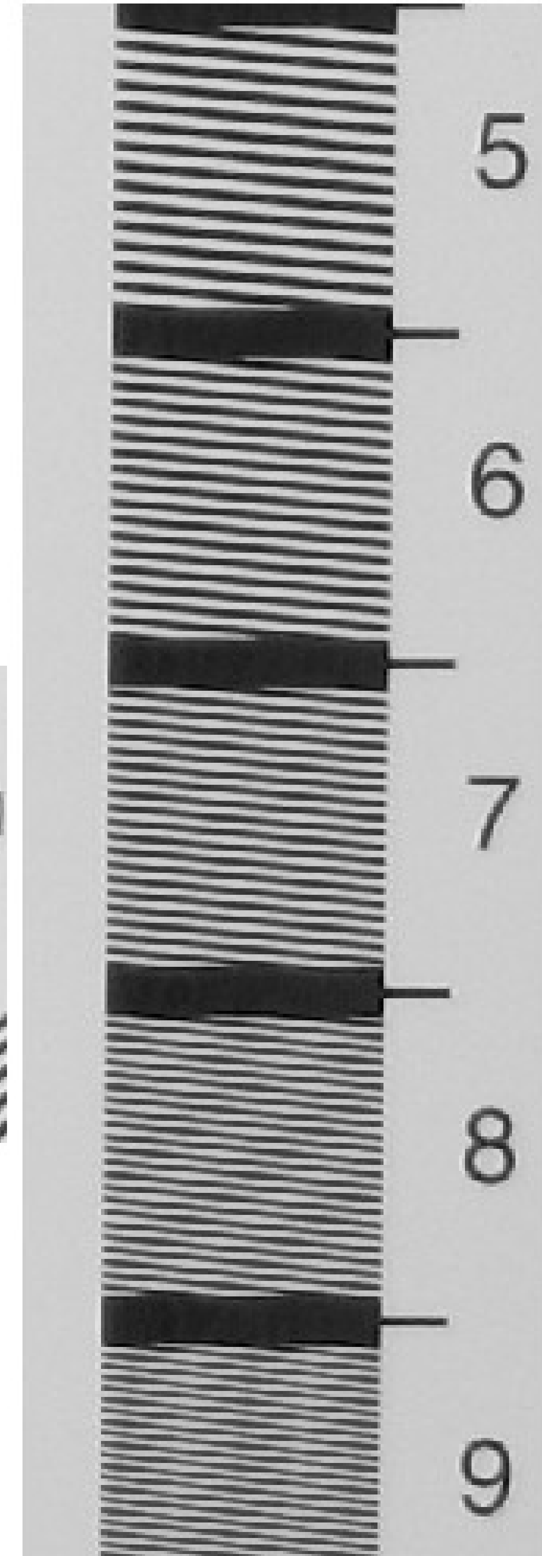
14
6 8 10 12 14 16 18
5 6 7 8
5 6 7 8 9

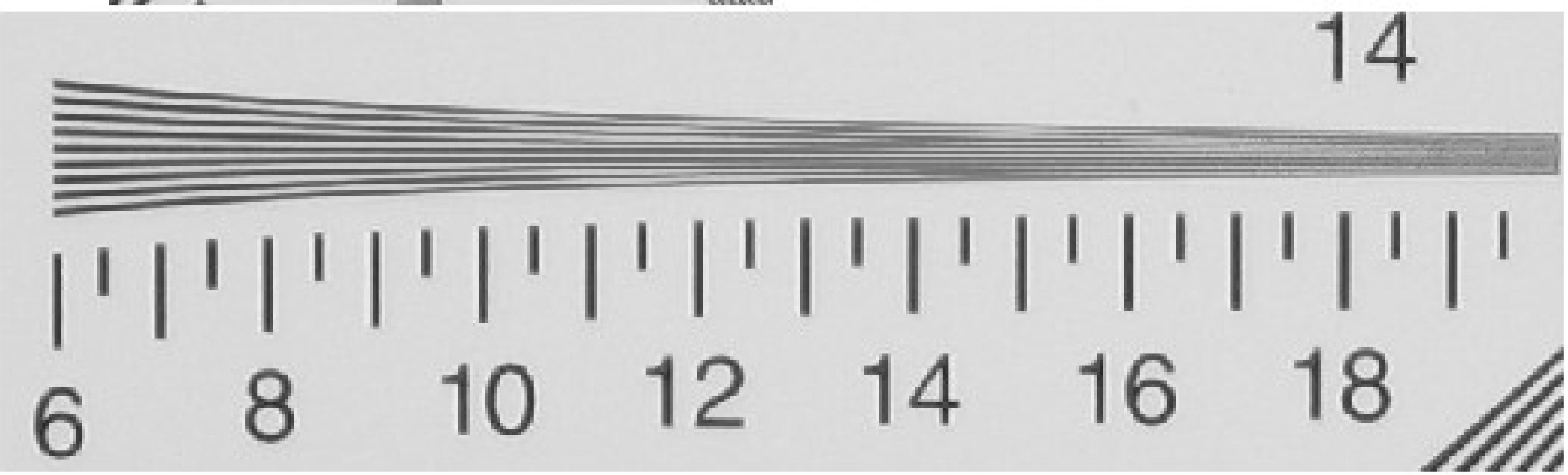


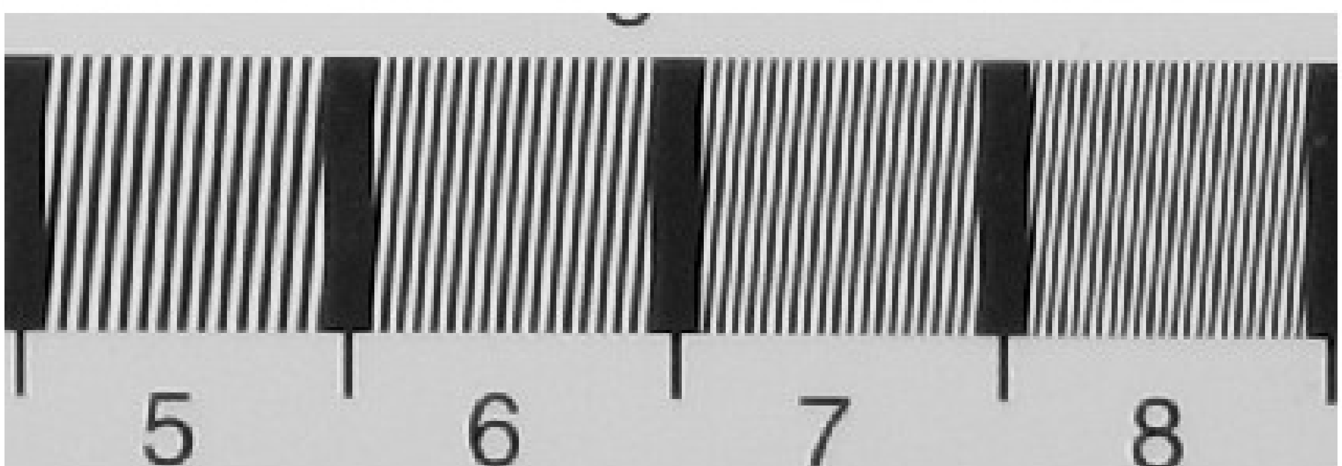
5
6
7
8
9
14
6
8
10
12
14
16
18
5
6
7
8

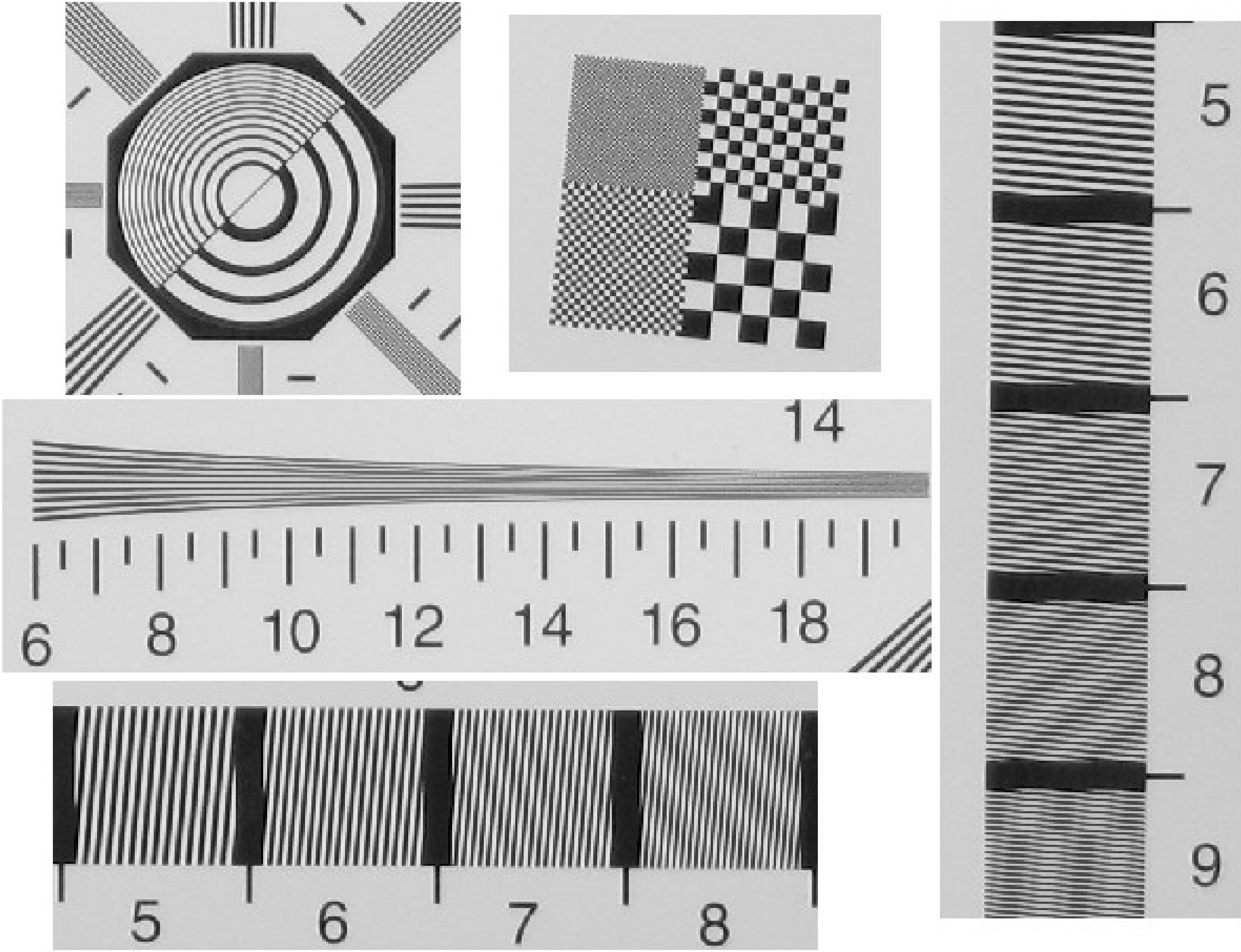


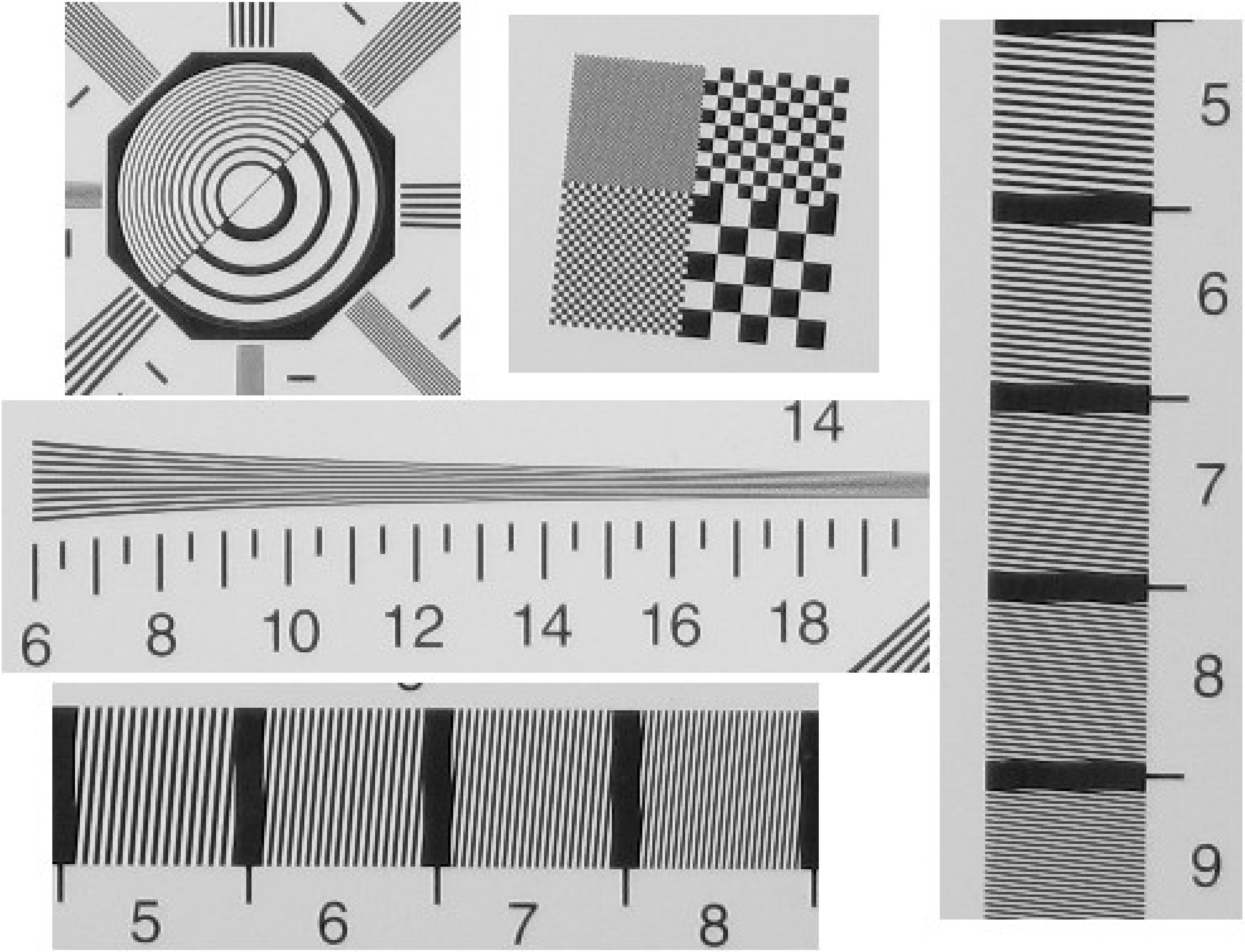


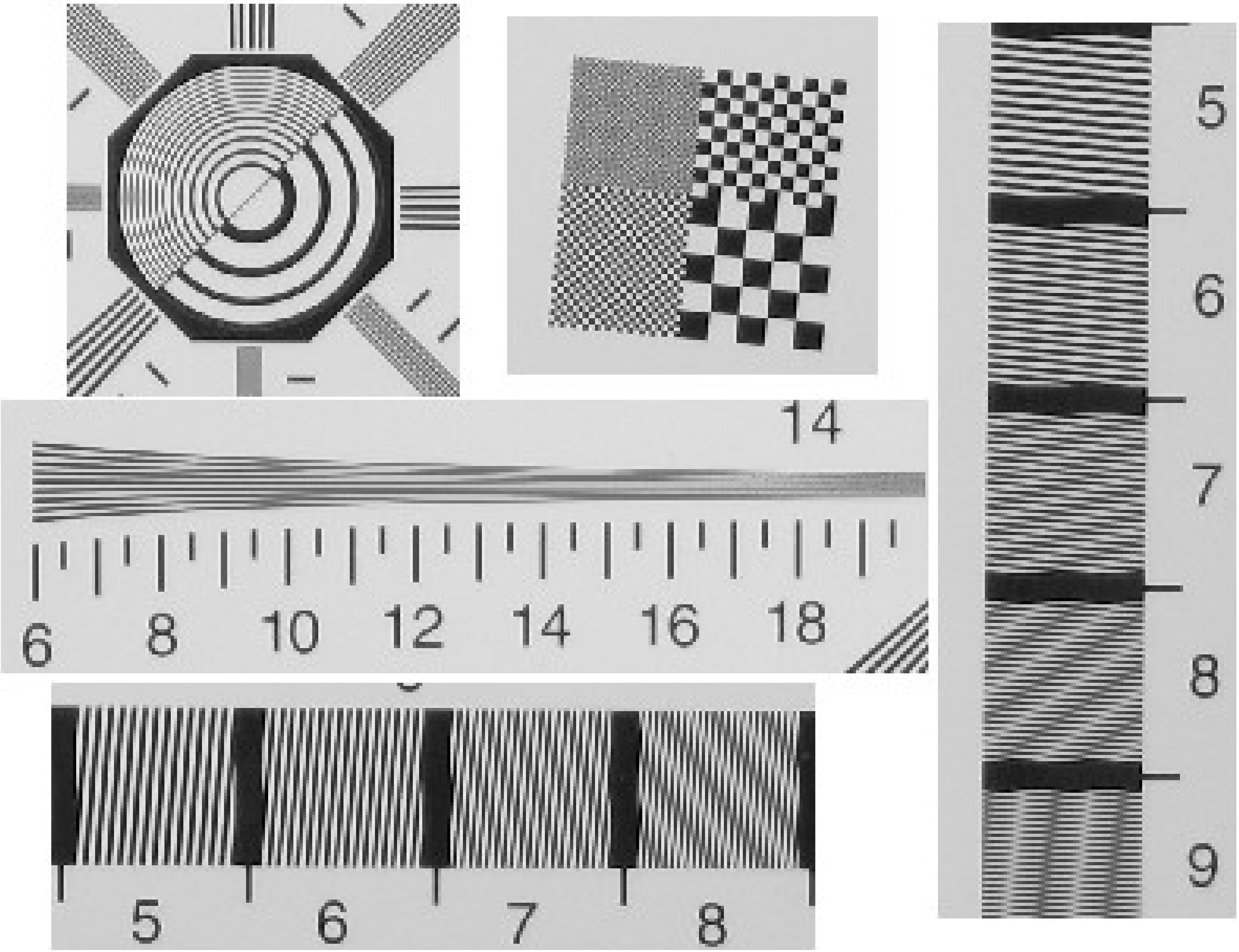

5
6
7
8
9

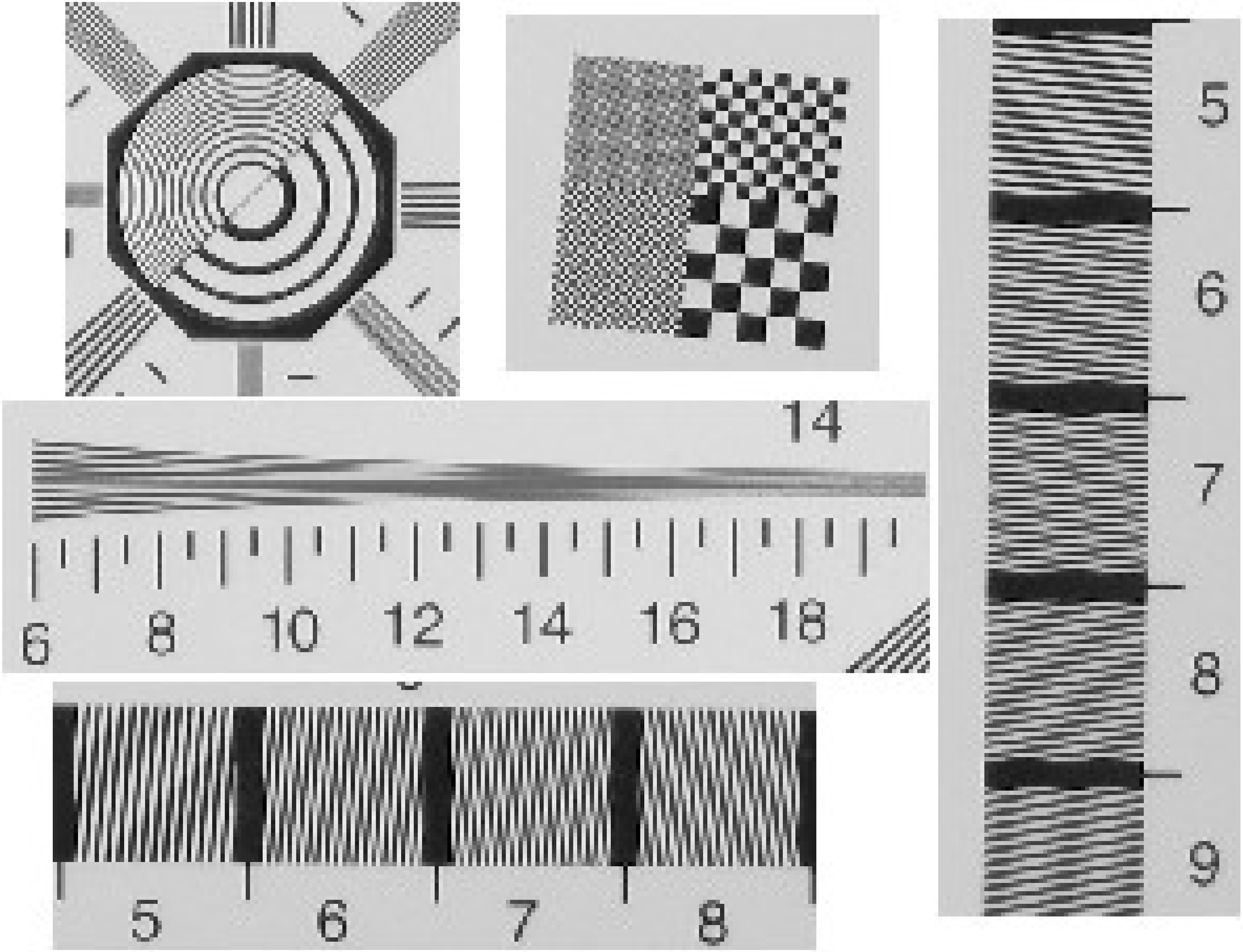

14
6
8
10
12
14
16
18


5
6
7
8

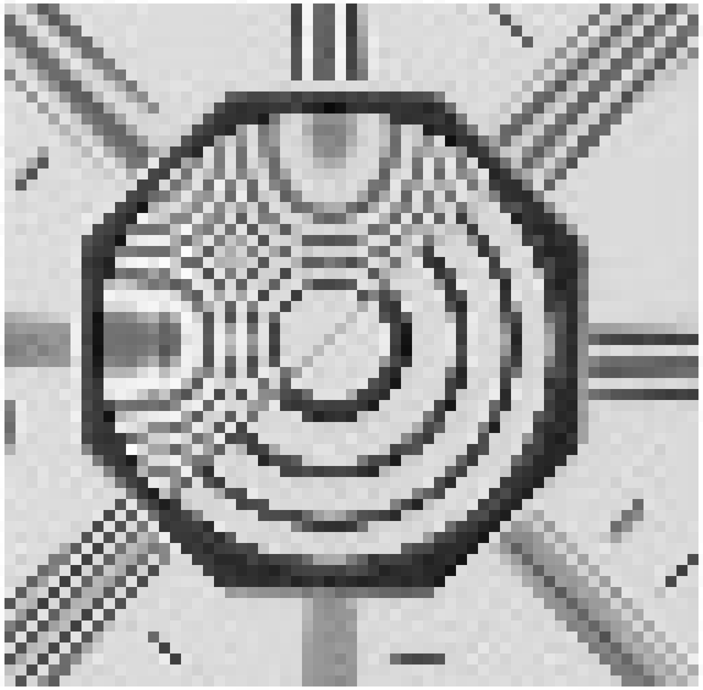

5
6
7
8
9
14
6
8
10
12
14
16
18
5
6
7
8

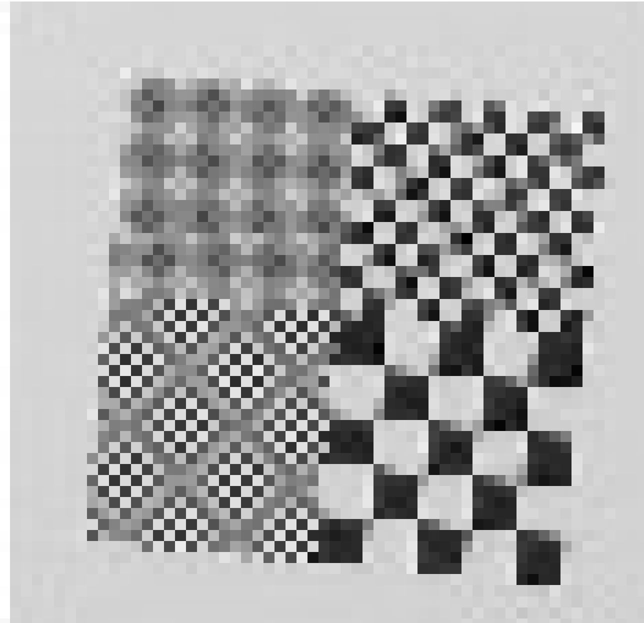

14
6 8 10 12 14 16 18
5 6 7 8
5 6 7 8 9

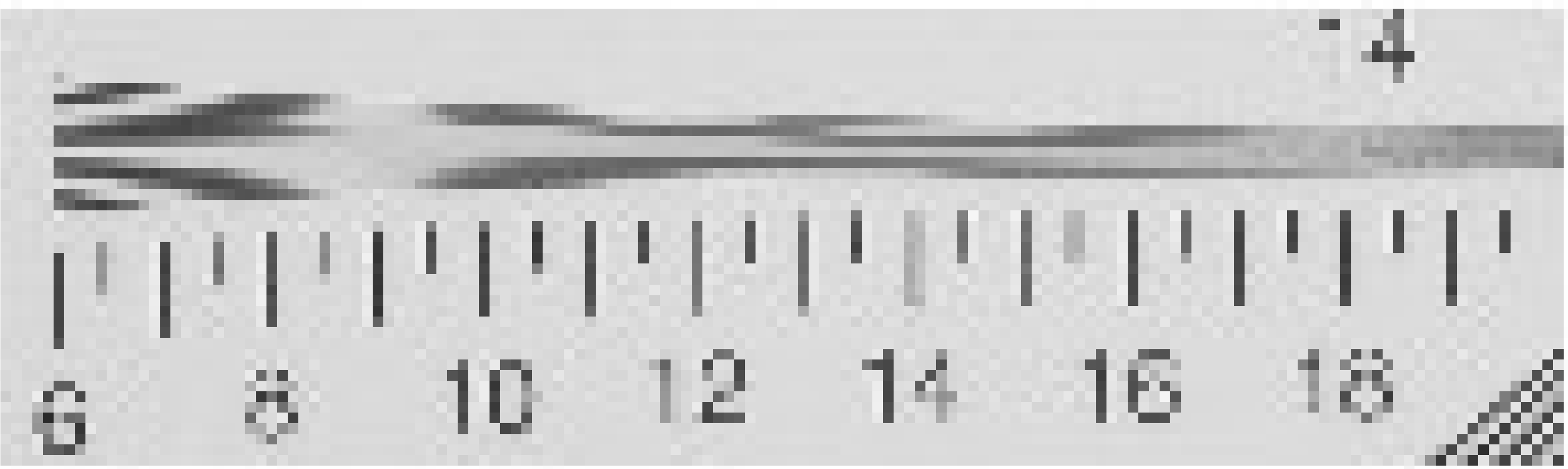

5
6
7
8
9
14
6
8
10
12
14
16
18
5
6
7
8

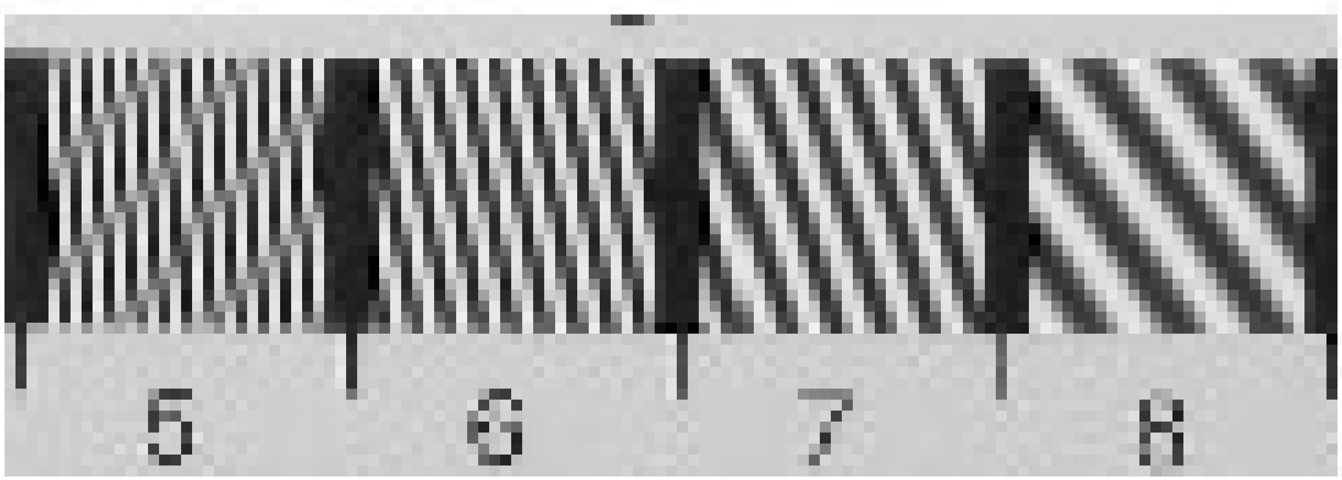


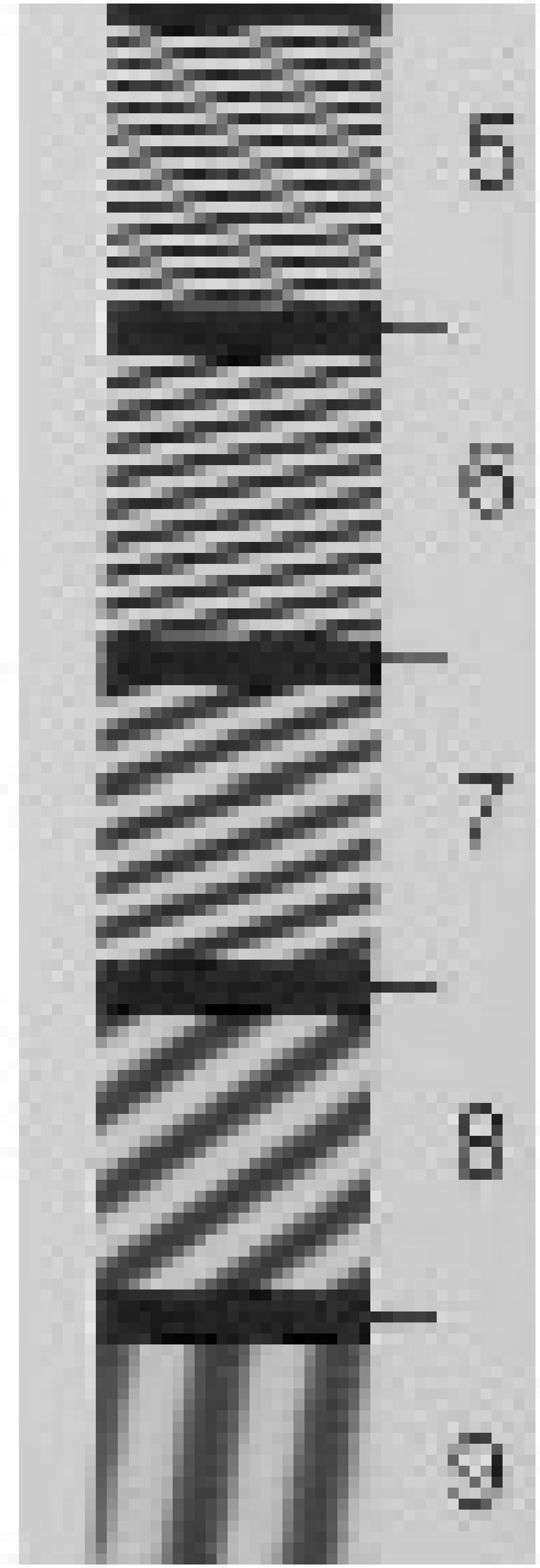
5
6
7
8
9
14
6
8
10
12
14
16
18
5
6
7
8

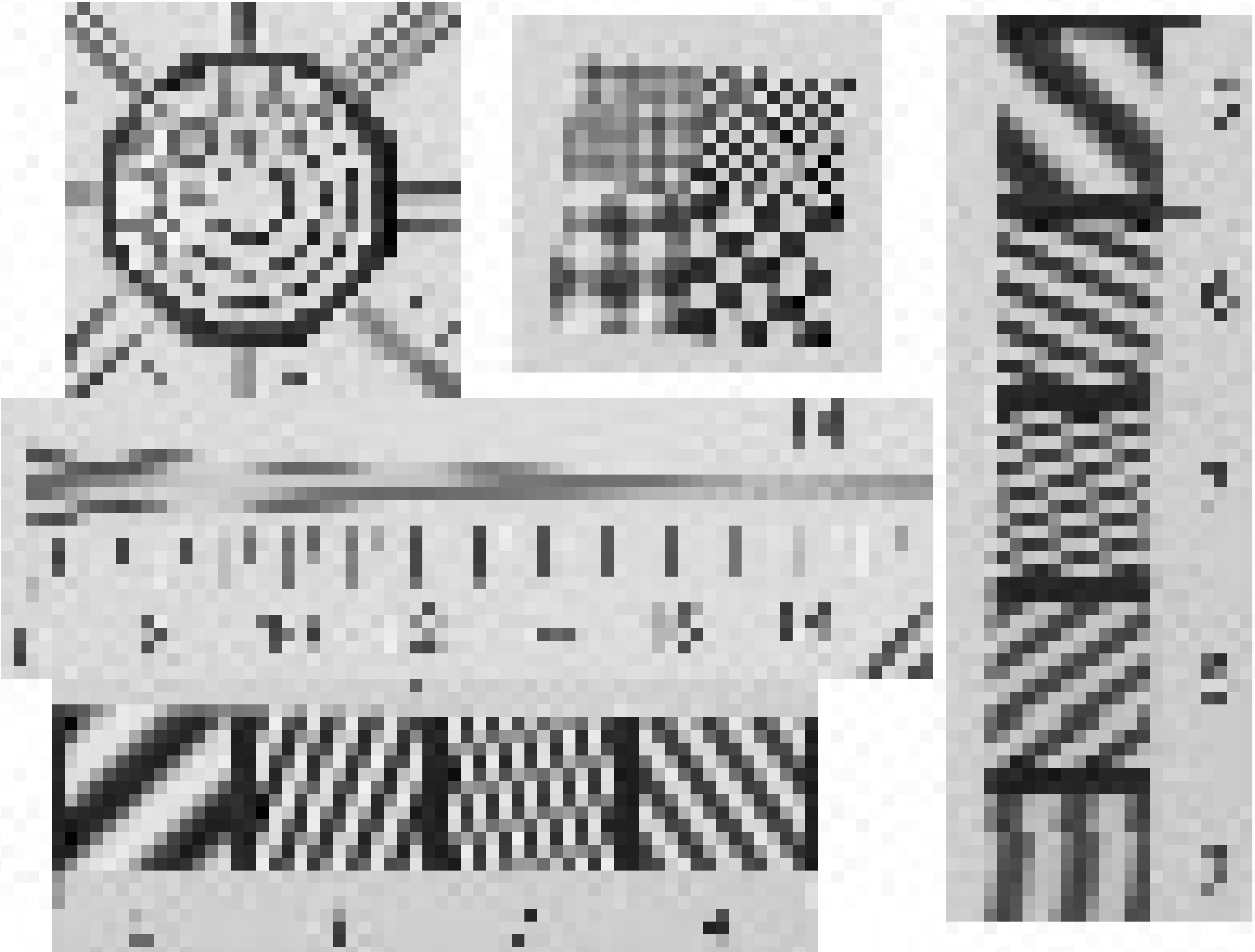


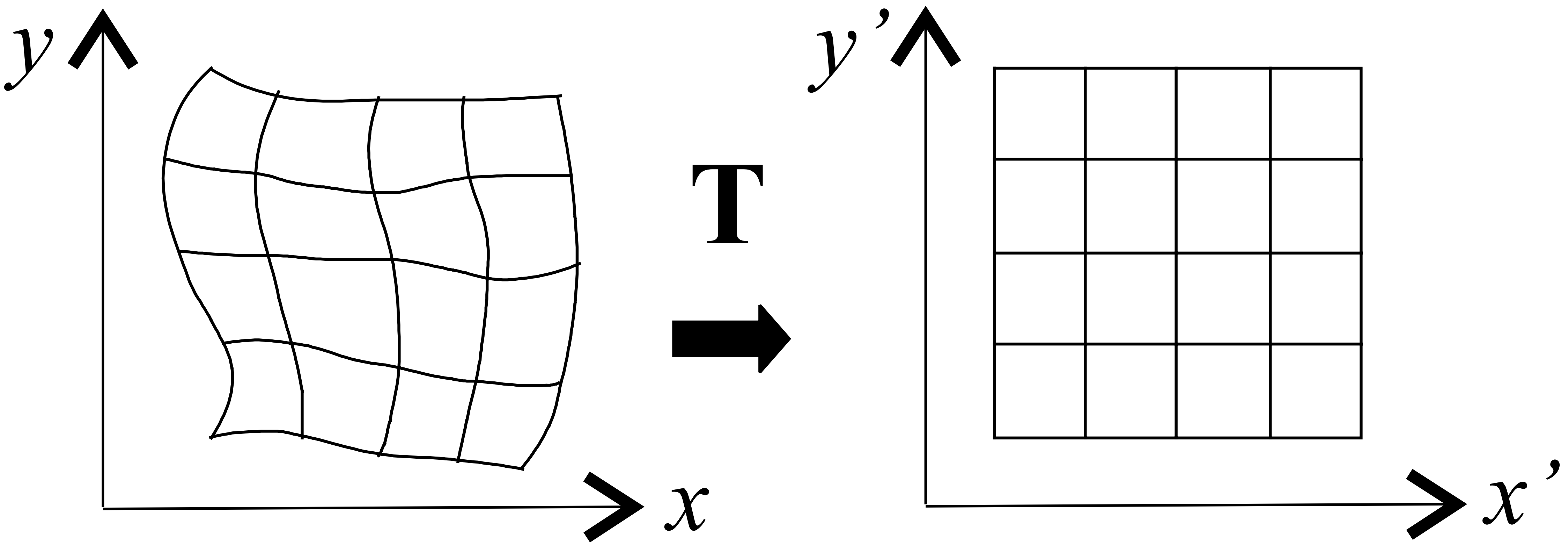


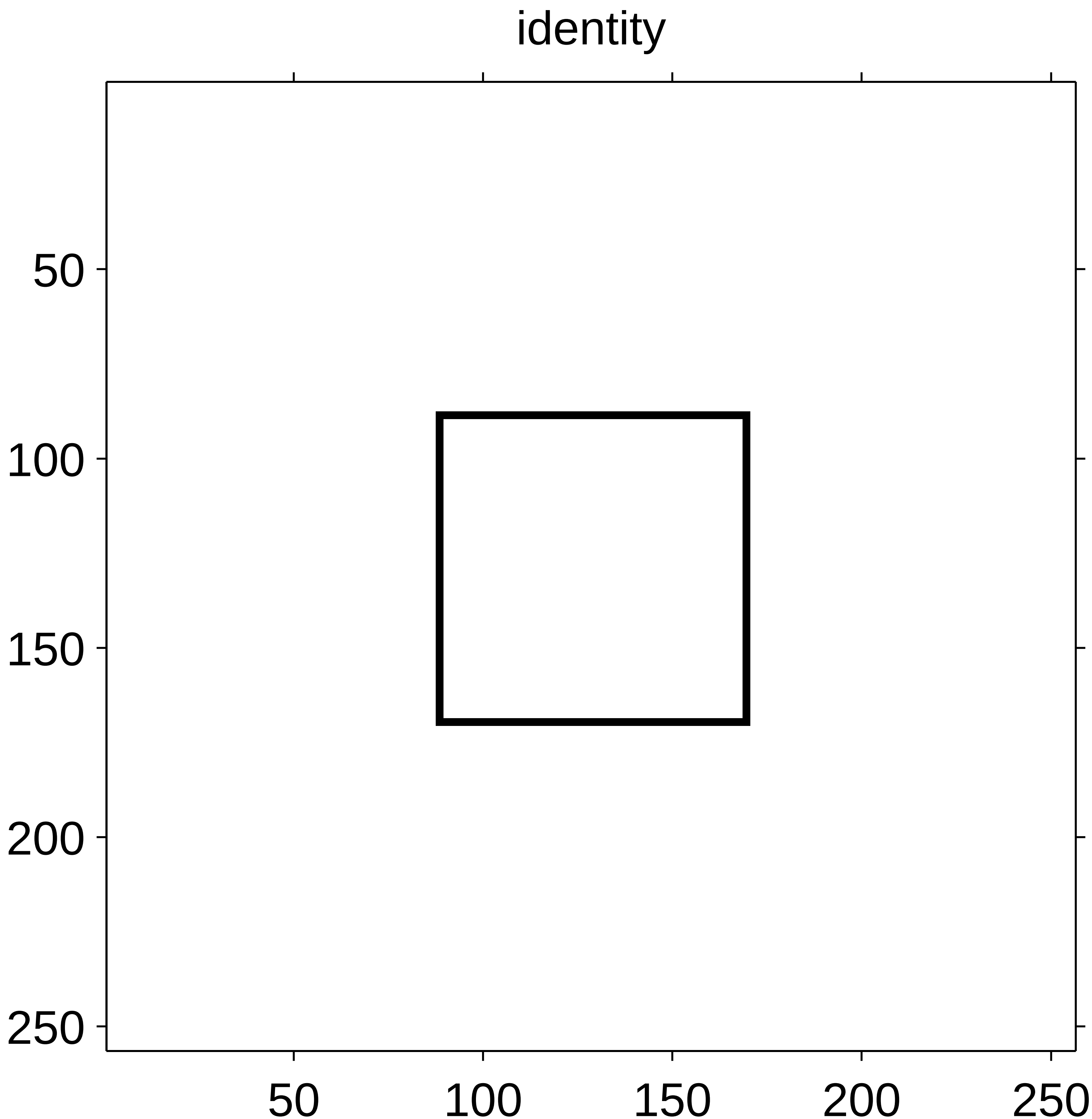

14
6
8
10
12
14
16
18

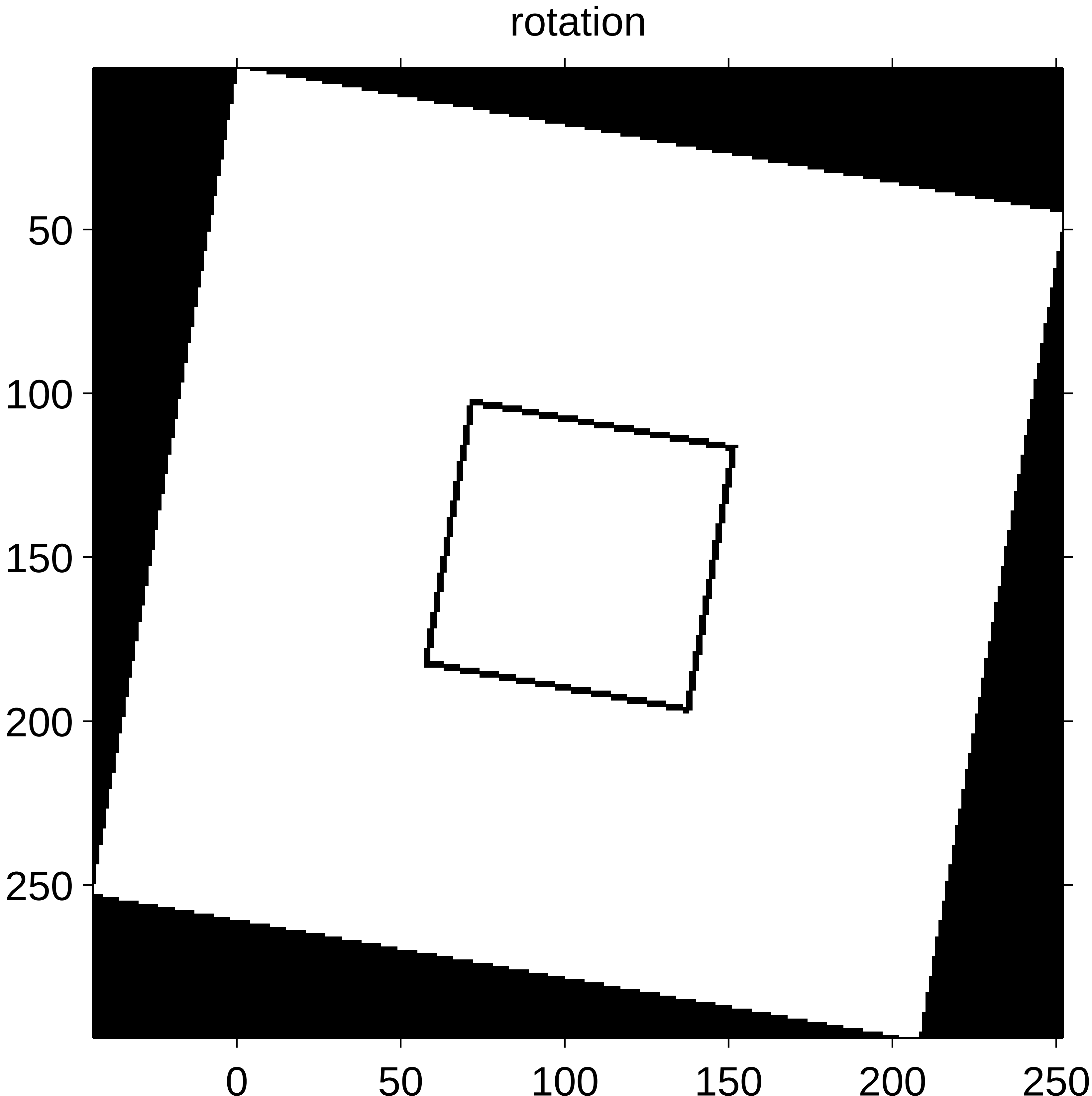

5
6
7
8

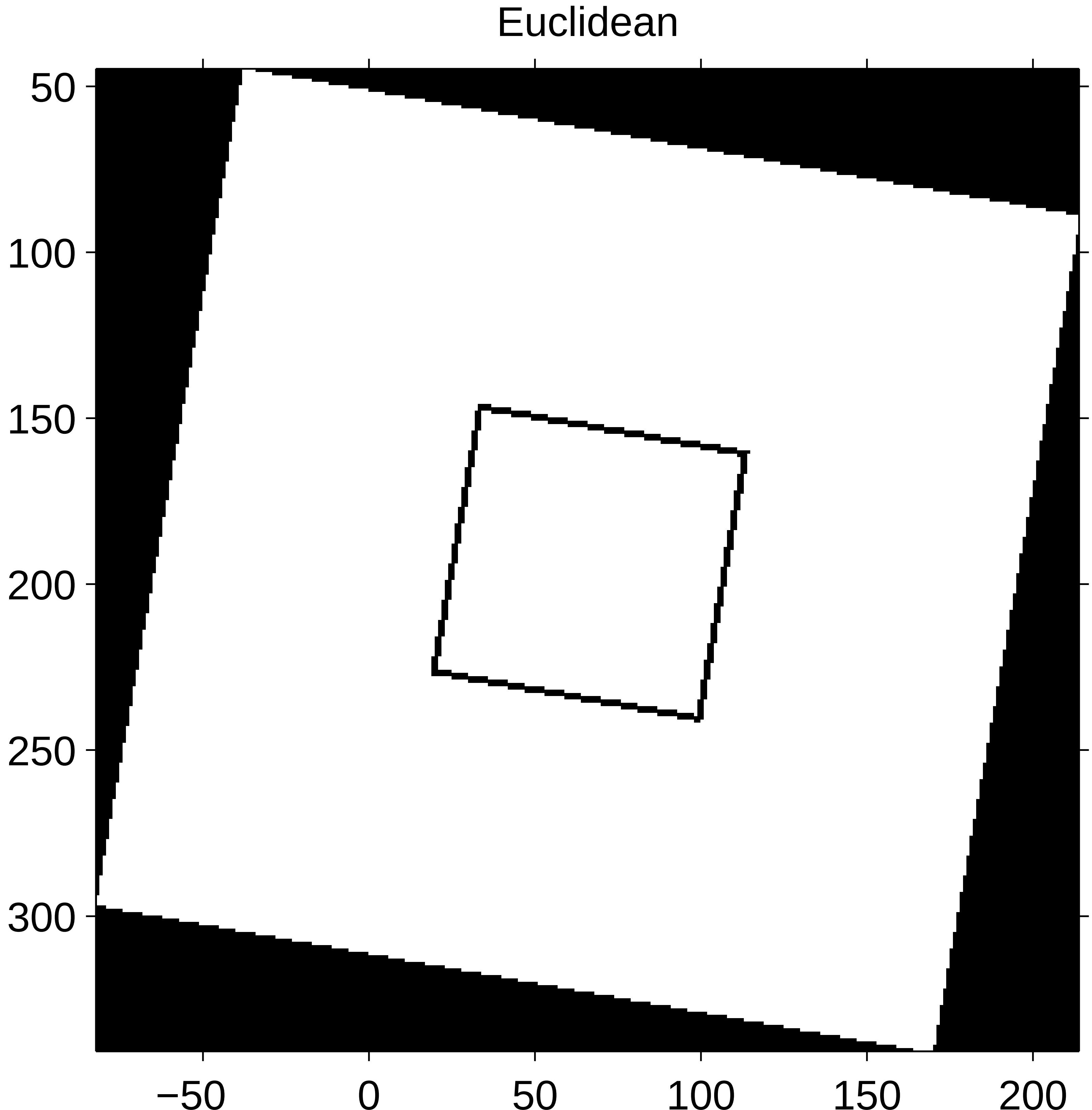

5
6
7
8
9

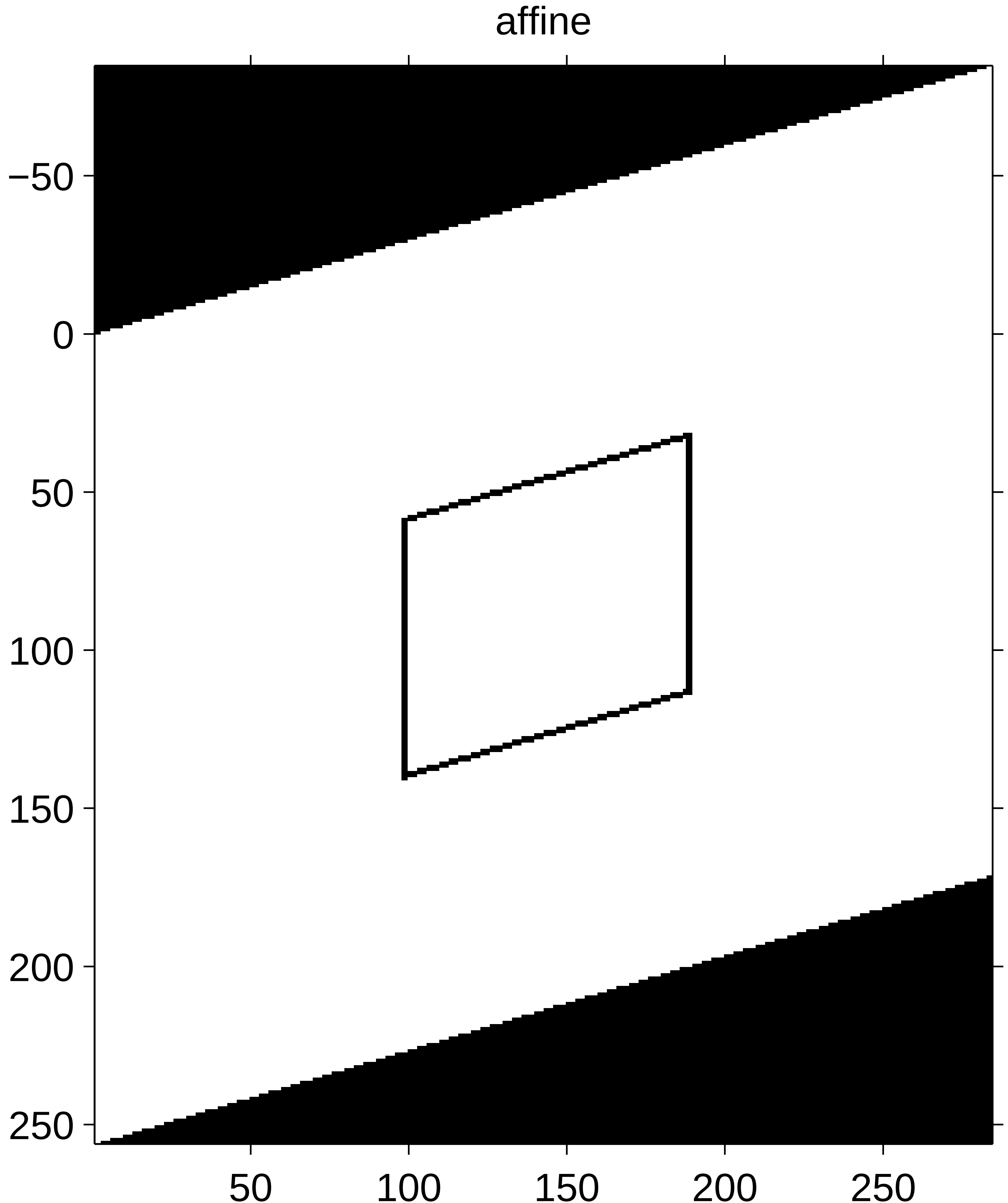


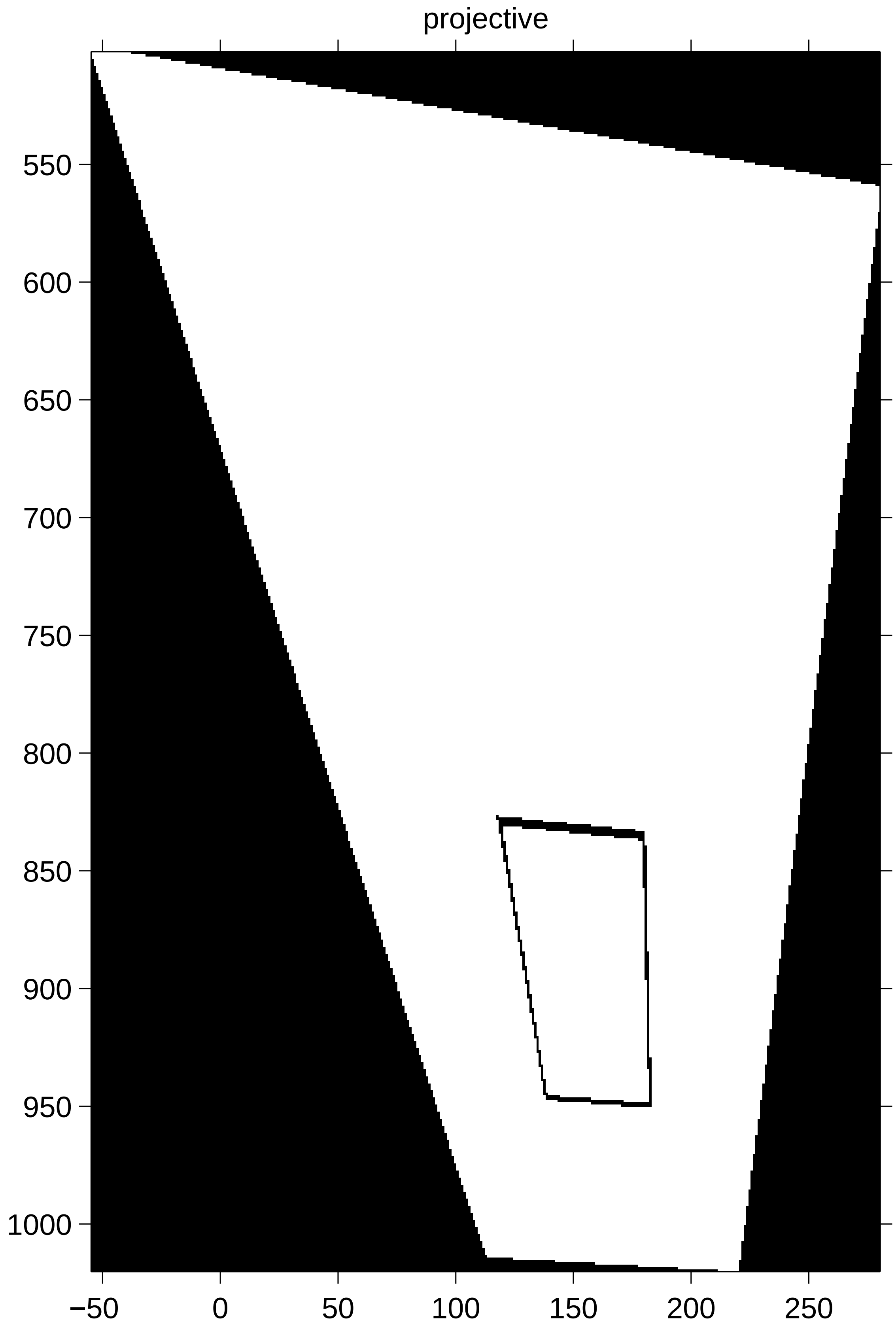

y
x
T
y'
x'


identity
50
100
150
200
250
50
100
150
200
250

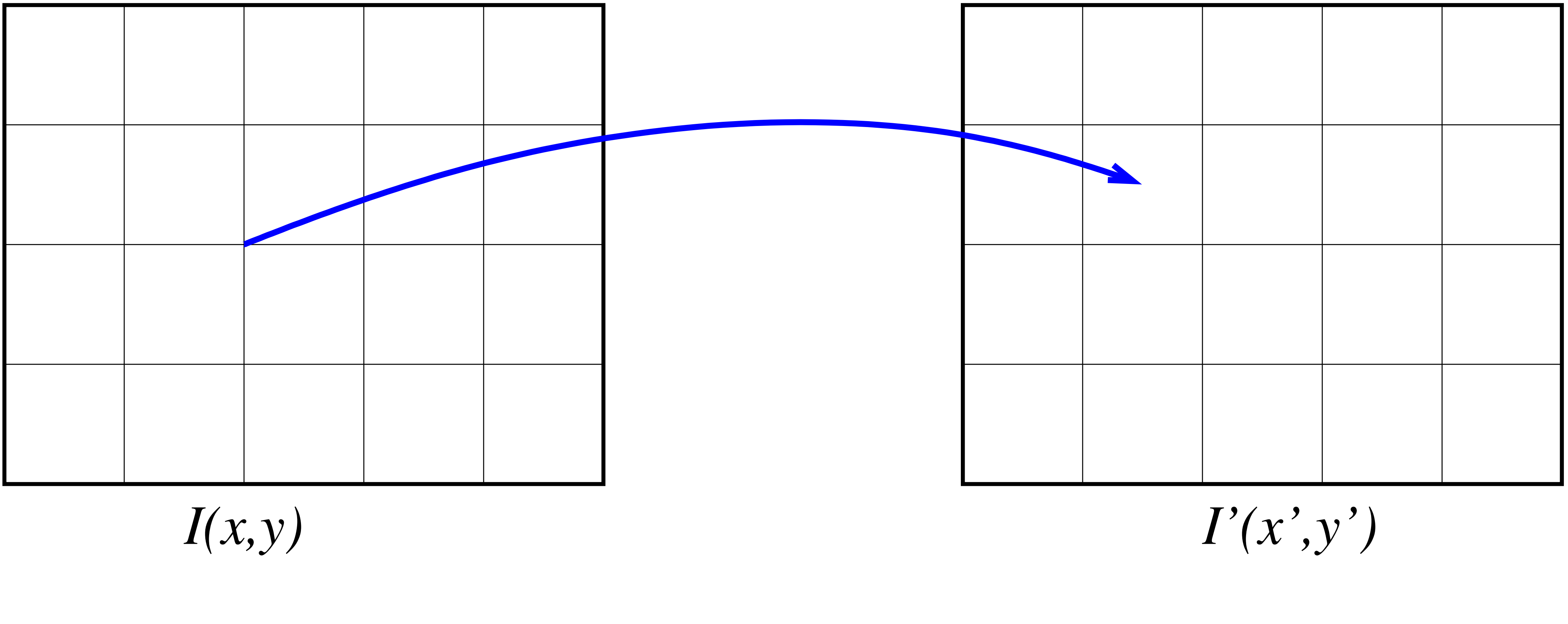

rotation
50
100
150
200
250
0
50
100
150
200
250


Euclidean
50
100
150
200
250
300
−50
0
50
100
150
200

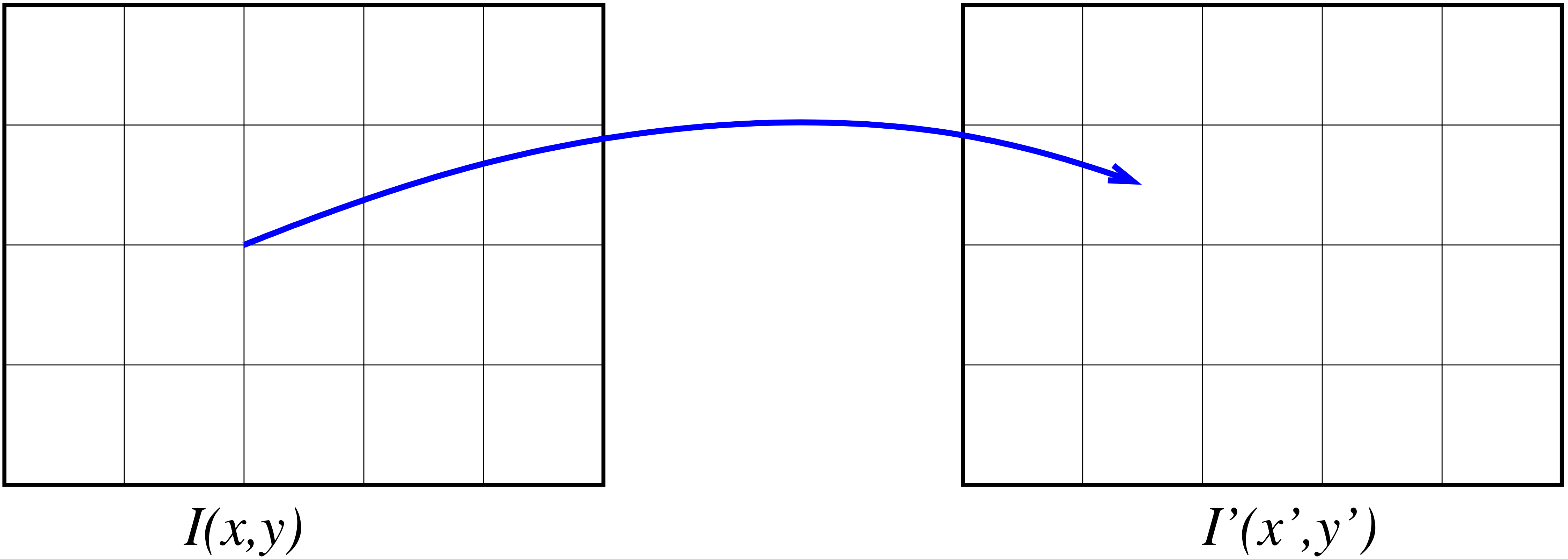

affine
−50
0
50
100
150
200
250
50
100
150
200
250

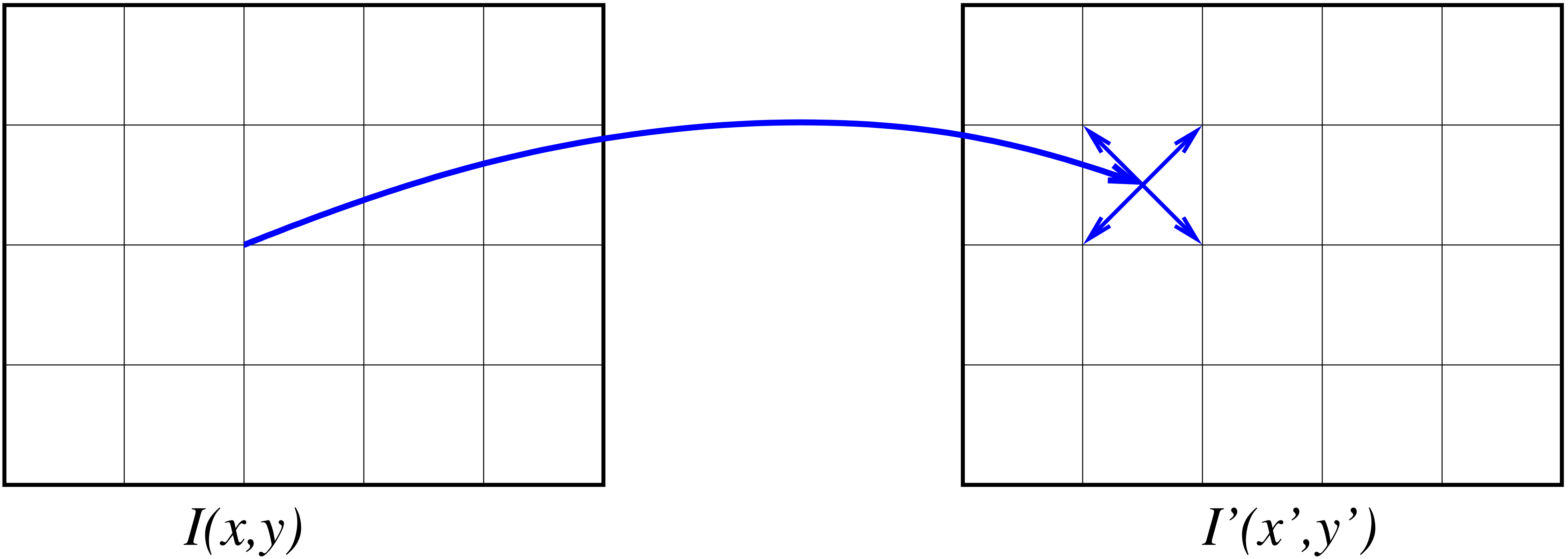


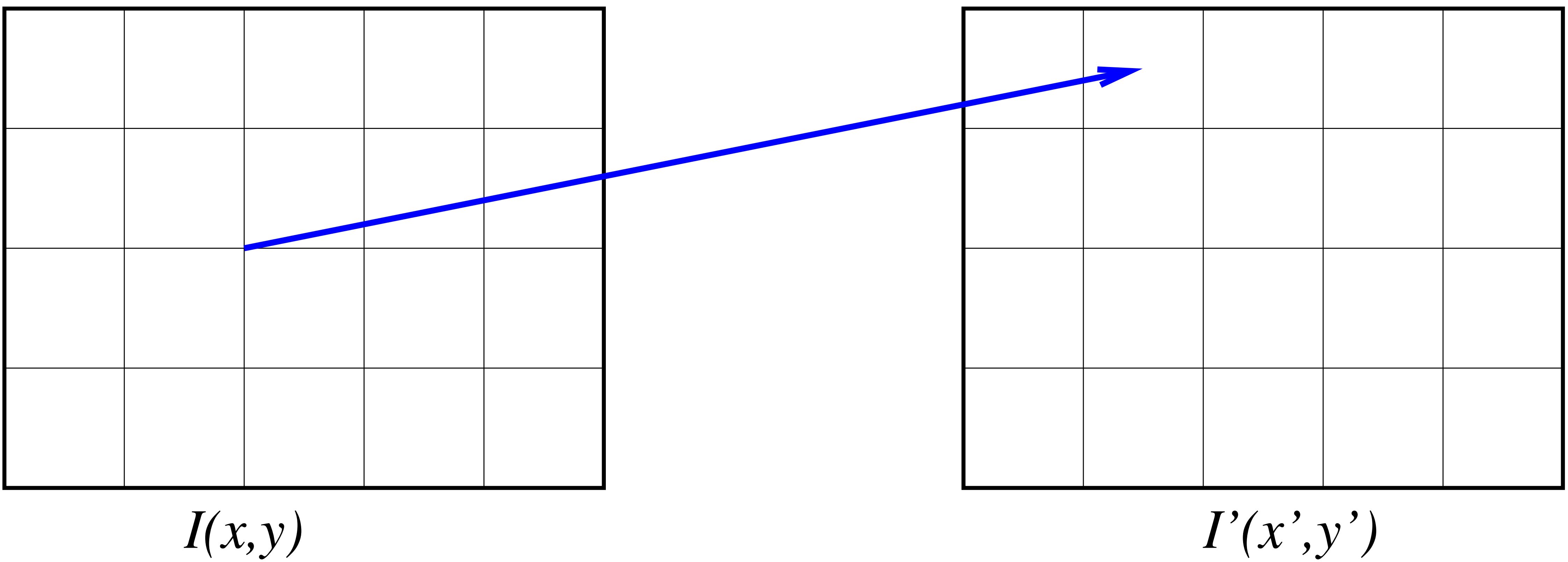
projective
550
600
650
700
750
800
850
900
950
1000
−50
0
50
100
150
200
250

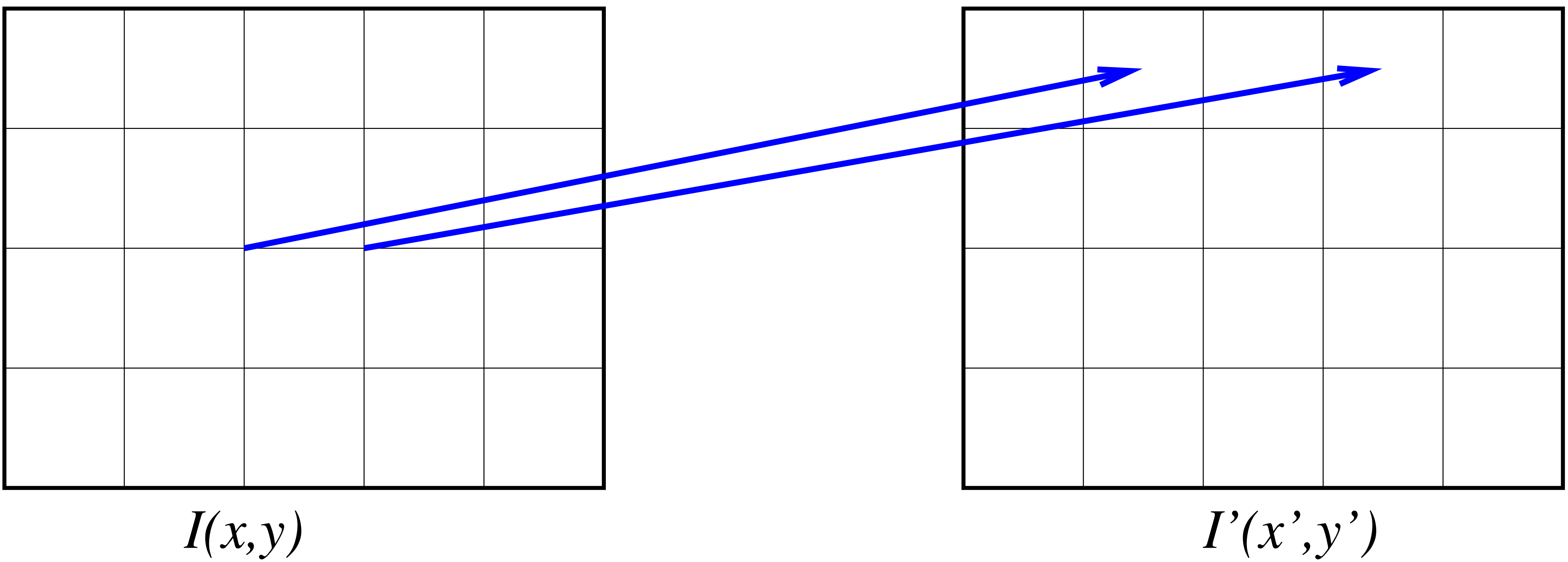


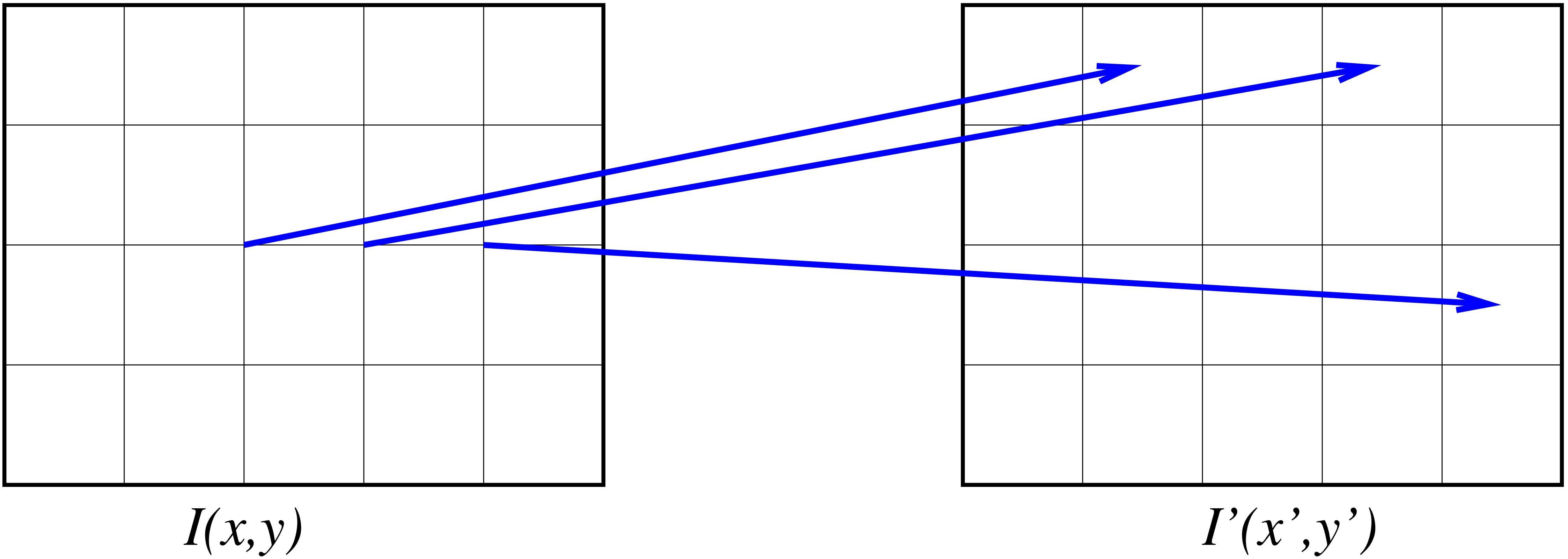
I(x,y)
I'(x',y')

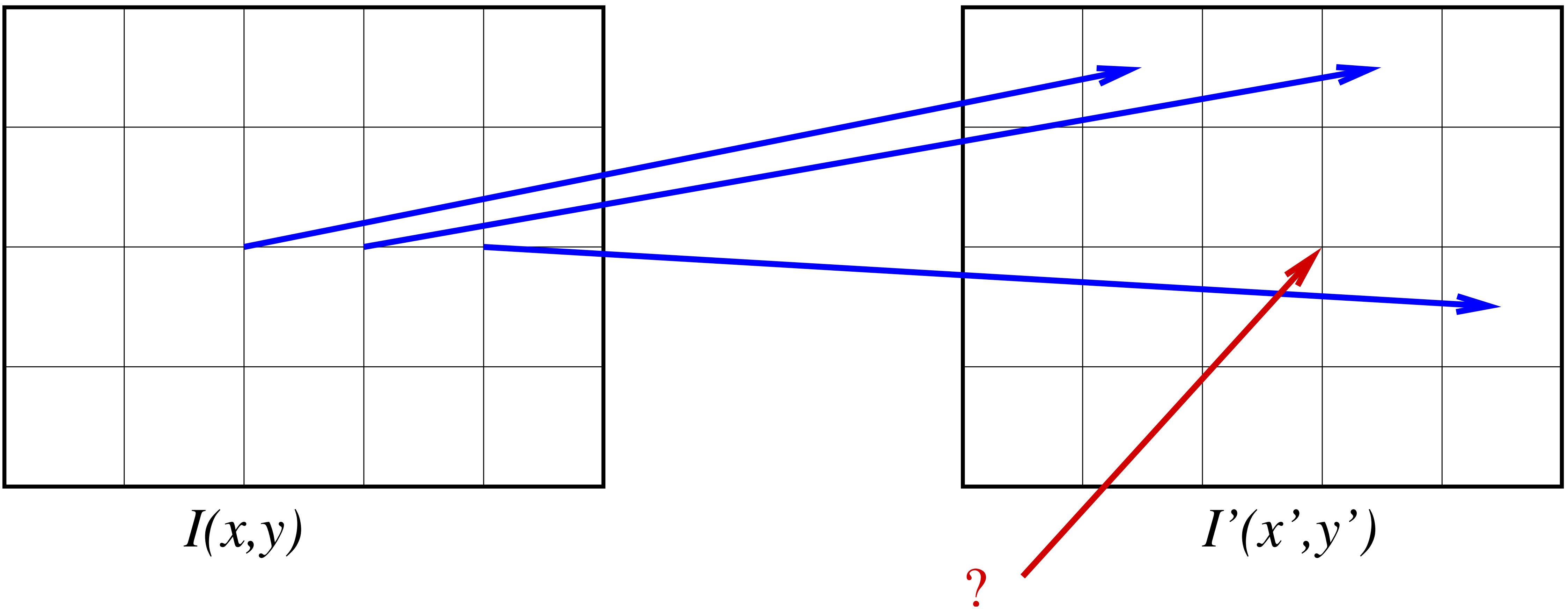

I(x,y)
I'(x',y')

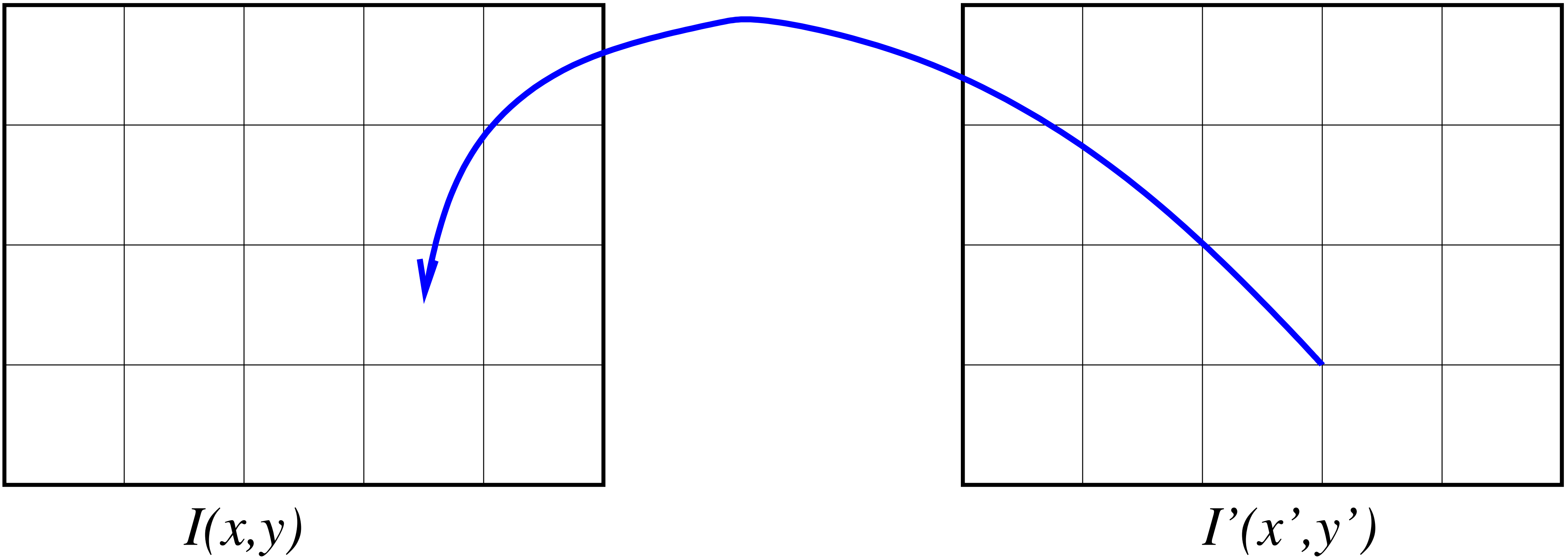

I(x,y)
I'(x',y')

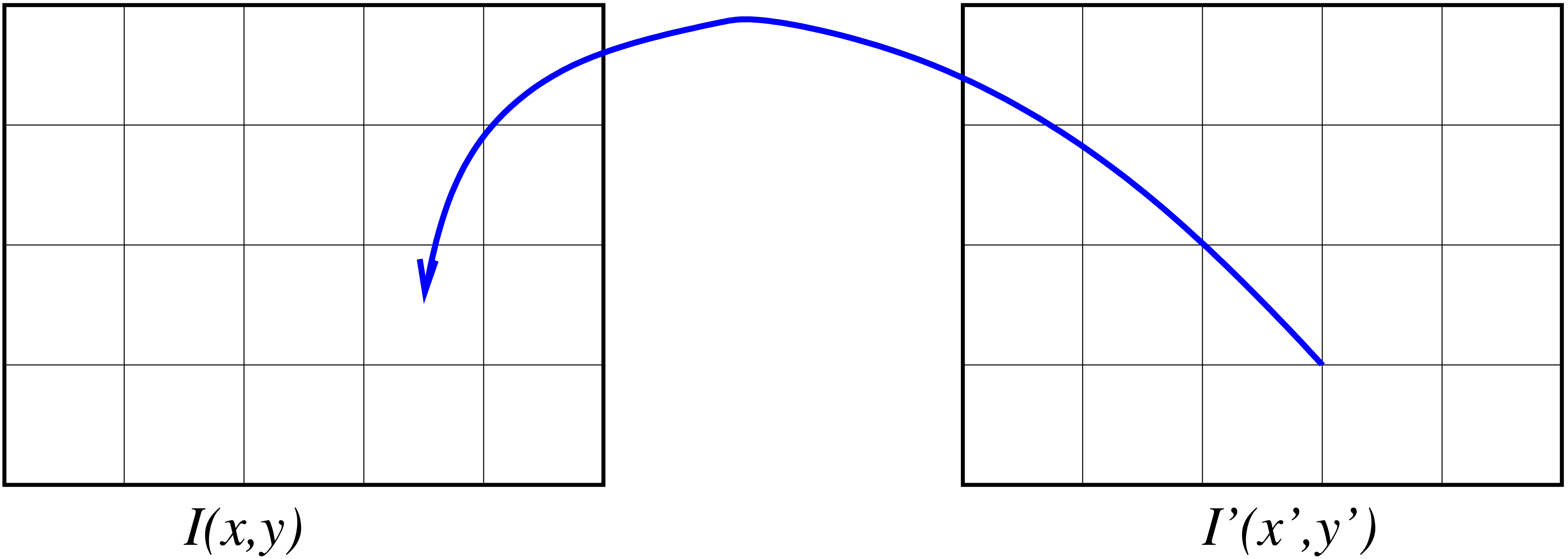

I(x,y)
I'(x',y')


I(x,y)
I'(x',y')

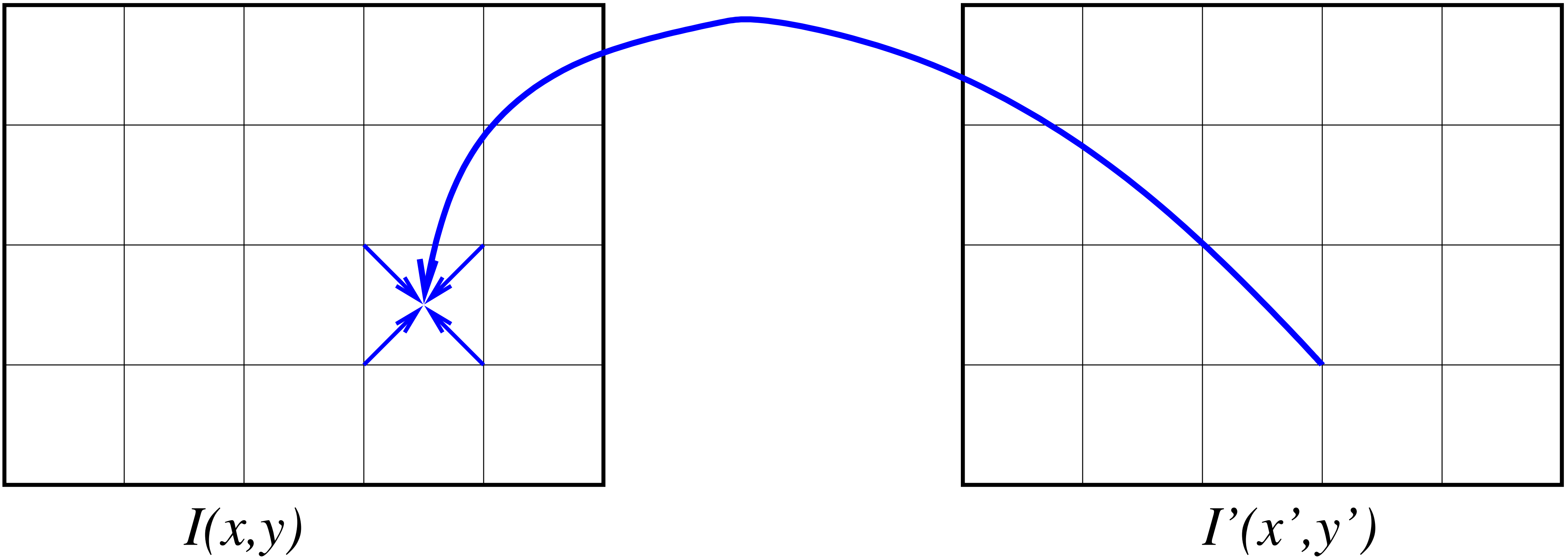

I(x,y)
I'(x',y')

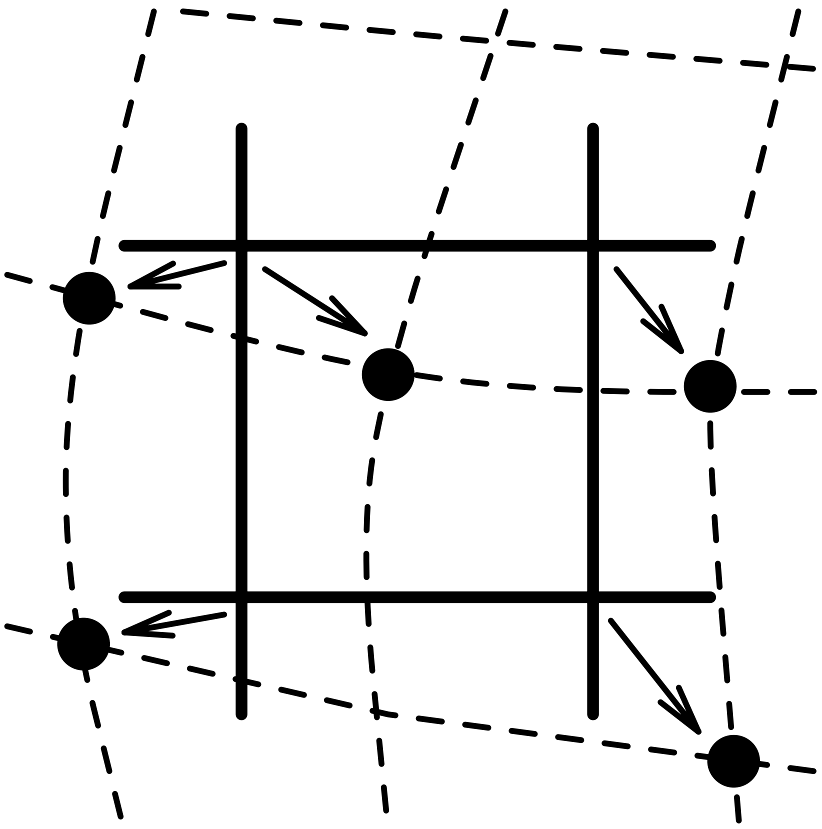

I(x,y)
I'(x',y')
?

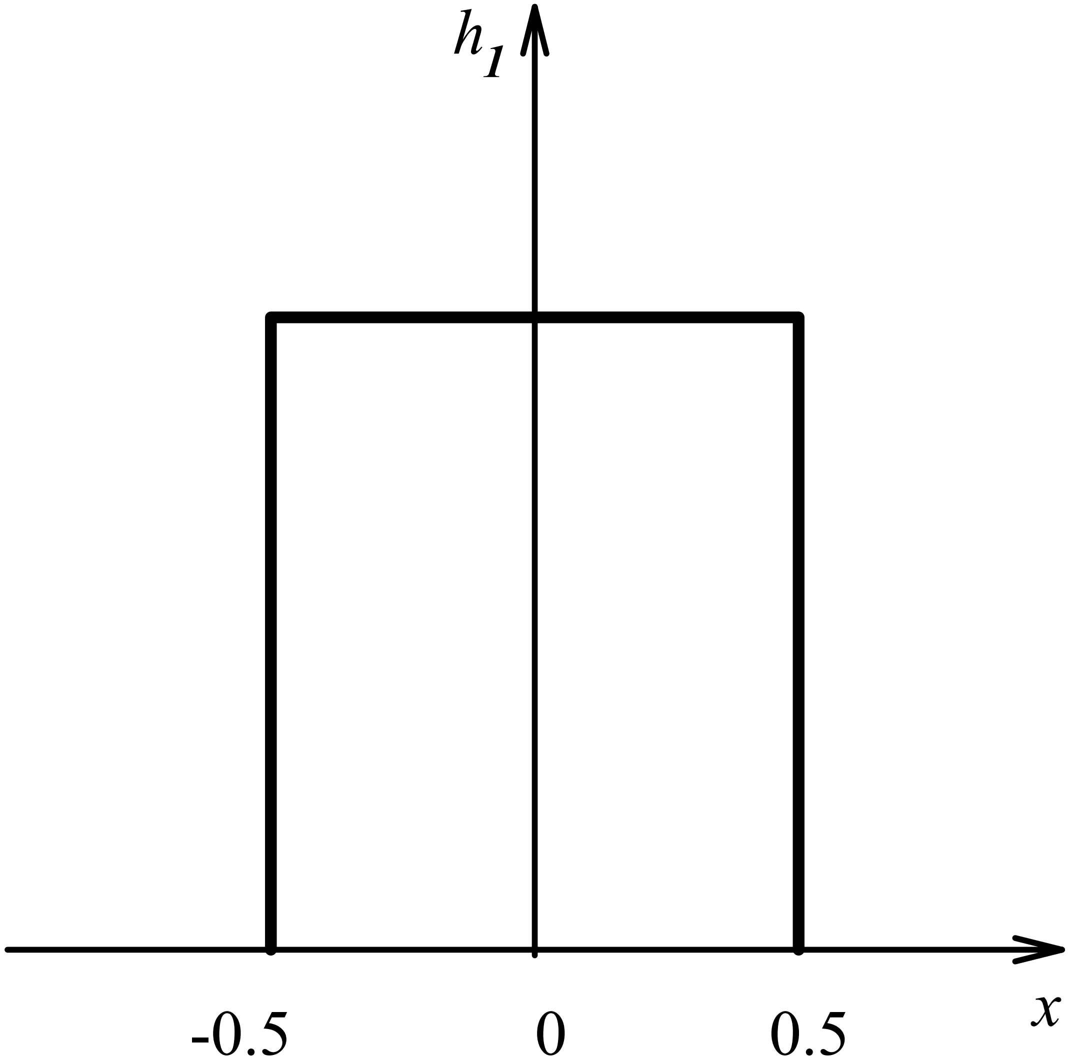

I(x,y)
I'(x',y')



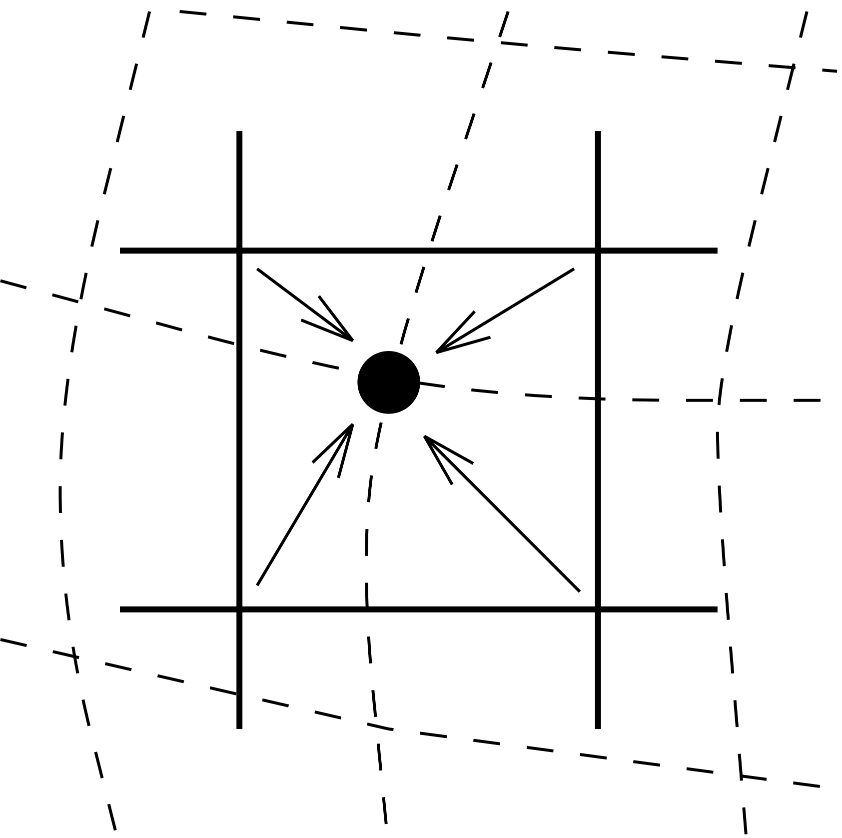
I(x,y)
I'(x',y')

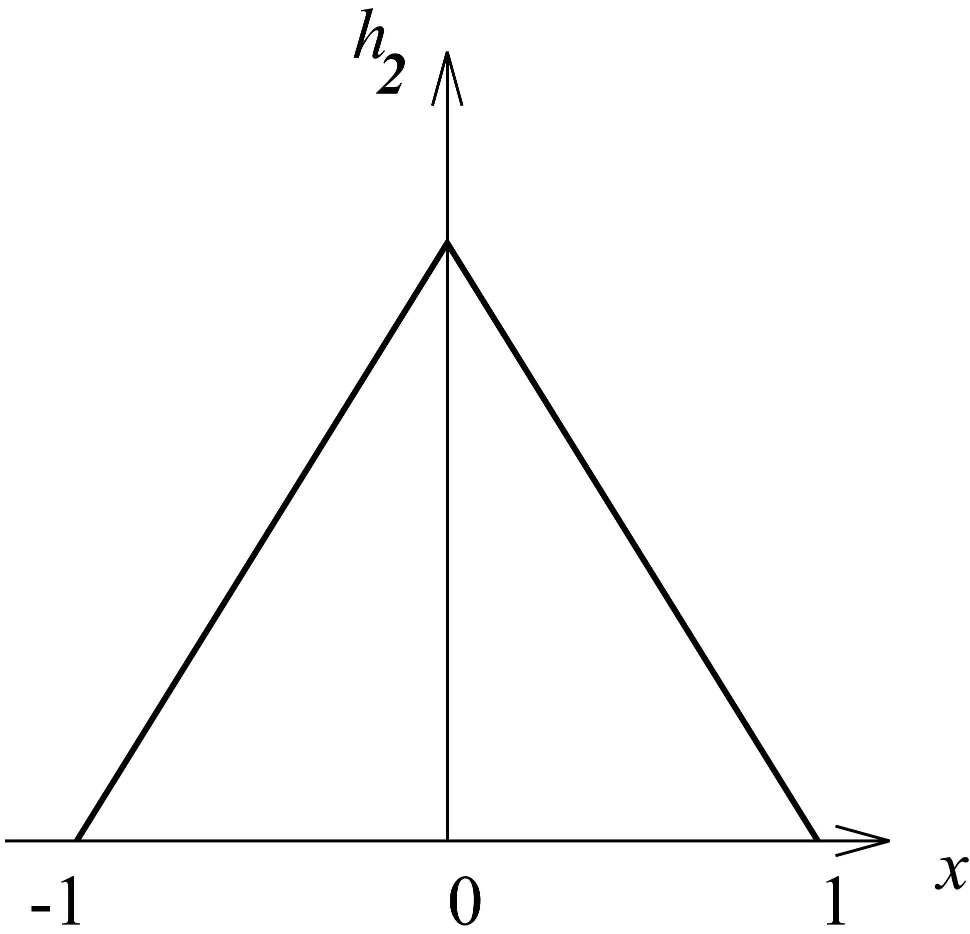

I(x,y)
I'(x',y')



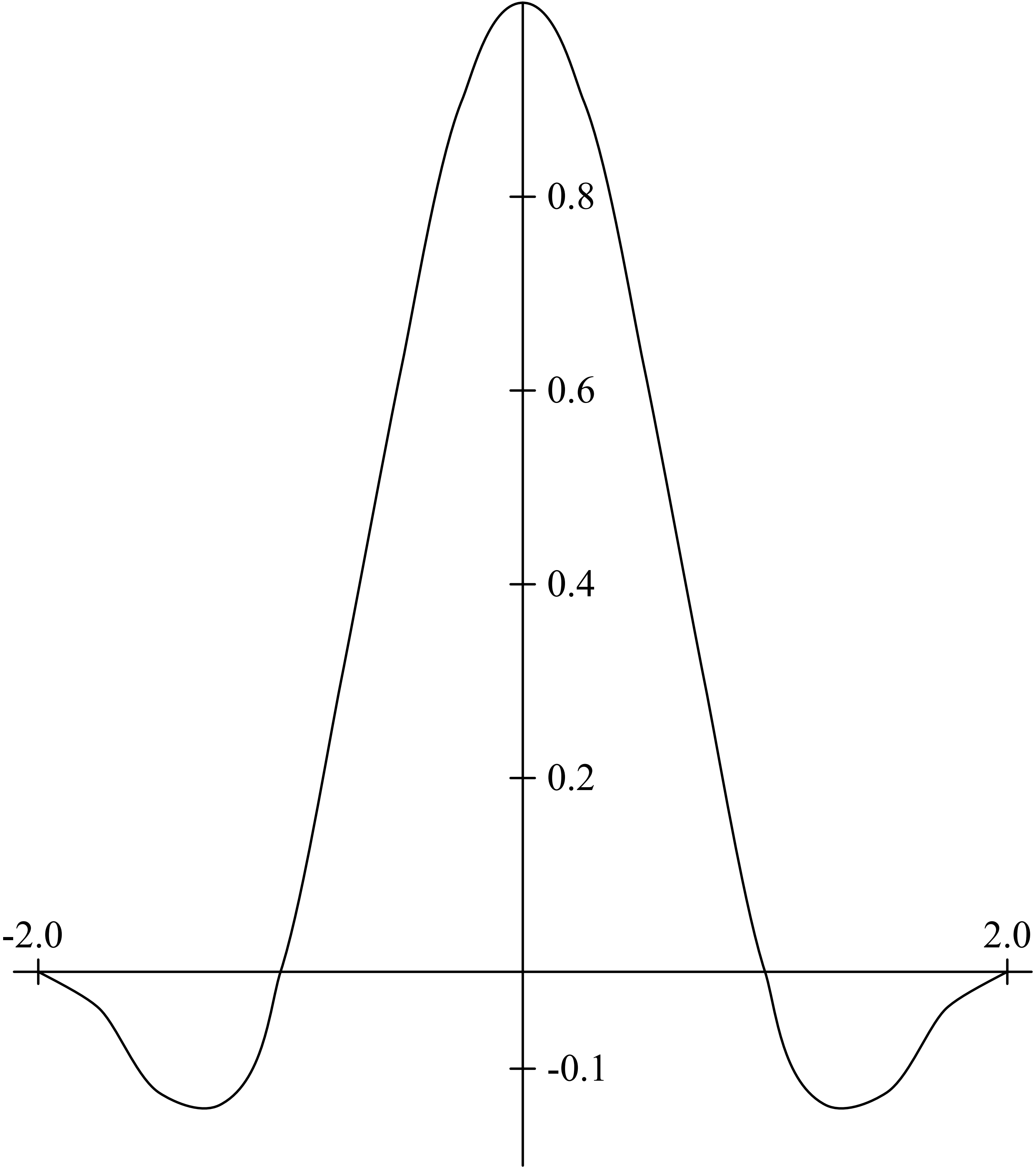


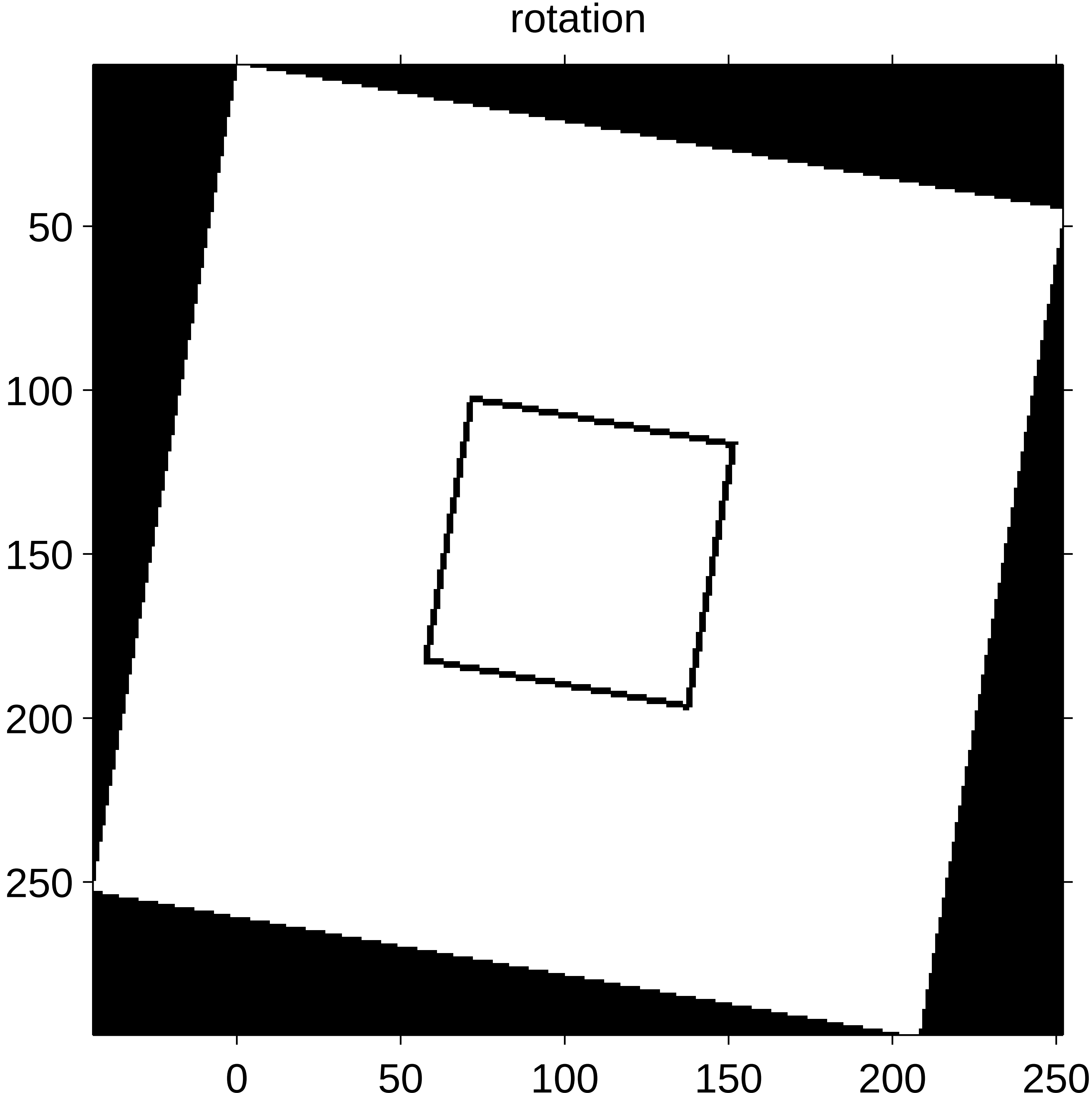
h_1
x
-0.5
0
0.5

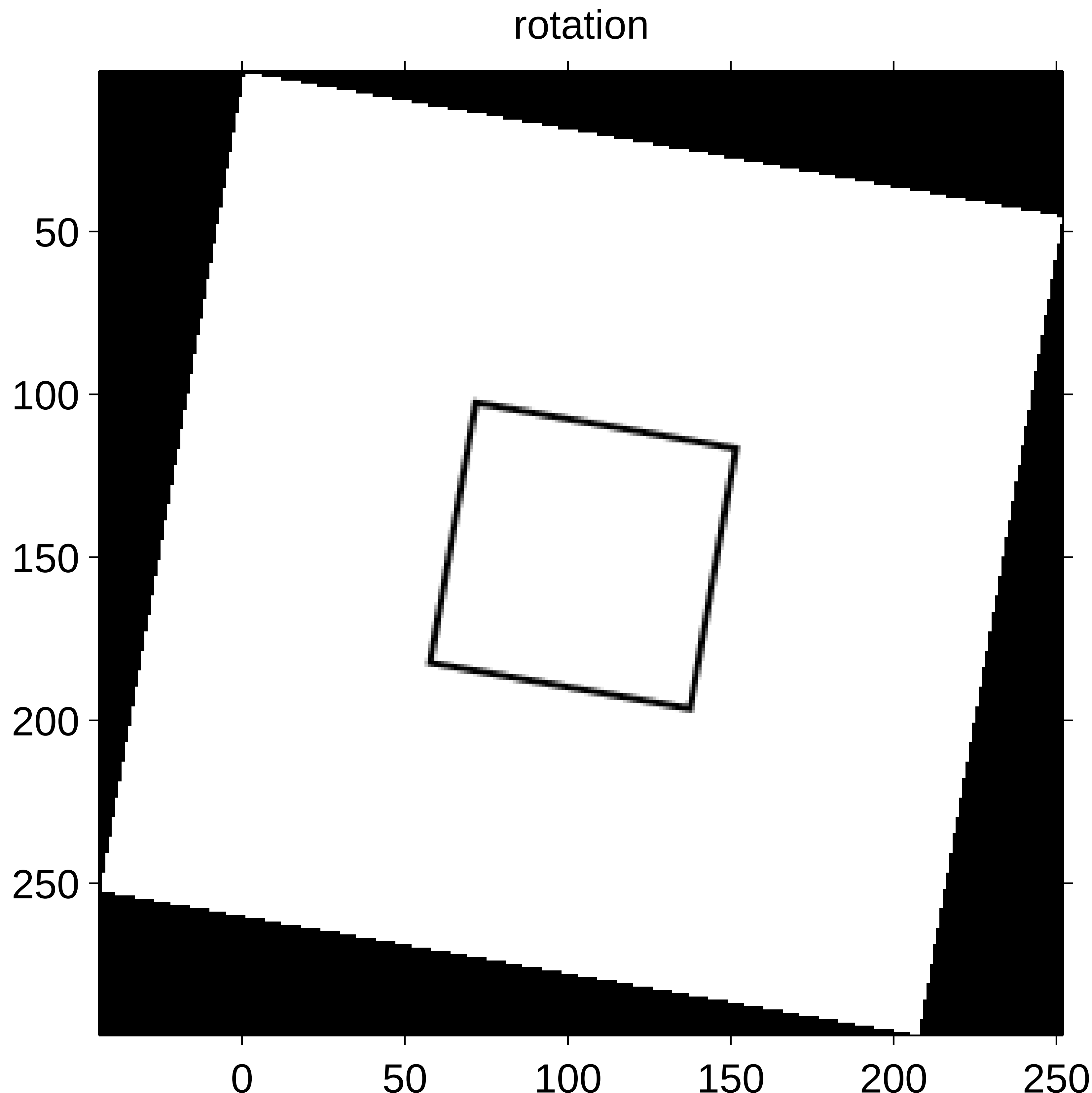




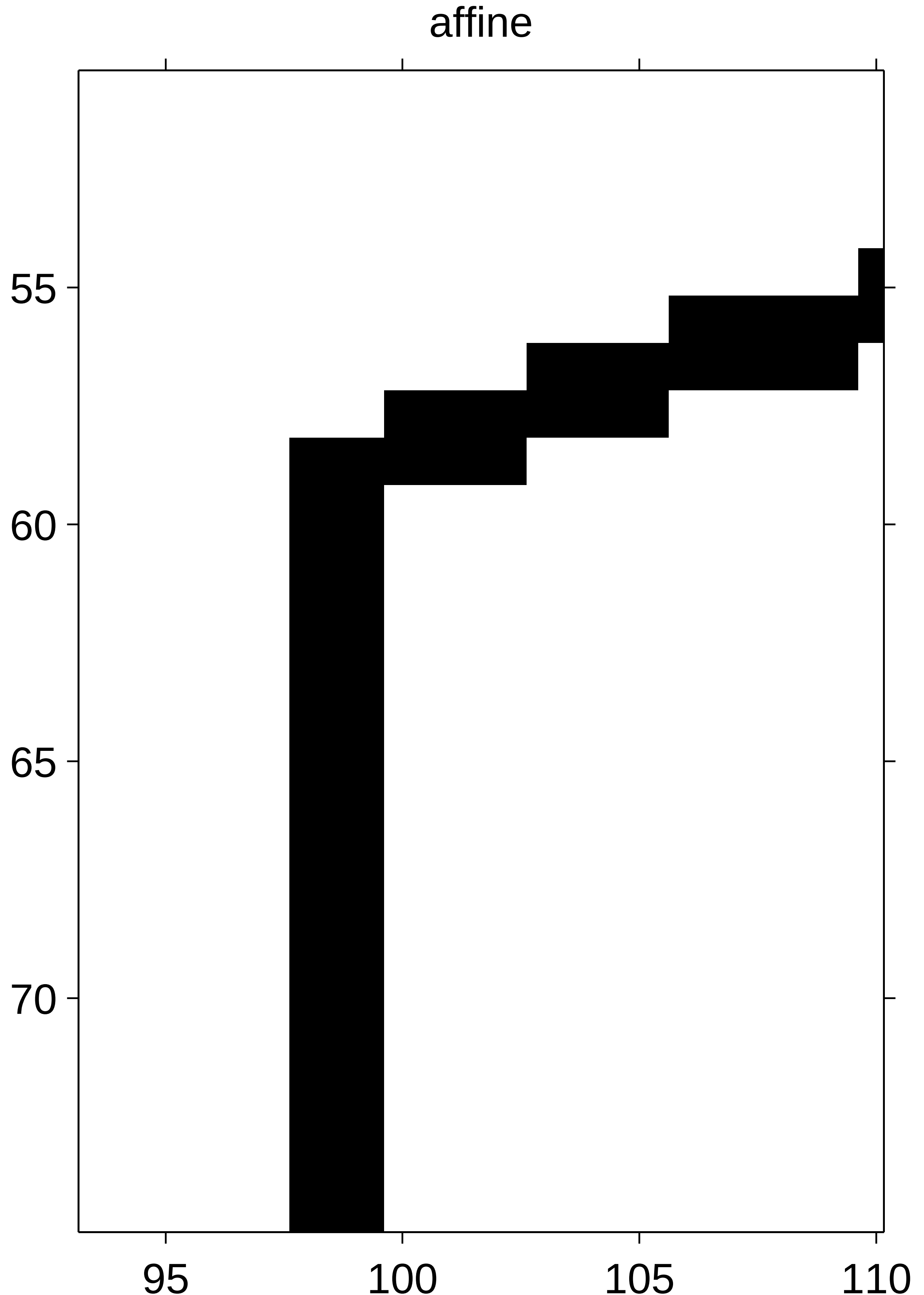
h_2
x
-1
0
1

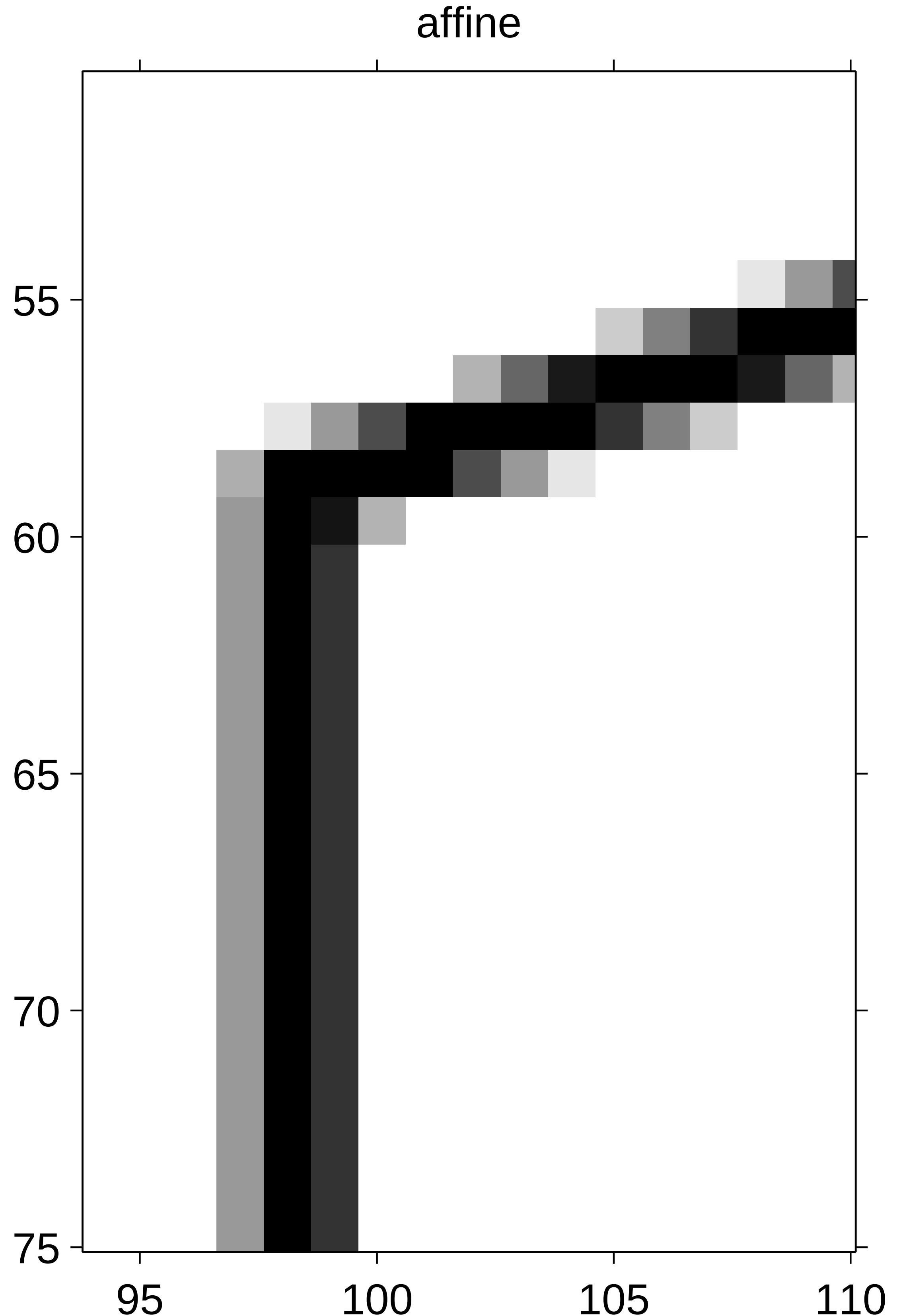

0.8
0.6
0.4
0.2
-2.0
2.0
-0.1


rotation
50
100
150
200
250
0
50
100
150
200
250



rotation
50
100
150
200
250
0
50
100
150
200
250


affine
55
60
65
70
95
100
105
110


affine
55
60
65
70
75
95
100
105
110