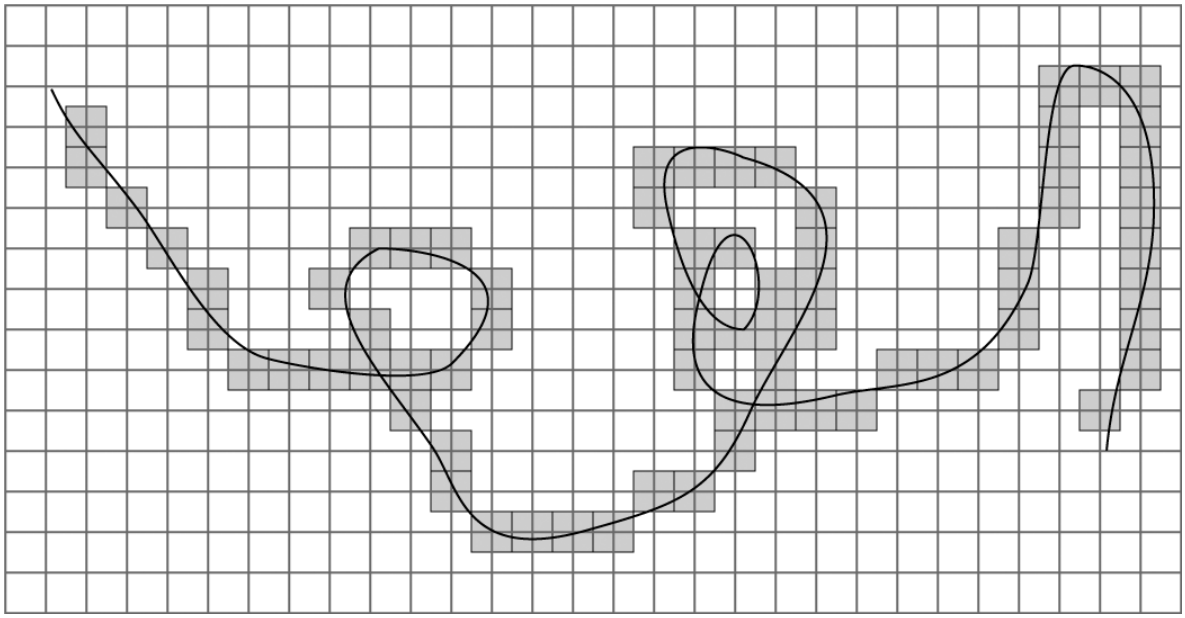
Grid-Intersection Digitization

The *grid-intersection digitization* $R(\gamma)$ of a planar curve or arc γ is the set of all grid points (i, j) that are closest (in Euclidean distance) to the intersection points of γ with the grid lines.



An intersection point may have the same minimum distance to two different grid points; such an intersection point contributes two grid points to $R(\gamma)$.

The figure above shows all the grid cells (2-cells) around all grid points in $R(\gamma)$.

Digitization of Straight Segments

Digitization of *straight segments* (i.e., segments of finite length of straight lines) is a standard routine in computer graphics.

We digitize a segment of a line $y = ax + b$ in the first octant (i.e., with slope $a \in [0, 1]$).

(By interchanging the startpoints and endpoints of the segment, we can handle octants "to the left of the y-axis". In the eighth octant, we use a y-increment of -1, and in the second and seventh octants, we interchange the roles of the x- and y-coordinates.)

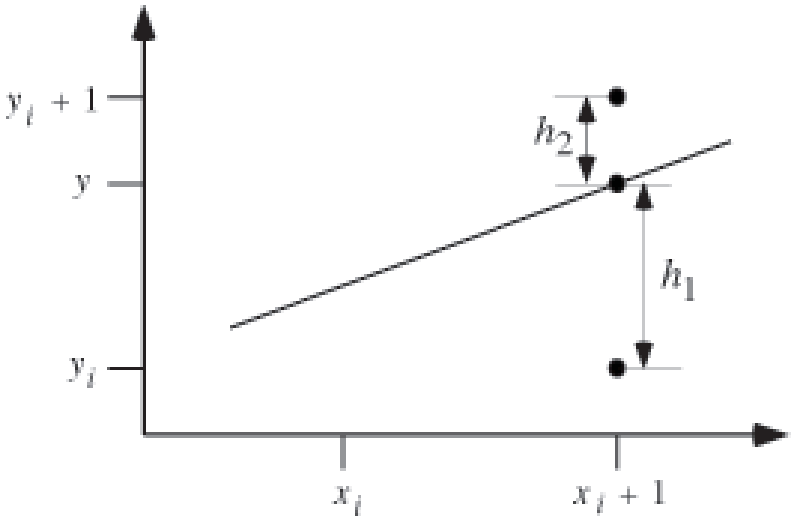
To draw the resulting *digital straight segment* (DSS for short), we increase the x-coordinate stepwise by $+1$; the y-coordinate is "occasionally" increased by $+1$ and remains constant otherwise.

The DSS is a sequence of grid points $(x_i, y_i), i \geq 1$. The point (x_1, y_1) is the grid point closest to the endpoint of the real segment. If we already have point (x_i, y_i), the next point has x-coordinate x_{i+1}, and, for its y-coordinate, we must decide between y_i and $y_i + 1$.

Bresenham's DSS algorithm

(published in 1965)

$$y = a(x_i + 1) + b$$
$$h_1 = y - y_i = a(x_i + 1) + b - y_i$$
$$h_2 = (y_i + 1) - y = y_i + 1 - a(x_i + 1) - b$$



To decide between y_i and $y_i + 1$, we use the following difference:

$$h_1 - h_2 = 2a(x_i + 1) - 2y_i + 2b - 1$$

We choose $(x_i + 1, y_i)$ if $h_1 < h_2$ and $(x_i + 1, y_i + 1)$ otherwise.

For reasons of efficiency (minimization of operations, and integer arithmetic only), we do not use $h_1 - h_2$ for this decision.

Rather, let $p = (x_1, y_1)$ and $q = (x_q, y_q)$ be the grid points closest to the endpoints of the segment, and let $dx = x_q - x_1$ and $dy = y_q - y_1$.

Let $e_i = dx \cdot (h_1 - h_2)$; thus $e_i = 2(dy \cdot x_i - dx \cdot y_i) + b'$, where $b' = 2dy + 2dx \cdot b - dx$ is independent of i.

Thus e_i can be updated iteratively for successive decisions at $x_i + 1$ and $x_i + 2$:

$$e_i = 2dy \cdot x_i - 2dx \cdot y_i + b'$$
$$e_{i+1} = 2dy \cdot x_{i+1} - 2dx \cdot y_{i+1} + b'$$

Thus

$$e_{i+1} - e_i = 2dy(x_{i+1} - x_i) - 2dx(y_{i+1} - y_i) = 2dy - 2dx(y_{i+1} - y_i)$$

because $x_{i+1} = x_i + 1$; this is sufficient for deciding about the y-increment. Let $x_1 = x_p = 0$ and $y_1 = y_p = 0$ give the starting value:

$$e_1 = 2dy \cdot x_1 - 2dx \cdot y_1 + 2dy + dx(2b - 1) = 2dy - dx$$

Bresenham algorithm for the first octant:

1. Let $dx = x_q - x_p$, $dy = y_q - y_p$, $x = x_p$, $y = y_p$, $b_1 = 2 \cdot dy$, $error = b_1 - dx$, and $b_2 = error - dx$.
2. Repeat Steps 3 through 6 until $x > x_q$. Stop when $x > x_q$.
3. Change the value of (x, y) in P to the value of a line pixel.
4. Increment x by 1.
5. If $error < 0$, let $error = error + b_1$, or else increment y by 1 and let $error = error + b_2$.
6. Go to Step 2.

At Step 1, we have $error = e_1 = 2dy - dx$.

The values $b_1 = 2 \cdot dy$ and $b_2 = 2 \cdot dy - 2 \cdot dx$ are used to efficiently update the variable $error$.

The algorithm runs in $\mathcal{O}(x_q - x_p)$ time because, for each i, it involves only a constant number of operations: setting one pixel value, two simple logical tests, one addition, and one or two increments.