Bresenham's Line Algorithm

This algorithm is used for scan converting a line. It was developed by Bresenham. It is an efficient method because it involves only integer addition, subtractions, and multiplication operations. These operations can be performed very rapidly so lines can be generated quickly.

In this method, next pixel selected is that one who has the least distance from true line.

The method works as follows: Assume a pixel $P_1'(x_1', y_1')$,then select subsequent pixels, one pixel position at a time in the horizontal direction toward $P_2'(x_2', y_2')$.

Once a pixel in choose at any step

The next pixel is

1.

Either the one to its right (lower-bound for the line)

One top its right and up (upper-bound for the line)

2.

The line is best approximated by those pixels that fall the least distance from the path betweenP_1', P_2'.

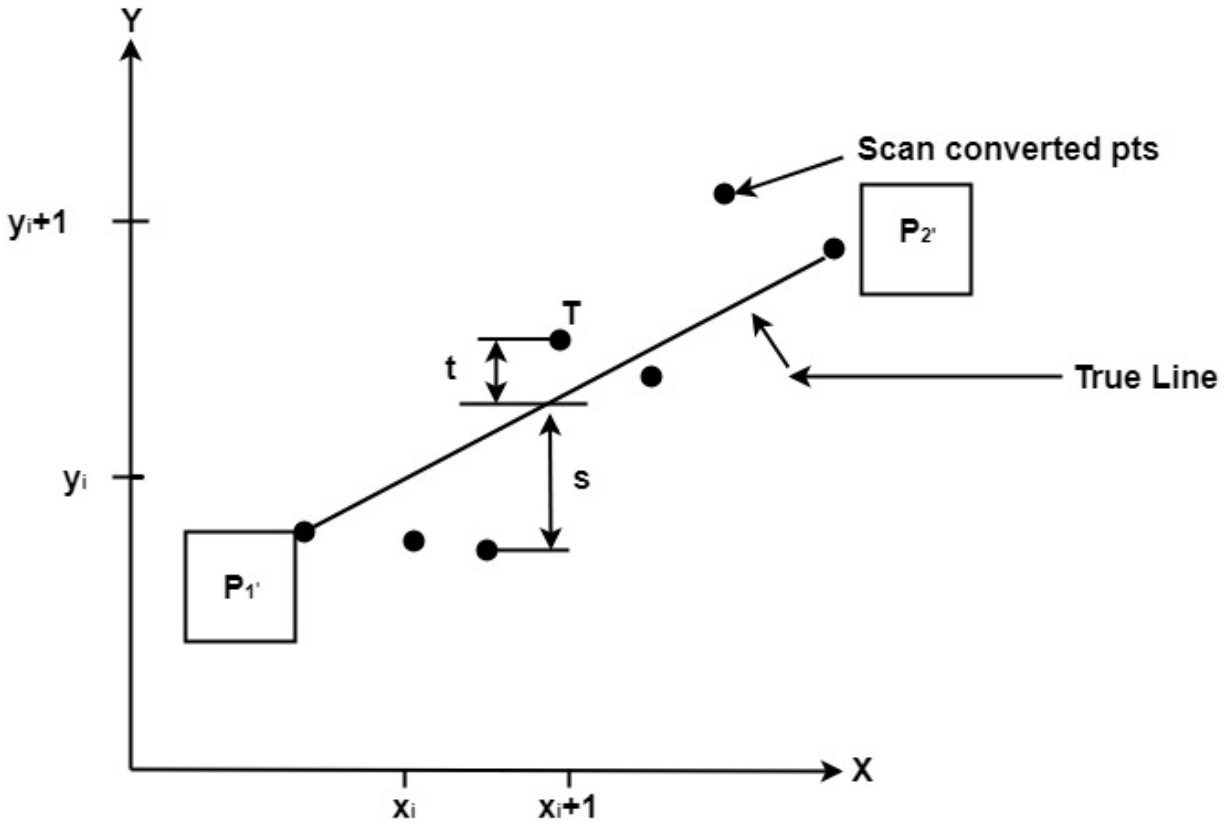

Fig: Scan Converting a line.

To chooses the next one between the bottom pixel S and top pixel T.
If S is chosen We have $x_{i+1}=x_i+1$ and $y_{i+1}=y_i$
If T is chosen We have $x_{i+1}=x_i+1$ and $y_{i+1}=y_i+1$
The actual y coordinates of the line at $x = x_{i+1}$is

$y=mx_{i+1}+b$

$$y = m(x_i + 1) + b$$

The distance from S to the actual line in y direction

$s = y - y_i$

The distance from T to the actual line in y direction $t = (y_i+1)-y$
Now consider the difference between these 2 distance values

$s - t$

When $(s-t) < 0 \Longrightarrow s < t$

The closest pixel is S
When $(s-t) \geq 0 \Longrightarrow s < t$
The closest pixel is T

This difference is

$s-t = (y-y_i)-[(y_i+1)-y]$

$= 2y - 2y_i - 1$

$$s - t = 2m(x_i + 1) + 2b - 2y_i - 1$$ [Putting the value of (1)]

Substituting m by $\Delta Y/\Delta x$ and introducing decision variable

$d_i=\Delta x\ (s-t)$
$d_i=\Delta x\ (2\ (x_i+1)+2b-2y_i-1)$
$=2\Delta xy_i-2\Delta y-1\Delta x.2b-2y_i\Delta x-\Delta x$
$d_i=2\Delta y.x_i-2\Delta x.y_i+c$
Where $c= 2\Delta y+\Delta x\ (2b-1)$

We can write the decision variable d_{i+1} for the next slip on

$d_{i+1}=2\Delta y.x_{i+1}-2\Delta x.y_{i+1}+c$

$d_{i+1}-d_i=2\Delta y.(x_{i+1}-x_i)- 2\Delta x(y_{i+1}-y_i)$

Since $x_{(i+1)}=x_i+1$,

we have $d_{i+1}+d_i=2\triangle y.(x_i+1-x_i)- 2\triangle x(y_{i+1}-y_i)$

Special Cases

If chosen pixel is at the top pixel T (i.e., $d_i \geq 0) \Longrightarrow y_{i+1}=y_i+1$

$d_{i+1}=d_i+2\triangle y-2\triangle x$

If chosen pixel is at the bottom pixel T (i.e., $d_i<0) \Longrightarrow y_{i+1}=y_i$

$d_{i+1}=d_i+2\triangle y$

Finally, we calculate d_1

$d_1=\triangle x[2m(x_1+1)+2b-2y_1-1]$

$d_1=\triangle x[2(mx_1+b-y_1)+2m-1]$

Since $mx_1+b-y_i=0$ and $m = Y/\Delta x$,

we have $d_1=2\triangle y-\triangle x$.

Advantage:

1. It involves only integer arithmetic, so it is simple.
2. It avoids the generation of duplicate points.