M30242 Graphics and Computer Vision

Lecture 5 Corner and Line Detection

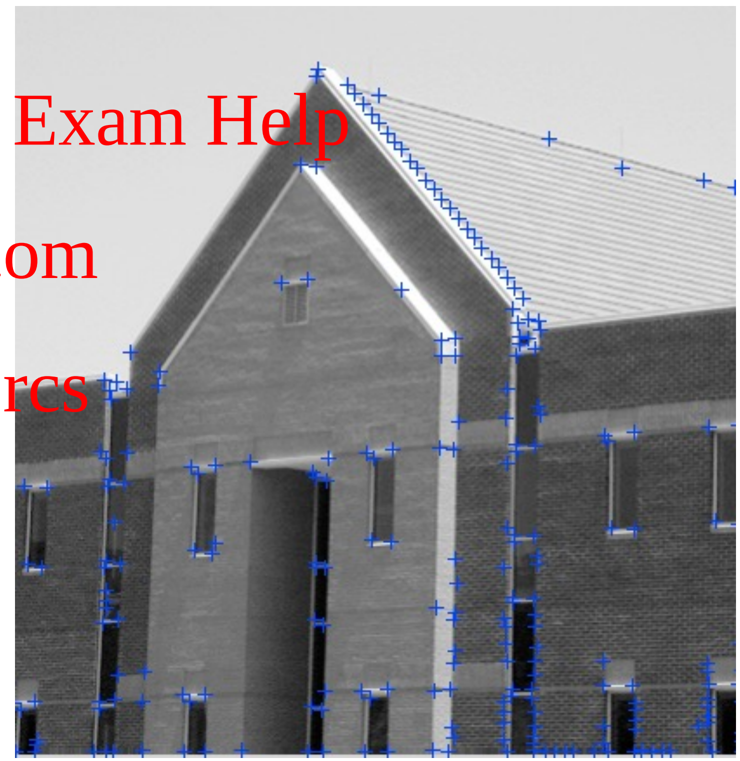
Overview

- Corner detection and Harris corner detector

- Hough transform - an introduction

Corner Detection

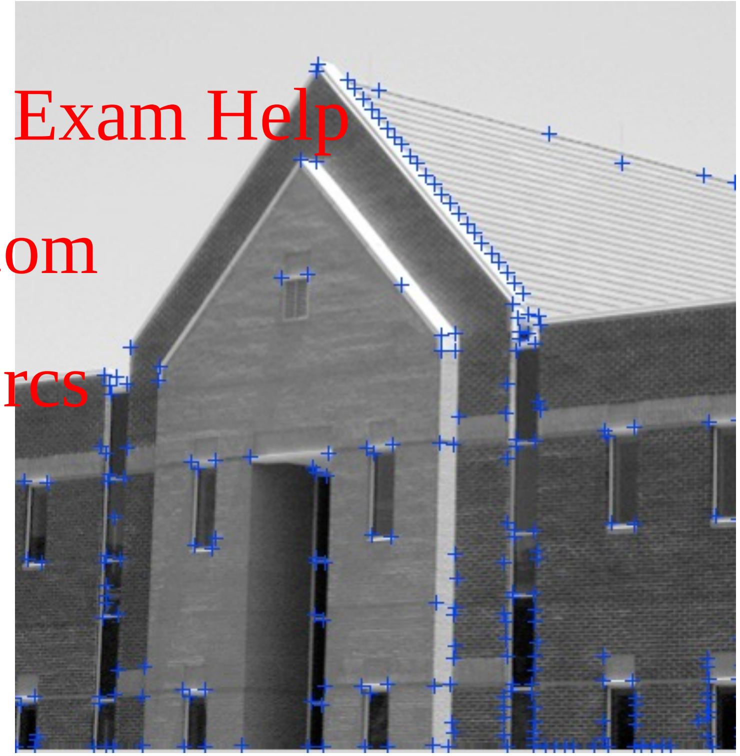
- We are often interested in detecting point features (corners) in an image.
  - Corners define lines
  - Complex shapes can be represented by the coordinates of the corners



- What characteristics do corners have?

Characteristics of Corners

- Corners are usually defined as regions in images where there are significant intensity gradients *in two or more directions*.



- This observation leads to the development of some corner detection methods.

Naïve Method

- The observation might suggest a simple approach for corner detection:
  - Applying edge detectors (e.g., Sobel or Prewitt) to detect edges in x- and y-direction at each pixel (i.e., computing f_x and f_y),
  - Applying a threshold T to f_x and f_y,
  - If both f_x and f_y at a pixel are higher than T, then it is a possible corner.
- But such a naïve method does not work very well.

Corner Detection - Hessian Matrix

- Instead of thresholding f_x and f_y, of single pixels, a sophisticated approach computes the *Hessian matrix* within a neighbourhood of a pixel:

  - Assume a pixel p and a neighborhood Q of p (e.g., 3x3, 5x5 etc)
  - The Hessian matrix, C, is defined as:

$$C = \begin{bmatrix} \sum_Q f_x^2 & \sum_Q f_x f_y \\ \sum_Q f_x f_y & \sum_Q f_y^2 \end{bmatrix} = \sum_Q \begin{bmatrix} f_x \\ f_y \end{bmatrix} \begin{bmatrix} f_x & f_y \end{bmatrix}$$

where Σf^2_x is the summation of the squares of the partial derivatives/gradients in x-direction over all the pixels within Q. The meaning for Σf^2_y is similar.

Cont'd

- In this approach, detecting the presence of corners is to compute the eigenvalues and eigenvectors of the Hessian matrix at a pixel:
  - *the eigenvectors encode the gradient directions* (i.e., the normal vector to the edge)
  - *the eigenvalues λ encode edge strength*
- As you can see, the approach is based more on mathematics than on intuition (Unfortunately, this is true for many many CV techniques).
- Some linear algebra are involved here.

Eigenvectors and Eigenvalues

- Roughly, given a (square) matrix A, find a vector $\mathbf{x}$ and a matrix λ so that the following holds:

$$A\mathbf{x} = \lambda\mathbf{x}$$

where A is a $\boldsymbol{n}$-by-$\boldsymbol{n}$ matrix and X is a n-component vector

$$\mathbf{X} = \begin{bmatrix} x_1 & x_1 & \cdots & x_n \end{bmatrix}^T$$

- And λ is a $\boldsymbol{n}$-by-$\boldsymbol{n}$ matrix but only the diagonal element are non-zero

$$\lambda = \begin{bmatrix} \lambda_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \lambda_n \end{bmatrix}$$

- The above is a set of linear equations (in matrix form).
- The solution to this set of linear equations, i.e., $\mathbf{x}$ and λ, are called the eigenvectors and eigenvalues of matrix A.

An Example

Suppose $A = \begin{bmatrix} 2 & 2 & 3 \\ 9 & 7 & 1 \\ -3 & 1 & 6 \end{bmatrix}$ then $A\mathbf{x} = \lambda\mathbf{x}$ becomes

$$\begin{bmatrix} 2 & 2 & 3 \\ 9 & 7 & 1 \\ -3 & 1 & 6 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

For the set of linear equations, solving for the values for λ (eigenvalues) and vector $\mathbf{X}$ (eigenvectors) is called the eigenvalue problem.

In Matlab

Create matrix A

>> a=[2 2 3; 9 7 1; -3 1 6]

Solve the eigenvalue problem:

>> [v,lamda]=eig(a)

This gives 3 eigenvalues

λ_1=9.5754, λ_2=3.8522, λ_3=1.5724

and 3 corresponding eigenvectors:

$$v_1 = \begin{bmatrix} 0.4201 \\ -0.7816 \\ 0.4612 \end{bmatrix} \qquad v_2 = \begin{bmatrix} 0.1951 \\ -0.7562 \\ 0.6246 \end{bmatrix} \qquad v_3 = \begin{bmatrix} 0.2706 \\ 0.9618 \\ 0.0420 \end{bmatrix}$$

Cont'd

- You can check the validity of the eigenvalues and eigenvectors by substituting them back into the equations:

$$\begin{bmatrix} 2 & 2 & 3 \\ 9 & 7 & 1 \\ -3 & 1 & 6 \end{bmatrix} v_1 = \lambda_1 v_1$$

- That is

$$\begin{bmatrix} 2 & 2 & 3 \\ 9 & 7 & 1 \\ -3 & 1 & 6 \end{bmatrix} \begin{bmatrix} 0.4201 \\ -0.7816 \\ 0.4612 \end{bmatrix} \approx 9.5754 \begin{bmatrix} 0.4201 \\ -0.7816 \\ 0.4612 \end{bmatrix}$$

- The same is true for the other two pairs of eigenvalues and eigenvectors.

Eigenvalues of Hessian and Corners

- Hessian is a 2x2 matrix, and therefore has 2 eigenvalues and 2 eigenvectors.

- For a small region (the neighbourhood of a pixel) on an image:

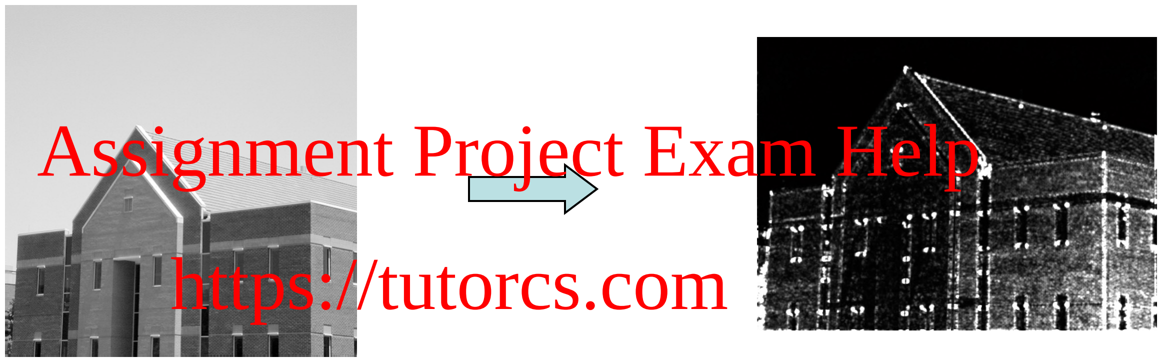
  - if the pixels are of constant intensity, *both eigenvalues will be very small*. – **No edge;**

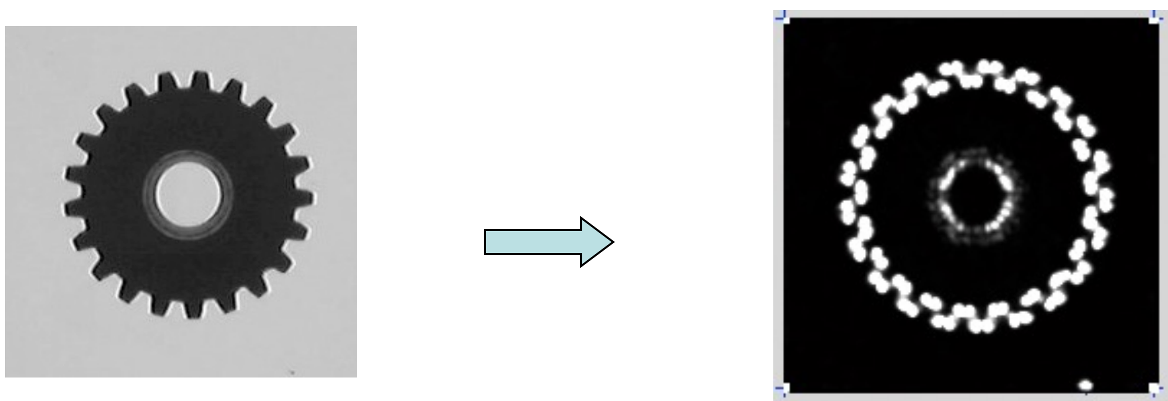
  - if it contains **one edge**, there will be *one large and one small eigenvalues*. The eigenvector associated with the larger eigenvalue will be parallel to the image gradient or orthogonal to the edge.
  - if it contains a corner (**more than one edges**), there will be *two large eigenvalues.*

Harris Corner Detector

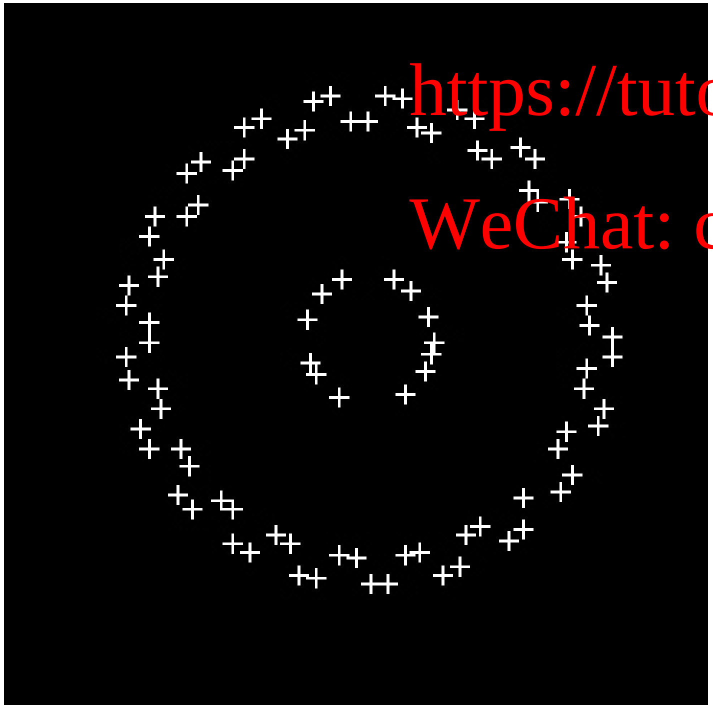
- The so-called Harris corner detector implements this idea, but there are two more factors to consider:
  - how big is an eigenvalue big? We need a threshold for λ_1, λ_2.
  - how should we choose the shape and size of the neighbourhood Q?
- Choosing the parameters:
  - A large neighborhoods leads to poor localization whereas a very small neighborhoods might not give good detection rates;
  - rectangular window is easy to use, but a circular window behaves isotropically.
- In real application, you may need to experiment with different parameters to get good results.

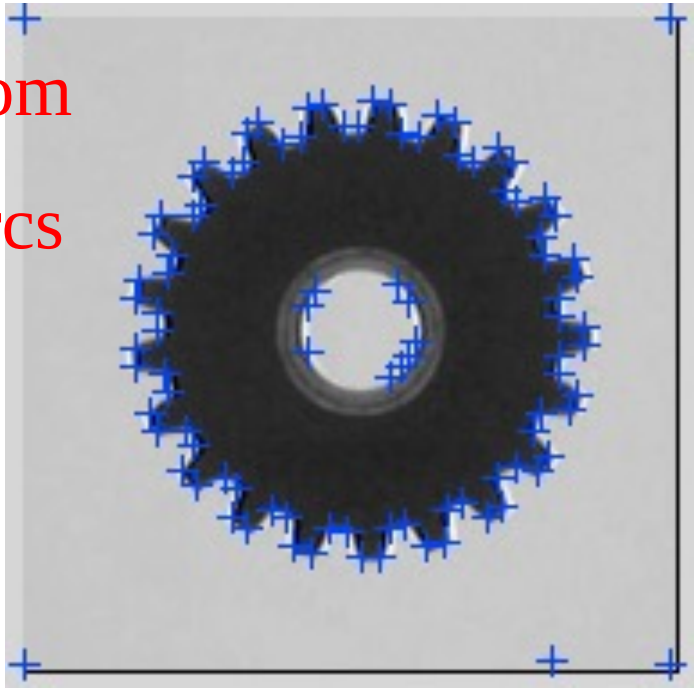
Examples





- Better result can be obtained by
  - first smoothing the ***gradient images*** of f_x, f_y (i.e., horizontal and vertical edges) with a Gaussian filter,
  - then constructing the Hessian matrix





Shape Detection and Hough Transform

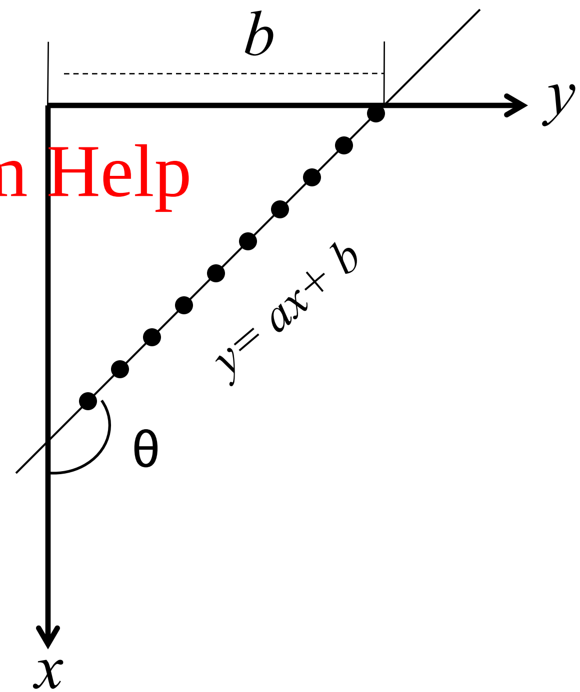
- Edge detection picks out the pixels of edges, but it does not say anything about the shapes of edges: is an edge a line or a circle?
- Detection of some common shapes, e.g., lines, circles, ellipses, etc., are very useful. They frequently appear in applications.
- Such shapes can be expressed by simple functions using a few parameters – we call them parametric curves.
- A very powerful method for detection of such shapes and their parameters (e.g., the end points of a line, the centre and radius of a circle, the centre and the long and short axes of an ellipse, etc) is Hough Transform (HT).

Line Detection By HT

- A line has the form

  $$y = ax + b$$

- This is called slope-intercept equation. It has two parameters: a and b
  - a is the slope ($a = \tan\theta$).
  - b is the intercept.
- The problem of line detection is this: given the evidence pixels of a line (the black dots in the figure), how we can find its slope ***a*** and intercept ***b***.

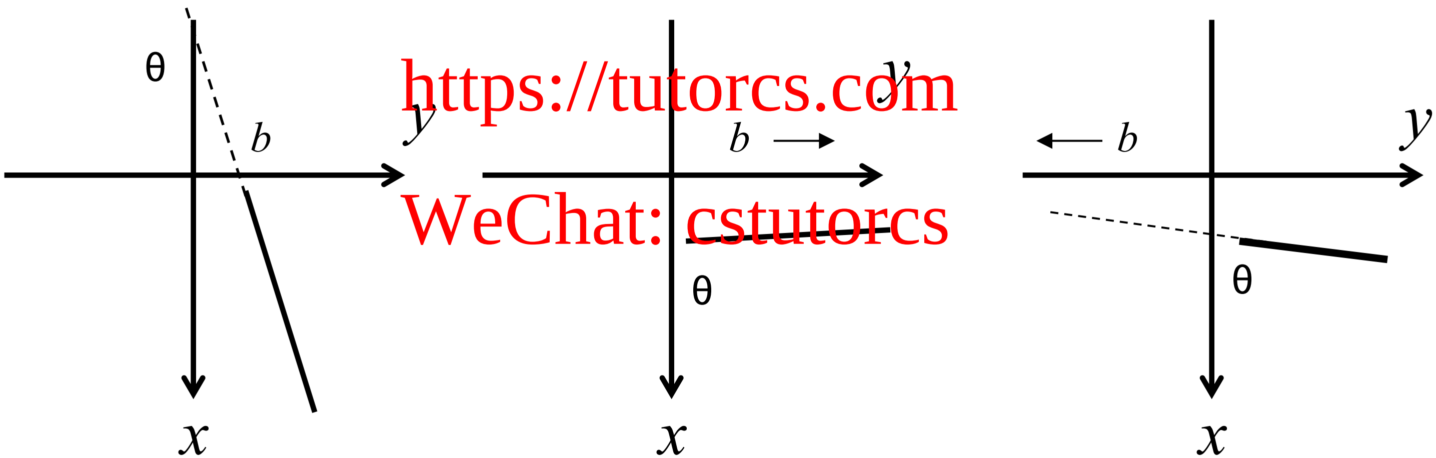


HT-Parameter Discretisation

- Hough transform has been named after its inventor Paul Hough in 1960s.
- Its principle is to decide the unique pair of a and b (representing the line) by voting using the evidence pixels (i.e., the pixels of the line).
- To do so, we divide values of parameters a and b into discrete values $a_1, a_2, \ldots, a_n$, and $b_1, b_2, \ldots, b_n$ and each pair of a_i - b_j represents a line.
- This process is called *parameter discretization/quantisation*.

Range of Parameter Values

- What are the range of the values of a and b?



$b \to 0$
$a = \tan\theta \to 0$

$b \to \infty$
$a = \tan\theta \to -\infty$

$b \to -\infty$
$a = \tan\theta \to \infty$

Discretisation of Parameters

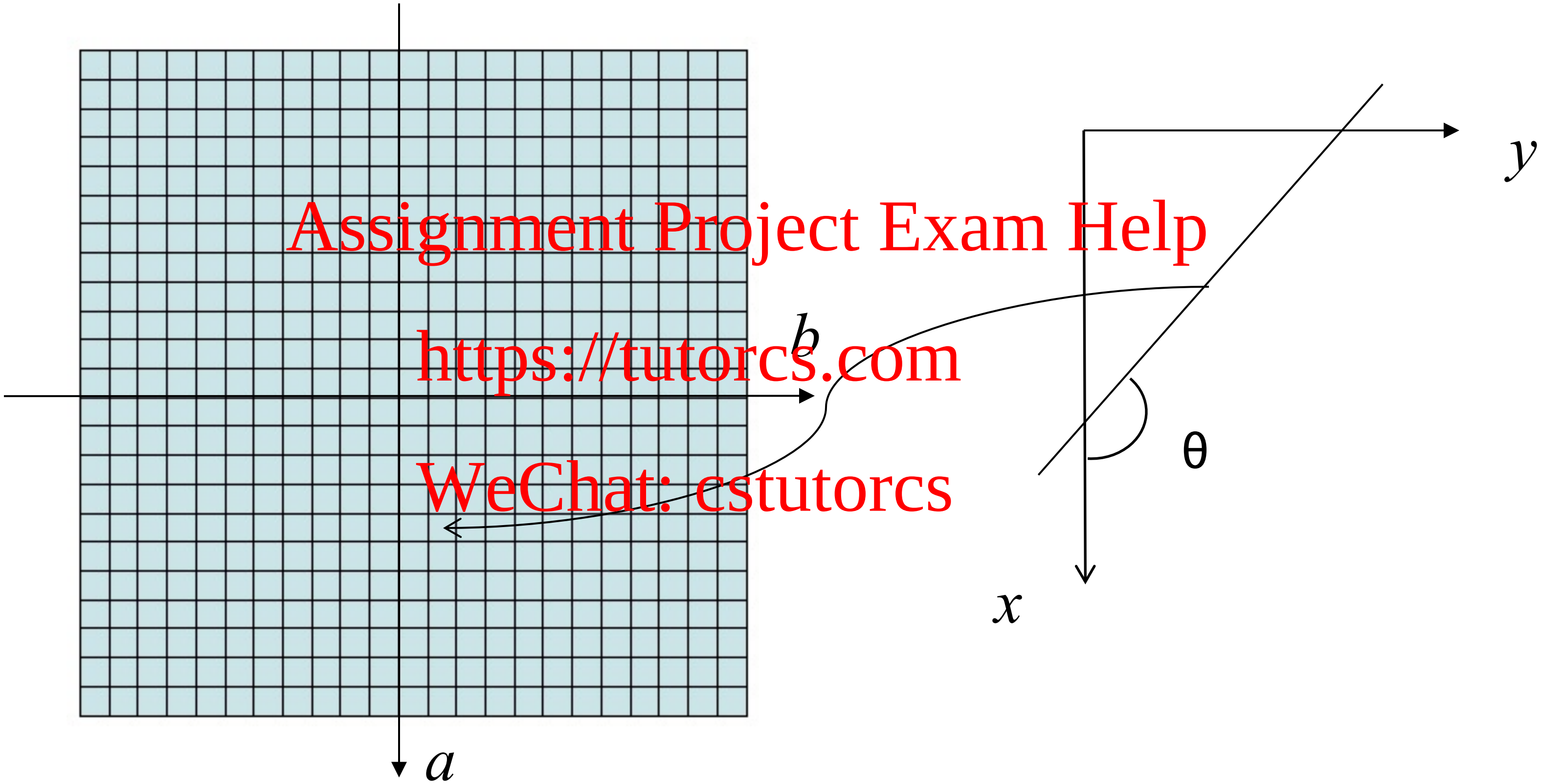
- The ranges for both a and b are (-∞, +∞), but we never work with infinity.
- We need to decide a workable range for a and b. E.g., we can limit the ranges to a finite size, e.g., from -200 to +200

  - Is this a good choice?

- Then we can divide the range into the discrete intervals (e.g., of size 5) for a and b:

  - a: -200, -195, -190,….+185, 190, 195, 200
  - b: -200, -195, -190,….+185, 190, 195, 200
- The effect of discretization is that each combination of a-b values defines a line in image space. Depending on the resolution of discretization, we can define a few thousands or even more lines.

A Graphical Representation of Parameter Space



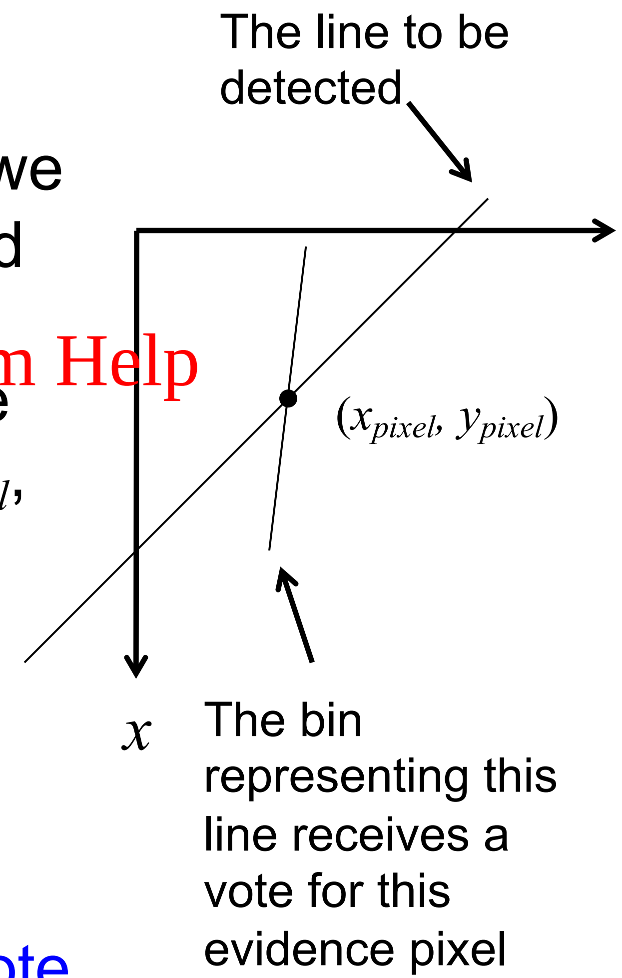
Here, each cell (is called a *bin*) represents a pair of a-b (a line) and these cells represent all the lines we will use for voting. This space is called the parameter space.

HT-Voting

- For each evidence pixel (of the line), we test which lines (of all the lines defined by the a-b pairs in parameter space) pass through the pixel by inserting the coordinates of the pixel, x_{pixel} and y_{pixel}, and an a-b pair into the line equation:

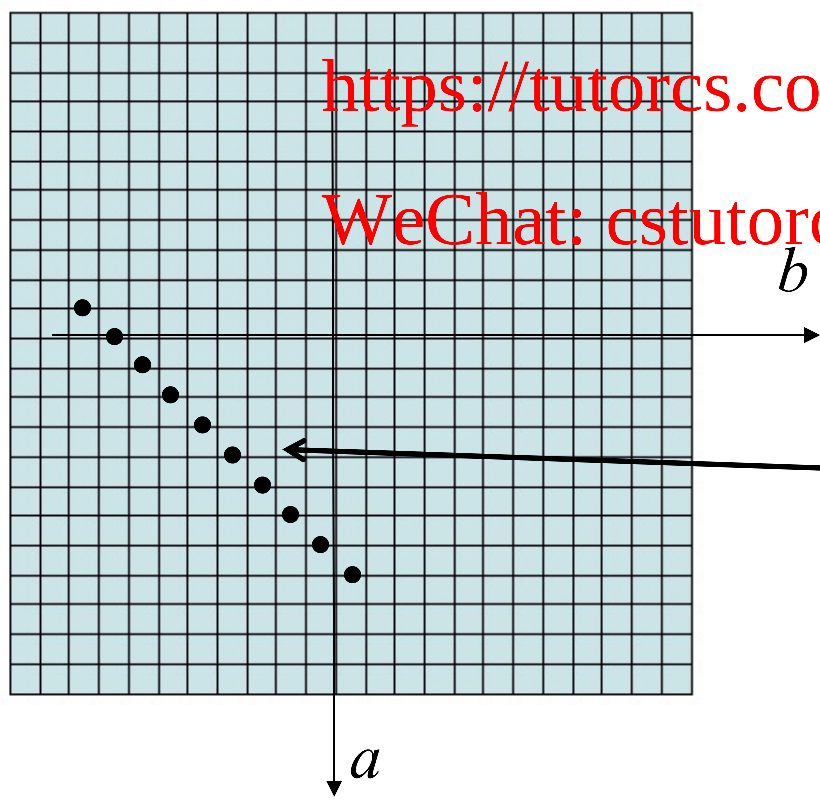
$$y_{pixel} = ax_{pixel} + b$$

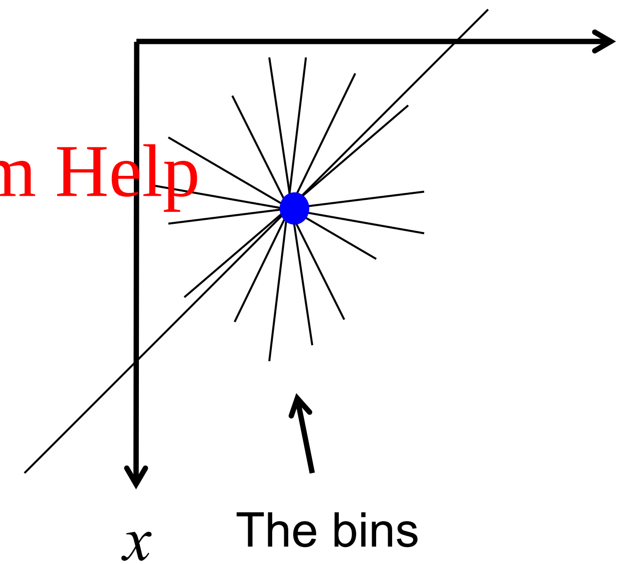
- If a line (a pair of a-b) passes through the evidence pixel, then the a-b pair satisfies the equation. We say the bin representing the a-b pair receives a vote from the evidence pixel.
- This is the process of *voting*.



Cont'd

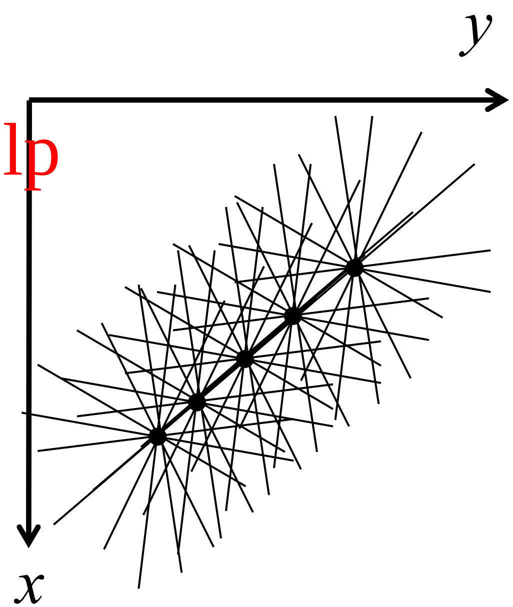
- The test will be done with every a-b pair(all cells in the parameter space!). Quite a few lines would be found to pass through the pixel, as shown in the figure.





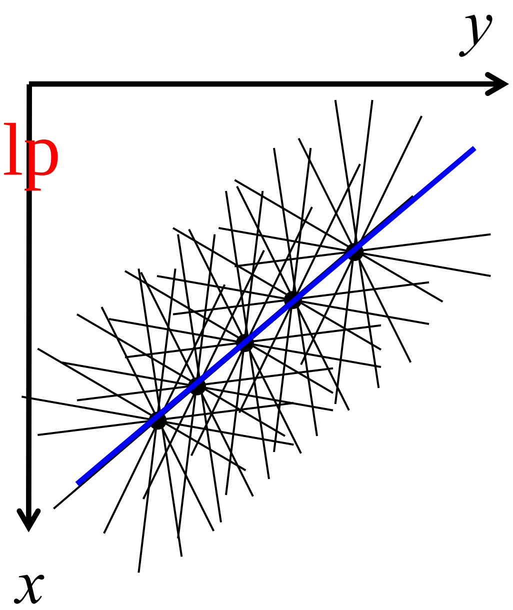
Cont’d

- Continue voting by using another evidence pixel on the line being detected.
- A new set of bins will receive a vote each. Among this new set of bins, there is one bin that has received a vote from the previous pixel already.
- If we do the same thing with all the evidence pixels and count the number of votes that each bin collects, which bin would have received the highest number of votes?



Cont’d

- It must be the bin that represents the blue line. In the figure, the blue line collects five votes and others get only one.
- By finding out the bin that collects the highest number of votes, we have known the parameter of the line (i.e., its a-b).
- Therefore, we can write down the equation for the “blue” line (if we wish) - which means we have detected the line successfully!



Graphically…

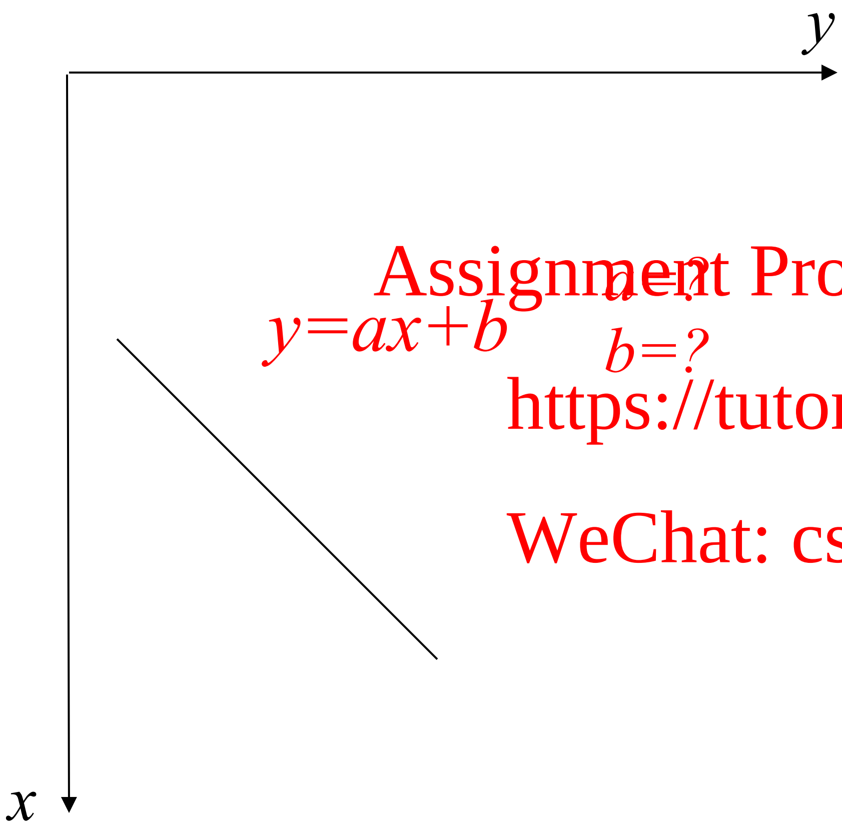


Image of a line

The line detection problem is to answer this question: what is the equation of the line?

$y = ax + b$

$a = ?$
$b = ?$

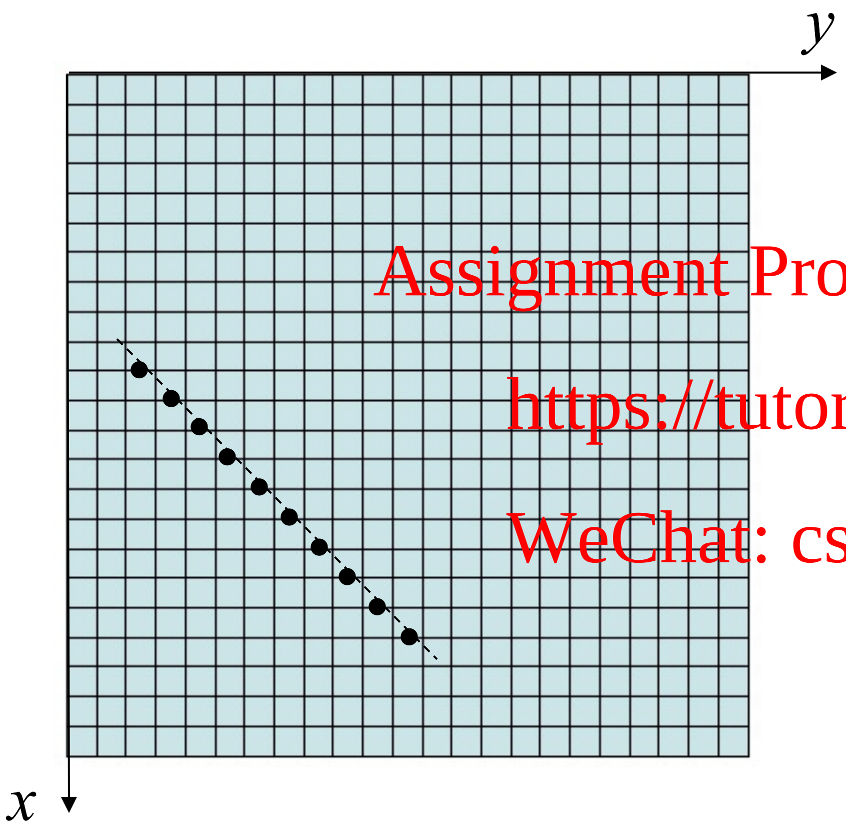
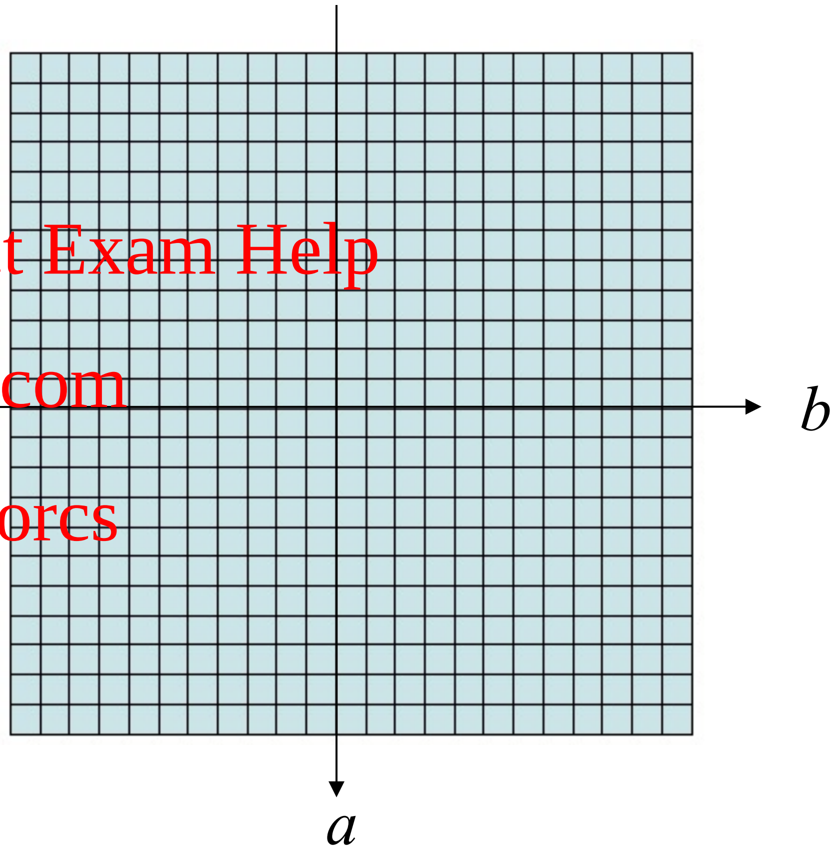
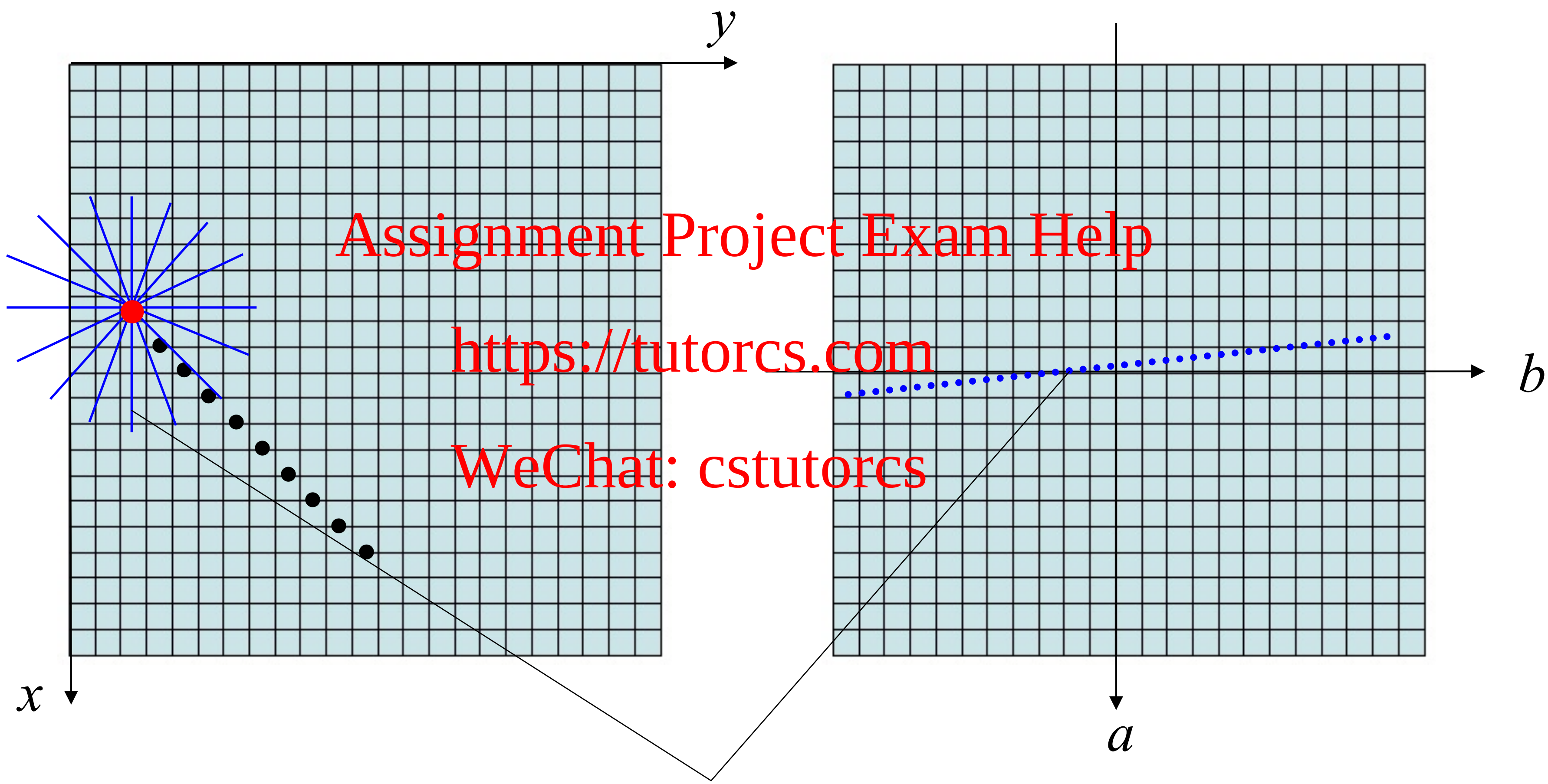


Image space (pixels)



Parameter space (each cell represents a line)

$y = ax + b$ $a = ?$ $b = ?$

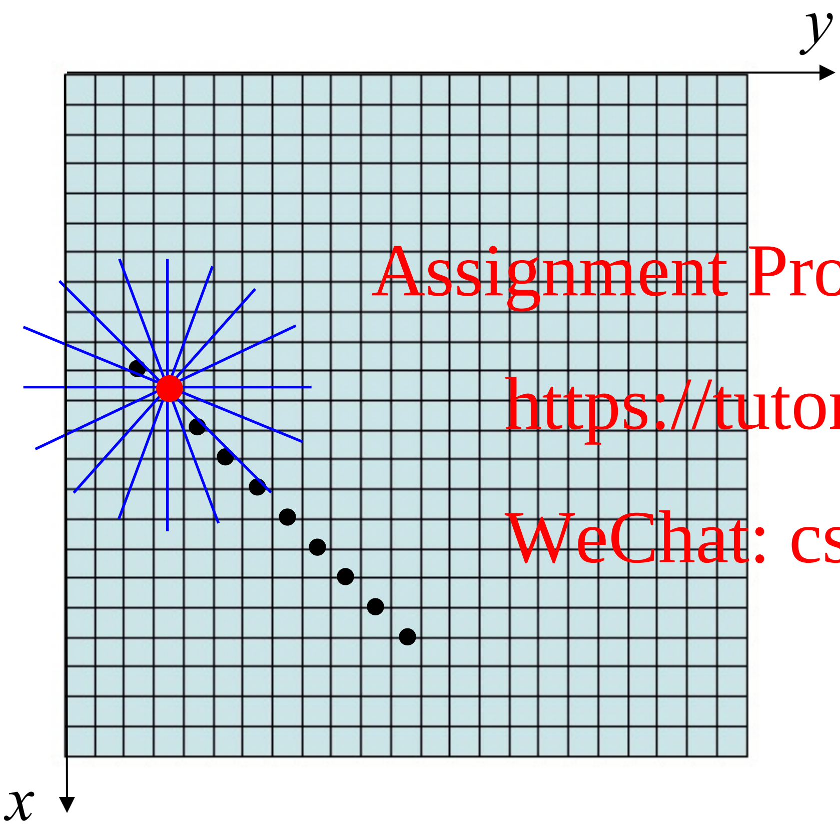


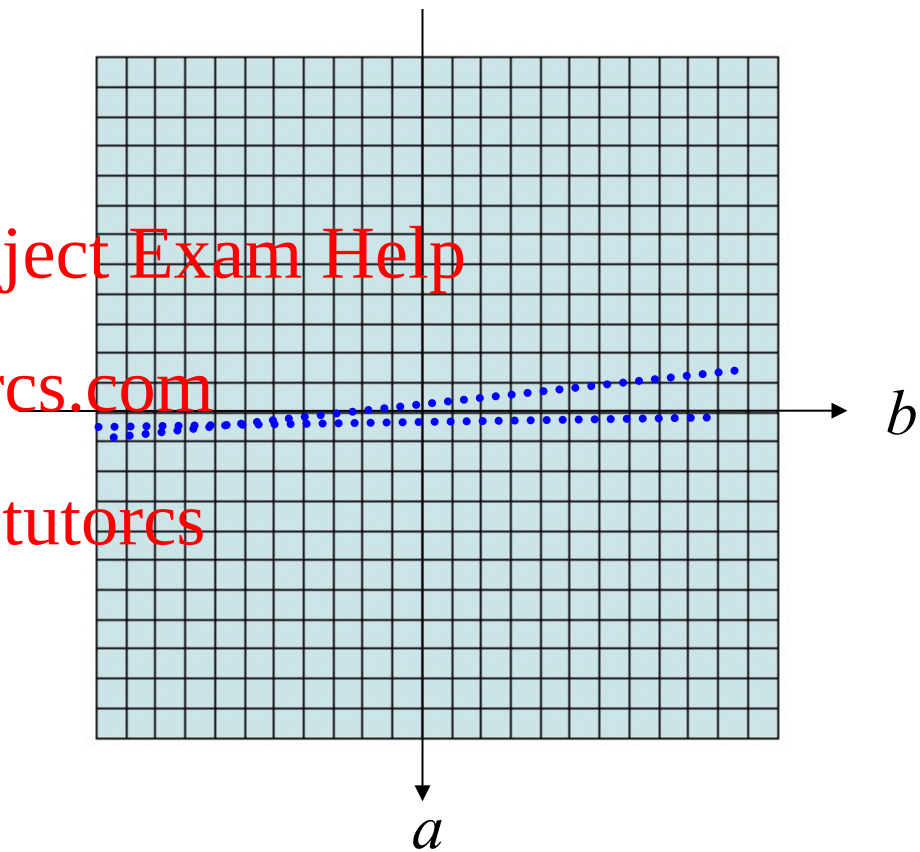
Each line in image space corresponds to a point (vote) in parameter space

$y=ax+b$

$a=?$

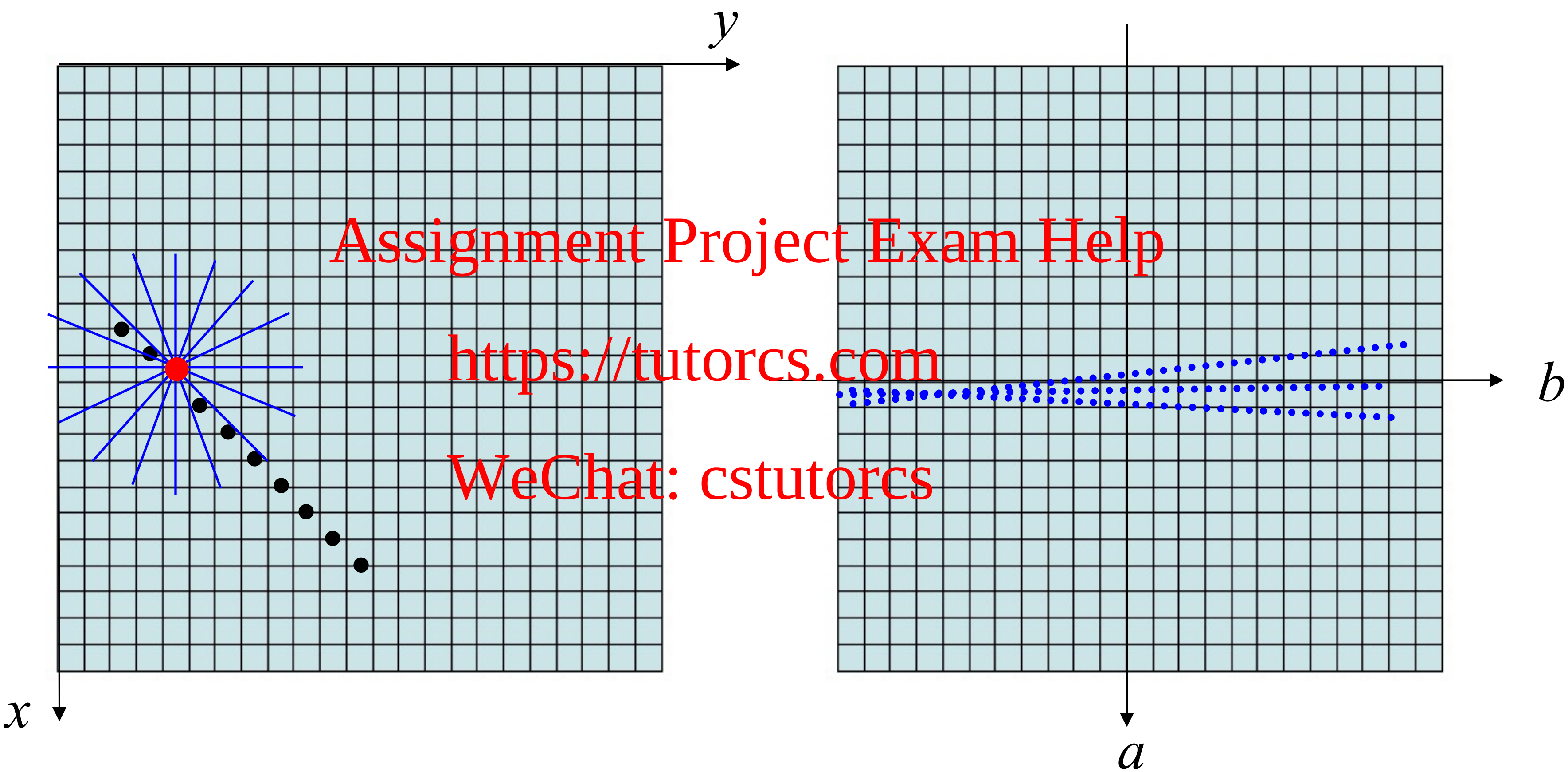
$b=?$





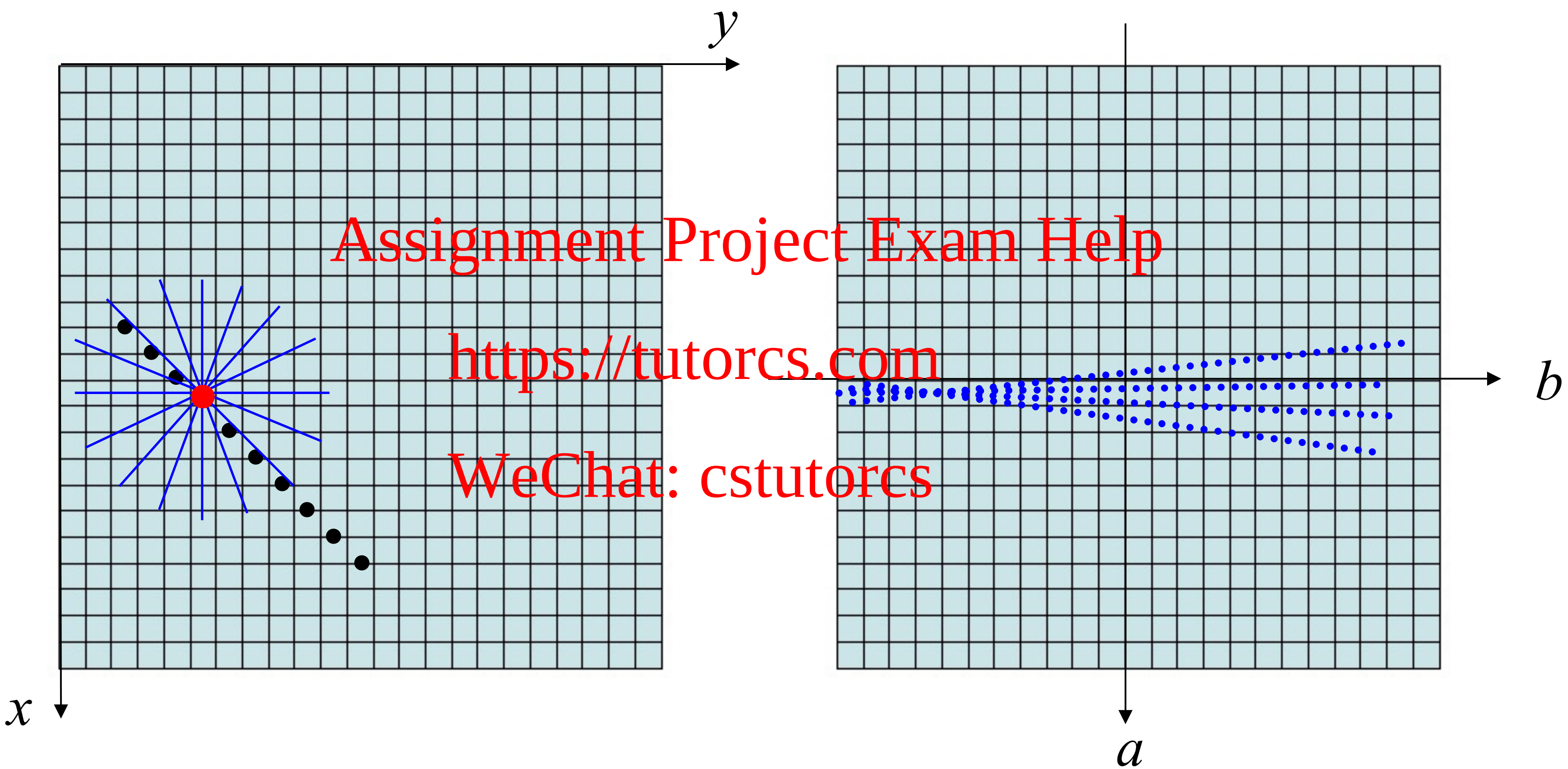
$y=ax+b$

$a=?$
$b=?$



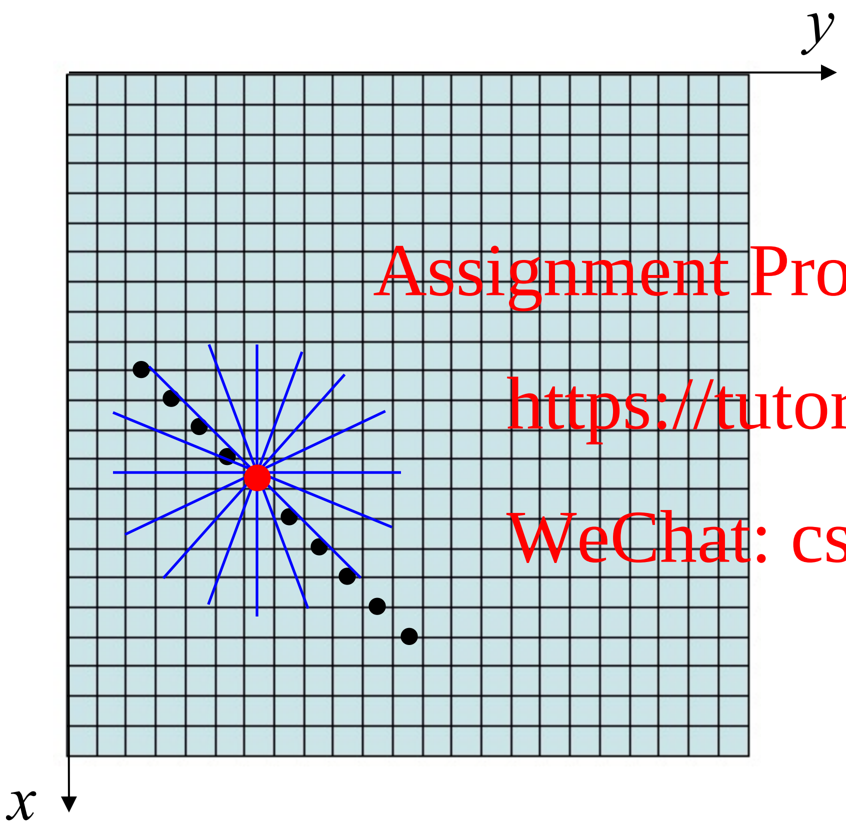
$y = ax + b$

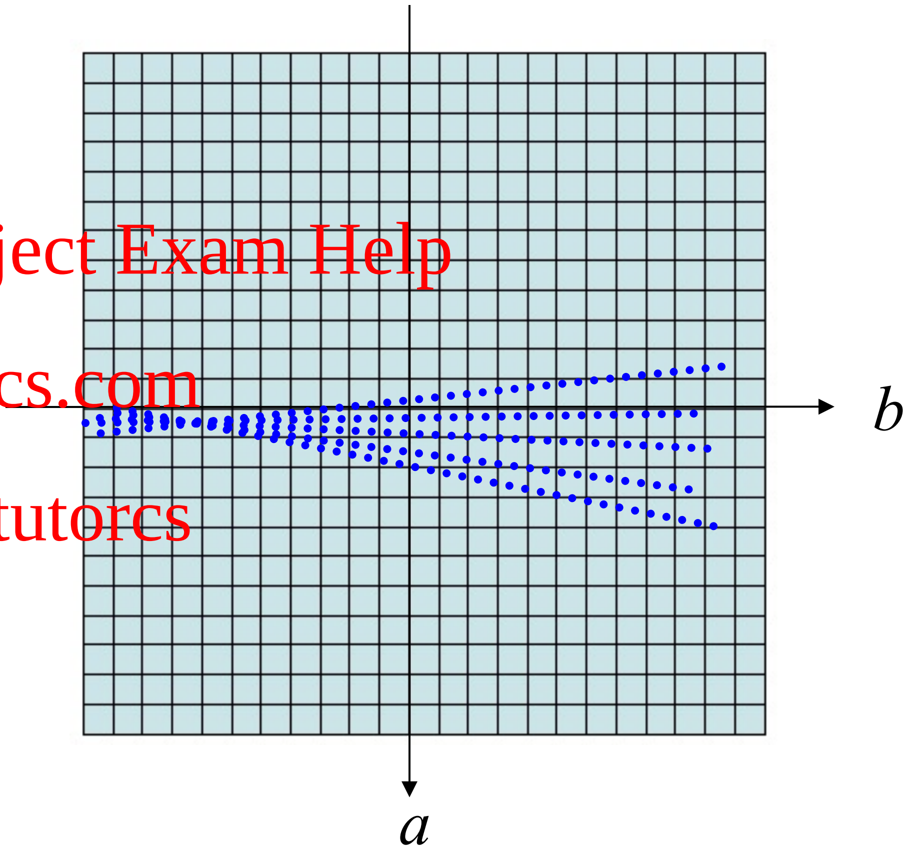
$a = ?$
$b = ?$



$y=ax+b$

$a=?$
$b=?$

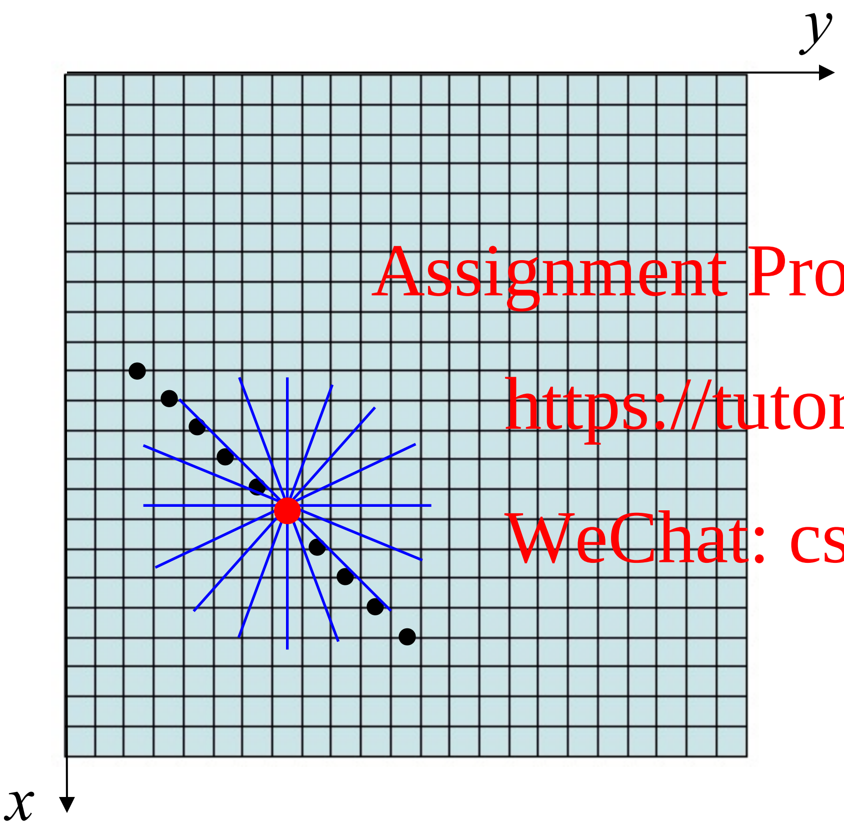


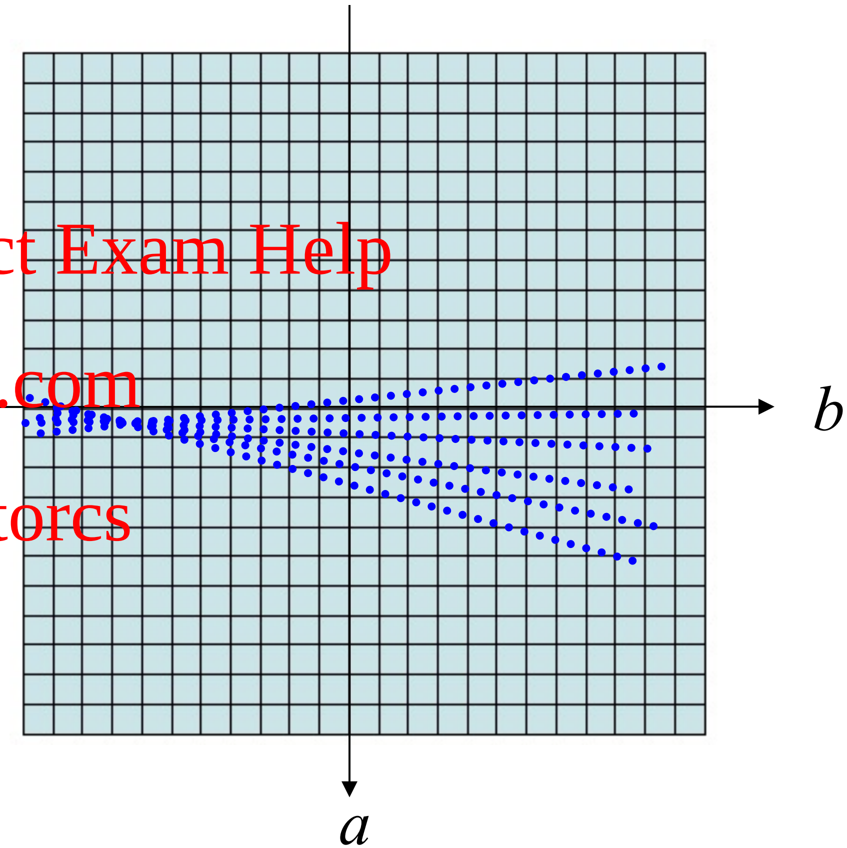


$y=ax+b$

$a=?$

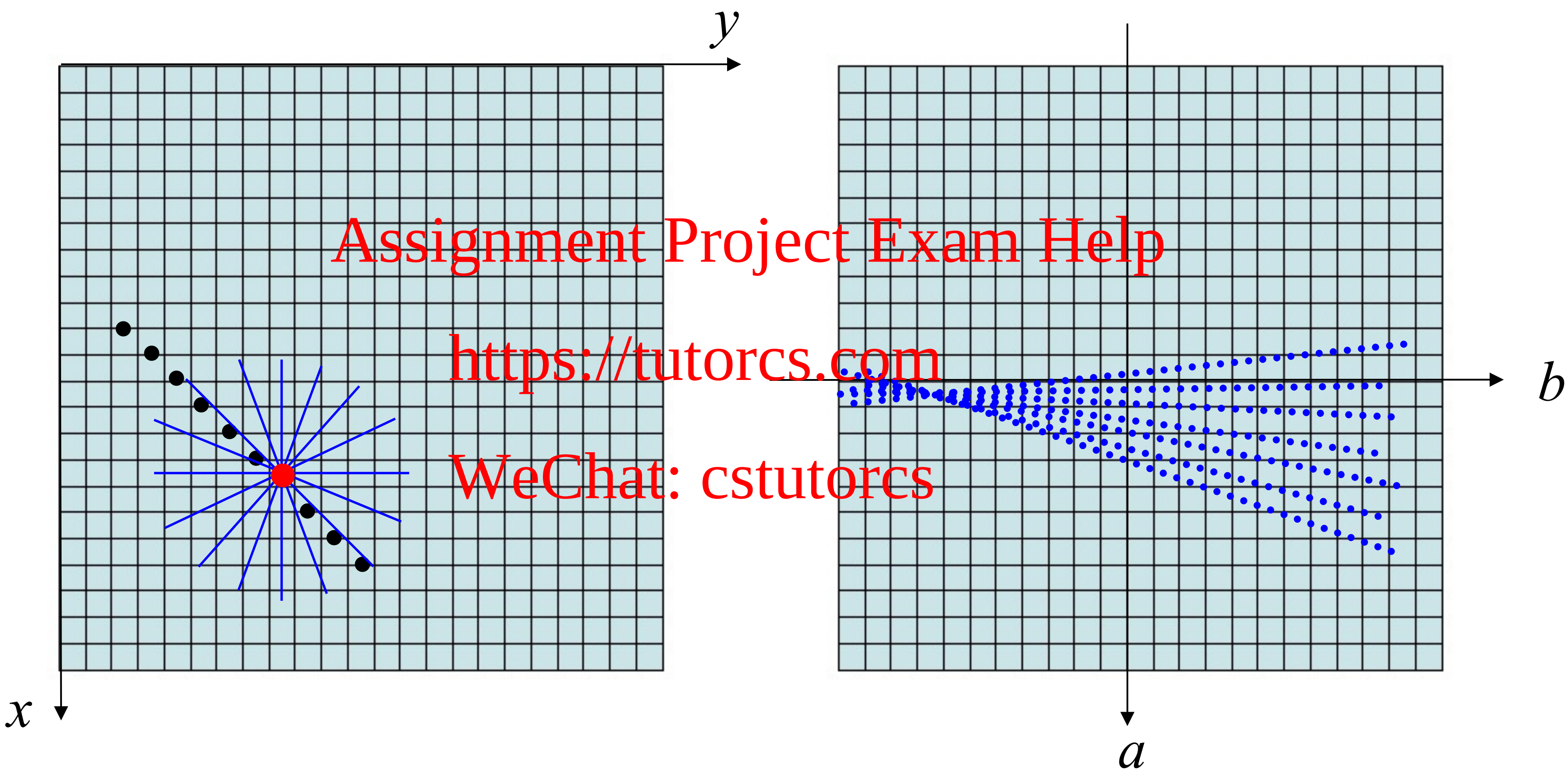
$b=?$





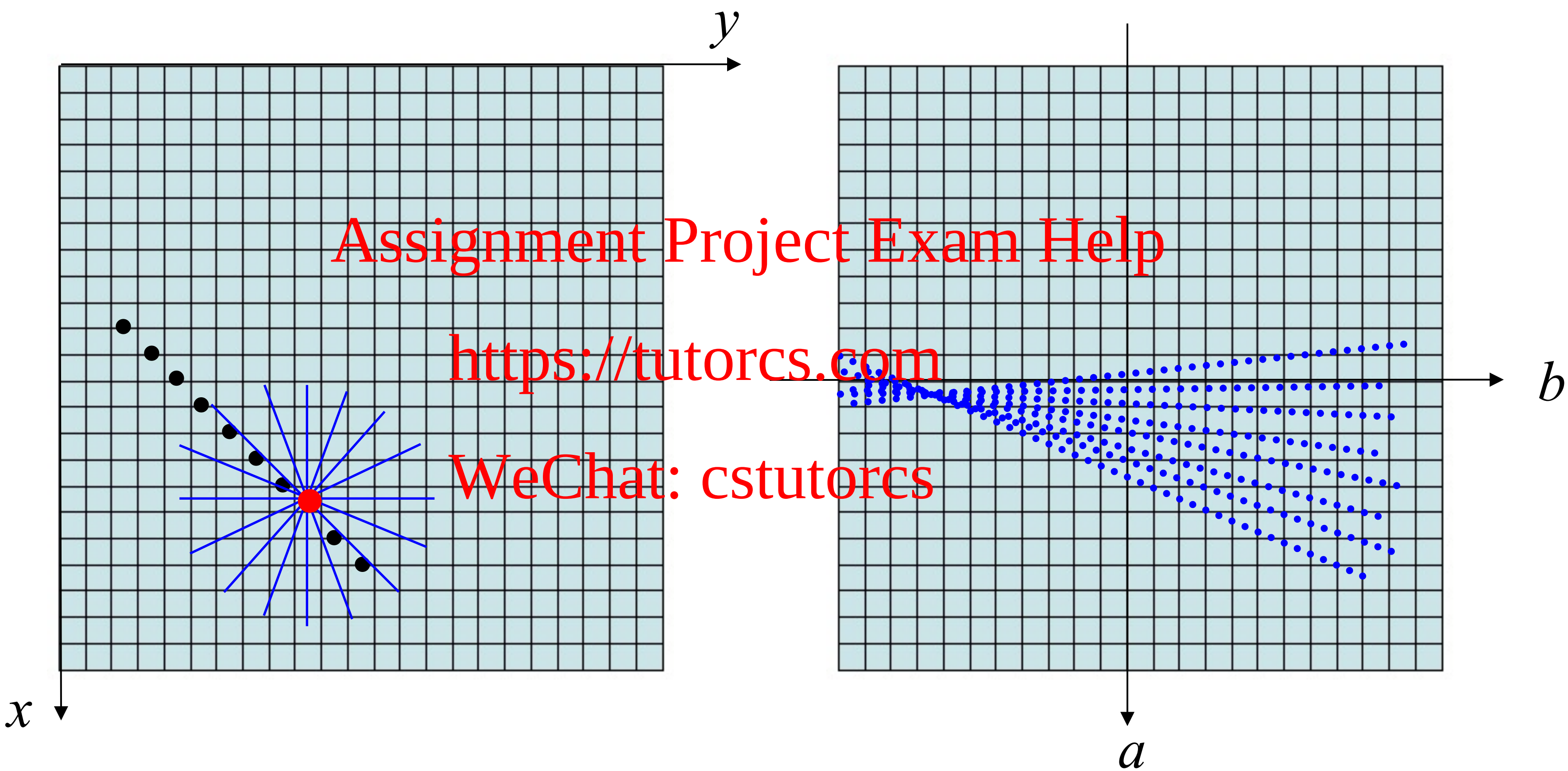
$y=ax+b$

$a=?$
$b=?$



$y = ax + b$

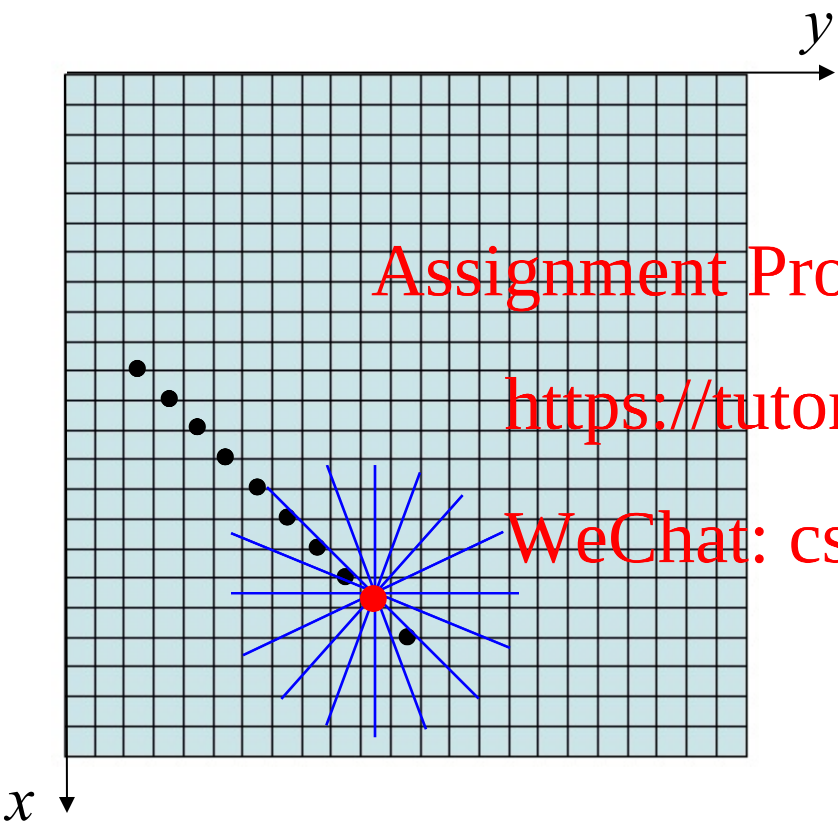
$a = ?$
$b = ?$

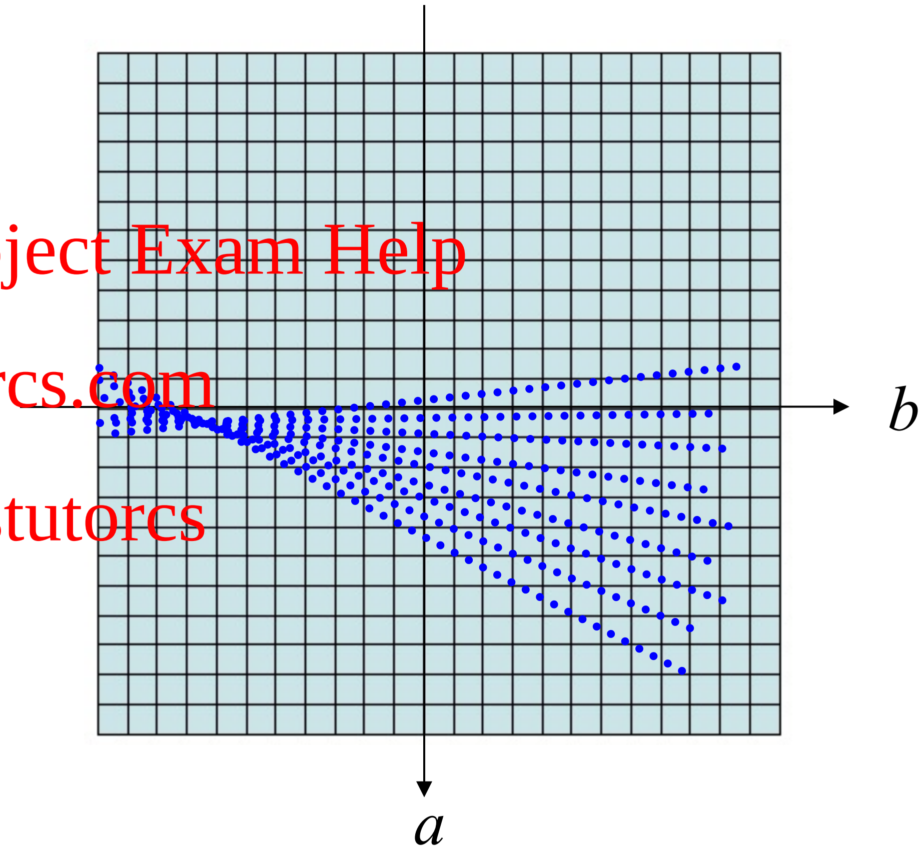


$y=ax+b$

$a=?$

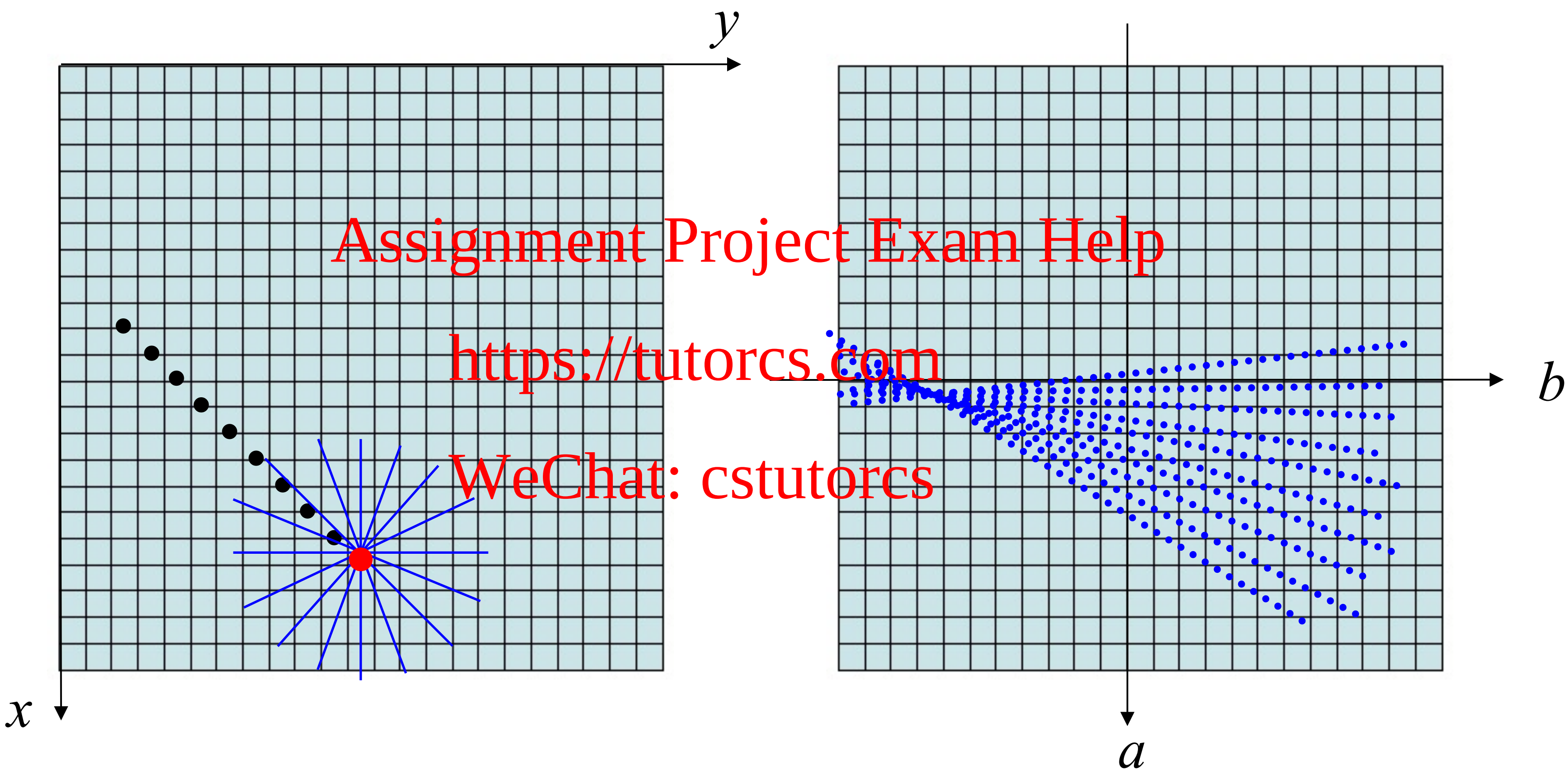
$b=?$





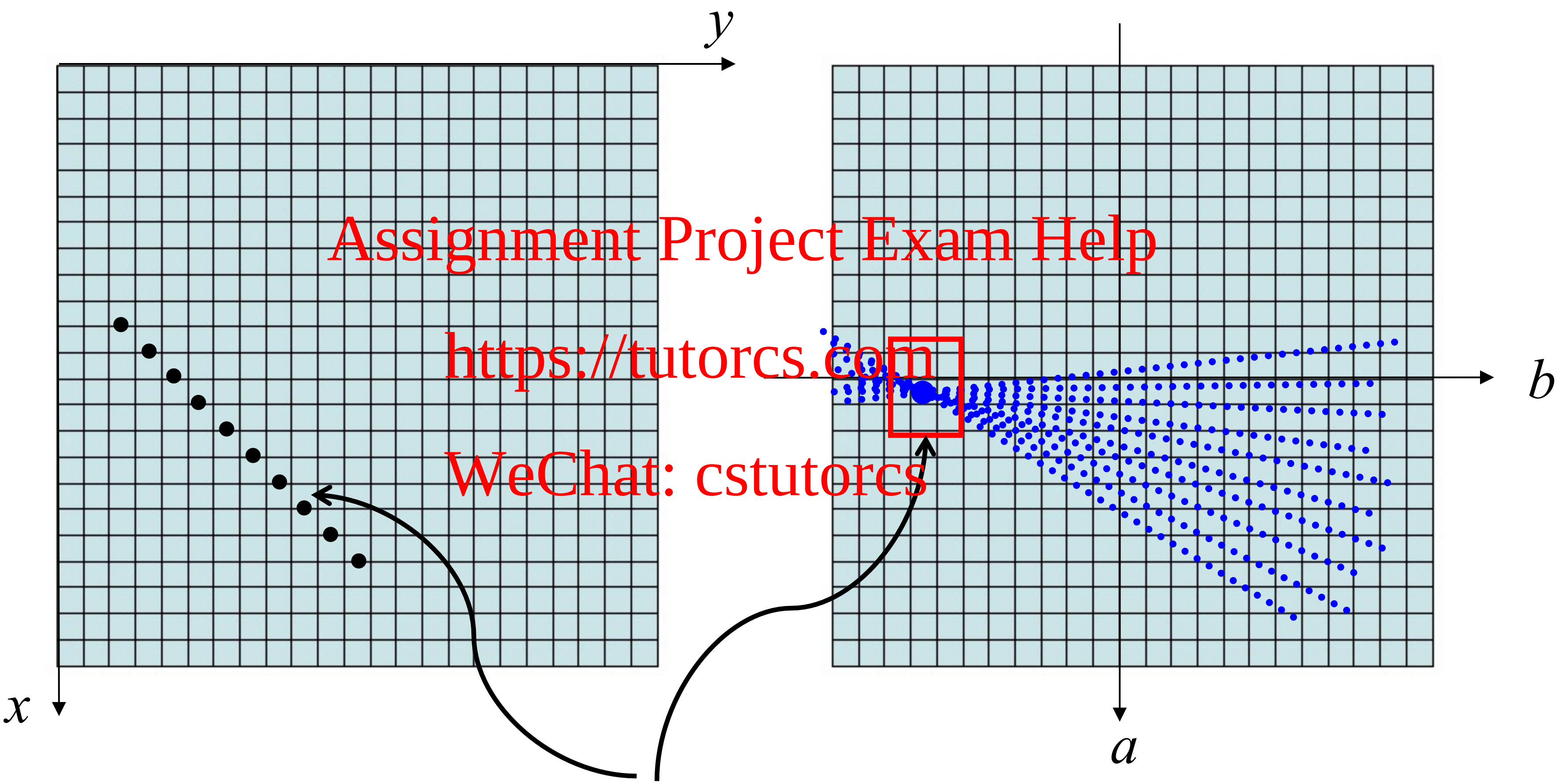
$y=ax+b$

$a=?$
$b=?$



$y=ax+b$

$a=?$
$b=?$



The cell that receives the most votes gives the a, b values of the line in the image on the left.

Think About…

- The way of dealing with the ranges of parameters: a: -∞ to +∞ and b: -∞ to +∞
  - limit the ranges to finite values, e.g., from -200 to +200
  - Is this a good choice?

- The resolution of discretisation, e.g.,
  - a: -200, -195, -190,….+185, 190, 195, 200
  - b: -200, -195, -190,….+185, 190, 195, 200
  - Is the division a good choice?
  - What implication does the resolution have on the line detection?

Further Readings

- Shapiro, L.G., Stockman, G.C., Computer Vision, Prentice-Hall, 2001, ISBN 0-13-030796-3
  - Section 10.3.4 for Hough transform