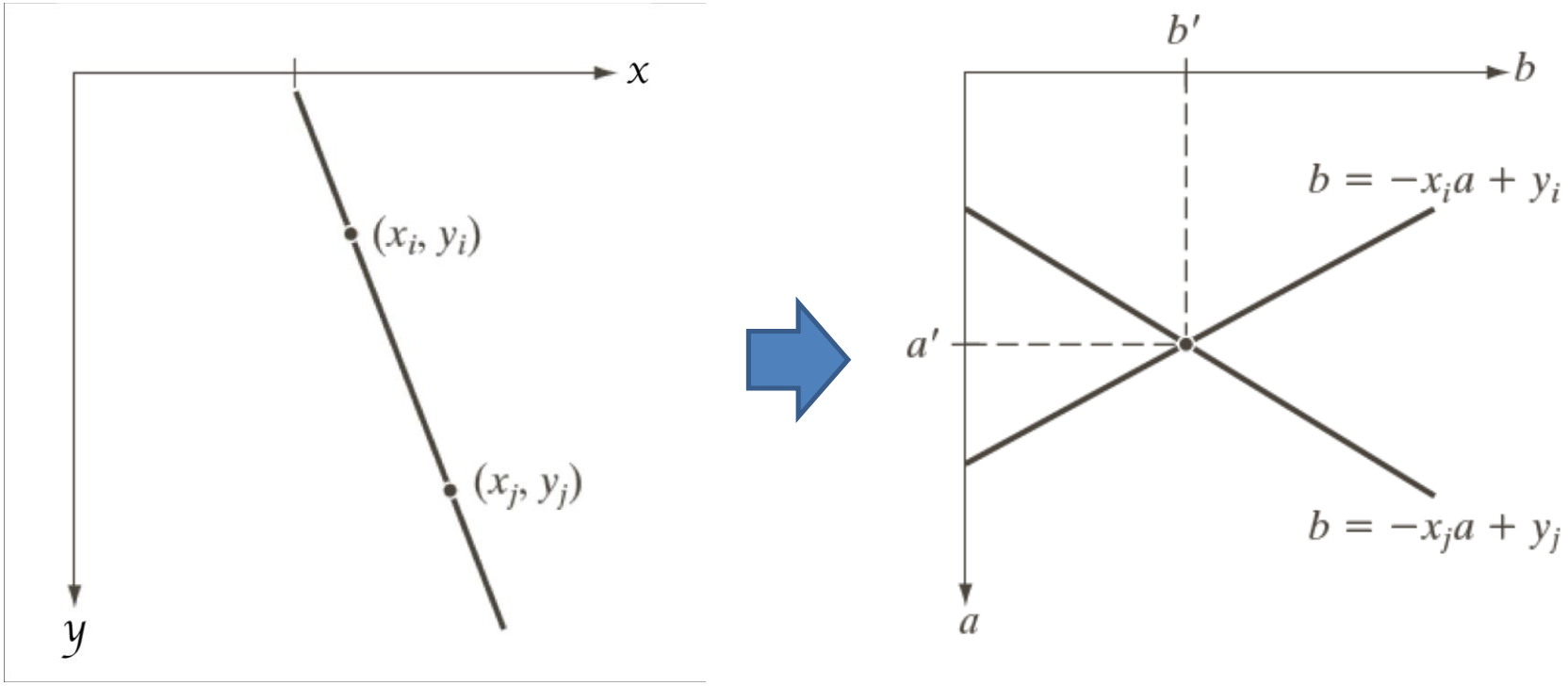
Line Detection

Sung Soo Hwang

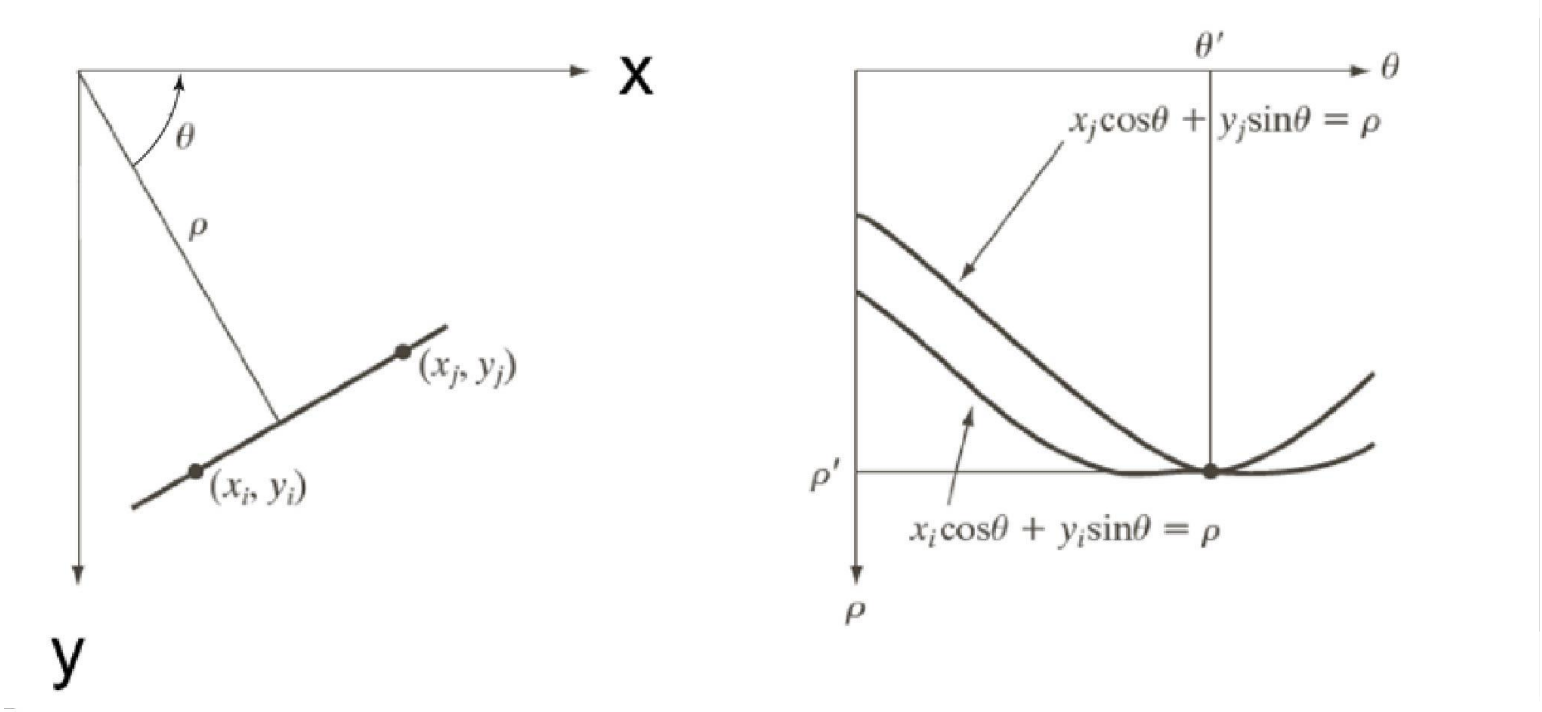
Line

- The concept of Hough transform
  - y=ax + b ➔ b= -ax + y
  - Find all combinations of (a, b) for each edge pixel
  - A combination of (a, b) which is used multiple times can be a line of an input image



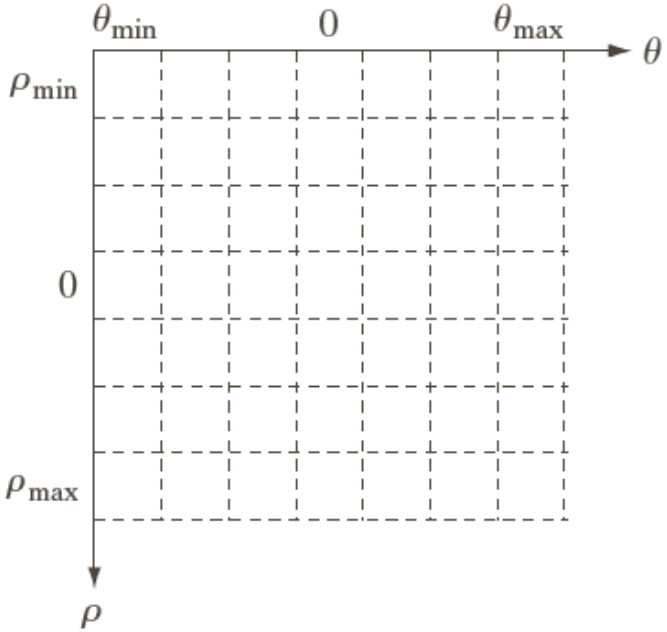
Line

- The concept of Hough transform
  - *a* becomes infinity as the line approaches the vertical direction
  - Use $\rho\theta$ representation instead
    - $xcos\theta + ysin\theta = \rho$



Hough Transform

- Algorithm
  1. obtain a binary edge image
  2. specify subdivision in the $\rho\theta$ – plane
  3. Examine the counts of the accumulator cells for high pixel concentrations
  4. Examine the relationship (connectivity) between pixels in a chosen cell



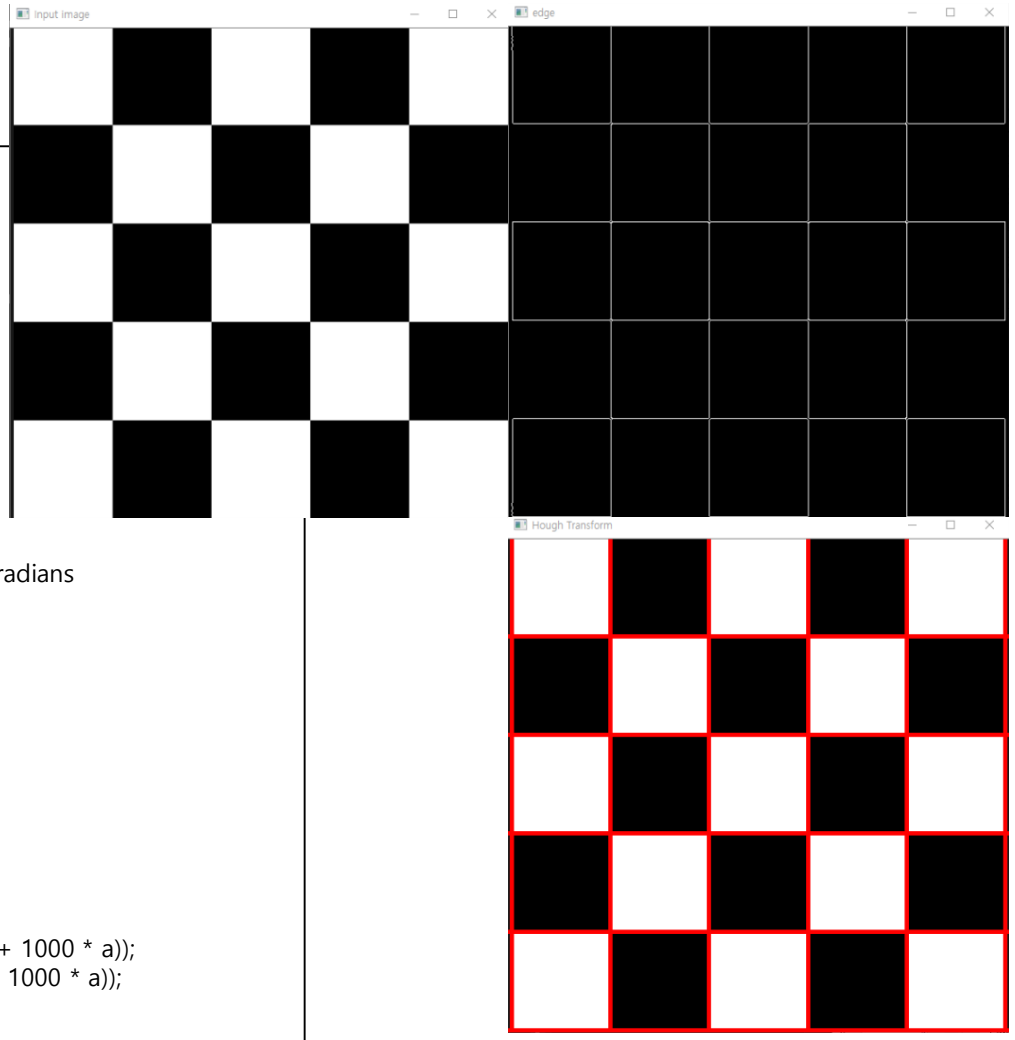
Hough Transform

- Circle detection
  - Hough transform is applicable to any function of the form g(v, c)=0, where v is a vector of coordinates and c is a vector of coefficients
    - Points lying on the circle
      - $(x - c_1)^2 + (y - c_2)^2 = c_3$
    - The result is a 3D parameter space

Line

- Houghlines

```
int main() {
	Mat image, edge, result;
	float rho, theta, a, b, x0, y0;
	Point p1, p2;
	vector<Vec2f> lines;
	image = imread("chess_pattern.png");
	result = image.clone();

	cvtColor(image, image, CV_BGR2GRAY);
	Canny(image, edge, 50, 200, 3);

	//applying Hough Transform to find lines in the image
	//edge: input Mat, lines: output vector of lines
	//1: (rho) distance resolution of the accumulator in pixels
	//CV_PI/180: (theta) angle resolution of the accumulator in radians
	//150: (threshold) accumulator threshold parameter
	HoughLines(edge, lines, 1, CV_PI / 180, 150);

	for (int i = 0; i < lines.size(); i++) {
		rho = lines[i][0];
		theta = lines[i][1];
		a = cos(theta);
		b = sin(theta);

		x0 = a * rho;
		y0 = b * rho;

		p1 = Point(cvRound(x0 + 1000 * (-b)), cvRound(y0 + 1000 * a));
		p2 = Point(cvRound(x0 - 1000 * (-b)), cvRound(y0 - 1000 * a));

		line(result, p1, p2, Scalar(0, 0, 255), 3, 8);
	}
	imshow("Input image", image);
	imshow("edge", edge);
	imshow("Hough Transform", result);
	waitKey(0);
}
```



Line

- Houghlinesp

```
int main() {
	Mat image, edge, result;
	float rho, theta, a, b, x0, y0;
	Point p1, p2;
	vector<Vec4i> lines;

	image = imread("chess_pattern.png");
	result = image.clone();
	cvtColor(image, image, CV_BGR2GRAY);
	Canny(image, edge, 50, 200, 3);
	//edge: input Mat, lines: output vector of lines
	//1: (rho) distance resolution of the accumulator in pixels
	//CV_PI/180: (theta) angle resolution of the accumulator in radians
	//50: (threshold) accumulator threshold parameter
	//10: (minLineLength) minimum line length.
	//300: (maxLineGap) Maximum allowed gap between points on the
	same line to link them
	HoughLinesP(edge, lines, 1, CV_PI / 180, 50, 10, 300);

	for (int i = 0; i < lines.size(); i++) {
		Vec4i l = lines[i];
		line(result, Point(l[0], l[1]), Point(l[2], l[3]), Scalar(0, 0, 255), 3, 8);
	}

	imshow("Input image", image);
	imshow("edge", edge);
	imshow("Hough Transform", result);
	waitKey(0);
}
```