Fitting and Matching

Bryan Chiang and Silvio Savarese

June 19, 2022

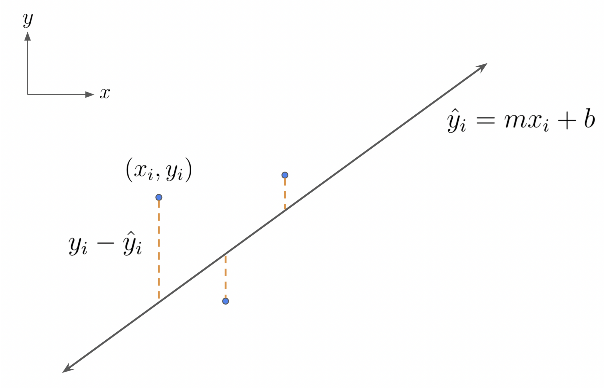
1 Overview

The goal of **fitting** is to find a parametric model that best describes the observed data. We obtain the optimal parameters of such a model by minimizing a chosen **fitting error** betwee the data and a particular estimate of the model parameters. A classic example is fitting a line to a set of given (x, y) points. Other examples we've seen in this class include computing a 2D homography H between set of point correspondences in different images or computing the fundamental matrix F using the eight-point algorithm.

2 Least-squares

Given a series of N 2D points $\{(x_i, y_i)\}_{i=1}^N$, the method of **least-squares** fitting attempts to find a line $y = mx + b$ such that the squared error in the y dimension is minimized, as illustrated in Figure (1).

Figure 1: Ordinary least squares.



Specifically, we want to find model parameters $w = \begin{bmatrix} m & b \end{bmatrix}^T$ to minimize the sum of squared residuals between y_i and the model estimate $\hat{y}_i = mx_i + b$, given in Equation (1). We define the residual as $y_i - \hat{y}_i$.

$$\begin{align} E &= \sum_{i=1}^{N} (y_i - \hat{y}_i)^2 \tag{1} \\ &= \sum_{i=1}^{N} (y_i - mx_i - b)^2 \tag{2} \end{align}$$

We can write this in matrix notation as:

$$\begin{align} E &= \sum_{i=1}^{N} (y_i - \begin{bmatrix} x_i & 1 \end{bmatrix} \begin{bmatrix} m \\ b \end{bmatrix})^2 \tag{3} \\ &= \| \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix} - \begin{bmatrix} x_1 & 1 \\ \vdots & \vdots \\ x_N & 1 \end{bmatrix} \begin{bmatrix} m \\ b \end{bmatrix} \|^2 \tag{4} \\ &= \|Y - Xw\|^2 \tag{5} \end{align}$$

The residual is now $r = y - Xw$, we assume X to be skinny and full rank. We want to find the B that minimizes the norm of the residual squared, which we can write as:

$$\begin{align} \|r\|^2 &= r^T r \tag{6} \\ &= (y - Xw)^T (y - Xw) \tag{7} \\ &= y^T y - 2y^T Xw + w^T X^T Xw \tag{8} \end{align}$$

We then set the gradient of the residual with respect to w equal to 0. Recall $X^T X$ is symmetric.

$$\begin{align} \nabla_w \|r\|^2 &= -2X^T y + 2X^T Xw \tag{9} \\ &= 0 \tag{10} \end{align}$$

This leads to the normal equations.

$$X^T Xw = X^T y \tag{11}$$

We now have a closed-form solution for w in Equation (12). A is full rank so A^TA is invertible.

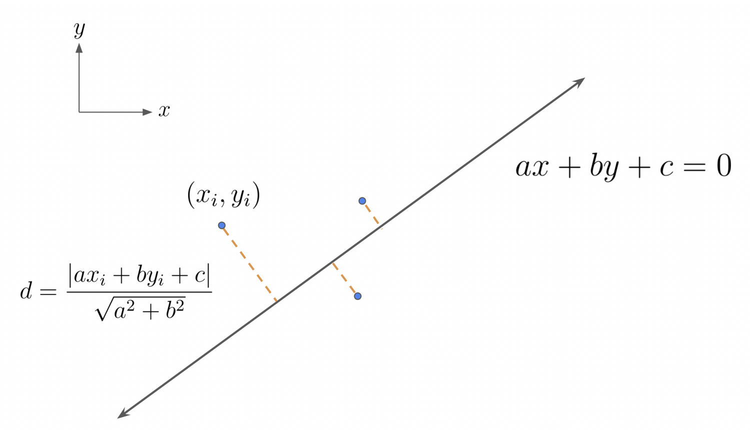
$$w = (X^TX)^{-1}X^Ty \tag{12}$$

However, note that this method fails completely for fitting points that describe a vertical line (m undefined). In this case, m would be set to extremely large number, leading to numerically unstable solutions. To fix this, we can use an alternate line formulation of the form $ax+by+d=0$. We can obtain a vertical line by setting $b=0$. Here's one way to think about this line representation. The line direction (slope) is given by $\vec{\mathbf{n}}$; the set of (x,y) satisfying $(x,y)\cdot(a,b)=xa+by=0$ is the line orthogonal to $\vec{\mathbf{n}}$. However, the line can also be arbitrarily shifted to location (x_0,y_0), so we have

$$\begin{aligned} a(x-x_0)+b(y-y_0) &= ax+by-ax_0-by_0 && (13)\\ &= ax+by+c && (14)\\ &= 0 && (15)\end{aligned}$$

where $c=-ax_0-by_0$. The slope of line is then $m=-\frac{a}{b}$, which now may be undefined. Earlier our residual was only in the y-axis. However, now that our new line parameterization accounts for error in the both the x- and y-axes, our new error is the sum of squared orthogonal distances, as illustrated in Figure (2).

Figure 2: Total least squares.

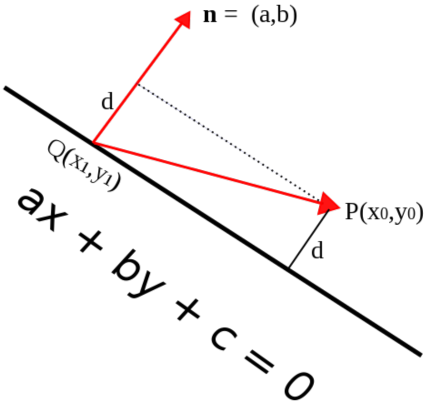


Given a 2D data point $P=(x_i,y_i)$ and a point on the line $Q=(x,y)$, the distance from P to the line is equivalent to the length of the projection of $\overrightarrow{QP}$ onto the normal vector $\vec{\mathbf{n}}$ orthogonal to the line. We have $\overrightarrow{QP} = (x_i - x, y_i - y), \vec{\mathbf{n}} = (a, b)$, which gives:

$$
\begin{aligned}
d &= \frac{|\overrightarrow{QP} \cdot \vec{\mathbf{n}}|}{\|\vec{\mathbf{n}}\|} && (16) \\
&= \frac{|a(x_i - x) + b(y_i - y)|}{\sqrt{a^2 + b^2}} && (17) \\
&= \frac{|ax_i + by_i + c|}{\sqrt{a^2 + b^2}} && (18)
\end{aligned}
$$

Recall Q lies on the line, so $c = -ax - by$.

Figure 3: Distance between a point and a line.



Our new set of parameters is now $w = \begin{bmatrix} a & b & c \end{bmatrix}^T$. To simplify the error, we make the solution unique and remove the denominator by constraining $\|\vec{\mathbf{n}}\|^2 = 1$, so the new error is

$$
\begin{aligned}
E(a, b, x_0, y_0) &= \sum_{i=1}^{N} (a(x_i - x_0) + b(y_i - y_0))^2 && (19) \\
&= \sum_{i=1}^{N} (ax_i + by_i + c)^2 && (20)
\end{aligned}
$$

where $a^2 + b^2 = 1$. However, putting this into matrix notation is still tricky due to the presence of c when the constraint is only on a, b. To simplify further, we note that the resulting line of best fit that minimizes E must pass through the data centroid $(\bar{x}, \bar{y})$, defined as

$$\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i \tag{21}$$

$$\bar{y} = \frac{1}{N}\sum_{i=1}^{N} y_i \tag{22}$$

For every $\vec{\mathbf{n}}$ and every possible set of points $\{(x_i, y_i)\}_{i=1}^{N}$, E is minimized when we set $c = -a\bar{x} - b\bar{x}$. In other words, given every point $(x_0, y_0) \in \mathbb{R}^2$, we have

$$E(a, b, x_0, y_0) \geq E(a, b, \bar{x}, \bar{y}) \tag{23}$$

To see why this is true, we define vectors w, z such that

$$w_i = a(x_i - x_0) + b(y_i - y_0) \tag{24}$$

$$z_i = a(x_i - \bar{x}) + b(y_i - \bar{y}) \tag{25}$$

We can then write the errors as

$$E(a, b, x_0, y_0) = \|w\|^2 \tag{26}$$

$$E(a, b, \bar{x}, \bar{y}) = \|z\|^2 \tag{27}$$

The relationship between w and z is then

$$w = z + h\mathbf{1} \tag{28}$$

where $h = a(\bar{x} - x_0) + b(\bar{y} - y_0) \in \mathbb{R}$ and $\mathbf{1}$ is a vector of all ones. z is orthogonal to $\mathbf{1}$ since

$$\begin{aligned}
z \cdot \mathbf{1} &= \sum_{i=1}^{N} z_i && (29)\\
&= a\sum_{i=1}^{N}(x_i - \bar{x}) + b\sum_{i=1}^{N}(y_i - \bar{y}) && (30)\\
&= a(\sum_{i=1}^{N} x_i - N(\frac{1}{N}\sum_{i=1}^{N} x_i)) + b(\sum_{i=1}^{N} y_i - N(\frac{1}{N}\sum_{i=1}^{N} y_i)) && (31)\\
&= 0a + 0b = 0 && (32)
\end{aligned}$$

Thus, by the Pythagorean theorem, we have

$$\begin{aligned}
E(a, b, x_0, y_0) &= \|w\|^2 && (33)\\
&= ||z||^2 + h^2 N && (34)\\
&\geq ||z||^2 = E(a, b, \bar{x}, \bar{y}) && (35)
\end{aligned}$$

We have shown that the line of best fit must pass through $(\bar{x}, \bar{y})$, so we can constrain c as

$$c = -a\bar{x} - b\bar{y} \tag{36}$$

We can then eliminate c by shifting all points to be centered around the data centroid (setting $(x_0, y_0) = (\bar{x}, \bar{y})$), which allows us to finally formulate the error as a matrix product.

$$\begin{aligned}
E &= \sum_{i=1}^{N} (a(x_i - \bar{x}) + b(y_i - \bar{y}))^2 && (37)\\
&= \| \begin{bmatrix} x_1 - \bar{x} & y_1 - \bar{y} \\ \vdots & \vdots \\ x_N - \bar{x} & y_N - \bar{y} \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} \|^2 && (38)\\
&= \|Xw\|^2 && (39)
\end{aligned}$$

where $w = \begin{bmatrix} a & b \end{bmatrix}^T$ and $\|w\|^2 = 1$. This is a constrained least-squares problem that we've seen before in previous lectures. By SVD (X full rank), we have

$$X = USV^T \tag{40}$$

$U \in \mathbb{R}^{N \times M}, V^T \in \mathbb{R}^{M \times M}$ are both orthonormal matrices, while $S \in \mathbb{R}^{M \times M}$ is a diagonal matrix containing the singular values of X in descending order. $M = 2$ here. Since U, V are orthonormal, we know that

$$\|USV^T w\| = \|SV^T w\| \tag{41}$$

$$\|V^T w\| = \|w\| \tag{42}$$

Setting $v = V^T w$, we can now minimize $\|SV^T w\|$ with the new but equivalent constraint that $\|v\|^2 = 1$. $\|SV^T w\| = \|Sv\|$ is minimized when

$v = \begin{bmatrix} 0 & 1 \end{bmatrix}^T$ since the diagonal of S is sorted in descending order. Finally, we obtain $w = VV^Tw = Vv$, so the w that minimizes the error is the last column in V.

Interpreting this from a gain perspective, we can write the SVD $X = USV^T$ as

$$X = \sum_{i=1}^{M} \sigma_i u_i v_i^T \tag{43}$$

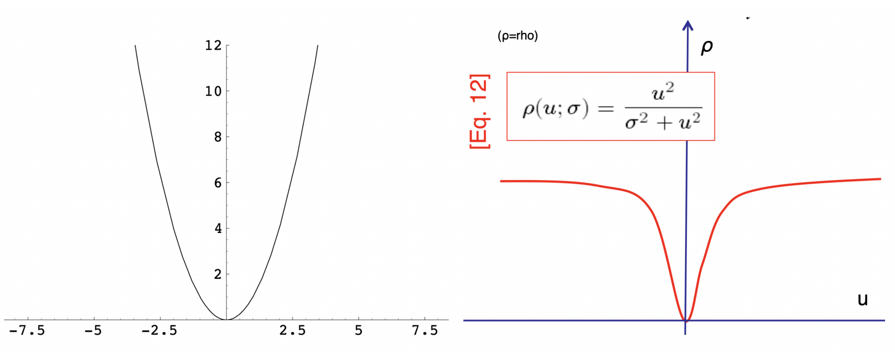
$v_1, \ldots, v_M$ are columns of V, σ_i are the diagonal values of $S = \text{diag}(\sigma_1, \ldots, \sigma_M)$, and $u_1, \ldots, u_M$ are the columns of U. Multiplying w by the SVD, USV^Tw, can then be viewed as first computing the components of w along the input directions $v_1, \ldots, v_M$, scaling the components by σ_i, and then reconstituting along the output directions $u_1, \ldots, u_M$. V^Tw gives the projection of w along each column in V (recall $\|v_i\|^2 = 1$). Similarly, Uw' can be seen as a linear combination of the output directions, $u_1w'_1 + \cdots + u_Mw'_M$. Thus, to find w that minimizes Xw subject to $\|w\|^2 = 1$ is simply choosing the input direction that minimizes the magnitude of the output vector, which is the last column of V.

In practice, least-squares fitting handles noisy data well but is susceptible to outliers. To see why, if we write the residual for the i-th data point as $u_i = ax_i + by_i + c$, and the cost as $C(u_i)$, our error can be generalized to

$$E = \sum_{i=1}^{N} C(u_i) \tag{44}$$

The quadratic growth of the squared error $C(u_i) = u_i^2$ that we've been using so far (illustrated on the left side of Figure (4) means that outliers with large residuals u_i exert an outsized influence on cost minimum.

Figure 4: Cost functions: squared residual (left) vs. robust cost function (right). The x-axis error is the residual u_i, and the y-axis is the cost $C(u_i)$.

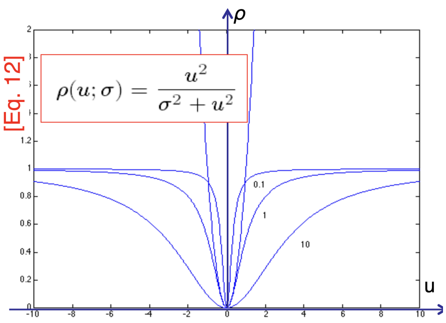


We can penalize large residuals (outliers) less by a robust cost function (right half of Figure (4)), such as

$$C(u_i, \sigma) = \frac{u_i^2}{\sigma^2 + u_i^2} \tag{45}$$

When the residual u_i is large, the cost C saturates to 1 such that their contribution to the cost is limited, but when u is small, the cost function resembles the squared error. However, now we need to choose σ, also known as the **scale parameter**. σ controls how much weight is given to potential outliers, illustrated in Figure (5). A large σ widens the quadratic curve in the center, penalizing outliers more relative to other points (similar to the original squared error function). A small σ narrows the quadratic curve, penalizing outliers less. If σ is too small, then most of the residuals will be treated as outliers even when they are not, leading to a poor fit. If σ is too large, then we do not benefit from the robust cost function and end up with the least-squares fit.

Figure 5: Comparison of different scale parameters.

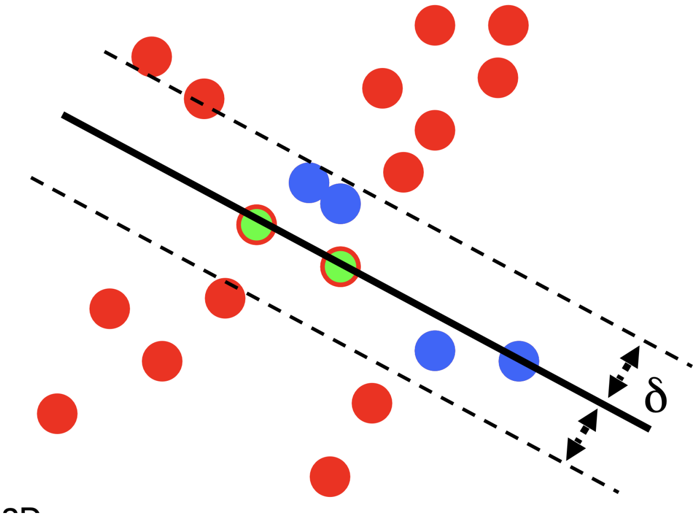


Since the robust cost functions are non-linear, they are optimized with iterative methods. In practice, the closed-form least-squares solution is often used as a starting point, followed by iteratively fitting the parameters with a robust non-linear cost function. For further details on robust estimators, please refer to Appendix 6.8 “Robust cost function” in [HZ]. In addition, while we derived least-squares fitting for a line in the notes; section 4.1 in [HZ] concisely covers the **direct linear transformation** (DLT) algorithm which is used to compute a homography using the same constrained least-squares optimization procedure.

3 RANSAC

Another fitting method called **RANSAC**, which stands for random sample consensus, is designed to be robust to outliers and missing data. We demonstrate using RANSAC to perform line fitting, but it generalizes to many different fitting contexts.

Figure 6: RANSAC procedure for line fitting.



We again have a series of N 2D points $X = \{(x_i, y_i)\}_{i=1}^{N}$ that we want to fit a line to, illustrated by the dots in Figure (6). The first step in RANSAC is to randomly select the minimum number of points needed to fit a model. A line requires at least two points, so we choose the two points in green. If we were estimating the fundamental matrix F, we would need to choose 8 correspondences to use the eight-point algorithm. If we wanted to compute a homography $H \in \mathbb{R}^{3\times 3}$, we would need 4 correspondences (we have two x, y coordinates for each correspondence) to cover the 8 degrees of freedom up to scale. The second step in RANSAC is to fit a model to the random sample set. Here, the two points in green are fitted (i.e., a line is drawn between them) to obtain the line in black. The third step step is to use the fitted model to compute the **inlier set** from the entire dataset. Given the model parameters w, we define the inlier set as $P = \{(x_i, y_i) \mid r(p = (x_i, y_i), w) < \delta\}$, where the r is the residual between a data point and the model and δ is some arbitrary threshold. Here, the inlier set is represented by the green and blue points. The size of the inlier set, $|P|$, indicates how much of the entire set of points agrees with the fitted model. With P, we can also obtain the **outlier set**, defined as $O = X \setminus P$. The outlier set here is comprised of the red points. We repeat these steps for a finite number of iterations M with a new random sample each time until the size of the inlier set is maximized. While RANSAC is simple and easy to implement for different fitting scenarios, we need to tuen several parameters such as the number of times to sample n and the tolerance threshold δ. RANSAC also assumes that there are a sufficient number of inliers to agree on a good model, and there is no upper bound on number of times to compute the parameters (except until exhaustion, which is often computationally unfeasible).

However, it's unecessary try every possible sample. We can estimate the number of iterations n to guarantee with probability p at least one random sample with an inlier set free of "real" outliers for a given s (minimum number of points required to fit a model) and $\epsilon \in [0, 1]$ (proportion of outliers). Now, we have that the chance that a single random sample of s points contains all inliers is $(1-\epsilon)^s$, so the chance that a single random sample contains at least one outlier is $1-(1-\epsilon)^s$. Now, the chance that all n samples contain at least one outlier is $(1-(1-\epsilon)^s)^n$, so the chance that at least one of the n samples does not contain any outliers is $p = 1-(1-(1-\epsilon)^s)^n$. We can now derive n as f

$$1 - p = (1-(1-\epsilon)^s)^n \tag{46}$$

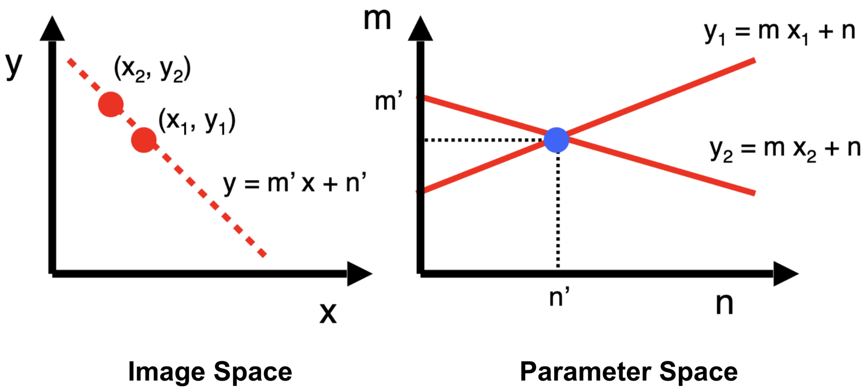
$$\log(1-p) = n\log(1-(1-\epsilon)^s) \tag{47}$$

$$n = \frac{\log(1-p)}{\log(1-(1-\epsilon)^s)} \tag{48}$$

4 Hough transform

We introduce another fitting method known as the **Hough transform**, which is another voting procedure.

Figure 7: Hough transform.



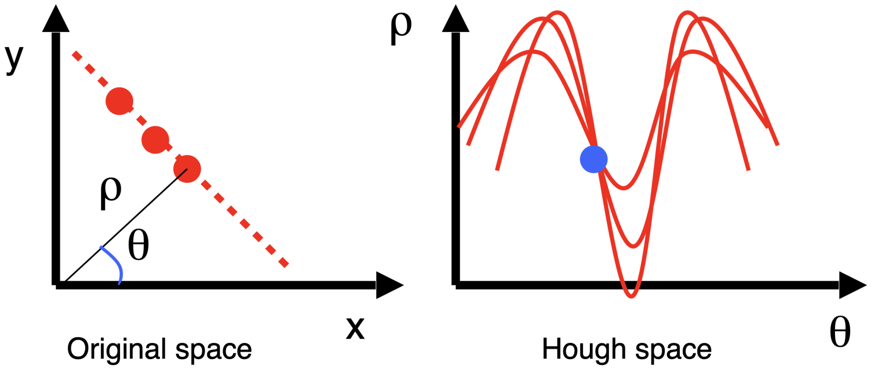
We again want to fit a line of the form $y = m'x + n'$ to a series of points $\{(x_i, y_i)\}_{i=1}^N$ in an image, illustrated on the left half of Figure (7). To find this line, we consider the dual parameter, or **Hough space**, illustrated on the right half of Figure (7). A point (x_i, y_i) in the image space (on the line $y_i = mx_i + n$) becomes a line in the parameter space defined by $n = -x_i m + y_i$. Similarly, a point in the parameter space (m, n) is a line in the image space given by $y = mx + n$.

We see that a line in the parameter space $n = -x_i m + y_i$ represents all of the different possible lines in the image space that pass through the point (x_i, y_i) in the image space. Thus, to find the line in the image space that fits both image points (x_1, y_1) and (x_1, y_1), we associate both points with lines in the Hough space and find the point of intersection (m', n'). This point in the Hough space represents the line in the image space that passes through both points in the image space. In practice, we would divide the Hough space into a discrete grid of square cells with width w for the parameter space. We would maintain a grid of counts for every $w \times w$ cell centered at (m, n) denoted $A(m, n) = 0$ for all (m, n) initially. For every data point (x_i, y_i) in the image space, we would find all (m, n) satisfying $n = -x_i m + y_i$ and increment the count by 1. After we do this for all data points, the point (m, n) in the Hough space with the highest count represent the fitted lines in the image space. We now see why this is a voting procedure: each data element (x_i, y_i) can contribute up to one vote for each candidate line in the image space (m, n).

However, there is a major limitation with the existing parameterization. As we discussed before with least-squares, the slope of a line in the image space is unbounded $-\infty < m < \infty$. This makes Hough voting an computationally and memory intensive algorithm in practice since there is no limit on the size of the parameter space that we are maintaining counts for. To solve this, we turn to the polar parameterization of a line, illustrated in Figure (8).

$$x\cos(\theta) + y\sin(\theta) = \rho \tag{49}$$

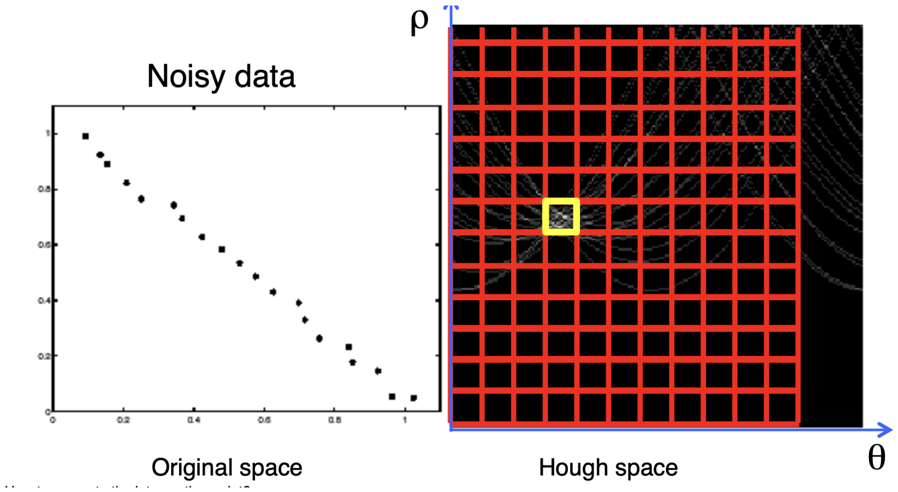
Figure 8: Polar representation of the parameter space.



The left half of Figure (8) shows that ρ is minimum distance from the origin to the line (bounded by the image size or maximum distance between any two points in the dataset), and θ is the angle between the x-axis and and normal vector of the line (bounded between 0 and π). We use the same Hough voting procedure as before, but now all possible lines in the Cartesian space going through a specific (x_i, y_i) corresond to sinusodial profile in the Hough space, as illustrated in the right half of of Figure (8).

In practice, noisy data points means that sinusoidal profiles in the Hough space that correspond to points on the same line in the image space, may not necessarily intersect at the same point in the Hough space. To sol ve this, we can increase the width w of the grid cells, which increases the tolerance to imperfect intersections, illustrated in Figure (9). While this can help deal with noisy data, it introduces yet another parameter that we need to tune. Small grid sizes may result in missed image space lines due to noise, while large grid sizes may merge different lines and reduce estimation accuracy since all ρ, θ within a cell are possible lines.

Figure 9: Increase grid cell size for Hough voting to deal with noise.



In addition, with uniform noise in the image space, there will be spurious peaks in the Hough space and no clear consensus of an appropiate model.