2.1 Least-Squares (LS) †

You are given a collection of data points (x_i, y_i), for $i = 1 \dots n$, and asked to fit a line to these data. The model of a line:

$$y = mx + b, \tag{2.1}$$

is parametrized by two parameters: the slope m and y-intercept b. Each data point (x_i, y_i) provides one constraint on these two parameters. A total of n such constraints provides an over-constrained system that can be expressed in matrix form as:

$$\begin{pmatrix} x_1 & 1 \\ x_2 & 1 \\ \vdots & \vdots \\ x_n & 1 \end{pmatrix} \begin{pmatrix} m \\ b \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} \tag{2.2}$$

$$X\mathbf{u} = \mathbf{y}. \tag{2.3}$$

This linear system of equations will have a solution only when the vector $\mathbf{y}$ is contained in the column space of matrix X: that is, when $\mathbf{y}$ can be expressed as a linear combination of the columns of the matrix X. From a geometric perspective, this will occur when all points (x_i, y_i) lie precisely on a line. If, however, the points deviate even slightly from a perfect line, then $\mathbf{y}$ will not be in the column space of X, and there will be no solution to the above linear system. It is in these situations that we seek a solution that minimizes some measure of goodness of fit of a line to the points.

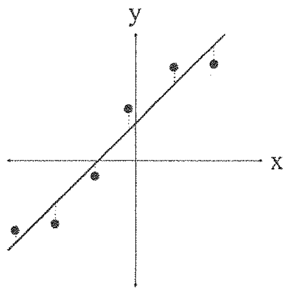


Figure 2.1 *Least-squares minimizes the vertical displacement of points from the line.*

Shown in Figure 2.1 are six data points that lie slightly off of a line. The fit line minimizes the overall vertical displacement of the points from the line. Minimizing this vertical distance, $mx + b - y$, lends itself to a particularly straight-forward optimization, termed least-squares. In matrix form, the vertical distance between points and a line with slope m and y-intercept b is given by:

$$\begin{pmatrix} x_1 & 1 \\ x_2 & 1 \\ \vdots & \vdots \\ x_n & 1 \end{pmatrix} \begin{pmatrix} m \\ b \end{pmatrix} - \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} \tag{2.4}$$

$$X\mathbf{u} - \mathbf{y}. \tag{2.5}$$

We seek to minimize the sum of these squared distances:

$$\sum_{i=1}^{n}(mx_i + b - y_i)^2 = (X\mathbf{u} - \mathbf{y})^T(X\mathbf{u} - \mathbf{y}). \qquad (2.6)$$

The minimization of this error is expressed as a quadratic error function in the parameters of the line:

$$E(\mathbf{u}) = \|X\mathbf{u} - \mathbf{y}\|^2, \qquad (2.7)$$

where $\|\cdot\|$ denotes vector norm. This quadratic error is minimized by differentiating:

$$\frac{dE}{d\mathbf{u}} = 2X^T(X\mathbf{u} - \mathbf{y}), \qquad (2.8)$$

setting equal to zero and solving for $\mathbf{u}$:

$$\begin{aligned} 2X^T(X\mathbf{u} - \mathbf{y}) &= 0 & (2.9)\\ X^TX\mathbf{u} &= X^T\mathbf{y} & (2.10)\\ \mathbf{u} &= (X^TX)^{-1}X^T\mathbf{y}, & (2.11) \end{aligned}$$

yielding the least-squares estimation of the line parameters. The matrix X^TX will be singular and hence not invertible if the rows of the matrix X are linearly dependent. Geometrically, this corresponds to one of two situations: (1) the n points are identical in which case there is no unique solution to fitting a line to a single point; or (2) the n points lie on a vertical line in which case $m = \infty$.

This basic framework is, of course, applicable to estimating any model that is linear in their unknown parameters. For example, a parabola $y = ax^2 + bx + c$ can be fit to n points by first constructing the following linear system:

$$\begin{pmatrix} x_1^2 & x_1 & 1 \\ x_2^2 & x_2 & 1 \\ \vdots & \vdots & \vdots \\ x_n^2 & x_n & 1 \end{pmatrix} \begin{pmatrix} a \\ b \\ c \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} \qquad (2.12)$$

$$X\mathbf{u} = \mathbf{y}, \qquad (2.13)$$

and then solving using the least-squares solution in Equation (2.11). Similarly, the matrix formulation of a plane $z = ax + by + c$ is:

$$\begin{pmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ \vdots & \vdots & \vdots \\ x_n & y_n & 1 \end{pmatrix} \begin{pmatrix} a \\ b \\ c \end{pmatrix} = \begin{pmatrix} z_1 \\ z_2 \\ \vdots \\ z_n \end{pmatrix} \qquad (2.14)$$

$$X\mathbf{u} = \mathbf{z}, \qquad (2.15)$$

and the least-squares solution is $\mathbf{u} = (X^TX)^{-1}X^T\mathbf{z}$.

Example 2.1 Where possible, write out the least-squares system for the following (where a, b, and c are the model parameters):

1. $y = a\log(x) + b$
2. $z = ax^y + b$
3. $y = x^a$

Weighted least-squares allows for a non-uniform treatment of the contribution of each individual point to the overall error. The error function of Equation (2.7) takes the form:

$$E(\mathbf{u}) = W\|X\mathbf{u} - \mathbf{y}\|^2, \tag{2.16}$$

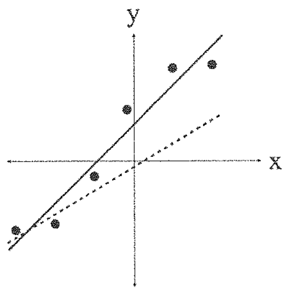


Figure 2.2
Least-squares (solid) and weighted least-squares (dashed).

where W is a diagonal $n \times n$ weighting matrix with diagonal elements w_i corresponding to the weight associated with the i^{th} point. A larger weight places more emphasis on minimizing that point's deviation from the model. This error function is minimized by differentiating:

$$\frac{dE}{d\mathbf{u}} = 2X^TW(X\mathbf{u} - \mathbf{y}), \tag{2.17}$$

setting equal to zero and solving for $\mathbf{u}$ to yield the weighted least-squares solution:

$$\begin{aligned} 2X^TW(X\mathbf{u} - \mathbf{y}) &= 0 && (2.18)\\ X^TWX\mathbf{u} &= X^TW\mathbf{y} && (2.19)\\ \mathbf{u} &= (X^TWX)^{-1}X^TW\mathbf{y}. && (2.20) \end{aligned}$$

Notice that if W is the identity matrix, then this solution reverts back to the least-squares solution of Equation (2.7).

Shown in Figure 2.2 are six data points fit with a line using least-squares (solid line) and weighted least squares (dashed line) where the two bottom left most points were weighted disproportionately relative to the other points. Notice how the line minimizes the error for these points at the price of significantly higher error for the remaining points.

2.2 Expectation Maximization (EM) †

The Expectation/Maximization (EM) algorithm simultaneously segments and fits data generated from multiple parametric models. For example, shown in Figure 2.3 are a collection of data points (x_i, y_i) generated from one of two linear models of the form:

$$\begin{aligned} y(i) &= a_1 x(i) + b_1 + n_1(i) && (2.21) \\ y(i) &= a_2 x(i) + b_2 + n_2(i), && (2.22) \end{aligned}$$

where the model parameters are a_1, b_1 and a_2, b_2, and the system is modeled with additive noise $n_1(i)$ and $n_2(i)$.

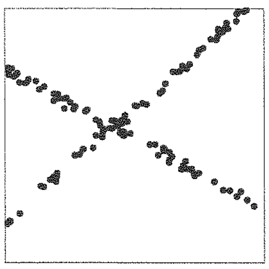


Figure 2.3 Data from two models

If we are told the model parameters, then determining which data point was generated by which model would be a simple matter of choosing, for each data point i, the model k that minimizes the error between the data and the model prediction:

$$r_k(i) = |a_k x(i) + b_k - y(i))|, \quad (2.23)$$

for $k = 1, 2$ in our example. On the other hand, if we are told which data points were generated by which model, then estimating the model parameters reduces to solving, for each model k, an over-constrained set of linear equations:

$$\begin{pmatrix} x_k(1) & 1 \\ x_k(2) & 1 \\ \vdots & \vdots \\ x_k(n) & 1 \end{pmatrix} \begin{pmatrix} a_k \\ b_k \end{pmatrix} = \begin{pmatrix} y_k(1) \\ y_k(2) \\ \vdots \\ y_k(n) \end{pmatrix}, \quad (2.24)$$

where the $x_k(i)$ and $y_k(i)$ all belong to model k. In either case, knowing one piece of information (the model assignment or parameters) makes determining the other relatively easy. But, lacking either piece of information makes this a considerably more difficult estimation problem. The EM algorithm is an iterative two step algorithm that estimates both the model assignment and parameters.

The "E-step" of EM assumes that the model parameters are known (initially, the model parameters can be assigned random values) and calculates the likelihood of each data point belonging to each model. In so doing the model assignment is made in a "soft" probabilistic fashion. That is, each data point is not explicitly assigned a single model, instead each data point i is assigned a probability of it belonging to each model k. For each model the residual error is first computed as:

$$r_k(i) = a_k x(i) + b_k - y(i) \quad (2.25)$$

from which the likelihoods are calculated. We ask, what is the likelihood of point i belonging to model k given the residual error. For our two model example:

$$\begin{aligned} P(a_k, b_k | r_k(i)) &= \frac{P(r_k(i)|a_k, b_k)P(a_k, b_k)}{P(r_k(i))} \\ &= \frac{P(r_k(i)|a_k, b_k)}{P(r_1(i)|a_k, b_k) + P(r_2(i)|a_k, b_k)}, \end{aligned} \tag{2.26}$$

for $k = 1, 2$.

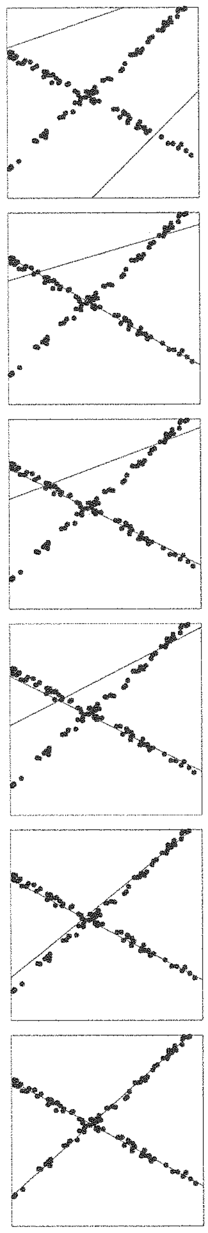


Figure 2.4 Six iterations of EM.

The expansion of the conditional probability is from Bayes rule: $P(B|A_k) = \frac{P(A_k|B)P(B)}{\sum_l P(A_l|B)P(B)}$. If we assume a Gaussian probability distribution, then the likelihood takes the form:

$$w_k(i) = P(a_k, b_k | r_k(i)) = \frac{e^{-r_k^2(i)/\sigma}}{e^{-r_1^2(i)/\sigma} + e^{-r_2^2(i)/\sigma}}, \tag{2.27}$$

where, σ is proportional to the amount of noise in the data.

The "M-step" of EM takes the likelihood of each data point belonging to each model, and re-estimates the model parameters using weighted least-squares. That is, the following weighted error function on the model parameters is minimized:

$$E_k(a_k, b_k) = \sum_{i=1}^{n} w_k(i)[a_k x(i) + b_k - y(i)]^2. \tag{2.28}$$

The intuition here is that each data point contributes to the estimation of each model's parameters in proportion to the belief that it belongs to that particular model. This quadratic error function is minimized by computing the partial derivatives with respect to the model parameters, setting the result equal to zero and solving for the model parameters. Differentiating:

$$\begin{aligned} \frac{\partial E_k(a_k, b_k)}{\partial a_k} &= \sum_{i=1}^{n} 2 w_k(i) x(i)[a_k x(i) + b_k - y(i)] \\ \frac{\partial E_k(a_k, b_k)}{\partial b_k} &= \sum_{i=1}^{n} 2 w_k(i)[a_k x(i) + b_k - y(i)], \end{aligned} \tag{2.29}$$

setting both equal to zero yields the following set of linear equations:

$$a_k \sum_{i=1}^{n} w_k(i) x(i)^2 + b_k \sum_{i=1}^{n} w_k(i) x(i) = \sum_{i=1}^{n} w_k(i) x(i) y(i) \tag{2.30}$$

$$a_k \sum_{i=1}^{n} w_k(i) x(i) + b_k \sum_{i=1}^{n} w_k(i) = \sum_{i=1}^{n} w_k(i) y(i). \tag{2.31}$$

Rewriting in matrix form:

$$\begin{aligned}
\begin{pmatrix} \sum_{i=1}^n w_k(i)x(i)^2 & \sum_{i=1}^n w_k(i)x(i) \\ \sum_{i=1}^n w_k(i)x(i) & \sum_{i=1}^n w_k(i) \end{pmatrix} \begin{pmatrix} a_k \\ b_k \end{pmatrix} &= \begin{pmatrix} \sum_{i=1}^n w_k(i)x(i)y(i) \\ \sum_{i=1}^n w(i)y(i) \end{pmatrix} \\
A\mathbf{x_k} &= \mathbf{b} \\
\mathbf{x_k} &= A^{-1}\mathbf{b}, \qquad (2.32)
\end{aligned}$$

yields a weighted least squares solution for the model parameters.

Example 2.2 Show that the solution in Equation (2.32) is identical to solving the set of linear equations in Equation (2.24) using weighted least-squares.

The EM algorithm iteratively executes the "E" and "M" step, repeatedly estimating and refining the model assignments and parameters. Shown in Figure 2.4 are several iterations of EM applied to fitting data generated from two linear models. Initially, the model parameters are randomly assigned, and after six iterations, the algorithm converges to a solution.

The EM algorithm can be sensitive to the value of σ used. It is often recommended that with a reasonable starting value, the value of σ can be updated on each EM iteration as follows:

$$\sigma = \frac{\sum_{i=1}^n w(i)r^2(i)}{\sum_{i=1}^n w(i)}. \qquad (2.33)$$

Most digital cameras capture color images using a single sensor in conjunction with an array of color filters. As a result, only one third of the samples in a color image are captured by the camera, the other two thirds are interpolated. This interpolation introduces specific correlations between the samples of a color image. When creating a digital forgery these correlations may be destroyed or altered. We describe the form of these correlations, and propose a method that quantifies and detects them in any portion of an image.

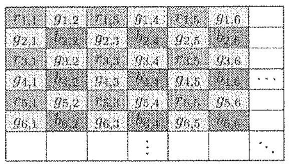

$r_{1,1}$ $g_{1,2}$ $r_{1,3}$ $g_{1,4}$ $r_{1,5}$ $g_{1,6}$
$g_{2,1}$ $b_{2,2}$ $g_{2,3}$ $b_{2,4}$ $g_{2,5}$ $b_{2,6}$
$r_{3,1}$ $g_{3,2}$ $r_{3,3}$ $g_{3,4}$ $r_{3,5}$ $g_{3,6}$
$g_{4,1}$ $b_{4,2}$ $g_{4,3}$ $b_{4,4}$ $g_{4,5}$ $b_{4,6}$ $\cdots$
$r_{5,1}$ $g_{5,2}$ $r_{5,3}$ $g_{5,4}$ $r_{5,5}$ $g_{5,6}$
$g_{6,1}$ $b_{6,2}$ $g_{6,3}$ $b_{6,4}$ $g_{6,5}$ $b_{6,6}$
$\vdots$ $\ddots$

Figure 2.5 Bayer array

A digital color image consists of three channels containing samples from different bands of the color spectrum, e.g., red, green, and blue. Most digital cameras, however, are equipped with a single CCD or CMOS sensor, and capture color images using a color filter array (CFA). The most frequently used CFA, the Bayer array, employs three color filters: red, green, and blue. The red and blue pixels are sampled on rectilinear lattices, while the green pixels are sampled on a quincunx lattice, Figure 2.5. Since only a single color sample is recorded at each pixel location, the other two color samples must be estimated from the neighboring samples in order to obtain a three-channel color image. Let $S(x,y)$ denote the CFA image in Figure 2.5, and $\tilde{R}(x,y)$, $\tilde{G}(x,y)$, $\tilde{B}(x,y)$ denote the red, green, and blue channels constructed from $S(x,y)$ as follows:

$$\begin{aligned} \tilde{R}(x,y) &= S(x,y) && \text{if } S(x,y) = r_{x,y} && (2.34)\\ &= 0 && \text{otherwise} \\ \tilde{G}(x,y) &= S(x,y) && \text{if } S(x,y) = g_{x,y} && (2.35)\\ &= 0 && \text{otherwise} \\ \tilde{B}(x,y) &= S(x,y) && \text{if } B(x,y) = b_{x,y} && (2.36)\\ &= 0 && \text{otherwise} \end{aligned}$$

where (x,y) span an integer lattice. A complete color image, with channels $R(x,y)$, $G(x,y)$, and $B(x,y)$ needs to be estimated. These channels take on the non-zero values of $\tilde{R}(x,y)$, $\tilde{G}(x,y)$, and $\tilde{B}(x,y)$, and replace the zeros with estimates from neighboring samples.

The estimation of the missing color samples is referred to as CFA interpolation or demosaicking. CFA interpolation has been extensively studied and many methods have been proposed. The simplest methods for demosaicking are kernel-based interpolation methods that act on each channel independently. These methods can be efficiently implemented as linear filtering operations on each color channel:

$$
\begin{aligned}
R(x,y) &= \sum_{u,v=-N}^{N} h_r(u,v)\tilde{R}(x-u,y-v) && (2.37)\\
G(x,y) &= \sum_{u,v=-N}^{N} h_g(u,v)\tilde{G}(x-u,y-v) && (2.38)\\
B(x,y) &= \sum_{u,v=-N}^{N} h_b(u,v)\tilde{B}(x-u,y-v), && (2.39)
\end{aligned}
$$

where $\tilde{R}(\cdot)$, $\tilde{G}(\cdot)$, $\tilde{B}(\cdot)$ are defined in Equations (2.34)-(2.36), and $h_r(\cdot)$, $h_g(\cdot)$, $h_b(\cdot)$ are linear filters of size $(2N+1) \times (2N+1)$. Different forms of interpolation (nearest neighbor, bilinear, bicubic, etc.) differ in the form of the interpolation filter used. For the Bayer array, the bilinear filter for the red and blue channels are separable. The 1-D filter is:

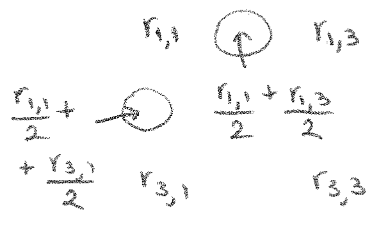
$$
h_l = \begin{pmatrix} 1/2 & 1 & 1/2 \end{pmatrix}. \qquad (2.40)
$$

The 2-D filter is the outer product between h_l and itself.

There are many other CFA interpolation algorithms including smooth hue transition, median filter, gradient-based, adaptive color plane, and threshold-based variable number of gradients. Regardless of their specific implementations, each CFA interpolation algorithm introduces specific statistical correlations between a subset of pixels in each color channel.

Since the color filters in a CFA are typically arranged in a periodic pattern, these correlations are periodic. Consider, for example, the red channel, $R(x,y)$, that has been sampled on a Bayer array, Figure 2.5, then CFA interpolated using bilinear interpolation. In this case, the red samples in the odd rows and even columns are the average of their closest horizontal neighbors, the red samples in the even rows and odd columns are the average of their closest vertical neighbors, and the red samples in the even rows and columns are the average of their closest diagonal neighbors:

$$
\begin{aligned}
R(2x+1,2y) &= \frac{R(2x+1,2y-1)}{2} + \frac{R(2x+1,2y+1)}{2}\\
R(2x,2y+1) &= \frac{R(2x-1,2y+1)}{2} + \frac{R(2x+1,2y+1)}{2}\\
R(2x,2y) &= \frac{R(2x-1,2y-1)}{4} + \frac{R(2x-1,2y+1)}{4}\\
&+ \frac{R(2x+1,2y-1)}{4} + \frac{R(2x+1,2y+1)}{4}.
\end{aligned}
$$


$r_{1,1}$
$r_{1,3}$
$\frac{r_{1,1}}{2} + \frac{r_{3,1}}{2}$
$\frac{r_{1,1}}{2} + \frac{r_{1,3}}{2}$
$r_{3,1}$
$r_{3,3}$

Note that in this simple case, the estimated samples are perfectly correlated to their neighbors. As such, a CFA interpolated image can be detected (in the absence of noise) by noticing, for example, that every other sample in every other row or column is perfectly correlated to its neighbors. At the same time, the non-interpolated samples are less likely to be correlated in precisely the same manner. Furthermore, it is likely that tampering will destroy these correlations, or that the splicing together of two images from different cameras will create inconsistent correlations across the composite image. As such, the lack of correlations produced by CFA interpolation can be used to expose it as a forgery.

We begin by assuming a simple linear model for the periodic correlations introduced by CFA interpolation. That is, each interpolated pixel is correlated to a weighted sum of pixels in a small neighborhood centered about itself. While perhaps overly simplistic when compared to the highly non-linear nature of most CFA interpolation algorithms, this simple model is both easy to parametrize and can reasonably approximate each of the CFA interpolation algorithms described above. Note that most CFA algorithms estimate a missing color sample from neighboring samples in all three color channels. For simplicity, however, we ignore these inter-channel correlations and treat each color channel independently.

If the specific form of the correlations is known (i.e., the parameters of the linear model), then it would be straightforward to determine which samples are correlated to their neighbors. On the other hand, if it was known which samples are correlated to their neighbors, the specific form of the correlations could be easily determined. In practice, of course, neither are known. To simultaneously estimate both we employ the expectation/maximization (EM) algorithm.

Recall that the expectation-maximization algorithm (EM) is a two-step iterative algorithm: (1) in the E-step the probability of each sample belonging to each model is estimated; and (2) in the M-step the specific form of the correlations between samples is estimated.

We first formulate the estimation for a 1-D signal. Let $f(x)$ denote a color channel (red, green, or blue) of a CFA interpolated signal. We begin by assuming that each sample in $f(x)$ belongs to one of two models: (1) M_1 if the sample is linearly correlated to its neighbors, satisfying:

$$f(x) \quad = \quad \sum_{u=-N}^{N} \alpha_u f(x+u), \tag{2.41}$$

where the model parameters are given by the linear coefficients $\alpha = \{\alpha_u | -N \leq u \leq N\}$ (N is an integer and $\alpha_0 = 0$) ; or (2) M_2 if the sample is not correlated to its neighbors, i.e., is generated by an "outlier process".

In the E-step, the residual error for the first model is:

$$r_1(x) \quad = \quad f(x) - \sum_{u=-N}^{N} \alpha_u f(x+u) \tag{2.42}$$

Assuming that the probability of observing a sampled generated by model M_1 follows a Gaussian, the likelihood then takes the form:

$$w_1(x) = Pr(\alpha \mid r_1(x)) = \frac{e^{-r_1^2(x)/\sigma}}{e^{-r_1^2(x)/\sigma} + 1/\delta}, \tag{2.43}$$

where a uniform distribution is assumed for the probability of observing a sample generated by the outlier model, M_2.

In the M-step, a new estimate of α is computed using weighted least squares to minimize the following quadratic error function:

$$E(\alpha) = \sum_x w_1(x) \left(f(x) - \sum_{u=-N}^{N} \alpha_u f(x+u) \right)^2 . \tag{2.44}$$

This error function is minimized by computing the gradient with respect to α, setting this gradient equal to zero, and solving for α yielding:

$$\alpha \quad = \quad (F^T W F)^{-1} F^T W \mathbf{f}. \tag{2.45}$$

where, if $N = 2$, for example, the vector $\mathbf{f}$ is:

$$\mathbf{f} = \left(f(3) \quad f(4) \quad f(5) \quad \ldots \right)^T, \tag{2.46}$$

and where the matrix F is:

$$F = \begin{pmatrix} f(1) & f(2) & f(4) & f(5) \\ f(2) & f(3) & f(5) & f(6) \\ f(3) & f(4) & f(6) & f(7) \\ \vdots & \vdots & \vdots & \vdots \end{pmatrix}, \tag{2.47}$$

and W is a diagonal weighting matrix with $w_1(x)$ along the diagonal. The E-step and the M-step are iteratively executed until a stable estimate of α is achieved.

In 2-D, let $f(x, y)$ denote a color channel (red, green, or blue) of a CFA interpolated image. We again assume that each sample in $f(x, y)$ belongs to one of two models: (1) M_1 if the sample is linearly correlated to its neighbors, satisfying:

$$f(x,y) \quad = \quad \sum_{u,v=-N}^{N} \alpha_{u,v} f(x+u, y+v), \tag{2.48}$$

where the model parameters are given by the linear coefficients $\alpha = \{\alpha_{u,v} | -N \leq u, v \leq N\}$ (N is an integer and $\alpha_{0,0} = 0$) ; or (2) M_2 if the sample is not correlated to its neighbors, i.e., is generated by an "outlier process".

In the E-step, the residual error for the first model is:

$$r_1(x,y) \quad = \quad f(x,y) - \sum_{u,v=-N}^{N} \alpha_{u,v} f(x+u, y+v) \tag{2.49}$$

Assuming that the probability of observing a sampled generated by model M_1 follows a Gaussian, the likelihood then takes the form:

$$w_1(x,y) = Pr(\alpha \mid r_1(x,y)) = \frac{e^{-r_1^2(x,y)/\sigma}}{e^{-r_1^2(x,y)/\sigma} + 1/\delta}, \tag{2.50}$$

where a uniform distribution is assumed for the probability of observing a sample generated by the outlier model, M_2.

In the M-step, a new estimate of α is computed using weighted least squares to minimize the following quadratic error function:

$$E(\alpha) = \sum_{x,y} w_1(x,y) \left(f(x,y) - \sum_{u,v=-N}^{N} \alpha_{u,v} f(x+u, y+v) \right)^2 \tag{2.51}$$

This error function is minimized by computing the gradient with respect to α, setting this gradient equal to zero, and solving the resulting linear system of equations. Setting equal to zero the partial derivative with respect to one of the coefficients, $\partial E / \partial \alpha_{s,t} = 0$, yields:

$$\begin{aligned} \sum_{x,y} w_1(x,y) f(x+s, y+t) \sum_{u,v=-N}^{N} \alpha_{u,v} f(x+u, y+v) \quad &= \\ \sum_{x,y} w_1(x,y) f(x+s, y+t) f(x,y). \end{aligned} \tag{2.52}$$

Re-ordering the terms on the left-hand side yields:

$$\sum_{u,v=-N}^{N} \alpha_{u,v} \left(\sum_{x,y} w_1(x,y) f(x+s,y+t) f(x+u,y+v) \right) = \sum_{x,y} w_1(x,y) f(x+s,y+t) f(x,y). \qquad (2.53)$$

This process is repeated for each component, $\alpha_{s,t}$ to yield a system of linear equations that can be solved using standard techniques. The E-step and the M-step are iteratively executed until a stable estimate of α is achieved.

Shown in the upper three panels of Figure 2.6, from top to bottom, is an original CFA interpolated image, the resulting CFA analysis displayed as a probability map, and the magnitude of the Fourier transform of the probability map. The results of the CFA analysis is displayed as the probability of each pixel belonging to model M_1 (i.e., correlated to their neighbors) in which a white pixel denotes a high probability and a black pixel denotes a low probability. The peaks in the Fourier transform correspond to the periodicity in the probability map which reveal the expected CFA correlations for an authentic image.

Shown in the lower three panels of Figure 2.6, from top to bottom, is an original CFA interpolated image, the resulting CFA analysis displayed as a probability map, and the magnitude of the Fourier transform of the probability map. In this case, the image does not contain the expected CFA correlations as can be seen by the lack of periodicity in the probability map and peaks in the Fourier transform.

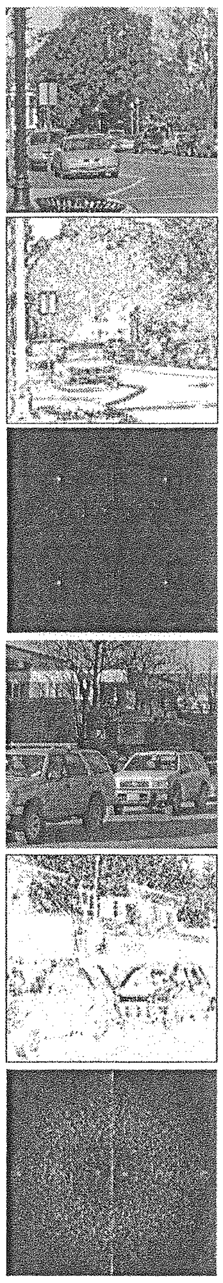


Figure 2.6 A CFA and non-CFA image and the results of CFA analysis.