提供：Optical Solutions 株式会社オプティカルソリューションズ
〒101-0032
東京都千代田区岩本町 2-15-8（MAS 三田ビル 3 階）
TEL：03-5833-1332　　FAX：03-3865-3318
http://www.osc-japan.com/

光学設計ノーツ　５０（ver.1.0）

デジタル画像の性質について

株式会社タイコ
牛山善太

今回は、これまでとがらりとテーマを変えて（本当は繋がっていますが）、所謂デジタル画像というものの性質について考えさせていただきたい。デジタル画像とは ccd や cmos 等の離散化された画素を持つ撮像素子を用いて、そこから得られる離散化された情報の塊としての画像を表すものとする。

１． Digital to Digital の場合の簡単な考察

ここでは最も基本的なものとして、非常に簡単なシステムを考える。Digital to Digital であって、離散化された原稿、被写体を、離散化された像面に結像させる場合を考える。原稿の画素を 2 個、受光面の画素も 2 個とミニマムの場合を想定する。ここで、光学系により原稿の画像が結像しているとする。その場合、レンズによって、ある任意の画素 j から出てレンズを透過する 1 というエネルギーが、像面上のどこかの画素 k にどのくらいのエネルギーとして分散して到達しているかという分布を

$$h_{jk} \qquad (1)$$

として表す。（この場合必ずしもエネルギーはすべて画素に収まる必要はない。）光学設計では所謂 PSF（point spread function）と言われるものに近く、ここでは point ではなく、pixel から出たエネルギーを扱っているので、仮に PXSF（pixel spread function）とでも呼ぼう（図 1）。

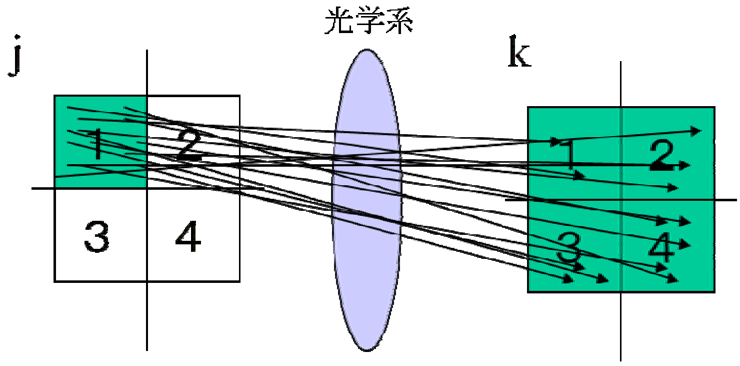

図1　PXSF　　４ ×４画素の場合

さらに原稿の各画素から射出してレンズを通過するエネルギーを fj、像面各画素が受け取るエネルギーを gk とする。この場合は j 、k ともに、1,2 であるので

$$h_{1,1}f_1 + h_{2,1}f_2 = g_1$$

$$h_{1,2}f_1 + h_{2,2}f_2 = g_2 \qquad (2)$$

という簡単な連立方程式が成立する。原稿上の発光ネエルギー分布が不明であっても、PXSF が分かっていて、画像の強度分布が分かっていれば、元画像の発光分布は計算できることになる。具体的に数値を入れて計算してみよう。

PXSF について考えれば、光源画素 1 からのエネルギー1 の光は光学系を介して像面画素 1,2 にそれぞれ、0.2、0.3 というエネルギーとして到達し、光源画素 2 からはそれぞれ、0.4、0.1 到達しているとする。像面各画素には 4、3 というエネルギーが到達しているとしよう。そうすると、（２）式から、

$$0.2\times f_1 + 0.4\times f_2 = 4$$

$$0.3\times f_1 + 0.1\times f_2 = 3 \qquad (３)$$

と出来る。これを解くのはいとも簡単で、

$$f_1 = 8$$

$$f_2 = 6 \qquad (４)$$

どなる。

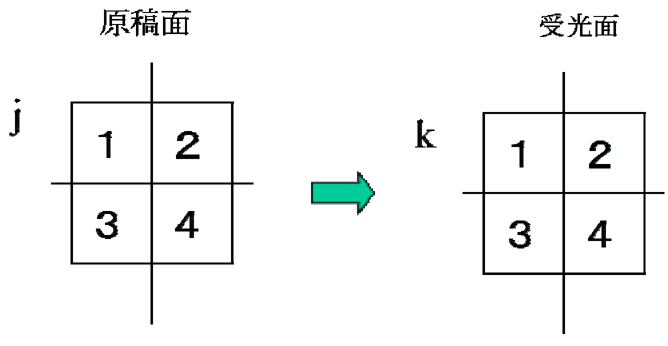

図2　　４×４画素

仮に画素が原稿面、受光面ともに 4×4 になったとしても（図 2）（2）式は以下のうに変数、式ともに数が増えて、

$$h_{1,1}f_1+h_{2,1}f_2+h_{3,1}f_3+h_{4,1}f_4=g_1$$

$$h_{1,2}f_1+h_{2,2}f_2+h_{3,2}f_3+h_{4,2}f_4=g_2$$

$$h_{1,3}f_1+h_{2,3}f_2+h_{3,3}f_3+h_{4,3}f_4=g_3 \quad (5)$$

$$h_{1,4}f_1+h_{2,4}f_2+h_{3,4}f_3+h_{4,4}f_4=g_4$$

と、なるだけである。ここでも未知数と式の数は同じなので解ける。これは画素数が増えて行ってもまったく同じことで、10,000×10,000 画素でも未知数と方程式の数は同じなので完全に解が得られる。より一般的に表現すれば、

$$\begin{pmatrix} g_1 \\ g_2 \\ \vdots \\ g_n \end{pmatrix} = \begin{pmatrix} h_{1,1} & h_{2,1} & \cdots & h_{n,1} \\ h_{1,2} & h_{2,2} & \cdots & h_{n,2} \\ \vdots & \vdots & \vdots & \vdots \\ h_{1,n} & h_{2,n} & \cdots & h_{n,n} \end{pmatrix} \begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ f_n \end{pmatrix} \quad (6)$$

である。因みに光学系の収差がない場合には、

$$\begin{pmatrix} g_1 \\ g_2 \\ \vdots \\ g_n \end{pmatrix} = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 \end{pmatrix} \begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ f_n \end{pmatrix} \quad (7)$$

となる。

$$\mathbf{g}=\left(g_1 \cdots g_n\right)^T \quad (8)$$

$$\mathbf{f}=\left(f_1 \cdots f_n\right)^T \quad (9)$$

として行列を A であらわせば、（6）式は

$$\mathbf{g}=\mathbf{Af} \quad (10)$$

と表せて解は

$$\mathbf{f} = \mathbf{A}^{-1}\mathbf{g} \tag{11}$$

である。

　　つまり、光学系の結像性能が得られていれば、その光学系の性能に関わらず、既知画像から元画像は、レンズから観察する方向からの見え方については、完全に復元することができる。その性質が digital to digital 画像には本来含まれている。これは線形代数的には実に当たり前の事実であるが、レンズ設計的には結構インパクトがある。収差補正はどうでもいいのだ。ただし、以下の特殊性がこれまでの検討には存在する。

１．　原稿をデジタル化している。一般的にはアナログである。自然を升目に区切って考えることにより、そこの区分内のすべての無限に近い数存在する点光源から出てレンズを透過する光の本当の分布と、画素内を均一化して考えた PXSF との分布誤差が連立方程式を解けなくする。

２．　エネルギー量のデジタル化（量子化）は考えていない。上式の画素ごとのエネルギー等については離散化することの誤差によって、連立方程式が解けなくなる。

大きな問題は以上の２つであろう。確かに測定誤差とかノイズの問題、あるいは計算量については現実的には重要な問題となろうが、遠い未来には解決できるはずで、ここでは本質的ではない。

　上記二つの大問題も、座標、量ともにデジタル化のピッチを細かくしていけば当然軽減できるが、いずれにしても連立方程式のシンプルで厳密な解法には大きな影響を与える。そこで、必要となるのが、これら大きな意味でのノイズを考慮して解に接近できるような、最小二乗法的な、あるいは統計的な、誤差を伴った測定値等を扱うための考え方である。

　例えば、ある f の値が一応得られているとき、（10）式の両辺の差を長さで表し、f そのものの長さ（このでの長さとは n 次元ベクトルの長さと考えて戴いて良い）を考えるとき、これらの量の２乗の和、

$$\left\|\mathbf{Af} - \mathbf{g}\right\|^2 + \alpha\left\|\mathbf{f}\right\|^2 \tag{12}$$

はいかなる量を表すであろうか？（α は正の定数とする。）当然得られた f によって第１項はなるべく小さい値となることが望まれる。また、その時のノイズを拾った方程式を解くことによって、f が第１項を０と成しても、画像として有り得ない様な突飛な値をとってしまっても困る。つまり第２項が非常な大きな値になっても不味い。言い換えれば、第１項が小さく、２項も小さければかなり尤もらしい結果が出ているとも考えられる。元画像のコントラスト範囲が大きい時などにおいては、かなり適当な感じもするが、α の値をいかにとるかによって、第２項の大きさをある程度まで許したりして、コントロールを効かせられるところ

が重要なところである。この様な量を用いてｆの落としどころを探ろうと言うのが最小二乗法の考え方である。

実は上式の値が最小になる解は、Ｉを単位行列、右肩Ｔが転置行列を表すとして、

$$\mathbf{f} = \left(\mathbf{A}^T\mathbf{A} + \alpha\mathbf{I}\right)^{-1}\mathbf{A}^T\mathbf{g} \qquad (13)$$

となる。これをチコノフ（Tikhonov）の正則化と呼ぶ。

参考文献

1) http://en.wikipedia.org/wiki/Tikhonov_regularization