2007－CG－126 (11)
2007／2／20

実時間インタラクティブColor2Grayシステム

呂 冬† 山口 泰††

カラー画像をモノクロプリンタへ出力する際にはカラー画像をグレースケール画像に変換する必要がある．通常，このようなグレースケール画像への変換においてはカラー画像の明度値を用いる．この時，色相と彩度の情報が失われるが，与えられたカラー画像の明度のダイナミックレンジが狭い場合には，カラー画像の重要な特徴が損なわれることがある．図1はMonetの「印象・日の出」をグレースケール画像にしたものであるが，明度のみではほとんど理解できない画像になっている．このように明度の差が小さく，色度が重要な役割をになう画像の場合には，視覚の特徴を保存するために，新しい考え方が必要となる．Goochら[4]のアルゴリズムは色度に伴う視覚特徴を反映できるモノクロ画像を作れるが，計算コストが高いため実用性に問題があった．そこで本研究では，色量子化を用いることにより画像をモノクロ化する時，GUIでパラメータを調整しながら画像の視覚特徴の変化をユーザに見せられるシステムを提案する．さらに，実験結果に基づいて手法の有効性などを議論する．

Interactive Color2Gray System

RO TOU† and YASUSHI YAMAGUCHI††

Visually important image features often disappear when color images are converted to grayscale. Color2Gray algorithm reduces such losses by attempting to preserve the salient features of the color image. Color2Gray algorithm is not practical use because of its high cost calculation. This paper aims at accelerating the Color2Gray algorithm in order to allow users to see the visual effects interactively while changing the control parameters.

1. はじめに

デジタル画像を光の記録と考えれば，モノクロ画像はフラットスペクトルを仮定した際の光の強さの記録と言える．しかし，人間はモノクロ画像の中に，視覚の特徴が保存されることを望んでいる．すなわち，単に光の強さを保存するだけではなく，画像の重要な特徴を表す色度の情報を反映できることが望ましい．本節では，カラー画像からモノクロ画像への変換における視覚特徴に関する研究を紹介する．

Photoshop[1][2]では固有の非線型な方法を使って，RGBの3成分から，明度への変換を行っている．この方法では，同じ明度の値を持っているピクセルに対して，モノクロ画像に変換する時，色相と彩度の違いを反映させることができない．

これに対して主成分分析(PCA)を用いることで，明度の軸を求める方法が考えられる．この場合PCAを行う空間の性質が重要となる．RGB空間と$L^*a^*b^*$空間で求めた主軸は大きく異なる可能性がある[4]．

Poisson Solvers[3]は，隣接する領域の勾配を用いて，結果画像の明度を制御する方法である．この方法の場合には図1のような画像ではうまく処理できるものの，離れた場所に同じような明度の領域が現れると，うまく計算ができない．

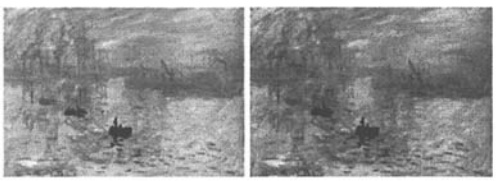

図1 入力のカラー画像(左)，Photoshopで作った結果(右)

Rascheら[5][6]は，色の視覚距離を保存するようにモノクロ化を施す方法を提案している．この方法ではピクセルの位置に関係なく，色の知覚的な差を保つことができた．また，色盲の人を対象として画像の色情報を2次元量に変換する手法を提案している．

Goochら[4]は2つの簡単なパラメータで，明度と色度の距離を調整でき，視覚的な特徴を保存したモノクロ画像を作る方法を提案した．しかしながら，この

† 東京大学大学院
ludong@graco.c.u-tokyo.ac.jp
†† 東京大学大学院
yama@graco.c.u-tokyo.ac.jp

アルゴリズムは相当時間がかかり，実用的な時間で計算できないために，パラメータの決定は勘に頼らざるを得なかった．

そこで本研究では，Gooch らの考え方に基づいて画像をモノクロ化する時，GUI でパラメータを調整しながら，画像の視覚特徴の変化をユーザに見せることを目指す．これにより，ユーザは自分の好きなモノクロ画像を作ることができるようになると考えられる．

2. Color2Gray

2.1 アルゴリズムの概要

Gooch らは，色相と彩度の情報を考慮した，カラー画像のモノクロ化方法 Color2Gray を提案した．同じ明度をもつ色数が多い画像に対して，直接明度チャンネルからモノクロ画像を作ると，同じモノクロの値になってしまう．つまり，色度に相当する視覚の差が失われる．そこで，色相と彩度の情報の影響を考慮して，元画像の明度データを増減する．この増減によって，元画像の明度の値が調整される．Color2Gray のアルゴリズムは次のような 3 つの段階に分けられる：

(1) RGB 色空間から CIE$L^*a^*b^*$ 色空間へ変換する
(2) 画像の各ピクセルペアについて，明度の差ならびに色相と彩度の差の情報から，視覚的嗜好を反映した明度の差 δ を計算する
(3) 結果画像における全てのピクセルペアの明度の差を最小二乗法により δ に近づける

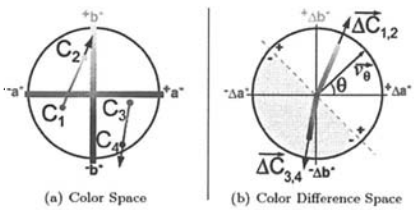

図 2 パラメータ θ の効果．(出展 [4])

δ_{ij} はピクセル i と j の間の色の違いを表す符号付きスカラー量で明度差と色度差とに基づいて計算される．δ_{ij} を計算するとき，2 つのパラメータ θ と α が用いられる．パラメータ θ と α はユーザの嗜好に合わせて，調整されるパラメータである．ここではまずパラメータ θ と α の意味を紹介し，次に δ_{ij} の計算について述べる．

2.1.1 パラメータ θ

θ は図 2 のように色度平面 (a^*b^* 平面) を 2 つの領域に分ける．この 2 つの領域では，色度差が元の明度値に対して，与える影響の符号が変わる．2 つピクセルの色度 (C_i, C_j) の間の色度差ベクトルを $\overrightarrow{\Delta C_{ij}}$ と表すものとする．角度 θ によって，色度差平面を正の領域と負の領域に分け，色度差の符号を $\overrightarrow{\Delta C_{ij}} \cdot \overrightarrow{v_\theta}$ の符号と同じものにする．ここで，$\overrightarrow{v_\theta}$ は θ 方向のベクトルであり，・は内積を表す．図 3 は θ の変化に伴う結果画像の差を示したものである．中心は元のカラー画像である．パラメータ θ によって，色度平面の分け方が変わるので，元の明度の値の増減のされ方が変化する．θ が 45 度の時，「45」という数字が見えるようになる．

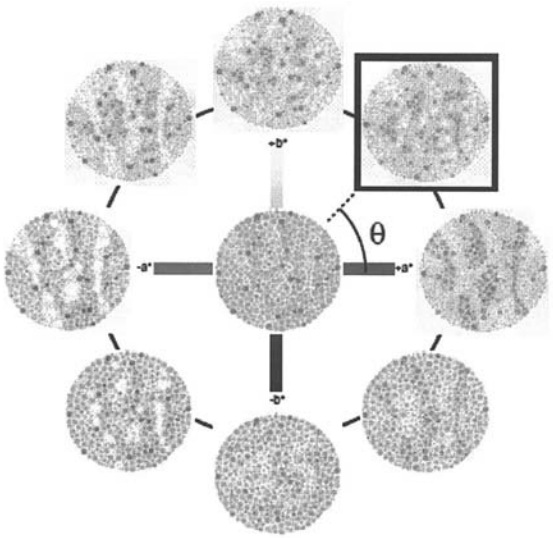

図 3 パラメータ θ による結果画像の変化．(出展 [4])

2.1.2 パラメータ α

$L^*a^*b^*$ 空間における a^* の範囲は [-500,500]，b^* の範囲は [-200,200]，L^* の範囲は [0,100] となる．つまり，色度の差 a^* や b^* に比べて明度の差 L^* は小さな値となる．そこで，色度差ベクトル $\overrightarrow{\Delta C_{ij}}$ の効果を抑えるために，パラメータ α を用いる．すなわち色度差に伴う明度の増減の量を $[-\alpha,\alpha]$ に抑える．図 4 はパラメータ α の効果を表している．パラメータ α を α=5,10,25 と変化させたもの．α が変化すると色度差による明度の増減の量が変化する．

図 4 パラメータ α による結果画像の変換．(出展 [4])

2.1.3 明度差 δ の計算

画像を RGB 空間から CIE$L^*a^*b^*$ 色空間に変換して得られる，i 番目のピクセルの明度値を L_i とする．ΔL_{ij} はピクセル i とピクセル j との明度差 $(L_i - L_j)$ を表す．同様にして，ΔA_{ij} と ΔB_{ij} はそれぞれピクセルペ

ア i, j の a^*, b^* の差を表すものとする．色度差ベクトル $\overrightarrow{\Delta C_{ij}}$ は $(\Delta A_{ij}, \Delta B_{ij})$ となる．

明度差 ΔL_{ij} と色度差 $\overrightarrow{\Delta C_{ij}}$ とを比較した上で，視覚の差 δ_{ij} を決める．$\overrightarrow{\Delta C_{ij}}$ のユークリッド距離を $||\overrightarrow{\Delta C_{ij}}||$ とする．この $||\overrightarrow{\Delta C_{ij}}||$ の効果の向きを定めるために，パラメータ θ を用いる．図2のように Δa^* 軸となす角度が θ となるベクトル $\overrightarrow{v_\theta}$ を作り，$\overrightarrow{\Delta C_{ij}} \cdot \overrightarrow{v_\theta}$ の符号を色度の符号とする．具体的な計算式は次のようになる．

$$\delta(\alpha,\theta)_{ij} = \begin{cases} \Delta L_{ij} & \text{if } |\Delta L_{ij}| > \text{crunch}(||\overrightarrow{\Delta C_{ij}}||) \\ \text{crunch}(||\overrightarrow{\Delta C_{ij}}||) & \text{if } ||\overrightarrow{\Delta C_{ij}}|| \cdot \overrightarrow{v_\theta} \geq 0 \\ \text{crunch}(-||\overrightarrow{\Delta C_{ij}}||) & \text{otherwise} \end{cases}$$

ただし，$\text{crunch}(x) = \alpha \tanh(x/\alpha)$，$\overrightarrow{v_\theta} = (\cos\theta, \sin\theta)$

視覚の特徴を保存したモノクロ画像では，結果画像の全てのピクセル間の明度差を δ_{ij} に近づけたい．このようにするために，次式を最適化することで，画像全体のピクセルの明度を定めるものとする．

$$f(\boldsymbol{g}) = \sum_{(i,j)\in\mathscr{P}} ((g_i - g_j) - \delta_{ij})^2 \quad (1)$$

$\mathscr{P}$ はすべてのピクセルのペアを表す．g_i と g_j は元のピクセルペア明度を表し，$\boldsymbol{g}$ はすべてピクセルの明度ベクトルである．

2.1.4 最小二乗法の解法

[4] では，全てのピクセルピア $\mathscr{P}$ に対して，式 (1) を最小化するために，「CompleteK」という方法を使っている．以下にその概要を示す．

式 (1) に 1/2 をかけると，次のようになる．

$$\frac{1}{2} f(\boldsymbol{g}) = \frac{1}{2} \sum_{(i,j)\in\mathscr{P}} ((g_i - g_j) - \delta_{ij})^2 \quad (2)$$

ここで，式 (2) は行列とベクトルによって，次式のように表現される．

$$\frac{1}{2}(A\boldsymbol{g} - \boldsymbol{b})^T (A\boldsymbol{g} - \boldsymbol{b})$$

(2) 式を展開すると

$$\frac{1}{2}\boldsymbol{g}^T A^T A \boldsymbol{g} - (A^T \boldsymbol{b})^T \boldsymbol{g} + \frac{1}{2}\boldsymbol{b}^T \boldsymbol{b}$$

となり，これを微分すると以下の式が得られる．

$$A^T A \boldsymbol{g} = A^T \boldsymbol{b} \quad (3)$$

すなわち，式 (3) の線形方程式を解くことで最小化できる．

ここで，$A^T A$ は次のような Hessian 行列になる．

$$\begin{bmatrix} (n-1) & \cdots & -1 \\ \vdots & \ddots & \vdots \\ -1 & \cdots & (n-1) \end{bmatrix}$$

式 (3) の右辺の各成分を d_k とすると，次のようになる．

$$d_k = [A^T \boldsymbol{b}]_k = \sum_{k<j} \delta_{kj} - \sum_{i<k} \delta_{ik}$$

したがって，

$$(n-1)g_k - \sum_{l\neq k,j} g_l - g_j = d_k$$
$$(n-1)g_j - \sum_{l\neq k,j} g_l - g_k = d_j$$

上の2つ式を引くと，以下の式が得られる．

$$d_k - d_j = n g_k - n g_j$$
$$g_k = \frac{d_k - d_j + n g_j}{n} \quad (4)$$

式 (4) によってピクセル j の明度からピクセル k の明度が求められる．したがって，任意のピクセルの明度を定めると，式 (4) を用いて，芋づる式に全てピクセルの明度を求めることができる．

2.2 Color2Gray のアルゴリズムの問題点

Color2Gray アルゴリズムでは各ピクセルについて，CIE$L^*a^*b^*$ 色空間の a^*b^* 平面上での距離を計算するために，他の全てのピクセルとの比較を行う必要がある．すなわち視覚的な嗜好を反映できる δ_{ij} を計算するための計算量はピクセル数を n とすると $O(n^2)$ となる．先行研究の計算時間のテスト結果は表1のようになっていた

表1 時間テスト結果 (出展 [4])

画像サイズ	時間 (秒)
100 × 100	12.7
150 × 150	65.6
200 × 200	204.0

このテスト結果をみると，150 × 150 の画像で 65.6 秒かかっている．300 × 300 の画像を処理するには，15 分以上かかるものと想像される．

3. 実時間の Color2Gray 処理

3.1 Color2Gray 高速化の基本的考え方

Color2Gray アルゴリズムの中で最も時間がかかるのは，ピクセルペア間の視覚の差 δ_{ij} の計算である．この計算量は全てピクセル数を n とすると $O(n^2)$ となり，実際の画像を処理することが困難であることは 2.2 節に述べたとおりである．画像を縮小して，そのピクセル数を大幅に減らせば，処理することも可能となるが，画像の詳細部分を失ってしまう恐れがある．そこで，ピクセル数を減らす替わりに，色数を減らすことによって処理の高速化を計ることとした．すなわち，まず，与えられたカラー画像に色量子化を施すことによって色数を減らす．この量子化画像に対する Color2Gray 処理がほぼリアルタイムで実現されるのであれば，ユーザはパラメータをインタラクティブに調整することが可能となる．

3.2 色量子化アルゴリズム

本研究は春日らによって提案された高速 K-means 色量子化法 [7] を用いる．高速 K-means 法は，K-means 法と同様な局所最適解が求まり，K-means 法より高速なアルゴリズムである．これは，K 個あるクラスタ中心を大まかに分類することによって，頻繁に行われるデータとクラスタ中心との距離計算の回数を減らすものである．春日らは高速 K-means 法が通常の K-means 法と同じ結果になり，なおかつ計算回数も少なくなる

という検証をしている．

3.3 量子化画像に対する最適化

3.3.1 最適化式の変形

Gooch らの Color2Gray では，以下の式を最小化することによって所望の明度を求めていた．

$$\sum_{(i,j)\in\mathcal{P}} ((g_i - g_j) - \delta_{ij})^2$$

ここで，$\mathcal{P}$ はピクセルペアの集合であり，g_i, g_j はピクセル i, j の明度．δ_{ij} は色度を考慮したピクセル i, j の明度差である．本研究ではピクセルペア間の計算をクラスタペア間の計算に置き換えることで，処理速度の向上を計る．すべてのピクセル画素値が色量子化されているとすると (1) 式は次のように変形できる．

$$\begin{aligned}
&\sum_{(i,j)\in\mathcal{P}} ((g_i - g_j) - \delta_{ij})^2 \\
&= \sum_{(Q_i,Q_j)\in\mathcal{Q}} \sum_{i\in Q_i, j\in Q_j} ((g_{Qi} - g_{Qj}) - \delta_{Qij})^2 \\
&= \sum_{(Q_i,Q_j)\in\mathcal{Q}} n_{Qi} n_{Qj} ((g_{Qi} - g_{Qj}) - \delta_{Qij})^2 \qquad (5)
\end{aligned}$$

ただし，Q_i, Q_j はクラスタ，$\mathcal{Q}$ はクラスタペアの集合であり，g_{Qi}, g_{Qj} は各クラスタの明度，δ_{Qij} はクラスタ間の明度差，n_{Qi}, n_{Qj} は各クラスタに属するピクセル数となる．色量子化によって得られたクラスタ数を K とすると，δ_{Qij} の計算は $O(K^2)$ となるので，K の値を抑えることによって処理速度を大幅に向上できると考えられる．

3.3.2 重み付き最小二乗法の解法

2.14 節で紹介したように「CompleteK」と呼ばれる方法により最小二乗法の線形方程式を $O(n)$ の手間で解くことができた．しかし，式 (5) の場合には行列 A^TA は Hessian 行列とならないためにこの方法を用いることはできない．

本研究では線形方程式の解法として LU 分解をベースとした方法を用いる．LU 分解は $O(K^3)$ のコストを要することから，クラスタの個数 K を慎重に選ぶ必要がある．なお本研究では比較的高速に LU 分解を実行する SuperLU[8] を用いた．

4. 実験結果

Gooch らの手法による結果と比較するために解像度を落とした画像を用いている．精度については平均絶対値誤差と平均相対誤差を示す．g は Gooch らの手法による結果画像の明度値，g' は提案する量子化を用いた手法による結果画像の明度値，n はすべてのピクセル数とする．平均絶対値誤差は次のようになる．

$$\frac{\sum_{i=1}^{n} |g_i - g'_i|}{n}$$

平均相対誤差は次のようになる．

$$\frac{\sum_{i=1}^{n} |g_i - g'_i|/g_i}{n} \qquad (6)$$

計算時間については提案手法による計算時間を 2 つの部分に分け，色量子化時間と LU 分解時間を測定した．色量子化は一回のみ計算すれば良く，パラメータ θ と α の調整には，LU 分解のみが問題となる．実用的な時間として LU 分解が 0.2 秒以下で処理されることが望まれる．なお使用した計算機は Pentium 4, CPU 2.80GHz, 1.00 GB RAM．OS は WindowsXP．開発環境には Microsoft Visual Studio.Net C++を用いた．Gooch らの手法の結果と比較するために実験にはいずれも 200 × 150 ピクセル程度の画像を用いた．

図 5 は図 1 のモネの「印象・日の出」の画像を Gooch らの手法でモノクロ化した画像である．画像サイズは 200 × 141 ピクセルである．また，Color2Gray のパラメータは θ = 45 度，α=15 である．処理時間は約 202 秒であった．

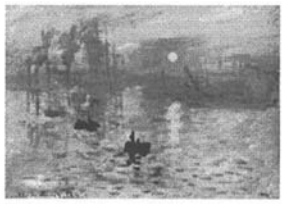

図 5 Gooch らの手法で計算した結果

図 6 は提案手法によって図 1 の画像をモノクロ化した結果と誤差画像である．クラスタ数 K による効果の差を調べるために K を変化させている．アルゴリズムの関係でクラスタ数は結果的に定まるために画像ごとにバラバラの値となっている．誤差画像 g_i'' は次のように計算する．

$$g_i'' = 3(g_i - g'_i) + 128$$

図 7 は図 1 の画像を提案手法によって処理した際の平均絶対値誤差，平均相対誤差，色量子化時間，LU 分解時間のグラフである．

図 8 は「犬」の画像と Photoshop でモノクロ化した画像，Gooch らの手法でモノクロ化した画像である．画像サイズは 200 × 138 ピクセルである．また Color2Gray のパラメータは θ= 295 度．α=16 である．処理時間は約 167 秒であった．

図 9 は提案手法によって図 8 の画像をモノクロ化した結果と誤差画像である． 図 10 は図 9 の結果の平均絶対値誤差, 平均相対誤差，色量子化時間，LU 分解時間のグラフである．

平均絶対値誤差と平均相対誤差いずれもクラスタ数の増加によって減少する．特に色数（クラスタ数）100 前後で平均絶対値誤差は 1 ～ 2 くらいに収まっている．L^* が [0,100] の値をとることから，この程度の誤差で収まれば，比較的な良好な結果と言えよう．またクラスタ数が 200 以上になると量子化の影響も少なく違和感のない結果になっている．

LU 分解については K=200～300 ぐらいでは 0.05 秒であるが，K=500～600 ぐらいになると 0.3 秒と急に

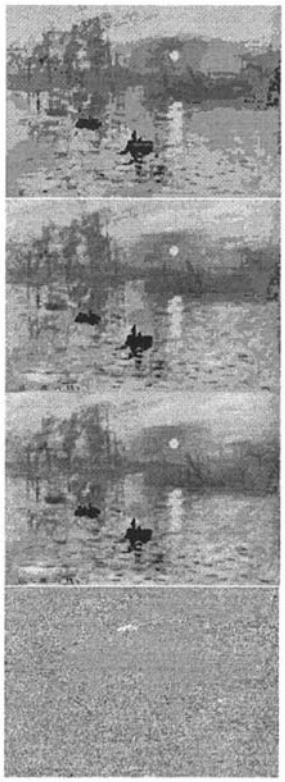

図 6 計算結果上から：K=17,K=65,K=484, 誤差画像 (K=484)

増える．インタラクティブに操作するためには 200〜300 程度が望ましい．

色量子化に要した時間を表 2 に示す．色数 K=500〜600 程度の場合，2 秒強の時間を要している．ただし，この処理は最初に 1 度だけ必要となるものである．参考までに Gooch らの手法処理時間を示すが，色量子化処理時間も十分に短いと言える．

表 2 時間比較

画像	色数	提案手法	Gooch らの手法
日の出	484	2.23 秒	201.982
犬	615	2.4 秒	167.165

5. 結論と展望

本研究では，Color2Gray のパラメータ α と θ をインタラクティブに調整することを目的として，処理の高速化を図った．色数を減らすことによって処理の高速化を実現し，画像の質をある程度保ったまま，ほぼリアルタイムで処理することが可能となった．ユーザはパラメータ値をインタラクティブに調整することが可能となった．

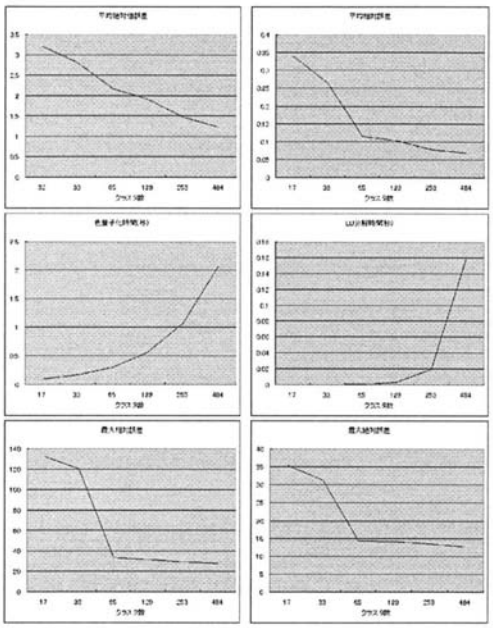

図 7 平均絶対値誤差 (左上)，平均相対誤差 (右上)，色量子化時間 (左中)，LU 分解時間 (右中)，最大相対誤差 (左下)，最大絶対誤差 (右下)

今後は他の量子化方法について実験し，画質と処理時間について検討したい．

参 考 文 献

1) Adobe Photoshop, 2004. Photoshop: Converting color images to black and white. http://www.adobe.com/tips/phs8colorbw/main.html.
2) Brown, R.. 2004. Photoshop tips: Seeing in black and white.
3) Fattal, R.. Lischinski, D. and Werman, M. 2002. Gradient domain high dynamic range compression.
4) Amy A, Gooch Sven C, Olsen Jack Tumblin Bruce Gooch. 2005 Color2Gray: salience-preserving color removal.
5) Rasche, K. Geist, R.. and Westall, J. 2005. Detail preserving reproduction of color images for monochromats and dichromats.
6) Rasche, K. Geist, R.. and Westall, J. 2005. Re-coloring images for gamuts of lower dimension.
7) Hideo Kasuga, 2000, Color Quatization Using the Fast K-Means Algorithm.
8) James W. Demmel, 1999, SuperLu Users' Guide.
9) Volk, C, 2000. Adobe Photoshop Tip of the Week Tutorial.http://www.carlvolk.com/photoshop21.htm.

図 8 入力のカラー画像 (上)，Photoshop で作った結果 (中)，Gooch らの手法による計算結果 (下)

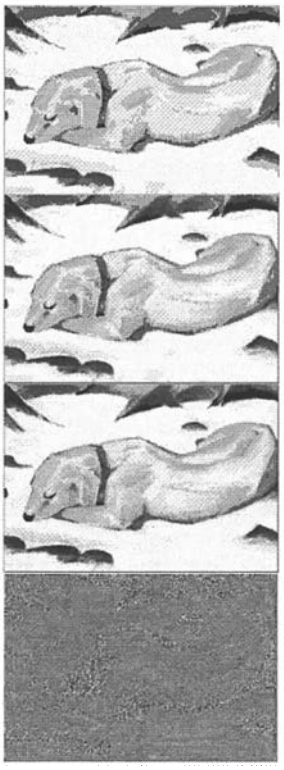

図 9 計算結果上から：K=32,K=106,K=615, 誤差画像 (K=615)

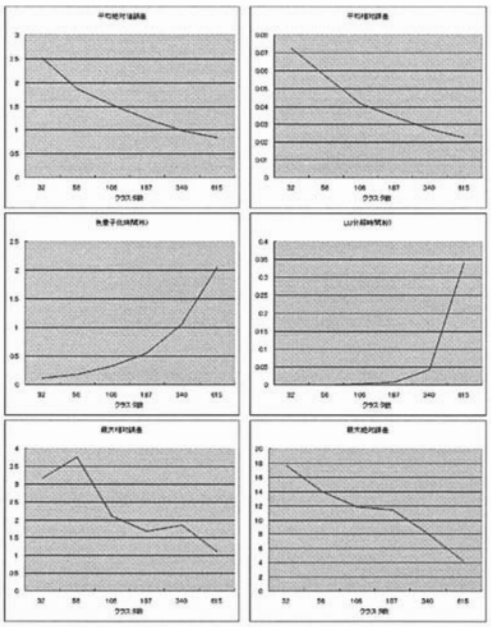

図 10 平均絶対値誤差 (左上)，平均相対誤差 (右上)，色量子化時間 (左中)，LU 分解時間 (右中)，最大相対誤差 (左下)，最大絶対誤差 (右下)