円上の画像圧縮

神戸大学大学院 数学専攻 042S019N 山口 晃広

2005年2月17日

要旨（ニーズ）

画像圧縮には、ＧＩＦで使われている圧縮のように、元の画像に復元できる可逆圧縮、ＪＰＥＧのように復元できない不可逆圧縮などがあります。
また、矩形状の画像だけではなく、円上の画像などを圧縮する必要がある場合もあります。
このように、画像圧縮は、用途に応じて色々な圧縮法を考える必要があります。

要旨（従来の方法の問題点と新しく開発した方法）

従来のＪＰＥＧ圧縮で使われているＣＯＳを使った固有関数展開では、円上の画像をそのまま圧縮しようとするとノイズが入ってしまいます。（ギブズ現象）

そこで、音の圧縮が、弦や膜といった物体の形に合わせた固有関数で展開することで、実際の音色を良く近似できるということに着目し、画像を貼り付ける物体の形に合わせた固有関数を使って展開することを試みました。

このアイデアは高山先生から頂きました。

具体的手法

具体的には、以下のようなラプラス方程式の固有関数を求め、その関数で音（今回は画像）を固有関数展開することを考えます。

$$-\triangle f = \lambda f$$

正方形や円といった圧縮したい物体の境界上で $f = 0$

弦の振動の固有関数展開を$\forall f(x) \quad x \in [0,1]$に適応した例

$$D = [0,1] \qquad f \text{は} x \in D \text{上定義された関数}$$

$$\begin{cases} -\triangle f = \lambda f \\ \triangle = \dfrac{\partial^2}{\partial x^2} \\ f(x) = 0 \quad x \in \partial D \end{cases}$$

上の方程式の固有関数は、

$$\sin \pi n x \qquad (n = 0, 1, 2 \ldots)$$

と求まり、弦の振動は

$$f = \sum_{n=1}^{\infty} a_n \sin \pi n x$$

のように表すことができます。

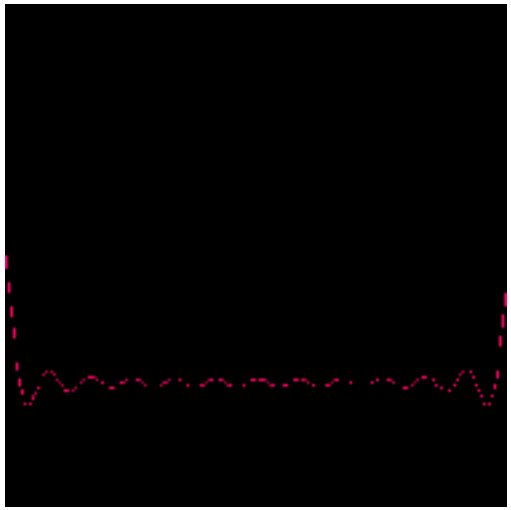
$$f = \sum_{n=1}^{N} a_n \sin \pi n x$$

この級数を上のように有限和で打ち切っても、もとの弦の音色を良く近似できていることが知られています。

この固有関数展開で、級数の和の数が増加するのにともない、例えば

$$f(x) = 1 \quad x \in [0, 1]$$

というグラフが、より良く近似されていくことが観察できます。



矩形上の膜の振動とＪＰＥＧの関係

$D = [0,1] \times [0,1]$　　f は $(x,y) \in D$ 上定義された関数

$$\begin{cases} -\triangle f = \lambda f \\ \triangle = \dfrac{\partial^2}{\partial x^2} + \dfrac{\partial^2}{\partial y^2} \\ f(x,y) = 0 \quad (x,y) \in \partial D \end{cases}$$

この場合、固有関数展開は、

$$f = \sum_{m,n \in \mathcal{N}} a_{mn} \sin \pi m x \sin \pi n y$$

となります。

この固有関数展開を使って画像を圧縮してみます。

$$f = \sum_{m=0}^{M} \sum_{n=0}^{N} a_{mn} \sin \pi m x \sin \pi n y \quad (M, N = 0, 1, 2, \ldots, 30)$$

級数の和が増加していくことで、圧縮率は下がりますが、画像がより綺麗に近似されていくことが分かります。

このsinをcosに変えたものがjpeg圧縮になっています。

$$f = \sum_{m=0}^{M} \sum_{n=0}^{N} a_{mn} \cos \pi m x \cos \pi n y \quad (M, N = 0, 1, 2, \ldots, 71)$$

係数の総数：$72 \times 72 = 5184$

このアイデアを円上の画像圧縮へ適応

$D = \{(x, y) | x^2 + y^2 \leq 1\}$　　f は $(x, y) \in D$ 上定義された関数

$$\begin{cases} -\triangle f = \lambda f \\ \triangle = \dfrac{\partial^2}{\partial x^2} + \dfrac{\partial^2}{\partial y^2} \\ f(x, y) = 0 \quad (x, y) \in \partial D \end{cases}$$

この場合、固有関数展開は、$x = r\cos\theta, y = r\sin\theta$ と極座標変換して、

$$\begin{cases} f(r,\theta) = \displaystyle\sum_{n=0}^{\infty}\sum_{k=1}^{\infty}(a_{nk}\cos n\theta + b_{nk}\sin n\theta)J_n(\mu_k^n r) \\ J_n(x) = \displaystyle\sum_{k=0}^{\infty}\frac{(-1)^k}{k!(n+k)!}\left(\frac{x}{2}\right)^{n+2k} \\ \quad \mu_k^n \in \{x \in \mathcal{R} | J_n(x) = 0, \quad (0 < \mu_1^n < \mu_2^n < \dots)\} \end{cases}$$

となります。

この展開はフーリエ・ベッセルの展開と言います。

$$f(r,\theta) = \sum_{n=0}^{N} \sum_{k=1}^{K} (a_{nk} \cos n\theta + b_{nk} \sin n\theta) J_n(\mu_k^n r)$$

$$N = 0, 1, 2, \ldots, 50 \quad K = 1, 2, 3, \ldots, 50$$

このベッセル関数を使った円上の画像圧縮を観察します。

係数の総数：$(51 \times 50) \times 2 = 5100$

結果の考察

円上の画像に対しては、ＣＯＳで展開するより、ベッセル関数を使って展開した方が綺麗に圧縮できることが分かりました。

矩形上の画像をＣＯＳを使って展開したものと、その正方形に内接する円上の画像をベッセル関数を使って展開したものでは、ほとんど圧縮能力は同じでした。（係数の総数が同じなら、見た目はほとんど優劣がつかない）

今後の課題は以下のようなものがあります。

・計算時間が非常にかかるので、ＪＰＥＧ圧縮のように効率の良い計算法を開発すること。（これができないと実用的ではありません。）
・同じアプローチで、球面に貼り付いたような画像に対する圧縮法を考えてみること。
・見た目では判断がつかない画像に対して、どちらが綺麗に圧縮できているかということを判断する方法を考えること。