順解析と逆解析を相互に利用する打楽器のインタラクティブなデザインインタフェース

山本 和彦　　五十嵐 健夫*

概要.

本稿では, 打楽器の形状と音色のデザインを効率的に行うために, コンピュータによって物体形状からそれを叩いたときの音色を計算する順解析と音色からそれを再現可能な物体形状を推定する逆解析を交互に行い, その支援を行うインタフェースを提案する. 本インタフェースでは, ユーザが物体形状を編集した場合には音色を, 音色自体を編集した場合には対応する物体形状をそれぞれインタラクティブにユーザにフィードバックする. 従来の打楽器を始めとする生楽器の設計では形状データの作成, 解析による確認, 出力による評価の 3 段階のプロセス間の非常に煩雑なトライ&エラーを繰り返す. これを効率化するために, 形状モデリングと同時にその音を解析して, 結果をインタラクティブにフィードバックするインタフェースも過去に提案されているが, 提示された情報を基にユーザはさらにどのような操作を加えれば目的とするものに近づけていくことができるのか, ということを判断するのは困難であるという問題がある. 本システムでは提示された音色情報自体をさらにユーザが編集してその影響を物体形状に反映させることを繰り返すことによってこの問題を解決する. 本稿で提案するような順問題と逆問題を相互に解いてデザインの支援を行うアプローチは楽器のみならず多くの一般的な製品の設計においても役立つ可能性がある.

1　はじめに

現在, 打楽器を含む多くの生楽器の設計プロセスは, 形状モデリング, 解析, 出力/評価の 3 段階から成り立っている. このプロセスでは, まず最初に CAD で形状データを作成する. その後, それを有限要素法等で解析してその音の情報 (モード周波数, 振幅) や実際にそれをものとして出力しても破綻が無いかなどを確認, 最後にそれを実際に物体として出力して今度はその実際の音や演奏感を評価する. しかしながらこの 3 つのプロセスはそれぞれが乖離しており, プロセス間を横断したトライ&エラーは非常に労力がかかる. 例えば, 解析の結果が気に入らなかった場合には再び形状を修正し直してから再解析を行うということを繰り返す. また, 解析プロセスを一度パスできたとしても, その後に出力して実際の出音が所望のものでなかった場合には再び形状モデリングと解析のプロセスを繰り返すことになってしまう. これは非常に煩雑な作業であり, 設計者はプロセス間を横断したトライ&エラーに労力が割かれてしまい, 本来の創造的なデザインに集中することが困難になってしまっているという問題がある.

この煩雑さを解消するために, 梅谷ら [1] は形状モデリングと同時にそれを叩いたときの音の基本周波数の情報を音としてユーザにフィードバックすることによって形状モデリングと解析のプロセスを統合した. しかし, これだけではフィードバックされた情報を基にユーザは, さらにどのような形状変形操作を加えれば目的とする音に近づけていくことができるのか, を判断することが困難であるというさらなる問題がある.

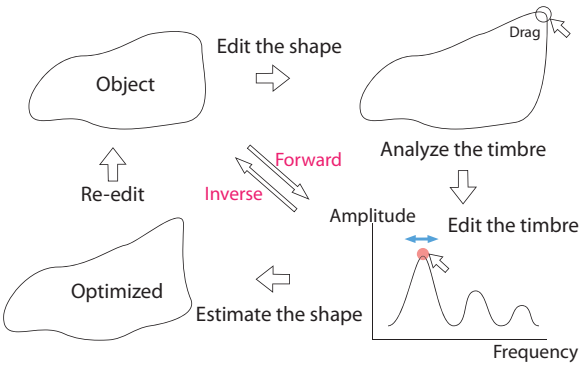

図 1. 順解析と逆解析を繰り返して形状と音色両面から打楽器をデザインする

そこで, 本稿では, 打楽器の形状と音色のデザインを効率的に行うために, コンピュータによって物体形状からそれを叩いたときの音色を計算する順解析と音色からそれを再現可能な物体形状を推定する逆解析を相互に行い, その支援を行うインタフェースを提案する.　本システムではユーザが物体形状を編集した場合には音色を, 音色自体を編集した場合には対応する物体形状をそれぞれインタラクティブにユーザにフィードバックする. これを使ってユーザは, 形状編集操作に応じて提示された音色情報自

* Kazuhiko Yamamoto, Takeo Igarashi, 東京大学

体をさらに編集してその影響を物体形状に再び反映させることを繰り返すことによって, 形状と音色両面から打楽器をデザインしていくことが可能である (図 1).

本稿で提案するような順解析と逆解析を交互に解いてデザインの支援を行うというアプローチは, ユーザが解析手法独特のノウハウ等の余計な問題に煩わせられることなく常にその時々で着目しているデザイン要素にのみ集中することができるようになるという利点がある. これは音だけではなく例えば剛性や熱伝導性能といった問題にも置き換えることで, 楽器のみならず多くの一般的な製品の設計においても役立つ可能性がある.

2 関連研究

コンピュータによって物体の振動音を計算するための最も代表的な手法は, 物体を有限要素法で離散化して固有値解析を行い, 算出されたモード (固有振動) の線形和としてその振動を近似するものである [3]. 固有値解析は非常に計算コストが高く一般的には前計算としてのみ行われる処理であり, インタラクティブな速度で計算を行いデザインツールに利用するためには高速化のアルゴリズムが必要である.

梅谷ら [1] は求める固有値を最低次の固有値のみに限定することで, ユーザが物体形状をモデリングすると同時にインタラクティブな速度で固有値解析を行い, その結果をフィードバック可能にした. この研究ではユーザが鉄琴の形状モデリングを行うと同時にその基本周波数の情報を音としてフィードバックすることによって, 複雑な形状の鉄琴であっても容易にその音高を調整することができる. しかし, 提示される情報が基本周波数のみであり, 楽器としての評価を行うためには不十分であった. C.B.Maxwell ら [4] は固有関数が既知である物体形状からの微小な変形による影響をモード空間に射影して自由度縮小を行うことで準インタラクティブな速度で基本周波数だけではなく高次の周波数成分も計算しており, 音色の評価をすることも可能である. しかし, 元の形状からの変形が大きくなると計算が破綻してしまうという問題がある.

コンピュータによるインタラクティブな物理シミュレーションをデザインツールに利用する研究は近年盛んに行われているが [6][13], そのほとんどが順解析か逆解析のどちらかを単独で利用するものである. 順解析とは, 原因を与えてそこから結果を計算する問題であり, 物体の運動シミュレーション等多くの問題がこれに属する. その逆が逆解析であり, 結果から原因を推定する問題である. 本研究で扱う音色からの物体形状の推定もこの逆問題に属する.

Design by Dragging[5] は順解析と逆解析両方を同時にデザインに利用できるように一つの環境に統合した最初の研究である. この研究では医療機器の

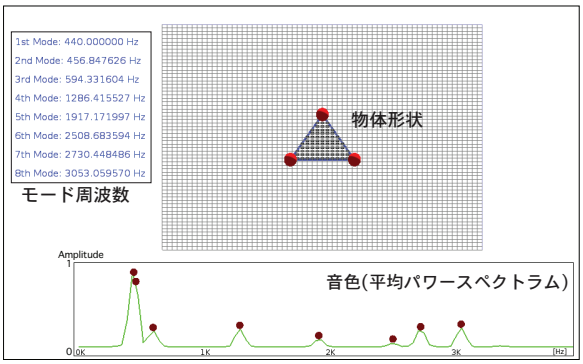

図 2. プロトタイプ画面

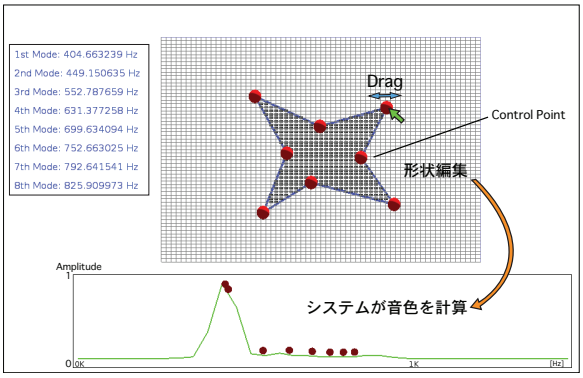

図 3. マウスによる物体形状モデリング

形状を予め細かくパラメータ化しておき, 前計算したシミュレーション結果の中からユーザが現在編集している状況に近いものを検索, 補間することで, ユーザが物体形状を編集した際には構造への負荷が, 結果として表示された構造への負荷状況を編集した場合にはそれによる形状への影響がインタラクティブにフィードバックされる. これによってデザイナーは解析ノウハウを一切気にするこの無く本来のデザイン作業に集中できる. しかし, 予め細かくパラメータ化された形状は編集の自由度に乏しく, デザイナーはツール制作者の想定を超えるような形状を作ることができないという問題がある.

3 ユーザインタフェース

図 2 が本研究で提案するインタフェースのプロトタイプ画面である. これによって打楽器の形状と音色を編集する. 中央に編集する打楽器の形状が表示してあり, 左側に現在の形状でのモード周波数を低周波数から順に表示, 下側には実際に音色を鳴らしたときの音のパワースペクトラムの時間方向の平均値およびその編集画面が用意されている. パワースペクトラムは横軸に周波数 (リニアスケール), 縦軸に正規化されたパワーを示している. 横軸は一番低い周波数のピークと高い周波数のピークが操作しや

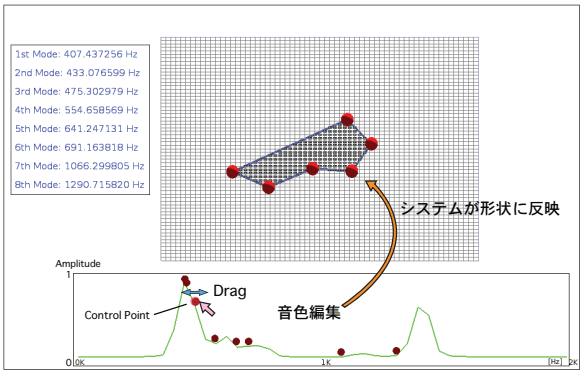

図 4. 周波数領域での音色編集

図 5. テーブルをあたかも打楽器のように叩くことで現在編集している音色を鳴らして確認する.

すいように全体がスケーリングされている.

ユーザは中央の画面において物体の周囲の制御点 (赤い点) をマウスでドラッグして 2 次元的に物体形状を変形する (図 3). また, 制御点を結ぶ辺の中間をクリックすることで物体周囲の任意の位置に制御点を追加することも可能である. ユーザが物体形状を変更するとそれを叩いたときの音色が再計算され, モード周波数とパワースペクトラムの表示が更新される. 一方, 下側のパワースペクトラム表示画面には周波数軸上で現在の形状でのモード周波数の位置に制御点 (赤い点) が設けられており, ユーザはこれをマウスでドラッグすることによりモード周波数の位置を移動させて音色自体を編集することができる (図 4). 音色が編集されるとシステムは現在の形状からできる限り近い形状でその音色を再現できるような形状を計算して画面中央に表示する. 音色編集はあまりにも大幅な変更を許してしまうと計算が収束しないか時間がかかってしまうので現在のモード周波数の近傍に編集範囲を制限する. ユーザはこの形状操作と音色操作の 2 つの操作を繰り返して最終的な目標の形状と音色を同時に作成していく.

ここで, ユーザにとってモード周波数を直接操作することは音色操作の方法として決して最適では無い. しかし音色と形状を結びつけるパラメータであるモード周波数を操作することは, 提案手法を実現する上で最も直接的であるためこの方法を採用した. また, モード周波数の位置はパワースペクトラムのピーク位置近傍となる場合が多いが, 減衰が早いモードではピークとして表れてこない場合もあり, ユーザにとって操作しにくくなってしまうこともある. この点については今後の改善が必要である.

さらに, 音色の確認についてはモード周波数やパワースペクトラムを視覚的に表示するだけでは不十分であり, 実際に音として耳で聴いて評価する必要がある. しかし, 楽器としてその出音が音楽的かどうかという評価を行うためには規則的に音を鳴らしたり, マウスでボタンをクリックしたときに音を鳴らす, というだけでは不十分であり, 実際にユーザがその音を使って音楽を演奏して評価するのが妥当である. そこで本研究では Possessing Drums[10] の手法を音色フィードバックの手段として採用する. これは音響信号処理によって, マイクから入ってきた音 (例えばテーブルを叩いた音) から駆動音 (テーブルの振動特性を差し引いたもの) を抽出した後に, 任意の音響伝達特性を再現できるような共振器に通して音色変換をすることによって身の回りの物体に任意の音を割り当てる研究である. これによって現在編集している物体の伝達特性 (それぞれのモード周波数を持つ共振器) にマイクからの音を通すことで, PC を使って設計を行っているユーザは目の前のテーブルを叩いてシミュレーションされた音色で音楽を試奏することでその評価を行うことが可能となる (図 5).

4 アルゴリズム

4.1 音色のシミュレーション

物体の振動音をシミュレーションするためには有限要素法で線形弾性体として離散化して固有値解析を行うのが一般的である. しかし, 従来固有値分解によって最低モードだけではなく高次モードも含めて求めることは非常に計算コストが高く, インタラクティブな計算に利用することは困難であった. そこで本研究では部分構造合成法 [2] を応用した前処理付きの高速な固有値解析手法を提案する.

有限要素法で離散化された自由度 N の固有値問題は次のような一般化固有値問題に帰着することができる.

$$\boldsymbol{K}\boldsymbol{\phi} = \lambda \boldsymbol{M}\boldsymbol{\phi} \tag{1}$$

$\boldsymbol{K}$ は $N \times N$ の剛性行列, $\boldsymbol{M}$ は質量行列, λ は固有値, $\boldsymbol{\phi}$ は固有ベクトルである. ここで任意の数 $n(n << N)$ だけ固有ベクトルを並べた行列 $\boldsymbol{U}$ によって $\boldsymbol{K'} = \boldsymbol{U}^T\boldsymbol{K}$, $\boldsymbol{M'} = \boldsymbol{U}^T\boldsymbol{M}$, $\phi' = \boldsymbol{U}\boldsymbol{\phi}$ と定義すると式 1 は,

$$\boldsymbol{K'}\boldsymbol{\phi'} = \lambda \boldsymbol{M'}\boldsymbol{\phi'} \tag{2}$$

と変形することができ, 本来の問題に比べて非常に小さな自由度の簡単な問題として解くことができる.

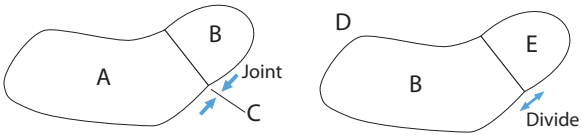

図 6. 部分構造の接合 (左) と分割 (右)

この問題を部分構造合成法を使って解く.

部分構造合成法 [2] は解析対象の構造体を複数の分系に分割し, 分系ごとに自由度縮小して重ねあわせることによって高速に解析を行う手法である. その中でも特に差分モード法 [7] ではある分系の固有関数が既知であった場合にその分系に関しては自由度縮小を行ったまま他の分系と合成することで計算コストを大幅に抑えることができる.

まず, 図 6 左のように物体 A に物体 B を接合し, その共有する節点群を添字 C で表すとする. ここで, 分系 A の固有関数 $\boldsymbol{U}_A$ と分系 B の固有関数 $\boldsymbol{U}_B$ がそれぞれ既知であるとすると, 全体の剛性行列 $\boldsymbol{K'}_{total}$ は自由度を縮小することができ, AB 全体の構造を一度に計算するより大幅に省メモリかつ少ない計算コストで処理することができる.

$$\boldsymbol{K'}_{total} = \begin{bmatrix} \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{U}_{A_C}^T \boldsymbol{K}_{CC} \boldsymbol{U}_{A_C} & \boldsymbol{U}_{A_C}^T \boldsymbol{K}_{CB} \boldsymbol{U}_{B_C} \\ \boldsymbol{0} & \boldsymbol{U}_B^T \boldsymbol{K}_{BC} \boldsymbol{U}_{A_C} & \boldsymbol{U}_B^T \boldsymbol{K}_{BB} \boldsymbol{U}_B \end{bmatrix} + \begin{bmatrix} \boldsymbol{U}_A^T \boldsymbol{K}_{AA} \boldsymbol{U}_A & \boldsymbol{U}_A^T \boldsymbol{K}_{AC} \boldsymbol{U}_{A_C} & \boldsymbol{0} \\ \boldsymbol{U}_{A_C}^T \boldsymbol{K}_{CA} \boldsymbol{U}_A & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \end{bmatrix} \quad (3)$$

各行列の添え字は節点の属する分系を表す. また, $\boldsymbol{U}_{A_C}$ は分系 A の固有関数の C 節点要素, $\boldsymbol{U}_{B_C}$ は分系 B の固有関数の C 節点要素である. 質量行列に関しては同様の形となるため省略する.

本研究では, 図 6 左における部分構造 A はユーザが編集する前の固有関数が既知の形状, B をユーザが編集した後に加わった部分形状と考えることができる. しかし新たに付け加えた形状 B の固有関数は未知であるためこの方法をそのまま利用することはできない. そこで本研究では未知の構造 B の固有関数を高速に求めるために前計算法を導入する.

まず, 前処理として計算領域全体を 2 次元の場合には正方形格子の有限要素法で離散化する. これは 3 次元の場合には立方体格子となる. あらかじめ十分な大きさ N をとり, 2^2, 3^2 4^2...N^2 節点の大きさの正方形形状それぞれにおいて十分な数の固有ベクトルを求めておく (図 7). ここで, 差分モード合成法 [7] では図 6 右のように固有関数 $\boldsymbol{U}_D$ が既知である構造 D から構造 E を取り除いたときの構造 B の自由度縮小された剛性行列 $\boldsymbol{K}_{B'}$ は,

$$\boldsymbol{K}_{B'} = \boldsymbol{U}_D^T \boldsymbol{K}_B \boldsymbol{U}_A = \boldsymbol{U}_D^T \boldsymbol{K}_D \boldsymbol{U}_D - \boldsymbol{U}_D^T \boldsymbol{K}_E \boldsymbol{U}_D \quad (4)$$

と表すことができる.

つまり, 図 6 右における未知の構造 B を覆う構造 D の特性が分かっていれば構造 B の自由度縮小された行列 $\boldsymbol{K}_{B'}$ を構造 B 単独の固有値分解を行うことなく求めることができる. この未知の構造 B を覆う構造 D として前計算した正方形領域の結果を使うと, 図 8 のように変形後の形状は, 変形されていない部分 + (変形された部分を覆う正方形領域 D - 変形された部分を除き, かつ変形されていない部分を除く D 内領域 (図 8 の青色部分)), となる. この第一項は既知であり, 第二項も差分モード合成法の分割処理で計算すれば固有値分解は不要であるのでこれは最終的に自由度が大幅に縮小された後の全体の式 3 の固有値分解を一度解くだけで計算が完了する. これによって従来より高速な固有値解析が可能となる.

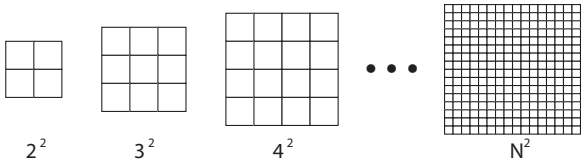

図 7. 前計算を行うグリッド

図 8. 変更後の形状は, 変形されていない部分 + (変形された部分を覆う正方形領域 D - 変形された部分を除きかつ変形されていない部分を除く D 内領域 (青色)) となる.

ここで, 計算精度に関してはそれぞれの部分構造においてどれだけ自由度を残しておくかに依存する. これはユーザが形状をどれだけ大きく変更したかということと計算コストのトレードオフで決定する.

4.2 所望の音色を実現するための形状最適化

いくつかのモード周波数, つまりは与えられた音色からそれに近いモード周波数を実現するような形状を推定する手法について説明する.

本研究では, ユーザが現在編集している形状からできる限り近い形状で所望のモード特性が得られる解をレベルセット法による形状最適化で求める. レベルセット法 [8] とは二つの相で構成される閉じた領域においてそれぞれの相の形態と形状を表現する方法である. 本研究の場合は物体表面からの距離関数 $\phi(x)$ がこれに相当し, 物体の内部で正, 外部で負の値を取り, 表面で零になる. レベルセット法に基づく構造最適化手法 [11] は, トポロジー最適化法に代表される他手法 [12] で問題となる形状の波打ち問題やグレースケール (物体と空洞の中間領域) 問題が発生せず, 計算コストも少ないという利点がある. ここで, 通常, 構造最適化問題では, 得られる構造が

初期構造に大きく依存してしまうことが問題として挙げられるが, 本研究ではユーザが現在編集している形状から最も近い最適形状を得ることを目的としているためにこの点はむしろ利点となる

最適化はレベルセットの関数である目的汎関数 $F(\phi(x))$ を最小化することによって行う. 線形弾性体の固有モード最適化問題においては目的汎関数は, 現在の固有値と目的とする固有値間の残差となる [11].

$$Minimize \quad F(\phi) = \sum_{k=0}^{s+1} \left\{ w_k \left(\frac{\lambda_k - \lambda_{target,k}}{\lambda^2_{target.k}} \right) \right\} \quad (5)$$

ここで λ_k は k 番目の固有値, $\lambda_{target,k}$ は目標とする k 番目の固有値, w_k は重みパラメータである. s は目標とする固有値モードの数であるが, $s+1$ まで考慮しているのは, 最適化の過程で固有ベクトルの形状が変化して次数が入れ替わってしまうことを防ぐためである. この最適化はレベルセット境界に適切な法線方向速度参照領域 $V_N(x,t)$ を与えて陽解法で移流させることで行う.

レベルセットの時間発展を表すレベルセット方程式の弱形式は参照領域 D において

$$\int_D \frac{\phi(x,t+\Delta t) - \phi(x,t)}{\Delta t} \Psi d\Omega + \int_D V_N(x,t) \Psi d\Omega = 0 \quad (6)$$

となる. ここで, Ψ は試行関数である. 形状最適化問題においては左辺第二項は目的汎関数のレベルセット関数に関する変分

$$\begin{aligned} \int_D f(\phi(x)) \psi d\Omega = & \int_D \sum_{k=0}^{s+1} \left\{ 2 w_k \left(\frac{\lambda_k - \lambda_{target,k}}{\lambda^2_{target.k}} \right) \right. \\ & \times (\epsilon(u_k) : \boldsymbol{D} : \epsilon(u_k) - \lambda_k \rho u_k \cdot u_k) \\ & \left. \times \delta(\phi(x)) H(\phi(x)) \right\} \psi d\Omega \end{aligned} \quad (7)$$

で表される. この $f(\phi(x))$ は目的汎関数の形状感度の被積分関数であり, ϵ はひずみテンソル, u_k は k 番目の固有ベクトル, $\boldsymbol{D}$ は弾性テンソル, $H(\phi(x))$ と $\delta(\phi(x))$ はレベルセット関数が $-h \leq \phi \leq h$(h は任意の実数) の領域で値を持つ Heavisible 関数と Dirac Delta 関数 [11] である. 式 6 は有限要素法で離散化して,

$$\phi^{t+\Delta t} = \phi^t - \Delta t \boldsymbol{E}^{-1} \boldsymbol{V}_N^t \quad (8)$$

$$\boldsymbol{E} = \bigcup^{n_e} \int_{V^e} \boldsymbol{N}^T \boldsymbol{N} d\Omega \quad (9)$$

$$\boldsymbol{V}_N^t = \bigcup^{n_e} \int_{V^e} f(\phi(x,t)) \boldsymbol{N} d\Omega \quad (10)$$

に従って目的汎関数が収束するまで各節点のレベルセット関数を更新することで解く. ここで, $\boldsymbol{N}$ は形状関数, $\bigcup$ は要素の重ね合わせを表す.

アルゴリズムの手順としてはユーザが現在編集している固有関数が既知の形状を初期条件とし, 式 10 にてレベルセットを更新する. 更新の度に 4.1 章の方法によって固有値解析をやり直し, 現在の固有値と固有ベクトルを設定し直す. また, レベルセット境界を更新すると, 境界から離れたところのレベルセットの値が無効になるので Fast Marching Method[9] 等を用いて全体を再初期化する必要がある.

ここで, レベルセット法では物体形状を距離関数によって陰的に表現しているため, 計算が収束した後にユーザがさらに形状を編集できるようにするためにはレベルセット法から明示的に物体表面のメッシュを生成してやる必要がある. 本研究では陰関数からのメッシュ生成には Marching Cube 法 [14] を用いた. さらに Marching Cube 法で生成されたメッシュはレベルセット格子と同程度の解像度となっているために細か過ぎてユーザが頂点を操作しにくくなってしまうという問題があるがスムージングを行い, 荒いメッシュに近似することでこれを解決した.

5 結果

5.1 システムの速度と精度

本稿で提案する手法で形状変形を行いながらモード周波数を算出していった場合と, それと同じ形状で従来の固有値解析を行った場合の節点数とおよその処理時間の関係を表 1 に示す. 括弧内の数字は変形前後で追加された節点数である. 評価に使ったシステムは, CPU: Intel Core i7 Quad 3.4GHz, RAM: 32GB で C++で実装し, 各形状において最低次から 8 つの固有周波数を求めた. 既存手法では節点数の増加に応じて計算時間が大幅に増加しているが, 提案手法では僅かな増加で済んでいる.

表 1. 固有値解析の時間 [ms]

節点数 (追加節点数)	既存手法	提案手法
6588(1314)	2000	400
9026(2475)	18000	840
13576(3081)	37000	1030

一方, 形状最適化の精度に関しても評価も行った. 最低次から 8 つのモードを対象として目標値を定め, 初期状態から最適化を行った結果が表 2 である. 結果の 6 次モードに新たなモードが入ってきており, 次数が変わっているが, 目標とする固有周波数群自体はほぼ得られてることが分かる. 8 次の目標値も達成するためには最適化の対象にするモードの数をさらに増やす必要がある.

5.2 エンジニアによる試用

本システムを実際に CAD や解析ソフトウェアの使用経験のあるエンジニア数名に試用してもらった後, インタビューを行った. 被験者には自分の好みの打楽器を作ることを想定してデザインを行うよう

表 2. 固有周波数最適化の結果 (単位 [Hz])

	初期状態	目標値	結果
1st	440	409	410
2nd	674	738	764
3rd	758	967	955
4th	1119	1367	1312
5th	4305	4776	4633
6th	8656	9500	5342
7th	9912	10649	9408
8th	11279	12567	10610

に指示した.

彼らは普段の設計では既に過去に実績のあるCADデータを雛型として, それに編集を加えていくことで新しい製品を作っていくことが多いとのことである. そのため, 本システムのアプローチは目的となる音色側からも操作を加えていくことができると楽器音に不快な成分が含まれていた場合に抑えるのが容易になったり, または新たな楽器音を探していく際にも非常に便利である, と意見を述べた. また, 逆解析の結果得られた形状そのものを再びインタラクティブに修正していけるため, 普段使っている解析ツールにも本システムが取り入れられると作業が容易になるとの意見を得た.

しかし, 実際の楽器は対称な形状のものがほとんどであるので, 対称性を前提にしたインタフェースにしたほうがよいとの意見も得られた. さらに, 生楽器では形状を自由に設計するという需要はあまり無く, 例えばスピーカ筐体等の設計に利用する方がより有用であるとの意見も多かった. これらは今後の課題とする.

6 まとめと今後の課題

本稿では, 物体形状からそれを叩いたときの音色を計算する順解析と音色からそれを再現可能な物体形状を推定する逆解析を相互に行うことで形状と音色両面からの効率的な打楽器デザインを行うことのできるインタフェースを提案し, 実際にエンジニアにも試用してもらうことでその創造的なデザインに役立つ可能性があることを確認した.

しかし, 本システムの有用性を評価するためにはソフトウェア上での評価だけでは不十分であり, 今後は実際に作成した打楽器を実物として出力して評価を行っていく必要がある. さらに, 実際の楽器では数多くの部品が組み合わされているため, 本稿の手法だけでは実用的とは言えない. 複数の部品の組み合わせを考慮することは今後の課題である. また, 音色からの形状推定ではユーザの操作次第では物理的に実現不可能な音色が設定されてしまうことがあるという問題もある. このため予めユーザに編集可能な範囲をシステム側から提示できることが好ましいが, 最適化を行う前の段階ではその音色が実現可能がどうかを知ることは難しく, 今後の課題である.

参考文献

[1] 梅谷信行, 高山健志, 三谷純, 五十嵐健夫. 実時間固有値解析による対話的な鉄琴のデザイン. 情報処理学会論文誌, Vol.52, No.4, pp. 1599–1607, 2011.

[2] 長松昭男, 大熊政明. 部分構造合成法. 培風館, 1991.

[3] Changxi Z., and Doug L. J. Toward High-Quality Modal Contact Sound. In *ACM Transactions on Graphics, 30(4)*, 2011.

[4] Cynthia B. M., and David B. Modal Parameter Tracking for Shape-Changing Geometric Objects. In *Proc. of the 10th Int. Conference on Digital Audio Effects (DAFx-07)*, 2007.

[5] Dane C., Chi-Lun L, Arthur G. E., Daniel F. K. Design by Dragging: An Interface for Creative Forward and Inverse Design with Simulation Ensembles. In *Transactions on Graphics 27(3)*, 2008.

[6] Greg S., Manfred L., Mitani J., and Igarashi T. SketchChair: An All-in-one Chair Design System for End-users. In *the fifth International conference on Tangible, Embedded and Embodied Interaction*, pp. 260, 2011.

[7] Mochizuki T., and Ogawa M. Efficient Approach for Structural Modification Analysis. In *Review of Automotive Engineering, Vol.27, No.1*, pp. 75–82, 2006.

[8] Osher S., and Fedkiw R. Level Set Methods and Dynamic Implicit Surfaces. In *Springer*, 2002.

[9] Sethian J. A. Level Set Methods and Fast Marching Methods. Cambridge Monographs on Applied and Computational Mathematics. In *Cambridge University Press, 2nd edition*, 1999.

[10] Yamamoto K. Possessing Drums: An Interface of Musical Instruments that Assigns Arbitrary Timbres to Personal Belongings. In *Journal of Information Processing, Vol.21 No.2*, pp. 274–182, 2013.

[11] Yamasaki S., Nishiwaki S., Izui K., and Yoshimura M. A structural optimization method based on the level set method using a new type of re-initialization scheme. In *International Conference on Computational Methods*, pp. 260, 2007.

[12] Yang R. J., and Chung C. H. Optimal Topology Design using Linear Programming. In *Computers and Structures*, Vol.53, pp. 265–275, 1994.

[13] Umetani N., Igarashi T., and Niloy J. M. Guided Exploration of Physically Valid Shapes for Furniture Design. In *ACM Transactions on Graphics 31(4)*, 2012.

[14] W.Lorensen, H.Cline. Marching Cubes: A High Resolution 3D Surface Construction Algorithm. In *Computer Graphics, 21 (4)*, pp. 163–169, 1987.