TR-AC-0056 010

スプラインQoSマッピング関数の構成法と自動計測に関する考察

坂本 将清(学外実習生)　山崎 達也

2001. 3.21

ATR環境適応通信研究所

スプラインQoSマッピング関数の構成法と自動計測に関する考察

坂本　将清[†] (sakamoto@haguro.nagaokaut.ac.jp)　山崎　達也[‡]
[†]長岡技術科学大学
[‡]㈱エイ・ティ・アール環境適応通信研究所

あらまし
マルチメディア通信におけるQoS(Quality of Service)の概念は用いられる階層によって異なり，隣接する階層間でそれぞれのQoSを対応づけするQoSマッピングの機構が必要である．本レポートではユーザの任意の品質要求に対応することを目的とし，少数のマッピングのサンプルデータより連続的なQoSマッピング関数を構成する方法を提案する．マッピング関数はスプライン関数により実現され，特に B-スプラインを用いた構成法が有用であることを述べる．さらにサンプルデータを取得するための測定者の負荷を軽減するため，QoSマッピングデータの自動計測法を提案し，実機において実装した結果を報告する．

Composition of Spline QoS Mapping Function and Implementation for Automatic Measurement

Masakiyo Sakamoto[†] (sakamoto@haguro.nagaokaut.ac.jp)　Tatsuya Yamazaki[‡]
[†]Nagaoka University of Technology
[‡]ATR Adaptive Communications Research Laboratories

Abstract
Expression of QoS in distributed multimedia applications is different from level to level. Therefore QoS mapping mechanisms, which translate the QoS expression from level to level, is required. In this report, a scheme to compose continuous QoS mapping functions from several discrete sample data is proposed in order to accept any QoS requirement from end-users. The mapping functions are specified by spline functions; especially B-splines are appropriate and useful to compose the mapping functions. Next, a method to measure the QoS mapping data automatically is proposed. The method is useful to mitigate tedious work of data measurement. The proposed automatic measurement method has been implemented in real PC and the experimental results of measurement are presented in this report.

1. 研究の背景

近年のコンピュータのダウンサイジングや通信ネットワークのデジタル/高速化の潮流に乗り，マルチメディア情報通信のモバイル化，パーソナル化が急速に進んでいる．そこで用いられる通信アプリケーションの動作環境はユーザの用いる端末やネットワークの種類に応じて変化し，それとともに使用可能な資源量も動的に変化する．またユーザのアプリケーションに対する要求もその時々で異なる．このような状況下で End-to-End でアプリケーションの QoS(Quality of Service)を保証したいという要求が高まってきており，そのためには使用可能な資源量およびユーザ要求に応じた QoS の適応制御が必要となる．その際，QoS の概念は用いられる階層によって異なり，隣接する階層間でそれぞれの QoS を相互に対応付けすることが必要であり，このような対応付けは QoS マッピングと呼ばれる[1].

一般的に QoS マッピングはテーブルルックアップ（表参照）方式と，数式を用いた方式に二分されるが[1]，本レポートではユーザの任意の品質要求に対応することを目的とし，連続した QoS マッピングが可能な数式を用いた方式を扱う．特に，数式の陽的な表現を与えなくても少数のサンプルデータより構成されるという利点を有する，スプライン関数を用いた QoS マッピング関数の構成法について検討する．さらにサンプルデータの取得についてはこれまで具体的な検討はほとんどされてきていなかったが，これらを人間による目視による取得では非常に負担が大きいという現実問題に着目し，サンプルデータを自動測定により取得する方法を提案する．そして実機上に提案する自動測定方式を実装し目視により得られたデータと比較，検討を行う．

以下，まず 2.で本レポートで取り扱う動画像メディアに対して QoS の定義を行い，3.で QoS マッピングについて述べ，スプライン関数を用いた二種類の構成法の提案を行う．またこれらの提案構成法の比較評価実験の方法と結果を 4.に示す．さらに 5.ではマッピング関数の自動計測法の提案を行い，6.で自動計測実験の方法と結果を示す．7.でまとめを行う．

2. QoS の定義

本研究では，ストリーミングメディア中でも特に動画像メディアを対象として研究を行っている．動画像メディアは MPEG-4 または M-JPEG(Moution JPEG)圧縮符号化され，伝送される．動画像の品質を決定するパラメータとして，フレームサイズ，フレームレート，量子化スケールの三つを用い，これらをまとめてアプリケーション QoS と呼ぶこととする．フレームサイズは１フレーム内の画素数に相当し，フレームレートは１秒間に表示されるフレームの枚数である．量子化スケールは圧縮符号量削減のために用いられる量子化のステップ幅に関連したパラメータであり，一般に量子化ステップ幅が大きい程圧縮符号量が小さくなるが，情報量削減により再生される動画像に劣化が生じユーザの知覚する品質に対する評価も低くなる．

また圧縮符号データを伝送するために必要な帯域を通信 QoS として定義し，送受信端末で動画像を圧縮・伸長・表示するために必要な CPU 稼働率を端末資源 QoS と定義して，これらをまとめて資源 QoS と呼ぶこととする．伝送に必要な帯域は，一秒間に送信端末から受信端末へ伝送される平均データ量(bit per second: bps)で表され，CPU 稼働率は百分率(%)で表される．

3. 提案する QoS マッピング手法

3.1 QoS マッピング

動画像メディアのストリーミングにおいて，あるアプリケーション QoS が与えられた時に，それに対して必要とされる資源 QoS をできるだけ正確に見積もることは，QoS-aware なマルチメディアアプリケーションを構成する上で重要である．QoS マッピングとは，このようなアプリケーション QoS と資源 QoS の対応付けを行うことをいう．本研究では，スプライン関数を用いた QoS マッピング手法を提案する．

3.2 スプライン関数を用いた QoS マッピング手法の提案

提案する QoS マッピング手法では，アプリケーション QoS と資源 QoS の対応付けを，マッピング関数と呼ばれる数式を用いて行う．マッピング関数は任意のアプリケーション QoS セットが与えられたとき，これに対応する資源 QoS を算出するものである．提案法では，あらかじめ用意されたアプリケーション QoS と資源 QoS のセットをサンプルデータとし，これらのデータをスプライン補間することで得られたスプライン関数をマッピング関数として用いている点に特徴がある．図 3.1 にマッピング関数を得る過程を示す．図中の点は，あらかじめ与えられたアプリケーション QoS と資源 QoS のセット（サンプルデータ）を示し，これら 4 つのサンプルデータをスプライン補間することで，図中の線で示されたスプライン関数が得られる．このスプライン関数をマッピング関数とし，サンプルデータ点以外のアプリケーション QoS に対する資源 QoS の推定を行う．

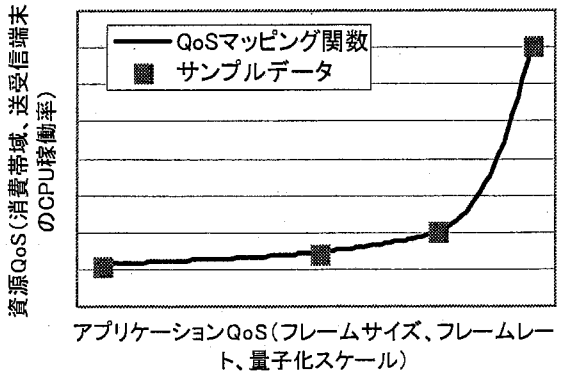


図 3.1 スプライン補間によるマッピング関数の獲得

スプライン関数は区分的多項式であり，多項式と多項式の結び目は節点（ふしてん）と呼ばれる．すなわち，節点と節点の間の区分的な関数にはそれぞれ一つの多項式が対応しており，かつ隣接する多項式は節点で適当な滑らかさを持つということが，スプライン関数の性質である．スプライン関数にはいくつかの表現方法があり，主に切断べき関数や B-スプラインなどが用いられる．n 個のサンプルデータ点$(x_1, y_1), (x_2, y_2), \cdots, (x_n, y_n)$が与えられているとき，これらをスプライン補間する方法にはいくつかあり，先程述べたスプライン関数の表現方法によって異なっている．スプライン関数に関する詳しい解説は例えば参考文献[2]-[4]にある．

本研究では，以下に示す2つのスプライン補間法を用いてQoSマッピング関数を生成し，それぞれを補間法①，②として比較，検証を行う．

（1）補間法①：切断べき関数による表現

n 個のサンプルデータ点$(x_1, y_1), (x_2, y_2), \cdots, (x_n, y_n)$と同じ節点 $x_1, x_2, \cdots, x_n$ を持つ$(2k-1)$次$(k<n)$の自然スプライン関数 $s(x)$を，切断べき関数により構成する．このとき

$$s(x_i) = y_i \ (i=1, 2, \cdots, n)$$

を満足する$(2k-1)$次の自然スプライン関数はただ1つだけ存在し最小補間性を持つ．本研究では，$k=2$ として3次のスプライン関数を用いる．3次とする理由は，1次や2次では近似の精度という点で不十分であり，3次を超えれば計算量がかなり増加するためである．

（2）補間法②：B-スプラインによる表現

B-スプラインは，自然スプラインや C-スプラインなど全てのスプライン関数の基底となるスプライン関数である．N個のサンプルデータ点$(x_1, y_1), (x_2, y_2), \cdots, (x_N, y_N)$を補間する$(m-1)$次のスプライン関数 $s(x)$の基底関数として，$(m-1)$次の B-スプラインを用いる．基底となる B-スプラインを構成するには，内部節点と呼ばれる n 個の節点 $\xi_1, \xi_2, \cdots, \xi_n$ $(\xi_1 < \xi_2 < \cdots < \xi_n)$ に加え，付加節点と呼ばれる $2m$ 個の節点 $\xi_{1-m}, \xi_{2-m}, \cdots, \xi_0$； $\xi_{n+1}, \xi_{n+2}, \cdots, \xi_{n+m}$ が必要となる．ここで $\xi_{1-m} < \xi_{2-m} < \cdots < \xi_0 \leq x_1$ および $x_N \leq \xi_{n+1} < \xi_{n+2} < \cdots < \xi_{n+m}$ である．このとき補間が行える条件として，サンプルデータ数 N と次数$(m-1)$および内部節点数 n の関係が $N=m+n$ となり，節点がシェーンバーグ・ホイットニ(Schoenberg-Whitney)の条件を満たしている必要がある．B-スプラインにおける節点は，必ずしもサンプルデータ点と一致させる必要はなく，シェーンバーグ・ホイットニの条件を満たす範囲であれば移動させることが可能である．節点を移動させるとスプライン補間曲線の形状が変化する．このため B-スプラインでは節点の決定が重要な意味を持ち，節点がQoSマッピングの精度に大きな影響を及ぼす．しかし現段階では節点を決定する理論的な方法はなく，試行錯誤的な判断に基づいて決定するしかない．本研究では3次の B-スプラインを用い，付加節点を

$$\xi_{1-m} = \xi_{2-m} = \cdots = \xi_0 = x_1 \quad (1)$$

$$\xi_{n+1} = \xi_{n+2} = \cdots = \xi_{n+m} = x_N \quad (2)$$

とし，内部節点は試行錯誤的な判断に基づいて決定している．

また本研究では，スプライン補間に端点条件を加えることで，スプライン補間の精度を向上させている．端点条件は，補間を行う区間を$[a, b]$とするとき，区間$[a, b]$の端でスプライン関数$s(x)$の微係数$s'(a)$, $s'(b)$を与える．端点条件を用いる場合には，補間を行う前にこれら二つの微係数を知っておく必要がある．提案法では，N個のサンプルデータ(x_1, y_1), (x_2, y_2), …, (x_N, y_N)が与えられたとき，補間区間を$[x_1, x_N]$として，端点条件は次式で与えられるものとする．

$$s'(x_1)=\frac{y_2-y_1}{x_2-x_1}, \qquad s'(x_N)=\frac{y_N-y_{N-1}}{x_N-x_{N-1}} \tag{3}$$

4. 提案法の評価実験

4.1 実験目的

提案したスプライン関数によるQoSマッピング手法の評価を行う．まずビデオQoS測定実験により，スプライン関数を用いたQoSマッピングを行うための基礎データの収集を行う．この測定で得られたデータのうちいくつかをサンプルデータとして，前述の提案法でQoSマッピング関数を構成し，このマッピング関数による推測値と実際の測定値との比較を行う．スプライン補間には，節点をサンプルデータ点と一致させた切断べき関数による方法（補間法①）とB-スプラインによる方法（補間法②）の二つを使用し，マッピング結果の比較を行う．

4.2 実験方法

二つのPC(OS: WindowsNT)をイーサネットで接続し，一方を送信端末とし他方を受信端末とする．送信端末にはカメラが備え付けられており，リアルタイムで取り込まれた動画像がMPEG-4符号化またはM-JPEG符号化により圧縮されて受信端末へ送られる．本実験ではM-JPEG符号化方式の測定の場合には受信端末としてノート型を用い，MPEG-4符号化方式の測定の場合には受信端末としてデスクトップ型を使用した．受信端末では送られてきた画像を伸長し画面へ表示する．また受信端末ではビデオのフレームサイズ・フレームレート・量子化スケールをユーザの指定により決定でき，この情報は送信端末に送られ，送信端末ではユーザの指定したパラメータにより動画像を圧縮し，受信端末へ送信する．送受信端末ではそれぞれ画像の取り込み・符号化，そして画像の伸長・表示によりCPUリソースが消費され，ビデオ伝送により帯域が消費される．実験ではこれら送受信端末のCPU稼動率と消費帯域を測定する．測定にはリソースモニタを使用し，ディスプレイに表示される値を測定者が記録する．

M-JPEG方式での測定では，3つのフレームサイズ640×480,320×240,160×120のそれぞれに対し，フレームレートと量子化スケールを小さい値から順に大きな値へと，受信

側の CPU 性能が飽和するまで変化させ，それぞれのアプリケーション QoS の組み合わせに対する資源 QoS を測定した．なお，量子化スケールは 5 から 95 までの間の 5 ステップ毎の 19 点を測定し，フレームレートは CPU 性能が飽和するまで 1 ずつ増加させた．なお M-JPEG 方式での測定では，端末資源 QoS は受信端末の CPU 稼働率のみ測定を行った．

MPEG-4 方式の測定では，3 つのフレームサイズ 352×288, 264×216, 176×144 のそれぞれに対し，M-JPEG 方式と同じ方法で資源 QoS を測定した．なお MPEG-4 方式では量子化スケールの測定点を 1,2,3,4,5,10,15,20,25,31 の 10 点とした．

以上の測定点よりスプライン関数を構成するために必要な少数のサンプルデータを選び出し，スプライン補間により QoS マッピング関数を生成した．生成した QoS マッピング関数を用いて，サンプルデータ以外のアプリケーション QoS に対する資源 QoS の推定値を求め，測定値との比較を行うことによりマッピング関数の精度を評価した．

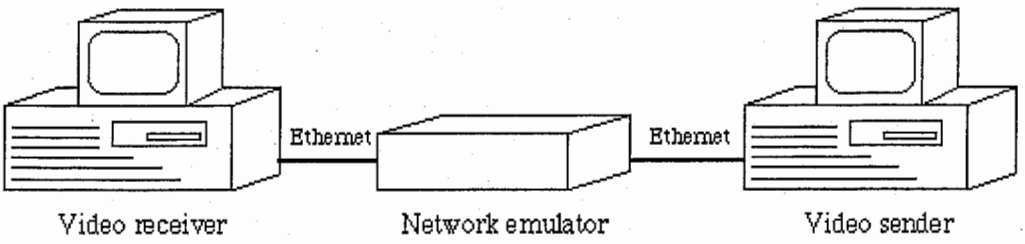


図 4.1　実験システム

4.3 測定結果

（1）M-JPEG 方式での測定結果

M-JPEG 方式での測定結果と QoS マッピング関数の一例を次ページの図 4.2-4.5 に示す．図 4.2, 4.3 はフレームサイズを 160×120，フレームレートを 10 に固定した場合の，量子化スケール 5～95 に対する消費帯域と受信端末の CPU 稼動率である．横軸は量子化スケール，縦軸は消費帯域[kbps]あるいは CPU 稼働率[%]を示す．図 4.4, 4.5 は，フレームサイズを 160×120，量子化スケールを 50 に固定した場合の，フレームレート 1～19 に対する消費帯域と受信側 CPU 稼動率で，横軸がフレームレート，縦軸が消費帯域[kbps]あるいは CPU 稼働率[%]を示している．図中の点が測定値を示し，線はマッピング関数を示している．

図 4.2, 4.3 は，測定値から量子化スケールが 5,50,75,95 の計 4 点をサンプルデータとして，スプライン補間により得られた QoS マッピング関数である．図 4.2(a), 4.3(a)は，補間法①によるマッピング関数であり，図 4.2(b), 4.3(b)は補間法②によるマッピング関数である．図 4.2(a)の補間法①によるマッピング関数は，測定値に対して振動しており，この場合サンプルデータ点を除いた測定値に対する誤差率を求めると，量子化スケールと帯域のマッピングでは 13.9%, 量子化スケールと受信端末 CPU 稼働率のマッピングでは 0.92%であった．図 4.2(b)は，補間法②により，マッピング関数が最も測定値に近づくように節点を試行錯誤的に変化させた結果であり，図 4.2(a)のような振動は見られない．この場合，節点に

は{5, 5, 5, 5, 70, 94, 95, 95, 95, 95}を使用している．誤差率は，量子化スケールと消費帯域のマッピングでは1.49%,量子化スケールと端末資源QoSのマッピングでは0.76%となり，補間法①と比べ，誤差率は大幅に減少している．

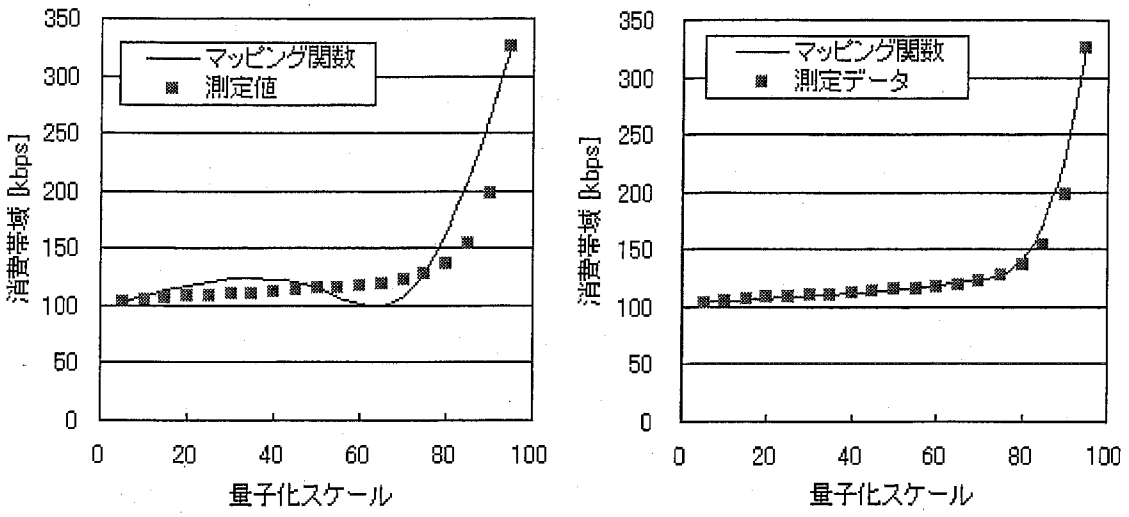
図4.4, 4.5は，測定値よりフレームレートが1, 7, 13, 19の計4点をサンプルデータとして，スプライン補間により得られたマッピング関数である．図 4.4(a), 4.5(a)は補間法①によるマッピング関数で，誤差率は，フレームレートと消費帯域のマッピングでは 3.77%，フレームレートと受信側 CPU 稼働率のマッピングでは 4.60%であった．図 4.4(b), 4.5(b)は補間法①によるQoSマッピング関数で,節点を{1, 1, 1, 1, 7, 13, 19, 19, 19, 19}とした時，誤差率はフレームレートと通信 QoS のマッピングで 3.56%，フレームレートと端末資源QoSのマッピングでは4.70%となった．

（2）MPEG-4方式での測定結果

MPEG-4方式での測定結果とマッピング関数の一例を図4.6-4.11に示す.図4.6-4.8はフレームサイズを352×288，フレームレートを5に固定した場合の，量子化スケール1～31に対する消費帯域および送受信端末のCPU稼働率である．図4.9-4.11は，フレームサイズを352×288，量子化スケールを1に固定した場合の，フレームレート1～8に対する消費帯域あるいは送受信端末のCPU稼働率である．図中の点が測定値を示し，線がマッピング関数を示す．

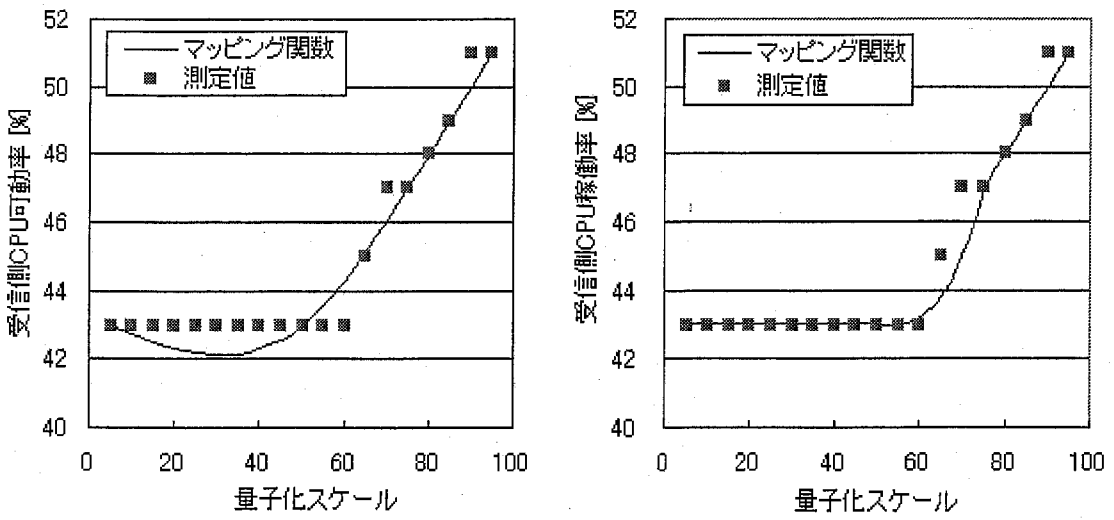
図4.6-4.8は，測定値より量子化スケールが1, 2, 5, 20, 31の計5点をサンプルデータとして，スプライン補間により得られたマッピング関数である．図4.6(a), 4.7(a), 4.8(a)は補間法①によるマッピング関数であり，サンプルデータ点を除いた測定値に対する誤差率は，量子化スケールと消費帯域のマッピングでは4.25×10^3%,量子化スケールと送信端末CPU稼働率のマッピングでは48.5%，量子化スケールと受信端末CPU稼働率のマッピングでは80.3%となった.図4.6(b), 4.7(b), 4.8(b)は補間法②によるスプライン補間を行ったマッピング関数であり，マッピング関数が測定値に近づくように節点を試行錯誤的に決定しており，節点には{1, 1, 1, 1, 2, 2, 5, 31, 31, 31, 31}を用いている．この場合の誤差率は，量子化スケールと消費帯域のマッピングでは11.3%,量子化スケールと送信端末CPU稼働率のマッピングでは0.85%,量子化スケールと受信端末CPU稼働率のマッピングでは10.4%となった.

図4.9-4.11は，測定値よりフレームレートが1, 4, 8の3点をサンプルデータとしスプライン補間により得られたマッピング関数である．図4.9(a), 4.10(a), 4.11(a)は補間法①によるマッピング関数であり，サンプルデータ点を除いた測定値に対する誤差率は，フレームレートと消費帯域のマッピングでは3.04%，フレームレートと送信端末CPU稼働率のマッピングでは4.10%，フレームレートと受信端末CPU稼働率のマッピングでは3.01%となった．図4.9(b), 4.10(b), 4.11(b)は補間法②によるマッピング関数であり，節点には{1, 1, 1, 1, 4, 8, 8, 8, 8}を用いた．この場合の誤差率は，フレームレートと消費帯域のマッピングでは3.63%，フレームレートと送信端末CPU稼働率のマッピングでは5.25%，フレームレートと受信端末CPU稼働率のマッピングでは5.21%となった.



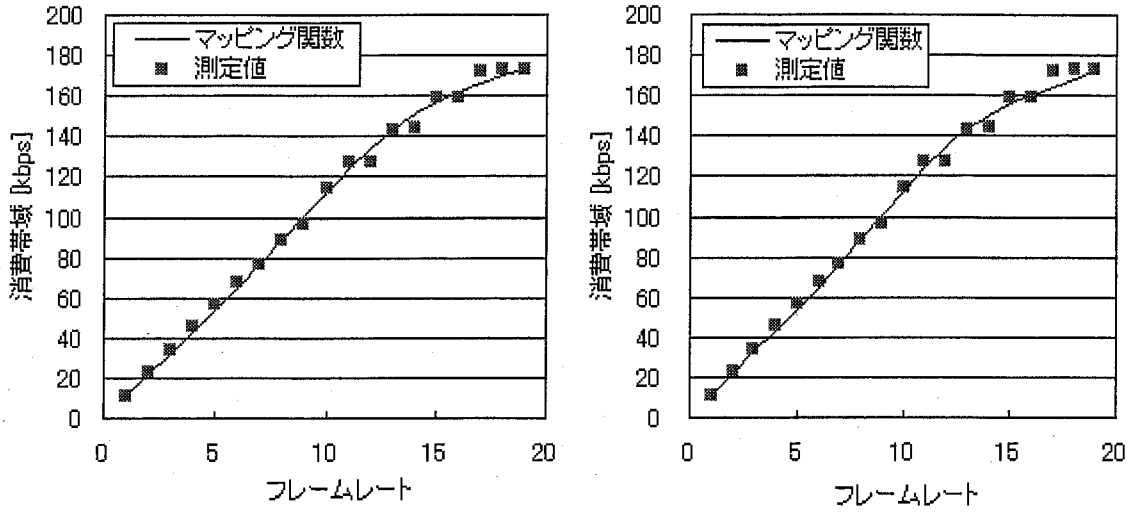
(a) 補間法①　　(b) 補間法②

図 4-2. M-JPEG 方式での測定結果①



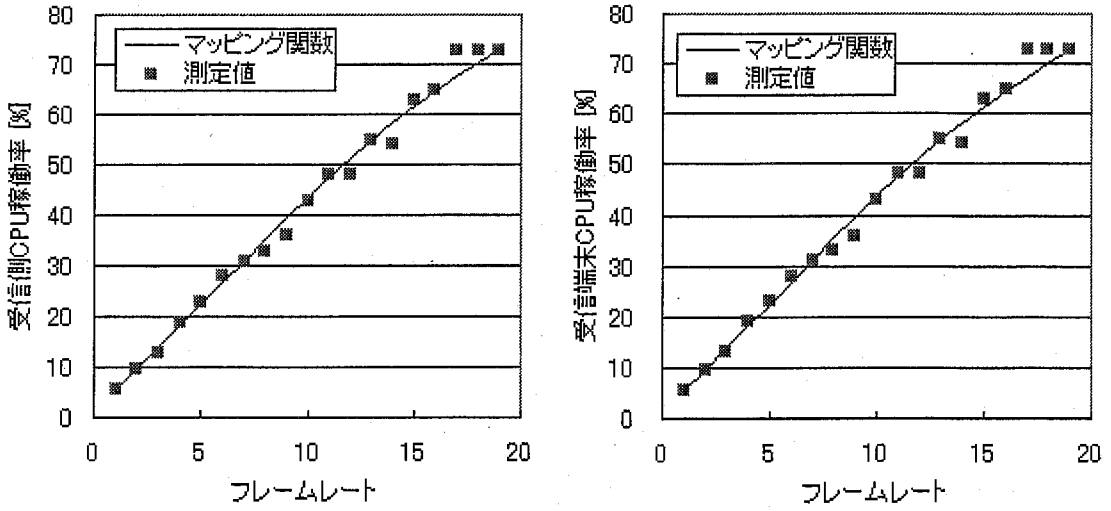
(a) 補間法①　　(b) 補間法②

図 4-3. M-JPEG 方式での測定結果②



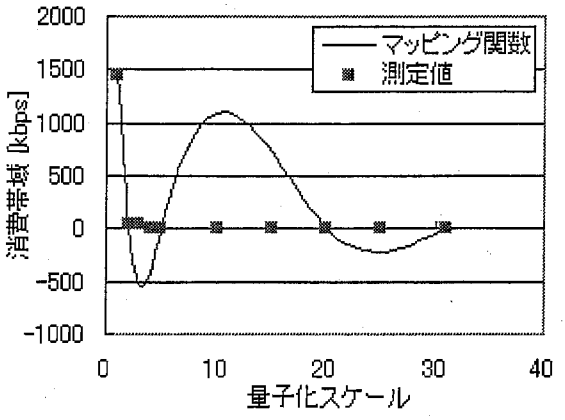
(a) 補間法①　　(b) 補間法②

図 4-4. M-JPEG 方式での測定結果③

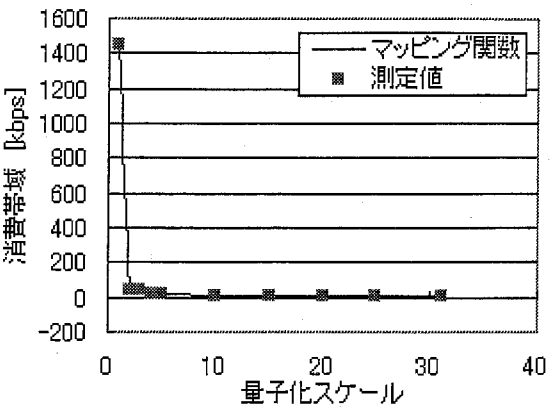


(a) 補間法①　　(b) 補間法②

図 4-5. M-JPEG 方式での測定結果④

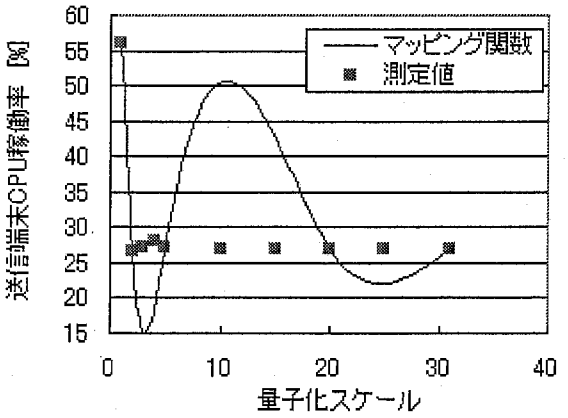


(a) 補間法①

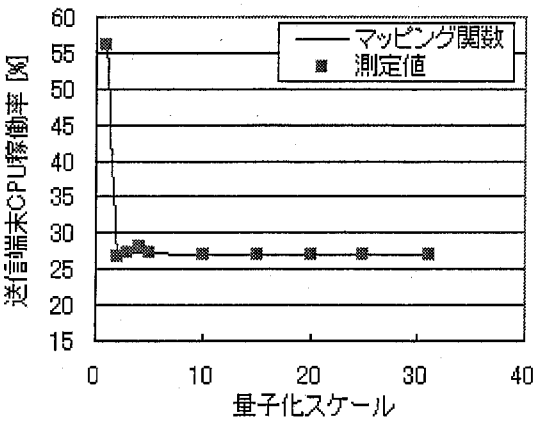


(b) 補間法②

図 4-6. MPEG-4 方式での測定結果①

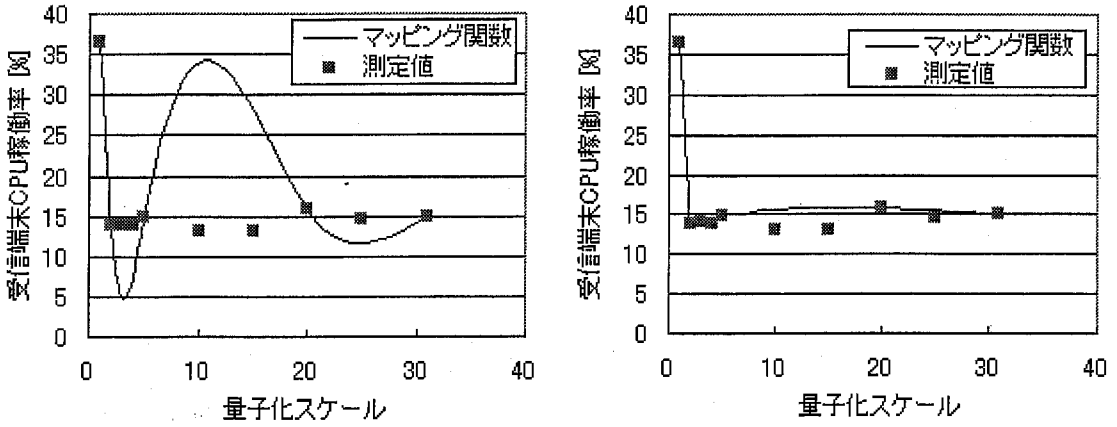


(a) 補間法①



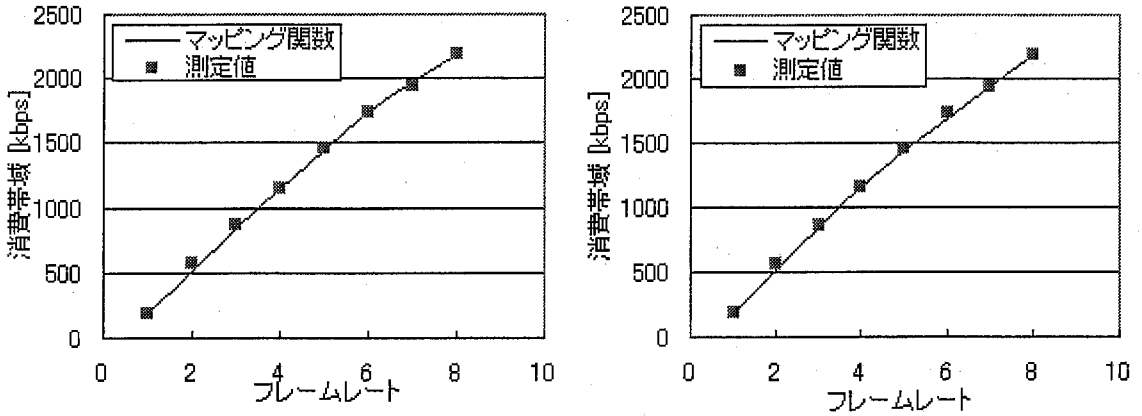
(b) 補間法②

図 4-7. MPEG-4 方式での測定結果②



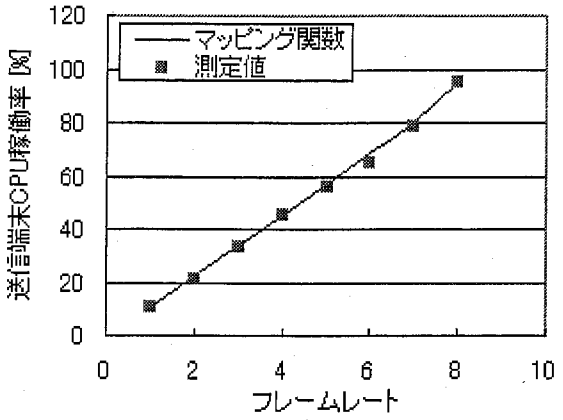
(a) 補間法①　　(b) 補間法②

図 4-8. MPEG-4 方式での測定結果③

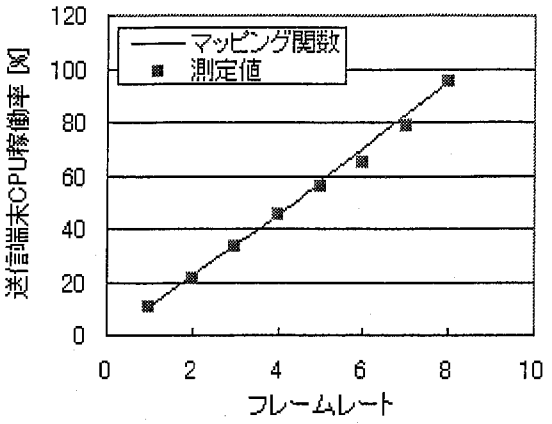


(a) 補間法①　　(b) 補間法②

図 4-9. MPEG-4 方式での測定結果④

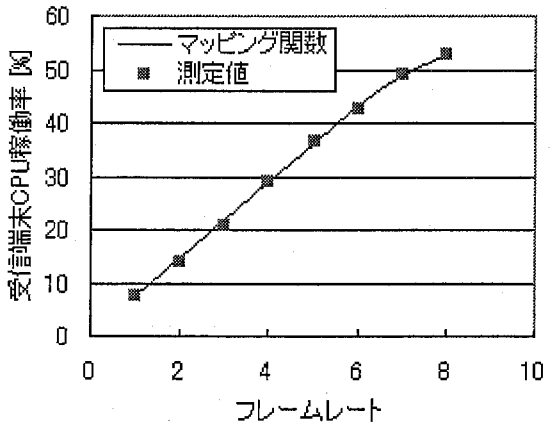


(a) 補間法①

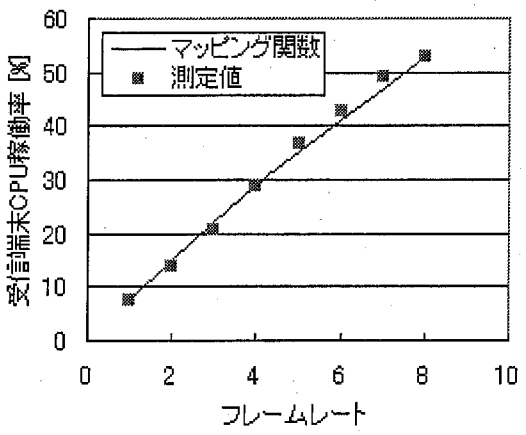


(b) 補間法②

図 4-10. MPEG-4 方式での測定結果⑤



(a) 補間法①



(b) 補間法②

図 4-11. MPEG-4 方式での測定結果⑥

4.4 考察

以上の結果より，アプリケーション QoS と資源 QoS の関係について，M-JPEG 方式および MPEG-4 方式のどちらにおいても，量子化スケールと資源 QoS のマッピング関数は非線形となり，フレームレートと資源 QoS のマッピング関数はほぼ線形となることがわかった．切断べき関数によるスプライン補間法を用いた場合にマッピング関数が振動した原因は，サンプルデータ数が不足したためであると考えられる．同じサンプルデータを用いて *B*-スプラインによるスプライン補間を行った場合，節点を試行錯誤的に変化させることで，マッピング関数の精度を向上させることができた．この性質は，サンプルデータ数が少な

くマッピング関数が非線形となる場合に特に有用である．サンプルデータ点を増加させれば，節点をサンプルデータ点と一致させた切断べき関数でも，B-スプラインと同等の精度を持つマッピング関数を得られる可能性はあるが，後述する QoS マッピング関数の自動計測を行うにあたって，測定精度を向上させるためサンプルデータ数を増加させると必然的にビデオ QoS の測定時間は長くなる．しかしながら測定中はユーザが PC を操作できないため，ユーザの利便性を考慮した場合サンプルデータ数は少なくすべきであり，従って B-スプラインの方がスプライン QoS マッピング関数の構成に適していると考えられる．B-スプラインでは，節点の選び方によってはマッピング精度が悪化する可能性もあり，節点位置は慎重に決定する必要がある．現状では節点位置の決め方に決定的な手法がなく，試行錯誤により決定している．B-スプラインにおけるで節点位置選択の効率化に関して 5.2 節において後述する．

5. QoS マッピング関数の自動計測

5.1 自動計測の必要性

スプライン関数を構成するために必要なサンプルデータを取得するビデオ QoS 測定において，前節の測定方法ではリソースモニタを用いてディスプレイに表示される値を測定者が逐次記録するため，大変な労力と時間を要した．そこでこの測定を自動化し，自動的に QoS マッピング関数を獲得する方法を提案する．提案する自動計測では，マッピング関数は B-スプラインにより構成することとする．B-スプラインを用いれば，少ないサンプルデータ数でも節点を変えることで精度の高いマッピング関数が得られるため，測定時間を短縮することができるためである．

5.2 節点決定方法

B-スプラインにおける節点決定方法の問題を解決するため，節点の位置を最初のサンプルデータだけに対して試行錯誤的に決定し，以後のサンプルデータに対しては固定する方法をとることにした．これは例えば，画像サイズ S 及びフレームレート FR をそれぞれある 1 つの値 S_1, FR_1 とした時の量子化スケールと資源 QoS のマッピングにおいて，最も精度の良いマッピング関数が得られる節点の位置を試行錯誤的に決定し，このようにして決定した節点の位置を，他の画像サイズ(S_2, S_3, S_4...)，フレームレート（FR_2, FR_3, FR_4,...）における量子化スケールと資源 QoS のマッピング関数を構成する際にも適用するという方法で，結果的に試行錯誤をするのは最初の一回のみで良いことになる．本手法を，前述のビデオ QoS 測定実験で測定した M-JPEG 方式での測定データに適用した．

フレームサイズ 160×120 の時の量子化スケール(5, 50, 75, 95)のデータをサンプルデータとし，残りを未知として上記の方法で量子化スケールと資源 QoS のマッピングを行った．まずフレームレートが 1 の時のサンプルデータを用いて，最も精度の良いマッピング関数

が得られる節点の位置を決定した．こうして決定した節点を，フレームレートが 2~19 の場合にも適用しマッピング関数を構成した．サンプルデータ点を除いた測定値に対する平均誤差率を求めると 1.8%であった．比較のため，同じサンプルデータを用い，節点をサンプルデータ点と一致させた切断べき関数での平均誤差率を求めると 13.5%であった．

以上の結果より，非線形性を持つ量子化スケールと資源 QoS のマッピングが，節点を毎回試行錯誤的に決定することなく行えることが確認できた．実際の自動計測では，MPEG-4 方式や M-JPEG 方式といった動画圧縮方式それぞれにおいて，アプリケーション QoS と資源 QoS のマッピング関数を *B*-スプラインにより精度良く構成するための節点をあらかじめ決定しておく必要がある．この作業は人間に頼らざるをえないが，この節点データを持たせておけば，サンプルデータを与えれば自動的にマッピング関数を生成することができる．

5.3 QoS マッピング関数自動計測アルゴリズム

本研究で行う自動計測では，前節のような送信端末と受信端末の通信を行わず，単体の PC で動画像の圧縮及び伸長にかかる QoS マッピングデータの測定を行うことが特徴である．提案では CPU 稼働率ならびに帯域の双方に対する自動測定を対象としているが，本研究では CPU 稼働率のみに対する実装を行っている．以下に実装の概要を示す．

（1）端末は，付属のビデオカメラより取得した動画像を，動画像符号器により所定のアプリケーション QoS で圧縮し，これにかかった CPU 稼動率を記録する．圧縮された動画像は，自身のハードディスクへ格納する．

（2）端末は，ハードディスクに格納してある動画像を動画像復号器により復号し，これにかかった CPU 稼働率を記録する．

（3）アプリケーション QoS を変更し，（1）と（2）を繰り返す．

上記の方法は，ビデオカメラが接続された単体の PC で，動画像の圧縮と伸長にかかるそれぞれの CPU 稼働率を測定する方法である．実際には，（1）で圧縮された動画像をビデオカメラの接続されていない他の PC のハードディスクへ移動し，この PC で伸長にかかる CPU 稼働率のみを測定することも可能である．

（1）～（3）の手順でサンプルデータを測定した後，あらかじめ決定された節点を用いて，*B*-スプラインによるマッピング関数の生成を行う．

6. QoS マッピング関数の自動計測実験

6.1 自動計測プログラムの動作

QoS マッピング関数の自動計測アルゴリズムに基づき，Windows 上で動作する自動計測プログラムを作成した．実験を行ったプログラムでは，ユーザはサンプルデータとなるアプリケーション QoS の各パラメータを入力し，計測プログラムはそのアプリケーション QoS で動画像の圧縮／伸長を行い，これにかかる CPU 稼働率を記録する．CPU 稼働率は

1秒毎に計測され，数秒間の平均値をそのアプリケーションQoSに対するCPU稼働率とする．ただし，動画像の圧縮／伸長の開始直後はCPU稼働率が安定しないため，計測を行わないアイドル時間を設ける．

6.2 実験方法

異なる二種類の実験構成に対して実験を行った．以下にそれぞれの実験構成状況および測定データを補間して得られるQoSマッピング関数の構成方法について述べる．また本実験では，計測を行わないアイドル時間を5秒間とし，CPU稼働率は10秒間の平均値とした．

（1）実験方法①

M-JPEG方式におけるQoSマッピング関数の自動計測を行う．測定にはビデオカメラが接続されたデスクトップPC(OS: Windows NT)を用い，圧縮と伸長にかかるCPU稼働率を測定する．この実験ではマッピング結果と測定値を比較するため，フレームサイズは352×288,264×216,176×144の3点，フレームレートは1～10の10点，量子化スケールは5から95までの間の5ステップ毎の19点，計570点を測定し，これら測定値の中のいくつかをサンプルデータとし，量子化スケールと端末資源QoSを対応づけするマッピング関数をB-スプラインにより生成する．サンプルデータ点と節点は前節のM-JPEG方式のマッピング結果を参考にし，量子化スケールが5, 50, 75, 95の計120点をサンプルデータとし，節点には{5, 5, 5, 5, 75, 94, 95, 95, 95, 95}を用いることとした．

（2）実験方法②

ビデオカメラが接続されたデスクトップPC(OS: Windows NT)では圧縮のみの測定を行い，保存されている圧縮された動画像データをノートPC(OS: Windows NT)に移して，ノートPCで動画像の伸長にかかるCPU稼働率を測定する．測定は，フレームサイズは352×288,264×216,176×144の3点，フレームレートは1～5の5点，量子化スケールは5から95までの間の5ステップ毎の19点，計285点を測定する．サンプルデータ点には量子化スケールが5, 50, 75, 95の計60点をサンプルデータとし，節点には{5, 5, 5, 5, 75, 94, 95, 95, 95, 95}を用いて，量子化スケールと端末資源QoSを対応づけするマッピング関数をB-スプラインにより構成する．

6.3 実験結果

（1）実験方法①の結果

実験方法①での計測結果の一例を図6.1-6.3に示す．これらはそれぞれフレームサイズが352×288，264×216，176×144で，フレームレートが3のときの量子化スケール5～95と端末資源QoSを対応付けするマッピング関数の自動計測結果である．図中の点は自動計測された測定値を示し，線がマッピング関数を示している．図中に表記してある誤差率は，マッピング関数による推定値と測定値との誤差率を示す．

図6.1-6.3の測定結果と，測定者の目視による測定方法でのCPU稼働率の測定結果（図

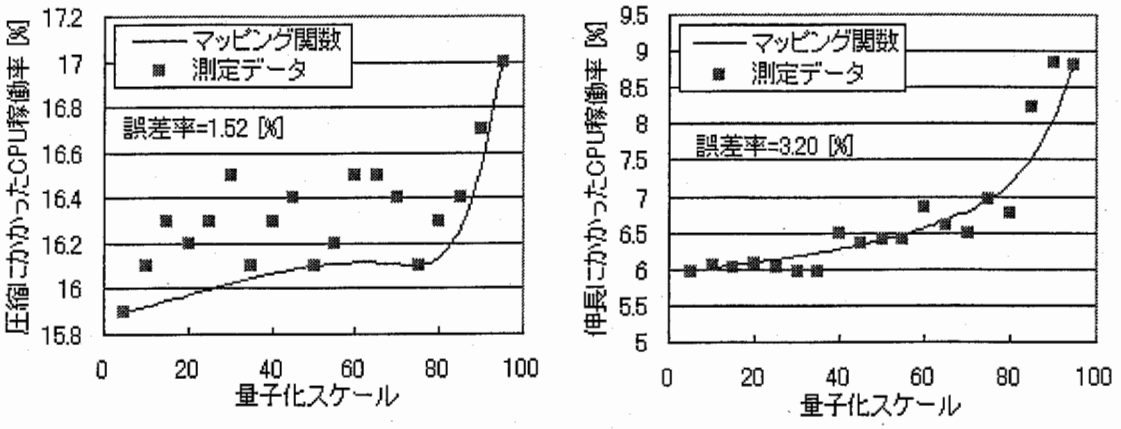
4.2, 4.3）を比較すると，図 6.1(b)の伸長にかかった CPU 稼働率の測定値（フレームサイズ 352×288）を除き，自動計測による測定値には比較的大きなばらつきがあることがわかる．今回行った自動計測では，測定値のばらつきの度合いについて次のような傾向が見られた．伸長にかかる CPU 稼働率の測定値については，フレームサイズが 352×288 の時にはばらつきは小さく，フレームサイズが小さいほどばらつきは大きくなった．また圧縮にかかる CPU 稼働率の測定値については，フレームサイズを変えてもばらつきの度合いは変化せずほぼ一定であった．

図 6.1-6.3 のマッピング関数は，自動計測結果のうち量子化スケールが 5, 50, 75, 95 の点をサンプルデータとし，スプライン補間により得られたマッピング関数である．図 6.1-6.3 では，サンプルデータがばらついている場合にマッピング関数は振動した形となり，測定値との誤差が大きくなっていることがわかる．特に，図 6.3(b)のフレームサイズが 176×144 の場合に伸長にかかる CPU 稼働率のマッピング関数について誤差が大きくなっている．実験方法②で測定した全パラメータのうち，サンプルデータ 120 点以外の測定値を未知のものとして，自動計測されたマッピング関数により CPU 稼働率の推定を行うと，平均誤差率は圧縮時 5.66%，伸長時 224%となった．伸長時の CPU 稼働率推定値で誤差が大きくなった原因は，図 6.3(b)のように，フレームサイズを 176×144 としたときの伸長にかかる CPU 稼働率の計測値が，圧縮時よりもばらついたためである．

（2）実験方法②の結果

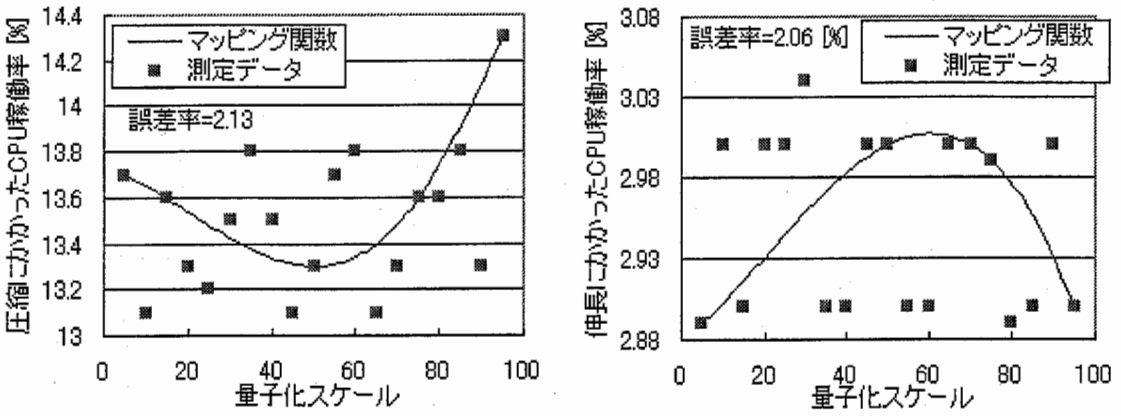
実験方法②での計測結果の一例を図 6.4-6.6 に示す．これらはそれぞれフレームサイズが 352×288，264×216，176×144 で，フレームレートが 3 のときの量子化スケール 5～95 と端末資源 QoS を対応付けするマッピング関数の自動計測結果である．図中の点は自動計測された測定値を示し，線がマッピング関数を示している．図中に表記してある誤差率は，マッピング関数による推定値と自動計測結果との誤差率である．

実験方法②では，デスクトップ PC で測定した圧縮時の CPU 稼働率については実験方法①と同じ程度のばらつきが見られたが，ノート PC で測定した伸長時の CPU 稼働率については，前節の目視による測定法と同等の精度で測定値を得ることができた．サンプルデータ 60 点以外の測定値を未知として，自動計測されたマッピング関数により CPU 稼働率の推定を行うと，平均誤差率は圧縮時 7.02%，伸長時 3.73%であった．



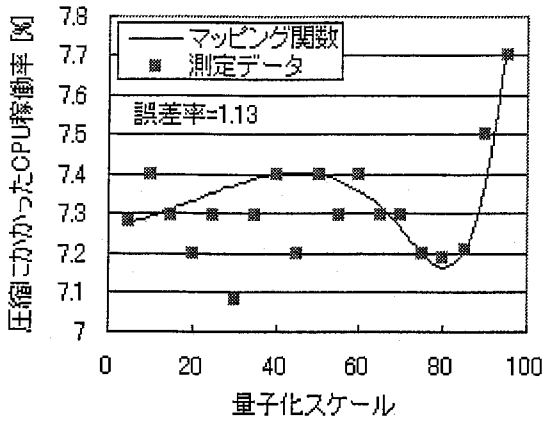
(a) 圧縮　　(b) 伸長

図 6-1. 自動計測結果①（フレームサイズ 352×288, フレームレート 3）

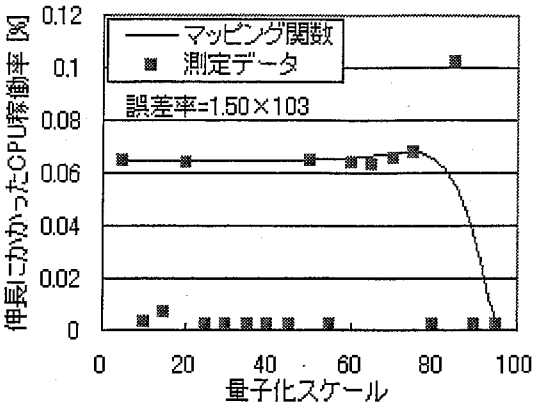


(a) 圧縮　　(b) 伸長

図 6-2. 自動測定結果②（フレームサイズ 264×216, フレームレート 3）

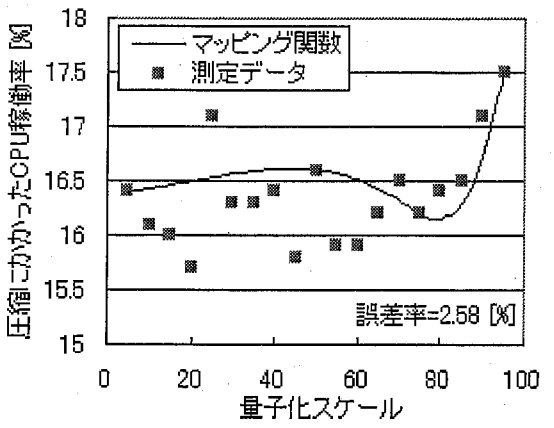


(a) 圧縮

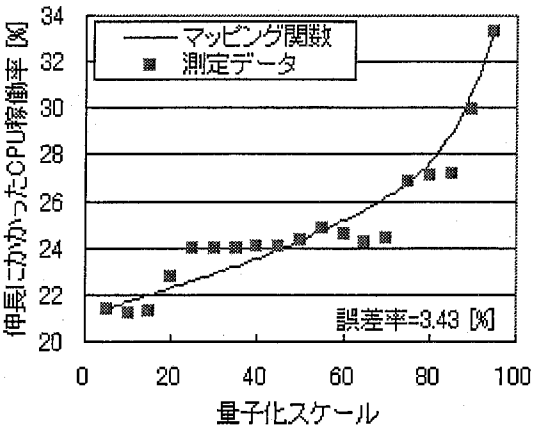


(b) 伸長

図 6-3. 自動測定結果③（フレームサイズ 176×144, フレームレート 3）

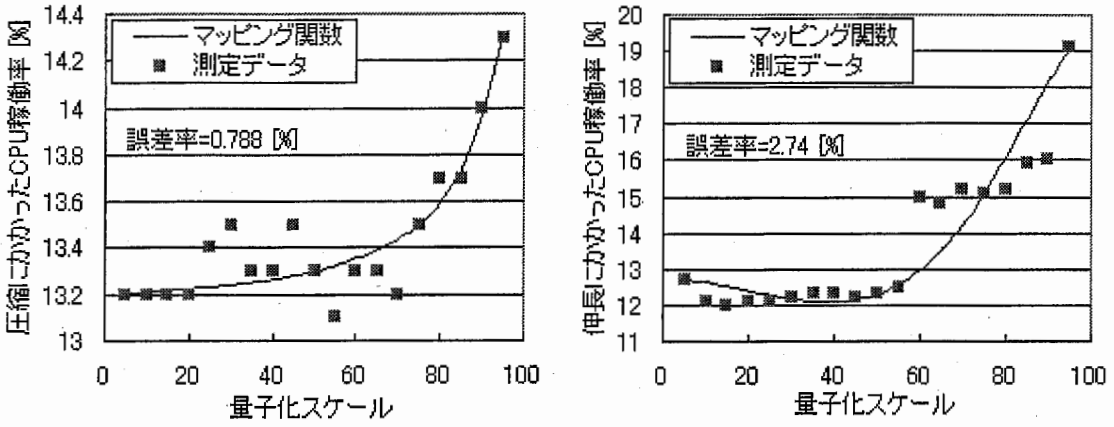


(a) 圧縮



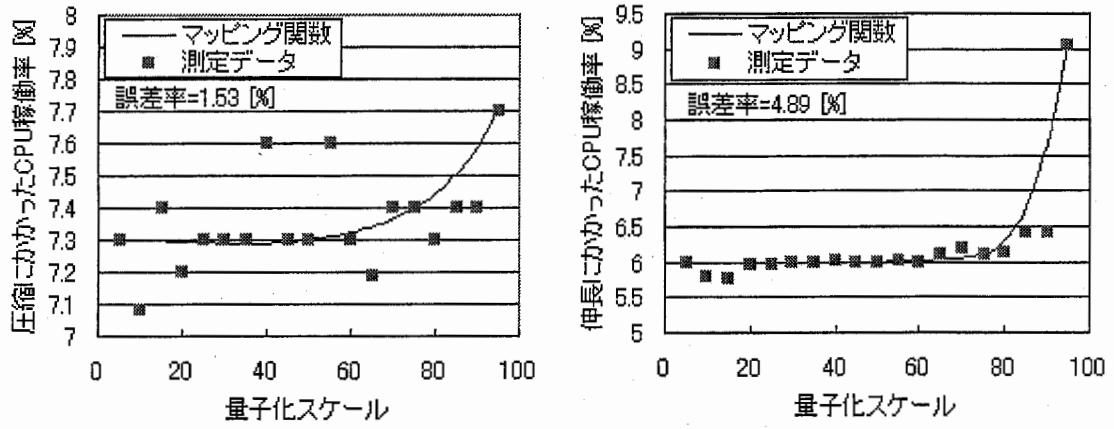
(b) 伸長

図 6-4. 自動計測結果①（フレームサイズ 352×288, フレームレート 3）



(a) 圧縮　　(b) 伸長

図 6-5. 自動測定結果②（フレームサイズ 264×216, フレームレート 3）



(a) 圧縮　　(b) 伸長

図 6-6. 自動測定結果③（フレームサイズ 176×144, フレームレート 3）

6.4 考察

本実験では，実験方法①で伸長時の CPU 稼働率推定での誤差率が 224%と非常に大きくなったのを除けば，3~7%の誤差率で推定を行うことができた．誤差が大きくなった原因は，自動計測による CPU 稼働率の測定値がばらついたため，*B*-スプラインによるマッピング関数の構成がうまく行えなかったことにある．今回行った自動計測では，実験方法②で行ったノート PC での CPU 稼働率にはばらつきがあまり見られず，デスクトップ PC で測定した CPU 稼働率にのみばらつきが見られた．そこで，デスクトップ PC での 1 秒毎の CPU 稼働率の計測値を調べてみたところ，動画像の圧縮および伸長を行っている最中に，CPU 稼働率が 10~20 秒間隔で最大 5%ほど変動していることがわかった．CPU 稼働率が変動する要因としては，他のアプリケーションが CPU に負荷を与えるケースと，フレームレート

が変動するケースの二つが挙げられる．前者のケースに関しては，使用したデスクトップPC上で，動画像の圧縮および伸長を行っていない状態ではCPU稼働率はほぼ0%で変動が見られなかったことから，他のアプリケーションがCPUに負荷を与えている可能性はほとんどないものと考えられる．後者のケースのような，圧縮および伸長を行っている最中にフレームレートが変動しCPU稼働率に影響を与えることは，今回の実験では十分にありうるものと考えられる．例えば，利用可能なCPUリソースの限界に近づいた場合には，指定したフレームレートを実現できず，フレームレートが変動することがある．しかし，ノートPCではCPU稼働率の変動は起こらないことから，デスクトップPCでのみフレームレートの変動が起きるかどうかは疑問の残る点である．

また実験方法②において，ノートPCで動画像の伸長にかかるCPU稼働率を自動計測した場合の測定値は，実時間で動画像伝送した場合の測定者による目視による測定法での測定値よりも，全体的に10%ほど小さくなることが確認された．またデスクトップPCでのCPU稼働率の測定値には，このような差異は認められなかった．この原因としては，自動計測では送受信端末での動画像の伝送を行っていないため，伝送に必要なCPUリソースだけCPU稼働率が減少した可能性が考えられる．ノートPCでのみ伝送によるCPU稼働率の差異が確認できた理由は，実験に使用したノートPCがデスクトップPCに比べ利用可能なCPUリソース量が小さかったために，動画像の伝送にかかるCPU稼働率が目に見える形で確認できたのではないかと考える．このようなことから，本研究で提案した自動計測方法では動画像の伝送に必要となるCPUリソースが測定できないため，CPUリソースの小さなPCでは，実際に動画像の伝送を行った場合でのCPU稼働率と自動計測されたマッピング関数による推定値との間に誤差が生じてしまう可能性がある．

7. まとめ

本レポートでは動画像メディアのQoSを対象とし，スプライン関数によるQoSマッピング手法の提案を行った．提案法の評価実験では，QoSマッピングの基礎データとなるビデオQoS測定を行い，この測定データを用いてスプライン関数によるQoSマッピング手法を評価した．続いてビデオQoS測定からQoSマッピング関数の生成までを自動化する手法の提案を行い，マッピング関数の自動計測実験を行った．

スプライン関数によるQoSマッピング手法の評価実験では，B-スプラインによる補間法を用いた場合，節点を変化させることで理想的なマッピング関数が得られた．しかし節点の決定方法には理論的な方法がなく試行錯誤的な判断に頼らざるを得ない．QoSマッピング関数の自動計測では，あらかじめ1回だけ試行錯誤により決定した節点位置を継続的に用いることで効率化を図った．

QoSマッピング関数の自動計測実験では，ビデオQoS測定での測定値にばらつきが生じ，B-スプラインによるマッピング関数の構成がうまく行えなかった場合もあった．今後QoS

マッピングの精度をより向上させるためには，ビデオ QoS 測定の測定値のばらつきを抑え，精度を向上させることが，適応的 QoS 制御アプリケーションへ実装するためにも重要な課題であると考える．

謝辞

本研究を遂行するにあたり、機会を与えて下さった，㈱エイ・ティ・アール環境適応通信研究所小宮山牧兒社長，蓮池和夫第一研究室室長，および様々な場面でご支援下さった第一研究室のみなさまに感謝致します．また第一著者は実習期間中ご指導頂いた第二著者に深謝致します．

参考文献

[1] 山崎達也, "QoS マッピングに関する考察(データ FD 付)", ATR テクニカルレポート TR-AC-0032, ㈱エイ・ティ・アール環境適応通信研究所, 1999. 6. 9.
[2] 市田浩三, 吉本富士市 著, "スプライン関数とその応用", 教育出版株式会社, 1994.
[3] 桜井明, 石井好, 吉村和美, 高山文雄 著, "スプライン関数入門" 東京電機大学出版局, 1981.
[4] 桜井明, 吉村和美, 高山文雄 著, "パソコンによるスプライン関数", 東京電機大学出版局, 1988.