高品位日本語文字高速出力方式†

山足公也†† 三浦修一†† 川端 敦††
瀧 勇次††† 谷藤真也††

高品位な日本語文字を出力するには，日本語アウトラインフォントが有効である．しかし，このアウトラインフォントの描画には，従来のドットフォントに比べ，文字展開に処理時間を要する．そこで，アウトラインフォント出力の高速化のため，日本語に適したフォントキャッシュについて検討した．特に，本報告では，フォントキャッシュの性能に深い関わりを持つ，高速登録文字検索方式，文字登録領域のメモリ管理方式について検討し，(1)書体属性・漢字コードをキーとする3段テーブル検索による文字検索，(2)文字登録領域のブロック化による高速メモリ管理，の2つの特徴を備えるフォントキャッシュアーキテクチャを提案した．このフォントキャッシュを実際のプリンタシステムに適応し，評価した結果，日本語文書の場合，フォントキャッシュへのヒット率が高く，有効に動作することを確認した．また，このフォントキャッシュを用いることにより，用いない場合に比べ，約40倍の性能向上を実現した．

1. まえがき

近年，ワークステーションや小型レーザビームプリンタの高性能化・低価格化が進んでいる．それに伴って，社内印刷等をワークステーションを用いて行う卓上出版（DTP：Desk Top Publishing）が急速に普及し始めている．このDTPは，ワークステーション上で，各種のサイズや字体（フォント）の文字を多用したテキストに，図面や写真（イメージ）を貼り付け，高品位な文書を作成するシステムである．

このようなマルチメディア文書を作成するシステムにおいて問題となるもののひとつに出力文書の品質がある．従来，ワープロなどでは，文字をイメージパターンで記憶するドットフォントを用いている．マルチメディア文書では，多種類の文字を使用するため，このドットフォントを拡大・縮小しなければならない．しかし，ドットフォントの拡大・縮小を行うと文字の輪郭がぎざぎざになったり，文字が潰れてしまい，文字の描画品質が著しく劣化してしまう．

それに対して，近年，文字を図形として扱い，文字の輪郭を座標の形で記憶するアウトラインフォントが注目されている．このアウトラインフォント方式による文字描画は，まず，所望の大きさにするため輪郭座標に座標変換を施し，その内部を塗りつぶすことにより行う．この方式では，文字情報を座標の形で表現しているため，文字の拡大・縮小を行いやすく，また，サイズ変換による描画品質の劣化も少ない．しかし一方で，前述のドットフォント形式に比べ描画速度が遅いという欠点がある．そこで，プリンタシステムにアウトラインフォントを応用するには，その処理を高速化する必要がある．本稿では，このアウトラインフォントの高速文字展開方式について検討する．

2. アウトラインフォントによる文字出力と検討課題

2.1 アウトラインフォントの文字品質

前述のようにアウトラインフォントはベクトル文字であり，**図1**に示したように拡大・縮小しても文字の品質の劣化は起こらない．また，文字へ各種のアフィン変換を施すことにより，図のような斜体も簡単に発生することができる．

しかし，アウトラインフォントで文字を出力する場合にも，小さな文字を出力すると文字が潰れてしまうという問題がある．つまり，アウトラインフォントを限られたメッシュにマッピングすることになり，量子化誤差が発生する．この量子化誤差により文字が潰れたり，本来同じ太さの線が異なる太さになったりし，文字品質が劣化してしまう．

アウトラインフォントの出力品質は，48ドット程度でワープロ並み，64ドット程度の大きさから高品位な文字出力を行うことができる．したがって，小さな文字を高品位に出力するためにドットフォントを併用する場合も多い．

† Fast Printing Method for High Quality Japanese Characters by KIMIYA YAMAASHI, SHUICHI MIURA, ATSUSHI KAWABATA (The 3rd Department, Hitachi Research Laboratory, Hitachi, Ltd.), YUJI TAKI (Semiconductor Design & Development Center, Hitachi, Ltd.) and SHIN-YA TANIFUJI (The 3rd Department, Hitachi Research Laboratory, Hitachi, Ltd.).

†† (株)日立製作所日立研究所第3部
††† (株)日立製作所半導体設計部開発センタ

株式会社日立製作所日立研究所
株式会社日立製作所日立研究所
株式会社日立製
株式会社日立製作所日立
株式会社日立製作所日立
株式会社日立製作所日立
株式会社日立製作所日立

図1 プリント例
Fig. 1 Printing examples.

2.2 フォントキャッシュ出力方式

アウトラインフォントの高速化手法のひとつにフォントキャッシュ方式がある．フォントキャッシュ方式とは，一度作成したアウトラインフォントのドットイメージをフォントキャッシュに登録しておき，再び同じ文字が使用される場合には，登録されているドットイメージを出力する．これにより，アウトラインフォントをドットイメージへ展開する時間を省き，高速な文字出力を可能にする手法である．その概念を図2に示す．この図のようにフォントキャッシュ内には，サイズや書体の異なるドットイメージが多数格納されるイメージ登録領域とそのドットイメージの格納場所を管理するキャッシュ管理部がある．そして，フォントキャッシュ内の文字を出力する際には，このキャッシュ管理部からそのフォントデータの格納場所を検索し，その文字を出力する．

2.3 フォントキャッシュの検討課題

（1） 登録文字検索法

文字を出力する際には，その文字のドットイメージがフォントキャッシュ内にあるかどうか，また，あるとしたら，イメージ登録領域のどこにあるかを検索した後，そのドットイメージを出力する．しかし，登録文字が，図2のように順不同に登録されている場合には，管理テーブルを1つ1つ順番に検索しなければならず，フォントキャッシュの性能を低下させてしまう．したがって，高速なフォントキャッシュを実現するには，この検索を効率的に行う必要がある．

（2） イメージ登録領域メモリ管理法

フォントキャッシュに文字を次々と登録していくと，登録領域が一杯になってしまう．すると新しい文字を登録するため，フォントキャッシュに登録されている文字を削除する必要がある．フォントキャッシュには，各種のサイズの文字が登録されている可能性があるため，単純に文字の登録・削除を繰り返すと，文字のドットイメージを記憶するイメージ登録領域の細分化（フラグメンテーション）が発生する．このような状態になると，新しい文字の登録領域を確保するため，図2に示すように，イメージ登録領域の未使用領域を1つにまとめるコンパクションを行う必要がある．このコンパクション操作は，使用中の登録領域ならびに空き領域をそれぞれ1つの連続領域に再配置する．そのため，処理時間がかかり，キャッシュの性能低下を招く．

このフラグメンテーションを避けるため，各文字サイズごとに登録領域をあらかじめ割り当てておく手法[3]が提案されている．しかし，この場合，10ポイントの文字だけを用いる文書を出力する場合でも，その他のサイズの文字登録領域をフォントキャッシュ内に確保することになる．すなわち，文字を登録していくうちに，10 ポイントの文字を登録する領域がなくなれ

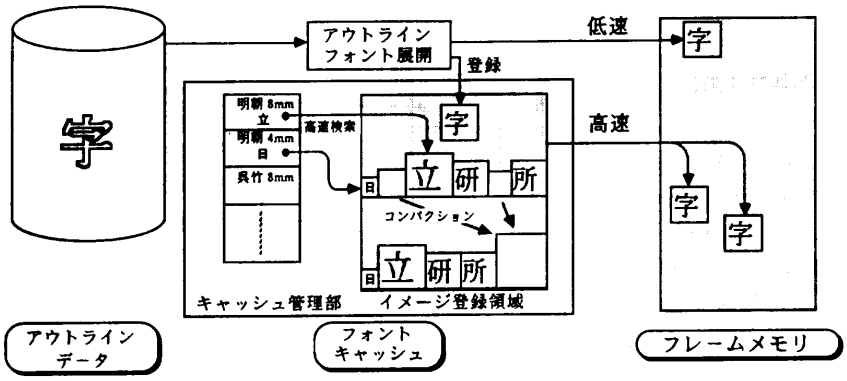


図2 フォントキャッシュ
Fig. 2 Font cache.

ば，イメージ登録領域内に空きエリアがあっても，10ポイントの登録文字を削除しなければならなくなる．これでは，フォントキャッシュのメモリ使用効率が悪く，また，無駄な文字削除により，フォントキャッシュの性能が落ちてしまう．したがって，各種の文字を効率良く登録でき，コンパクション操作の伴わない高速なメモリ管理方式が求められる．

（3） 削除文字選択法

さらに，文字削除を行う際には，どの文字を削除するかを選択しなければならない．しかし，文書中で用いる文字には，ある文字は良く使用し，また他の文字はあまり使用しないなど，その使用率に偏りがある．それを単純に削除したのでは，使用率の高い文字が削除され，使用率の低い文字ばかりが残り，フォントキャッシュから文字を出力する確率が低下してしまう．つまり，文字削除対象を決定するアルゴリズムはフォントキャッシュの性能に対して重要であり，それについて検討する必要がある．

（4） ドットフォントとの併用法

高品位化のため小さな文字を出力する際には，ドットフォントを併用する場合がある．しかし，単純にドットフォントとアウトラインフォントの2つのフォントを独立に管理したのでは，文字を出力するたびにドットフォントで描画するのか，アウトラインフォントで描画するのかを調べなければならず，文字出力性能の低下につながる．そのため，アウトラインフォントとドットフォントとを併用し，効率良くフォントを切り換える機能を設ける必要がある．

3. フォントキャッシュの構成と管理法

本章では，2.3 節で述べたフォントキャッシュの検討課題に基づき，高速なアウトラインフォント出力方式を提案する．

3.1 登録文字の高速検索方式

前述のようにフォントキャッシュの中には，書体属性（書体やサイズ）の異なる多数の文字が格納される．したがって，文字を検索する場合，その文字コードのほかに，書体属性をキーにして文字を検索することになる．例えば，図2の方式では，1つの管理テーブル内に書体属性の異なる文字の検索キーが格納されるので，1文字ごとにこれらのキーをすべてチェックすることになる．

しかし，多くの文書では各文字ごとに書体属性を変更することはなく，文字列単位で書体属性を変更する．同一の文字列の中では，先頭文字以外では書体属性がその前と変わらないのだから，先頭文字以外では，文字コードだけをキーとして検索できることが望ましい．言い換えると，キャッシュ管理部を，文字列に共通な書体属性を管理する部分と，各文字固有の文字コードを管理する部分とで構成するようにしたほうがより効率的な検索が可能となる．すなわち，書体属性を管理する書体管理テーブルと文字コードをインデックスとしてイメージ登録領域内の登録文字を指し示すコード管理テーブルとでキャッシュ管理部を構成する．これにより，書体属性をキーとする書体管理テーブルの検索が，文字列の最初の文字だけで済み，その後の文字に関しては，文字コードをキーとしてコード管理テーブルを検索するだけでよい．

ただ，この方式では，日本語のように1つの書体の文字数が非常に多い場合には，コード管理テーブルが大きくなってしまう恐れがある．しかも，その全エントリのうちフォントキャッシュに登録されている文字のエントリしか使用されず，メモリの使用効率が悪い．

この問題を解決するには，コード管理テーブルを**図3**に示すように，日本語コード上位・下位バイトに対応した2段のテーブルで構成する方式が有効である．この際，登録文字のない2段目のコード管理テーブルについては，領域を確保せず，必要になったときにその領域を確保するようにする．これにより，不要なコード管理テーブルを確保しないため，日本語のよ

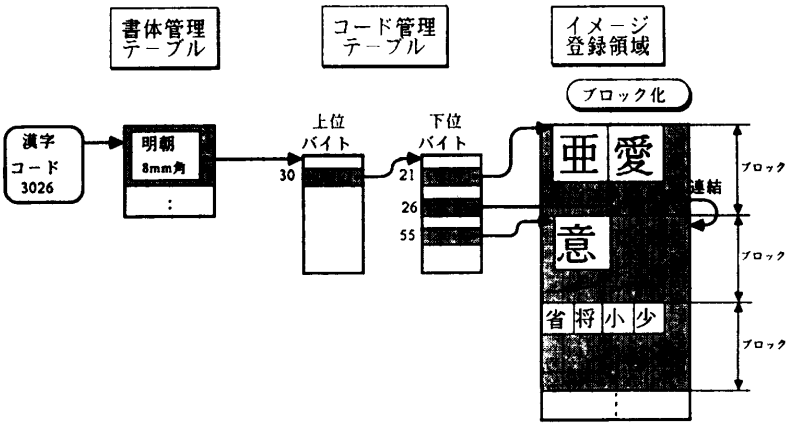


図3 フォントキャッシュの構成
Fig. 3 Structure of font cache.

うな文字種の多い場合にも，メモリ使用効率の良いテーブル管理が行えるようになる．

3.2 イメージ登録領域の高速メモリ管理方式

（1） イメージ登録領域のブロック化

イメージ登録領域内のフラグメンテーションが発生する原因は，異なる書体属性の文字を1つの領域に混在させることにある．この問題を解決するため，ここでは，イメージ登録領域を一定の大きさでブロック化し，同一のブロックには同一の書体属性をもつ文字だけを登録する方式を提案する．ブロック化方式では，ブロック単位で文字削除や領域確保を行うことになるので，イメージ登録領域内に小さな空き領域は発生しない．このため，イメージ登録領域のコンパクション操作が不要となり，高速なメモリ管理が期待できる．

（2） 最適ブロックサイズ

前述のように，イメージ登録領域は満杯になったときには，ブロック単位に文字データを削除し，そこを新しい登録領域とする．ブロックサイズをあまり大きくとると，ブロック削除のとき，たくさんの文字データが削除されることになり，キャッシュの効率が低下する．したがって，ブロックのサイズはできるだけ小さいほうがよい．しかし，あまり小さくすると，大きな文字が1つのブロックに納まりきらなくなる．ここでは，登録文字の管理を簡単にするため，1つの文字を複数のブロックにまたがって格納しないことにする．このことから，イメージ登録ブロックの最小ブロックサイズは，おのずと最大サイズの文字を1文字登録できる大きさになる．

この場合，当然この領域内に登録できる文字数には限りがあり，ブロックが満杯になってしまう可能性がある．領域が足りなくなった場合には，図3に示すようにブロックをリスト形式で連結し，領域を拡張する．このように，ブロックをリストでつなぐという方式を用いることにより，イメージ登録ブロックのブロックサイズをフォントキャッシュに登録する最大文字サイズにしても問題は起こらない．

フォントキャッシュに登録する最大文字の大きさは，フォントキャッシュを適応するシステムによって異なる．しかし，普通の文書では，10-12 ポイントの全角文字を多用し，ワープロ等ではせいぜいその4倍角文字までしか用いない．また，大きな文字は，見出し用にしか使用せず，フォントキャッシュに登録しておいても，再利用される可能性は低い．そこで，ここでは，イメージ登録ブロックサイズとして，40 ポイント程度までの文字が登録できる大きさにするのが適当であると考えた．

（3） 登録文字管理法

文字を削除する場合，イメージ登録領域からデータを消すだけでなく，キャッシュ管理部のデータも変更する必要がある．ここでは，検索テーブルとイメージ登録ブロックとの最適な組合せという観点から，文字削除の問題についてさらに検討する．

1つのイメージ登録ブロックに格納する文字の単位として，図3の検索テーブルの構成上，1文字ごと，書体ごと，下位バイト管理テーブル（コード管理テーブルの2段目のテーブル）ごとの3つの場合が考えられる．まず，1文字ごとの場合には，最も多く使用される文字サイズが，イメージ登録ブロックサイズ（フォントキャッシュに登録する文字の最大サイズ）よりも小さいため，イメージ登録ブロック内に未使用領域が多く発生し，メモリの使用効率が悪い．また，書体ごとの場合には，日本語のような1書体あたりの文字数が多い言語に対して，キャッシュの効率が低下する．つまり，キャッシュが満杯のとき，新しい文字を登録するため，1書体分の文字（数千文字）が一度に削除され，文字削除による損失が大きい．それに対して，下位バイト管理テーブルごとを基準とする方式は，キャッシュが満杯のときに削除される文字数が，このテーブルに登録される最大256文字に限定される．したがって，書体ごとの場合に比べ，文字の再利用が効率的に行える．

また，下位バイト管理テーブルに登録される文字数が多くなると，1つのイメージ登録ブロックでは納まりきらなくなる可能性がある．その場合には，前述のように，未使用のイメージ登録ブロックをリストを用いて連結し，その領域を拡張する．

3.3 削除文字の選択方法

文字削除を行う場合には，削除する下位バイト管理テーブルを検索しなければならない．そのアルゴリズムとして，従来のディスクキャッシュのキャッシュアルゴリズムにおいて一般的な LRU 法 (Least Recently Used Method) を採用した．

LRU 法による文字削除は，以下のようにして実現できる．つまり，フォントキャッシュ内の登録文字を使用する際に，その使用時刻を下位バイト管理テーブルに記録する．そして，文字削除を行う際には，その下位バイト管理テーブル内の使用時刻を調べ，その中で，最も古い下位バイト管理テーブル，および，それ

につながるイメージ登録ブロックを削除対象とする．

3.4 ドットフォントとアウトラインフォントとの併用法

フォントキャッシュに登録されている文字は，元々はアウトラインフォントであるが，実質，ドットイメージである．すなわち，このフォントキャッシュに登録されている文字はドットフォントとみなすことができる．そこで，フォントキャッシュのドットイメージ管理機構を用いてドットフォントの管理を行うことを検討する．

ドットフォントをフォントキャッシュを用いて管理するには，ドットフォントをまず，フォントキャッシュに登録する．すなわち，システム立ち上げ時にそのドットフォントに相当する書体属性を書体管理テーブルに登録し，ROM などにある各ドットフォント文字へのポイントをコード管理テーブルに登録する．次に，ドットフォント文字を出力する際には，アウトラインフォント文字の場合と同じように書体属性と文字コードをキーとしてフォントキャッシュを検索し，対応するドットイメージを描画する．

ドットフォントの場合，ドットイメージそのものは，フォントキャッシュ内ではなく，別の領域に格納されているため，フォントキャッシュの空きエリア管理の対象にならない．しかし，コード管理テーブルには，削除対象となるアウトラインフォントのテーブルとそうでないドットフォントのテーブルが混在する．したがって，文字削除時にその下位バイト管理テーブルがドットフォントのものなのか，アウトラインフォントのものなのかを判別する必要がある．

文字削除の際には，前述のように下位バイト管理テーブルに記録された使用時刻を参照して，その削除対象を決定する．そこで，ドットフォントの下位バイト管理テーブルの使用時刻を常に特別な値（0）にしておく．これにより，文字削除を行う際に，ドットフォントのテーブルかどうかを見分けることができ，ドットフォントに対する文字削除を禁止することができる．

以上により，フォントキャッシュの機構をほとんど変えることなくドットフォントを組み込むことができる．また，文字を出力する際にも，ドットフォント/アウトラインフォントの区別を意識せず，高速にドットフォントとアウトラインフォントとを切り換えて文字を出力することができる．

4. フォントキャッシュの性能評価

4.1 プリンタシステム構成

本フォントキャッシュを実際のプリンタシステムに適応した場合の性能について述べる．評価システムは，CPU (68020…16 MHz) と数値演算プロセッサ (68881) で構成した．また，出力装置として 400 DPI のレーザビームプリンタを使用し，フレームメモリはA3 対応のため 3.6 M バイトとした．また，フォントキャッシュの総容量は，約 600 K バイトにした．そのシステム構成を**図 4** に示す．このプリンタでは，図のように文書を記述したデータがネットワークを通して送られてくると，まずインタプリタ部で字句解釈し，それぞれの描画制御部を起動する．文字描画の場合には，文字描画制御部が起動される．この文字描画制御部は文字列から順次文字コードを取り出し，その文字コードと書体情報をフォントキャッシュに送り，文字描画を行う．なお，本評価システムでは，文字フォントとして，直線群で構成されるアウトラインフォントデータを使用している．

4.2 ヒット率評価

上記プリンタシステムでのフォントキャッシュの有効性についての実際の文書を用いて検討する．ここではフォントキャッシュの有効性の指標としてヒット率を用いる．ヒット率とは出力文字がフォントキャッシュに登録されている割合であり，次のように定義する．

ヒット率＝(キャッシュに登録されている文字数)/出力文字数

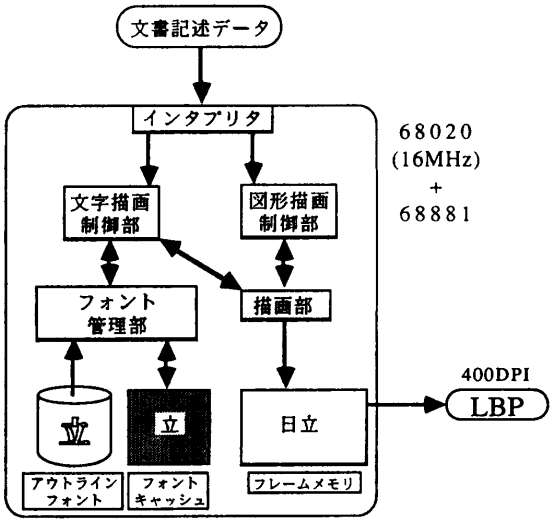


図 4 システム構成
Fig. 4 Printer control part.

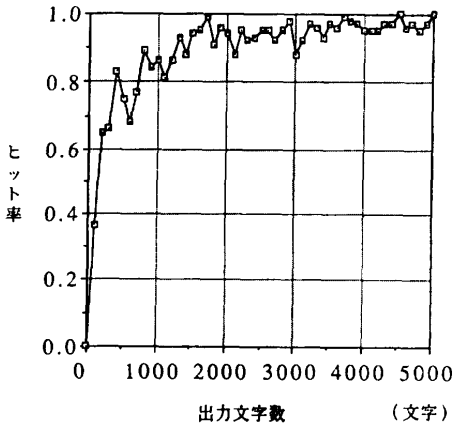


図 5 ヒット率評価
Fig. 5 Evaluation of hit ratio for font cache.

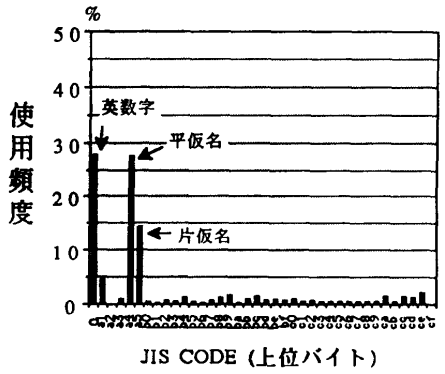


図 6 文字種ごとの使用頻度
Fig. 6 Appearance frequency of characters.

ここで，フォントキャッシュの各パラメータ（書体管理テーブル数・コード制御テーブル数・イメージ登録ブロック数）は無限にあるとした．また，評価用文書として文字数が 17,758 文字ある研究論文を用いた．この評価では，フォントキャッシュの動作を把握しやすいように，文書内の文字は，一定の書体・サイズの文書であるとした．**図 5** にその評価結果を示す．この図では，ヒット率をその文字数の前 100 文字の中で，フォントキャッシュに登録されていた割合で計算している．

この図より，フォントキャッシュへのヒット率が急俊に立ち上がっていることがわかる．特に，出力文字数が 1,500 文字程度でフォントキャッシュへのヒット率が 90% を越える．これは，日本語の場合に対して，フォントキャッシュが有効に動作することを示している．また，上記の評価用文書のほかにいくつかの文書を評価した場合でも同じ傾向を得た．

次に，フォントキャッシュへのヒット率が急激に上昇する理由について考察する．まず，評価用文書中に使用されている文字種数は 657 種であった．つまり，出力文字数に比べ，登録されている文字数が非常に少ないことが分かる．また，**図 6** にこの文書に現れる文字種（文字コード (JIS CODE) の上位バイト）に対する文字の使用頻度を示す．この図より，日本語の文書において，その使用文字の種類には，かなり偏りがあることが分かる．特に，英数字，仮名の使用率が非常に高く全体の 70〜80% にも及んでいる．これは，日本語の場合，助詞や送り仮名の使用率が高く，また技術文書においては，頻繁に英数字が用いられていることに対応している．一方，文字コード d0 以上のものは，第二水準の漢字であり，使用される確率は低い．

以上のように，1 つの文書内に用いられる文字種が比較的少なく，また，文字の使用確率に偏りがあるため，日本語のような文字種の多い言語に対しても，フォントキャッシュは有効に動作する．

4.3 システム性能

上記のプリンタシステムにおいて，実システムでの描画性能を**図 7** に示す．評価用文書として，前述の文書を用いた．フォントキャッシュを用いない場合には，文字の複雑さ等により若干の変動はあるが，1 文字平均 60 msec の時間がかかる．一方フォントキャッシュを用いた場合には，1 ページ目では，まだ，フォントキャッシュ内に文字が登録されていない処理時間がかかる．しかし，出力を行うにつれフォントキャッシュ内の登録文字が増加し，フォントキャッシュへのヒット率が上昇するため，図のように文字の出力速度が急激に速くなる．最終的にほぼ文字出力速度が一定

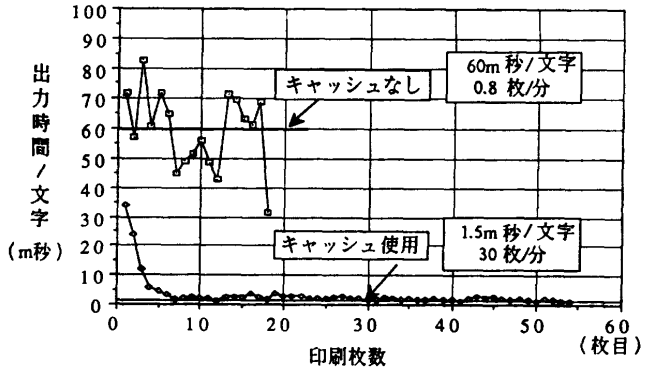


図 7 プリンタシステム性能
Fig. 7 Transition of character drawing speed.

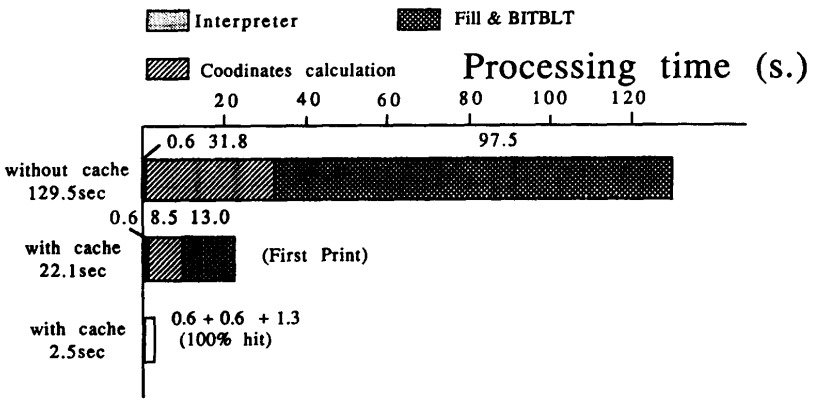


図 8 描画処理時間分析
Fig. 8 Details of drawing time.

になったときの平均出力時間は，1.5 msec であり，フォントキャッシュを用いることにより，40 倍の性能向上を実現している．

また，**図 8** に評価用文書のあるページ（10 ポイント，1,721 文字）の描画時間の内訳を示す．この図は，フォントキャッシュを用いなかった場合，フォントキャッシュに文字が登録されていない状態からこのページを出力した場合（ファーストプリント），100% フォントキャッシュにヒットした場合の 3 つの場合の内訳を示している．また，この図では，処理時間を大きく 3 つに分類した．まず，文書データを字句解釈し各制御部を駆動させるインタプリタ時間，次にアウトラインフォントの座標データを所定の大きさに座標変換する座標変換時間，さらに，実際の文字を描画する描画時間である．ここで，描画方法は，スキャンライン法を用いている．

このグラフより，フォントキャッシュを用いることにより，ファーストプリントの場合でも，フォントキャッシュを用いない場合に比べ 6 倍の性能が向上し，100% ヒットした場合には，さらに約 9 倍の性能が向上することが分かる．実際のプリンタ性能は，ファーストプリント時と 100% ヒット時の間の値をとるが，図 5 から数枚の文書を出力した後では，ヒット率が 90% を越え，キャッシュを用いない場合に比べ，前述のように 40 倍程度の性能が出る．また，ファーストプリント時と 100% ヒット時の差はアウトラインフォントを展開する時間であり，さらに高速な文字描画を行うには，アウトラインフォントの展開時間も高速化する必要がある．

グラフより，アウトラインフォントを描画する際には，塗りつぶし時間が全体の 70% 以上の処理時間を占めている．また，アウトラインフォントを所定の大きさに座標変換する時間が全体の 1/4 を占め，無視できないことが分かる．

この評価システムでは，アウトラインフォントが直線で構成されているが，ベジエ曲線などの 2 次，3 次曲線で構成される場合には，この座標計算時間の占める割合がさらに増えると予想される．したがって，キャッシュヒット率の低い初期段階の表示性能を向上するには，

（1） 塗りつぶし処理の高速化

（2） 高次補間曲線の展開および座標変換の高速化

の 2 点について，そのアルゴリズムの高速化を検討する必要がある．

5. むすび

卓上出版システムにおける高品位日本語文字出力方式について検討した．まず，高品位な文字出力を行うためアウトラインフォントを用いた．さらにこのアウトラインフォントを高速に出力するため日本語に適したフォントキャッシュの構成を提案した．これにより，フォントキャッシュを用いない場合に比べ，実際のシステム性能として 40 倍の性能向上を実現した．

今後，さらに高速なアウトラインフォント描画を行うため，塗りつぶし，座標変換の高速化等を検討する．また，開発したフォントキャッシュをプリンタだけではなく，ディスプレイ等の出力装置へも応用していきたい．その場合，2 章で述べた小さな文字の展開技術が問題となると考えられ，ヒンティング技術やグレイスケール文字などの描画手法について検討していく必要がある．

参考文献

1) Karow, P.: *Digital Formats for Typefaces*, URW Verlag, Hamburg (1987).
2) Knuth, D.E.: *The METAFONT Book*, Addison-Wesley, MA (1986).
3) 美馬ほか：フォントバッファに関する検討，第35回情報処理学会全国大会論文集，p. 2501 (1987).
4) 山足ほか：高速・高品位日本語文字出力方式の検討，第37回情報処理学会全国大会論文集，p. 1909 (1988).
5) Foley, J.D. and Van Dam, A.: *Fundamentals of Interactive Computer Graphics*, Addison-Wesley, MA (1982).
6) Newman, W.M. and Sproull, R.F.: *Principles of Interactive Computer Graphics*, McGraw-Hill, New York (1979).

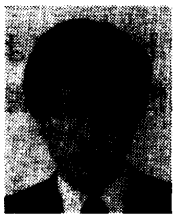
(平成元年3月15日受付)
(平成元年9月12日採録)

山足 公也（正会員）

昭和36年生．昭和59年京都大学工学部電気工学第2学科卒業．昭和61年同大学大学院修士課程修了．同年(株)日立製作所入社．現在，日立研究所において，ワークステーションに関する図形描画技術・マンマシーン技術の研究開発に従事．

三浦 修一

昭和58年京都大学大学院工学研究科修士課程修了．同年(株)日立製作所入社，日立研究所に勤務し，リアルタイム画像処理装置の研究開発に従事．現在，ワークステーションにおけるプリンタ制御技術の研究開発に従事．電子情報通信学会会員．



川端 敦

昭和32年生．昭和56年東京大学工学部計数工学科卒業．昭和58年同大学大学院修士課程修了．同年(株)日立製作所入社．現在，日立研究所において生体情報工学，分散処理システム，マン・マシンインタフェースの研究に従事．電子情報通信学会，計測自動制御学会各会員．

瀧 勇次

昭和39年生．昭和62年宮崎大学工学部応用物理学科卒業．同年(株)日立製作所入社．以来，日立研究所にて，文書・文字処理，グラフィクスの応用に関するソフトウェアの研究・開発に従事．現在，半導体設計部開発センタにて，ゲートアレイ DA システムに関する研究開発に従事．

谷藤 真也（正会員）

昭和22年生．昭和48年早稲田大学理工学研究科修了．同年(株)日立製作所日立研究所入社．鉄鋼プロセスの計算機制御，知識工学応用システムの研究に従事．現在はエンジニアリングワークステーションのソフトウェアおよびユーザインタフェースの研究を担当．計測自動制御学会，電気学会などの会員．