簡易ベクターグラフィックスのテキストへの埋め込み手法

Embedding Simple Vector Graphics in Plain Text

中嶋 海介　五十嵐 健夫*

概要. 簡単なベクター画像を短いテキストで表現し，利便性を向上させる手法を提案する．図形表現は，グラフィカルな情報を伝達する際に文字表現よりも適切であるが，コンピュータ上で扱いにくいという欠点をもつ．たとえばプログラムのソースコードなどの純粋なテキストには図を直接含められず，別個の画像ファイルにすると紛失の可能性や検索コストが問題となる．本研究では，既存技術よりも効率的にベクター画像を圧縮する手法を提案し，簡単な図形のテキスト表現が多くの局面で手軽に利用可能なものであることを示す.

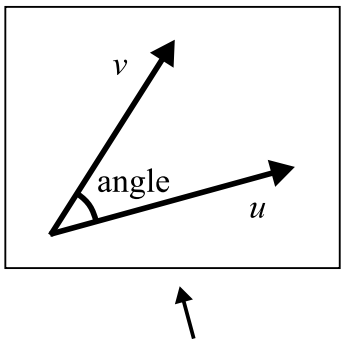
1　はじめに

近年のコンピュータの普及により，ますます多くの情報が電子的に記録されるようになってきている．人間が扱う情報は文字，図形などから構成され，今まで紙に情報が記録される際はこれらが自由に使われてきた．特に図・イラストは，情報を効果的に伝えることができるので重要である．

しかしコンピュータ上では，文字情報は扱いやすいが図形情報は扱いづらいという問題がある．たとえばプログラムのソースコードなどの純粋なテキストファイルには図を直接含められない．文字のみを駆使して図（ASCII アート）が作られることがあるが，これには非常に手間がかかる [4]．別個の画像ファイルを作成し，その場所を記録しておくこともできるが，今度は紛失の可能性や検索コストが問題となる [2]．さらに，Word, PowerPoint, PDF などの文字・図形両方を扱えるファイルフォーマットもあるが，上述のソースコードなどの用途には使えないうえ，対象アプリケーションの外では図形データを解釈することはおろかコピー・貼り付け・保存もできない．

そこで，本稿では図（ベクター画像）を短いテキストに圧縮する手法を提案する（図 1）．このテキスト表現を用いると，テキストファイルやチャットなどに図を直接含められ，図形エディタへのコピー・ペーストでいつでも閲覧・編集が可能となる．別個の画像ファイルにしたときに発生する管理コストも全くない．さらに，テキストエディタなどのソフトウェアが本手法に対応していれば，テキストファイルに埋め込まれた図のインライン表示なども可能になり，利便性が高い．

以下では，まず既存技術との比較を行い，提案手法およびそれを実装したシステムについて説明する．




```
// <diagram>MZeA3F0Aa4vxxDHAcQ</diagram>
double angle(Vector2D u, Vector2D v) {
    return Math.acos(innerProduct(u, v));
}
```

図 1. ベクター画像のテキスト表現．ソースコード中など，多くの場所に埋め込むことができる．

また，開発したシステムによってどのような図が描け，どのように利用できるかについて，行った調査・ユーザテストの結果に基づいて述べる．最後に本手法の応用例，今後の展望などについてまとめる．

2　関連技術

文字以外の情報をコンピュータ上で扱いやすくするための研究は今までに多く行われてきた．

ソースコードの表現力不足を解消する研究としては Barista [1] がある．Barista は多種類のメディアを表示可能なソースコードエディタの作成を支援するツールキットで，ソースコード中のコメントに図を挿入したり，ソースコード中の計算式を通常の数式として表示したりする（たとえば `sqrt(x*x + y*y)` を $\sqrt{x^2+y^2}$ と表示する）機能をもつ．しかし，図は一つのかたまりとして挿入・移動・削除できるだけで，図の内容を編集するためには既存の外部ソフトウェアが必要となり，個別の画像ファイルを操作す

* Kaisuke Nakajima and Takeo Igarashi, 東京大学大学院 情報理工学系研究科 コンピュータ科学専攻

る手間がかかってしまう．また，テキスト以外のメタデータがすべて XML としてファイル中に埋め込まれるため，Barista をベースにせずに作られた通常のエディタではユーザにとって意味をなさない文字が画面一杯に表示されてしまうこともある．

Blog や Wiki などに載せる図を手軽に扱えるシステムとしては，Web ブラウザ上で動くドローツール Willustrator [5] がある．従来必要だった画像ファイルのダウンロード・アップロードの手間が省け，すべてのデータが Web サーバ上に保存されるため，各種 Web アプリケーションと一緒に使う際は利便性が高い．しかし一方で，Willustrator がセットアップされた Web サーバがあってそこにユーザがアクセス可能でなければ使えず，ローカルファイル上に図を取り込みたい場合などには利用しにくいという問題もある．

また，ベクター画像を XML で表現するファイルフォーマット SVG や，電子メールに任意バイナリを添付するための MIME 符号化などの技術もあり，原理的にはこれらを用いて図をテキスト中に挿入することができる．しかし，これらは人手でテキストに埋め込まれることを想定した技術ではないため，データに冗長性がある．簡単な図を表現したデータでも非常にサイズがかさんでしまい，気軽に利用することはできない．一方，提案手法では非常にコンパクトな表現が得られるので，作成した図を各所で気軽に使うことができる．

3 システム

提案する図形のテキスト表現を利用可能な簡易ドローツールを Java で実装した．

3.1 利用手順

システムを起動すると，通常のドローツールと同様の編集画面が表示される（図 2）．ユーザは直線・矢印・長方形・楕円・テキストを描画でき，フリーハンドでベジェ曲線を描くこともできる [3]．また，通常の操作で切り取り・コピー・貼り付け・移動なども行える．

ユーザが図に手を加えるごとに，提案手法による図のテキスト表現が更新されて表示される．これをコピーし，テキストエディタに貼り付けることで図を保存できる．また，テキストファイルに含まれているテキスト表現をコピーし，本システムに貼り付けることで図を閲覧・修正できるようになる．

後述するとおり，多くの図が数十〜数百文字（80 文字毎に改行して 1 〜 10 行程度）で表現できるため，SVG などと違って気軽にコピー・ペーストできる．

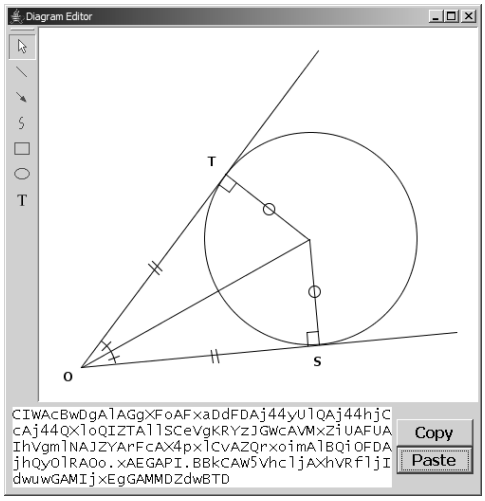


図 2. システム概観．通常のドローツールと同様に図を編集できるほか，図のテキスト表現が常時表示されていてコピー・ペーストができる．

3.2 テキスト表現の作成方式

現状では比較的単純な方式で図形情報の圧縮を行っている．各図形要素について，それを復元するのに必要十分な座標情報を記録している（可逆圧縮）．座標を構成する x, y 値それぞれを 11 ビット（$[-1024, 1023]$ の範囲）で表現し，最後に得られたビット列全体を BASE64 と同様の方式で符号化する（6 ビットずつを 64 種類の ASCII 印字可能文字 `[A-Za-z0-9.-]` に対応させる）．ビット列が 8 ビット単位で区切れるとは限らないので，BASE64 そのままでの符号化はしていない．

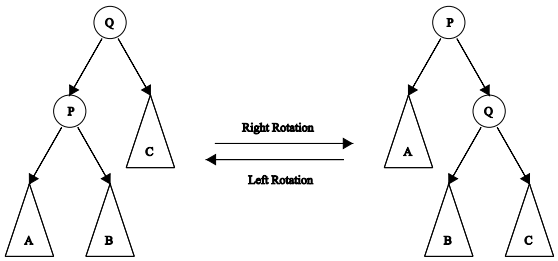
4 結果

4.1 描ける図の例

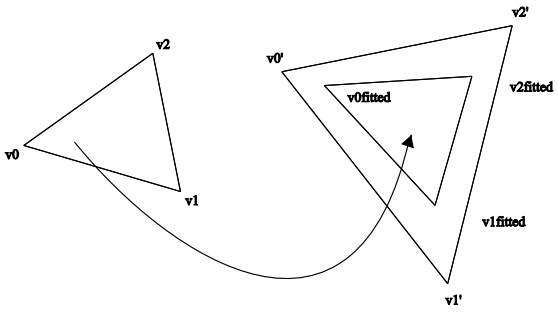
本システムを用いて筆者が描いた図の例を図 3 に示す．また，図 3 の内容を各方式でテキスト表現した際のデータ量の違いを図 4 に示す．筆者が調べた限り，数学・コンピュータ関連の書籍に載っている図を提案手法で表現するには高々 500 文字程度（実質数千ビット）で足りることが多く，テキストファイルに埋め込んでも特に邪魔にならないと考えられる．

4.2 ユーザテスト

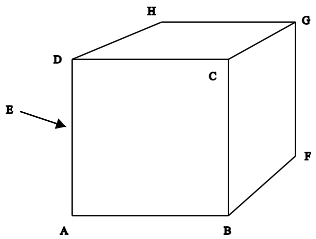
被験者二名（コンピュータに比較的習熟している理系大学生）に，図 5 (a), (b) に示した図を本システムで描いてもらった．この際，作図にかかった時間・出力されたテキスト表現のサイズを計測した（表 1）．既存ドローソフトと大きく変わらないインタフェースであることもあり，比較的容易に本システムを使用できているようだった．また,「便利そうなのでこのシステムを日常的に使ってみたい」,「既存ドロー



(a)
CIKAcAQAgMLYhABUBCwJAHAyBDQJARAyABwMgEBEABwMgKBEABARAKBEADgRAUBEAEQMgUBEAEQMgOBE
BEADgMAcCDwBgEAIDEGCwAVEwZkEAFBMQZBABQgFgHATAuAEwLgZAuAFgHAZAuBEgDgWAcDFWKwAUMzV
isAFBAzAuDkC4A2AcDkC4A2AcDMC4M6ZBABQgOwMg4BEAOARA.BECOQBgEAIAOwMg.BEBOgDg2AcDOmC
wAVATwA4QAHAEUDIQgRAEIEQSARAUECQRQMgj4BwBACAEUDISARAz9iEAFRNEZBABQxPwJA7AyDH2JQD
ljpZ2h0IFJvdGF0aW9uEeAoC8CgEuAsB0CwMgIyANIGVmdCBSb3RhdGlvbC



(b)
CAGg1RSBFAUgRReC9AXgvQaCVMU4QQCdjIwPipAJ2MDF.NEAnYxAigVTuAtA7gLTSENA0hDSKBVAnIYT
chZA3IWTMjJAzIySchhMg4TQDdjAnM7gJQDdjInM0RHwDdjEnMpobwIdjBmaXR0ZWQztBmAh2MmZpdHR
lZDOENsCHYxZml0dGVkQMAkhaD6KqWAwYjhMEJMwgjB



(c)
EAg4TiDj2AL49iDj2AL4Tgvj2AL4TiDhOAg4TingmAg49ini2Ap4Jini2AL4ThfgmAX4JingmEE4hgri
WMKRCgBQTIEQoAUIx5BkAFDMJYUgBRDAwIgAUUysi6AFGMqwKABRzFgB4AUI

図 3. 筆者が描いた図の例および本手法によるテキスト表現．いずれも言葉だけでは表現しにくく，本手法により手軽に扱えると利便性が向上すると考えられる．(a) 二分探索木の回転操作．(b) 三角形のフィッティング．(c) 立方体頂点のラベル付け．

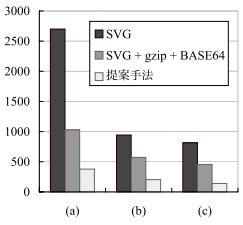
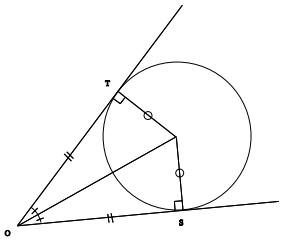
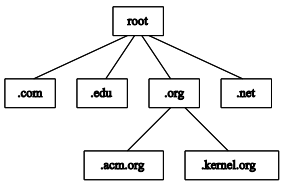


図 4. SVG 単体，SVG + gzip + BASE64，および提案手法で図 3 (a) – (c) を表現したときの文字数の違い．



(a)



(b)

図 5. ユーザテストで被験者に描いてもらった図．

表 1. ユーザテスト結果（作図所要時間およびテキスト表現サイズ）．テキスト表現サイズは図形プリミティブの種類・個数のみで決まるため，両被験者間で一致した．

	図 5 (a)	図 5 (b)
被験者 1 所要時間	367 秒	522 秒
被験者 2 所要時間	261 秒	319 秒
テキスト表現サイズ	183 字	367 字

ソフトがどれも備えているような機能はつけてほしい」などの意見・感想をもらった．

5 応用例

5.1 プログラムのソースコード

図 1 で示したように，本手法を用いることで図を直接ソースコードに含めることが可能である．また，エディタ側で図を解釈する機能をプラグインなどとして実装すると，埋め込んだ図を提案システムにコピー・ペーストする手間すら不要になる．未対応の通常のエディタでも，図が大きなスペースをとらないため，作業の邪魔にはならない．

なお，このような機能に関するニーズは，筆者の周囲でコンピュータグラフィックス関連のプログラムを書いている人々の間で特に多い．外部画像ファイルを作ってまで説明をしたくはないが，言葉だけで表現するのは難しい，という内容が多いようである．

5.2 Wiki

Wiki は近年広く利用され，多くの情報が記録されているが，テキスト情報が簡単に編集できるのに対し，図を編集するにはダウンロード・アップロード・画像ファイルの管理・ベクター画像からラスター画像への変換などの手間がかかってしまう．提案手法を用いると，HTML ソース中に

```
<img src="http://www.example.com/draw.cgi?
diagram=CMEYLADb3J6QDoOAUArB6DsIYLD" />
```

などとして図のテキスト表現を直接埋め込み，CGI にラスター画像を生成させることができる（既存の

Web ブラウザを何も改変せずに表示可能）．今回作成したドローツールは Web ブラウザ上でも実行できるため，HTML 中に埋め込まれた図を読み込ませてシームレスに編集することもできる．

5.3 TEX

TEX ソースファイルに図のテキスト表現を埋め込んでおき，プリプロセッサで EPS ファイルを生成すれば，通常と同様に組版を行える．図を外部ファイルとして作成して個別に管理する手間が省ける．

6 まとめと今後の課題

簡単なベクター画像を短いテキストで表現する手法を提案した．本手法を用いることで，ソースコードなどの任意のテキストに図を手軽に含めることができる．Word などの専用アプリケーションで作成した図よりも広範囲で利用でき，外部画像ファイルにした場合に生じる諸問題にも悩まされずに済む．

今後の研究の方向性としては，以下のようなものを考えている．

6.1 ベクター画像の非可逆圧縮

現在の方式でも簡単な図を数十文字から数百文字程度で表現でき，実用上大きな問題はないと思われるが，より少ない容量で必要な図が表現できれば利用がさらに手軽になるのは明らかである．そこで，今回扱ったようなベクター画像に特化した圧縮方法を検討している．

鍵となるのは，ほとんどの場合に正確な座標を記述する必要がないことである．たとえば図 6 (a), (b) に示した図形では各部分の長さ・角度などが異なっているが，人間にとってはどちらでもよい．一方，図形要素の相対位置・接続関係などが変化すると人間には異なる図形として認識されてしまう．このように，人間の知覚にとって重要な図の本質的データのみをうまく保存し，効果的な非可逆圧縮方式を開発できるのではないかと考えている．またこの際，ユーザが重要な部分とそうでない部分を指定して，圧縮率と正確さのトレードオフを調整できるようなインタフェースについても検討を行っている．

6.2 機能拡充および長期的ユーザテスト

提案手法により，さまざまな局面で図形情報の利用が容易になるが，既存のグラフィックスソフトウェアと比べると機能が圧倒的に少ないため，現在のシステムでは描きにくい図もあると考えられる．これを解消するため，必要以上にシステムが複雑にならない範囲で基本的な機能の追加を行いたい．

また，今回簡単なユーザテストは行ったものの，提案手法を日常的なツールとして使った場合に何か問題が起こるかどうか確認できていないため，今後長期的なユーザテストを行う予定である．

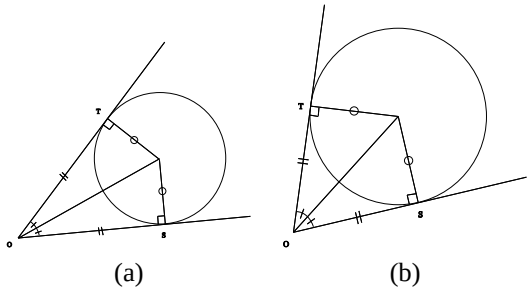


図 6. ベクター画像の非可逆圧縮の可能性．(a), (b) の各図形要素の座標情報はまったく異なるが，人間にとって本質的な違いはない．

参考文献

[1] A. J. Ko and B. A. Myers. Barista: An Implementation Framework for Enabling New Tools, Interaction Techniques and Views in Code Editors. In *CHI '06: Proceedings of the SIGCHI conference on Human Factors in computing systems*, pp. 387–396, 2006.

[2] T. LaToza, G. Venolia, and R. DeLine. Maintaining Mental Models: A Study of Corrective and Perfective Maintenance Tasks. In *Proceeding of the 28th international conference on Software engineering*, pp. 126–135, 2005.

[3] P. J. Schneider. An Algorithm for Automatically Fitting Digitized Curves. In *Graphics Gems I*, pp. 612–626. Academic Press, 1990.

[4] D. Spinellis. *Code Reading: The Open Source Perspective*. Addison Wesley, 2003.

[5] 神原 啓介, 安村 通晃. Willustrator: Web 上での協同イラストレーション. 第 13 回インタラクティブシステムとソフトウェアに関するワークショップ (WISS 2005) 論文集, pp. 139–140. 日本ソフトウェア科学会 ISS 研究会, 2005.