音声入力を用いたビジュアルプログラミング環境の操作効率向上

田中 秀明† 平井 重行††

Efficient Visual Programming using Speech Input

HIDEAKI TANAKA,† SHIGEYUKI HIRAI††

1.はじめに

音声認識技術の向上および実用化に伴い、それを活用したインタフェース研究が行われている[1][2]。我々はそのようなインタフェース研究の一つとして、音声認識技術を Puredata (Pd)や Max/MSP、vvvv などのビジュアルプログラミング環境で活用する研究を行っている。これらのプログラミング環境は、ある機能を持つオブジェクトを線で接続することで、データフローをビジュアル的に記述するプログラミングスタイルを持っている。図 1 に Puredata の画面例を示す。この環境では、マウスカーソルでオブジェクトの配置や線の接続を行う一方、オブジェクトの機能やパラメータを文字列としてキー入力を行う必要が生じる。プログラミング作業中は片手をマウスとキーボード間で頻繁に移動させることとなり、そのオーバーヘッドのためにプログラミング効率が悪くなることが多々ある。そこで、我々は音声入力を導入してマルチモーダルな文字列入力を行い、キー入力を極力減らすことで操作のオーバーヘッドを少なくすることを提案する。

本稿では Puredata 上で音声入力を実現して、効率化に関する基礎的な実験を行ったので、それらについて報告する。

2.システム概要

Puredataに対して音声入力を行うシステムの概要は図2の通りである。マイク入力した音声から音声認識エンジンを用いて文字列を得て、それをキーボードイベントに変換するプログラムを介してPuredataへ送ることで、プログラミング中の音声入力を実現した。音声認識については、Puredata で扱うオブジェクト名など認識対象がある程度限定できるため、独自の記述文法に基づいて認識を行えるJulianを用いた。

以下に本システムのために作成したデータおよびソフトウェアについて説明する。

2.1 音声認識用データの作成

Julian を用いて音声認識を行うために dfa ファイルと dict ファイルを作成した。dfa ファイルの中身は「1 0 2 0 0」のようになっており、dict ファイルの中身は「1 ［metro］ m e t o r o」のように

†京都産業大学理学研究科
†Graduate School of Division of Science, Kyoto Sangyo University
††京都産業大学理学部
††Faculty of Science,Kyoto Sangyo University

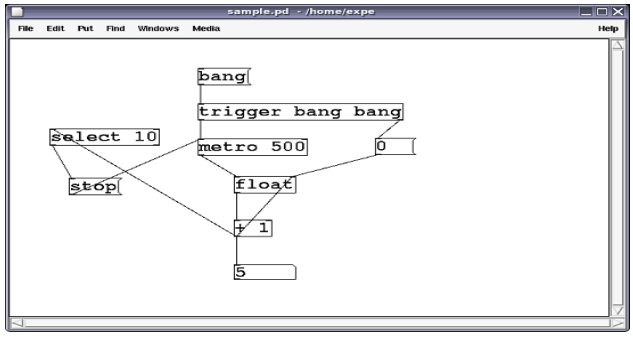

図 1. Puredata の画面例

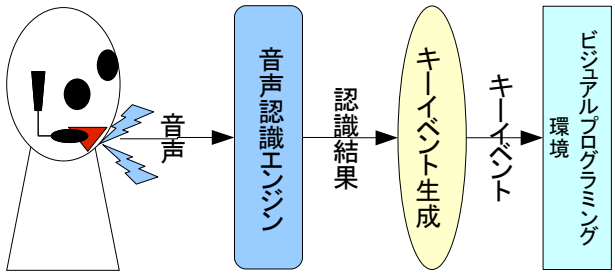

図 2. システム概要

なっている。現時点では、Puredataのオブジェクト約80種、記号25種、アルファベット26種、0ー999までの数字が認識可能である。

2.2キーイベント生成・送信プログラム

認識結果からキーイベントを生成し、送信する流れは以下のようになる。

1. Julian の認識結果を受け取る
2. 認識結果から必要な部分を抽出
3. 抽出結果からキーイベントを生成
4. キーイベント送信

現時点のシステム はLinux 上で実現しており、Julian の認識結果のデータから生成するキーイベントは X-Window のキーイベントとして Puredata へ送信している。このソフトウェアは C 言語で開発した。

以上のことを行うことにより、Puredata 上で音声認識によるキー入力を実現した。

3. 操作効率に関する基礎実験

音声入力を行うことによる Puredata の操作効率の変化について、基礎的な被験者実験を行った。

3.1 実験方法

被験者は提示された文字列をオブジェクトとして 10 個生成することとし、5 個ずつ横一列になるよう配置する。この文字列入力を、キーボードで 10 個生成し、次に音声で 10 個生成する順番でそれらに

かかる時間を計測した。オブジェクトボックス生成にはマウス操作に使わない手（基本的に左手）でショートカットキーを使用することとした。キーボードと音声で10個ずつオブジェクト生成するのを1セットとし、各被験者とも文字列毎に7セットの実験を行った。ただし、キーボード入力については、入力作業中の修正は行ってもよいが、生成完了後の見直しての修正作業は禁止した。また、音声入力の際には、発話ミスに関しては言い直しを可とし、認識エンジンによる誤認識があった場合は、修正などせずに実験を続けることとした。

3.2 入力文字列の分類

今回の実験で被験者が入力する文字列は、事前に予備実験を行い、音声認識率が高いものを採用した。また、使用した文字列はキータイプ数に応じてA～Cに分類した。Aは"metro"や"bang"等のタイプ数3～5回のもの、Bは”namecanvas”等のタイプ数6以上のもの、Cは"&"、"4"等のタイプ数が1回で終わるものとした。

3.3 実験環境及び条件

実験環境を以下に示す。

OS　：VineLinux（Kernel2.4）
CPU/メモリ　：Pentium4 3.00GHz/1024MB
ビジュアルプログラミング環境 ：PureData(0.39-2)
音声認識エンジン：Julian(3.5)
音素モデル　：Juliusディクテーション実行キットに同封のWebから学習した語彙数6万のモデル
言語モデル　：2.1節で説明したデータ
音声入力機器　：ヘッドセットマイク

実験は20代～30代男性10名、20代の女性2名の被験者で行った。被験者には実験目的と作業についての説明を行い、各操作方法の説明をし、10分間のPuredataの操作及び音声入力の練習時間を与えた。実験中はビデオによる撮影を行い、各試行中のキーボード入力と音声入力のそれぞれの作業時間を計測した。

3.4 実験結果・考察

実験の結果から、A～Cのデータ毎に被験者全体の平均と標準偏差を求め、図3のグラフにまとめた。

グラフのデータは、キーボード入力と音声入力の1回の試行の平均作業時間を示している。なお、繰り返し同じ作業を行うことでの慣れや学習能力の関係上、この結果には各作業回数7セットの内、最初の2セット分は計算から除外した。

図3の棒グラフより、A～Cのどの文字列群においても、音声入力のほうが操作時間の短いことがわかる。Bのようにキータイプ数が多いものは、手の移動によるオーバーヘッドより文字列入力の速度差による時間短縮が大きな要因と言える。Cの1タイプの場合は、操作時間にそれほど差はないが、音声のほうが時間は短い。また、標準偏差に着目すると、キーボード入力は音声入力に比べて大きな値であることから、キーボードのほうが被験者間またはセット間のばらつきが大きいことがわかる。

キーボード入力で操作時間が多くかかっている要因は、マウス-キーボード間の手の移動はもちろんだが、キーボードへ手を移動した際にホームポジションを探す時間がかかるほか、マウスへ手を戻した際にマウスが動いてカーソルも動き、次のマウス操作までタイムラグが発生するということが挙げられる。

音声による作業速度のばらつきが少ないことから、どのような文字列に対しても、同じような速度で入力が可能であることがわかる。このことにより、あまりプログラミングに慣れていない人でも効率よく文字列入力が行えると言える。

以上より、Puredata上において、キーボードによる文字列入力に代わり、音声による入力を行うことで効率のよい作業を行えることがわかった。

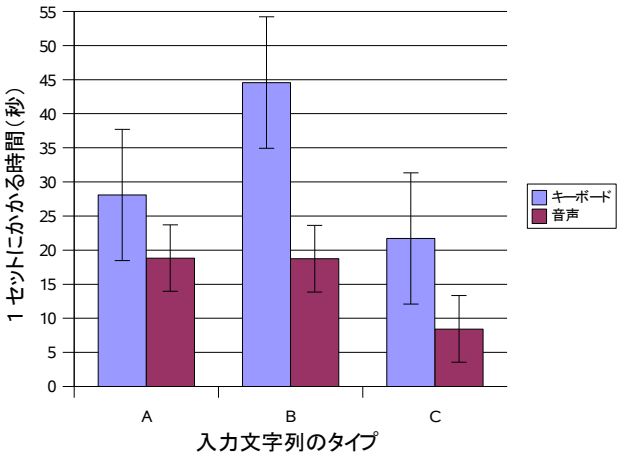

図3.　実験結果

4.　まとめ・今後の課題

ビジュアルプログラミング環境において効率よくプログラミングを出来るようにするために、キーボードの代わりに、音声で文字列入力を行うことを提案し、Puredataを用いてオブジェクト生成と配置の基礎的な実験を行った。この結果から、音声入力を行うことでキーボード入力の際に生じる様々な弊害を解消することができ、文字列入力とオブジェクトの配置作業が効率よく行えるという結果が得られた。

今後は、オブジェクトの生成・配置作業だけでなく、オブジェクト間を線で結ぶ作業についても実験を行い、さらに具体的なプログラム作成作業における評価を行う予定である。

参考文献

[1]Steve Whittaker,　Brian Amento：Semantic Speech Editing,　Proc. CHI 2004,　pp.527-534.
[2]Kazutaka Kurihara, Masataka Goto, Jun Ogata and Takeo Igarashi, "Speech Pen: Predictive Handwriting based on Ambient Multimodal Recognition," Proc. CHI2006, pp.851-860.