ながらデバッグマスク:口の表情を入力としてソースコードの読み上げとバグの検出・記録が可能な仕組み

Debugging Mask: A system that can read source code and detect and record bugs by using oral shapes as input

5116E018-7　塙　克樹　　指導教員　橋田　朋子　准教授

KATSUKI Hanawa　　Assoc. Prof. HASHIDA Tomoko

概要：従来のプログラミング方法は手を動かしての入力とモニタからの視覚情報の提示を必要とする．本稿では，この問題を解決するために，発声の伴わない口の表情を入力とし聴覚情報の提示でデバッグを行うシステムを提案する．外部に音声に漏らすことなく使用可能な口の表情を 5 種類，マスク型デバイスで識別して，ソースコードの読み上げとバグの検出および記録を操作するシステムを実装した．聴覚情報でのデバッグに関する予備検討とマスク型デバイスの精度実験を行った．

キーワード：プログラミング，SVM，デバッグ

Keywords: programming, SVM, debug

1. はじめに

筆者らはプログラミングの必要性が増す中で，プログラミングを行うことが可能な環境を拡げていくことが望ましいと考える．この課題に対する先駆的な試みとして音声でプログラミングを行う方法が挙げられるが[1][2]，公共空間でこの方法を用いることが難しい．公共空間において一人で話すことは，他人に不快感を覚えさせ，気まずい行為とされているためである．

本研究では”公共空間での歩行時”のように従来プログラミングを行うことが難しかった場面において，周りに迷惑をかけず簡易なプログラミングを行うことを目指す．そのためにプログラミングの中でも，確認作業に近いデバッグに注目する．また歩行時でも任意に動かすことができ，他者に迷惑をかけることのない口の表情を入力として用いた．ソースコードをユーザにヘッドフォンで聴覚情報として提示し，ユーザの発声を伴わない口の表情を識別し入力として用いることで，デバッグを行えるシステムを提案する．今回は市販のマスクの下にフォトリフレクタモジュールを複数配置することで，発声を伴わない 5 種類の口の表情の識別を実現する．本システムを用いてる様子を図 1 に示す．

本稿では聴覚情報の提示によるデバッグに関する予備検討の結果と提案システムの詳細，口の表情の識別精度を報告する．

2. 予備検討

歩行時に聴覚情報でソースコードを提示して実際にバグを発見することができるのかを明らかにするために実験を行った．比較として視覚情報でバグ発見に関する実験を行った．用いたバグは単純なスペルミスのバグであるが，視覚的に類似した文字のスペルミスとランダムなスペルミスの 2 条件用意した．20 代の 4 人の被験者に対して視覚によるソースコードの提示と聴覚によるソースコードの提示を交互に行った．被験者には歩行しながらソースコードのバグの検出を指示し，バグの発見率を計測した．

図 1　口の表情を識別しソースコード読み上げを操作することでバグを聴覚情報の提示で発見を行っている様子

実験の結果，聴覚情報の提示によるバグの発見率は 2 条件どちらでも 75.0%となった．歩行時に視覚情報の提示によって行った場合のバグ発見率はランダムなスペルミスでは 100%なのに対して，視覚的に類似した文字のスペルミスのバグでは 83.0%となった．このことから，視覚に劣るものの，視覚的類似性によらず歩行時に聴覚情報を提示することでバグを発見できることが示唆された．

3. システム

提案システムの構成を図 2 に示す．提案システムはユーザの口の表情を検出する表情検出部と機械学習を用いた口の表情を識別する表情識別部，口の表情に対応してソースコードの読み上げとバグが存在する行の記録を行う出力部から構成される．

3.1 表情検出部

マイコン（Arduino Uno），AD コンバータ（MCP3208-CI/P）と 14 個のフォトリフレクタモジュールから構成されるマスク型デバイスを製作した．このデバイスは市販

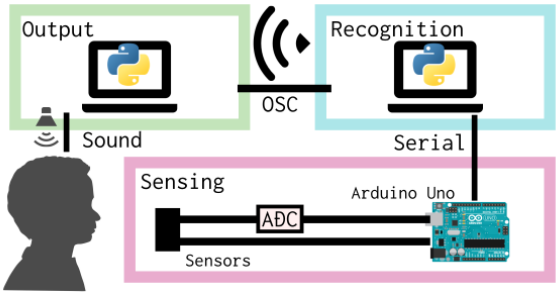

図 2　システム構成図

図 3　識別に用いた 5 種類の口の表情

の紙マスクの下に設置し，口を覆う形になっている．各フォトリフレクタモジュールと顔との距離をリアルタイムに計測することで，口の周りの動きを検出している．口の表情を識別するために，取得したセンサデータをシリアル通信で PC に送信している．

3.2 表情識別部

本システムでは無表情（Normal），笑顔（Smile），驚き顔（Surprised），下顎を右に動かした口の表情（Right），下顎を左に動かした口の表情（Left）の 5 種類の口の表情を識別した．識別を行った口の表情を図 3 に示す．これらは個人間での理解と共有が簡易な口の表情である．5 種類の口の表情を識別するために Python 上で Support Vector Machine（SVM）を用いた．5 種類の口の表情ごとに学習は 10 試行繰り返した．収集したデータを用いて，5 種類の口の表情を識別するための識別モデルを個別に生成した．学習によって識別された口の表情を，ソースコードの読み上げとバグが存在する行の記録を行う出力部に OSC 通信で送信している．

3.3 出力部

送信された口の表情に応じて，Python によって制作したソフトウェアを用いてソースコードの読み上げもしくはバグが存在すると判断した行の記録を行った．Right でソースコードを一行下に移動して読み上げ，Left でソースコードを一行上に移動して読み上げる．Smile で同じ行を再び読み上げる．Surprised でバグがあるとユーザが判断したソースコードの行を記録する．

4. 精度実験

製作したマスク型デバイスを用いた口の表情の識別精度を評価するために，実験を行った．

4.1 実験の手順

20 代の 4 人の被験者はマスク型デバイスを装着して，

表 1　混合行列：5 種類の口の表情の認識率

		Oral Shape				
	Actual/Predict	N	SM	SU	R	L
Oral Shape	Normal(N)	100	0	0	0	0
	Smile(SM)	2.5	97.5	0	0	0
	Surprised(SU)	0	0	92.5	7.5	0
	Right(R)	5	5	0	90	0
	Left(L)	0	0	0	0	100

実験データの収集を行った．室内環境で椅子に座った姿勢でモニタと正対させたうえで，5 種類の口の表情ののうちランダムに選ばれた 1 つを指示し被験者に行わせた．すべての口の表情で 10 セットずつデータの計測が完了した時点で終了とした．

4.2 実験の結果

分析には 10 分割交差検証を用いており，5 種類の口の表情での被験者毎の認識率を算出した．具体的には，ある 1 人の被験者から収集したセンサデータのうち 9 回分のデータを教師データとして学習し，その口の表情の識別モデルを生成する．その被験者の残り 1 回分のデータをテストデータとして，生成したモデルの精度を算出する．これを 1 セットずつずらして 10 回全ての組み合わせに対して同様の操作を行い，算出された結果の平均値を求めた．この結果をその被験者の認識率とし，4 人の被験者全員の認識率の混合行列を求めた．4 人の被験者全員の認識率の混合行列を表 1 に示す．実験の結果，平均認識率は 96.0%（SD=4.5）であった．被験者内であれば，製作したマスク型デバイスを用いて 5 種類の口の表情の識別を十分に行うことができる精度があると判断した．

5. まとめと今後の展望

本論文では，公共空間で歩行時のように従来プログラミングを行えない環境で口の表情を識別し入力として用いた．そのことによって，ソースコードの聴覚情報での提示とバグの発見が可能なシステムを提案した．聴覚によってバグ発見が可能であるかを調べるための予備検討を行い，聴覚によってバグを発見できることが示唆された．また，口の表情を識別するためのマスク型デバイスを作成し精度を評価した．5 種類の口の表情の十分な識別精度を得た．

今後は口を使う上での入力の少なさと聴覚による得意な領域を考慮した新しいプログラミングの形を検討していく．

参考文献

[1] “Using Python to Code by Voice”. https://www.youtube.com/watch?v=8SkdfdXWYaI, (参照 2017-12-12).

[2] Amber Wagner, Ramaraju Rudraraju, Srinivasa Datla, Avishek Banerjee, Mandar Sudame, and Jeff Gray. 2012. Programming by Voice: A Hands-Free Approach for Motorically Challenged Children. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '12). ACM, Austin, TX, USA. 2087-2092