Android端末のための実行履歴を用いた音声対話コンテンツ編集システム

山口 大介[1] 堤 修平[1] 山本 大介[1] 高橋 直久[1]

概要：我々の研究室では，音声対話システム構築ツールキット MMDAgent を Android 向けに移植した Android 版 MMDAgent の研究を行なっている．Android 版 MMDAgent は，FST 形式で記述された対話スクリプトを読み込ませることで対話内容を自由にカスタマイズすることができる．FST ファイルはパソコン上のテキストエディタなどによって編集することができ，深い専門知識が無い者でも編集ができるようになっている．しかし，Android 版 MMDAgent の FST ファイルは Android 端末をパソコンに接続しなければ，カスタマイズすることが煩わしいと感じる場合が多く，記述された対話内容の量が多いほど，後から編集したいと感じた箇所を探すことや，適切な編集をすることが困難となってくる．Android 端末上で使用するアプリの簡単なカスタマイズは，使用する端末上で完結する方が好ましいと考えられるので，本稿では MMDAgent の対話の元となっている FST ファイルを Android 端末単体でカスタマイズすることのできるシステムを提案し，その提案システムに基づいて実装したプロトタイプシステムについて述べる．さらに，プロトタイプシステムを用いた評価実験から，提案システムの有用性を評価する．

Voice Interaction Contents Editor Based on History for Android Devices

DAISUKE YAMAGUCHI[1] SHUHEI TSUTSUMI[1] DAISUKE YMAMOTO[1] NAOHISA TAKAHASHI[1]

1. はじめに

近年，Apple の Siri[1] や，NTT ドコモのしゃべってコンシェル [2] など，スマートフォン向けの音声対話システムが普及しつつある．これらの音声対話システムでは，ユーザは仮想の秘書エージェントとの音声対話を通じて，経路案内や天気予報などの情報を得ることができると同時に，簡単な雑談なども行なうことができる．我々の研究室では音声対話システム構築ツールキット MMDAgent[3] を Android 端末向けに移植した Android 版 MMDAgent[4] に関する研究を行なっている．MMDAgent は，名古屋工業大学正門前にも設置されている双方向音声案内デジタルサイネージ [5] にも用いられており，上記の音声対話システムとは異なった音声対話を実現している．MMDAgent はネットワークを介さない対話が可能となっているため，より人間に近い，遅延の少ない対話が実現されている．MMDAgent はネットワークを介した対話を行なわない代わりに，FST ファイルと呼ばれる対話スクリプトで記述されたファイルを読み込むことで，そのファイル内に記述されているキーワードに関する対話だけを実行する．

MMDAgent をインストールすると，MMDAgent 起動時の処理や，「こんにちは」，「自己紹介」などの簡単なキーワードに対する対話が既に記述されている FST ファイルが付属している．FST ファイルはパソコン上のテキストエディタなどによって編集することができ，反応するキーワードを任意に増やすことやキーワードに対する返答を変更することなどができる．しかし，Android 版 MMDAgent の FST ファイルを編集しようとすると，Android 端末上のエディタの機能性や Android 端末の画面サイズの小ささなどにより，編集が困難と感じる場合が多い．しかし，Android 端末上で使用するアプリの簡単なカスタマイズは Android 端末上で完結するほうが好ましいと考えられる．

[1] 名古屋工業大学大学院工学研究科
Graduate School of Engineering, Nagoya Institute of Technology

2. MMDAgent

2.1 MMDAgent の特徴

MMDAgent は音声対話のための高度な機能を備えたツールキットであり，音声認識，音声合成，3D モデルの描画や物理演算などを統合したシステムである．音声認識エンジンとして Julius[6] を，音声合成エンジンとして OpenJTalk を [7]，3D モデルとして MikuMikuDance 形式 [8] を，物理演算エンジンとして Bullet Physics[9] を採用している．また，MMDAgent は音声対話を記述するためのスクリプト言語に基づいて音声対話の処理を実行する．スクリプト言語は後述する FST(Finite State Transducer) 形式に基づいており，音声認識による特定のキーワードの認識や，センサーによるユーザの検出などをトリガーとして，次の状態に遷移する有限オートマトン形式を記述可能である．

2.2 FST 形式の特徴

MMDAgent は FST 形式で記述されたテキストファイル (以降 FST ファイルと呼ぶ) を読み込むことで，音声対話を実現している．FST 形式とは，状態番号，次状態番号，条件，出力の順に並ぶ対話スクリプトである．
以下に，MMDAgent が「こんにちは」と認識した際に，お辞儀をしながら「こんにちは」と返答するように記述された部分を含む FST ファイルの一部と，その「こんにちは」の処理における状態遷移図をそれぞれ図 1,2 に示す．

```
1 10    RECOG_EVENT_STOP|こんにちは    <eps>
1 10    RECOG_EVENT_STOP|こんにちわ    <eps>
10 11   <eps>                          MOTION_ADD|mei|action|Motion¥mei_greeting¥mei_greeting.vmd|PART|ONCE
11 12   <eps>                          SYNTH_START|mei|mei_voice_normal|こんにちは
12 1    SYNTH_EVENT_STOP|mei           <eps>
```

図 1　FST ファイル

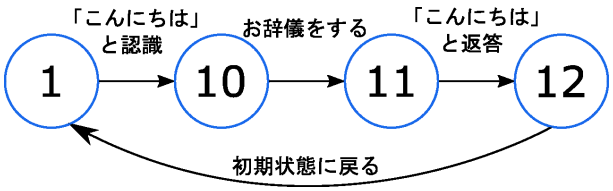


図 2　FST ファイル

図 1 中の番号が”1”のような状態を初期状態 (又は待機状態) と呼ぶ．MMDAgent が何もしていない時は，基本的にこの初期状態にある．そして，MMDAgent が初期状態にあるときに音声認識によって「こんにちは」と認識すると，次の状態，例えば図 1 中の番号”10”の状態へ遷移する．番号”10”の状態は，図 1 中の 3 行目の状態となっており，出力は「お辞儀をする」となっているので，MMDAgent がお辞儀をする．また，図 1 中の 3 行目の条件を見ると<eps>となっている．これは，「条件無しで次の状態へ遷移する」という条件なので，すぐに図 1 中の番号”11”の状態へと遷移する．番号”11”の状態の条件は，番号”10”の状態と同じなので，番号”12”の状態へと遷移し，出力は，「こんにちはと返答する」となっている．図 1 中の番号”12”の状態は，条件が「音声合成が終わった時」となっている．これは，「こんにちはと返答する」などの音声合成の出力が終了すると遷移するという条件である．出力はないので，音声合成が終わると初期状態に遷移するという遷移である．これらの状態遷移を状態遷移図で表したのが，図 2 である．

2.3 FST ファイルの編集における問題点

同一 FST ファイル上で状態番号が重複してしまうと，FST ファイル中の，同じ状態番号の上に書かれた行だけが実行され，下に書かれた行は実行されないなどの FST ファイルを記述した者の意図しない動作が起こることがある．

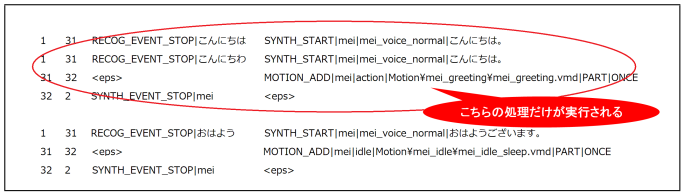


図 3　状態番号の重複

他にも，誤字などの記述ミスが起こる場合がある．例えば，「MOTION_ADD」を「MOTION_ABB」と誤って記述してしまった場合，MMDAgent は上手く動作しなくなってしまう．

Android 版 MMDAgent の FST ファイルを編集する場合，Android 端末上で編集するには画面サイズの小ささや，エディタがパソコンに比べて不便であるなどの理由により困難と感じる場合がある．そのため，Android 版 MMDAgent の FST ファイルを編集する場合，パソコン上で FST ファイルを編集してから，使用する Android 端末に保存するという方法が主である．

3. 提案システム

3.1 目的

FST ファイルの編集における問題点として同じ状態番号を使用してしまうことや，誤字などにより MMDAgent が正常に動作しないことが挙げられる．これらの問題は，FST ファイルが視覚的に同じような行が並んでしまっているため，FST ファイルに記述されている内容を正確に理解することが困難であるためであると考えられる．そこで，直接 FST ファイルを編集しないようにすることで Anroid 版 MMDAgent の FST ファイルを楽に編集できるようにしようと考えた．

3.2 実現上の課題

Android 版 MMDAgent には，実行された出力コマンドを表示する機能が実装されている．この機能を，イベント表示機能と呼ぶことにする．以下の図 4 にイベント表示機能で表示された画面を示す．



図 4　イベント表示機能

イベント表示機能で表示された出力イベントは，MMDAgent がどのように認識したか，実際に実行した出力コマンドなどしか表示されない．そのため，状態番号などの FST ファイルを編集する際に必要な情報が不足している．

また，仮にどの行を編集するか決定できたとしても，編集内容を決定するための処理を新たに追加しなくてはいけない．しかし，Android 端末上で FST ファイルを編集しようとすると，画面の小ささや，ソフトキーボードなどの存在により編集が面倒と感じる場合が多いと考えられる．

3.3 提案システムの構成

MMDAgent は FST ファイルを読み込み，そのファイルに記述されている内容に応じた処理を行なっている．そのため，MMDAgent が行なった処理を 1 行ごとに分割し，Andoroid 端末内にテキスト形式で保存することができる．保存したテキストファイルを用いることで，実行履歴として，MMDAgent が実行した履歴を Android 端末上に表示することができる．表示した実行履歴は，任意の 1 行を選択可能となっており，編集したいと感じた行を，タップすることによって選択できる．編集したい行が決定すると，FST ファイルの編集内容を決定するリストに切り替わる．例えば，実行履歴中のモーションに関する行を選択すると，モーションの削除，追加，変更の 3 つの編集内容から 1 つを選択できるようになっている．そして，その中から追加や変更を選択した場合，新たに追加 (変更) するモーションを選択できるリストに切り替わる．削除を選択した場合は，そのまま編集内容が決定される．これらの編集内容を選択形式によって決定することにより，新たに FST ファイルを編集し，自動的に FST ファイルの再読み込みを行なうという流れになっている．

4. 実現方法

4.1 FST ファイル編集機能

提案システムを用いた FST ファイルの編集は，3 つのステップを行なうことで実現している．

実効履歴表示

MMDAgent は FST ファイルを読み込んで，処理を実行している．例えば，「こんにちは」と認識すると，FST ファイル中の「こんにちは」をキーワードとして行なう処理を FST ファイルから読み込み，MMDAgent 上で実行している．この一連の処理を行なう際に，読み込んだ処理を Android 端末内に履歴ファイルとして保存する機能を追加した．この機能によって保存された履歴ファイルを読み込むことで，MMDAgent が実行した履歴だけを取得することができる．

編集したい行を選択するためには，MMDAgent 上のメニューボタンから実効履歴を表示する．以下に，MMDAgent 上のメニューボタンの画面を図 5 に示す．

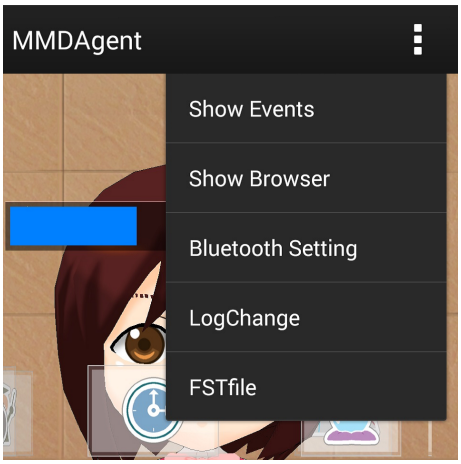


図 5　MMDAgent 上のメニューボタン

実効履歴を表示するために，保存されている履歴ファイルを読み込み，その後 Android 画面上にリスト表示する．このとき，表示されるリストは状態番号，次状態番号，条件，出力のセットを 1 行とし，1 行ごとに分割して表示することで，編集を行ないたい 1 行だけを選択できるようする．また，このセットを 1 行とすることで FST ファイル内に記述されている形式と同じ形式で表示することができる．そのため，FST ファイルに知識のある者ならば，実効履歴を見ても親しみが沸くと考えた．また，FST ファイルにまったく知識がない者でも，「MOTION」や「こんにちは」など，部分的に意味が分かれば良い場合が多いため，編集の妨げにはならないと考えた．このように実効履歴を用いることで，MMDAgent の動作を確認した直後

に FST ファイルを編集することができる．FST ファイルは，MMDAgent の使用方法にもよるが，名工大正門前の MMDAgent などのように，デジタルサイネージとして使用される場合，記述内容が数千行に及ぶ場合がある．また，FST ファイルの記述形式の性質上，視覚的に非常に似ている行が複数行に渡って記述されるため，実効履歴を用いずに編集したい行を選択することは困難であると考えられる．しかし，実効履歴を用いることで，MMDAgent の動作を確認した後，実効履歴中の数行から選択するだけでよくなる．以下に，表示されるリストの例を図 6 に示す．

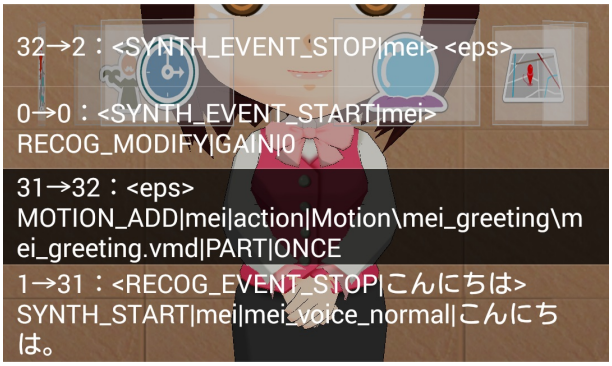


図 6　実行履歴

編集内容指定

ユーザが編集を行ないたいと感じた行を実効履歴から選択すると，選択した内容に応じた，編集内容を指定するリストに切り替わる．例えば，ユーザが実効履歴中のモーションに関係する 1 行を選択するとモーションの追加，変更，削除の 3 つの選択肢があるリストが表示される．さらに，モーションを編集する場合，変更後や追加後モーションを決定しなければいけないため，同様に変更や追加できるモーションのリストが表示される．これにより，ユーザは FST ファイルを編集するために文字をタイプする必要がなくなるため，FST ファイルに記述される内容に誤字脱字が起こらなくなる．以下に，実効履歴中のモーションに関係する 1 行を選択された後の，FST ファイル編集内容を指定するリストを表示したリストの例を図 7，8 に示す．

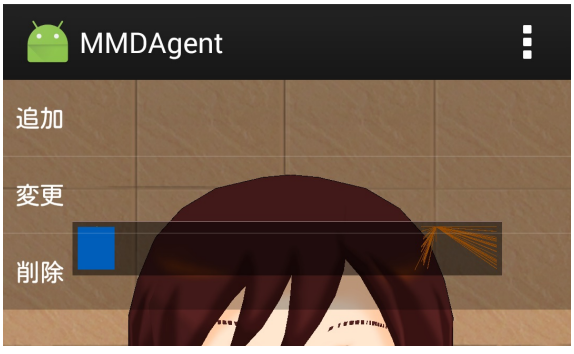


図 7　モーションに関して変更できる内容を表すリスト

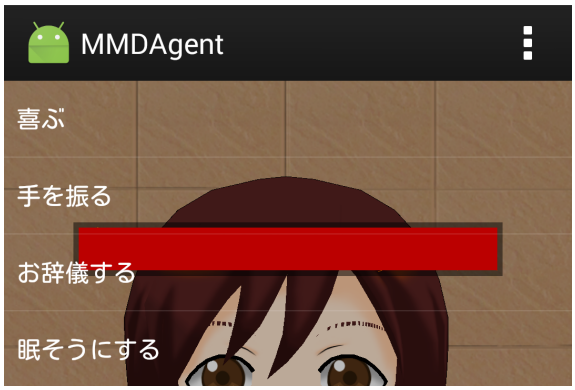


図 8　変更できるモーションを表すリスト

FST ファイルへ出力

最後に，上記の方法によって決定した編集内容を実際の FST ファイルに反映させる．

まず，Android 端末内にある FST ファイルを読み込み 1 行ごとに分割する．実行履歴中から選択された行とまったく同じ部分を，読み込んだ FST ファイルから探す．

次に，探した部分の前後から，編集内容に適した行を判定し，既に FST ファイルに記述されている内容を編集したい内容に更新する．このとき，行の増減によって状態番号がずれ，遷移が上手くいかなくなってしまう場合があるので，そのずれを修正する．この修正後の状態番号は，FST ファイル内の使われていない番号を判定し，決定する．

最後に，更新された FST ファイルを再読み込みし，動作中の MMDAgent に反映させる．

4.2　対話追加機能

FST ファイル内に新しく対話を追加しようとした場合，反応するキーワード，返答，返答の声のトーン，モーションの 4 つの要素を決定することができれば残りの処理は，FST ファイル編集機能とおおむね同じである．そのため，4 つの要素を簡単に決定できるようなインターフェースを新しく提案する．以下に，実際の画面を図??に示す．



図 9　新しく対話を追加するための画面

キーワード及び返答を，テキストボックスを用いて入力できるようにし，声のトーン及びモーションをドロップダウンリストを用いて指定できるようにした．キーワードや返答などの，1 単語程度であれば，Android 端末上でタイプしたとしても誤字が少ないと考えた．また，声のトーンやモーションについては，実効履歴を用いた FST ファイルの編集と同様に選択式にすることで，FST ファイルの編集を簡易化した．

4.3 実効履歴ハイライト機能

実効履歴の表示方法として，以下の 3 つを提案する．

(1) 実行履歴をそのまま表示する
(2) 実行履歴中の処理の種類に応じて色分けを行なう
(3) 実効履歴中の対話ごとに色分けを行なう

それぞれの実装画面を以下の図 10，11，12 に示す．

4.3.1 実効履歴をそのまま表示する場合

今回，実効履歴をそのまま表示するというのは，実効履歴の文字色や背景色を一色のみで表示することを表す．図 10 に示すように実効履歴の文字色が一色であるため画面内に変化が無く，実効履歴が見やすいと感じることは少ないと考えた．

4.3.2 処理の種類に応じて色分けを行なう場合

図 11 に示すように，実効履歴を処理ごとに色分けすることで，色を見ただけで目的の行を発見できるようになると考えた．「RECOG_EVENT_STOP」を緑色，「MOTION_ADD」を青色，「TIMER_EVEMT」を淡青色，「SYNTH_EVENT」を黄色，その他を白色の合計 5 色で表示するようにした．

しかし，実効履歴に表示される色の種類が増えると，実効履歴が見づらくなってしまう場合が考えられるので，評価実験などによって適切な色の数を検討する必要があると考えられる．

4.3.3 対話ごとに色分けを行なう場合

図 12 に示すように実行履歴を対話ごとに色分けすることで，実行履歴中の対話がグループ化されるため，目的の行とは関係無い行に対する注意がなくなるためユーザの負担などが軽減され，目的の行を発見しやすくなると考えた．

プロトタイプシステムでは，対話と思われる行を緑色，それ以外を白色として表示したが，実行履歴中の対話部分が上手く強調されていないと思われるので，境界線など，より対話部分の強調を行なうように表示する必要があると考えられる．

5. プロトタイプシステム

提案手法によって，FST ファイルの編集を行なうことができるシステムを実装した．

MMDAgent はバージョン 1.5 を使用した．MMDAgent の FST ファイルを読み込む部分は C++で実装されているため，履歴ファイルを保存する機能は C++を用いて実装した．その他の，編集内容指定，FST ファイルへ出力，対話追加機能，実効履歴ハイライト機能については，Java[12] を用いて実装した．開発環境は AndroidSutdio[13] を用いた．

6. 評価実験と考察

6.1 目的

実際に，プロトタイプシステムを用いて FST ファイルを編集してもらい編集にかかった時間を測定する．さらに，パソコンを用いた FST ファイルの編集と比べてどれくら

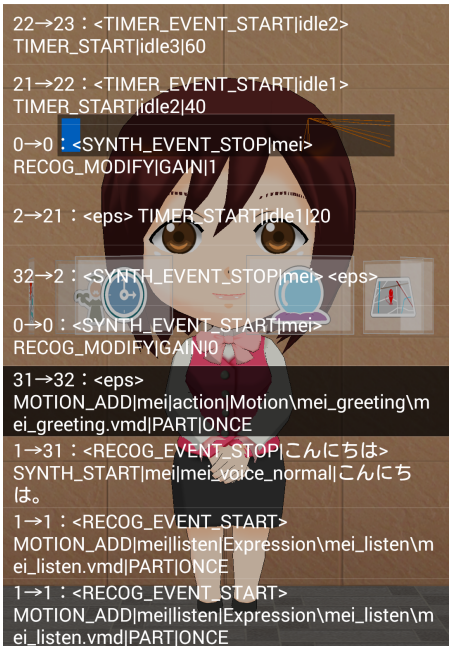


図 10 実効履歴が一色の場合

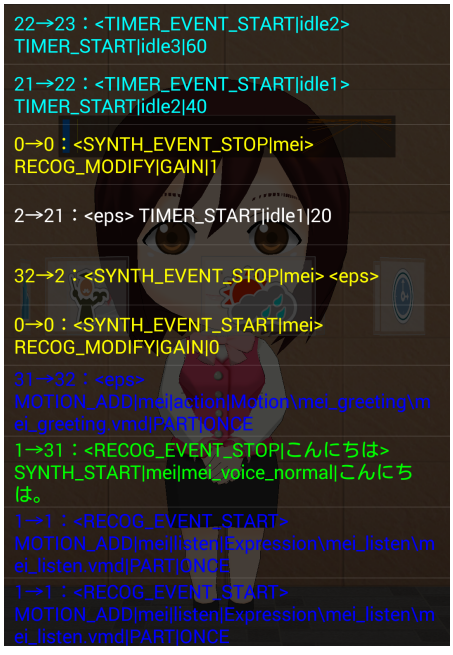


図 11 処理ごとに色分けを行なった場合



図 12 対話ごとに色分けを行なった場合

い編集が楽になったか，実行履歴は見やすいかなどをアンケートによって評価する．また，今回の評価実験では，実効履歴表示機能，FST ファイル編集機能，対話追加機能について行なった．

6.2 実験方法

今回，評価実験を行なった内容は，Android 端末内の FST ファイルにおいて，

(1) 返答にモーションがあるときに，そのモーションを違うモーションに変更してもらう場合
(2) 返答にモーションがないときに，モーションを追加してもらう場合
(3) 新しく対話を追加してもらう場合

の 3 つの場合において，被験者にそれぞれパソコンを使った編集と，提案システムを使った編集を行なってもらい，それぞれの編集にかかった時間を測定した．

評価実験を行なう際に使用した FST ファイルは, MMDAgent に元から保存されている FST ファイルから「おやすみ」に関する処理だけを追加したものとした．そのため，数千行記述されているような FST ファイルと比べると，今回の評価実験で使用した FST ファイルは比較的見やすいものであると言える．

3 つのキーワード「こんにちは」,「こんばんは」,「おやすみ」のモーションを変更，または追加してもらった．編集する 3 つのキーワードは順番を固定して編集してもらった．測定する時間は，3 つ全てのキーワードに対するモーションの編集が終わるまでの時間を計測する．また，それぞれのキーワードの編集前のモーションと，編集後のモーションは実験者側で指定する．対話を追加する場合については，実験者側で認識キーワード，モーション，声のトーンを指定し，返答についてはパソコンの編集と提案システムを用いた編集の両方で，同じ返答に編集するという条件の下, 被験者の自由とした．

6.3 実験結果と考察

それぞれの編集手法でかかった時間の計測結果とを以下の表 1,2 に示す．また，それらをグラフ化したもの及び，アンケート結果を以下の図 13,14,15,16 に示す．

表 1 モーションの変更にかかった時間の計測結果

被験者番号	従来の編集手順 [秒]	提案システム [秒]
1	251	69
2	169	58
3	240	65
4	369	106
5	257	83
6	195	62

表 1，2，3 から見ると明らかなように，往来の編集手法

表 2 モーションの追加にかかった時間の計測結果

被験者番号	従来の編集手順 [秒]	提案システム [秒]
1	332	58
2	310	48
3	295	51
4	349	76
5	424	91
6	201	51

表 3 対話の追加に掛かった時間の計測結果

被験者番号	従来の編集手順 [秒]	提案システム [秒]
1	136	47
2	270	68
3	176	65
4	172	43
5	314	69
6	124	40

ではモーションの変更に比べてモーションの追加にかかる時間が長くなるのに対し，提案システムを用いた編集ではモーションの変更と追加にかかる時間に殆んど差がない．これは，従来の編集手法では，モーションを変更する場合 FST ファイル内に元々書いてあるモーションに関する行の一部を変更すれば良く，モーションを追加する場合 FST ファイル内に新しくモーションに関する行を追加し，状態番号が使われていないかなどをチェックしなければならないためであるのに対し，提案システムを用いた編集では，モーションの変更，追加のどちらの場合でも同じように実行履歴のリストから選択するだけで良いからであると考えられる．よって，FST ファイルの編集において，FST ファイルに新しく 1 行を追加する作業というのは大変と感じる場合があると言える．

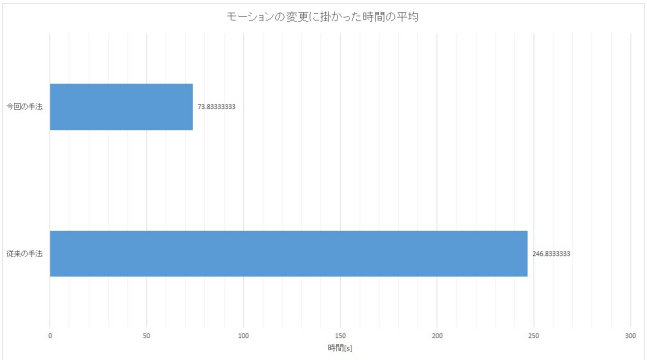


図 13 モーションを変更するのにかかった時間

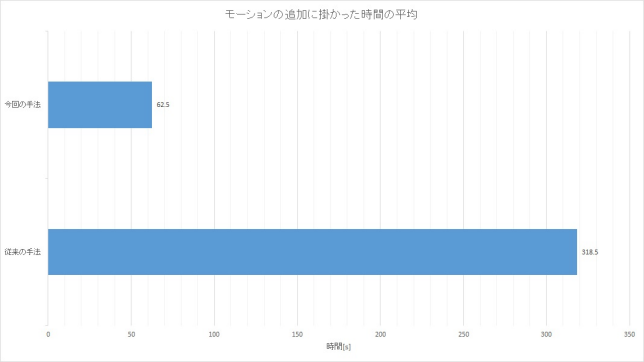


図 14　モーションを追加するのにかかった時間

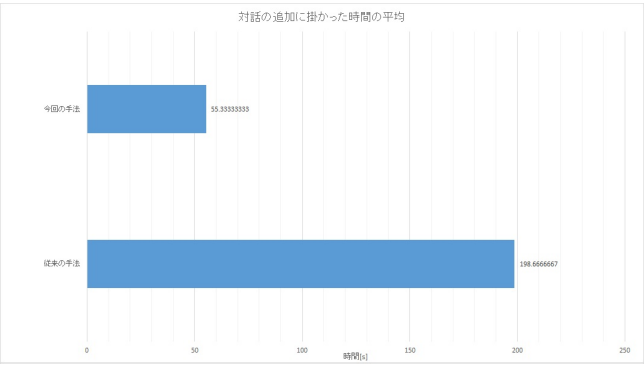


図 15　対話を追加するのにかかった時間

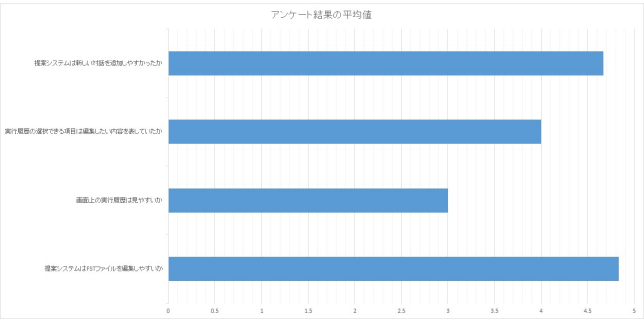


図 16　アンケート結果

表 1 及び図 13 より，従来の編集手法では，FST ファイル内の 3 つのモーションの変更にかかる時間はおよそ 4 分となっているのに対し，提案システムを用いたモーションの変更にかかる時間はおよそ 1 分程度である．これらの測定結果より，従来の編集手法に比べて提案システムでは，モーションの変更にかかる時間を約 70%削減することに成功したことが分かった．また，従来の編集手法でモーションの変更を行なったいるときの状態番号のミスなどによる誤編集は被験者全員が一回以上あったのに対し，提案システムによる FST ファイルの編集では誰もミスが出なかった．これらの結果より，従来の編集手法に比べて提案システムによる編集手法の方がモーションの変更が楽であると考えられる．

同様に，表 2 及び図 14 より，提案システムによるモーションの追加にかかる時間は，従来の編集手法に比べ約 80%削減されている．また，従来の編集手法でのミスは，被験者の二人に一人以上がミスをしたのに対し，提案システムによる FST ファイルの編集ではミスは出なかった．これらの結果より，従来の編集手法に比べて提案システムによる編集手法の方がモーションの追加が楽であると考えられる．

また，表 3 及び図 15 より，従来の編集手法では，1 つの対話を追加するのにおよそ 3 分ほどかかってしまうが，提案システムを用いた場合 1 分程度に短縮される．また，従来の編集手法で対話を追加する場合三人に一人程度がミスをしたのに対し，提案システムによる対話の追加ではミスは出なかった．

評価実験を行なっている際に，従来の編集手法において最も時間の掛かる要因と考えられたのは，入力イベントや出力コマンドの綴りを書き間違えることだった．綴りを書き間違えるた状態で Android 版 MMDAgent を起動すると，MMDAgent は動作しないため，もう一度パソコン上で FST ファイルを確認しなおし，編集を行なった箇所を正しく編集しなおさなければならない．そのため，FST ファイルの編集を行なう回数が増加し，データ転送などの回数も増加するため編集に時間がかかるようになると考えられる．しかし，提案手法ではタイピングなどによる FST ファイルの編集は殆んど行なわないため，誤った編集を行なったまま MMDAgent が起動されることは無いため，従来の編集手法に比べて編集時間が短くなったと考えられる．

また，提案手法では FST ファイルの編集方法が，基本的に選択式となっているため自分で FST ファイルを記述する必要が無く，リストの中から選択するだけでよいため編集時間が大幅に短くなったと考えられる．

更に，図 16 のアンケート結果から，「提案システムは FST ファイルを編集しやすいか」という質問に対しては，「編集しやすい」という解答が多かったことや，「実行履歴は見やすいか」という質問に対して「見やすい」という解答が少ないことから，実行履歴を用いることで FST ファイルの編集が楽になったとは言えないと考えられる．そこで，実行履歴をより見やすく改善する必要があると考えられる．

7. 関連研究

7.1 タブレット端末のための複合状態を用いた音声対話コンテンツ編集手法

タブレット端末のための複合状態を用いた音声対話コンテンツ編集手法 [10] では，FST ファイルの編集手法として，タブレットのタップ操作を用いて手軽に編集できるシステムを提案している．このシステムでは，よく使う一連の処理をテンプレートとして用意し，そのテンプレートを

用いることで手軽に FST ファイルの編集が可能となっている．しかし，既に書かれている処理に対して編集をしようとすると，編集したいと感じた部分を探すのに手間がかかってしまう場合がある．

7.2 音声対話コンテンツにおける状態遷移ログの可視化と簡単化手法

音声対話コンテンツにおける状態遷移ログの可視化と簡単化手法 [11] では，状態遷移ログと内部メッセージログを対応付けて状態遷移図として可視化することで，状態遷移ログの情報を直感的に得ることができるようになっている．さらに，状態遷移ログから得られた情報を見やすくするために，見かけ上の状態数を削減し簡単化している．しかし，Android 版 MMDAgent では実装されていないためパソコンを使った MMDAgent の編集が前提となっている．

8. おわりに

本論文では，Android 端末内の FST ファイルを編集するにあたって，Android 版 MMDAgent の FST ファイルを編集する際の問題を解決し，新しい FST ファイルの編集手法として，Android 画面上に実行履歴を表示し，その実行履歴を用いて FST ファイルを編集するいうシステムを提案した．提案した実現法をもとにプロトタイプシステムを実装し，実際に使用してもらうことで FST ファイルの編集において，モーションの変更について 80%—時間が削減された．モーションの追加について 70%時間が削減された．新しい対話の追加について 72%時間が削減された．これらの実験結果より，モーションの追加や変更，新しい対話の追加については，パソコンを用いた編集よりも編集時間が短くなることを実証した．

しかし，実行履歴が見やすいかという質問に対しては高い評価を得られなかった．そのため，実行履歴をキーワードや処理ごとに色分けし，より実行履歴を見やすくする手法を提案した．

今後の課題として，より実行履歴を見やすくするためにキーワードに対応する文字色を何色にするのが見やすいのか．また，実行履歴を処理ごとにグループ化する場合，どうすれば見やすくなるのかについて評価実験を行なうことが挙げられる．その他に，実効履歴上で検索機能やソート機能の実装などが挙げられる．

謝辞

本研究は JSPS 科研費 26330136，25700009 及び，科学技術振興機構 CREST の助成を受けたものです．
この場を借りて，感謝の意を表します．

参考文献

[1] AppleInc.,Siri，
入手先 ⟨http://www.apple.com/ios/siri/⟩（参照 2016-01-26）.

[2] NTT ドコモ, しゃべってコンシェル,
入手先 ⟨http://www.nttdocomo.co.jp/service/information/shabette_concier/⟩（参照 2016-01-26）.

[3] 李晃伸, 大浦圭一郎, 徳田恵一, 魅力ある音声インタラクションシステムを構築するためのオープンソースツールキット, 電子情報通信学会技術研究報告. NLC, 言語理解とコミュニケーション Vol.111,No.364,159-164,2011.

[4] 山本大介, 大浦　圭一郎, 西村良太, 打矢隆弘, 内匠逸, 李晃伸, 徳田恵一, スマートフォン単体で動作する音声対話 3D エージェント「スマートメイちゃん」の開発, インタラクション 2013,IPSJ Symposium Series Vol.2013,No.1,pp.675-680

[5] Akinobu Lee,Keiichiro Oura,Keiichi Tokuda,MMDAgent fully open-source toolkit for voice interaction systems Proceedings of the ICASSP 2013,pp.8382-8385,2013.

[6] Lee,A. and Kawahara,T.
Recent Development of Open-SourceSpeech Recognition Engine Julius, APSIPA,pp.131-137,2009

[7] 大浦圭一郎, 酒向慎司, 徳田恵一, 日本語テキスト音声合成システム OpenJTalk, 日本音響学会春季講論集,Vol.1,No.2-7-6,pp.343-344,2010.

[8] MikuMikuDance,
入手先 ⟨http://www.nttdocomo.co.jp/service/information/shabette_concier/⟩（参照 2016-01-26）.

[9] BulletPhysics,
入手先 ⟨http://www.nttdocomo.co.jp/service/information/shabette_concier/⟩（参照 2016-01-26）.

[10] 若林敬太郎, 山本大介, 高橋直久, タブレット端末のための複合状態を用いた音声対話コンテンツ編集手法, 丸手メディア, 分散協調とモバイルシンポジウム 2014 論文集 2014,pp.781-788,2014.

[11] 田中佑太朗, 山本大介, 高橋直久, 音声対話コンテンツにおける状態遷移ログの可視化と簡単化手法, マルチメディア, 分散協調とモバイルシンポジウム 2015 論文集 2015,pp.1263-1270,2015.

[12] Java,
入手先 ⟨http://www.oracle.com/⟩（参照 2016-02-04）

[13] AndroidStudio,
入手先 ⟨http://developer.android.com/intl/ja/index.html⟩（参照 2016-04-15）