VisualMimic: スクリーンショットを利用したGUI操作自動化のためのビジュアル開発環境

深堀 孔明　　坂本 大介　　五十嵐 健夫 *

概要. 本稿では，マウス操作やキー操作といったインタラクティブなアプリケーションに対する操作を自動化するためのスクリプト開発環境 Visual Mimic を提案する．本システムでは，ユーザは操作したい GUI 要素をスクリーンショット画像で指定することで，特定の API に依存しない GUI 操作を記述できる．具体的には，ユーザはまず自動化したい GUI 操作を例示により記述する．その後，生成されたスクリプトはグラフベースのビジュアルエディタ上に表示され，ユーザはドラッグ＆ドロップでスクリプトを編集する．結果として，ユーザは従来の開発環境に比べて少ない労力で GUI 操作を自動化することが可能になる．

1 はじめに

GUI アプリケーションや PC ゲームのようなインタラクティブなアプリケーションは，ユーザがマウス操作やキー操作といった GUI 操作を行うことで動作する．そのため，GUI アプリケーションの動作テストや，RPG ゲームにおけるキャラクタのレベル上げといった単調な作業を行う場合でも，常にユーザがパソコンを操作しつづける必要があり，ユーザに大きな負担がかかってしまう。Automator [1] のような GUI 操作を自動化するためのツールもあるが，専用の API を利用して各アプリケーションのインタフェースを操作することで実現しており，Flash ゲーム内のボタンのような，ユーザが独自に作成した GUI 要素への操作は記述できない．

一方，Sikuli [2] という GUI 操作の自動化ツールでは，操作したい GUI 要素をそのスクリーンショット画像で指定する．システムは指定された画像をデスクトップ画面から探し，マッチした領域に対して GUI 操作を実行する．このように GUI 要素の見た目を利用することで，任意の GUI 要素に対する操作を記述できる．しかし，Sikuli では各操作をテキストベースのスクリプトとして記述する必要があり，プログラミングが得意でないユーザにとって負担が大きいという問題がある．

そこで本研究では，Sikuli をエンドユーザ向けに改良した開発環境 Visual Mimic を提案する（図 1）．ユーザはまず、自動化したい GUI 操作を例示により記述する。生成されたスクリプトはグラフベースのビジュアルエディタ（以下キャンバスと呼ぶ）に表示され，ユーザはスクリプトをドラッグ＆ドロップで編集できる．本研究の最大の貢献は，例示プログラミングとビジュアルプログラミングを用いることで，ユーザが最小の労力で Sikuli と同様な GUI 操作の自動化スクリプトを作成できるようにしたことである．

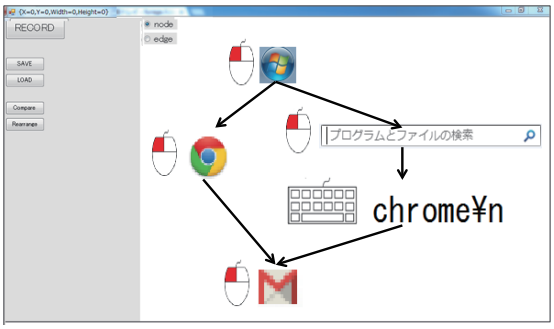

図 1. VisualMimic のインタフェース

2 Visual Mimic

ユーザが図 1 左上の録画ボタン（”RECORD”）を押すと，システムはバックグラウンドでユーザのマウス・キー操作の監視を始める．ユーザがマウスカーソルを動かすと，システムは現在マウスカーソルが接触している GUI 要素を自動で検出し，赤色の矩形でハイライトする（図 4-a 左）．ユーザがマウスクリックをすると，現在ハイライトされている GUI 要素をクリックする命令がキャンバス上に追加される（図 4-a 右）．同様に，ユーザが文字列をタイプすると，キー操作命令がキャンバスに追加される．

なお，マウスカーソルが接触している GUI 要素の検出には，デスクトップ画面の画素変化を利用している．マウスカーソルが動いた瞬間に，カーソル周辺で色が変化した領域を GUI 要素として識別する（図 2）．システムが望ましい領域をハイライトしない場合は，カーソルを大きく円状に動かすジェスチャによって，手動で領域を指定するモードへ移行することもできる（図 3）．

* Koumei Fukahori, Daisuke Sakamoto, Takeo Igarashi, 東京大学

図 2. GUI 要素の自動検出アルゴリズム

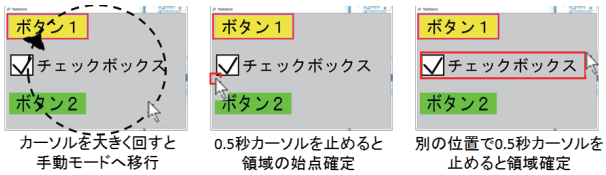

図 3. ハイライト領域の手動指定

例示により記述されたスクリプトはキャンバス上にグラフベースのビジュアル言語として表示される．各ノードはひとつの GUI 操作，エッジはスクリプトの処理の流れを表す．ユーザは任意のノードからスクリプトの実行を開始でき，システムはエッジにそってノードをたどり，対応する GUI 操作を順次実行する．例えば，図 4-b の一番上のノードが実行されると，システムはまずデスクトップ画面からボタン 1 の画像を探し，マッチした領域の中央にマウスカーソルを動かし，左クリックを行う．画像マッチングには OpenCV の *cvMatchTemplate()* 関数を利用しており，指定画像とのマッチ率が 80 %を超えた領域に対して各操作を行う．

ユーザはノード間にエッジを追加・削除することでスクリプトの処理の流れを編集できる（図 4-b）．枝分かれを作った場合，現在実行可能な GUI 操作の方へ処理が進む．例えば，ボタン 1 のクリック操作が終わったら，システムはデスクトップ画面からチェックボックスを探し，チェックがついている場合は左側のノードに，ついていない場合は右側のノードに処理が進む．また，後退辺を加えることで処理をループさせることができる．

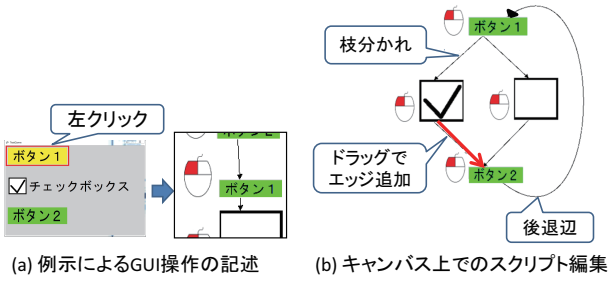

図 4. VisualMimic 上でのスクリプトの作成

3 利用例

図 5 は VisualMimic を用いて作成したスクリプトの例である．図 5-a のスクリプトでは，Google Chrome から GMail を開き，WISS の投稿システムから届いたメールをアーカイブする．また，図 5-b は YouTube 動画をリピート再生するスクリプトであり，動画の再生が終わるたびにリプレイボタンを自動でクリックする．これらのスクリプトでは処理の流れが矢印で表示されており，テキストベースのスクリプトと比べてスクリプトの構造が理解しやすくなっている．また，Sikuli のようにユーザがソースコードを書く必要がなく，比較的少ない手数でスクリプトを記述できる．例えば，図 5-b のスクリプトは 6 回のマウスクリックだけで作成できた．

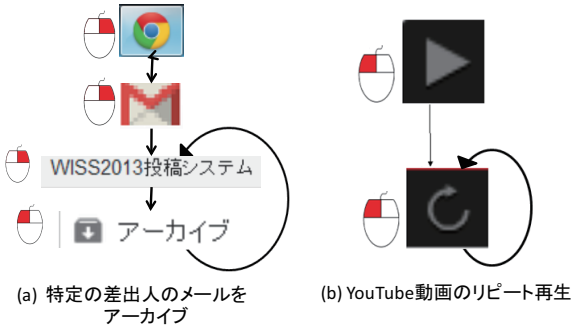

図 5. VisualMimic で作成したスクリプトの例

4 今後の課題

現状，本システムは変数を扱う仕組みを用意しておらずチューリング完全でない．したがって今後は，変数を扱うためのインタフェースをキャンバス上に実装する予定である．それに伴って for 文や if 文といった構文をキャンバス上で記述できるようにすることで，本システムの高機能化を進める．

また，本システムの GUI 要素の検出手法は完全ではない．現在の実装では，デスクトップ画面の色変化を利用して GUI 要素を識別するため，GUI 要素が色変化しない場合や，カーソルが動いた瞬間に GUI 要素でないオブジェクトがたまたま色変化した場合，本手法は正しく動作しない．今後は単純な色変化だけではなく，他のデスクトップ画面の画像特徴量も利用した GUI 要素の検出手法について検討していきたい．

参考文献

[1] Automator. http://support.apple.com/kb/HT2488.

[2] T. Yeh, T.-H. Chang, and R. C. Miller. Sikuli: using GUI screenshots for search and automation. In *Proc. UIST '09*, pp. 183–192, 2009.