Automação de Tarefas Por Meio de Reconhecimento de Imagens Sintéticas em Interfaces Gráficas

Horst E. R. Erdmann, Paulo S. Rodrigues
Centro Universitário da FEI — São Bernardo do Campo — Brasil
horstao@gmail.com, psergio@fei.edu.br

Abstract

This paper presents a study and a metodology to automate tasks in graphical interface programs, thus allowing the repetition of a step sequence performed by the user during his interaction with a software. The proposed study consists of three phases: Capture, Analisys and Reproduction. In the Capture phase, the steps performed by the user are recorded. In the Analisys phase the objects that have been interacted by the user are extracted, a quantized array with the features of each object is created and each object is then classified. In the third phase, the objects are searched in the screen and suffer the interaction recorded on the first phase once they are found.

Resumo

Este trabalho apresenta um estudo e uma metodologia para automação de tarefas em programas com interfaces gráficas, permitindo a repetição de uma seqüencia de passos realizada por um usuário durante sua interação com um software. A metodologia proposta é constituída de três etapas: Captura, Analise e Reprodução. Na etapa de Captura, as interações com o sistema operacional são gravadas bem como uma imagem da tela no momento da interação. Na fase de analise, as imagens gravadas são processadas para extração dos objetos de interesse e suas propriedades. Os objetos então são classificados para tornar sua busca mais robusta na terceira etapa. Finalmente, durante a reprodução os objetos são procurados na tela e sofrem a interação gravada na primeira fase quando encontrados.

1 Introdução

Existem diversas tarefas realizadas em softwares que são passíveis de serem automatizadas. Entre elas se destacam o preenchimento de um formulário eletrônico por diversas vezes para o cadastro de uma grande quantidade de informações em um sistema novo. Desenvolvedores podem também desejar automatizar algumas ações para executar testes regressivos em novas versões do software que estão desenvolvendo. [2].

Atualmente, existem dois tipos de ferramentas que possibilitam a automação de processos em programas. A primeira, grava as interações do usuário de forma "cega", ou seja, são gravadas as posições e cliques do mouse, além das teclas digitadas e os tempos entre cada interação. Geralmente, este tipo de automatizador deixa de funcionar se o tempo de resposta da aplicação mudar ou se os objetos mudarem de posição. Por sua vez, o segundo tipo utiliza as funcionalidades do Sistema Operacional (SO) para encontrar os objetos gráficos na tela e realizar as ações desejadas através de suas propriedades. Este apresenta um resultado melhor na detecção dos objetos em relação ao primeiro, uma vez que não depende da posição dos objetos na tela, mas sim de suas propriedades. No entanto, este tipo de automatizador é restrito a alguns sistemas operacionais proprietários e, a detecção correta dos objetos depende diretamente da linguagem em que o programa-alvo (programa cuja função se deseja automatizar) foi escrito.

2 Método Proposto

Tendo em vista as limitações dos dois tipos de ferramentas de automação citadas acima, a execução "cega" do primeiro e a dependência do SO e da linguagem do programa-alvo no segundo, este trabalho propõe um método alternativo de automação de processos onde o reconhecimento de objetos em interfaces gráficas é feito exclusivamente através de características visuais como cor, forma e textura.

O método proposto é composto de três fases: Captura, Análise e Reprodução. Na primeira etapa, denominada Captura, para cada interação i realizada pelo usuário, são armazenados os cliques do mouse e a digitação do teclado em um vetor I_i enquanto o *printscreen* no momento exato da interação é gravado em S_i, assim como as coordenadas do mouse preenchem o vetor P_i.

A segunda etapa deste processo é conhecida com Análise. Nesta etapa, cada *printscreen* é analisado em con-

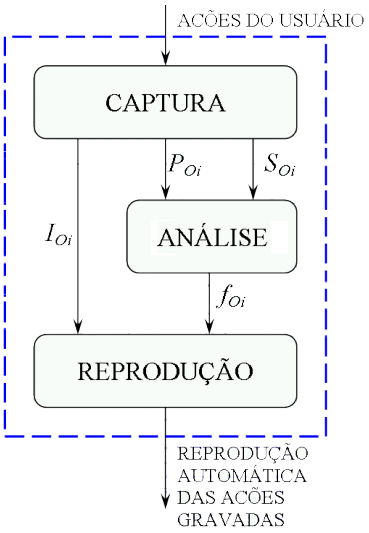

Figura 1. Etapas envolvidas no método proposto e os vetores resultantes ao final de cada etapa. O processo inteiro deve ser capaz de capturar as interações de um usuário e reproduzi-las de forma robusta ao final.

junto com a coordenada do mouse ou alteração do teclado naquele instante. Essa análise permite o isolamento do objeto O_i com o qual houve a interação i. Com o objeto O_i separado do resto da imagem, é criado um vetor $f_{O_i} = \{c_1, c_2, \ldots c_3\}$ com as características que foram extraídas do objeto. A partir deste momento, f_{O_i} irá representar, de forma quantizada, o objeto O_i. Com f_{O_i} é possível fazer a classificação de O_i em uma das classes definidas na Tabela 1.

Classe	Objetos
C1	Botões
C2	Ícones
C3	Caixas de Texto
C4	Áreas de Texto

Tabela 1. Classes de Objetos

Na terceira etapa, chamada de Reprodução, para cada uma das i interações que foram gravadas, O_i é procurado na tela através da comparação de f_{O_i} com as características extraídas de todos os objetos que são exibidos na tela. Quando O_i é encontrado, sofre a mesma interação I_i definida durante a gravação.

3 Objetos de Detecção

Em uma interface gráfica, existem diversos tipos diferentes de objetos. Cada um destes objetos possui características singulares que os diferenciam dos demais.

3.1 Botões

Botões (Figura 2a) geralmente são retangulares, possuem uma única cor e palavras simples no centro. Porém, estas características não são predominantes e podem mudar na medida em que novas versões do programa-alvo forem desenvolvidas. A forma pode mudar de um retângulo com ângulos retos para outro com bordas suavizadas. A cor de fundo também pode mudar, deixando de ser homogênea para ser um degradê de cores. O texto do botão também pode variar de diversas formas, desde a aplicação de *anti-aliasing*, alteração do tipo de fonte ou substituição do texto por uma palavra mais adequada a sua função.

3.2 Ícones

Ícones (Figura 2b) são pequenas figuras com um conteúdo semântico bem definido, cuja imagem está diretamente relacionada com a ação executada pelo sistema quando o objeto é ativado. Geralmente, esses objetos não variam muito em forma, cor ou textura, mas quando são trocados (em uma nova versão do programa-alvo, por exemplo), podem ter todas as suas propriedades completamente alteradas de modo a inviabilizar seu reconhecimento, a menos que seja analisado seu significado semântico. Porém, neste trabalho, não será estudada a interpretação do significado semântico de cada ícone uma vez que este tema, por si só, é suficientemente complexo para ser estudado de forma exclusiva.

3.3 Caixas de Texto

Caixas de Texto (Figura 2c), assim como os botões, são em sua maioria objetos retangulares e sua cor de fundo mais comum é branco e, geralmente, não possuem texto antes da receberem dados de entrada. As alterações que estes objetos podem sofrer são variações em sua largura, variação do tipo de fonte do texto e o uso de *anti-aliasing* para as fontes. O maior obstáculo para a detecção correta deste objeto porém é a contextualização, ou seja, a capacidade do algoritmo de associar uma caixa de texto aos objetos vizinhos a ela quando existe mais de um destes elementos na tela. Este fato é comum em formulários onde se possui várias caixas de texto com características visuais exatamente iguais, porém, os dados que cada uma representa são definidos pelos textos ao lado das caixas.

3.4 Áreas de Texto

Áreas de Texto (Figura 2d) são objetos dentro dos quais podem existir textos longos. Este tipo de objeto geralmente possui fundo branco, assim como as caixas de texto. Dentre as alterações que as áreas de texto podem sofrer estão as variações de largura e altura, conforme a janela que envolve a área de texto em questão é redimensionada, maximizada ou restaurada. Além disto, uma área de texto pode ter seu conteúdo interno em fontes de tipos, tamanhos e cores diferentes.

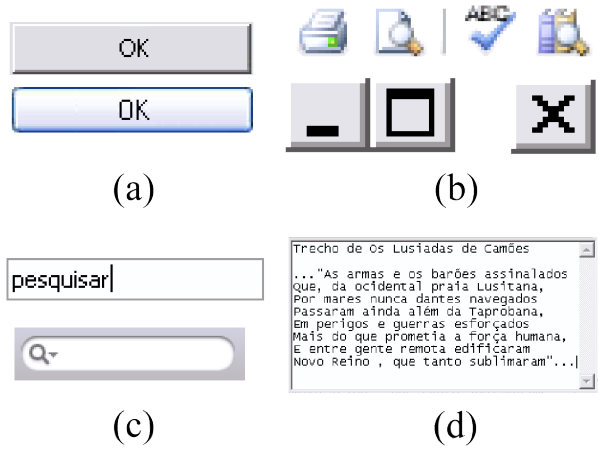

Figura 2. Exemplos de objetos em interfaces gráficas. botões em (a), ícones em (b), caixas de texto em (c), uma área de Texto (d).

4 Técnicas Utilizadas

4.1 Separação do Objeto de Interesse

A separação do objeto de interesse O_i com o qual houve interação é a parte mais crítica desta proposta. Uma boa separação inicial proporciona informações mais apuradas sobre os objetos e, conseqüentemente, torna o *matching* mais preciso.

Durante a etapa de gravação, são criados três vetores: o vetor de *screenshots* $S_i = \{s_1, s_2, \ldots s_i\}$, o vetor de coordenadas $P_i = \{p_1, p_2, \ldots p_i\}$ e o vetor de ações realizadas $I_i = \{a_1, a_2, \ldots a_i\}$ onde são guardadas as informações de cada ação realizada pelo usuário. Para cada interação i gravada, são gravados um *screenshot* S, uma coordenada P e uma ação I.

A primeira tarefa a ser realizada na etapa de analise é a extração do objeto O_i a partir da imagem S_i e da coordenada P_i. O algoritmo para segmentação de S_i foi selecionado através da comparação dos resultados dos algoritmos k-means ([9], [1]), entropia ([14], [5], [12]), modelos deformáveis ([8], [6], [15]) e ainda limiar adaptativo ([10]). A Figura 3 mostra a comparação entre os algoritmos de segmentação estudados. A técnica de limiar adaptativo oferece bons resultados para imagens sintéticas a um baixo custo computacional, justificando seu uso neste trabalho. Após a segmentação de S_i, aplica-se um algoritmo de detecção de contornos que resulta em um vetor com as coordenadas de cada contorno encontrado Vc. Estes contornos são filtrados através de uma função de seleção condicional. Os resultados desta função são as coordenadas dos limites de O_i.

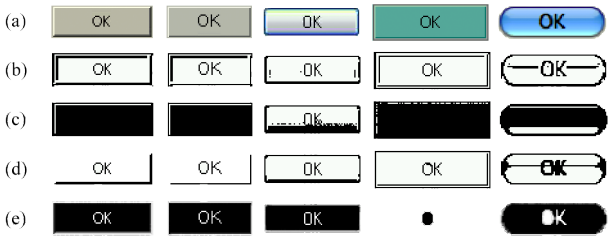

Figura 3. Comparação entre os algoritmos de segmentação estudados.Em (a), cinco imagens de botões. Em (b),(c),(d) e (e), o resultados das segmentações das imagens utilizando respectivamente limiar adaptativo, k-means, entropia clássica e modelos deformáveis.

4.2 Dados de Treinamento

Para a classificação dos objetos de interesse é necessária uma base de dados inicial com exemplos de objetos de cada *cluster*. A performance de um algoritmo classificador supervisionado depende diretamente da base de dados utilizada para seu treinamento ou parametrização. Neste aspecto, o ideal é uma base de dados com grande volume de objetos e bastante variação entre objetos da mesma classe de modo a representar a maioria dos objetos encontrados em situações reais.

Para criação da base de dados que será utilizada neste trabalho, serão extraídos *printscreens* de seis fontes de dados independentes, cada uma com um sistema operacional diferente e com cinqüenta aplicações instaladas. Será utilizada uma rotina de navegação aleatória que abre cada aplicativo e simula o clique do mouse em posições randomicas para extrair o *screenshot* quando ocorre uma alteração na tela.

4.3 Classificação dos Objetos

Após a criação de f_{O_i} com as principais características de O_i, é possível utilizar uma função de clusteirização $FC(O_i)$ para classificar O_i em uma das sete classes listadas na Tabela 1 de acordo com os valores de f_{O_i}. A classificação de f_{O_i} pode ser feita através de redes Bayesianas ([7], [11]), redes neurais ([3], [16]) ou simplesmente através do cálculo da divergência entre f_{O_i} e a centróide G_x de cada *cluster*.

4.4 Comparação entre Objetos

O reconhecimento do objeto será feito através da comparação entre f_{O_i} e as features extraídas de cada um dos n objetos exibidos na tela $T = \{O_1^T, O_2^T, \ldots O_n^T\}$, que resulta em um vetor $M = \{D(f_{O_i}, f_{O_1^T});\ D(f_{O_i}, f_{O_2^T});\ \ldots D(f_{O_i}, f_{O_n^T})\}$. Ordenando-se M, encontra-se a menor distância entre objetos encontrados na tela. Esta distância deve ser menor que um limiar para que o reconhecimento seja realizado. Sem esta verificação, o resultado do *matching* não é O_i mas sim o objeto mais semelhante a O_i caso não haja outro.

Existem diversas técnicas para a medida de distâncias, entre dois vetores. Segundo Linda em [13], as principais classes de medidas são as de cor, textura e forma. Porém, com a representação de todos estes atributos em um vetor, é possível fazer uso de qualquer medida de distância para cálculo de similaridade entre vetores. A medida de similaridade entre vetores pode ser calculada através da distância de *Minkowski* e da distancia de *Kullback-Leiber*, também conhecida por entropia relativa [11].

A distância de *Minkowski* utilizada em [11] é uma generalização da distancia euclidiana e é sensível à deslocamentos de histograma, fenômeno este, que ocorre principalmente quando há alteração de iluminação ou ruído. em imagens sintéticas, como as estudadas neste trabalho, estes dois fatos ocorrem em menor escala, viabilizando o uso desta técnica. A distância de *Minkowski* entre dois vetores p e q é calculada através da Equação (1).

$$D_M(p,q) = \left[\sum_{i=1}^{n} |p_i - q_i|^x\right]^{1/x} \tag{1}$$

O cálculo da similaridade entre dois vetores através das distancias euclidiana e de *Minkowski* possui uma fragilidade conhecida como erro da normalização relativa que ocorre quando o espaço vetorial envolve vários atributos heterogêneos que, embora normalizados, os atributos com menores intensidades podem ser suplantados por aqueles nas maiores faixas, independentemente de sua capacidade discriminativa.

De uma forma mais forma, pode-se calcular a distância dois vetores $\partial = [f_1, f_2, \ldots f_n]$ e $\partial^{'} = [f_1^{'}, f_2^{'}, \ldots f_n^{'}]$, onde f_i, $f_i^{'} \in [0, 1]$. Denomina-se a faixa de *features* com menores valores por ϵ_A e a faixa com maiores valores por ϵ_B. Se ϵ_A for suficientemente próximo de 0 e ϵ_B suficientemente próximo de 1, a distância euclidiana, será principalmente induzida por ϵ_B. Se, por acaso, os vetores associados ao espaço ϵ_A forem os mais discriminantes, isso torna os vetores associados a ϵ_B falsos discriminantes.

É interessante notar que este problema também ocorre se for usado para análise multivariada para determinar as características mais discriminantes no espaço determinado por ϵ_A e ϵ_B. Nesse caso, o problema ocorre porque os autovalores e autovetores calculados normalmente com o PCA também são altamente dependentes de ϵ_A e ϵ_B.

Uma possível solução para este problema é composta da utilização de *features* representadas por distribuições probabilísticas ao invés de valores absolutos e do cálculo da distância entre duas distribuições probabilísticas f e f' através da entropia relativa ou distância de *Kullback-Leiber* conforme a Equação (2) [4].

$$D_{KL}(f \| f') = \sum_x f(x) log \frac{f(x)}{f'(x)} \tag{2}$$

A Equação (2) trata da divergência entre duas distribuições probabilísticas. Quando existem mais de uma distribuição probabilística com interdependência entre elas, o cálculo da divergência entre dois vetores é realizado através do uso da regra da cadeia aplicada à Equação (2) e possui complexidade $O(n^{|f|})$ para uma quantidade de features n com tamanho $|f|$. O alto custo computacional inviabiliza o uso deste algoritmo em tempo real. Porém, os objetos de estudo deste trabalho são sintéticos e nestes casos não há interdependência entre as *features* de forma que é possível a utilização da somatória das divergências entre as distribuições probabilísticas como medida de similaridade entre objetos. Sendo assim, a divergência entre dois vetores de distribuições probabilísticas $f_{O_i} = \{f_{O_i}^1, f_{O_i}^2, \ldots f_{O_i}^n\}$ e $f_{O^T} = \{f_{O_i^T}^1, f_{O_i^T}^2, \ldots f_{O_i^T}^n\}$, com um total de n distribuições cada, é obtida através da soma das divergência entre cada uma de suas distribuições conforme Equação (3).

$$D(f_{O_i}, f_{O^T}) = \sum_{x=1}^{n} D_{KL}(f_{O_i}^x \| f_{O^T}^x) \tag{3}$$

Este algoritmo tem complexidade $O(n|f|)$ para uma quantidade de features n com tamanho $|f|$ e seu resultado tem domínio em $\{0,\ n\}$, uma vez que cada $D_{KL}(f_x \| f'_x)$ pode assumir valores entre zero e um.

5 Resultados Obtidos

Neste trabalho, foram realizados dois testes para avaliar a performance dos métodos de extração do objeto e de classificação do mesmo. O algoritmo de extração do objeto de interesse foi avaliado através de sua execução para extrair quinhentos objetos de cada classe, cada um a partir de um *screenshot* e uma coordenada onde houve o clique. Os resultados dos testes de performance com o algoritmo de extração podem ser vistos na Tabela 2.

Classe	Extrações Corretas	Performance
Botões	401	80%
Ícones	480	96%
Caixas de Texto	470	94%
Áreas de Texto	498	99%

Tabela 2. Resultado da aplicação do método proposto para extração de objetos

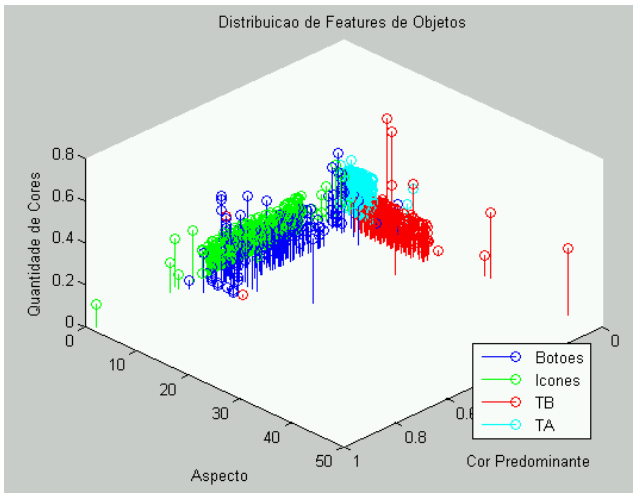

Figura 4. Representação da distribuição das características dos objetos de cada classe. Botões em azul, Ícones em verde, caixas de texto em vermelho e áreas de texto em ciano

Como pode-se observar na Tabela 2, o algoritmo de extração de objetos conseguiu extrair uma quantidade maior de ícones, caixas de texto e áreas de texto corretamente. Isto se deve ao fato de que estes elementos possuem bordas bastante diferentes do *background*, facilitando a extração destes elementos. Os botões que foram extraídos de forma errada, apresentaram este resultado devido à coordenada da interação estar exatamente em cima do texto do botão, fazendo com que a imagem extraída seja somente a letra e não o objeto. Por sua vez, o resultado da extração de textos foi baixo principalmente porque os caracteres não são conectados, sendo assim, só foram extraídos corretamente os textos cujas palavras estavam sublinhadas.

Além da avaliação do algoritmo de extração de objetos, foi também avaliado classificador através da distância euclidiana entre um objeto e a centróide de cada classe. Com uma base de dados inicial com trezentos elementos das classes Botões, Ícones, *Textboxes* e *Textareas*, foram extraídas quatro features de cada objeto. A média entre as *features* de cada classe define a centróide do *cluster*, ou seja, o valor médio de cada feature para cada classe. Na Figura 4, estão representadas três features de cada elemento de cada classe.

Após o cálculo da centróide G_C de cada classe C, a classificação de um objeto O é obtida através da distancia euclidiana entre O e G_C. Na tabela 3 estão relacionados os resultados da classificação para cada *cluster*. Foram extraídas as *features* de duzentos elementos de cada classe, diferentes daqueles utilizados para cálculo das centróides G_C. A performance do algoritmo de classificação é obtida por meio da contagem de elementos classificados corretamente. Este algoritmo teve uma performance excelente na classificação de ícones, principalmente devido a feature f_4, aspecto, que em ícones apresenta valores próximos de um.

Classe	C_{f_1}	C_{f_2}	C_{f_3}	C_{f_4}	Perf.
Botões	0,365	0,582	0,122	4,329	65,2%
Ícones	0,364	0,308	0.114	0.958	99.5%
Caixas de Texto	0.053	0.750	0.111	12.633	78.5%
Áreas de Texto	0.012	0.826	0.144	4	78,5%

Tabela 3. Resultado da classificação de objetos através da distancia euclidiana

6 Conclusões

Os algoritmos implementados para realização da extração dos objetos de interesse e da classificação destes objetos ainda não obtiveram a performance desejada. Para a classificação, porém, é necessário o levantamento de outras características com maior capacidade discriminante.

Referências

[1] L. Bastos, P. Liatsis, and A. Conci. Automatic texture segmentation based on k-means clustering and efficient calculation of co-occurrence features. In *Systems, Signals and Image Processing, 2008. IWSSIP 2008. 15th International Conference*, pages 141–144, 2008.

[2] P. Bernardo and F.Kon. A importância dos testes automatizados. *Engenharia de Software Magazine*, 3:54–57, 08.

[3] C. Bishop. *Neural Networks for Pattern Recognition*. Oxford, 1995.

[4] T. Cover and J. Thomas. *Elements of Information Theory*. John Wiley and Sons, Inc., 1991.

[5] I. Esquef. Tecnicas de entropia em processamento de imagens. Master's thesis, Centro Brasileiro de Pesquisas Fisicas, 2002.

[6] C. Gentile, O. Camps, and M. Sznaier. Segmentation for robust tracking in the presence of severe occlusion. *Image Processing, IEEE Transactions*, 13:166–178, 2004.

[7] F. Jensen and T. Nielsen. *Bayesian Networks and Decision Graphs*. Springer, 2007.

[8] M. Kass, A. P. Witkin, and D. Terzopoulos. Snakes: Active contour models. *International journal of Computer Vision*, pages 321–331, 1988.

[9] A. Niemisto, T. Korpelainen, R. Saleem, O. Yli-Harja, J. Aitchison, and I. Shmulevich. A k-means segmentation method for finding 2-d object areas based on 3-d image stacks obtained by confocal microscopy. In *Engineering in Medicine and Biology Society, 2007. EMBS 2007. 29th Annual International Conference of the IEEE*, pages 5559–5562, 2007.

[10] M. Pi and H. Zhang. Two-stage image segmentation by adaptive thresholding and gradient watershed. In *CRV '05: Proceedings of the 2nd Canadian conference on Computer and Robot Vision*, pages 57–64. IEEE Computer Society, 2005.

[11] P. Rodrigues. *Um Modelo Bayesiano Combinando Análise Semântica Latente e Atributos Espaciais para Recuperação de Informação Visual*. PhD thesis, Universidade Federal de Minas Gerais, 2003.

[12] P. Rodrigues, G. Giraldi, R. Chang, and J. Suri. Object tracking in image sequence combining hausdorff distance, non-extensive entropy in level set formulation. *Topics in Biomedical Engineering International Deformable Models*, pages 477–515, 2005.

[13] L. Shapiro and G. Stockman. *Computer Vision*. Prentice Hall, 2000.

[14] A. Tavares. Aspectos matemáticos da entropia. Master's thesis, Universidade de Aveiro, 2003.

[15] P. Tissainayagama and D. Suterb. Object tracking in image sequences using point features. *Pattern Recognition*, 38:105–113, 2005.

[16] C. Walt and E. Barnard. Data characteristics that determine classifier performance. *SAIEE Africa Research journal*, 98:87–93, 2007.