Captura de Histórico de Imagens

Abstract. When processing an image, common processing systems don't worry about capturing the image history. Capturing the image history might be very helpful in image processing. In this work, we are presenting a processing system that capture and reuse image history. This system is divided in 4 modules: history generator, history processor, history database and history conversion. The image processor platform is Python and the common history language is XML.

JULIANA CRISTINA BRAGA, GERALD JEAN FRANCIS BANON
INPE - Instituto Nacional de Pesquisas espaciais
São José dos Campos - SP - Brasil
{juliana, banon}@dpi.inpe.br

1 Introdução

Um dos problemas enfrentados por usuários que desenvolvem experimentos em sistemas computacionais, é a falta de uma ferramenta que monitore, capture e armazene automaticamente os passos, funções e parâmetros utilizados em seus experimentos. O registro dessas informações é chamado de histórico do processamento [2],[3].

Este problema é particularmente crítico para usuários de sistemas de processamento de imagem. Neste caso, as imagens são geradas sem uma documentação completa o que, muitas vezes, inviabiliza seu reaproveitamento. Podemos listar alguns benefícios que o armazenamento do histórico viabiliza para usuários de "softwares" de processamento de imagens:

Documentação precisa e padrão: atualmente, o histórico do processamento de imagem, se realizado, é feito manualmente, possivelmente incompleto, estando sujeito a erros. Além disso, cada usuário possui o seu modo de documentar uma imagem, ficando difícil a troca de informações entre usuários diferentes. Com a geração de histórico automático essa documentação torna-se precisa e padronizada facilitando assim, a consulta e troca de dados.

Reaproveitamento de processamentos existentes: muitas vezes, ao lermos um artigo científico sobre determinado processamento, temos a necessidade de reproduzi-lo para dar continuidade ao mesmo. Essa reprodução seria eficaz consultando-se o histórico desse processamento e repetindo os passos nele descritos.

Adaptação de processamentos existentes: através do histórico pode-se também repetir um processamento bem sucedido com novos dados fontes, ou então testar a influência de um parâmetro sobre o resultado final. Para isso, basta trocar as referências dos dados ou parâmetros no histórico e processá-lo novamente. Em outras palavras, o usuário teria a opção de programar novos processamentos a partir de exemplos de outros tratamentos já executados.

Economia de espaço em disco: certas imagens podem ser vistas como resultados intermediários dentro de uma cadeia de processamento, neste caso cada uma delas deveria ser automaticamente apagada depois de terminado o último tratamento que as envolve, deixando-se mais espaço para o armazenamento de futuros tratamentos. Caso necessite usar uma dessas imagens novamente, é suficiente consultar seu histórico para saber como a mesma foi obtida e gerá-la outra vez.

Rapidez na transmissão de dados: com as tecnologias atuais, a troca de informações através de Internet cresce a cada dia. Transmitindo somente os históricos, ao invés da imagem, haveria uma sensível economia de tempo na transmissão de dados.

Globalização das informações: a existência do histórico implica em uma documentação que poderá ser facilmente divulgada, seja através de artigos, "e-mails", páginas de Web e outras mídias. Esta é uma maneira de evitar a repetição de esforços de pesquisa e facilitar sua continuidade.

Tendo em vista a importância de armazenar o histórico de um processamento, objetivou-se com esse trabalho: propor um Gerenciador de Histórico que crie e utilize o histórico de um processamento; implementar um protótipo para testar o uso do Gerenciador; e testar o protótipo em aplicações de processamento de imagens.

Nas seções a seguir, apresenta-se a noção de histórico e as descrições do Gerenciador de Histórico e do protótipo desenvolvido.

2 Noção de histórico

A noção de histórico tem dois aspectos, cada um deles relacionado ao conceito matemático de mapeamento.

Considera-se agora o primeiro mapeamento. Seja *B* um conjunto de frases de uma certa linguagem. Estas frases podem ser geradas por uma gramática do tipo forma normal de Backus-Naur (BNF) [11].

Por exemplo, as seguintes regras definem uma gramática:

element = "1" | "2" | "3" | "4" | "5" | "6" |
max = "∨" "(" *sentence* "," *sentence* ")"
min = "∧" "(" *sentence* "," *sentence* ")"
inv = "Γ" "(" *sentence* ")"
sentence = *element* | *max* | *min* | *inv* | *sentence*

As seguintes expressões são exemplos de frases nesta linguagem:

2

Γ(2)

3

∧(Γ(2), 3)

Observa-se que a última frase é uma composição das anteriores. A Figura 1 ilustra este fato.

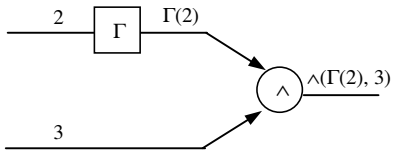


Figura 1 – Composição de frases.

Admite-se agora que a cada frase da linguagem pode se associar um único "*element*" através de um operador chamado exec. Em outros termos, o operador exec é um mapeamento de *B* em *C* = {1, 2, 3, 4, 5, 6}, isto é, exec: *s* ∈ *B* ↦ *e* ∈ *C*.

Se *e* ∈ *C*, então por definição, uma frase *s* de *B* *é um histórico para e*, se e somente se, exec(*s*) = *e*.

A

Figura 2 ilustra o primeiro mapeamento

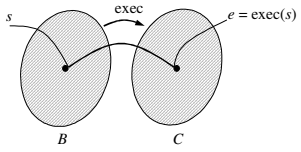


Figura 2– Mapeamento exec.

No exemplo anterior, admite-se que as frases elementares: *element*, *max*, *min* e *inv* são processadas da seguinte maneira:

exec(*e*) = *e*

exec(∨ (*a*, *b*)) = max{*a*, *b*}

exec(∧ (*a*, *b*)) = min{*a*, *b*}

exec(Γ(*e*)) = 6 - *e*

onde *e*, *a*, *b* ∈ *C*.

Desta forma o resultado do processamento do exemplo da Figura 2 fica como mostrado na Figura 3 :

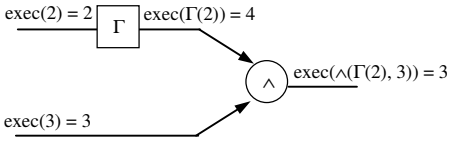


Figura 3– Execução de frases.

Neste trabalho a linguagem será o Python e o operador exec será o processador Python.

Considera-se agora o segundo mapeamento. O segundo mapeamento estabelece o critério para se dar nomes reservados às frases de uma dada linguagem. Os nomes serão úteis para referenciar frases e armazená-las.

Criado uma frase, dá-se um nome reservado, isto é, que nunca foi usado antes e que nunca mais será usado. Para isto, escolhe-se de forma apropriada um nome dentro de um conjunto infinito de nomes. Seja hist o mapeamento do conjunto *A* de nomes para o conjunto *B* de frases.

Seja *n* ∈ *A*, então por definição, uma frase *s de B* *é o histórico de n* se e somente se *s* = hist(*n*).

Desta forma, a cada nome escolhido corresponderá um único histórico. Procurar-se-á, na medida do possível, que nomes diferentes correspondam a históricos diferentes. Neste caso, o mapeamento hist seria injetora ("onde-to-one").

Um nome para um histórico *s* servirá de nome para o local onde estará sendo armazenado *s* e o resultado de sua execução.

A Figura 4 mostra a composição dos dois mapeamentos introduzidos anteriormente.

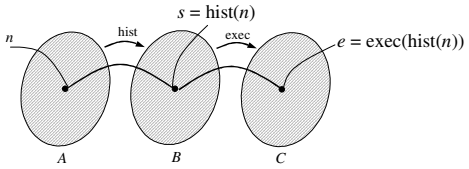


Figura 4 – Composição dos mapeamentos hist e exec.

Definido o mapeamento hist é possível nomear às frases da Figura 1 como mostrado na Figura 5.

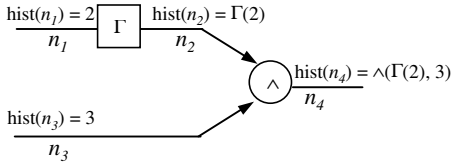


Figura 5 – Alocação de frases aos nomes.

Observa-se que hist(n_4) pode ainda ser escrito como:

hist(n_4) = ∧(hist(n_2), hist(n_3)) hist(n_3) = 3

hist(n_2) = Γ(hist(n_1)) hist(n_1) = 2

Isto é, a partir dos nomes n_1, n_2, n_3 seria possível "calcular" (efetuando substituição) o histórico de n_4.

3 Gerenciador de Histórico

O Gerenciador Histórico foi baseado na noção de histórico descrito na seção anterior. Ele deverá monitorar um sistema de processamento em busca de informações para a criação do histórico. Além disso, o Gerenciador permitirá o armazenamento e manipulação de históricos existentes.

O Gerenciador possui arquitetura Cliente/Servidor. Do lado do cliente encontram-se o usuário do sistema e o navegador ("browser") e do lado do servidor encontram-se os 4 módulos que fazem parte do Gerenciador: Módulo Criação do Histórico; Modulo Processamento do Histórico; Módulo Banco de Histórico; e Módulo Conversão de Histórico.

A seguir é dada uma descrição de cada módulo.

3.1 Módulo de Criação do Histórico

O Módulo de Criação do Histórico tem como funções: monitorar um sistema de processamento visando capturar as operações e operadores de processamento utilizados pelo usuário do sistema; e registrar as operações e operadores capturados, gerando assim, o histórico do processamento.

O histórico é registrado na linguagem nativa de desenvolvimento do módulo, e é chamado de histórico nativo. No exemplo ilustrado na Figura 5, o histórico nativo poderia ser registrados na seguinte forma simplificada:

$hist(n_4) = \wedge(n_2, n_3)$ $\quad n_4 = \wedge(n_2, n_3)$
$hist(n_2) = \Gamma(n_1)$ $\quad n_3 = 3$
$hist(n_3) = 3$ $\quad n_2 = \Gamma(n_1)$
$hist(n_1) = 2$ $\quad n_1 = 2$

ou ainda,

Sugere-se que o histórico seja inserido dentro de um campo de metadado específico para descrição de imagens como, por exemplo, o modelo "Metadado para Documentação e Recuperação de Imagens [8].

Como indicado na seção anterior, nomes reservados servem para identificar os históricos gerados. O uso de nomes reservados permitirá que o histórico seja único em qualquer escopo, podendo ser utilizado globalmente sem que haja repetição de nome (ou seja, o mesmo nome utilizado para mais de um histórico). Desta forma, o Gerenciador proposto poderá se inserir em uma arquitetura cliente/servidor. No entanto, como os nomes reservados são expressões extensas, o usuário poderá dar nomes de sua conveniência, chamados apelidos, aos históricos. Estes serão automaticamente renomeados com nomes reservados.

Formação dos nomes reservados

Para a formação dos nomes reservados, segue-se a regra empregada na definição do conceito de "Uniform Repositoy for a Library" (UR*Lib*) [1].

O nome dado a um histórico é formado da seguinte maneira:

<apelido>__<domainname>__<username>__<year>_
<month><day>_<hour>_<minute>[_<second>],

onde: apelido é um nome sugerido pelo usuário na hora de definir um novo processamento; domainname é o nome de domínio onde pertence a máquina do usuário; username é o nome do usuário que está conduzindo o processamento; year, month e day formam a data e hour, minute e second formam o horário de criação do histórico.

O Gerenciador aplica a regra de formação de nomes para históricos nativos, transformando os apelidos em nomes reservados. O histórico que contém nomes reservados é chamado de *histórico uniforme*.

Sejam a_1, a_2, a_3 e a_4 os apelidos dos históricos $hist(n_1)$, $hist(n_2)$, $hist(n_3)$, $hist(n_4)$ da Figura 5. Chamamos de histórico simplificado de $hist(n_4)$ o histórico com estes apelidos. Este é mostrado a seguir:

$a_4 = \wedge(a_2, a_3)$ $\quad a_3 = 3$
$a_2 = \Gamma(a_1)$ $\quad a_1 = 2$

O mesmo histórico uniforme ficaria:

a_4_dpi.inpe.br_juliana_2003_03_10_13_45=∧
(a_2_dpi.inpe.br_juliana_2003_03_10_13_30,
a_3_dpi.inpe.br_juliana_2003_03_10_13_40)
a_3_dpi.inpe.br_juliana_2003_03_10_13_40=3
a_2_dpi.inpe.br_juliana_2003_03_10_13_30=Γ(a_1
dpi.inpe.br_juliana_2003_03_10_13_15)
a_1_dpi.inpe.br_juliana_2003_03_10_13_15=2

3.2 Módulo de Processamento do histórico

O módulo Processamento do Histórico tem como funções: Alteração ou edição de históricos nativos e execução total ou parcial do processamento registrado nos mesmos.

Antes de executar novamente o histórico, o processador verifica se este foi ou não alterado pelo usuário. Se não alterado, o processador lê o histórico, verifica a disponibilidade dos dados envolvidos no processamento e realiza todo ou parte do processamento descrito no mesmo. Se alterado, o processador verifica a coerência dos dados que sofreram modificações e executa o processamento indicado no histórico alterado.

3.3 Módulo Banco de Histórico

O módulo Banco de Histórico armazena dados e seus respectivos históricos em uma biblioteca digital na Web chamada UR*Lib*. Uma vez armazenados na UR*Lib* [1] os dados ficarão disponíveis na Web para busca através de palavras-chaves e recuperação via download.

3.4 Módulo Conversão de Histórico

Este módulo deverá transformar o histórico nativo em uma linguagem extensível ao XML [4]. Sendo assim, o módulo oferecerá facilidade de uso e manipulação de histórico. Além disso, o módulo permitirá a transformação do histórico para outras linguagens como: HTML, C++, Java, etc. Para tanto, fará uso da XML "Stylesheet Language Transformations" (XSL) [5] e "Shemas" XML [6].

4 Protótipo

O protótipo está em fase de desenvolvimento. Os módulos Criação de Histórico e Banco de histórico estão em fase final e o módulo Processamento de Histórico está em fase inicial. O módulo Conversão de Histórico ainda não foi desenvolvido.

A linguagem de desenvolvimento do protótipo é o Python [7]. Python é uma linguagem interpretada, bastante portável, orientada a objetos. [7]

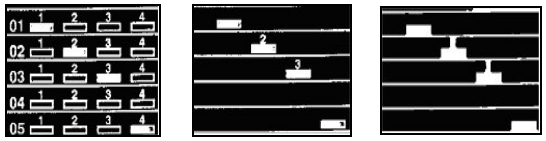
O protótipo está sendo desenvolvido na arquitetura Cliente/Servidor. Este utiliza potencial da UR*Lib* que, através do servidor Apache [10], permite carregar e

descarregar arquivos "on-line". Optou-se por Python e UR*Lib* pois eles possuem código aberto e têm distribuição gratuita, fato que beneficia a comunidade científica.

5 Resultados

O protótipo foi testado sobre o processamento de imagem mostrado na Figura 6

Observa-se que a imagem img1 sofre uma operação de abertura resultando na imagem img2, que por sua vez, sofre uma operação de fechamento resultando na imagem img3.



(a) (b) (c)

Figura 6 – (a) imagem img1; (b) imagem img2; (c) imagem img3.

Os históricos gerados foram registrados como mostrado a seguir:

```
import adpil; import morph;
img3__dpi_inpe_br__juliana__2003__07_04_12_58
=morph.mmclose(img2__dpi_inpe_br__juliana__200
3__07_04_12_57_52,morph.mmsedisk(4));
img2__dpi_inpe_br__juliana__2003__07_04_12_57
_52=morph.mmareaopen(img1__dpi_inpe_br__julia
na__2003__07_04_12_57,500);
img1__dpi_inpe_br__juliana__2003__07_04_12_57
=adpil.adread('form1.tif');
```

Os Metadados de img3 mostram algumas tags adicionais. Essas tags obedecem ao padrão da UR*Lib* e descrevem informações importantes para busca e captura do histórico na Web. Observa-se que o histórico de img3 é uniforme e nativo. Sendo assim, informações sobre domínio, usuário, data e hora são visíveis nesse histórico. O histórico reservado é muito grande e por isso o protótipo disponibiliza, quando possível, o histórico simplificado mostrado a seguir:

```
import adpil;
import morph;
img3=morph.mmclose(img2,morph.mmsedisk(4));
img2=morph.mmareaopen(img1,500);
img1=adpil.adread('form1.tif');
```

O histórico mostrado foi armazenado na UR*Lib* no endereço:
http://hermes.dpi.inpe.br:1905/rep/dpi.inpe.br/juliana/2003/07.04.12.57.52. Para acessar o histórico, clicar no link "referência completa" e para acessar img3 clicar no link "Exemplo de aplicação de uma abertura seguida de um fechamento morfológico".

A recuperação do histórico pode ser feita utilizando a palavra chave: history img3* a partir da página inicial da UR*Lib*: http://hermes.dpi.inpe.br:1905.

6 Conclusões

Desenvolveu-se neste trabalho um Gerenciador de Histórico. Para testá-lo foi implementado um protótipo na linguagem Python.

O Gerenciador mostrou-se eficiente para geração, armazenamento e processamento de histórico de imagens. Considerou-se a globalização do histórico um ponto forte do trabalho, o qual só foi possível pelo uso de nomes reservados.

Pretende-se após o término deste trabalho dar continuidade nos seguintes aspectos:

- Permitir a alteração consistente do histórico para posterior processamento.
- Desenvolver a linguagem HML e gerar o histórico nesta linguagem.
- Transformar o histórico em HML para outras linguagens de programação.
- Testar o uso do Gerenciador de histórico para outras aplicações além do processamento de imagem.
- Acrescentar ao modelo metadados que contenham informação a respeito do conteúdo semântico dos dados.

7 Referências

[1] Banon, Gerald Jean Francis. *What is URLib*?: Disponível em repositório da UR*Lib*: <iconet.com.br/banon/2001/02.10.22.55>

[2] Banon, G, J, F. *Implementação de um sistema de tratamento de imagens usando uma definição ampla de imagens*; III Simpósio de Sensoriamento Remoto, Rio de Janeiro, RJ, 28 a 30 de novembro de 1991.

[3] Banon, G, J, F. A service for APL2 Global Variable Manipulation. Centro científico Brasília-Brasil. Relatório técnico. Maio de 1986.

[4] Especificação XML: *http://www.w3.org/TR/XML/*

[5] Especificação XSL: *http://www.w3.org/TR/XSL/*

[6] Especificação de Schemas: *http://www.w3.org/Tr/xmlschema*

[7] G. Rossum, *Python Tutorial*, F. L. Drake Jr., editor, April 15, 2001.

[8] Garcia, Simone de Souza; Moura, Ana Maria Carvalho; Campos, Maria. Luiza Machado Campos. *Metadados para Documentação e Recuperação de imagens*; Relatório Técnico RT 041/DE9/Abr99, IME.

[9] Home page oficial do Python: www.python.org

[10] Home page oficial do Apache: h*ttp://www.apache.org*

[11] P.Naur (ed.), *Revised report on the algorithmiclanguage Algol 60*, Comm. ACM 6 (1963), 1-17 e Comput. 1-5(1963), 349-367.