Trabalho Final de Linguagens Formais e Autómatos

Daniel Magueta, Francisco Teixeira, Gil Teixeira, Gonçalo Arieiro, Jorge Oliveira, Mário Liberato

Universidade de Aveiro

Contents

Trabalho Final de Linguagens Formais e Autómatos

Este documento serve como suporte para a realização da documentação do trabalho, bem como o relatório contendo informação sobre as funções da linguagem e funções auxiliares sob a forma de plugins.

Classe Image

Foi criada uma classe Image que guarda uma imagem (*elem*), o caminho (*path*), o nome (*name*), e a sua extensão (*extension*). O nome e a extensão são obtidos através de funções internas que concatenam o caminho. Esta classe foi criada para conseguirmos guardar o nome e a sua extensão e ao efetuar uma cópia da imagem esta permanecer com parte do nome e extensão originais.

Save

Guarda uma imagem dando a variável da mesma como argumento. O ficheiro é guardado no directório do programa e contém o nome copy_of_"nome da imagem"."extensão", no caso de existirem cópias então será adicionado um número para não sobrescrever o ficheiro.

Exemplo: copy_of_imagem (2) .jpg

Show

Abre uma certa imagem numa janela com resolução igual o inferior a 1280 x 720, fechando a janela ao pressionar qualquer tecla. O título da janela contém o nome da imagem.

Gray

Converte uma dada imagem para escala de cinzentos.

Blur

Desfoca a imagem usando *Gaussian Blur* aceitando valores de 1 a 100. No entanto os valores pares são reduzidos em 1 devido à forma da função.

Crop

Recorta uma imagem dando as coordenadas do ponto que será o canto inferior esquerdo do resultado e dando as dimensões de largura e altura do recorte;

Exemplo: crop (x, y, largura, altura, imagem)

Nota: Caso as dimensões de largura e altura da imagem crop ultrapassem as dimensões da imagem então retornará um recorte do máximo possível sem exceder os limites da imagem.

Brightness

Altera o brilho da imagem aceitando valores de 0 a 100 sendo o predefinido 50 ou seja, valores menores a 50 reduz o brilho e superiores vão aumentar. Utiliza recurso de uma função *"colours"* para obter este resultado;

Contrast

Altera o contraste da imagem aceitando valores de 0 a 300 sendo o predefinido 100. Valores inferiores a 100 reduz o contraste e vice-versa. Recorre à função auxiliar "colours" para obter este resultado;

Rotate

Roda a imagem mantendo a resolução da mesma aceitando uma variável inteira sendo os graus da rotação da imagem. No código Python são aceites valores negativos, no entanto estes não são utilizados na nossa gramática;

Resize

Modifica as dimensões da imagem utilizando os 2 argumentos dados para a nova largura e altura. No código Python não são aceites valores menores ou iguais a 0 apesar de no caso da nossa linguagem apenas surgir o caso de um valor ser 0;

Scale

Função semelhante a resize porém apenas aceita um argumento que será uma percentagem de 1 a 1000% sendo o predefinido a 100%. Esta função modifica a resolução da imagem mantendo as suas proporções, o que poderá ser de maior utilidade (em termos de simplicidade) ao programador;

Edges

Esta função retorna uma imagem com os contornos encontrados na imagem sendo uma imagem de fundo preto com os contornos a branco;

Extract

Extrai as caras de uma dada imagem guardando-as em formato "*jpg*";

Difference

Retorna a diferença (das cores) entre duas imagens. Caso as imagens não tenham a mesma resolução então será criada uma imagem com a resolução equivalente à menor altura e largura das duas imagens dadas;

Convert

Esta função converte uma imagem dada para um certo tipo, dado sob a forma de um valor. Existem dois de tipos de conversões: Conversão do "colour space" e remoção de certos "canais" de cores. O "colour space" predefinido é BGR. De seguida são encontradas as possíveis conversões com o seu código:

- **1 -** Tons de vermelho
- **2 -** Tons de verde
- **3 -** Tons de azul
- **4 -** Tons de cinzento
- **5 -** *"Colour space"* alterado para **HLS**
- **6 -** *"Colour space"* alterado para **LUV**
- **7 -** *"Colour space"* alterado para **XYZ**
- **8 -** *"Colour space"* alterado para **RGB**
- **9 -** *"Colour space"* alterado para **YUV**
- **12 -** Tons de vermelho e verde
- **13 -** Tons de vermelho e azul
- **23 -** Tons de verde e azul

Funções Auxiliares (presentes no código Python)

Colours

Esta é uma função de suporte para a modificação de brilho e contraste da imagem, aceita 2 variáveis: "A" correspondente ao contraste e "B" que corresponde ao brilho.

Filter

Processa uma imagem visitando todos os pixéis e alterando os valores de azul, verde e vermelho de cada um. Esta função é utilizada por *"convert"* quando se trata da alteração dos tons de cor.

Open

Função para abrir uma imagem dando o *"path"* da mesma, é utilizada ao criar uma variável imagem.

Copy

Função que copia um objeto da Classe Image. A função Copy é usada em diversas funções para evitar modificar a imagem original no caso de ser dada uma modificação de uma imagem a outra variável.

Exemplo (Python): img = open("imagem.png")
img2 = blur(img,50)

Gestão de erros

Se for dada um argumento inválido, como por exemplo, números negativos ou fora do alcance dos valores da função, então é escrito um erro na consola, mas a função retorna a imagem dada sem qualquer alteração.

Gerador de Código

O gerador de código é uma peça que "encaixa" entre o *parser* e o produto final. É capaz de gerar código em *Python* ou *Java*, precisando apenas dos plugins adequados para cada função da linguagem. Estes plugins contêm informação sobre cada função da linguagem, incluindo o nome, número e nome dos argumentos, linguagem e implementação da função na linguagem-alvo. É criado um objecto gerador de código, e depois são chamadas as suas funções para construir o programa final. O gerador gere todas as dependências de funções e bibliotecas externas (cada plugin reporta as suas dependências). Se uma função ou biblioteca for necessária, a função/*import* será adicionada ao código final. Finalmente, obtém-se o código final na linguagem-alvo.

É de se notar que, embora o *LFACodeGenerator* suporte Python e Java, apenas plugins para Python foram incluídos.

Manual de Instruções

Na pasta Linguagem Final existe um ficheiro "linguagemfinal.txt" com exemplos da nossa linguagem. Esta linguagem pode ser feita em blocos usando "{}" onde temos a vantagem de isolar variavéis ou então sem usando blocos. Uma imagem pode ser atribuída a uma variável da seguinte forma: a = "nome da imagem" ou então a = "caminho para a imagem"; de seguida podemos trabalhar a imagem através da variável, e sempre que chamamos uma função colocamos parênteses, como por exemplo show(), open(), gray(), etc. Sempre que terminamos uma tarefa colocamos ";" no final. No ficheiro temos mais exemplos práticos de como a nossa linguagem funciona.

Contribuição dos autores

Gil Teixeira fez algumas funções de OpenCV.

Francisco Teixeira fez a gramática original.

Gonçalo Arieiro e **Daniel Magueta** fizeram a gramática final e o listener para essa gramática, bem como outras funções de OpenCV.

Mário Liberato e **Jorge Oliveira** fizeram o gerador de código utilizando um sistema de plugins, bem como os plugins para esse fim polindo as funções em Python.

Percentagems da contribuição para o trabalho realizado

Gil Teixeira:

Francisco Teixeira:

Gonçalo Arieiro:

Daniel Magueta:

Mário Liberato:

Jorge Oliveira: