Introdução à Programação e Resolução de Problemas

2010/2011

Trabalho Prático
Tratamento de imagens em Python

Jorge Granjal,
Tiago Baptista, Ernesto Costa, Luís Paquete,
Francisco C. Pereira

Projecto 1

Imaginemos

1.1 Introdução

Pretende-se com o presente Trabalho Prático implementar uma aplicação em Python que disponibilize diversas operações de tratamento de imagens. Para além dessas operações, o programa deverá permitir a leitura e a gravação das imagens a partir de ficheiro. Dado que o nosso objectivo não é lidar com formatos de imagem complexos e com compressão, tais como o JPEG ou TIFF, iremos utilizar o formato de imagem PPM (Portable Pixel Map). Este formato permite armazenar imagens de forma bastante simples, tal como passamos a descrever.

1.2 O formato de imagem PPM

Uma imagem pode ser armazenada em ficheiro de forma bastante simples, recorrendo ao formato PPM. Passamos a descrever o formato PPM no modo ASCII, que deverá ser utilizado pela aplicação sempre que for necessário ler ou gravar uma imagem a partir de ficheiro.

Um ficheiro PPM em modo ASCII começa por uma linha com a palavra "P3", que identifica o formato PPM nesse modo. A segunda linha do mesmo ficheiro armazena um comentário, normalmente utilizado para identificar a aplicação responsável pela criação do ficheiro. A terceira linha armazena dois valores inteiros, separados por um espaço: o primeiro valor representa a largura da imagem e o segundo a sua altura. A quarta linha do ficheiro armazena um valor inteiro, que indica o valor máximo de cor presente na imagem. A partir da quinta linha, são armazenados os valores dos compo-

nentes RGB (Red, Green, Blue) de cada píxel da imagem. Por exemplo, a quinta linha armazena o valor de **vermelho** do primeiro píxel, a sexta linha o valor de **verde** do mesmo píxel e a sétima linha o seu valor de **azul**. As 3 linhas seguintes no ficheiro armazenam os valores RGB do segundo píxel da imagem, e assim sucessivamente. O exemplo seguinte ilustra as primeiras 10 linhas de um ficheiro PPM no formato ASCII. Trata-se neste exemplo de uma imagem criada pela aplicação GIMP e com dimensão de 200x123 píxeis:

```
P3
# CREATOR: GIMP PNM Filter Version 1.1
200 123
255
133
120
117
123
134
212
...
```

As imagens no formato PPM podem ser visualizadas recorrendo a várias aplicações, entre as quais o popular GIMP (pode ser obtido em http://www.gimp.org). Este programa poderá ser utilizado como auxiliar na preparação de imagens para testar o correcto funcionamento da aplicação.

A seguir descrevem-se as operações a implementar na aplicação. Começaremos por descrever as operações gerais, relativas ao tratamento de dados e ficheiros, e em seguida as operações de processamento de imagens. As operações de processamento de imagens encontram-se divididas em operações fundamentais e operações avançadas.

1.3 Operações gerais relativas ao processamento de dados, ficheiros e interface com o utilizador

1.3.1 Leitura e armazenamento de imagens a partir de ficheiro

O programa deve permitir o carregamento de imagens a partir de ficheiro, cujo nome deve ser solicitado ao utilizador. O programa deve igualmente permitir o armazenamento das imagens modificadas em ficheiro, novamente utilizando o nome indicado pelo utilizador. Para armazenar a imagem em memória deverá utilizar a estrutura de dados que considerar mais adequada. É importante notar que todas as operações que envolvam a modificação da imagem devem ser implementadas exclusivamente com recurso à estrutura de dados que representa a imagem na memória do computador, ou seja, os ficheiros são utilizados apenas para carregar em memória uma imagem ou para salvaguardar uma nova versão da imagem após a sua alteração.

1.3.2 Operações de tratamento de imagens

Tal como referido no ponto anterior, as operações de modificação de imagens devem ser implementadas exclusivamente em memória. Em particular, a imagem original deve ser carregada a partir de um ficheiro para memória, onde deve ficar armazenada numa estrutura de dados de tipo considerado adequado. As operações de modificação de imagens devem ser implementadas com recurso às estruturas de dados que armazenam a imagem ou, se necessário, a estruturas de dados auxiliares. Quando o utilizador o solicitar, o programa deverá permitir o armazenamento da imagem activa (eventualmente entretanto modificada) novamente em ficheiro.

1.3.3 Visualização de imagens

O programa deverá permitir visualizar a imagem carregada a partir de ficheiro, bem como a imagem alterada após cada operação de modificação, em janelas separadas. A visualização das imagens deve ser implementada com recurso exclusivamente às funcionalidades da *PIL* (Python Imaging Library), em particular recorrendo às seguintes funções do módulo *Image* dessa biblioteca:

new	Criar uma nova imagem
putpixel	Modificar o valor RGB de um píxel numa determinada posição
show	Afixa a imagem

Note que não deverá utilizar qualquer outra função do módulo Image, ou qualquer outra função de tratamento de imagens disponível noutros módulos do Python. As funções descritas no quadro anterior são suficientes para a afixação de uma imagem a partir da sua representação em memória, dados os valores das componentes de cor (RGB) de cada píxel da imagem.

O objectivo do presente trabalho, tal como foi referido anteriormente, passa pela implementação das operações de modificação de imagens recorrendo directamente à(s) estrutura(s) de dados que representam uma imagem na memória do computador, sem utilizar portanto funções de modulos externos ao seu programa.

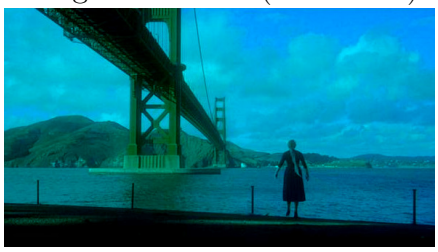
1.3.4 Interface para selecção de comandos

O programa deve disponibilizar ao utilizador um menu que permita a selecção das operações de tratamento de imagem disponíveis. A selecção de cada opcção deve desencadear as operacções necessárias, tais como por exemplo o pedido dos nome(s) dos ficheiro(s) ou outros valores necessários ao tratamento da imagem com a operação seleccionada.

1.4 Operações fundamentais de processamento de imagem

1.4.1 Alteração de vermelhos, verdes e azuis

A alteração de componentes específicos de cor permite implementar diversas operações de tratamento de imagens. Pretende-se que o utilizador possa seleccionar qual dos 3 componentes (vermelho, verde ou azul) de cor pretende alterar em toda a imagem, bem como a respectiva percentagem de alteração. No exemplo seguinte a a imagem original foi modificada através da alteração da componente vermelha:

Imagem original	Imagem alterada (vermelhos)
	

1.4.2 Criação de moldura

Pretende-se com esta operação criar uma moldura preta em redor da imagem. O utilizador deve poder seleccionar a largura da moldura. Note que a imagem original deve ser preservada integralmente, ou seja, nenhum píxel da imagem original deve ser descartado ao introduzir a moldura. No exemplo seguinte a imagem original foi modificada através da introdução de uma moldura com 5 pixeis de largura:

Imagem original	Imagem com moldura (5 pixeis)
	

1.4.3 Passagem a tons de cinza

Uma imagem em tons de cinza pode ser definida como uma imagem que tem valores iguais nas componentes vermelha, azul e verde de cada píxel . Esta operação deverá permitir a conversão de uma imagem a cores numa imagem em tons de cinza. Esta operação pode ser implementada substituindo cada píxel por outro calculado a partir da média dos valores do píxel original. No exemplo seguinte a imagem original foi modificada através da sua passagem a tons de cinza:

Imagem original	Imagem em tons de cinza
	

1.4.4 Rotação da imagem

O programa deverá permitir rodar a imagem original 90, 180 ou 270 graus, de acordo com o sentido de rotação indicado pelo utilizador (sentido dos ponteiros do relógio ou em sentido inverso). No exemplo seguinte a imagem original é rodada 180 graus:

Imagem original	Imagem rodada 180 graus
	

1.4.5 Reflexão de imagem

O programa deverá permitir reflectir horizontal ou verticalmente uma imagem. No exemplo seguinte a nova imagem é obtida por reflexão horizontal da imagem original:

Imagem original	Reflexão horizontal da imagem
	

1.4.6 Corte de imagem

O corte de uma imagem com o objectivo de seleccionar uma área de interesse é uma operação frequentemente utilizada nos programas de tratamento de imagens. Pretende-se desta forma permitir que o utilizador possa cortar a imagem original, de acordo com uma região seleccionada. O utilizador deverá seleccionar a zona da imagem a preservar indicando as coordenadas dos píxeis correspondentes aos cantos superior esquerdo e inferior direito dessa zona na imagem original. No exemplo seguinte a imagem final foi obtida por corte da imagem original, seleccionando como imagem pretendida a zona na imagem original definida pelas coordenadas 200,200 (coordenadas do píxel no canto superior esquerdo) e 500,10 (coordenadas do píxel no canto inferior direito):

Imagem original	Imagem obtida por corte
	

1.4.7 Negativo de imagem

O negativo de uma imagem pode na prática ser obtido calculando o negativo dos 3 componentes de cor que definem cada píxel da imagem. No exemplo seguinte a imagem original foi modificada através da sua passagem a negativo:

Imagem original	Negativo da imagem
	

1.5 Outras operações de processamento de imagem

1.5.1 Esteganografia

As técnicas de esteganografia permitem em geral ocultar informação considerada secreta, utilizando para o efeito mensagens aparentemente inócuas. Os dados que descrevem uma determinada imagem podem igualmente ser manipulados no sentido de ocultar uma mensagem secreta, de forma totalmente transparente. Pretende-se portanto que o programa ofereça uma opcção para ocultar uma mensagem de texto num ficheiro de imagem, ou em alternativa ler uma mensagem secreta armazenada de forma oculta num ficheiro de imagem.

A técnica a utilizar para ocultar uma mensagem secreta passa por utilizar os bits menos significativos de cada píxel (ou das 3 componentes de cor do píxel) para armazenar a mensagem. De facto, é possível verificar que a mudança dos bits menos significativos de cada componente de cor ou píxel afectam muito pouco a imagem final, o mesmo não podendo ser dito dos bits mais significativos. Se for alterado apenas o bit menos significativo em cada componente de cor ou píxel da imagem, é de facto muito difícil ou mesmo impossível distinguir a imagem alterada da original. Por exemplo, é muito difícil distinguir entre os píxeis com o valor 10000000 (128 em decimal) e 10000001 (129 em decimal). Já alterações a bits mais significativos são facilmente perceptíveis. Por exemplo, a diferença entre os valores 00001000 (8 em decimal) e 10001000 (136 em decimal) já é perceptível.

A técnica a implementar consistirá portanto na alteração do bit menos significativo de cada componente de cor (RGB) de cada píxel da imagem original, em tantos píxeis quantos os necessários para armazenar a informação secreta. Considerando que a informação secreta consiste numa string, o que se pretende é ocultar cada um dos 8 bits que formam cada caracter na ima-

gem, utilizando para o efeito o bit menos significativo de cada componente de cor dos píxeis. O exemplo seguinte ilustra o armazenamento dos primeiros dois caracteres da string "IPRP"num ficheiro de imagem:

Caracter a ocultar	Componentes RGB originais	Componentes RGB modificados
...	...	...
I (**01001001**)	R 01100111 G 01011010 B 10001101 R 10100111 G 01011010 B 10101101 R 11001010 G 10011011	R 01100110 G 01011011 B 10001100 R 10100110 G 01011011 B 10101100 R 11001010 G 10011011
P (**01010000**)	B 01010111 R 10101010 G 11101101 B 10100101 R 01111010 G 00101101 B 11011010 R 10101001	B 01010110 R 10101011 G 11101100 B 10100101 R 01111010 G 00101100 B 11011010 R 10101000
...	...	...

Tal como o exemplo anterior ilustra, o código ASCII de cada caracter é armazenado nas componentes RGB da imagem, utilizando para o efeito o bit menos significativo de cada componente. O armazenamento é sequencial, ou seja, o primeiro do bit do caracter seguinte começa na componente imediatamente a seguir à ultima componente de cor utilizada para armazenar o último bit do caracter anterior. Por último, refira-se que o programa deverá ser capaz de determinar se um determinado ficheiro de imagem dispõem ou não de uma mensagem secreta armazenada.

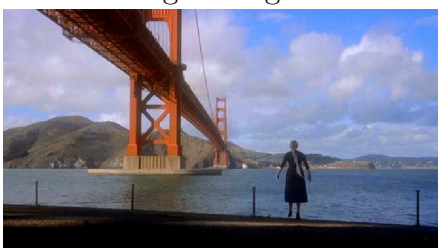
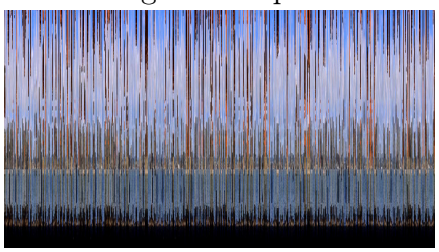
1.5.2 Encriptação

O objectivo desta operação é proteger uma imagem considerada confidencial, utilizando para o efeito um algorítmo de encriptação, que iremos utilizar para encriptar ou desencriptar uma imagem. Embora existam inúmeros algorítmos de encriptação em uso actualmente, o nosso propósito passa apenas por poder

modificar a imagem de modo a torná-la imperceptível quando comparada com a imagem original. Como tal, iremos implementar um algorítmo bastante simples, que passamos a descrever.

A operação de encriptação implementada pelo nosso algoritmo consiste em trocar a ordem às colunas da imagem original. Para que tal seja possível, o algorítmo deve dispor de informação secreta que indique qual a nova ordem de cada coluna na imagem encriptada, relativamente à imagem original. Esta informação secreta consistirá numa série de números, de dimensão igual à dimensão horizontal da imagem, para que todas as colunas possam ser trocadas.

A série de números representativa da nova ordem das colunas na imagem encriptada deverá ser uma série pseudo-aleatoria gerada pelo programa, com recurso às funções do módulo *random* do Python. Para que seja possível utilizar séries pseudo-aleatórias diferentes, o programa deverá solicitar uma password ao utilizador e derivar a semente inicial de geração de números aleatórios a partir dessa password. O programa deve somar o código ASCII de todos os caracteres que constituem a password pedida ao utilizador, e utilizar o valor resultante como semente no processo de geração de números aleatórios. O exemplo seguinte ilustra o resultado da encriptação de uma imagem utilizando a password "IPRP":

Imagem original	Imagem encriptada
	

O processo de desencriptação decorrerá de forma inversa, ou seja, considerando o número de ordem de cada coluna a repor para obter novamente a imagem inicial, de acordo com os valores da série de números pseudo-aleatórios gerada utilizando a mesma password.

1.6 Regras, critérios e métodos de avaliação

1.6.1 Constituição de grupos de trabalho

O presente trabalho deve ser executado em grupos de 2 alunos. Em situação alguma serão aceites grupos com mais do que 2 alunos. Excepcionalmente, e apenas em casos bem fundamentados, se aceitarão trabalhos individuais.

1.6.2 Metas de avaliação

A metas de avaliação são obrigatórias e os seus objectivos são descritos a seguir:

META 1: O programa deverá apresentar as seguintes funcionalidades (ver secção 1.3): **a**. Leitura de imagem a partir de ficheiro para estrutura de dados em memória. O utilizador deverá poder indicar o nome do ficheiro pretendido. **b**. Utilizando as funções já referidas da PIL, o programa deverá apresentar a imagem numa janela. **c**. Armazenamento da mesma imagem noutro ficheiro, novamente de acordo com nome indicado pelo utilizador.
META 2: O programa deverá apresentar as seguintes funcionalidades (ver secção 1.4): **a**. Apresentação do menu com as opcções suportadas pelo programa, bem como a leitura da opcção seleccionada pelo utilizador. **b**. Tratamento da imagem com as operações fundamentais de processamento de imagens descritas anteriormente.
META 3: Entrega e defesa do trabalho, na sua versão final e implementando todas as operações descritas anteriormente (ver as secções 1.3, 1.4 e 1.5.1).

Nota: Factores que podem valorizar o seu programa incluem: optimização do desempenho das operações de processamento de imagens, tratamento de excepções;

Note Bem

1. Não cumprir a Meta 1 ou entregar fora do prazo faz perder 20% da nota;

2. Não cumprir a Meta 2 ou entregar fora do prazo faz perder 30% da nota;

3. Não cumprir a Meta 3 implica reprovação na cadeira.

4. A fraude denota uma grave falta de ética e constitui um comportamento não admissível num estudante do ensino superior e futuro profissional licenciado. Qualquer tentativa de fraude na elaboração do presente Trabalho Prático conduzirá inevitávelmente à reprovação imediata na cadeira