# Relatório de Arquitetura de Sistemas e Computadores I

Luís Rosado - 34249
Daniel Soares - 34222

26 de Maio de 2017

**Resumo**

O trabalho de Arquitetura de Sistemas e Computadores teve como objetivo desenvolver um programa para detetar contornos em imagens a cores.

# 1 Execução do Programa

Na execução do programa é pedido ao utilizador o nome do ficheiro a ler, este ficheiro é uma imagem no formato RGB, de seguida é gerado o nome do ficheiro de saída, que é o mesmo da imagem com a extensão .GRAY e guardado para ser usado mais tarde. É aberto o ficheiro e a imagem guardada em memoria. A imagem lida é em primeiro lugar transformada em tons de cinzento e guardada em memoria. De seguida é usada a operação de convolução horizontal e vertical na imagem original e as respetivas imagens geradas guardadas em memoria. É depois gerada uma imagem final, com os contornos a partir das duas imagens de convolução. Por fim essa imagem final é depois guardada num ficheiro com o nome anteriormente gerado.

# 2 Funções

1. main: Inicializa os argumentos das funções que executam as operações e chama-as.
2. ask_filename: Esta função pede o nome do ficheiro a ler ao utilizador e guarda-o em *readfile*, são aceites nomes com 31 caracteres. Depois de receber o nome do ficheiro, é retirado o carácter newline '\n'. Tenta abrir o ficheiro e, se houver sucesso fecha-o, se não pede novamente o nome ao utilizador. É chamada depois a função *writefilename*. No final da execução, *readfile* contem o nome do ficheiro a ler e *savefile* o nome do ficheiro a gravar.
3. writefilename: Esta função é usada para gerar o nome do ficheiro de saída com a extensão '.GRAY'. Copia os caracteres para *savefile* até encontrar o carácter '.' ou até chegar à posição 26, o sítio limite para escrever a extensão sem overflow.
   Acrescenta depois em *savefile* *.Gray* $\backslash 0$.
4. read_rgb_image: Abre o ficheiro com o nome guardado em *readfile* le o ficheiro e copia os dados para o *originalbuffer*, no fim fecha o ficheiro.
5. rgb_to_gray: A função le os 3 bytes de cada pixel da imagem no *originalbuffer* efetua o cálculo para a conversão do pixel para escala de cinzento e guarda-o no *graybuffer*
6. convolution: Esta função faz a convolução da imagem com um operador Sobel. Assume-se que no limite da imagem não definida o valor dos pixeis é 0 e que o tamanho da imagem é de 512x512. É usado o *graybuffer* para ler os pixeis, e *h_Sobel* ou *v_sobel* para ler a matriz do operador, o resultado guardado no respetivo buffer,

*h_sobelbuffer* ou *v_sobelbuffer*. A implementação está separada em 3 partes.

**1ª linha**
Para fazer a convolução da primeira linha esta é dividida em 3 partes. 1º pixel: Faz o cálculo para este pixel apenas usando os 3 adjacentes que fazem parte da imagem, assume-se que os restantes 5 que não fazem parte da imagem têm valor 0.
Pixeis intermédios: Faz o cálculo para os 510 pixeis intermédios da linha que são iguais, apenas com os 5 adjacentes, assume que os restantes 3 que não fazem parte da imagem têm valor 0.
Último pixel: Faz o cálculo para este pixel apenas usando os 3 adjacentes que fazem parte da imagem, assume-se que os restantes 5 que não fazem parte da imagem têm valor 0.

**Linhas intermédias:**
Faz a convolução das 510 linhas intermédias que são iguais.
1º pixel: Faz o cálculo para este pixel apenas usando os 5 adjacentes que fazem parte da imagem, assume-se que os restantes 3 que não fazem parte da imagem têm valor 0.
Pixeis intermédios: Faz o cálculo para os 510 pixeis intermédios da linha que são iguais, utiliza todos os 8 pixeis adjacentes.
Útimo pixel: Faz o cálculo para este pixel apenas usando os 5 adjacentes que fazem parte da imagem, assume-se que os restantes 3 que não fazem parte da imagem têm valor 0.

**Última linha**
Para fazer a convolução da última linha esta é dividida em 3 partes 1º pixel: Faz o cálculo para este pixel apenas usando os 3 adjacentes que fazem parte da imagem, assume-se que os restantes 5 que não fazem parte da imagem têm valor 0.
Pixeis intermédios: Faz o cálculo para os 510 pixeis intermédios que são iguais, apenas com os 5 adjacentes, assume que os restantes 3 que não fazem parte da imagem têm valor 0.
Último pixel: Faz o cálculo para este pixel apenas usando os 3 adjacentes que fazem parte da imagem, assume-se que os restantes 5 têm valor 0.

7. contour: A função recebe o *h_sobelbuffer* e o *v_sobelbuffer*, lê cada pixel dos dois buffers na mesma posição, faz a sua média, ajusta o contraste e grava-o no *contourbuffer* na mesma posição lida.
8. write_gray_image: Esta função lê o *contourbuffer* e guarda cada pixel da imagem num ficheiro de imagem .GRAY. Abre ou cria o ficheiro usando o nome em *savefile*.

# 3 Conclusão

Através deste trabalho foi-nos possível consolidar alguns conhecimentos na área de Arquitetura de Sistemas e Computadores. Conseguimos superar as dificuldades que apareceram na parte da operação Sobel e ficámos satisfeitos por conseguirmos concretizar os objetivos propostos, faltando apenas, para complementar, a possibilidade de realizar a operação para imagens de vários tamanhos. A execução do trabalho transmitiu-nos um elevado interesse para a área de modificação de imagens e esperamos que seja útil para o futuro no mercado de trabalho.