Compiladores

Linguagem *iml*

Departamento de Electrónica, Telecomunicações e Informática
Universidade de Aveiro

Abril de 2025

Objectivos

O objectivo geral deste trabalho é o desenvolvimento de uma linguagem de programação compilada – i.e. que crie programas numa linguagem de programação genérica (Java, C++, Python, ...) – que permita manipulação de imagens em tons de cinza (por exemplo imagens PGM[1]).

As imagens são representadas como matrizes de valores reais no intervalo [0, 1], onde 0 corresponde ao preto e 1 ao branco. Uma característica distintiva do tipo image é a sua natureza saturante. Isto significa que, ao contrário dos outros tipos de dados, os valores dos pixels permanecem dentro do intervalo [0, 1], mesmo após operações que excedam esses limites.

As operações aritméticas padrão são suportadas para os tipos de dados *percentage* e *number*, permitindo a realização de cálculos. No entanto, o verdadeiro objectivo da linguagem IML reside nas suas operações especializadas para o tipo *image*.

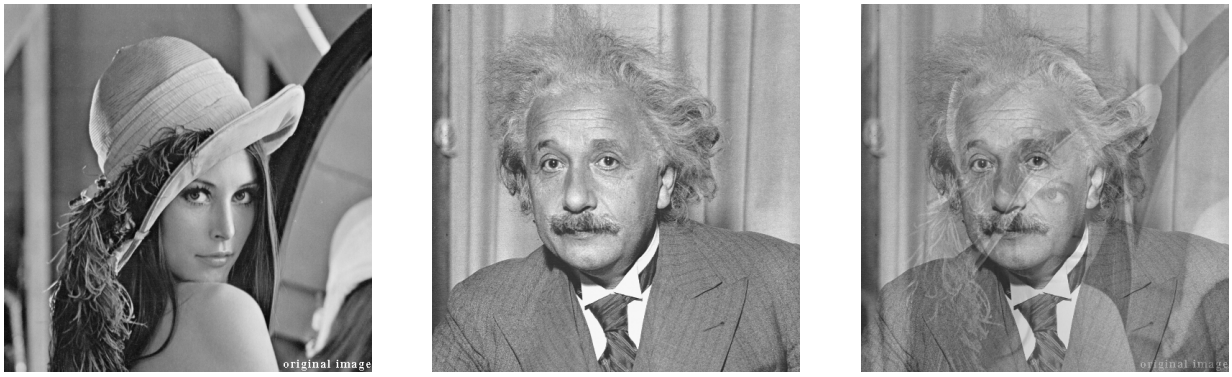
- As operações pixel a pixel, prefixadas por um ponto (.), permitem aplicar transformações aritméticas e booleanas a cada pixel individualmente. Por exemeplo a soma de duas imagens utiliza o operador .+.
- Os operadores unários .- quando aplicados a uma imagem devem produzir o inverso da imagem. Ou seja, cada pixel resultante será o oposto do pixel original.
- Os operador -*, /* e +* representam as operações de escala horizontal, vertical e em ambos os eixos respectivamente.
- Da mesma forma os operadores unários -, / e + aplicam a operação de flip na vertical, horizontal e em ambos os eixos respectivamente.
- Finalmente existem as seguinte operações morfológicas *erode*, *dilate*, *open* e *close*.

[1] `https://en.wikipedia.org/wiki/Netpbm`

Em baixo pode encontar um exemplo da aplicação da linguagem (os ficheiros têm extensão *iml*):

```
// load images from files
image i0 is load from "images/sample00.pgm"
image i1 is load from "images/sample01.pgm"

// ajust horizontal scale if needed
number i0cols is columns of i0
number i1cols is columns of i1

if i0cols != i1cols then
    i0 is i0 -* (i1cols / i0cols)
done

// ajust vertical scale if needed
number i0rows is rows of i0
number i1rows is rows of i1

if i0rows != i1rows then
    i0 is i0 |* (i1rows / i0rows)
done

// blend two images together
image r is i0 .* 30% .+ 70% .* i1

// store the resulting image
r store into "images/blend.pgm"
```

Um exemplo da execução do código anterior pode ser vista na Figura 1.



(a) Sample 00: Lena (b) Sample 01: Einstein (c) Blend image

Figura 1: O resultado da execução do programa anterior. Combina as duas imagens numa que é a mistura.

As operações morfológicas em imagens de tons de cinzento são como ferramentas que modificam a forma dos objetos na imagem, usando um "molde" chamado elemento estruturante. Imagine que o elemento estruturante é uma pequena forma que deslizamos sobre a imagem, e dependendo da forma do molde, podemos engrossar ou afinar os contornos dos objetos, remover pequenos detalhes ou preencher buracos. Essas operações ajudam a

limpar a imagem, realçar características importantes e preparar a imagem para análise ou reconhecimento de padrões[2].

Segue um exemplo de uma aplicação de uma operação morfológica:

```
// load image from file
image i is load from "images/sample03.pgm"
// generate an image with the secundary language
image k is run from read "Path: "

// sample morphological operation example
// applying by this order removes slat pepper noise
image r is (i open by k) close by k

// store the resulting image
r store into "images/clean.pgm"
```

Um exemplo da execução do código anterior pode ser vista na Figura 2.

(a) Sample 03: Letra J com ruído sal pimenta.

(b) Clean : imagem após a aplicação das operações morfológicas.

Figura 2: O resultado da execução do programa anterior. Utiliza operações morfológicas para remover o ruído sal pimenta com distorção mínima.

A descoberta da sintaxe desta linguagem deve ser feita recorrendo os programas de exemplo. Os programas exemplo com o prefixo "min" representam exemplos das características mínimas. Da mesma forma, os programas exemplo com o prefixo "des" representam exemplos das características desejáveis.

A linguagem secundária (interpretada, com extensão *iiml*) vai ser uma versão interpretada simplificada da linguagem principal. Além da aritmética padrão e input the valores, permite criar imagens e colocar figuras geométricas simples (rectangulo, circulo, e cruz). É bastante util para a instanciação de elementos estruturante para as operações morfológicas.

Como exemplo de um programa simples nesta linguagem:

[2] https://en.wikipedia.org/wiki/Mathematical_morphology

```
// read image size from input
number l is number(read "Size: ")
// create a new image
image size l by l background 0
// read circle radius from input
number r is number(read "Radius: ")
// place a circle in the image
// other forms are: rect, cross, plus
place circle radius r at l/2 l/2 with intensity 1
```

Não é expectavel que os grupos implementem os métodos de manipulão de imagens, devem recorrer a bibliotecas próprias para isso. Fica ao critério de cada grupo, no entanto ficam aqui sugestões para cada linguagem de destino: i) para C++ podem recorrer da biblioteca OpenCV[3], ii) para Java podem recorrer da biblioteca ImageJ[4], e iii) Para python podem recorrer do wrapper para OpenCV[5] ou da biblioteca Scikit-Image[6].

Características da solução

Apresentam-se a seguir um conjunto de características que a solução desenvolvida pode ou deve contemplar. Essas características estão classificadas a 3 níveis:

- mínima – característica que a solução tem obrigatoriamente que implementar;
- desejável – característica não obrigatória, mas fortemente desejável que seja implementada pela solução (apenas considerada se as mínimas forem cumpridas);
- avançada – característica adicional apenas considerada para avaliação se as obrigatórias e as desejáveis tiverem sido contempladas na solução.

Características mínimas

Os exemplos `min-01.iiml`, `min-01.iml`, `min-02.iml` e `min-03.iml` indicam algum código fonte que tem de ser aceite (e devidamente compilado e interpretado) pelas linguagens a desenvolver.

A linguagem deve implementar:

- Instrução para carregar e salvar imagens em tons de cinzento (.pgm). Além disso operações morfológicas, escala, flip e aritméticas aplicadas pixel a pixel (como descrito nos objectivos).

[3] `https://opencv.org/`
[4] `https://imagej.net/ij/`
[5] `https://pypi.org/project/opencv-python/`
[6] `https://scikit-image.org/`

- Os tipo de de dados imagem, string, número, e percentagem.
- Aceitar expressões aritméticas standard para os tipos de dados numéricos (número e percentagem). Deve também aceitar a operação de concatenação de texto, usando para isso o operador de adição.
- Instrução de escrita no *standard output*.
- Instrução de leitura de texto a partir do *standard input*.
- Operadores de conversão entre tipos de dados (por exemplo, **number**("10") para converter para número).
- Instruções para desenhar uma imagem.
- Execução da linguagem secundária (interpretada pela linguagem destino).
- Verificação semântica do sistema de tipos.

A linguagem interpretada é um subset da linguagem IML, que deve suportar o programa `kernl.iiml`. Deve também incluir:

- Instrução para a criação de uma imagem
- Instrução para desenhar figuras geométricas

Características desejáveis

Os exemplos `des-01.iml`, `des-02.iml`, `des-03.iml`, `des-04.iml`, e `des-05.iml` indicam algum código fonte que se enquadra nas características desejáveis.

- Permitir a definição de expressões booleanas (predicados) contendo, pelo menos relações de ordem e operadores booleanos (conjunção, disjunção, etc.). Tendo em conta que podem ser definidas pixel a pixel quando aplicadas a imagens.
- Operadores any pixel, all pixel que permitem verificar se uma imagem ou lista booleana possuem pelo menos um, ou todos os valores verdadeiros, respectivamente.
- Operador count pixel in que permite contar pixeis de uma determinada intensidade numa imagem
- Incluir a instrução condicional (operando sobre expressões booleanas).
- Adição da lista como um novo tipo de dados. De notar que são listas de tipos. Devem permitir a adição e remoção de elementos da lista.

- De reparar que uma matriz pode ser criada como uma lista de listas (e assim sucessivamente). Desta forma é possivel introduzir matrizes (lista de listas de números ou percentagem) que podem ser operadas com imagens. Existe tambem o operador de indexação, para manipular elementos dentro das listas, de notar que este pode ser aplicado recursivamente.
- A operação de armazenamento de uma lista de imagens deve gerar um gif como ficheiro de resultado.
- Incluir instruções iterativas tipo **for** e tipo **until** (operando sobre listas e expressões booleanas respectivamente).
- Sobre as imagens são suportados mais duas operações morfológicas: top hat e black hat. Estas sãi usadas para realçar o contraste das imagens (tons claros e escuros respectivamente).

Características avançadas

- Implementar funções e variáveis locais às mesmas.
- Implementar uma tabela de símbolos que resolva o problema dos contextos de declaração.
- Extender a linguagem para suportar tratamento de erros (por exemplo:"File not found") para lidar com erros de runtime.
- Estender a linguagem para suportar images a cores (RGB) com todas as operações definidas anteriormente.
- ...