

Objetivo

Desenvolvimento de um programa em Matlab para fazer análise de imagens que contêm códigos de barras lineares, códigos matriciais (QR codes) e outros objetos sem significado. O programa deve ser capaz de interpretar imagens fornecidas e de gerar os resultados pedidos conforme descrito adiante. Serão dadas imagens exemplo para permitir o desenvolvimento, mas as imagens usadas para obter os resultados de avaliação serão novas.

Natureza das imagens

As imagens a processar terão um número variável de objetos dos diversos tipos, com eventual ruído ou textura na zona do fundo. Os objetos estarão delimitados por caixilhos retangulares ou quadrados. A figura 1 ilustra um exemplo de imagem a processar.

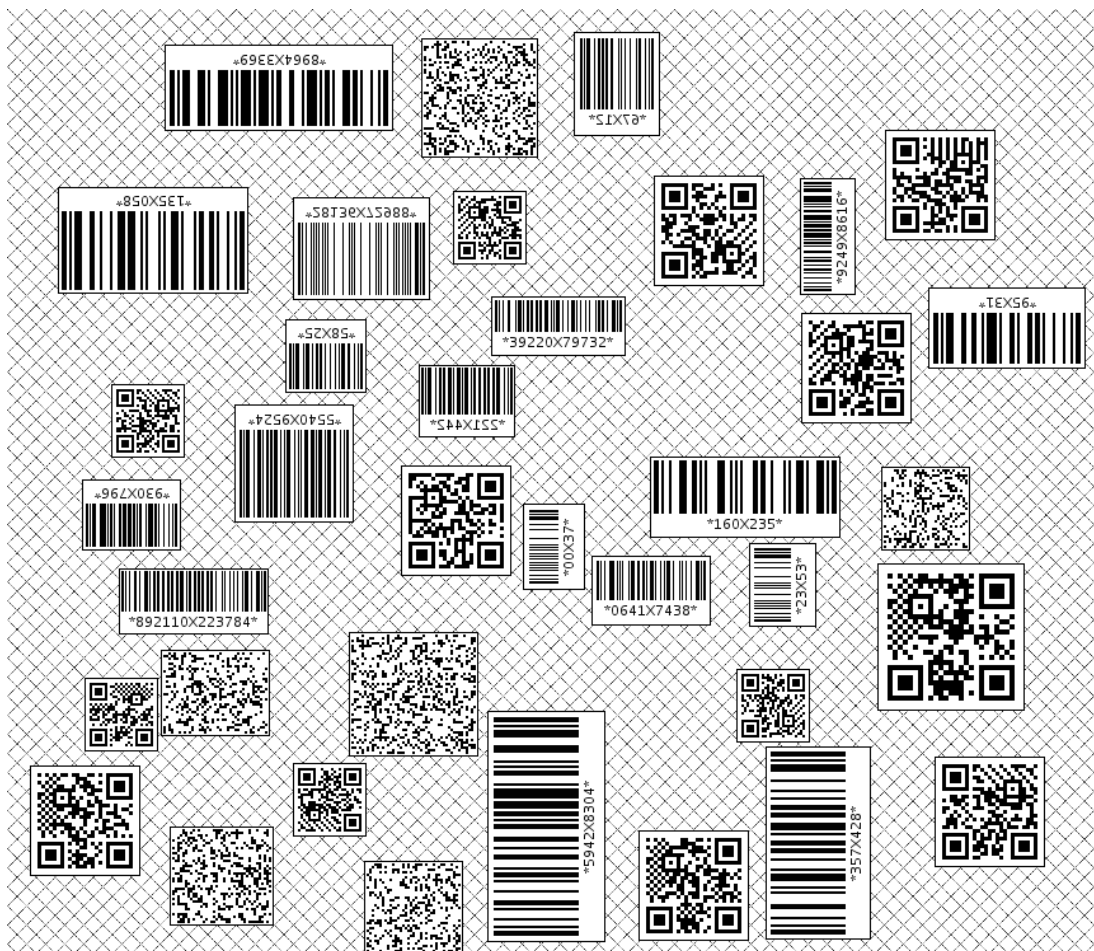


Figura 1: Exemplo de uma imagem com códigos de barras, QR codes e outros sem significado.

As características gerais das imagens são as seguintes:

- As imagens serão em níveis de cinzento.
- As imagens conterão 3 tipos de objetos: códigos de barras lineares, códigos matriciais (QR codes) e outras figuras retangulares ou quadradas sem significado.
- Os objetos poderão existir em qualquer uma das 4 orientações possíveis.

- Os objetos existirão em diversas escalas, mas sempre em múltiplos inteiros de pixels.
- Os códigos de barras podem surgir refletidos em simetria axial.
- Na menor escala, as barras nos códigos de barras terão larguras mínimas de 1 pixel.
- Na menor escala, as unidades de informação nos QR codes e nos objetos sem significado serão quadrados com 2×2 pixels.
- O número total de objetos, e a informação que contêm, será variável em cada imagem.
- Os objetos estarão envolvidos por um caixilho quadrado preto.
- Haverá sempre espaço branco, de pelo menos um pixel, junto ao caixilho da parte de dentro.
- O fundo da imagem, fora dos caixilhos, poderá conter ruído/textura de intensidade não preta.
- Os pixels dos objetos serão em níveis de cinzento mais escuros do que o seu fundo, com exceção de em cerca de 20% dos objetos onde poderá haver um nível superior de ruído no seu interior.
- Não é garantido que o tipo de letra dos dígitos nos códigos de barras seja sempre o mesmo, por isso não se recomenda contar apenas com a análise dos caracteres.

Parâmetros a detetar em cada imagem

Em cada imagem, os principais parâmetros e características a detetar pelo programa do aluno são:

- Número **total** de objetos sem significado.
- Número **total** de códigos de barras.
- Número **total** de códigos matriciais.
- Número de códigos de barras em cada uma das **4 orientações** possíveis.
- Número de códigos de barras **válidos** com reflexão axial.
- Número de códigos de barras **inválidos** de acordo com o enunciado.
- Número **total** acumulado de dígitos representados nos códigos de barras **válidos**.
- Número de códigos de barras **válidos** em cada uma das 3 codificações possíveis ('L', 'R', 'G').
- *String* com os dígitos centrais dos códigos de barras **válidos** ordenados de forma crescente.

Os códigos de barras

Neste trabalho, os códigos de barras traduzem grupos de dígitos decimais. Cada dígito é representado por um conjunto de 7 barras pretas ou brancas. Existem várias formas de codificação dos dígitos, sendo uma delas a codificação 'L'. Se uma barra branca for representada por um '1' e uma barra preta por um '0', então o código para representar por exemplo o dígito '5', na codificação 'L', será o seguinte: 1 0 0 1 1 1 0. Cada grupo de dígitos é delimitado por um código de início (start) e um código de fim (end); estes códigos são diferentes entre si.

- O código delimitador de início é dado por: 0 0 1 0 1 1 0 1 1 1 0 (onze barras)
- O código delimitador de fim é dado por: 0 1 1 1 0 0 0 1 0 1 0 0 (doze barras)

Assim, um código de barras para representar simplesmente o dígito '5' na codificação 'L', seria como ilustrado na figura 2.

Delimitador de início											Dígito '5'							Delimitador de fim											
0	0	1	0	1	1	0	1	1	1	0	1	0	0	1	1	1	0	0	1	1	1	0	0	0	1	0	1	0	0

Figura 2: Representação do dígito '5' na codificação 'L' incluindo os delimitadores de início e fim.

A representação completa da sequência '0123456789', na codificação 'L', incluindo os delimitadores, é ilustrada na figura 3, onde os delimitadores 'start' e 'end' estão representados a sombreado para mais fácil interpretação das regras mas que, nas imagens reais, seguirão as mesmas regras das restantes barras dentro do símbolo.

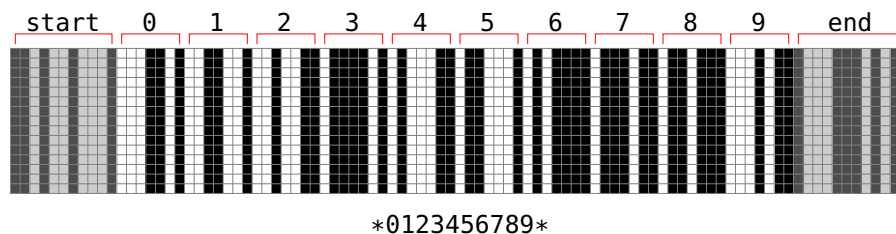


Figura 3: Sequência '0123456789' na codificação 'L' com os delimitadores destacados a sombreado.

Para além da codificação 'L', existem também as codificações 'R' e 'G'. Qualquer destas codificações pode existir num código de barras. A tabela 1 representa essas codificações para os 10 dígitos decimais.

Dígito	Codificação 'L'	Codificação 'R'	Codificação 'G'
0	1 1 1 0 0 1 0	0 0 0 1 1 0 1	1 0 1 1 0 0 0
1	1 1 0 0 1 1 0	0 0 1 1 0 0 1	1 0 0 1 1 0 0
2	1 1 0 1 1 0 0	0 0 1 0 0 1 1	1 1 0 0 1 0 0
3	1 0 0 0 0 1 0	0 1 1 1 1 0 1	1 0 1 1 1 1 0
4	1 0 1 1 1 0 0	0 1 0 0 0 1 1	1 1 0 0 0 1 0
5	1 0 0 1 1 1 0	0 1 1 0 0 0 1	1 0 0 0 1 1 0
6	1 0 1 0 0 0 0	0 1 0 1 1 1 1	1 1 1 1 0 1 0
7	1 0 0 0 1 0 0	0 1 1 1 0 1 1	1 1 0 1 1 1 0
8	1 0 0 1 0 0 0	0 1 1 0 1 1 1	1 1 1 0 1 1 0
9	1 1 1 0 1 0 0	0 0 0 1 0 1 1	1 1 0 1 0 0 0

Tabela 1: Códigos para as três codificações possíveis de códigos de barras.

A figura 4 ilustra a representação da string '12345' nas três codificações L', 'R' e 'G'.



Figura 4: As codificações 'L', 'R' e 'G' para a sequência '12345'.

Os códigos de barras que não estiverem em nenhuma das codificações 'L', 'R', ou 'G' serão **inválidos**. Os códigos de barras poderão surgir em qualquer das 4 orientações, como ilustrado na figura 5.




Figura 5: Em cima: as 4 orientações possíveis dos códigos de barras (R0, R90, R180, R270). Em baixo: os mesmos códigos mas com reflexão axial.

Recorde-se que é possível que um código de barras apareça refletido em relação a um eixo central com a direção das barras. Isso por si só não o invalida e a sua rotação é aquela que terá depois de corrigida a reflexão axial. A parte de baixo da figura 5 ilustra as versões refletidas nas diversas orientações.

Os códigos matriciais

Nas aplicações industriais, os códigos matriciais podem ser de naturezas variadas. Porém, neste trabalho, será feita a restrição aos **QR codes** que se caracterizam por certas propriedades fáceis de identificar visualmente. Contrariamente aos códigos de barras, a informação está codificada nas duas dimensões como uma verdadeira imagem. Cada unidade de informação é um pequeno elemento quadrado binário (preto ou branco), portanto, equivalente a uma espécie de *macropixel*. Para este trabalho não é necessário decodificar o conteúdo dos QR codes, apenas o número total presente na imagem. As características de interesse para este trabalho são as seguintes:

- Os QR codes são **quadrados**.
- Os elementos de informação formam um arranjo matricial com número ímpar de unidades de informação, obedecendo à regra **$17+4N$** , com $N=1,2,3,\dots$, como 21×21 , 25×25 , 29×29 , etc.
- Um QR code tem **3 sub-padrões fixos** que definem a sua orientação: 
- Os QR codes têm uma zona branca à sua volta chamada *Quiet Zone* de dimensão variável, mas que não conta para a dimensão do QR code.

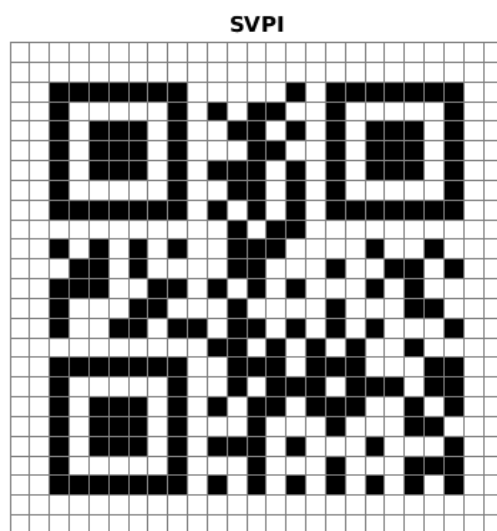


Figura 6: QR code de 21×21 (menores dimensões válidas) que representa a string 'SVPI' (embora não seja necessário decodificar o conteúdo do QR code). Cada quadrícula da grelha sobreposta no QR code delimita uma unidade de informação binária.

Algumas sugestões de técnicas para realizar o trabalho

Indicam-se de seguida algumas sugestões de técnicas possíveis para abordar o problema, mas são meramente indicações que não têm de ser seguidas, e nem sequer são obrigatórias para a solução.

- Detetar os cantos delimitadores dos caixilhos dos objetos com filtros de convolução ou outras técnicas.
- Isolar as regiões dos códigos a partir das caixas individuais de cada objeto; pode-se fazer uma análise linha a linha e coluna a coluna a partir dos cantos detetados para obter os limites dos caixilhos.
- Detetar se é código de barras, código matricial ou objeto sem significado.
- Detetar a orientação do código e eventualmente reorientá-lo para o processar.
- Para os códigos de barras detetar se está refletido axialmente e eventualmente corrigir para o continuar a processar.
- Fazer os testes de verificação da validade do código e outras eventuais propriedades.

Formato do ficheiro com o trabalho do aluno (.m file)

O trabalho de cada aluno deve ficar contido num único ficheiro Matlab do tipo .m e que terá o formato de função (function), devendo retornar um único valor que é o numero mecanográfico do aluno. Se o aluno desenvolver funções auxiliares necessárias ao trabalho, elas deverão ficar incluídas nesse mesmo ficheiro, na parte final, depois da função principal.

Formato do ficheiro de resultados

Quando executado, o programa a desenvolver pelo aluno deve ler um conjunto de imagens fornecidas com nomes do género `svpi2024_TP1_img_MMM_NN.png` onde `NN` poderá variar de 01 até 99; esse número deverá ser detetado automaticamente pelo programa do aluno, como descrito adiante. `MMM` será um número de três dígitos que designa a sequência de imagens em análise e também deverá ser detetado automaticamente a partir no nome do ficheiro.

O programa a desenvolver pelo aluno deve analisar as imagens da sequência, uma por uma, e gerar uma tabela de estatísticas com tantas linhas quantas as imagens da sequência, e onde em cada linha se indicam os parâmetros pedidos. Esta tabela será sujeita a avaliação como descrito mais adiante, e deve ser escrita pelo programa do aluno num ficheiro com nome `tp1_nnnnnn.txt` onde `nnnnnn` é o número mecanográfico do aluno. Nesse ficheiro, as respostas para cada imagem devem aparecer numa linha e separadas por vírgulas. Por exemplo, para o aluno número 999999, o código Matlab do trabalho deve gerar o ficheiro `tp1_999999.txt` que deverá conter os seguintes valores separados por vírgulas:

`NumMec` Número mecanográfico do aluno.
`NumSeq` Número da sequência da imagem (Cf. nome do ficheiro de imagem).
`NumImg` Número da imagem na sequência (Cf. nome do ficheiro de imagem).
`TotNM` Número de objetos sem significado.
`TotCB` Número total de objetos com códigos de barras.
`TotQR` Número total de objetos com QR codes.
`R0` Número de códigos de barra na orientação de 0°.
`R90` Número de códigos de barra na orientação de 90°.
`R180` Número de códigos de barra na orientação de 180°.
`R270` Número de códigos de barra na orientação de 270°.
`RefLCB` Número de códigos de barras **válidos** com reflexão axial.
`BadCB` Número de códigos de barras inválidos.
`TotDigCB` Número total acumulado de dígitos representados nos códigos de barras **válidos**.
`CBL` Número de códigos de barras válidos na codificação 'L'.
`CBR` Número de códigos de barras válidos na codificação 'R'.
`CBG` Número de códigos de barras válidos na codificação 'G'.
`StringCB` *String* com os dígitos centrais dos códigos de barras válidos ordenados crescentes.

Todos os campos são numéricos à exceção do campo `StringCB` que terá de ser representado como *string* em virtude da sua natureza.

Um parâmetro não verificado na imagem deve constar na tabela com valor 0. Por exemplo, se não houver na imagem nenhum objeto do tipo código de barras com orientação de 270°, o parâmetro `R270` para essa imagem terá o valor 0.

Exemplo de resultados

Como exemplo concreto, tomem-se as duas imagens da figura 7 que se pretendem analisar e cujos ficheiros têm nomes: `svpi2024_TP1_img_111_02.png` e `svpi2024_TP1_img_111_04.png`.

Se o número mecanográfico do aluno fosse 999999, o ficheiro de resultados `tp1_999999.txt` para esta sequência de duas imagens teria o seguinte conteúdo:

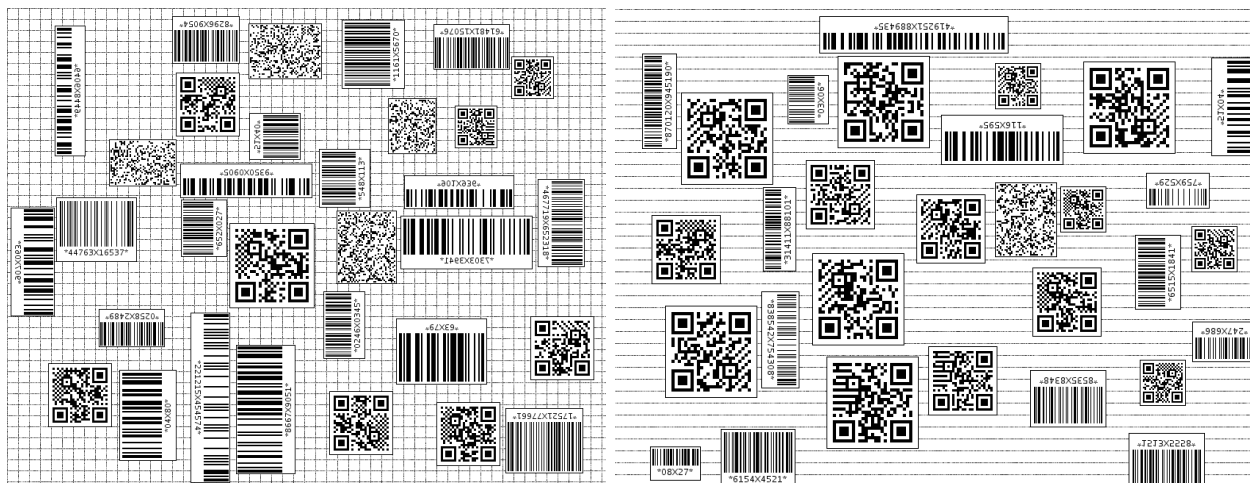


Figura 7: Exemplos de duas imagens a processar: sequência 111, imagens 02 e 04.

999999,111,2,4,20,8,2,7,7,4,5,2,150,3,9,6,002223334445557789
999999,111,4,1,14,15,2,4,6,2,2,3,97,6,4,1,01234466799

Para mais fácil referência e verificação, seguem os valores numa tabela com indicação do nome dos campos.

NumMec	NumSeq	NumImg	TotNM	TotCB	TotQR	R0	R90	R180	R270	ReflCB	BadCB	TotDigCB	CBL	CBR	CBG	StringCB
999999	111	2	4	20	8	2	7	7	4	5	2	150	3	9	6	002223334445557789
999999	111	4	1	14	15	2	4	6	2	2	3	97	6	4	1	01234466799

Adicionalmente, e para melhor verificação, seguem imagens auxiliares (figura 8) onde se indica a informação em falta e corrigem os códigos refletidos, bem como se assinala com fundo cinzento os códigos de barras inválidos. Não é necessário que o trabalho do aluno gere estas imagens, elas são meramente auxiliares para se compreender os resultados indicados.

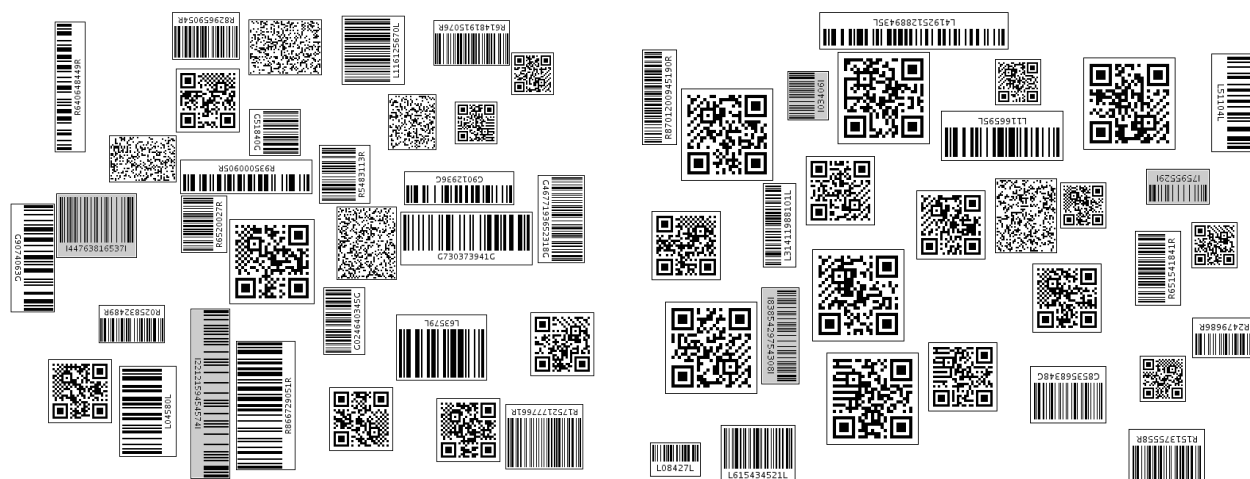


Figura 8: Soluções das imagens da figura 7 onde se indicam os tipos de codificação dos códigos de barras, se completam os seus valores numéricos, e se sombreiam os códigos de barras inválidos!

Procedimento para entrega do trabalho

O aluno deve executar o comando `SVPI_ProcessTP1_2024` (fornecido on-line) usando como argumento o nome do seu ficheiro Matlab. Por exemplo, se o programa do aluno se designar `tp1_999999.m` então, dentro da pasta que o contém, deve executar o comando:

```
SVPI_ProcessTP1_2024('tp1_999999.m')
```

Se o programa do aluno executar sem erros, será criado o ficheiro `cod_svpi2024_tp1_nnnnnn.vsz` (com `nnnnnn`=999999 neste exemplo). Este é o ficheiro que deve ser entregue no E-learning e que é um ficheiro de arquivo que inclui o código desenvolvido pelo aluno. Se o ficheiro de respostas `tp1_nnnnnn.txt` não for gerado pelo programa do aluno, então o comando `SVPI_ProcessTP1_2024` aborta em erro. Só o ficheiro com extensão `*.vsz` é aceite para entrega no E-learning e nenhum outro será considerado.

Resumo das condições necessárias para se conseguir a geração do ficheiro a entregar

`cod_svpi2024_tp1_nnnnnn.vsz`:

- O programa do aluno não pode ter erros de execução.
- A pasta onde está a ser executado não pode ter nenhum ficheiro `svpi2024_TP1_img_*.png`
- Os ficheiros de imagem `svpi2024_TP1_img_*.png` têm de estar na pasta anterior (`../`)
- O programa do aluno (formato de função Matlab) tem de devolver o número mecanográfico.
- O programa do aluno tem de gerar o ficheiro `tp1_nnnnnn.txt` onde `nnnnnn` é o número mecanográfico.

Avaliação

A avaliação é baseada em dois parâmetros principais:

1. Conformidade dos resultados do trabalho de acordo com imagens de avaliação.
2. Apreciação do código fornecido.

A conformidade dos resultados é feita essencialmente com base na taxa de resultados corretos em comparação com os resultados reais resultantes de uma análise correta às imagens de avaliação. A apreciação do código fornecido poderá contemplar a eficiência geral, nomeadamente no tempo de execução. Programas com tempos de execução demasiado longos (mais de 30 segundos, em média, por imagem de uma sequência) poderão ser penalizados, ou mesmo excluídos de avaliação. O código entregue pelos alunos será sujeito a um sistema de verificação de plágio (sistema MOSS). Todos os trabalhos que apresentarem um índice de plágio maior do que um dado limite são passíveis de análise e tratamento específicos.

Observações e recomendações sobre o trabalho e o Matlab

- O programa a executar deve-se chamar `tp1_nnnnnn.m` (onde `nnnnnn` é o número mecanográfico)
- O programa deve procurar as imagens na pasta anterior à sua (`../`).
- O programa deve correr de forma não interativa, ou seja, não deve ter interface gráfica nem deve esperar entrada de dados do utilizador.
- O programa deverá ser capaz de determinar quantas imagens a pasta tem, para as poder abrir.

O programa deve gerar, na pasta corrente, o ficheiro `TP1_nnnnnn.txt`, (onde `nnnnnn` é o número mecanográfico do aluno) nos moldes descritos anteriormente.

As imagens a processar estarão na pasta anterior à pasta que contém o programa. Por exemplo, se o programa do aluno estiver numa pasta de nome parcial `"as_minhas_pastas_locais/svpi/tp1/"` então as imagens estarão em `"as_minhas_pastas_locais/svpi/"`.

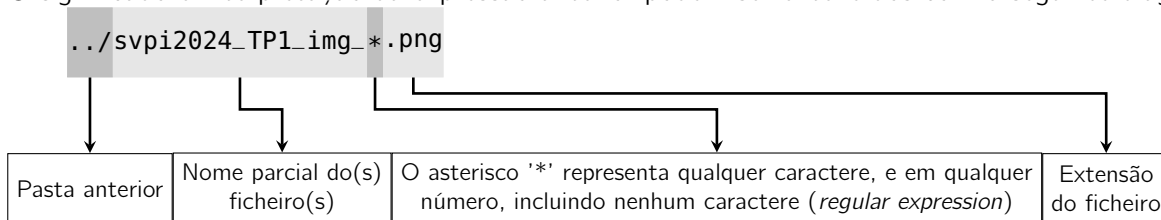
Ou seja, para tornar o programa independente do nome da pasta onde se encontre, ele deve procurar as imagens na pasta anterior. Assim, recomenda-se que essa pasta seja adicionada ao `path` do Matlab, e isso pode-se fazer recorrendo ao seguinte comando a colocar no início do programa:

```
addpath('../');
```

Para obter uma lista de ficheiros, para depois os poder processar, sugere-se o uso da função `dir()` do Matlab para listar o conteúdo de uma pasta. Neste caso, e para obter a lista de todas imagens que estão na pasta anterior, pode-se fazer algo como:

```
listaF=dir('../svpi2024_TP1_img_*.png');
```

O significado e interpretação da expressão anterior podem ser entendidos com o seguinte diagrama:



Neste caso, a variável `listaF` conterá uma listagem de todos os ficheiros de imagens. A variável será uma estrutura em Matlab (*structure*) que conterá toda a informação sobre os ficheiros. Por exemplo:

```
size(listaF,1) -> dá o número de ficheiros encontrados pelo comando dir.
listaF(1).name -> dá o nome do primeiro ficheiro da lista.
```

Uma vez que a pasta anterior (onde estarão as imagens) já foi colocado no `path`, um comando como `A=imread(listaF(1).name)` não deverá ter problemas em encontrar o ficheiro.

Para a geração de resultados de avaliação, além da contabilização dos parâmetros dos objetos presentes na imagem, como descrito no enunciado, e como o enunciado também refere, é necessário incluir o número da sequência e o número da imagem na sequência, que se extraem do próprio nome do ficheiro da imagem. Basta pensar que o nome do ficheiro é um *array* de caracteres sempre com o mesmo comprimento e, portanto, as indicações do número de sequência e do número de imagem estão sempre na mesma posição:

s v p i 2 0 2 4 _ T P 1 _ i m g _ M M M _ N N . p n g

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27

Ferramentas auxiliares

Para os estudantes que o pretendam, é possível fazer uso de um comando disponibilizado na página Moodle da unidade curricular, e que se designa: `vs_getsubimages`.

Este comando implementa uma função que aceita uma matriz que contem uma das imagens a analisar, e devolve um *'cell array'* com todas as imagens dos objetos já extraídos, e separados da imagem original. Deste modo, os estudantes que optarem por esta solução não precisam de escrever o código para fazer essa deteção e separação. Porém, e como isso representa uma parte importante do trabalho, os programas entregues que usarem esta função/comando terão um **desconto** na avaliação de até **3 valores** (na escala de 0 a 20). O excerto de código abaixo ilustra um uso desta função:

```
close all
fName='sequencias/seq200/svpi2024_TP1_img_200_01.png';
Z=im2double(imread( fName ));
regions=vs_getsubimages(Z); %extract all regions
N=numel(regions);
SS=ceil(sqrt(N));
for k=1:N
    subplot( SS, SS, k)
    imshow( regions{k} )
end
```

Para esta facilidade poder ser usada, basta que o ficheiro `vs_getsubimages.p` esteja na pasta corrente do trabalho. No entanto, ele pode estar presente na pasta corrente e não ser usado!