

Sistemas de Visão e Percepção Industrial

Trabalho prático nº 1 – Março/Abril 2017

Deteção e análise de códigos de barras lineares e matriciais

Objetivo

Desenvolvimento de um programa em Matlab para fazer análise de imagens que contêm vários códigos de barras lineares e códigos matriciais. O programa deve ser capaz de interpretar imagens fornecidas e de gerar os resultados pedidos conforme descrito no enunciado. 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 serão em níveis de cinzento.
- As imagens conterão códigos de barras lineares e códigos matriciais.
- Os códigos de barras lineares e matriciais poderão existir em qualquer uma das 4 orientações possíveis.
- Os códigos de barras e matriciais existirão em diversas escalas, mas sempre em múltiplos inteiros de *pixels*.
- 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 serão quadrados com 2x2 *pixels*.
- O número total de códigos de barras e matriciais, e a informação que contêm, será variável em cada imagem.
- Os códigos de barras e matriciais estarão envolvidos por um caixilho quadrado ou retangular preto.
- Haverá um espaço branco variável entre o caixilho e os códigos propriamente ditos (a chamada *Quiet Zone*).
- O fundo da imagem, fora dos caixilhos, poderá conter ruído ou textura de intensidade não preta.
- As barras serão em níveis de cinzento mais escuras do que o seu fundo.

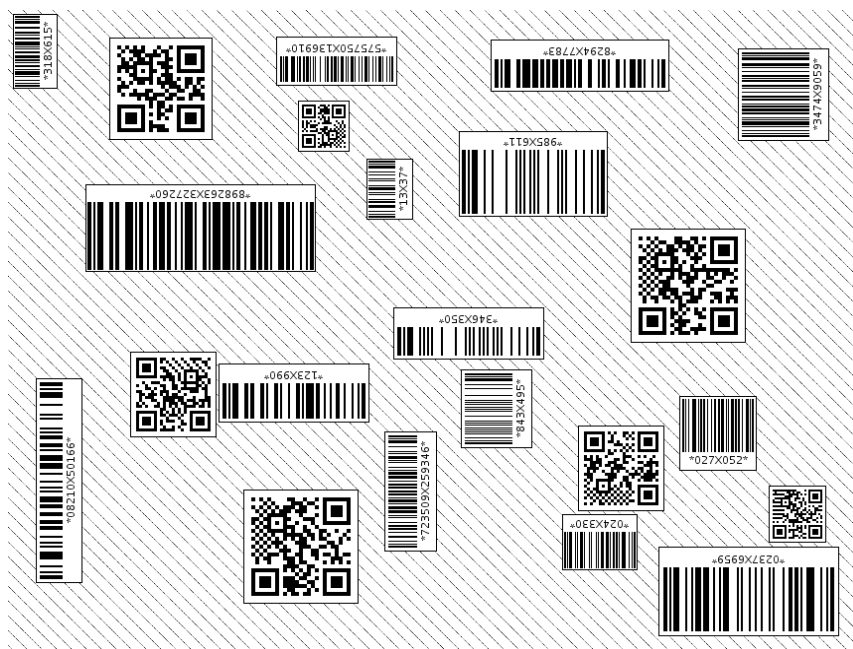


Fig. 1: Exemplo de uma imagem com códigos de barras e QR codes

Parâmetros a detetar em cada imagem

- Número **total** de códigos de barras presentes.
- Número **total** de códigos matriciais presentes.
- Número total de códigos de barras em cada uma das **4 orientações** possíveis.
- Número total de códigos matriciais em cada uma das **4 orientações** possíveis.
- Número de códigos de barras **inválidos** de acordo com o enunciado.
- Número de códigos matriciais **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 decrescente.

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.

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 o seguinte:

Delimitador de início	Dígito '5' na codificação 'L'	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

Fig. 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 dos restantes símbolos.

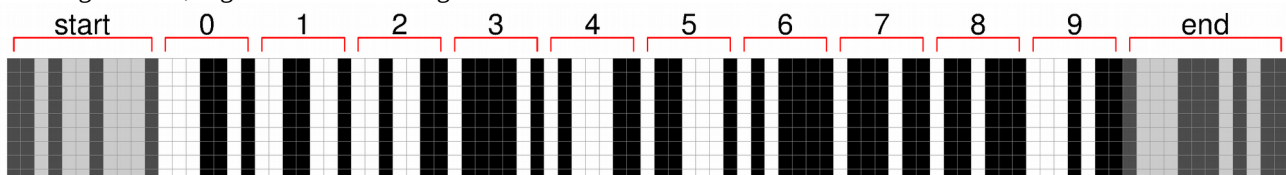


Fig. 3: Representação da sequência “0123456789” na codificação “L” com os delimitadores destacados a sombreado.

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 possíveis dos códigos de barras lineares.



Fig. 4: As codificações “L”, “R” e “G” para a 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 aparecer numa imagem em qualquer uma das 4 orientações, como ilustrado na figura 5:



Fig. 5: As 4 orientações possíveis dos códigos de barras (R0, R90, R180, R270)

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*. 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**. Assim, são legítimos os arranjos de: 21×21, 25×25, 29×29, etc. Arranjos fora desta regra são **inválidos**.
- 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.

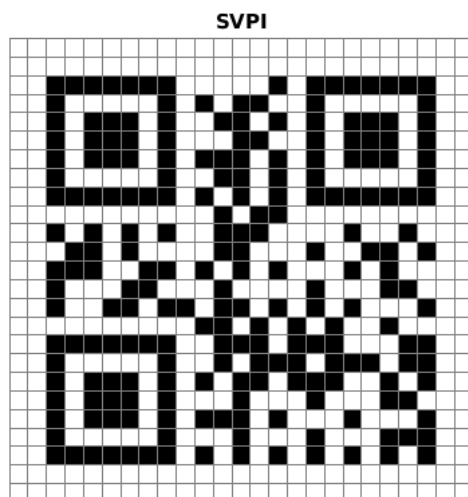


Fig. 6: QR code de 21×21 (menores dimensões válidas) que representa a string "SVPI". Cada quadrícula da grelha sobreposta no QR code delimita uma unidade de informação binária.

Os algoritmos de armazenamento de informação em QR codes são sofisticados mas, neste trabalho, a análise ao conteúdo dos QR codes será limitada ao seguinte:

- Detecção da sua orientação (múltiplos de 90°).
- Detecção da sua dimensão.

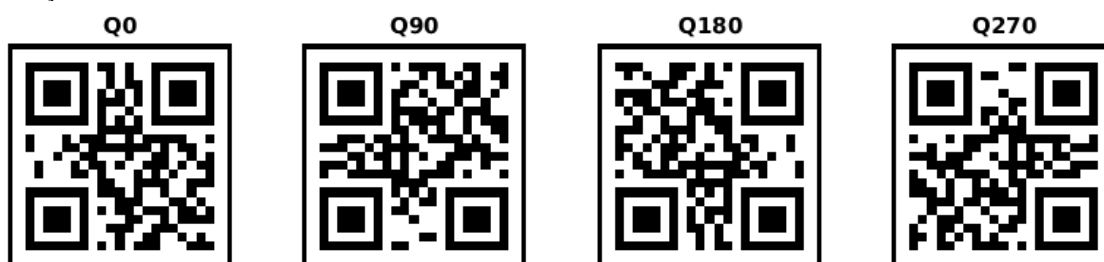


Fig. 7: Representação de um mesmo QR-code de 21×21 nas 4 orientações diferentes.

Algumas sugestões de metodologia para realizar o trabalho

- Detetar os cantos delimitadores dos caixilhos das imagens com filtros de convolução ou outras técnicas.
- Isolar as regiões dos códigos a partir das caixas individuais de cada código; 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 ou código matricial.
- Detetar a orientação do código e eventualmente reorientá-lo para o processar.
- Fazer os testes de verificação da validade do código e outras eventuais propriedades.

Resultados a gerar

Quando executado, o programa a desenvolver pelo aluno deve ler um conjunto de imagens fornecidas com nomes do género **svpi2017_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_#####.txt** onde ##### 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 99999, o código Matlab do trabalho deve gerar o ficheiro **tp1_99999.txt** que deverá conter os seguintes valores separados por vírgulas:

- Número mecanográfico do aluno; [NumMec]
- Número da sequência; [NumSeq]
- Número da imagem na sequência; [NumImg]
- Número **total** de códigos de barras presentes. [TotCB]
- Número **total** de códigos matriciais presentes. [TotQR]
- Número de códigos de barras em cada uma das **4 orientações** possíveis. [R0, R90, R180, R270]
- Número de códigos matriciais em cada uma das **4 orientações** possíveis. [Q0, Q90, Q180, Q270]
- Número de códigos de barras **inválidos** de acordo com o enunciado. [BadCB]
- Número de códigos matriciais **inválidos** de acordo com o enunciado. [BadQR]
- Número **total** acumulado de dígitos representados nos códigos de barras **válidos**. [TotDigCB]
- Número de códigos de barras **válidos** em cada uma das **3 codificações** possíveis ('L', 'R', 'G'). [CBL, CBR, CBG]
- *String* com os dígitos centrais dos códigos de barras **válidos ordenados** de forma decrescente. [StringCB]

Um parâmetro não verificado na imagem deve constar na tabela com valor 0.

Por exemplo, se o problema tivesse uma imagem com nome **svpi2017_TP1_img_123_01.png**, ilustrada na figura 8 do lado esquerdo, para o aluno com número 99999, o ficheiro **tp1_99999.txt** a gerar deveria ter o seguinte conteúdo e formato (uma linha, porque só há uma imagem. Para NN imagens, haveria NN linhas).

99999,123,1,17,8,4,5,6,2,1,4,3,0,5,1,92,4,5,3,876654442222

O significado desta linha compreende-se melhor pela análise da seguinte tabela e da imagem na figura 8:

NumMec	NumSeq	NumImg	TotCB	TotQR	R0	R90	R180	R270	Q0	Q90	Q180	Q270	BadCB	BadQR	TotDigCB	CBL	CBR	CBG	StringCB
99999	123	1	17	8	4	5	6	2	1	4	3	0	5	1	92	4	5	3	876654442222

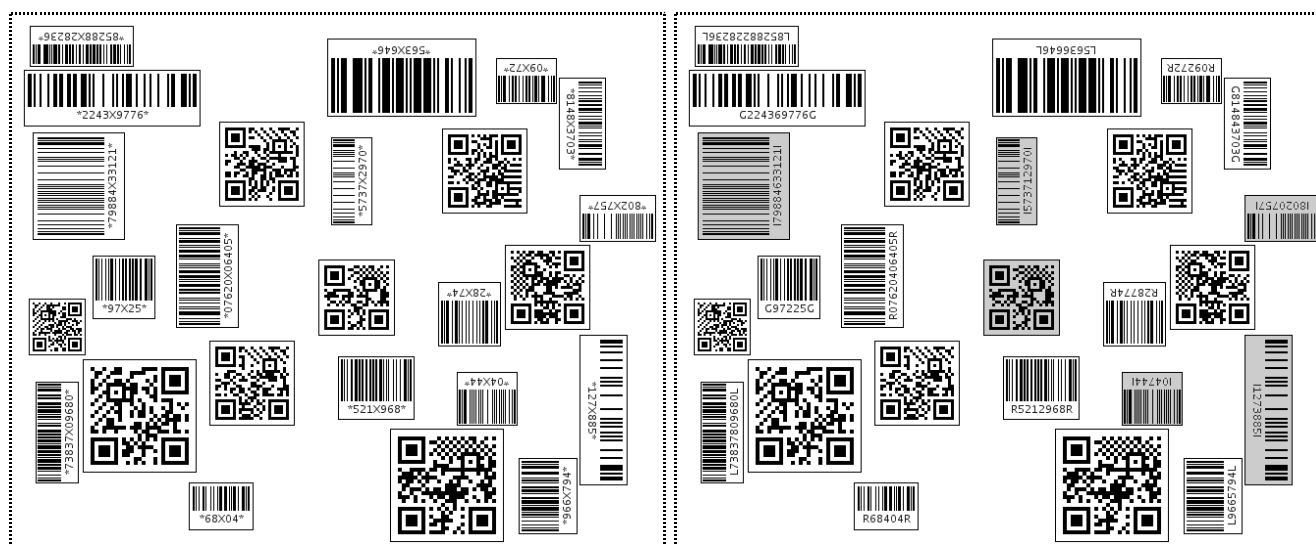


Fig. 8: Representação da imagem 1 da sequência 123. À esquerda está a imagem original a processar. À direita está uma imagem auxiliar que ilustra algumas soluções: a sombreado estão os códigos inválidos; todos os dígitos dos códigos de barras estão representados; a codificação dos códigos de barras está indicada [L,R,G, I(nválido)]

A figura 8 representa do lado esquerdo um exemplo de uma imagem a processar e do lado direito uma imagem auxiliar para melhor compreender as soluções (essa imagem auxiliar não existirá na avaliação!). Na imagem auxiliar é fácil identificar os códigos inválidos, os caracteres dos códigos de barras bem como a identificação da codificação usada em cada código de barras. Neste exemplo de imagem não há QR codes com orientação de 270°.

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.

Procedimento para entrega do trabalho

O aluno deve executar o comando `SVPI_ProcesSTP1_2017` (fornecido *on-line*) usando como argumento o nome do seu ficheiro Matlab. Por exemplo, se o programa do aluno se designar `tp1_99999.m` então, dentro da pasta que o contém, deve executar o comando: `SVPI_ProcesSTP1_2017('tp1_99999.m')`

Se o programa do aluno executar sem erros, será criado o ficheiro `cod_svpi2017_tp1_nnnnn.vsz` (com `nnnnn=99999` 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_nnnnn.txt` não for gerado pelo programa do aluno, então o comando `SVPI_ProcesSTP1_2017` 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 `cod_svpi2017_tp1_nnnnn.vsz` :

- O programa do aluno não pode ter erros de execução.
- A pasta onde está a ser executado não pode ter nenhum ficheiro do tipo `svpi2017_TP1_img_*.png`
- Os ficheiros de imagem `svpi2017_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_nnnnn.txt` onde `nnnnn` 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. Todos os trabalhos que apresentarem um índice de plágio maior do que um dado limite são passíveis de análise e tratamento específicos, conforme comunicado nas aulas.

Observações e recomendações sobre o trabalho e o Matlab

O trabalho deve ser feito para ser compatível em versões do Matlab R2012 ou posterior.

O programa a executar deve-se chamar `tp1_nnnnn.m` (onde `nnnnn` é o numero mecanográfico do aluno)

O programa deve procurar as imagens na pasta anterior à sua; usar o seguinte prefixo: `../`

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 e analisar.

O programa deve gerar, na pasta corrente, o ficheiro `TP1_nnnnn.txt`, (onde `nnnnn` 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/"`.

Nota: usar sempre o símbolo `"/"` como separador de pastas independentemente do sistema operativo onde o trabalho for desenvolvido.

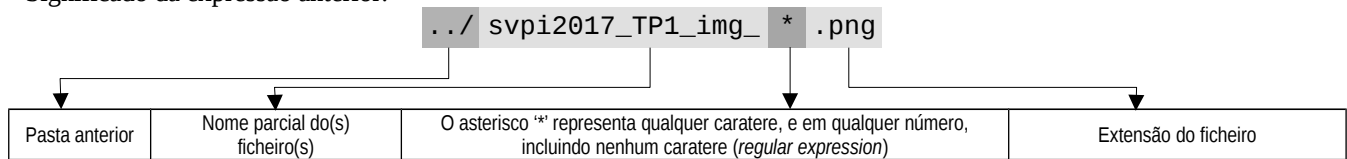
Ou seja, para tornar o programa independente do nome da pasta onde se encontre, ele deve procurar as imagens na pasta anterior que se representa pela expressão `"../"` (dois pontos e uma barra). Por exemplo, se o programa quisesse abrir a imagem `img1.png` que se encontra na pasta anterior, deverá utilizar um comando similar a:

```
A=imread('../img1.png');
```

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(' ../svpi2017_TP1_img_*.png');
```

Significado da expressão anterior:



Neste caso, a variável `listaF` conterá uma listagem de todos os ficheiros que obedecem à regra do nome das 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.
%o comando numel(listaF) dá o mesmo resultado.
%listaF(1).name -> dá o nome do primeiro ficheiro da lista.
%Nesta abordagem, não esquecer que para a abrir o ficheiro será
%preciso dar o caminho correto, ou seja, será preciso
%concatenar '../' com o nome. Por exemplo, para o 1º ficheiro da lista:
% A=imread( strcat('../', listaF(1).name));
% etc.
```

Para a geração de resultados de avaliação, além da contabilização dos parâmetros dos códigos de barras e matriciais, 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:

```
svpi2017_TP1_img_MMM_NN.png
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 designado `vs_getsubimages` que é disponibilizado na página Moodle da unidade curricular.

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 de códigos de barras e matriciais 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 usem 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/svpi2017_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!