

Sistemas de Visão e Percepção Industrial

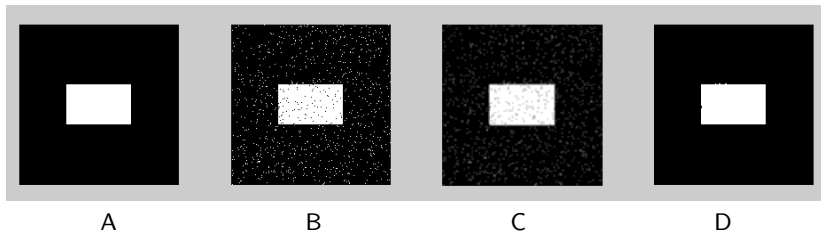
Aula Prática nº 4

Operações com filtros.

- 1 Filtros de média e mediana
- 2 Detecção e contagem de pontos isolados
- 3 Detecção de estruturas simples em imagens binárias

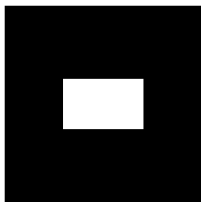
Exercício 1)

- Sintetizar uma imagem A de fundo preto de 200x200 com um retângulo branco de 50x80 no centro.
- Juntar-lhe ruído aleatório do tipo 'salt & pepper' a 5% usando a função `B=imnoise(..., 'salt & pepper',...);`
- Filtrar a imagem com filtro de:
 - média [`C=filter2(F,...);` com `F=ones(3,3)/9`]
 - mediana (3x3) [função `D=medfilt2(...,[3,3])`]
- Representar as 4 imagens num subplot, como ilustrado:

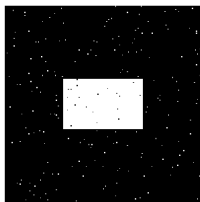


Exercício 2)

- Da imagem A do exercício anterior com ruído 'salt & pepper' a 1% (B), obter uma imagem que só contém os pontos isolados. (C)
- Contar e indicar automaticamente os pontos isolados de (C)



A



B



203 pontos isolados

C

- Notas:
 - Usar a função `imnoise()` com 'salt & pepper' a 1%.
 - Usar função `filter2()` com filtro de pontos isolados.
 - Inicializar o gerador aleatório com `rng(0)` para resultados iguais aos ilustrados.
 - Pode-se usar a função `abs()` para calcular valores absolutos.
 - Filtro de pontos isolados: $\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$.

Exercício 3)

- Escrevendo o código adequado, avaliar quais dos seguintes filtros também podem ser usados para detetar pontos isolados na imagem $T = \text{imnoise}(A, \text{'salt \& pepper'}, 0.005)$, onde A é a usada nos exercícios anteriores. Testar para pontos isolados brancos e pretos.

$$F1 = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad F2 = \begin{bmatrix} 1 & 2 & 1 \\ 2 & -12 & 2 \\ 1 & 2 & 1 \end{bmatrix} \quad F3 = \begin{bmatrix} -1 & 1 & -1 \\ 1 & 4 & 1 \\ -1 & 1 & -1 \end{bmatrix} \quad F4 = \begin{bmatrix} 1 & 2 & 3 \\ 4 & -100 & 5 \\ 6 & 7 & 8 \end{bmatrix}$$

- Usar o código num ciclo for e matrizes tri-dimensionais para facilitar a programação. Completar e adaptar a seguinte sugestão de código.

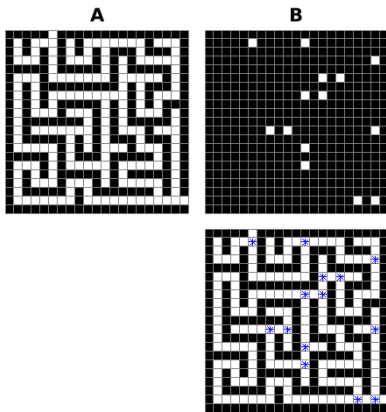
```
%% Ex 3 (inc)
rng(0);
T=imnoise(A, 'salt & pepper', 0.005);
F=zeros(3,3,4);
F(:,:,1)=[ 1 1 1; 1 -8 1; 1 1 1];
F(:,:,2)=[ 1 2 1; 2 -12 2; 1 2 1];
F(:,:,3)=[-1 1 -1; 1 4 1; -1 1 -1];
F(:,:,4)=[ 1 2 3; 4 -100 5; 6 7 8];
W=[ xxxx ]; %replace the missing code xxxx
Nw=[ xxxx ]; %replace the missing code xxxx
for n=1:4
    X=filter2( xxxx ,T); %replace the missing code xxxx
    ZW=(X == W(n));
    ZB=( xxxx ); %replace the missing code xxxx

    subplot(2,4, n), imshow(ZW)
    str=sprintf('Total %d', nnz(ZW)); xlabel(str);

    subplot(2,4,4+n), imshow(ZB)
    str=sprintf('Total %d', nnz(ZB)); xlabel(str);
end
```

Exercício 4)

- Determinar um filtro de 3x3, e a respectiva função discriminante, para obter a imagem B a partir de A, ou seja, criar um detetor de entroncamentos no labirinto presente em A.



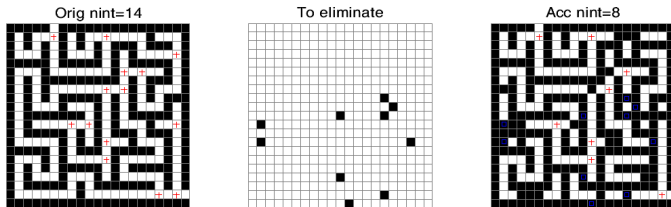
```
A=[  
0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
0 1 0 1 1 1 0 1 0 1 1 1 1 1 1 1 0 1 1 1 0  
0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 0 0 1 0 0 0 1 0  
0 1 1 1 0 1 0 1 1 1 0 1 0 1 0 1 1 1 1 1 1 1 0  
0 0 0 0 0 1 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 1 0  
0 1 1 1 0 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 0 1 0  
0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 1 0  
0 1 0 1 0 1 1 1 1 1 1 1 1 1 1 1 0 1 0 1 1 1 0  
0 1 0 1 0 1 0 0 0 0 0 0 0 0 1 0 1 0 1 0 0 0 0  
0 1 0 1 1 1 0 1 1 1 0 1 0 1 0 1 0 1 1 1 0 1 0  
0 1 0 0 0 0 0 0 0 0 1 0 1 0 1 0 0 0 0 0 0 1 0  
0 1 0 1 1 1 1 1 1 1 1 1 0 1 0 1 1 1 1 0 1 1 0  
0 1 0 0 0 0 0 0 1 0 1 0 1 0 0 0 0 1 0 0 0 1 0  
0 1 1 1 1 1 0 1 0 1 0 1 0 1 1 1 0 1 1 1 0 1 0  
0 0 0 0 0 1 0 1 0 0 0 1 0 0 0 0 0 0 1 0 1 0  
0 1 1 1 0 1 0 1 1 1 1 1 1 0 1 1 1 1 1 1 0 1 0  
0 1 0 1 0 1 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 1 0  
0 1 0 1 1 1 0 1 1 1 1 0 1 0 1 1 1 1 1 1 0 1 0  
0 1 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 1 0 1 0  
0 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1  
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
];
```

Ilustração dos pontos relevantes na imagem A.

Exercício 5) Opcional

Sendo $Z=A$ (imagem do exercício anterior), escrever um programa que:

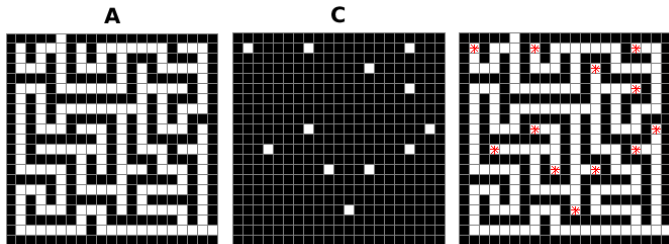
- 1 Gere uma imagem aleatória X da mesma dimensão de Z com cerca de 2.5% de pixels a preto;
- 2 Altere a imagem Z forçando a preto os mesmos pixels pretos de X ;
- 3 Calcule o número de entroncamentos em Z : $nint$;
- 4 Execute ciclicamente os pontos 1, 2 e 3 enquanto $nint > 0$.
- 5 Ilustre a evolução da imagem Z sucessivamente alterada.



Exemplo de uma etapa intermédia quando $nint=8$

Exercício 6)

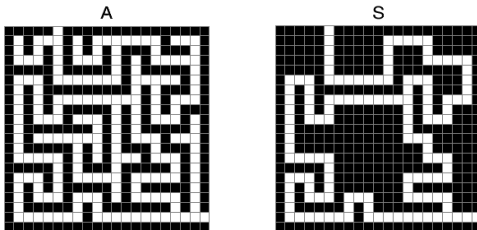
- Determinar um filtro de 3×3 , e a respetiva função discriminante, para obter a imagem C a partir de A, ou seja, criar um detetor de 'becos sem saída' no labirinto presente em A (que é a mesma do exercício 4).
 - Os pontos de entrada/saída do labirinto não são 'becos sem saída'.



- Será possível definir um mesmo filtro que possa ser usado para os exercícios 4 e 6?

Exercício 7) Opcional**

- Encontrar o caminho da solução do labirinto por eliminação recursiva de todos os becos sem saída até não haver mais nenhum, ou seja, gerar a imagem S a partir da imagem A.
 - NB: Acautelar a não eliminação da entrada/saída do labirinto.



- Para aferir a robustez do algoritmo testá-lo em versões rodadas de A. Sugestão: usar a função `rot90()` do matlab para rodar uma matriz em múltiplos de 90° .