Trabalho de LAPR1

Reconstrução e Identificação de Imagens Utilizando Eigenfaces

1 Preâmbulo

A unidade curricular Laboratório/Projeto I é uma unidade que tem por objetivo consolidar e estender os conhecimentos de todas as unidades curriculares que funcionam no primeiro semestre. Esta unidade curricular inclui também um módulo de competências pessoais e métodos de trabalho que visam preparar os alunos para trabalhar eficazmente em equipas, melhorar a elaboração de relatórios e desenvolver competências de apresentação oral.

Assim, com o projeto descrito neste enunciado pretende-se que os alunos desenvolvam uma aplicação em linguagem *Java* (Horstmann, 2015) onde apliquem um processo básico de desenvolvimento de aplicações informáticas, valorizando todas as fases do ciclo de desenvolvimento, desde a análise e conceção aos testes de validação. Pretende-se também que os alunos elaborem um relatório que descreva a aplicação concebida, o processo de desenvolvimento e que apresentem e critiquem os resultados obtidos.

No corrente documento é apresentada o enunciado do projeto.

2 Objetivos

O projeto a desenvolver no corrente ano letivo é uma aplicação informática para a instituição *European Intelligence Agency (EIA)* que permita representar e identificar indivíduos utilizando eigenfaces (vetores próprios) (Turk & Pentland, 1991) obtidos de um conjunto de imagens/fotografias de faces humanas. A EIA possui uma base de imagens de grande dimensão, que não pretende partilhar, e quer encontrar uma representação compacta para esta base de imagens que possa ser distribuída por todas as instituições europeias afiliadas para identificar indivíduos que circulam em cada um dos países onde estas atuam. Para atingir os seus objetivos a EIA pretende que a base do sistema a desenvolver seja a decomposição de uma imagem/matriz em valores e vetores próprios, em particular as eigenfaces. A EIA pretende também um estudo de sensibilidade ao número de eigenfaces (de valores e vetores próprios) a utilizar na decomposição de cada imagem.

2.1 Funcionalidades da Aplicação e Algoritmos

Considerando que uma imagem na escala de cinzento pode ser representada através de uma matriz, para facilitar o desenvolvimento e avaliação do projeto, a EIA define três tarefas/funcionalidades a incluir na aplicação:

1. Decomposição Própria (Larson, 2012; Ghaoui L., 2024) de uma matriz simétrica;
2. Reconstrução de cada imagem (imagem em escala de cinzento (0-255)) utilizando as eigenfaces (Turk & Pentland, 1991) mais relevantes;
3. Dada uma imagem, identificar a imagem mais próxima/parecida que existe na base de imagens utilizando os pesos de um conjunto de eigenfaces.

Para a execução destas tarefas é também necessário implementar módulos para leitura e escrita de matrizes/imagens guardadas em pastas e módulos de cálculo de um conjunto de estatísticas e métricas.

2.1.1 Decomposição Própria de uma Matriz Simétrica

A decomposição própria (Larson, 2012; Ghaoui L., 2024) de uma matriz é um processo utilizado para expressar uma matriz através do produto de três matrizes, combinando vetores próprios e valores próprios. É utilizada em áreas como a Resolução de Sistemas Lineares, a Modelação de Sistemas Dinâmicos e a Compressão de Imagem.

A decomposição própria de uma matriz simétrica A pode ser definida como:

$$A = PDP^T,$$

em que P é a matriz cujas colunas são os vetores próprios unitários de A; D é uma matriz diagonal cujos elementos são os valores próprios associados aos vetores próprios de A e P^T é a matriz transposta de P.

Em algumas áreas de aplicação, como a compressão de imagem, por vezes é necessário reduzir a dimensão da representação e faz-se a decomposição própria utilizando um subconjunto de vetores próprios unitários. Esta técnica tem por objetivo aproximar a matriz original utilizando apenas os vetores próprios associados aos valores próprios de maior valor absoluto. Ao utilizar este processo, reduzimos a complexidade computacional preservando as características principais da matriz.

Se utilizarmos apenas um subconjunto de k vetores próprios (associados aos valores próprios de maior valor), obtemos uma aproximação de A representada por

$$A_k = P_k D_k P_k^T,$$

em que P_k contém apenas os k vetores próprios unitários selecionados; D_k é a matriz diagonal contendo os k valores próprios com maior valor absoluto e P_k^T é a matriz transposta de P_k.

Para calcular a qualidade da reconstrução/aproximação utilizamos uma métrica como o Erro Absoluto Médio

$$EAM(A, A_k) = \frac{1}{MN} \sum_{i=1}^{M} \sum_{j=1}^{N} |A(i,j) - A_k(i,j)|,$$

em que A e A_k são as matrizes que representam, respetivamente, a matriz original e a matriz reconstruida com os k valores e vetores próprios selecionados. As variáveis M e N representam, respetivamente, o número de linhas e colunas da matriz.

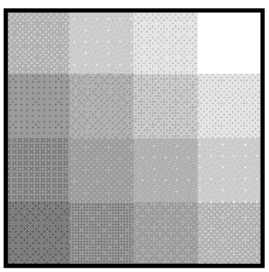
2.1.2 Reconstrução e Identificação de Imagens utilizando Eigenfaces

A reconstrução e identificação de imagens digitais são tarefas de primordial importância que podem ser realizadas de forma automática utilizando um conjunto de algoritmos baseados em eigenfaces. Nesta secção é apresentado o conceito de imagem digital e dois algoritmos, um para reconstruir imagens e outro para identificar a imagem mais próxima existente numa base de imagens.

2.1.2.1 Imagem Digital

Uma imagem digital é uma representação discreta de uma imagem bidimensional utilizando um conjunto finito de números em que cada elemento da imagem (um pixel) tem um valor para uma determinada localização. Por outras palavras, uma imagem digital é uma função bidimensional $f(x, y)$, em que (x, y) e o valor de f são quantidades discretas e finitas proporcionais à intensidade da imagem (Queiroz & Gomes, 2006; Gonzalez & Woods, 2006).

Uma imagem digital pode ser representada através de uma matriz, sendo que cada coeficiente da matriz representa a intensidade da imagem numa determinada localização. Na Figura 1 está



124	135	141	255
110	120	128	141
108	115	120	135
98	110	119	128

Figura 1: Exemplo de imagem digital e respetiva matriz. Neste exemplo é utilizada um escala de cinzentos (Fonte: Revista Informática na Medicina (www.informaticamedica.org.br))

representada uma imagem e a respetiva representação matricial. Neste caso foi utilizada a escala de cinzentos para representar a imagem e daí os coeficientes da matriz terem valores numéricos inteiros compreendidos no intervalo 0 a 255.

2.1.2.2 Reconstrução/Representação de Imagens utilizando Eigenfaces

A reconstrução de uma imagem digital a partir de sua representação em um espaço de menor dimensão é uma operação extremamente útil, especialmente ao armazenar ou transmitir imagens pela internet, sobretudo no caso em que estamos a trabalhar com imagens de grande dimensão. Existem vários métodos que permitem comprimir imagens (transformada discreta de cosseno, compressão fratal, compressão por Wavelets, SVD, etc.), uns com perdas de informação e outros sem qualquer perda (Gonzalez & Woods, 2006). Entre os métodos mais simples estão aqueles que são baseados na decomposição de uma matriz/imagem utilizando os seus valores e vetores próprios. Este métodos reconstroem/representam uma matriz/imagem utilizando um subconjunto de valores e vetores próprios. Sendo que a perda de informação está relacionada com o número de valores próprios utilizados na reconstrução, em que, quanto maior o seu número melhor é a qualidade da imagem reconstruida.

Um método particularmente simples e eficaz para reconstruir/representar imagens é a utilização de eigenfaces. De uma forma muito sucinta podemos descrever este método em 6 passos:

1. Transformar cada imagem da base de imagens, com N imagens, para o formato vetor;
2. Construir uma matriz de imagens, M, em que cada coluna da matriz é uma imagem da base de imagens;
3. Construir a matriz de covariâncias C :
   - Calcular o vetor (coluna) média, μ, com a média de todas as imagens presentes na matriz;
   - Para cada imagem I_i, calcular $\Phi_i = I_i - \mu$;
   - Construir uma matriz $A = [\Phi_1, \ldots, \Phi_N]$;
   - Calcular $C = \frac{1}{N} AA^T$;
4. Calcular os valores e vetores próprios de C;
   Nota: Como AA^T e A^TA têm os mesmos valores próprios podemos utilizar (para otimizar o cálculo (trabalhar com uma matriz de menor dimensão)) A^TA em vez de AA^T. Obtendo

depois os valores próprios de AA^T fazendo $Av_i = u_i$, em que v_i são os vetores próprios de A^TA.

5. Normalizar Vetores Próprios da matriz C por forma a que $||u_i|| = 1$. Estes vetores próprios normalizados são também designados por eigenfaces;
6. Reconstruir a imagem I_i utilizando todos ou um subconjunto de k eigenfaces e os pesos associados a cada eigenface:

$$\bar{I}_i = \mu + \sum_{j=1}^{k} u_j w_j,$$

em que $w_j = u_j^T \Phi_i$.

Para calcular a qualidade da reconstrução utilizamos uma métrica como o Erro Absoluto Médio $EAM(I, \bar{I})$, em que I e $\bar{I}$ são as matrizes que representam, respetivamente, a imagem original e a imagem reconstruida com k valores próprios.

2.1.2.3 Identificação de Imagens utilizando Eigenfaces

A identificação de indivíduos por comparação com cada imagem presente numa base de imagens é uma tarefa que é realizada em várias atividades, como a monitorização de espaços públicos para identificar potenciais terroristas. Sendo a identificação por comparação de imagens uma tarefa de grande e crescente complexidade para os seres humanos, principalmente em tarefas de monitorização em que estão presentes um grande número de indivíduos e câmaras a recolher imagens em espaços públicos, algoritmos baseados em eigenfaces podem ser utilizados na tarefa para identificar indivíduos de forma automática.

Um algoritmo simples e eficaz consiste em comparar os pesos associados a cada eigenface da imagem alvo com os pesos de cada uma das imagens disponíveis numa base de imagens. Em detalhe, dada uma imagem de um indivíduo, I^{nova}, e uma base de imagens, D, podemos identificar um individuo realizando os seguintes passos (algumas variáveis estão descritas na Secção 2.1.2.2):

1. Calcular $\Phi^{nova} = I^{nova} - \mu$, em que μ é o vetor (coluna) média de todas as imagens;
2. Calcular $w_j^{nova} = u_j^T \Phi^{nova}$, sendo que u_j são as eigenfaces calculadas da matriz de covariâncias (C);
3. Construir um vetor $\Omega_{nova} = [w_1^{nova}, \ldots, w_k^{nova}]$ para a imagem a identificar e um vetor $\Omega_j = [w_1^j, \ldots, w_k^j]$, para cada uma das N imagens da base de imagens;
4. Calcular a distância euclidiana entre Ω_{nova} e cada um dos Ω_j;
5. Devolver a imagem da base de imagens que está associada a Ω_j e cuja distância a Ω_{nova} é a menor de todas as distâncias calculadas.

O número de pesos (e eigenfaces) a utilizar na comparação também pode ser um subconjunto das eigenfaces existentes. No caso de não utilizarmos todas as eigenfaces, devemos selecionar as eigenfaces associadas aos maiores valores próprios.

2.1.2.4 Visualização de Imagens

A visualização de uma imagem digital consiste em mapear cada um dos coeficientes de uma matriz num pixel de um qualquer ecrã. Em alternativa, também se podem guardar imagens digitais em ficheiros JPG, GIF ou em outros formatos.

3 Trabalho a Desenvolver

O trabalho a desenvolver no projeto de LAPR1 pode ser dividido num conjunto de tarefas:

- Estudo das técnicas de decomposição própria de matrizes (ver Secção 2.1.1) e reconstrução (ver Secção 2.1.2.2) e identificação (ver Secção 2.1.2.3) de imagens.
- Desenvolvimento, em linguagem Java, de uma aplicação que executa na consola, com interface em modo de texto, com menus, que implementa as seguintes funcionalidades:
  - Decomposição própria de uma matriz simétrica (conforme descrito na Secção 2.1.1)). O cálculo dos valores e vetores próprios deve ser realizado com o auxílio da biblioteca la4j - linear algebra for Java (http://la4j.org/apidocs/).
    A aplicação deve permitir ao utilizador definir o número de valores próprios a utilizar na decomposição e a localização do ficheiro CSV que contém a matriz de entrada. Após efetuar a decomposição, a aplicação deve apresentar na consola o número e os valores e vetores próprios utilizados na decomposição e também o EAM. A matriz reconstruida deve ser apresentada na consola e escrita para uma pasta com a designação *Output*, num ficheiro com o formato CSV.
  - Reconstrução de cada uma das imagem disponíveis numa pasta utilizando as eigenfaces (conforme descrito na Secção 2.1.2.2). A aplicação deve permitir ao utilizador definir o número de eigenfaces a utilizar na reconstrução e a pasta onde estão localizadas as imagens a reconstruir. Como output a aplicação deve apresentar na consola o vetor μ, a matriz de covariâncias C e o número de eigenfaces utilizadas na reconstrução de cada imagem. Para cada imagem I da base de imagens deve também ser apresentado na consola o vetor com os pesos w que foram utilizados na reconstrução da imagem, gerando a imagem $\bar{I}$. As eigenfaces e cada uma das imagens reconstruidas devem ser escritas, simultaneamente nos formatos CSV e JPG, para pastas com a designação, respetivamente, *Eigenfaces* e *ImagensReconstruidas*.
  - Identificar a imagem mais próxima/parecida que existe numa dada base de imagens, a uma dada imagem (conforme descrito na Secção 2.1.2.3). A aplicação deve permitir ao utilizador especificar o número de eigenfaces a utilizar na identificação da imagem mais próxima/parecida, especificar a localização da nova imagem e a localização da base de imagens. Como output a aplicação deve apresentar na consola o número de eigenfaces/pesos utilizados na comparação e o vetor Ω_{nova}. Para cada imagem também deve ser apresentada na consola o vetor de pesos Ω_i e a distância euclidiana de cada vetor Ω_i ao vetor Ω_{nova}. A imagem mais próxima deve ser escrita, no formato JPG, para uma pasta com a designação *Identificacao*.
- No desenvolvimento devem ser utilizadas boas práticas de codificação e registo de informação, tal como ensinado em APROG.
- Elaborar um relatório em que são descritas: as técnicas/métodos de decomposição de matrizes e reconstrução e identificação de imagens; a metodologia de trabalho que utilizaram para desenvolver a aplicação; a implementação das técnicas/métodos; apresentação e discussão dos resultados obtidos. A introdução das técnicas/métodos de decomposição de matrizes e reconstrução e identificação de imagens deve ser realizado com o auxílio de exemplos ilustrativos.

3.1 Formato dos Dados de Entrada e Saída

A aplicação a desenvolver neste projeto terá que processar matrizes e imagens quadradas com até 256 linhas e 256 colunas. Todos as matrizes e imagens de entrada serão representadas em ficheiros CSV e as matrizes e imagens de saída serão representadas através ficheiros CSV ou JPG, consoante as funcionalidades. No moodle serão disponibilizados exemplos de matrizes, imagens e pastas com conjuntos de imagens a utilizar nas diferentes funcionalidades. Nas matrizes/imagens representadas no formato CSV, os coeficientes da matriz são números inteiros no intervalo 0 a 255 que aparecem separados por uma vírgula e sem espaços em branco a separar valores e vírgulas.

4 Outros Requisitos

- A aplicação deve ser desenvolvida em linguagem *Java* e estruturada e organizada em módulos. Será valorizada uma correta decomposição modular e o reaproveitamento de módulos. No final do trabalho deve resultar um ÚNICO projeto.
- Para o cálculo exato dos valores e vetores próprios será utilizada a biblioteca la4j - linear algebra for Java (http://la4j.org).
- A geração de ficheiros JPG deve ser realizada seguindo exemplo disponibilizado no moodle.
- A aplicação pode ser executada em modo interativo ou sem interação por parte do utilizador.
  - No modo interativo, a aplicação deverá ser chamada da linha de comandos utilizando o comando: *java -jar nome_programa.jar*. Neste modo todos os parâmetros necessários são solicitados em tempo de execução ao utilizador.
  - No modo não interativo, o utilizador especifica todos os parâmetros necessários à execução da aplicação na linha de comando, sendo necessário especificar: a funcionalidade a executar, o número de eigenfaces/pesos a utilizar, a localização da pasta com matriz/imagem de input, a localização da base de imagens e o nome do ficheiro de saída TXT onde é guardada toda a informação que seria apresentada na consola se a funcionalidade fosse executada no modo interativo. Neste modo, o comando terá a seguinte sintaxe: *java -jar nome_programa.jar -f X -k Y -i Z -d W nome_ficheiro_saida.txt*. Neste caso:
    - o valor associado ao parâmetro f identifica a funcionalidade a executar. X toma os valores 1, 2 ou 3, representando, respetivamente, as funcionalidades de decomposição própria de matriz, a reconstrução de imagens utilizando eigenfaces e a identificação de imagens utilizando pesos das eigenfaces.
    - o valor associado ao parâmetro k identifica o número de vetores próprios / eigenfaces a utilizar na decomposição/reconstrução/identicação. Y toma valores inteiros positivos e -1. Caso o valor de Y seja -1 ou um valor superior ao número de valores próprios reais existentes, na decomposição/reconstrução/identicação devem ser utilizados todos os valores próprios reais da matriz.
    - o valor associado ao parâmetro i identifica a localização do ficheiro CSV onde está localizada a matriz/imagem de input a utilizar nas funcionalidade 1 e 3. Para executar a funcionalidade 2 não é necessário especificar este parâmetro.
    - o valor associado ao parâmetro d identifica a localização da base de imagens a utilizar nas funcionalidade 2 e 3. Para executar a funcionalidade 1 não é necessário especificar este parâmetro.

- Todos os métodos desenvolvidos terão, obrigatoriamente, de estar associados a testes unitários.

```
Bool test_find_max_eigenvalue(matrix, expectedMaxEigenValue)
{

maxEigenvalue = find_max_eigenvalue(matrix);

if(expectedMaxEigenValue==maxEigenvalue)
return True;
else
return False;

}
```

Algoritmo 1: Exemplo de métodos de teste unitários

Por exemplo, se o aluno criar o método $find\_max\_eigenvalue(matrix)$ para determinar o maior valor próprio de uma matriz, também deve criar o método $test\_find\_max\_eigenvalue(matrix, expectedMaxEigenValue)$ (ver Algoritmo 1), em que *expectedMaxEigenValue* é o valor do maior valor próprio conhecido da matriz. Estes testes são extremamente úteis para determinar se os métodos estão de acordo com a sua especificação e se a edição destes não alterou a funcionalidade.

- A aplicação deve ter um interface simples e intuitivo que permita ao utilizador interagir com a aplicação de forma rápida e minimizando os erros. A interface da aplicação será em modo de texto e deverá incluir um conjunto de menus que permita aceder às funcionalidades de forma rápida. A interface da aplicação também deve permitir parametrizar cada uma das funcionalidades selecionadas de acordo com o que é apresentado nos objetivos do projeto.

5 Método de Trabalho

- Todos os alunos devem utilizar a metodologia de trabalho definida no eduScrum (Delhij & Solingen, 2013). Cada um dos grupos deve escolher um Scrum Master e este deve ser responsável por gerir a execução de tarefas. A aplicação GitHub Projects (https://github.com/) deve ser utilizada como suporte à gestão do projeto.

- A aplicação será desenvolvida utilizando o sistema de controle de versões Git e o GitHub (https://github.com/). Todos os alunos terão que criar uma conta no GitHub com o endereço de email do ISEP (i.e. 1XXXXXX@isep.ipp.pt) e cada grupo terá acesso a um repositório, a que terá de associar uma *project board* de acordo com o template fornecido. A designação do repositório e da *project board* deve seguir o formato: *lapr1-24-25_TTT_GG* em que TTT são as 3 letras da turma (ex: DAB) e GG corresponde a 2 dígitos do número de grupo (ex: 01). O repositório deve ser partilhado com todos os docentes que lecionam a turma onde o grupo está inserido.

- O grupo deve guarda todo o material desenvolvido para a realização do projeto (ex: relatório e apresentação) no Canal Privado do Teams. Apenas o código Java existente no GitHub não deve ser guardado no Canal Privado do Teams. Todos os docentes que lecionam a turma onde o grupo está inserido devem ter acesso a todo o material.

6 Submissão do Trabalho

O trabalho desenvolvido deverá ser entregue através de link disponível no moodle até às 23h55m do dia 12 de Janeiro de 2024. Na submissão devem constar obrigatoriamente três ficheiros:

- O projeto Java, incluindo toda a estrutura de diretorias e ficheiros do projeto, num único ficheiro ZIP. O executável também deve estar incluído no ficheiro ZIP.
- Relatório em formato PDF e não ultrapassando as 30 páginas. A escrita do relatório deve seguir as instruções formais e o modelo a ser disponibilizado nas aulas TP (módulo de competências).
- Apresentação em formato PDF não ultrapassando os 8 slides.

Nota: Os ficheiros deverão identificar, obrigatoriamente, o número do grupo e a turma a que os alunos pertencem (Exemplo: "lapr1-24-25_DAB_01_projeto.ZIP", "lapr1-24-25_DAB_01_relatorio.PDF"e "lapr1-24-25_DAB_01_apresentacao.PDF").

Referências

Delhij, A., & Solingen, R. (2013). *The eduscrum guide: The rules of the game.* (Disponível em http://eduscrum.nl/file/CKFiles/The_eduScrum_Guide_EN_December_2013_1.0.pdf)

Ghaoui L., C. G., Tsai A. (2024). *Linear algebra and applications.* VINUNIVERSITY.

Gonzalez, R. C., & Woods, R. E. (2006). *Digital image processing (3rd edition).* Upper Saddle River, NJ, USA: Prentice-Hall, Inc.

Horstmann, C. (2015). *Big java: Early objects, 6th edition.* Wiley. Retrieved from `https://books.google.pt/books?id=ib12CwAAQBAJ`

Larson, R. (2012). *Elementary linear algebra.* Cengage Learning.

Queiroz, J. E., & Gomes, H. M. (2006). Introdução ao processamento digital de imagens. *RITA*, *13*(2), 11–42.

Turk, M., & Pentland, A. (1991). Face recognition using eigenfaces. In *Proceedings. 1991 ieee computer society conference on computer vision and pattern recognition* (p. 586-591). doi: 10.1109/CVPR.1991.139758