

# MC920 - INTRODUÇÃO AO PROCESSAMENTO DIGITAL DE IMAGEM

## 2S2019

### RELATÓRIO TÉCNICO - TRABALHO 5

Andressa Gabrielly Macedo Marçal  
RA 262878

## 1. Especificação do Problema

A Analise de Componentes Principais (do inglês, Principal Component Analysis - PCA) é uma formulação matemática utilizada na redução de dimensionalidade de dados.

No contexto de processamento de imagens, a técnica PCA pode ser empregada na compressão de imagens com um certo grau de perda.

Na abordagem PCA, a informação contida no conjunto de dados é representada por meio de uma estrutura com dimensões reduzidas baseada na projeção dos dados em um subespaço gerado por um sistema de eixos ortogonais.

O sistema de eixos pode ser obtido por meio da técnica de Decomposição em Valores Singulares (do inglês, Singular Value Decomposition - SVD).

### 1.1. Objetivo

O objetivo deste trabalho é aplicar a técnica SVD para comprimir imagens digitais. A SVD de uma matriz real é:

$A_{n \times p}$  corresponde a fatoração

$$A = U \Sigma V^T$$

em que  $U$  é uma matriz unitária real  $n \times n$ , uma matriz retangular diagonal  $n \times p$  com números reais não-negativos na diagonal e  $V$  uma matriz unitária real  $p \times p$ .

## 2. Implementação

Abaixo o pseudocódigo de como deve ser feita a implementação.

---

#### Algoritmo 1: Compressão com Análise de Componentes Principais

---

```
input : Imagem  $f$  com dimensões  $M \times N$  pixels
        Número de componentes  $k$ 
output: Imagem  $g$  com dimensões  $M \times N$  pixels
1 # dividir a imagem RGB em três canais e aplicar a técnica SVD em cada canal
2 for  $i = 1 : 3$  do
3    $[U_f(:, :, i), S_f(:, :, i), V_f(:, :, i)] = \text{svd}(\text{double}(f(:, :, i)))$ 
4 # considerar apenas  $k$  componentes e combinar novamente os canais
5  $g = \text{zeros}(M, N)$ 
6 for  $i = 1 : 3$  do
7    $U_g(:, 1:k, i) = U_f(:, 1:k, i)$ 
8    $S_g(1:k, 1:k, i) = S_f(1:k, 1:k, i)$ 
9    $V_g(1:k, :, i) = V_f(1:k, :, i)^T$ 
10   $g(:, :, i) = U_g(:, :, i) * S_g(:, :, i) * V_g(:, :, i)$ 
11 return  $g$ 
```

---

Figure 1. Pseudocódigo

## 3. Avaliação da Compressão

A avaliação da compressão deve ser realizada pela taxa de compressão e pela raiz do erro médio quadrático (**RMSE**), cujas medidas são definidas como:

$$\rho = \frac{\text{quantidade de memória requerida para representar } g}{\text{quantidade de memória requerida para representar } f}$$

$$\text{RMSE} = \sqrt{\frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} [f(x, y) - g(x, y)]^2}$$

Figure 2. Calculo do RMSE

em que  $f$  e  $g$  são as imagens original e comprimida, respectivamente.

## 4. Execução do Script

1. Executar o script **trabalho5.py** diretamente no terminal do sistema Linux;
  - Comando: **python3 trabalho5.py**

```
watch.png: k = 1. RMSE: 33.015186905085315. Ratio: 0.7172106688702199.
watch.png: k = 5. RMSE: 23.608257749599318. Ratio: 1.014637274480099.
watch.png: k = 10. RMSE: 20.157342652212705. Ratio: 1.1777633360877748.
watch.png: k = 15. RMSE: 17.963105337846148. Ratio: 1.2726581508515815.
watch.png: k = 20. RMSE: 16.49907844637902. Ratio: 1.3540734173438003.
watch.png: k = 25. RMSE: 15.356763518769162. Ratio: 1.414171888628747.
watch.png: k = 30. RMSE: 14.383707833426142. Ratio: 1.4670528623238306.
watch.png: k = 35. RMSE: 13.586627056792364. Ratio: 1.5174534040306662.
watch.png: k = 40. RMSE: 12.884113780911878. Ratio: 1.569728974429601.
watch.png: k = 45. RMSE: 12.24339346430448. Ratio: 1.6096941996051968.
watch.png: k = 50. RMSE: 11.657865695848281. Ratio: 1.6414333998990038.
watch.png: k = 60. RMSE: 10.60747598752051. Ratio: 1.6937347931873479.
watch.png: k = 70. RMSE: 9.707419250820086. Ratio: 1.7289615181563605.
watch.png: k = 80. RMSE: 8.90798718728051. Ratio: 1.7549637331864298.
watch.png: k = 90. RMSE: 8.201742539082863. Ratio: 1.7729802336960015.
watch.png: k = 100. RMSE: 7.573434431645204. Ratio: 1.7864533122159483.
watch.png: k = 120. RMSE: 6.516640762414662. Ratio: 1.8056598035165037.
watch.png: k = 140. RMSE: 5.668920996234231. Ratio: 1.817677489326539.
watch.png: k = 160. RMSE: 4.981211463993412. Ratio: 1.824905629151173.
watch.png: k = 180. RMSE: 4.40984280863134. Ratio: 1.8303794817059174.
watch.png: k = 200. RMSE: 3.9186040753786817. Ratio: 1.8282892622687417.
watch.png: k = 220. RMSE: 3.495895181085988. Ratio: 1.8240428315659.
watch.png: k = 240. RMSE: 3.128730841977619. Ratio: 1.8179256759858606.
watch.png: k = 250. RMSE: 2.964538438312323. Ratio: 1.8141081921682045.
```

Figure 3. Script em execução

### 4.1. Bibliotecas Utilizadas

- **OpenCV2:** import cv2
- **Numpy:** import numpy as np
- **Matplotlib:** import matplotlib.pyplot as plt
- **OS:** import os

### 4.2. Funções Utilizadas

- **def return\_svd(image, k):**
  - **@param1 image:** Matriz de pixels M x N;
  - **@param2 k:** Número de componentes;
  - **@return U, S, V;**
- **def compress(image, k):**

Método para compressão.

  - **@param1 image:** Matriz de pixels M x N;
  - **@param2 k:** Número de componentes;

- **@return new\_image;**

- **def rmse\_images(image, new\_image):**

Método para calcular taxa de compressão e raiz do erro médio quadrático (RMSE) da imagem.

- **@param1 image;**

- **@param2 new\_image;**

- **@return rmse;**

- **def plot\_(x, rmse, ratio, nome):**

Método para plotagem dos resultados e do gráfico da imagem;

- **@param1 x;**

- **@param2 rmse;**

- **@param3 ratio;**

- **@param4 nome;**

- **def main(arq):**

Função principal, onde chama o menu de opções do script;

### 4.3. Entrada de Dados

As imagens de entrada estão no formato PNG (*Portable Network Graphics*). Alguns exemplos encontram-se disponíveis no diretório:

[http://www.ic.unicamp.br/helio/imagens\\_coloridas/](http://www.ic.unicamp.br/helio/imagens_coloridas/)

### 4.4. Leitura das Amostras

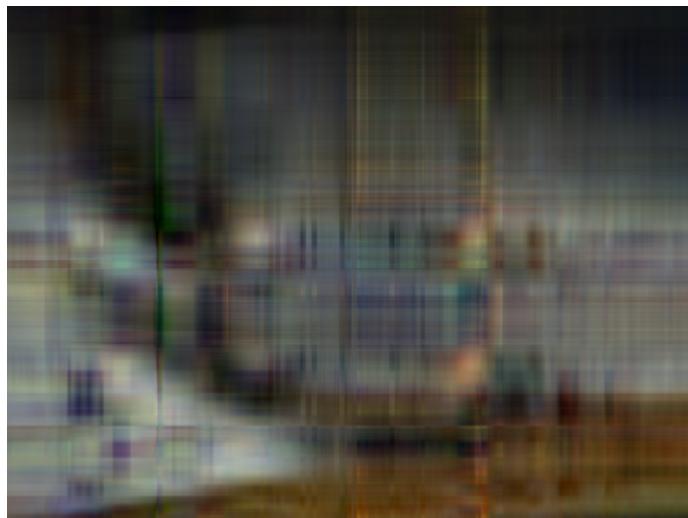
1. Das amostras de imagens utilizadas na implementação do projeto:

- monalisa.png
- baboon.png
- peppers.png
- watch.png

No anexo do projeto, seguem todas as imagens com alterações feitas, em todas as imagens com compressão de **k=1** até **k=250**.



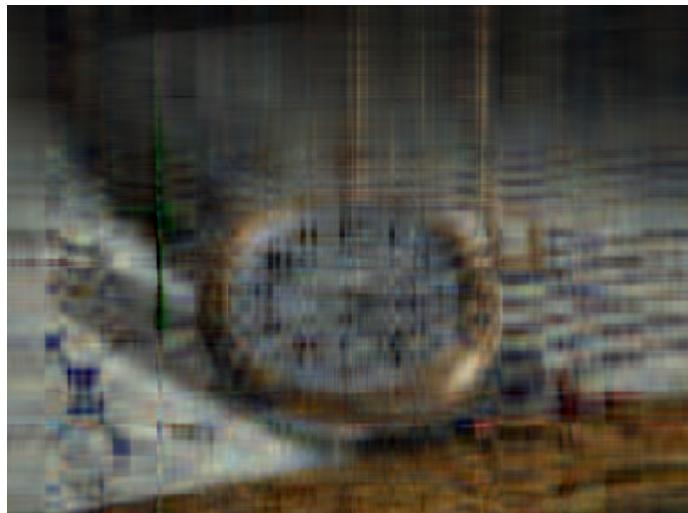
**Figure 4. Imagem Original Watch**



**Figure 6. k=5**



**Figure 5. Watch com k=1**



**Figure 7. k=10**



**Figure 8. k=20**



**Figure 10. k=45**



**Figure 9. k=35**



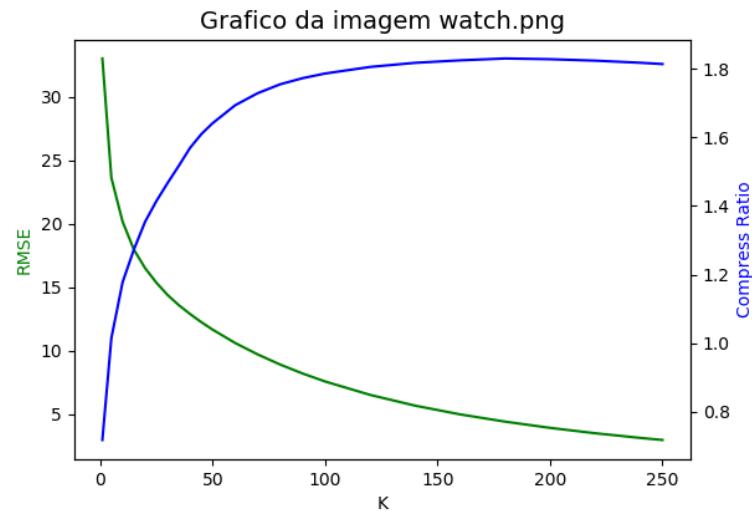
**Figure 11. k=50**



**Figure 12. k=160**



**Figure 13. k=250**



**Figure 14. Gráfico Watch**

## 5. Resultados

- O resultado de maior impacto visual, deixando o ruído mais visível são quando K assume os valores 1, 5, 10, 15, 20, 25, 30, 40, 45, 50, até o 60 é facilmente notável o borramento na imagem;
- A partir da compressão de K = 70 os níveis de ruídos começam a diminuir, deixando a imagem mais nítida;
- Os melhores resultados assumem quando K começa a assumir valores de K = 150 até K = 250, onde a imagem passa a ficar bem mais nítida, chegando o mais próximo da original;

## 6. Dificuldades Encontradas

Neste trabalho não houve dificuldades grandes, pois ao realizar pesquisas vi que existe um método para SVD já implementado na biblioteca numpy, método esse que me ajudou muito no meu problema. Utilizei o seguinte trecho de código:

```
U, S, V = np.linalg.svd(image[:, :, k])
return U, S, V
```

## 7. Conclusão

- **/input.**
  - **baboon.png**
  - **monalisa.png**
  - **peppers.png**
  - **watch.png**
- **/output:** A nomenclatura do arquivo salvo segue pelo **Nome do arquivo + compress + quantidade de k na imagem.**
  - **/baboon**
    - \* baboon\_compress\_k1
    - \* baboon\_compress\_k20
    - \* baboon\_compress\_k250
  - **/monalisa**
    - \* monalisa\_compress\_k15
    - \* monalisa\_compress\_k30
    - \* monalisa\_compress\_k50
  - **/peppers**
    - \* peppers\_compress\_k35
    - \* peppers\_compress\_k70
    - \* peppers\_compress\_k100
  - **/watch**
    - \* watch\_compress\_k180
    - \* watch\_compress\_k200
    - \* watch\_compress\_k220
- **/graficos:**
  - Gráficos das 4 imagens bases utilizadas, onde exibo e cruzo informações do RMSE e Compress Ratio, referente a K.
- Relatório Técnico em PDF;
- script trabalho5.py
- O arquivo completo, contendo todos os arquivos, imagens processadas, script e relatórios compactado na extensão .zip, com nome **trabalho5.zip**.

## 8. Referências

[1] <https://docs.scipy.org/doc/numpy/reference/generated/numpy.linalg.svd.html>