

MC920: Introdução ao Processamento de Imagem Digital

Trabalho 5

Luciano Zago - 182835

Universidade Estadual de Campinas — 21 de Novembro de 2019

Introdução

O objetivo do trabalho é utilizar a técnica de Análise dos Componentes Principais (PCA) para comprimir imagens com um certo grau de perda. Essa técnica utiliza da Decomposição em Valores Singulares (SVD) em cada canal da imagem de entrada para separar a imagem em seus componentes principais. Selecionando um número reduzido k de componentes, permite-se comprimir a imagem final.

1 Especificação do Problema

Utilizaremos uma imagem colorida RGB de entrada e retornaremos uma imagem comprimida com um certo grau de perda referente ao valor do parâmetro k .

Será utilizado as técnicas de Análise dos Componentes Principais (PCA) e Decomposição em Valores Singulares (SVD).

O algoritmo utilizado será apresentado na seção 2.

As imagens de entrada utilizadas nesse trabalho estão representadas na Figura 1.

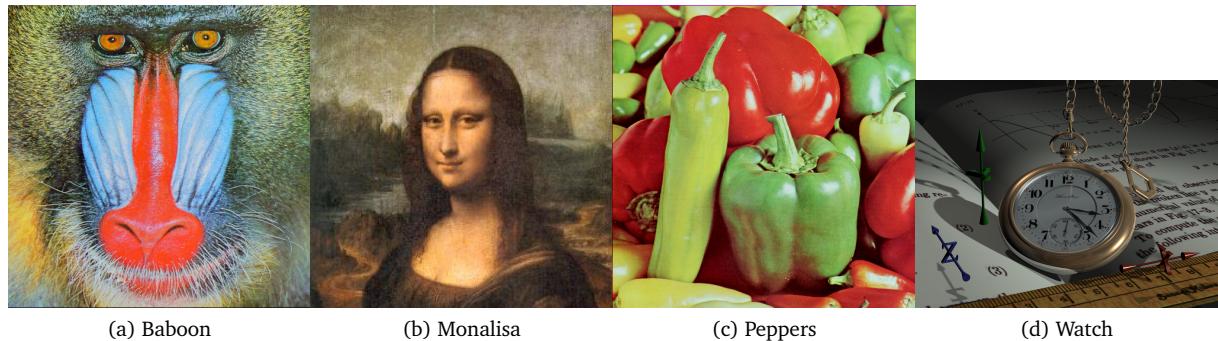


Figura 1: Imagens de entrada

2 Implementação

O programa em Python deve ser usado da seguinte forma:

```
./compression.py image k output
```

Argumentos:

- k representa o número de componentes principais a serem representados na imagem de saída

- `image` e `output` representam respectivamente os arquivos PNG de entrada e saída

Para rodar todas as opções de `k:(1 5 10 20 30 40 50)` em todas as imagens de exemplo da pasta `input`, existe o script `./run.sh > output.txt 2>&1` que facilita o processo, e que retorna o tempo de execução.

O algoritmo implementado para a compressão usando SVD foi o seguinte:

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  $g$ 
```

As avaliações da compressão foram implementadas da seguinte forma:

- Taxa de compressão: `getsize(file_out) / getsize(file_in)`
sendo `file_in` e `file_out` respectivamente os arquivos PNG de entrada e saída
- RMSE: `np.sqrt(np.mean((img_in - img_out)**2))`
sendo `img_in` e `img_out` respectivamente as imagens em forma matricial de entrada e saída

Além disso, foi utilizada o nível máximo de compressão do PNG para reduzir o tamanho do arquivo de saída.

3 Resultados

Os resultados utilizando as diferentes imagens de entrada com diferentes valores de compressão k estão representados na Figura 2.

Analizando-se os resultados obtidos da Tabela 1, percebe-se que a qualidade e compressões obtidas para certo k variam consideravelmente dependendo da imagem de entrada.

Utiliza-se a taxa de compressão para avaliar quantitativamente a compressão e o erro RMSE para avaliá-la qualitativamente. Escolhendo um valor de compressão máximo de 0.8 e um erro RMSE máximo de 20, podemos garantir pouca perda de qualidade na imagem com um nível de compressão de pelo menos 20%. Porém, esses valores só ocorreram com a imagem Monalisa usando $k=10$ e Peppers usando $k=30$, que estão representados na Figura 2.

Outra informação relevante é que valores de k muito altos podem, ao invés de comprimir, aumentar o tamanho da imagem final. Caso que ocorreu na imagem Monalisa para $k \geq 40$ e na imagem Watch para $k \geq 20$.

4 Conclusão

Nesse trabalho foi possível utilizar a técnica de Análise dos Componentes Principais (PCA) para comprimir imagens com um certo grau de perda. Verificamos assim, que a essa técnica foi eficiente para compressão

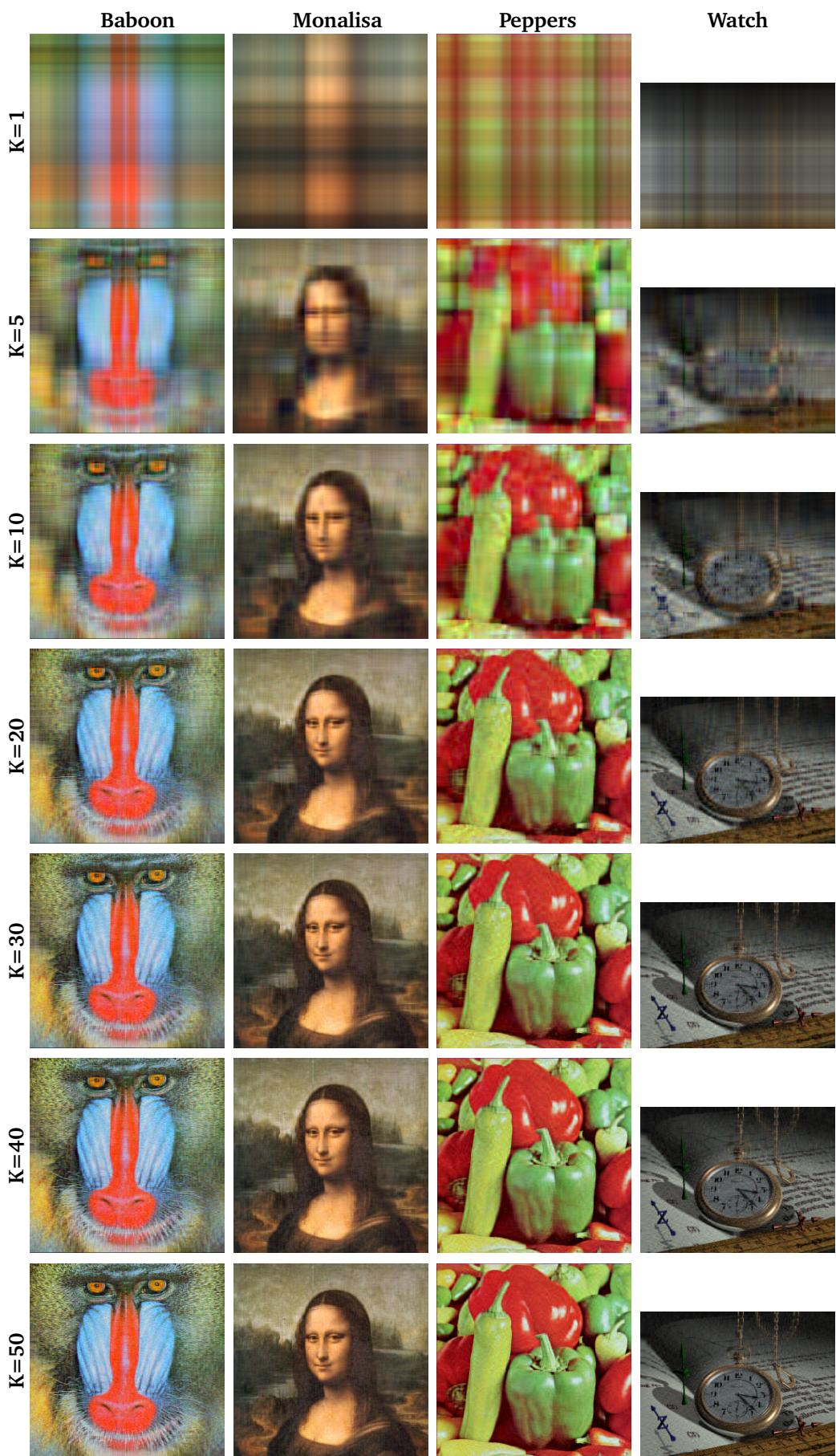


Figura 2: Resultados com as imagens de entrada padrão para valores de K:(1 5 10 20 30 40 50)

k	Baboon		Monalisa		Peppers		Watch	
	Compressão	RMSE	Compressão	RMSE	Compressão	RMSE	Compressão	RMSE
1	0.35	43.52	0.49	33.94	0.34	51.83	0.51	33.02
5	0.52	31.01	0.67	15.25	0.47	29.46	0.75	23.61
10	0.62	27.88	0.78	10.50	0.55	21.47	0.91	20.16
20	0.74	24.89	0.90	7.51	0.64	14.57	1.09	16.50
30	0.80	22.77	0.96	6.13	0.70	11.39	1.23	14.38
40	0.83	20.99	1.01	5.23	0.85	9.44	1.36	12.88
50	0.86	19.43	1.04	4.53	0.87	8.12	1.45	11.66

Tabela 1: Taxa de compressão e RMSE

das imagens Monalisa usando $k=10$ e Peppers usando $k=30$, considerando ótimas as medidas da taxa de compressão máxima como 0.8 e o erro RMSE máximo como 20.

Além disso, valores de k muito altos podem aumentar o tamanho da imagem final, que é um comportamento indesejado para esse trabalho. Esse caso ocorreu na imagem Monalisa para $k >= 40$ e na imagem Watch para $k >= 20$.