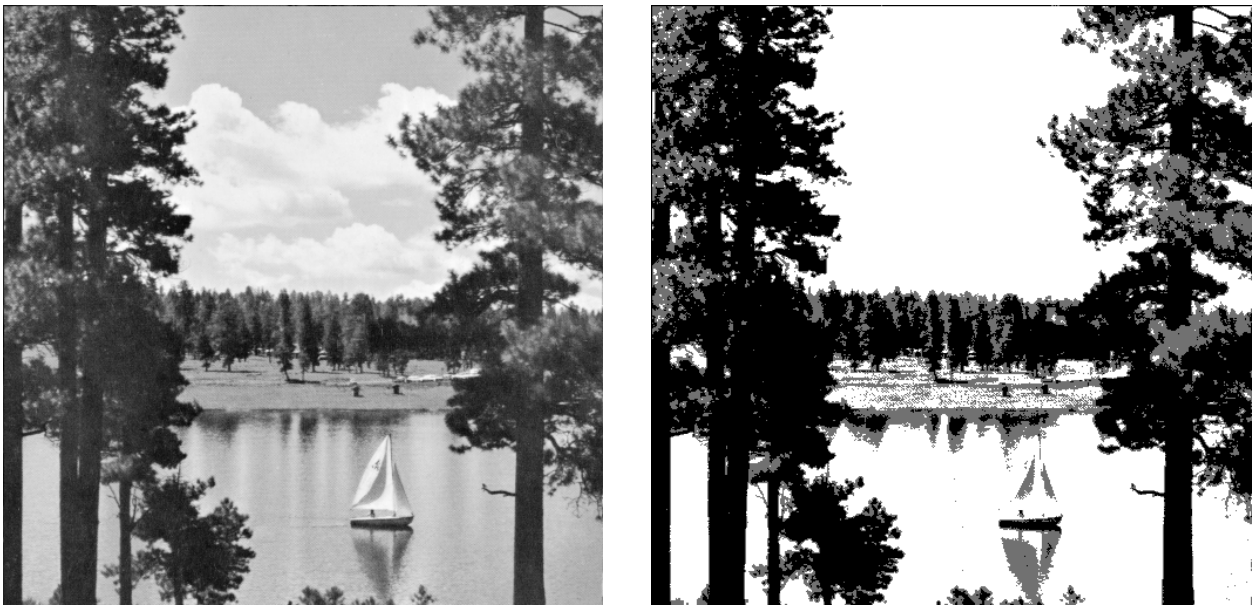
Clusterização de Imagens

1 Considerações Iniciais

Neste trabalho, você deverá implementar em Linguagem C o algoritmo k-means para realizar a clusterização de imagens em tons de cinza, manipulando diretamente os pixels sem o uso de bibliotecas externas. O algoritmo será utilizado para dividir os pixels em diferentes k grupos com base na similaridade de intensidade do pixel. A Figura 1 representa um exemplo de imagem resultante do algoritmo k-means com o valor de k = 3.



(a) Imagem de entrada

(b) Imagem clusterizada

Figura 1: Comparação entre a imagem original e a imagem segmentada utilizando o algoritmo k-means com $k = 3$. Em (a), apresenta-se a imagem original, enquanto em (b) é exibida a imagem clusterizada, onde os pixels foram agrupados em três clusters de acordo com suas características.

Sugestão para iniciar uma pesquisa na área, consulte:

Bishop, C. M. Pattern Recognition and Machine Learning. Springer, 2006.

Gonzalez, Rafael C.; Woods, Richard E. Processamento digital de imagens. 3a Edição. Editora Pearson, 2010.

2 Introdução Teórica

O algoritmo k-means é uma técnica de aprendizado não supervisionado amplamente utilizada em processamento de imagens e visão computacional. Ele funciona dividindo um conjunto de dados em k clusters, minimizando a distância entre os elementos de cada cluster e seu centróide. O algoritmo pode ser descrito pelos seguintes passos:

1. Inicialização dos centróides: Escolhem-se k centróides iniciais, que podem ser selecionados aleatoriamente ou de maneira sistemática.

2. Atribuição de clusters: Cada ponto (neste caso, cada pixel da imagem) é atribuído ao cluster cujo centróide esteja mais próximo, de acordo com uma métrica de distância, como a distância euclidiana.
3. Atualização dos centróides: Para cada cluster, recalcula-se o centróide como a média de todos os pontos atribuídos ao cluster.
4. Convergência: Repete-se os passos 2 e 3 até que os centróides não mudem significativamente ou até atingir um número máximo de iterações.

3 Objetivo

Sua missão será implementar e conduzir experimentos de clusterização com o algoritmo k-means em uma base de dados fornecida. Para isso, você deverá testar diferentes valores de k e avaliar os resultados obtidos. Você pode utilizar o padrão ouro das segmentações fornecidas no dataset e avaliar quantitativamente os resultados obtidos com a métrica DICE (Ref.: `https://cs.adelaide.edu.au/~carneiro/isbi14_challenge/evaluation.html`). Suas conclusões devem ser apresentadas ao professor. O valor k deve poder ser ajustado pelo usuário de sua aplicação.

Observações:

- Os experimentos deverão ser executados utilizando a base de imagens no formato PGM, que está disponível no diretório do projeto no DropBox.
- Você é livre para escolher a métrica de distância que preferir. Pesquise sobre as métricas que podem ser utilizadas para computar a distância entre os valores dos pixels e os centróides.
- Avalie uma faixa de valores para k e apresente suas conclusões. Qual valor de k apresenta o melhor desempenho? É possível estabelecer alguma relação entre o valor de k e o histograma da imagem?
- A equipe deve apresentar, juntamente com códigos, os resultados da clusterização e suas principais conclusões.

Nas seções seguintes deste documento estão descritas algumas informações necessárias para o correto desenvolvimento deste trabalho.

4 Formato PGM

Neste trabalho, deve-se utilizar o formato PGM (*portable graymap*) para armazenar imagens em arquivos. Este formato tem duas variações, uma binária (o PGM "normal" ou *raw*) e outra textual (o PGM ASCII ou *plain*). Em ambos os casos, o arquivo deve conter um cabeçalho e a matriz correspondente à imagem. O exemplo a seguir mostra um arquivo PGM textual:

```
P2
5 4
16
9 4 5 0 8
10 3 2 1 7
9 1 6 3 15
1 16 9 12 7
```

A primeira linha do arquivo contém obrigatoriamente uma palavra-chave, que deve ser "P2" no caso de um arquivo PGM textual e "P5" no caso de um arquivo PGM binário. A segunda linha contém dois números inteiros que indicam o número de colunas e o número de linhas da matriz, respectivamente. A terceira linha contém um número inteiro positivo *maxval*, que deve ser igual ao maior elemento da matriz. Na definição do formato PGM, *maxval* não pode ser maior que 65535. Para fins deste trabalho, entretanto, *maxval* é no máximo 255. Os demais números do arquivo são os elementos de uma matriz de inteiros com os tons de cinza de cada ponto da imagem. Cada tom de cinza é um número entre 0 e *maxval*, com 0 indicando "preto" e *maxval* indicando "branco".

O formato PGM também permite colocar comentários. Todo o texto que vai desde um caractere '#' até (e inclusive) o próximo fim de linha é um comentário e deve ser ignorado. Este é um exemplo de arquivo PGM textual com um comentário:

```
P2
# feep.pgm
24 7
15
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 3 3 3 0 0 7 7 7 7 0 0 11 11 11 0 0 15 15 15 0 0 0
0 0 0 3 0 0 0 7 0 0 0 0 0 11 0 0 0 0 0 15 0 0 0 0
0 0 0 3 0 0 0 7 7 7 0 0 0 11 0 0 0 0 0 15 15 15 0 0
0 0 0 3 0 0 0 7 0 0 0 0 0 11 0 0 0 0 0 15 0 0 0 0
0 0 3 3 3 0 0 7 0 0 0 0 0 11 11 11 0 0 15 15 15 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

O formato PGM binário tem cabeçalho análogo ao do PGM textual, usando a palavra chave "P5" em vez da "P2". O que muda é o modo como é armazenada a matriz de tons de cinza. No formato PGM textual, essa matriz é guardada como uma sequência de caracteres ASCII contendo as representações decimais das tonalidades de cinza. No formato PGM binário, a matriz é guardada como uma sequência de *bytes*, sendo o valor de cada *byte* (de 0 a 255) uma tonalidade de cinza. O número de *bytes* da sequência é exatamente igual ao número de elementos da matriz[1]. Este é um exemplo de arquivo PGM binário:

```
P5
# feep.pgm
24 7
15
```

... (sequência de 24 x 7 bytes com as tonalidades de cinza)

Existem dois outros formatos de arquivo muito semelhantes ao PGM: o PBM (*portable bitmap*), para imagens monocromáticas (só preto e branco, sem tons de cinza), e o PPM (*portable pixmap*), para imagens coloridas. No primeiro, os elementos da matriz podem assumir apenas os valores 0 e 1. No segundo, os elementos da matriz são triplas de números inteiros positivos correspondentes às intensidades das cores vermelha, verde e azul nos pontos da imagem. Quem quiser saber mais detalhes sobre esses formatos, visite as seguintes páginas:

- http://en.wikipedia.org/wiki/Portable pixmap
- http://netpbm.sourceforge.net/doc/pbm.html
- http://netpbm.sourceforge.net/doc/pgm.html
- http://netpbm.sourceforge.net/doc/ppm.html

[1] Na verdade, essa descrição se aplica apenas a arquivos PGM com $maxval \leq 255$, que são considerados neste trabalho. No caso $256 \leq maxval \leq 65535$, a matriz é guardada como um sequência de pares de *bytes* e o número de *bytes* da sequência é o dobro do número de elementos da matriz.

5 Sobre a organização dos códigos fontes

ATENÇÃO:

1. A nota máxima deste trabalho é 10.0. A distribuição dos pontos segue como definido a seguir:

   - **Entrada e saída com arquivos texto ou binário:** 1.0
   - **Implementação do algoritmo k-means:** 5.0
   - **Experimentos** : 2.0
   - **Avaliação dos resultados** : 2.0

   Critérios para cada tópico acima:

   - Organização dos códigos: 20%
   - Domínio da solução adotada: 50%
   - Análise Técnica: 30%

2. Todos os arquivos fontes do seu programa devem conter um cabeçalho como o seguinte:

```
/*************************************************************/
/* Aluno: Joao de Souza                                      */
/* Matricula: 12345                                          */
/* Avaliacao 04: Trabalho Final                              */
/* 04.505.23 - 2024.2 - Prof. Daniel Ferreira                */
/* Compilador:...(DevC++ ou gcc) versao ...                  */
/*************************************************************/
```

   A organização do programa deve obedecer a construção de arquivos de cabeçalhos e permitir a compilação separada. Deve-se utilizar processo de *linkedição*. Procure dividir seus códigos em arquivos sempre que necessário. Enfim, procure construir bibliotecas genéricas que proporcione posterior reuso.

3. O trabalho deverá ser realizado em **equipes com no máximo 4 (quatro) participantes**. **Nota individual.** Sugiro, portanto, que marquem seminários entre a equipe para que todos possam atingir o mesmo nível de conhecimento.

4. Códigos fontes copiados (com ou sem eventuais disfarces) receberão nota **ZERO**.

5. É muito importante que seu programa tenha comentários e esteja bem indentado, ou seja, digitado de maneira a ressaltar a estrutura de subordinação dos comandos dos programas (conforme estudado em aula).

6. O professor poderá executar o programa tantas vezes quantas forem necessárias para testar os vários casos possíveis para as entradas.

7. Os códigos devem ser disponibilizados ao professor por meio de um link no GitHub. Deve ser construído um README com detalhes para a execução.

8. No dia da apresentação é **indispensável** o comparecimento de toda a equipe. Faltas devem ser devidamente justificadas para segunda-chamada.

9. Datas para as apresentações:

   - **Dia 1: 12 de Fevereiro de 2025 (Quarta-Feira) a partir das 07h40.**
   - **Dia 2: 17 de Fevereiro de 2025 (Segunda-Feira) a partir das 07h40**.

   A ordem de apresentação será definida posteriormente por sorteio.