Trabalho 1 – Processamento Digital de Imagens e Sinais – DCC066 – 2018/1
Departamento de Ciência da Computação – UFJF
18 de março de 2018

Prof. Marcelo Bernardes Vieira (marcelo.bernardes@ufjf.edu.br)

Espaços de Cor

Instruções:

- O valor deste trabalho é de **20 pontos**. O resultado deverá ser apresentado por todos os elementos do grupo até o dia **17 de abril de 2018**, impreterivelmente. Após esta data o grupo receberá nota nula.
- Na apresentação, os grupos deverão demonstrar que suas soluções para cada um dos itens desta especificação funcionam com diversas imagens.
- As especificações abaixo **não têm informações completas para sua implementação**. Os grupos **devem** procurar o professor para tirar dúvidas e entender detalhes do que foi pedido.
- Este trabalho tem a carga de 30 homem·hora: em um grupo de 2 pessoas, cada um deve dedicar 15 horas para finalizar o trabalho.
- Todos os grupos não têm permissão de possuir ou manter consigo o código ou os resultados de outro grupo, incluindo trabalhos antigos. É aconselhável que os grupos mantenham sigilo sobre suas soluções.
- Qualquer fraude, ou mera tentativa de fraude, será punida com **anulação DE TODAS AS NOTAS** de todos os trabalhos de todos os grupos envolvidos. O fato também será relatado à direção do ICE juntamente com uma solicitação de suspensão dos alunos fraudadores.

Projete e implemente um sistema para processamento básico de imagens usando Python:

- Use a biblioteca *Python Imaging Library*-PIL ou PILLOW (na versão 3 do Python), a biblioteca NumPy, a biblioteca SciPy para facilitar o desenvolvimento. Outras bibliotecas podem ser usadas livremente mas deverão ser relatadas ao professor.
- Opcionalmente, os alunos podem implementar uma funcionalidade em C++. Porém, a funcionalidade deverá ser exportada e utilizando dentro do ambiente Python.
- Implemente **todas** as funcionalidades listadas nesta especificação.
- Na apresentação, o professor pode solicitar que uma imagem específica seja processada. Portanto, os alunos devem construir um sistema flexível, sem qualquer restrição sobre tamanho da imagem de entrada.
- Dicas: use o PyCharm como editor para ambiente Python; o pacote TKInter e Click facilitam a interface através da linha de comando; a biblioteca Matplotlib pode ser usada para visualizar imagens e gerar histogramas.
-
- **Atenda a todas as especificações a seguir**. Este trabalho faz parte de um projeto maior.

---

1. **Funcionalidades básicas de entrada e saída**:

   - Aprenda como ler imagens .jpg e .png. O leitor deve ser universal, ou seja, qualquer imagem nesses formatos devem ser lidas. Após ler uma imagem, independentemente de seu espaço de cor, converta-a para uma imagem colorida (RGB) com três canais de 8 bits. As imagens .png podem ter um extra canal de opacidade, formando uma imagem RGBA. O canal de opacidade com 8 bits pode ser mantida para futuro processamento à critério do grupo.
   - Implemente um conversor de imagem RGB para imagem em tons de cinza (TC) de 8 bits e vice-versa. **Todas as funcionalidades abaixo, exceto quando diretamente especificado, devem ter versões para imagens coloridas RGB e imagens TC**.
   - Aprenda a salvar imagens RGB e TC nos formatos .jpg e .png.
   - Use algum visualizador básico de alguma biblioteca Python para visualizar suas imagens.

- Implemente uma função que, dadas duas imagens do mesmo tipo e tamanho, calcula o Erro Quadrático Médio (*Mean Squared Error* - MSE) e a Razão Sinal-Ruído (*Signal-to-Noise Ratio* - SNR) entre elas.
- O grupo deve fazer parametrização de cada funcionalidade por linha de comando para facilitar o uso e a avaliação do trabalho.

2. **Transformações de espaço de cor**:

   - Implemente a função de correção gamma.
   - Implemente a função de equalização de histograma.
   - Implemente a função de binarização por limiarização. Faça binarizadores para imagens RGB e para imagens TC. Implemente o algoritmo de Otsu para calcular o limiar automaticamente e dê a opção para o usuário o usar nos binarizadores.

3. **Quantização**:

   - Implemente um quantizador universal. Ou seja, dada uma imagem qualquer, o seu programa deve ser capaz de convertê-la em uma nova imagem cujo o número de bits por canal pode ser livremente definido pelo usuário. Para simplificar, limite até no máximo 16 bits por canal. No caso de redução de bits, utilize os métodos apresentados em sala de aula (ou mais sofisticados) para garantir que contornos de quantização sejam reduzidos a um mínimo.

4. **Desenvolvendo o trabalho**:

   - Antes de começar a codificar, aprenda os conceitos básicos do Python. Há muitos tutoriais na internet. Você pode começar lendo imagens .jpg e .png e criando uma interface simples para visualizá-las. Lembre-se que o grupo deverá fazer uma boa apresentação mostrando a eficácia de seu sistema.
   - Não escreva nenhum código antes de entender completamente cada funcionalidade. Tente fazer cada tópico desta especificação na ordem proposta porque grande parte das funcionalidades depende diretamente ou pode ser implementada a partir das antecessoras.
   - Não deixe de consultar o livro a respeito de como implementar o que foi pedido. Consulte os livros do curso para desenvolver cada etapa, sobretudo o do Gonzalez. Nele vocês encontrarão vários atalhos e exemplos.
   - Aprenda funcionalidades das bibliotecas que podem ser úteis no desenvolvimento do trabalho como: recorte de subimagem, duplicação de imagens, combinação de imagens, redimensionamento de imagens, rotação e espelhamento de imagens, desenho de texto sobre a imagem, etc.