Sessão prática 4 – Restauração de imagem e pseudo-cor

Nota:
As imagens lenina.viff, gauss-kernel-21x1.viff, lenina-blur-float.viff, map-roi.viff, map-roi-4x.viff, avg4x4.viff, interpolacao.viff e retina.viff podem ser descarregadas da página da disciplina (gec.di.uminho.pt/mcc/vpc/imagens/)
As restantes imagens encontram-se na directoria $KHOROS/sampledata/data/images/

4.1 Restauração de imagem

A imagem lenina.viff foi degradada com o kernel gauss-kernel-21x1 através de uma convolução. Visualize estas duas imagens e a imagem degradada lenina-blur-float.viff. Esta degradação de imagem corresponde a uma fotografia tirada em movimento.

Para restaurar a imagem calcule a transformada da imagem degradada e a transformada do filtro (não se esqueça de transformar o filtro numa imagem de 256x256 com a função <Data Manip, Size & Region Operators, Pad>. Divida a transformada da imagem degradada pela transformada do filtro <Arithmetic, Two Operand Arithmetic, Divide>. Aplique a transformada inversa ao resultado e visualize a imagem restaurada. Nota: Neste exercício utilize a opção *Unity Scaling* ao calcular a FFT.

Este exemplo corresponde à situação ideal. No entanto, na prática a imagem a restaurar irá conter ruído aleatório. Para simular este ruído converta a imagem lenina-blur-float.viff para o tipo unsigned byte <Data Manip, Data Convertion, Convert Data Type>. Subtraia as duas imagens para verificar o ruído introduzido <Arithmetic, Two Operand Arithmetic, Subtract>. Aplique agora o processo do parágrafo anterior à imagem com ruído. Tente encontrar uma justificação para o resultado obtido, para tal compare os espectros do kernel, da imagem degradada e da imagem restaurada.

Um dos problemas da filtragem inversa surge na presença de valores pequenos na transformada do filtro. Estes valores pequenos, ao serem utilizados como divisores na filtragem inversa, podem amplificar fortemente o ruído existente na imagem. Daí surge a explicação para os resultados anteriores. Uma solução para este problema baseia-se em desprezar estes valores ao efectuar a divisão, tornando o resultado para esses pontos igual a zero. Esta operação é efectuada pelo filtro <Data Manip, Frequency Filter, InvFilter>. Utilize este filtro como alternativa à divisão e visualize o resultado. Experimente agora a mesma operação, utilizando o filtro de Wiener <Data Manip, Frequency Filter, Wiener>, com o valor de 0.05. Experimente comparar o resultado dos dois filtros para o mesmo parâmetro e visualizar as alterações introduzidas pela modificação do valor do parâmetro.

4.2 Interpolação de imagem

Visualize a imagem map-roi.viff e o respectivo espectro. A imagem map-roi-4x.viff resulta da expansão desta imagem 4 vezes, sem interpolação. Visualize também esta imagem e o respectivo espectro. Efectue uma interpolação por replicação dos pixeis; para tal efectue uma convolução da imagem expandida com o kernel avg4x4; visualize a imagem resultante e o respectivo espectro. Repita a operação com o kernel interpolacao. Efectue a interpolação no domínio das frequências, aplicando um filtro passa-baixo (Butterworth de ordem 6 e raio 0.10) à imagem map-roi-4x.viff. Qual das três alternativas lhe parece produzir melhores resultados?

4.3 Pseudo-cor

Visualize a imagem retina.viff. Utilize o operador <`Khoros 1, Generate Data, Picewise Linear`> para gerar um imagem de 128x20 pixels que será utilizada como referência para as cores. No operador indique um tom máximo de 255, um período em Y de 128, *Y Rise Time* de 128 e *Sampling Frequency* de 1. Converta esta imagem para unsigned byte e utilize a função *Pad* para inserir uma borda branca na imagem. Junte esta imagem à original com o operador <`Data Manip, Size & Region Operator, Inset`>. Visualize a imagem resultante, introduzindo um mapa de cores com a função <`Visualization, Map Display & Manip, Autocolor`>.