Nome: Tiago Figueiredo Gonçalves
Fiz o projeto usando octave, portanto não garanto que tudo funcione em matlab! Para todas convoluções envolvendo algum arquivo (áudio, vídeo, imagem) usei “valid” que tira a parte “zero-padded”.

Parte 1. Sinais
1.1 Projeto de Filtro, passa alta ws = 0.6pi, wp = 0.75pi, As = 50dB.

Arquivos: highpass1.m, highpass2.m, ideal_lp.m

Primeiro, olha-se a tabela das janelas para saber qual janela escolher para a altura de corte desejada. A janela de Hamming tem altura de pico lateral -53db portanto escolhi ela. Após isso, basta achar qual tamanho da janela usar de acordo com a largura de transição desejada e a janela. Feito isso, a janela está definida e basta sobrepor ela a um filtro passa-alta ideal que pode ser obtido por um filtro passa-tudo menos um filtro passa-baixa ideal. O resultado desse projeto está no arquivo highpass1.m e o arquivo highpass2.m contém o resultado do mesmo processo escolher uma janela de Kaiser no lugar da de Hamming.

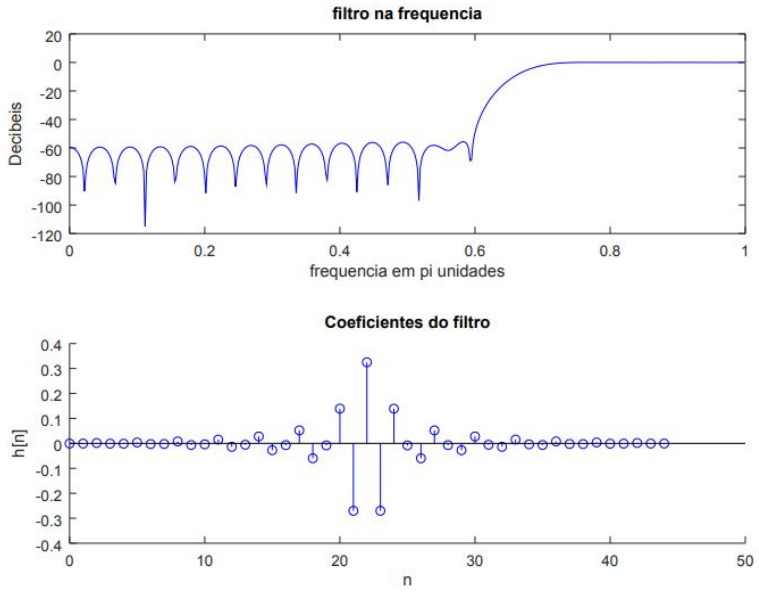


Imagem usando janela de Hamming

1.2 Reverb com filtro IIR

Arquivos: reverb.m, sp.m, sp04reverb1x.wav

Para um filtro H(z) = 1 / (1 - a * z^-D), é equivalente a fazer h(n) = x(n) + a * h(n - D). Esse tipo de filtro está implementado no arquivo reverb.m. Para a > 0, esse filtro funciona como ecos pequenos com decaimento exponencial, como se o ouvinte estivesse numa sala pequena e o som estivesse preso na sala. O resultado “sp04reverb1x.wav” reflete isso, o eco é aumentado nesse caso. Para a < 0, o filtro ainda funciona como ecos pequenos com decaimento exponencial, porém a cada eco a onda chega refletida além de decaída, possibilitando esse filtro a ser usado para cancelamento de eco. O resultado com a = -0.25 tem apenas parte do eco retirado, com a = -0.5 tem o eco retirado mais ainda porém com a = -0.85 o decaimento é tão pequeno que acaba fazendo mais eco no áudio.

Parte 2 Processamento de Imagens

2.1 Remoção de ringing

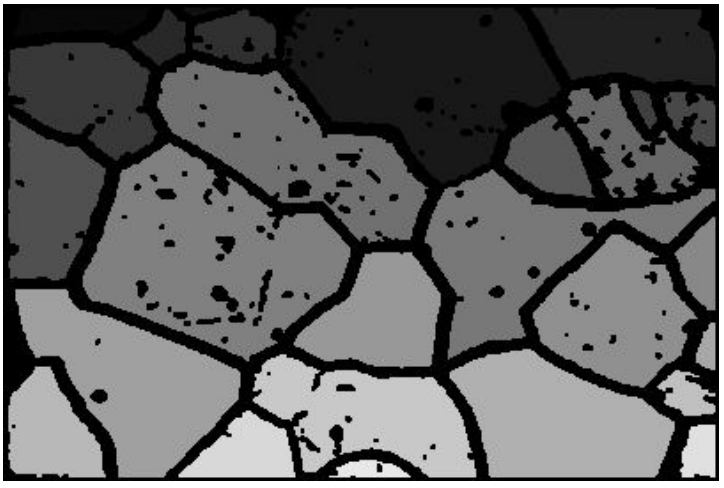
Arquivos: lena.m, lena_mean3.bmp, lena_mean5.bmp, lena_iir.bmp, lena_gauss.bmp, gaussian2d.m

Inicialmente testei borrar a imagem com um filtro de média. Quanto mais borrada a imagem, menos perceptível fica os anéis porém fica muito mais difícil de distinguir elementos na imagem. Tendo testado isso, a procura de um jeito melhor de borrar a imagem procurei usar um filtro gaussiano sem muito sucesso. Tive então a ideia de usar filtro IIR nas direções X e Y para remover os anéis, esse filtro foi bom em tirar os anéis porém isso só acontece quando o decaimento escolhido é pequeno e a imagem fica muito borrada. Tentei ainda usar detecção de borda com filtro de sobel e tentar retirar as bordas finas mas não consegui fazer a imagem ficar sem grandes distorções.

2.2 Contagem de regiões

Arquivos: texturecounter.m, texture.bmp

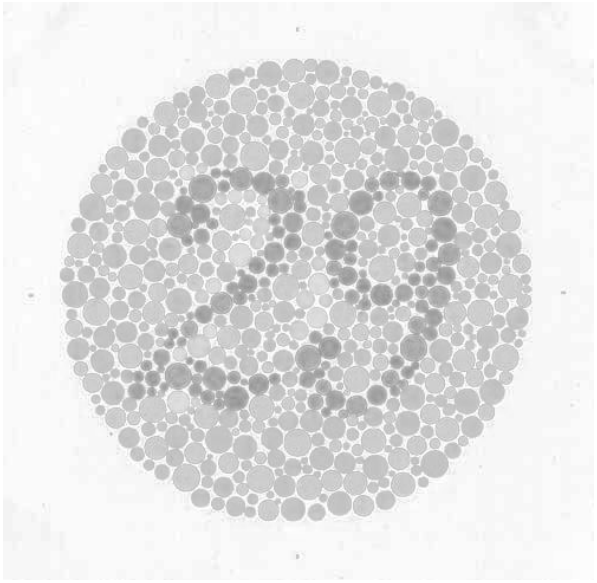
O algoritmo que usei tem 2 etapas: detecção de bordas e componentes conexas em regiões dentro das bordas. Para pegar as bordas, usei filtros Sobel e peguei a raiz quadrada da soma dos quadrados das bordas horizontais e verticais. Depois, usei “disjoint set union” para achar as componentes conexas e apenas componentes conexas com pelo menos uma certa quantidade de pixels foi aceita. Essa quantidade de pixels é ajustável mas poderia ser colocada como padrão alguma porcentagem da imagem para não ter problemas com imagens de alta resolução. Foram achadas 29 regiões.



2.3 Daltonismo

Arquivos: dalton.m, dalton2.bmp, dalton3.bmp

Tentei sem muito sucesso compartilhar um pouco entre as cores (por exemplo, vermelho fica vermelho com um pouco de verde), para destacar as diferentes cores para os diferentes tipos de daltonismos. Isso me levou a idéia de extrair informação de cada cor base da imagem e ver que tons a cor do número usa. Usei um algoritmo de corte total de acordo com os valores RGB (imagem dalton3.bmp), mudando os valores de corte cheguei à conclusão que muito R e pouco G é o que procura no número dessa imagem. O número tem muito vermelho e pouco verde, portanto fiz um filtro que a cor passa completamente se existe muito vermelho e pouco verde e caso contrário fica claro de acordo com a ausência de vermelho/presença de verde (imagem dalton2.bmp). A imagem em tons de cinza “gray” foi obtida e a expressão “image2(i, j) = 255 - gray(i, j)*((double(R(i, j)) / 200) * ((255 - double(G(i, j))) / 200));” é o filtro descrito na linha anterior.



Parte 3 Processamento de Vídeo

3.1 Destacar os carros passando

Arquivos: videoprocessing.m, video2.avi

Como a câmera fica estática, tentei primeiramente fazer um vídeo que destaca a diferença absoluta entre os pixels de dois frames seguidos. Apenas isso já conseguiu um resultado muito bom, porém ainda tinha uma parte das folhas que estava ficando com tom de roxo escuro. Para minimizar isso, escureci um pouco a imagem e eu não poderia imaginar resultado melhor.

Silhueta dos carros (com cores invertidas)

Parte 4 Processamento de Áudio

4.1 Retirar o som do carro passando

Arquivos: car.m, car2.wav

Analisei o espectrograma do áudio para identificar aproximadamente a frequência da fala. Usando o espectrograma e ouvindo os áudios, usei filtros FIR passa baixa e passa alta para identificar qual o limite da faixa de frequência o carro usa. O carro usa tanto frequências baixas e frequências altas, provavelmente devido ao efeito doppler. Sabendo essas faixas de frequência, usei um filtro passa faixa. O carro realmente saiu do áudio mas não consegui tirar mais o ruído que ficou (fiquei mudando o tamanho da janela e de coeficientes do filtro).

4.2 Retirada de ruído usando filtros FIR/IIR sem conhecimento prévio do ruído

Arquivos: som.m, teste_de_som2.wav

Usei um filtro FIR passa-alta para procurar a frequência do ruído. Achado a janela de frequência aproximadamente [0.1, 0.2], usei um filtro corta-faixa e ajustei o tamanho da janela. Além disso, testei usar o filtro de reverberação com esperanças de a reverberação cancelar o próprio ruído porém não achei nenhuma janela de reverb que teve esse efeito.