UNIVERSIDADE FEDERAL DE MINAS GERAIS — UFMG
INSTITUTO DE CIÊNCIAS EXATAS — ICEX
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO — DCC

Disciplina: DCC884/DCC043 — Visão Computacional
Professor: Mario Fernando Montenegro Campos (`mario@dcc.ufmg.br`)
Período: 1º semestre / 2011
Aulas: Terças e Quintas, das 9:25 às 11:05, na sala 2014 do ICEx
Página: `http://www.verlab.dcc.ufmg.br/cursos/verlab/visao/2011-1/index`

2ª Lista de Exercícios

Data da entrega: 17 de maio de 2011
Valor: 7 pontos

A lista é individual e as datas de entrega são fixas, devendo o trabalho ser entregue no início da aula. Para cada dia de atraso na entrega será descontado 0,5 ponto do valor da lista.

Não há formato específico para apresentar as respostas e resultados. Porém, clareza na redação é um requisito essencial. Embora o resultado possa ser manuscrito, lembre-se de que "o que não se pode ler não se pode avaliar". Assim, sugere-se que o trabalho seja digitado, podendo o desenvolvimento matemático ser manuscrito.

O Matlab ou Scilab são recomendados para o desenvolvimento das aplicações, pois permitem o desenvolvimento rápido de aplicativos e já incorporam várias rotinas matemáticas e de manipulação de imagens. O Scilab é gratuito e pode ser baixado da página `http://www.scilab.org/`. Você também pode usar outras linguagens de programação, mas nesse caso lembre-se de incluir as instruções de compilação dos programas e eventuais arquivos auxiliares, como "makefiles" ou arquivos de projeto.

Em todos os casos, não se exige nenhuma interface gráfica: os programas podem ler os dados de entrada por argumentos de linha de comando ou por digitação pelo usuário, e podem gravar a saída em arquivos no disco.

Para arquivos gráficos, recomenda-se o uso dos formatos PNG ou JPEG. Nesse sentido o Matlab e o Scilab ajudam, pois ambos disponibilizam rotinas de leitura e gravação para esses formatos. Para os entusiastas de C e C++, a biblioteca FreeImage (`http://freeimage.sourceforge.net/`) é uma alternativa gratuita e multi-plataforma. Para a visualização dos arquivos, os aplicativos IrfanView (Windows) e xv (Linux) são boas ferramentas e também são gratuitos.

Em qualquer situação, os programas devem ser entregues com documentação de uso ou devem ser auto-explicativos.

1 Exercícios Teóricos

1. Descreva o que é ruído em uma imagem.

2. Faça uma pesquisa sobre as fontes de ruídos no processo de formação das imagens. Liste e explique cada uma das fontes encontradas.

3. O que é uma filtragem linear e uma filtragem não linear? Por que a filtro da mediana é um exemplo de filtro não linear?

4. Procure saber e explique como é o processo de criação de uma máscara para um filtro passa-baixas utilizando o método dos mínimos quadrados no domínio espacial dada uma especificação no domínio da freqüência.

5. Descreva um algoritmo para detectar as bordas de uma imagem e sua localização. Explique o que ocorre com a detecção de bordas e sua localização em imagens com muito ruído?

2 Exercícios Práticos

1. Implemente uma convolução em Matlab que receba uma imagem e um *kernel* (quadrado) e produza como saída uma nova imagem resultante da convolução desse *kernel* com a imagem de entrada. Utilze o seu códico para cada uma dos ítens abaixo. Uma das imagens é a Lena (512x512), codificada em TIFF da imagem original B&W (*e.g.* artigo de Burt & Adelson). A outra imagem a ser utilizada pode ser de sua escolha.

   a) Utilizando um filtro *boxcar* 7x7, filtre a imagem. Esteja atento em como lidar com as bordas da imagem. Se sua imagem original for N×N e você quiser uma imagem de saída também N×N, será necessário pensar cuidadosamente o que fazer com as bordas. Caso não consiga ver uma saída razoável, consulte o help da função *conv2* do Matlab e leia a respeito de bordas.

   b) Subtraia a imagem filtrada da imagem original e observe o que"restou". Como você descreveria o resultado dessa subtração? (Observação: valores nessas imagens podem ser negativos – utilize *imagesc* ou desenvolva seu próprio método para exibir imagens com valor real.

   c) Crie seu próprio *kernel* 7×7 que se assemelhe a um "blur" Gaussiano. (Não se esqueça de normalizar o seu filtro para que a soma seja um). Filtre as imagens utilizando o seu *kernel*.

   d) Subtraia o "blur" Gaussiano da imagem original. Como o resultado dessa subtração se compara com a imagem obtida no passo 1b? O que acontece quando varia o sigma do filtro Gaussiano?

   e) Agora que escreveu o próprio código, você pode apreciar as rotinas do Matlab. Repita os passos anteriores utilizando as rotinas conv2 ou filter2 do Matlab. Qual escolher dependerá de como você prefere fazer as coisas. Você obteve os mesmos resultados? Foram próximos? Qual a diferença?

2. Na OpenCV estão presentes dois extratores de características. Um baseado no trabalho de Shi e Tomasi (cvGoodFeaturesToTrack) e outro no algoritmo SURF (cvExtractSURF). Explique o princípio de cada um dos métodos explicitando suas vantagens e desvantagens. Implemente um rastreador simples utilizando como extrator de caracaterísticas o SURF (há um exemplo do uso das funções de extração e comparação de características usando SURF no diretório da OpenCV: samples/c/find_obj.c)

3 Exercícios de Pesquisa

1. Pesquise sobre os algoritmos *Seeded Up Robust Features* (SURF) e *Scale-Invariant Feature Transform* (SIFT). Considere agora a saída que um dispositivo RGB-D fornece (intensidade nos três planos de cor + informação de profundidade). Proponha um descritor que tome proveito das informações de profundidade e que seja mais preciso e apresente menos falsos positivos e negativos do que o SIFT e o SURF.