Disciplina: DCC884 — Visão Computacional
Professor: Mario Fernando Montenegro Campos (mario@dcc.ufmg.br)
Monitor: Wagner Ferreira de Barros (wbarros@dcc.ufmg.br)
Período: 1º semestre / 2009
Aulas: Terças e quintas, das 09:25 às 11:05, na sala 2029 do ICEx
Página: <http://www.verlab.dcc.ufmg.br/cursos/visao/2009-1/index>

2ª Lista de Exercícios

Data da entrega: 30/abril/2009

A lista é individual e as datas de entrega são fixas, devendo o trabalho ser entregue no início da aula. Cada lista vale 5 pontos na nota final. Para cada dia de atraso na entrega será descontado 1 ponto do valor da lista. Esta lista de exercícios é composta de duas partes: a primeira de exercícios teóricos e a segunda com implementações práticas. Não há formato específico para apresentar as respostas e resultados.

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 OpenCV (<http://sourceforge.net/projects/opencvlibrary/>) 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. Explique a diferença entre uma imagem de profundidade e uma imagem de intensidade. Uma imagem de profundidade poderia ser adquirida usando somente câmeras de intensidade, ou seja, sem luz laser ou algo parecido? Justifique.
2. O que é o ruído? Como podemos quantificá-lo em uma imagem? Como o ruído pode afetar as computações em Visão Computacional?
3. O que é um filtro linear? O filtro da mediana é linear? Prove sua resposta.
4. O que é um filtro separável? Quais são as vantagens da separabilidade? O filtro da média é separável? Por quê?

2 Exercícios Práticos

1. Qualquer filtro de ruídos inevitavelmente destrói parte da informação de interesse. Por causa disso, os filtros devem ser usados de maneira criteriosa: o uso de um filtro muito agressivo degrada muito a imagem original, enquanto filtros “leves” tendem a preservar muito ruído.

Em paralelo, um dos aspectos importantes do estudo de filtros de ruídos é saber como cada um deles afeta a imagem. Com isso, adquire-se sentimento prático para saber qual filtro deve ser usado em cada caso.

Com isso em mente, escreva um programa capaz de aplicar o filtro gaussiano ou o filtro da mediana sobre uma imagem de entrada. O usuário deve ser capaz de informar o tamanho da janela dos dois filtros. Importante: *não utilize funções prontas para esse fim, a menos que isso seja indicado.*

2. Baixe o material suplementar da lista 2 (Landscapes.rar) disponível no site da disciplina. Aplique o filtro gaussiano sobre as imagens, utilizando janelas (*kernels*) de tamanho 3×3 , 7×7 e 15×15 pixels. Observe os efeitos do filtro — o que ocorre com a imagem como um todo, com os detalhes, com as bordas e com os pequenos grupos de dois ou três pixels que têm cores distintas de sua vizinhança. O objetivo do exercício não é fazer você descrever detalhadamente o que ocorre com cada tamanho de janela, mas sim “sentir” e relatar a influência desse filtro sobre a imagem. Dica: Trabalhe com as imagens em tons de cinza.

(a) Repita o mesmo para o filtro da mediana.

(b) Tome uma imagem que tenha gerado resultados interessantes para as questões anteriores. Corrompa a imagem com ruído aditivo gaussiano. Descreva os parâmetros do ruído. Passe novamente os filtros gaussiano e da mediana sobre a imagem ruidosa. Discuta suas impressões sobre os resultados obtidos: se os filtros são capazes de eliminar os ruídos, se é possível reduzir substancialmente os ruídos preservando-se as características originais da imagem, etc.

(c) Repita a questão anterior, porém use ruído do tipo sal-e-pimenta.

(d) Tome uma imagem como referência e analise o tempo de execução da aplicação de um dos filtros (media ou gaussiano) para cada um dos tamanhos de kernel considerados. Para isso, aplique cada filtro, no mínimo 5 vezes e, para cada aplicação conte o tempo gasto (dica: use as funções tic e toc do matlab), calcule e exiba o tempo médio e o desvio padrão e responda: Qual conclusão pode-se tirar a respeito do tempo de execução do algoritmo em função da variação do tamanho do kernel de convolução?

3. Implemente o algoritmo para detecção de quinas (CORNERS) descrito no livro texto da disciplina (Trucco), e exiba como resultado do algoritmo as quinas encontradas nas imagens de entrada, sobrepostas com pequenos retângulos ilustrando a posição de cada quina. Teste seu algoritmo para as imagens disponíveis no material suplementar da lista 2 (arquivo *Landscapes.rar* disponível na página da disciplina, variando o tamanho da vizinhança, de tamanho $2N + 1$, de cada pixel observado na imagem considerando-se $N \in \{3, 7, 9\}$. Discuta os resultados com base nos falsos positivos e falsos negativos observados.

4. Implemente os algoritmos de detecção de bordas de Sobel e de Roberts e aplique-os às imagens do exercício anterior. Sobreponha as bordas obtidas à imagem original de forma que essa sobreposição apareça em uma cor diferente da imagem original.

Compare os resultados obtidos na sua implementação contra os resultados da função *edge* (para os algoritmos de Sobel e Roberts) disponível no Matlab.