Universidade Federal de Minas Gerais — UFMG
Instituto de Ciências Exatas — ICEx
Departamento de Ciência da Computação — DCC

Disciplina: DCC884 — Visão Computacional
Professor: Mario Fernando Montenegro Campos (`mario@dcc.ufmg.br`)
Monitor: Daniel Balbino de Mesquita(`balbino@dcc.ufmg.br`)
Período: 1º semestre / 2013
Página: `http://www.verlab.dcc.ufmg.br/cursos/visao/2013-1/index`

2ª Lista de Exercícios

Data da entrega: 20/mai/2013

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 0,5 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. 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 OpenCV 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. Defina o ruído em uma imagem. Quais são as suas principais causas? Aponte alguns exemplos de problemas tratados pela Visão Computacional em que o ruído pode prejudicar os resultados. Justifique sua resposta.

2. No processo de filtragem linear de uma imagem, explique por que a utilização de um *kernel* não-negativo funciona como um filtro passa baixas. Em que condições esse procedimento é capaz de suprimir o ruído?

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. O que é um filtro separável? Quais são as vantagens da separabilidade? O filtro da média é separável? Por quê?

5. As bordas de uma imagem (assim como o ruído) correspondem a variações bruscas de intensidade entre pixels de uma mesma vizinhaça. Explique como você faria um algoritmo detector de bordas para uma imagem, a partir da idéia de filtragem linear. O que ocorre com a detecção de bordas e sua localização em imagens com muito ruído?

2 Exercícios Práticos

1. Conforme já discutido, o projeto de filtros para o tratamento de imagens é de fundamental importância para a maioria dos problemas de que tratam a Visão Computacional. Qualquer filtro de ruídos inevitavelmente destrói parte da informação de interesse em uma imagem. 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. Assim sendo, escreva um programa capaz de aplicar os filtros da média, gaussiano e da mediana sobre uma dada imagem de entrada. Além da imagem e do tipo de filtragem escolhida, o algoritmo ainda deve receber informações relevantes a cada tipo de filtro, como por exemplo, tamanho do kernel utilizado, desvio padrão (para o filtro gaussiano), entre outras características. Importante: Não utilize funções prontas para esse fim, a menos que isso seja indicado.

   (a) Baixe da internet uma imagem (rostos, paisagens, etc). Inicialmente aplique o filtro da média sobre a imagem escolhida, fazendo variar o tamanho da janela (kernel), desde 3×3 pixels até algo como 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 cor distinta de sua vizinhança.

(b) Repita o mesmo para o filtro gaussiano. Alguma diferença perceptível entre esse e o filtro da média?

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

(d) Agora corrompa a imagem com ruído aditivo gaussiano (utilize funções prontas para esse fim). Descreva os parâmetros do ruído. Passe novamente os filtros da média, 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.

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

2. 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.

   Para os usuários de Matlab, o desenho do retângulo sobre a figura pode ser feito da seguinte forma:

```
im=imread('img.jpg'); % Imagem carregada

imshow(im); % Exiba a imagem antes
            % de desenhar o retângulo na região desejada

hold on; % Diz ao matlab para desenhar os próximos
         % comandos em cima da janela atual, que
         % contém a imagem plotada

r=rectangle('Position', [x y w h]); % Desenha um
                                   % retângulo de tamanho
                                   % w por h na posição (x,y)
                                   % e retorna um handle na
                                   % variável r
```

   Estes comandos irão desenhar um retângulo de borda preta na imagem. Para alterar a cor da borda, basta executar:

```
set(r, 'EdgeColor', [r g b]) % Altera a cor da borda para a
                             % cor (r,g,b). Por padrão, cada
                             % canal deve estar no domínio [0,1]
```

Para listar outras características do retângulo que podem ser alteradas
(e os seus valores atuais), execute:

```
get(r)
```