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: Cláudio dos Santos Fernandes (`csantos@dcc.ufmg.br`)
Período: 1º semestre / 2012
Página: `http://www.verlab.dcc.ufmg.br/cursos/visao/2012-1/index`

1ª Lista de Exercícios

Data da entrega: 16/abr/2012

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 conceito de Visão Computacional. Quais são os tipos de problemas tratados neste contexto? Estes problemas são fáceis? Quais são as ferramentas utilizadas? Justifique.

2. O que ocorre com a imagem capturada por uma câmera projetiva se alterarmos a distância focal?

3. O modelo mais simples de câmeras projetivas é o *pinhole*, que não sofre com as limitações de profundidade de campo e dispensa um conjunto complexo de lentes e ajustes. A construção é tão simples que há na Internet vários tutoriais para construir câmeras *pinhole* de papel ou com latas de biscoito. Não obstante, as câmeras comerciais fogem desse modelo e são construídas com grandes aberturas. (Curiosamente, em geral as câmeras profissionais e as melhores lentes são as que possuem as maiores aberturas.)

   (a) Explique por que, com todas as vantagens, as câmeras *pinhole* não são amplamente usadas.

   (b) Quanto maior a abertura da lente, mais restrita é a profundidade de campo. Por quê?

   (c) Fotografias com tempo de exposição muito curto tendem a ter a profundidade de campo muito reduzida. Por exemplo, fotos que registram um beija-flor parado no ar costumam apresentar as pontas das asas fora de foco. Explique como o tempo de exposição e a profundidade de campo estão relacionados.

4. Explique por que superfícies Lambertianas são frequentemente referenciadas como tendo um brilho que independe do ângulo de visualização.

5. Imagine um ponto $Pw = [Xw; Yw; Zw]$ localizado no espaço 3D, e imageado por uma câmera c com parâmetros intrínsecos e extrínsecos conhecidos. Explique por que não é possível descobrir a posição real deste ponto no espaço, apenas com base em uma imagem I produzida por esta câmera. Justificar utilizando o "modelo projetivo" de câmeras. De que forma seria possível resolver este problema?

2 Exercícios Práticos

1. Escreva um programa (C++, Python ou Matlab) que mostre imagens de profundidade como imagens de intensidade (níveis de cinza codificam distâncias) ou como imagens sombreadas (ou tonalizadas) pelo coseno (*cosine shaded image*). Experimente com imagens disponíveis na Internet.

2. A aquisição de uma imagem de intensidade está sujeita à ação de diversas fontes de ruídos. O objetivo deste exercício é ter uma noção de quanto os ruídos podem interferir no processo de captura de imagens.

   Use uma câmera fixa para obter 10 imagens de uma mesma cena estática, ou seja, sem objetos em movimento. *Importante:* Salve as imagens em um formato sem perdas, de preferência PNG.

   Para melhorar os resultados, observe as dicas a seguir:

   - Procure uma cena iluminada por luz natural, evitando iluminação artificial;
   - Procure uma cena com objetos claros e escuros, mas evite a saturação (estouro de branco);
   - Para fixar a câmera, o melhor é usar um tripé;
   - O experimento pode ser feito com qualquer câmera: *webcam*, câmera fotográfica ou filmadora. O laboratório possui algumas câmeras que podem ser usadas neste experimento;
   - Tente desativar o ajuste automático de exposição (*auto gain control*) e não use *flash*.

   Responda as questões a seguir:

   (a) Descreva o seu sistema de captura, o tipo, marca e características da câmera, assim como as características das imagens obtidas.

   (b) Estime o desvio-padrão da aquisição do ruído em cada pixel, $\sigma(i, j)$. Gere uma imagem em escala de cinza para representar o mapa de ruídos medidos na imagem. Os valores de intensidade dos pixels devem ser escalados de modo que o maior ruído seja representado por branco (ou seja, intensidade `255`) e a ausência de ruído seja representada por preto (ou seja, intensidade `0`).

   Analise a imagem resultante. Em condições ideais — ou seja, segundo o modelo de ruído aditivo, branco e uniforme —, a imagem deve ser praticamente toda branca, indicando que a magnitude do ruído é aproximadamente a mesma em todos os pixels (e, portanto, sempre próxima do máximo), sem relação com a estrutura da imagem capturada. Foi esse o resultado que você obteve? Caso contrário, analise criticamente e tente tirar conclusões: quais são as regiões com o maior ruído? Há alguma relação entre o mapa de ruídos e as imagens da cena?

   (c) Plote um gráfico que apresente, para uma única linha de varredura no conjunto de imagens, a média de cada pixel e os valores acrescidos e reduzidos de $\sigma(i, j)$.

(d) Estime a média do ruído (a média de $\sigma(i,j)$). O que este valor indica?

(e) Qual o pior caso de aquisição de ruído?

(f) Uma fonte comum de ruídos é a causada pela compressão das imagens, usualmente feita segundo os padrões JPEG ou MPEG. É por isso que o enunciado pede para salvar as imagens em um formato sem perdas. Agora vamos analisar a compressão como fonte de ruídos.

Os programas que fazem a conversão de imagens para JPEG permitem configurar o nível de compressão. É comum que apresentem uma escala qualitativa de compressão, permitindo a geração de arquivos de alta qualidade ou de tamanho pequeno.

Converta as 10 imagens originais para o formato JPEG, gerando imagens de alta qualidade. (Dica: o aplicativo IrfanView permite converter vários arquivos de uma vez.) Repita os exercícios (b), (d) e (e) para esse novo conjunto de imagens. Analise os resultados, comparando-os com os obtidos originalmente. Repita o processo para arquivos JPEG de baixa qualidade e discuta se, no seu caso, a compressão gerou ruídos significativos.

3. Para obter-se uma imagem radialmente distorcida a partir de uma figura I_o, basta afastar cada um de seus pixels do centro da imagem por um fator proporcional à sua distância original. Em outras palavras, pixels mais próximos do centro sofrerão uma distorção maior do que a recebida por aqueles mais distantes.

   Matematicamente, se c_o são as coordenadas do centro da imagem, cada pixel da imagem distorcida I_d terá coordenadas dadas por:

$$p_d = c_o + r(K_0 n^{1/2} + K_1 n^{1/4} + ...)$$

   Em que $r = p_o - c_o$, $n = |r|$, p_o são as coordenadas do pixel p na imagem I_o e p_d são as coordenadas do pixel p na imagem distorcida (I_d).

   Sem utilizar funções especiais da linguagem/biblioteca empregada, resolva os seguintes itens:

   (a) Escreva um programa que realize distorção radial de uma imagem e submeta uma imagem distorcida com parâmetro $K_0 = 0.05$.

   (b) Escreva um programa que realize a transformação inversa à distorção (*undistort*). Experimente executá-lo com as imagens fornecidas e parâmetros $K_0 = 0.05$ e $K_1 = -0.025$.