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:	Armando Alves Neto (`aaneto@dcc.ufmg.br`)
Período:	1º semestre / 2008
Aulas:	Segundas e quartas, das 09:25 às 11:05, na sala 2029 do ICEx
Página:	`http://www.dcc.ufmg.br/verlab/doku.php?id=cursos:visao:index`

3ª Lista de Exercícios

Data da entrega: 28/maio/2008

A lista é individual e as datas de entrega são fixas, devendo o trabalho ser entregue em formato impresso 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 como o espaço de parâmetros (θ, ρ) oferece melhor representação do que o espaço (m, n) para a Transformada de Hough. O que significa um ponto no espaço de Hough, em relação ao plano da imagem ?

2. A detecção de linhas pode ser implementada como um procedimento do tipo *template matching*, que nada mais é do que a filtragem da imagem com máscaras que respondam a linhas em diferentes orientações, seguida de um limiar para selecionar os pixels que pertençam a linha. Compare o *template matching* com a Transformada de Hough como métodos para a detecção de linhas em imagens.

3. O 1º Exercício Prático trata do cálculo dos autovalores e autovetores correspondentes aos pixels de uma imagem. A Transformada de Hough poderia fazer uso desses dados para melhorar os resultados? Como?

2 Exercícios Práticos

1. O arquivo "corners.m", disponível no site do curso, calcula os dois autovalores $\lambda_1 \geq \lambda_2$ como apresentado no algoritmo CORNERS na página 84 do livro-texto [Trucco and Verri, 1998]. O autovetor correspondente a λ_1 também é calculado.

   (a) Como você pode usar esses autovalores para a detecção de arestas? (O conjunto de arestas detectadas não deve incluir as quinas).

   (b) Como você encontra a direção do gradiente?

   (c) Implemente um programa com base nas suas idéias expostas nas questões anteriores. Para as arestas, use *nomaximum suppression* mas não execute a histerese. Utilize um algoritmo simples de limiar (*thresholding*) para determinar se um pixel deve ou não pertencer a uma aresta. As quinas não necessitam ser pontos isolados (pule os passos 3 e 4 do algoritmo CORNERS). Seu programa deve produzir uma imagem que é branca exceto onde existe um marcador na forma de ponto nos locais onde existem arestas e um marcador em forma de quadrinho no local onde uma quina for detectada. Experimente o seu algoritmo com a imagem *taj_mahal.jpg* (Figura 1), disponível no site do curso. Se quiser, experimente também com imagens da internet.

2. O arquivo "sequencia.zip" apresenta uma pequena sequência de imagens representando os quadros de um vídeo. Neste vídeo, observa-se uma barra retangular efetuando movimento de translação e rotação

em relação ao eixo Z de uma câmera. Utilizando o algoritmo da questão anterior (CORNERS), e o modelo projetivo da câmera, apresente um gráfico que mostre a distância do centro da barra até o centro da câmera ao longo do tempo. Quando necessário, assuma valores arbitrários para os parâmetros da câmera (distância focal f, razão de aspecto s, etc) e desconsidere quaisquer distorções de lente. O mesmo é válido para as dimensões da barra. Normalize o valor final obtido. Comente o resultado final.

Figura 1: Imagem de teste.

Referências

Emanuele Trucco and Alessandro Verri (1998). *Introductory Techniques for 3-D Computer Vision*, volume 1. Prentice Hall PTR, Upper Saddle River, NJ, USA.