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: 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

3ª Lista de Exercícios

Data da entrega: 05/Junho/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. Descreva as diferenças entre características locais e globais e dê exemplos de características de imagens para cada classe (local e global). Seus exemplos devem ser diferentes dos exemplos citados no livro texto da disciplina.

2. Defina a Transformada de Hough e descreva um algoritmo para executar a transformada de Hough para formas em geral como retas, circulos de raio dado, elipses, etc..

3. 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. Por exemplo, linhas verticais, horizontais ou em 45^o podem ser detectadas através da convolução da imagem com templates do tipo:

-1	-1	-1
2	2	2
-1	-1	-1

Horizontal

-1	-1	2
-1	2	-1
2	-1	-1

$+45^o$

-1	2	-1
-1	2	-1
-1	2	-1

Vertical

2	-1	-1
-1	2	-1
-1	-1	2

-45^o

Compare o template matching com a Transformada de Hough como métodos para a detecção de linhas em imagens. Ressalte os prós e contras de cada método com relação à generalidade, detecção de falsos positivos e falsos negativos, robustez e eficiência computacional.

2 Exercícios Práticos

1. Neste exercício você irá usar técnicas de extração de características em imagens para identificar as 18 linhas que subdividem o tabuleiro de xadrez exibido na Figura 1.

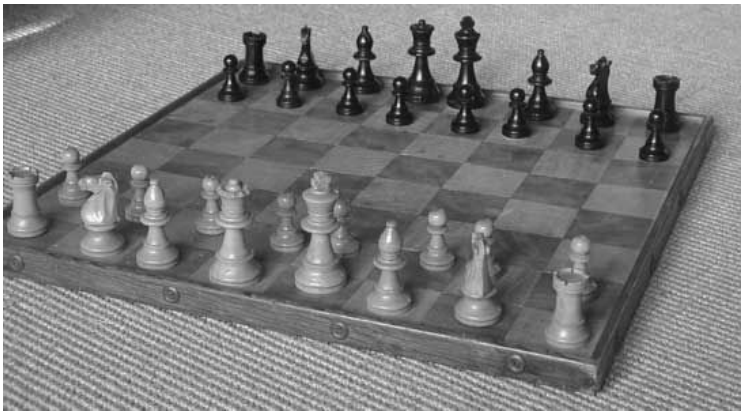

Figura 1: Tabuleiro de Xadrez.

Para isto, você poderá fazer uso dos programas `houghTrans.m` e `ht_plot2.m` disponíveis no material complementar da LE3 (no site da disciplina).

O programa `houghTrans.m` faz uso da função *hEdge(I)*, onde *I* representa a imagem de entrada. A função *hEdge* é responsável pelo pré-processamento da imagem de entrada retornando uma imagem binária contendo suas bordas. Essa imagem será, posteriormente, utilizada no processo de detecção de retas na transformada de Hough.

(a) Implemente e otimize a função *hEdge* de forma que, o programa *houghTrans* detecte o maior número possível de retas entre os padrões claros e escuros do tabuleiro de xadrez da Figura 1. Nenhuma modificação poderá ser feita nos programas *houghTrans.m* e *ht_plot2.m*.

(b) Analise criticamente os resultados obtidos: todas as retas importantes foram detectadas? Há retas detectadas que não existem na imagem original? Há retas que foram detectadas mais de uma vez - ou seja, com mais de uma combinação

de pares q, r? Por quê? Filosoficamente, o que poderia ser feito para melhorar os resultados?

Dica: use a função *edge* do Matlab para o detector de *Canny* variando seus parâmetros.

2. Faça sua implementação da transformada de Hough para detecção retas e círculos.

   (a) Gere duas imagens sintéticas de mesmo tamanho, sendo que, uma delas deve conter uma única reta e, a outra, um único círculo e, ambas as formas geradas, devem possuir o mesmo número de pontos. Compare o tempo de execução dos dois algoritmos considerando a imagem da reta para o detector de retas e, a imagem do círculo para o detector de círculos.

   (b) Corrompa as imagens de bordas geradas no item anterior com ruído do tipo sal-e-pimenta, e estime a robustez dos dois algoritmos plotando os erros nos parâmetros estimados como uma função da quantidade de ruído adicionada.