CS 223b: Introduction to Computer Vision
Assignment #1: Image Features
(out of 25 points)

Due date: Monday, January 19th, 2003 (no extension)
N.B.: You are not allowed to work in teams on this assignment.

1 Corner Finder (20 pts.)

(a) **(15 pts.)** Implement in MATLAB the algorithm *CORNERS* on page 84 of *Trucco and Verri*. The function header should be:

function [c, lamba2] = corners(I, N, tau)

where **I** is the image (a matrix), **N** refers to the window size, which should be $(2N + 1) \times (2N + 1)$, and **tau** is the threshold on λ_2. The function returns a $2 \times K$ matrix **c** and a K-dimensional vector **lambda2**, where K is the number of features found. The columns of **c** are the coordinates of the center of each selected feature window, and **lambda2** contains the values of λ_2 for each feature. Features in **c** and values in **lambda2** are sorted in decreasing order of the magnitude of λ_2.

(b) **(5 pts.)** Test your algorithm on the images called **test1.pgm-test3.pgm** located on the class website in the directory:

http://robots.stanford.edu/cs223b/homework/hw1

Write a function which displays your results:

function showResults(Image, k, c, N, lambda2)

which draws the k best corners from your detector's output as $(2N + 1) \times (2N + 1)$ squares centered at each of the feature points.

(c) **To hand in:** you are responsible for handing in the following:

1. An electronic copy of your MATLAB m-file called **corners.m** (we will be testing your code on some images similar to the ones online).
2. An electronic copy of your MATLAB m-file called **showResults.m**

The code must be emailed to **cs223b@cs.stanford.edu** before midnight on the day the assignment is due.

2 Subpixel Corner Finder (5 pts.)

The previous algorithm does a reasonable job finding corners. There exist, however, more sophisticated techniques for refining these initial estimates to achieve subpixel accuracy. The classic textbook subpixel corner detector (Harris) makes direct use of the orthogonality between the image gradient vector ∇I at any point $\mathbf{p}$ in the neighborhood of the true (unknown) corner point $\mathbf{q}$ and the vector connecting $\mathbf{p}$ and $\mathbf{q}$. See figure 1. Mathematically, this orthogonality property is expressed in the form of a single scalar equation ($\nabla I \cdot (\mathbf{q} - \mathbf{p}) = 0$ or, the dot product = 0), with unknowns x and y, the coordinates of $\mathbf{q}$. Considering a single point $\mathbf{p}$ leads to an underconstrained problem (one equation, two unknowns). However, accumulating the set of scalar constraints for a set of points $\mathbf{p}$ in the neighborhood of $\mathbf{q}$ is the mathematical way to lift the under-constrained problem, turn it into an overconstrained problem and solve explicitly for x and y. A least squares method can be used for that purpose.

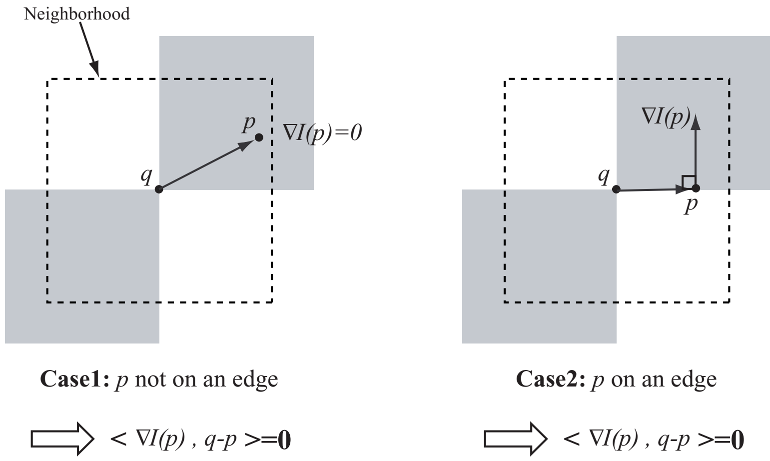

Fig. 1: **The subpixel corner finder:** For any pixel $\mathbf{p}$ in the neighborhood of the true corner $\mathbf{q}$, the vector $\mathbf{pq}$ is always orthogonal to the gradient of the image $\nabla I(\mathbf{p})$ at $\mathbf{p}$.

Let $\mathbf{q}^k = [x_k \quad y_k]^T$ be an initial (or current) guess for the corner point $\mathbf{q}$. The objective is to compute a refinement for this corner location: $\mathbf{q}^{k+1}$. Consider a small neighborhood $\mathcal{W}$ centered around $\mathbf{q}^k$, and denote by $\mathbf{p}_i$ ($i = 1, \ldots, N$) the set of points in $\mathcal{W}$. At every point $\mathbf{p}_i$, consider the residual dot product function $\epsilon_i(\mathbf{q}^{k+1})$:

$$\epsilon_i(\mathbf{q}^{k+1}) = \epsilon(\mathbf{q}^{k+1}, \mathbf{p}_i) = \left\langle \nabla I(\mathbf{p}_i), \mathbf{q}^{k+1} - p_i \right\rangle \tag{1}$$

where $< ., . >$ is the standard scalar product and $\nabla I(\mathbf{p}_i)$ is the 2×1 image gradient vector at $\mathbf{p}_i$:

$$\nabla I(\mathbf{p}_i) = \begin{bmatrix} \frac{\partial I}{\partial x} \\ \frac{\partial I}{\partial y} \end{bmatrix} (\mathbf{p}_i) \tag{2}$$

(a) **(5 pts.)** Derive the expression for the solution vector $\mathbf{q}^{k+1}$ that minimizes the following global cost function:

$$C(\mathbf{q}^{k+1}) = \sum_{i=1}^{N} \epsilon_i^2(\mathbf{q}^{k+1}) \tag{3}$$

The is known as the *least squares* solution. Hint: Set the derivative of C with respect to the coordinates x_{k+1} and y_{k+1} to zero and solve for x_{k+1} and y_{k+1}.

(b) **To hand in:** you are responsible for handing in the following:

1. The derivation of your expressions for x_{k+1} and y_{k+1}

This work must either be emailed to **cs223b@cs.stanford.edu** or delivered to Gates 114 by midnight on the day the assignment is due.