CS 223b: Introduction to Computer Vision
Assignment #1 Solutions

1 Corner Finder

Here's an example solution:

```
function [c, lambda2] = corners(I,N,tau)

% this function calculates the gradient using convolution with the
% derivative of a Gaussian
[Ex Ey] = my_grad(I);

[rows cols] = size(I);

filter = ones(2*N+1, 2*N+1);

Sx = conv2(Ex.*Ex,filter,'same');
Sy = conv2(Ey.*Ey,filter,'same');

Sxy = conv2(Ex.*Ey, filter, 'same');

allc = -ones(rows*cols,2);
lam2 = -ones(rows*cols,1);

idx = 1;
for i=2*N+1:rows-(2*N+1)          % ignore image margins
  for j=2*N+1:cols-(2*N+1)
    C = [Sx(i,j),     Sxy(i,j)
         Sxy(i,j), Sy(i,j)];
    evals = eig(C);
    l2 = min(evals);
    if l2 > tau
        lam2(idx) = l2;
        allc(idx,:) = [i, j];
```

```
            idx = idx + 1;
        end
    end
end

allc = allc(1:idx-1,:);
lam2 = lam2(1:idx-1);

if idx == 1
    fprintf(1, 'No corners using this threshold!\n');
    lam2 = 0;
    c = [-1 -1];
    return
end

clen = length(lam2);

[lam2, indices] = sort(lam2);
lam2 = lam2(length(lam2):-1:1);
indices = indices(length(indices):-1:1);
tmpc = allc;

for (i=1:size(allc,1))
    tmpc(i,:) = allc(indices(i),:);
end

bmsk = ones(1,clen);
fprintf(1,'Number of potential corners = %d\n', clen);

for i=1:clen
    for j=i+1:clen
      if bmsk(j)
        if (abs(tmpc(i,1)-tmpc(j,1)) <= N)      & (abs(tmpc(i,2)-tmpc(j,2)) <= N)
              bmsk(j) = 0;
        end
      end
    end
end

newlen = sum(bmsk);
fprintf(1,'Number of final corners = %d\n', newLen);
c = zeros(newlen,2);
lambda2 = zeros(1,newlen);
```

```
idx = 1;
for i=1:clen
  if bmsk(i)
    c(idx,:) = tmpc(i,:);
    lambda2(idx) = lam2(i);
    idx = idx + 1;
  end
end
```

2 Subpixel Corner Finder

Without loss of generality, we let $q^{k+1} \equiv q$. So:

$$C(q) = \sum_{i=1}^{N} (\nabla I^T(p_i) \cdot (q - p_i))^2$$

Where:

$$\nabla I(p_i) = \begin{bmatrix} I_{x_i} \\ I_{y_i} \end{bmatrix}$$

$$\nabla q = \begin{bmatrix} q_x \\ q_y \end{bmatrix}$$

$$\nabla I(p_i) = \begin{bmatrix} p_{x_i} \\ p_{y_i} \end{bmatrix}$$

so,

$$C(q) = \sum_{i=1}^{N} (I_{x_i} \cdot (q_x - p_{x_i}) + I_{y_i}(q_y - p_{y_i}))^2$$

$$\frac{\partial C(q)}{\partial q} = \left[\frac{\partial C}{\partial q_x}, \frac{\partial C}{\partial q_y}\right]$$

$$\frac{\partial C(q)}{\partial q_x} = 2 \sum_{i=1}^{N} (I_{x_i}(q_x - p_{x_i}) + I_{y_i}(q_y - p_{y_i}))I_{x_i}$$

$$\frac{\partial C(q)}{\partial q_y} = 2 \sum_{i=1}^{N} (I_{x_i}(q_x - p_{x_i}) + I_{y_i}(q_y - p_{y_i}))I_{y_i}$$

Regrouping, we get:

$$\left[\frac{\partial C(q)}{\partial q}\right]^T = 2 \sum_{i=1}^{N} \begin{bmatrix} I_{x_i} \\ I_{y_i} \end{bmatrix} \begin{bmatrix} I_{x_i} & I_{y_i} \end{bmatrix} (q - p_i)$$

$$\left[\frac{\partial C(q)}{\partial q}\right]^T = 2(Gq - b)$$

Where:

$$G = \sum_{i=1}^{N} \nabla I(p_i) \nabla I^T(p_i) = \begin{bmatrix} \sum_{i=1}^{N} I_{x_i}^2 & \sum_{i=1}^{N} I_{x_i} I_{y_i} \\ \sum_{i=1}^{N} I_{x_i} I_{y_i} & \sum_{i=1}^{N} I_{y_i}^2 \end{bmatrix}$$

$$b = \begin{bmatrix} \sum_{i=1}^{N} (I_{x_i}^2 p_{x_i} + I_{x_i} I_{y_i} p_{y_i}) \\ \sum_{i=1}^{N} (I_{y_i}^2 p_{y_i} + I_{x_i} I_{y_i} p_{x_i}) \end{bmatrix}$$

So the solution becomes:

$$q = q^{k+1} = \begin{bmatrix} x_{k+1} \\ y_{k+1} \end{bmatrix} = G^{-1} b$$