TP4
Feature detection

The objective of this practical work is to implement the Harris corner detector. Note that in order to manipulate pixel values, float arrays will be advantageously used.

1. Use the previous practical work to write a c program that computes the images I_x and I_y of the gradients of an image (smoothed), in the x and y directions respectively, using the Sobel operator.

2. Using I_x and I_y, compute the values of I_x^2, I_y^2 and $I_{xy} = I_x \times I_y$ and store them in arrays. Smooth these values using the binomial filter implemented in the previous practical work.

3. Compute the Harris function $H = \det C - \alpha \text{ trace}^2 C$ at each pixel. Values of this function depend on the autocorrelation matrix C defined by $C = \begin{pmatrix} I_x^2 & I_{xy} \\ I_{xy} & I_y^2 \end{pmatrix}$, where α is a parameter of the detector to be tuned and where the elements of C, at any pixel, correspond to values of that pixel in the arrays of the previous question.

4. Display the image of H.

5. How does α influence H (typical values range from 0.01 to 0.2) ?

6. Display the n local maxima of H where n is a parameter that can be modified.

7. Shi and Tomasi proposed an improvement of the Harris detector. Assume that λ_1, λ_2 are the eigenvalues of the matrix C and take the minimum of the 2 values. Corners are then the local maxima of this min eigenvalue.

   Using the following function that computes the eigenvalues of the matrix $C = \begin{pmatrix} A & B \\ C & D \end{pmatrix}$ find the Shi-Tomasi corners and compare them to Harris.

Edmond.Boyer@inrialpes.fr

```
#define tolerance 0.1e-20
void eigenvalues(double A, double B, double C, double D,
    double* lambda1, double* lambda2)
{
    if(B*C <= tolerance  ) {
        *lambda1 = A; *v1x = 1; *v1y = 0;
        *lambda2 = D; *v2x = 0; *v2y = 1;
        return;
    }

    double tr = A + D;
    double det = A * D - B * C;
    double S = sqrt( square(tr/2) - det );
    *lambda1 = tr/2 + S;
    *lambda2 = tr/2 - S;
}
```