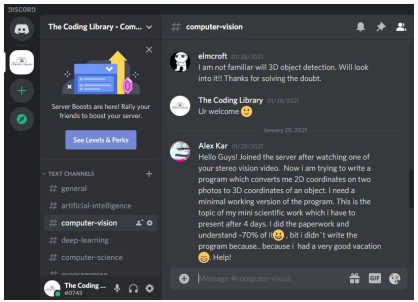
THE CODING LIBRARY

Computer Vision

Corner Detection



Discord Link in Description

Corners in Images

THE CODING LIBRARY

- Features in images
  - Matching points between frames of an environment
    - Relate images to each other and extract information
  - Features are uniquely recognizable
- Types of features in images
  - Edges
  - Corners (interest points)
  - Blobs (regions of interest)
- Corner is an intersection of two edges
  - Point in which the directions of the two edges change
  - Gradients have a high variation

How to Find Corners in Images

THE CODING LIBRARY

- Determine Cornerness values for each pixel
  - Produces a Cornerness map
- Non-maxima suppression
  - Multiple responses
  - Compare to local neighbours
- Threshold the Cornerness map
  - To get the most significant corners

Harris Corner Detector

THE CODING LIBRARY

- Variation of intensity

$$E(u,v) = \sum_{x,y} w(x,y)[I(x+u, y+v) - I(x,y)]^2$$

where:

- $w(x,y)$ is the window at position (x,y)
- $I(x,y)$ is the intensity at (x,y)
- $I(x+u, y+v)$ is the intensity at the moved window $(x+u, y+v)$

- Using Taylor expansion, rearranging and on matrix form

$$M = \sum_{x,y} w(x,y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \qquad E(u,v) \approx \begin{bmatrix} u & v \end{bmatrix} M \begin{bmatrix} u \\ v \end{bmatrix}$$

- Calculate a score of the probability that it is a corner

$$R = det(M) - k(trace(M))^2$$

where:

- det(M) = $\lambda_1 \lambda_2$
- trace(M) = $\lambda_1 + \lambda_2$

a window with a score R greater than a certain value is considered a "corner"

Harris Corner Detector

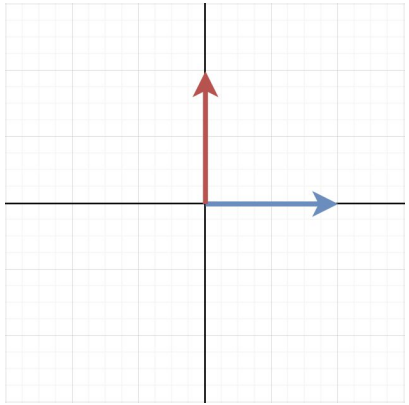
THE CODING LIBRARY

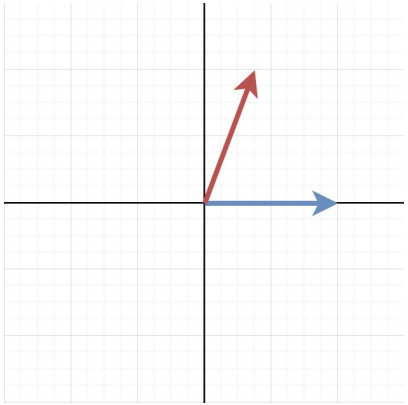
Eigenvalues and Eigenvectors

THE CODING LIBRARY

- Linear Algebra
- An eigenvector changes by a scalar factor when a linear transformation is applied to it
- Eigenvalue is the factor which the eigenvector is scaled

$$A\mathbf{u} = \lambda\mathbf{u}.$$





Shi-Tomasi Corner Detection

THE CODING LIBRARY

- Uses eigenvalues to detect corners and same approach as Harris Corner
  - Calculate the corner quality at every pixel in the image
- Non-maximum suppression
- Corners with minimal eigenvalues less than some threshold are rejected
- Remaining corners are sorted by quality in descending order
- Throws away corners where there is a better corner at a distance less than a maximum distance

Shi-Tomasi Corner Detection

THE CODING LIBRARY

Harris Corner VS Shi-Tomasi

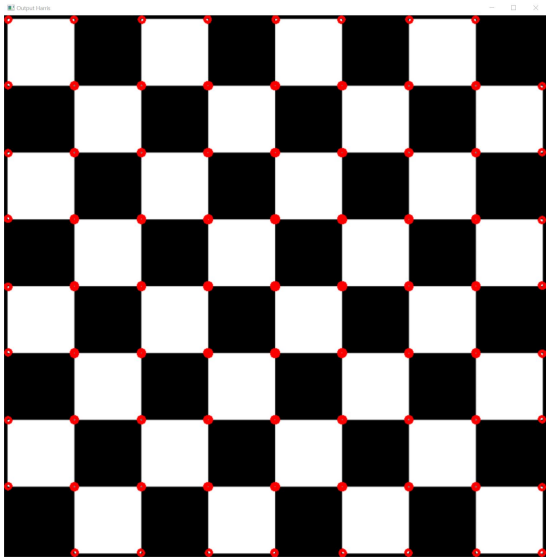
THE CODING LIBRARY

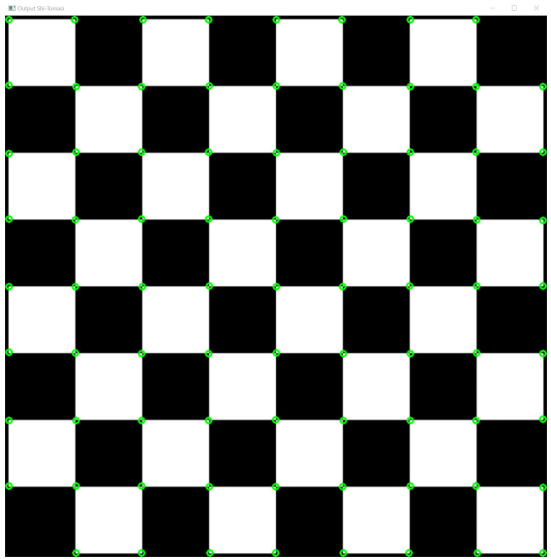
```
cornerHarris(gray, output,
    3,              // Neighborhood size
    3,              // Aperture parameter for the Sobel operator
    0.04);          // Harris detector free parameter
```

```
goodFeaturesToTrack(gray,
    corners,
    100,            // Max corners to detect
    0.01,           // Minimal quality of corners
    10,             // Minimum Euclidean distance between the returned corners
    Mat(),          // Optional region of interest
    3,              // Size of an average block for computing a derivative covariation matrix over each pixel neighbothood
    false,          // Use Harri Detector or cornerMinEigenVal - Like when you create your own
    0.04);          // Free parameter for the Harris detector
```

Harris Corner VS Shi-Tomasi

THE CODING LIBRARY

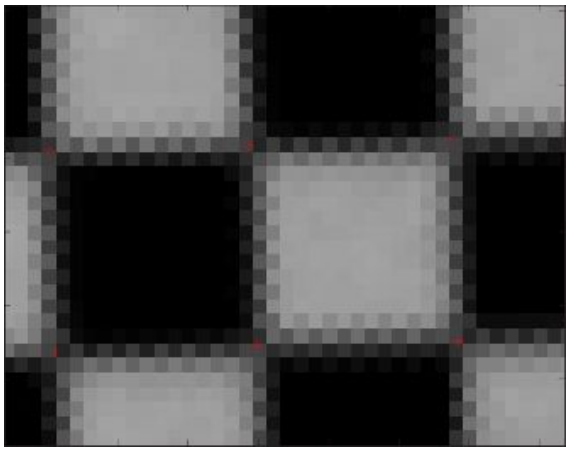


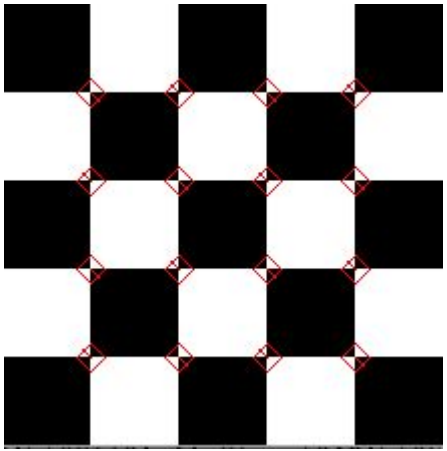


Corner Detection in Subpixels

THE CODING LIBRARY

- A way to find more exact corner positions by using subpixels
  - Normally it is integer pixels





Create Your Own Corner Detector

THE CODING LIBRARY

- Create your own corner detector like the Harris and Shi-Tomasi
- Find the eigenvalues and eigenvectors to determine if a pixel is a corner
- Find the minimum eigenvalues for corner detection