מכון טכנולוגי חולון
Holon Institute of Technology

Algorithms in multimedia and machine learning in the Python environment

Harris Corner Detection

Idan Montekyo & Yarin Shlomo

Lecturer - Yakir Menahem

תשפ"א

https://youtu.be/zVIaCBO8HR4

Background

Harris corner detector is an algorithm to find corners in an image.

What is a corner?

A corner is an intersection between two edges.
Corners can be referred as ‘key-points’ or ‘features’,
and generally used as interest points.

Harris corner detection algorithm is used in computer vision applications such as:

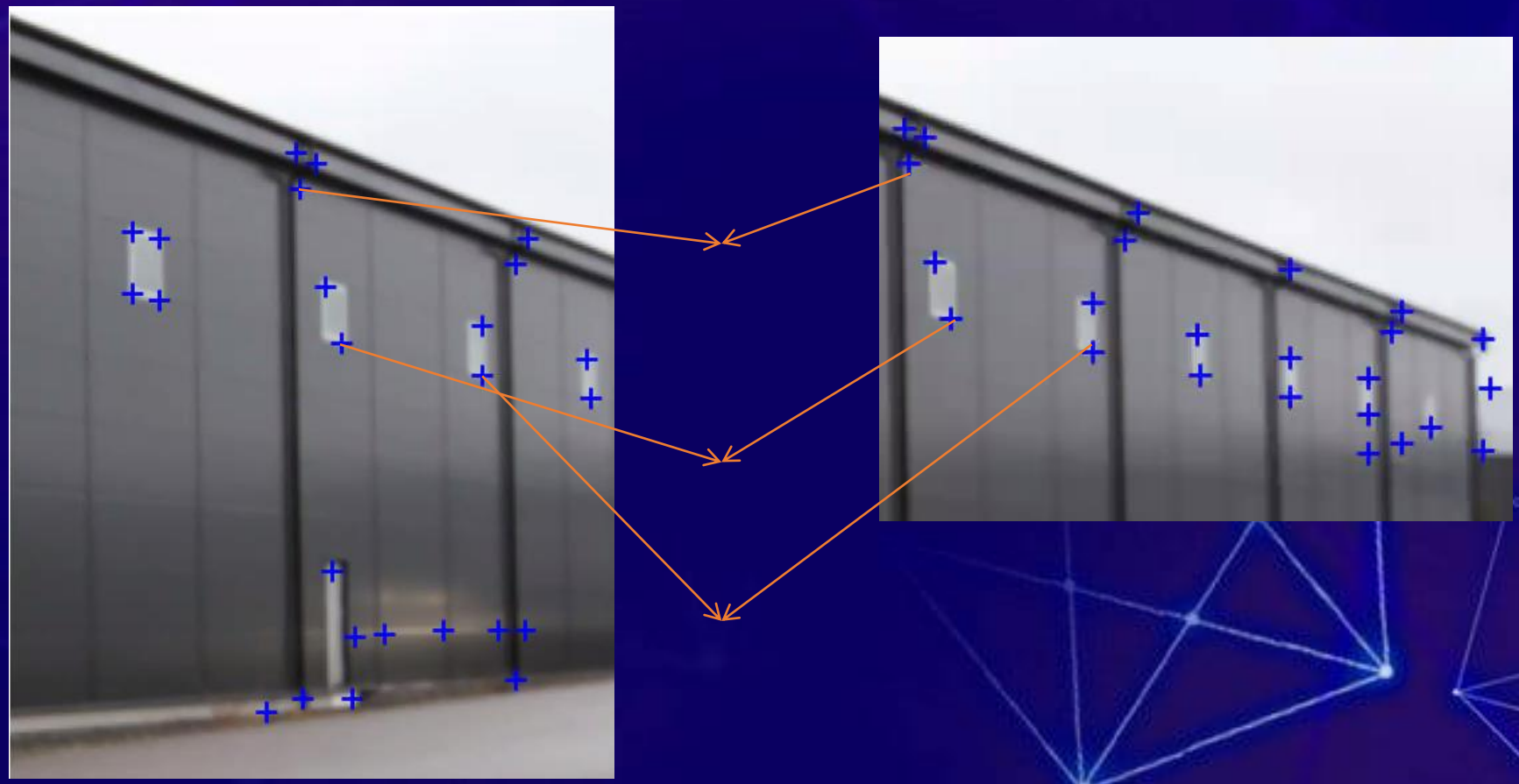
- Object detection
- Image stitching
- Morphing

The need

An example of usage in picture stitching, using corner detection:

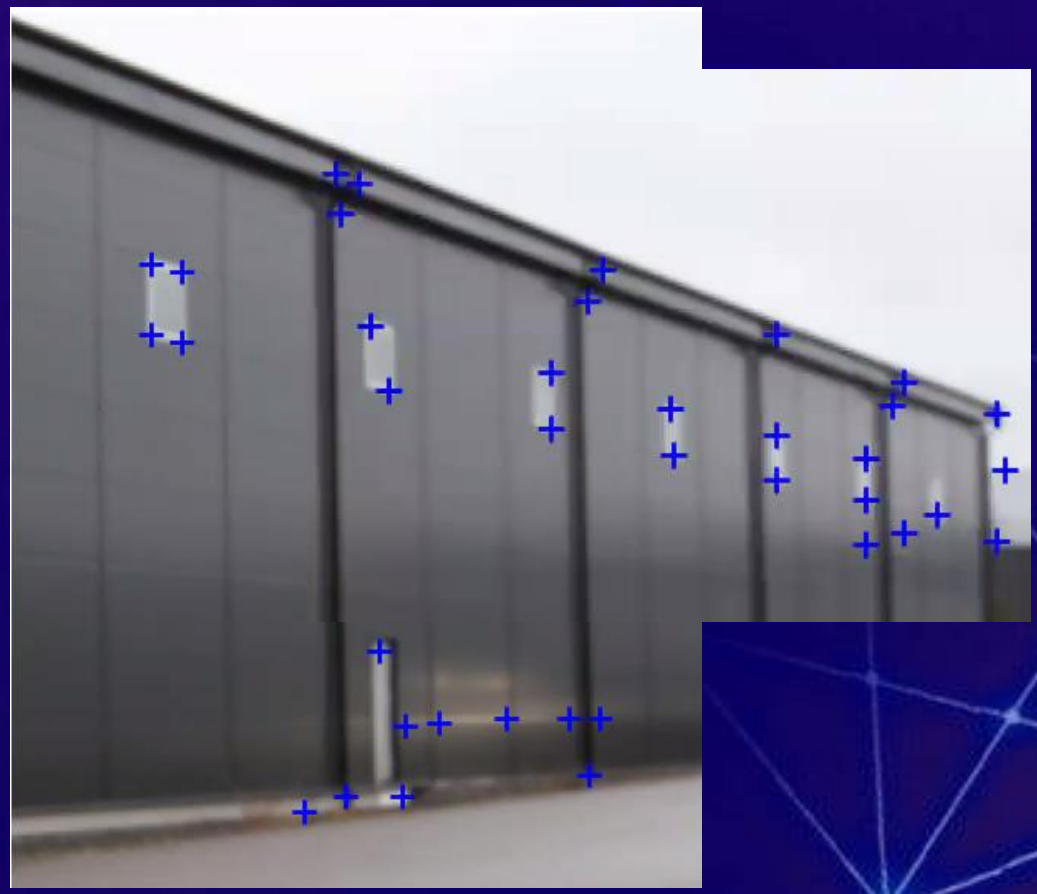
The need

An example of usage in picture stitching, using corner detection:



The need

An example of usage in picture stitching, using corner detection:



Existing Solutions

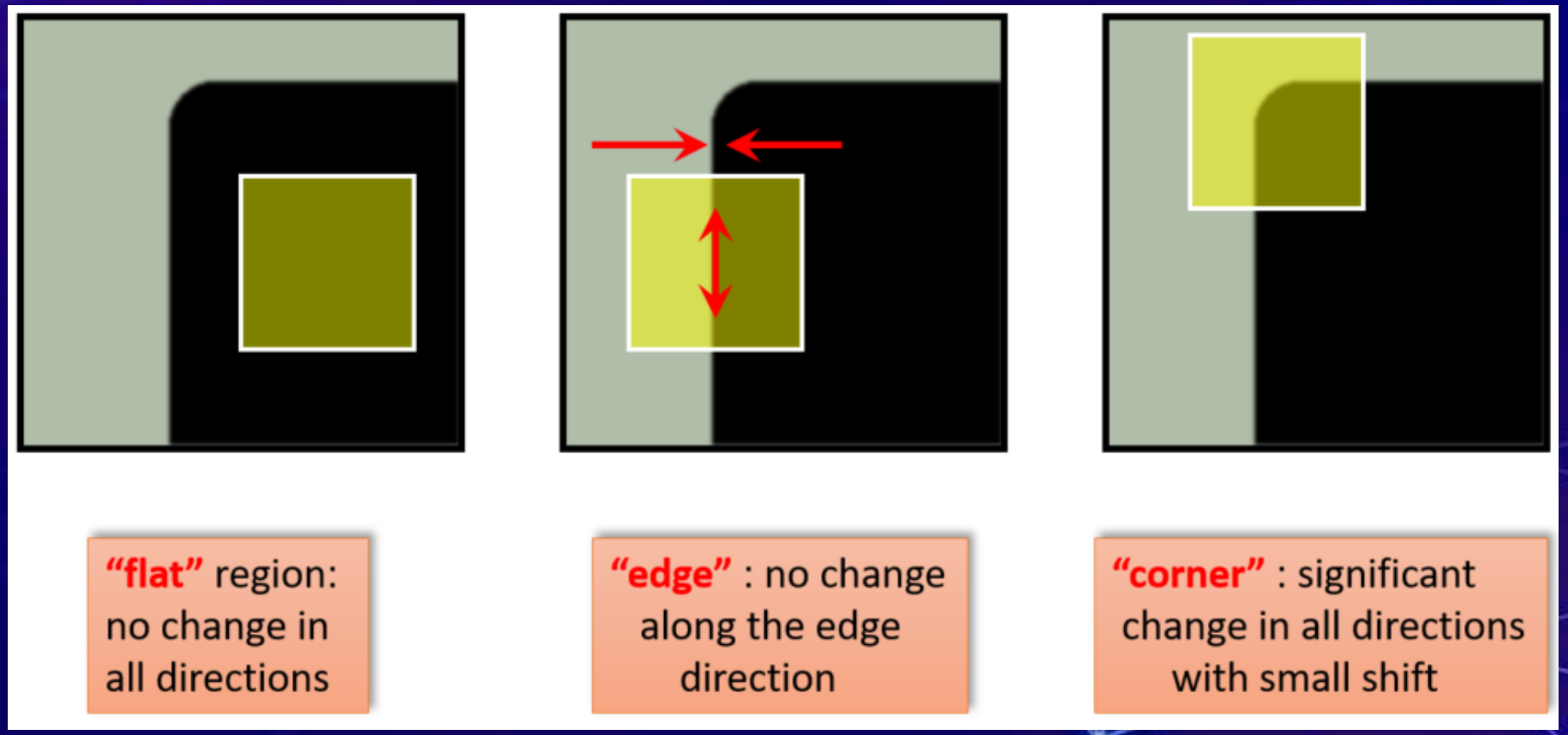
There are many algorithms for corner detection:

- Harris Corner Detection
- Shi-Tomasi Corner Detection
- Scale-invariant feature transform (SIFT detection)

All of the algorithms provide corner-detection features, although each algorithm uses a different method.

How do we find a corner?

Easily recognized by looking through a small window
Shifting the window should give large change in intensity



Mathematical background

Matrix **Determinant** and **Trace** calculation

(Also calculation of **eigenvalues**)

$$M = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

$$\det(M) = ad - bc = \lambda_1 \cdot \lambda_2$$

$$\mathrm{tr}(M) = a + d = \lambda_1 + \lambda_2$$

Harris Corner Detection Algorithm

Algorithm flow:

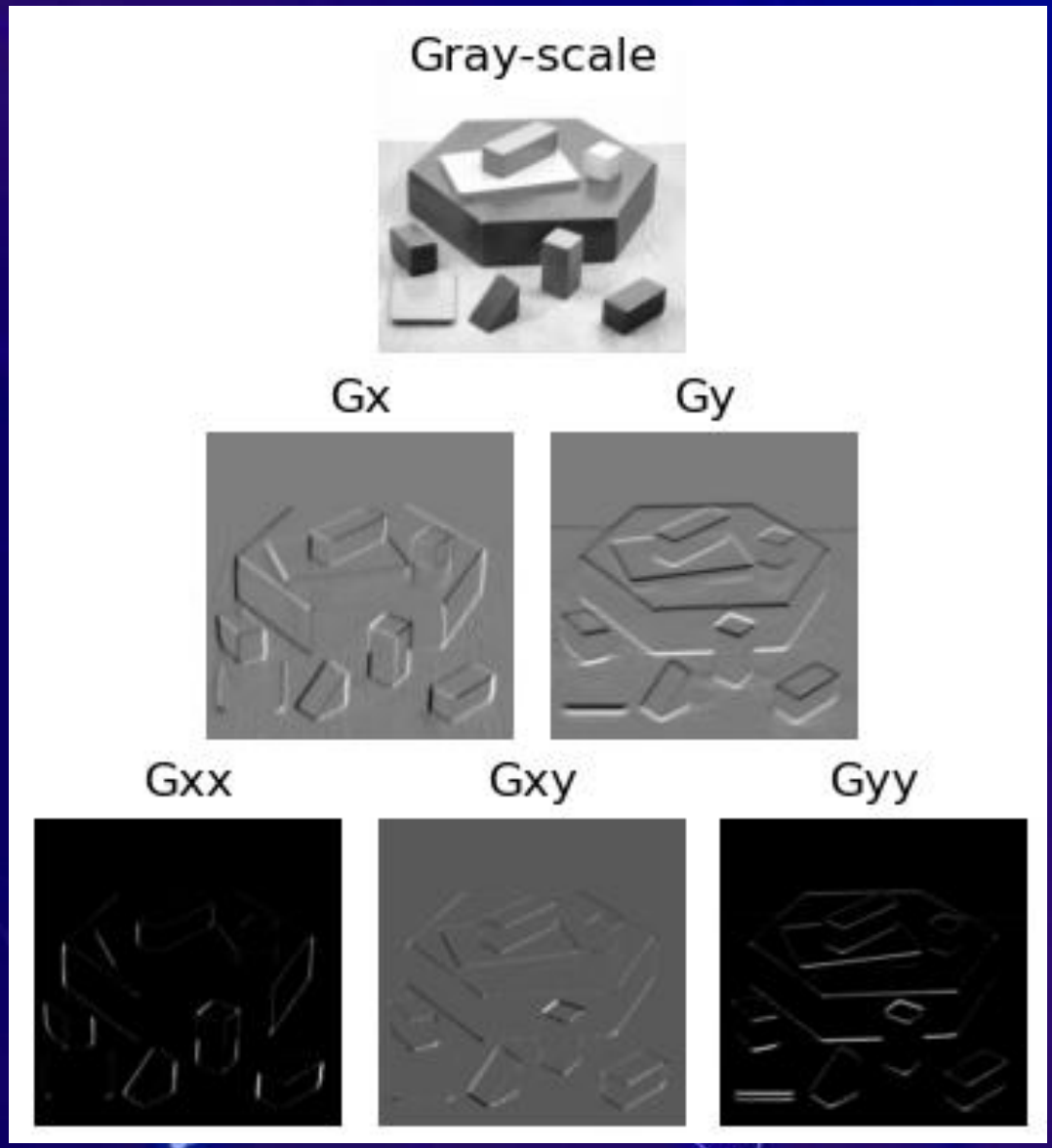
1. Convert the image to gray-scale (2D)
2. Compute image gradients: G_x , G_y
3. Compute 2nd order gradients: G_x^2 , G_{xy} , G_y^2
4. Apply Gaussian-mask on 2nd order gradients
5. For each pixel (i, j) define the matrix M
6. For each pixel compute the score R
7. Threshold R and perform NMS (Non-Maxima Suppression)

Harris Corner Detection Algorithm

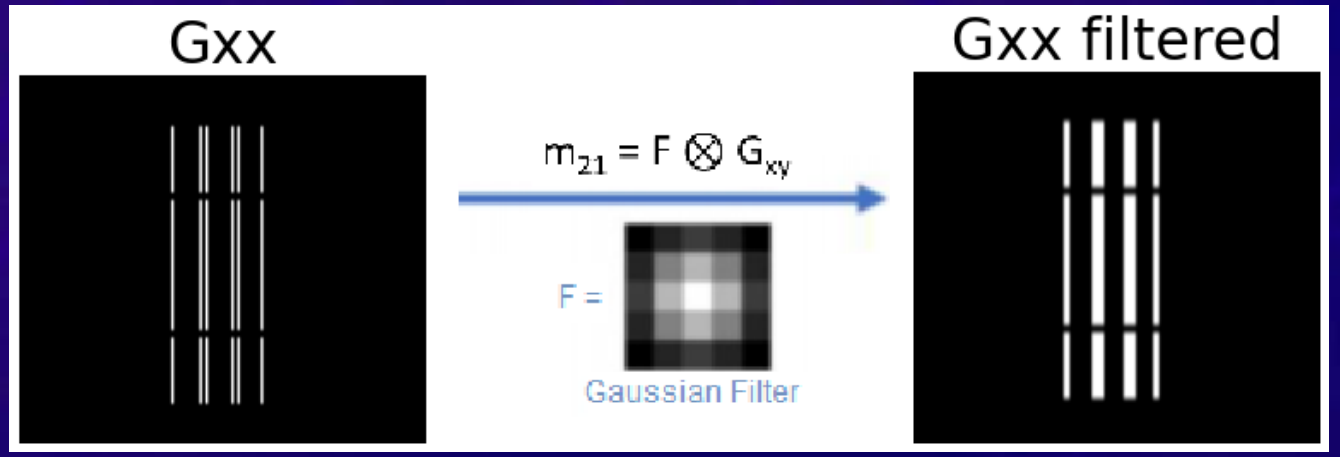
1. Convert the image to gray-scale (2D):
In gray-scale it is easier to track changes in color and directions.

2. Compute image gradients G_x , G_y

3. Compute 2nd order gradients G_x^2 , G_{xy} , G_y^2



4. Apply Gaussian-mask on 2nd order gradients:
(Compute the sums of 2nd order gradients at each pixel)



5. For each pixel (i, j) define the matrix M:

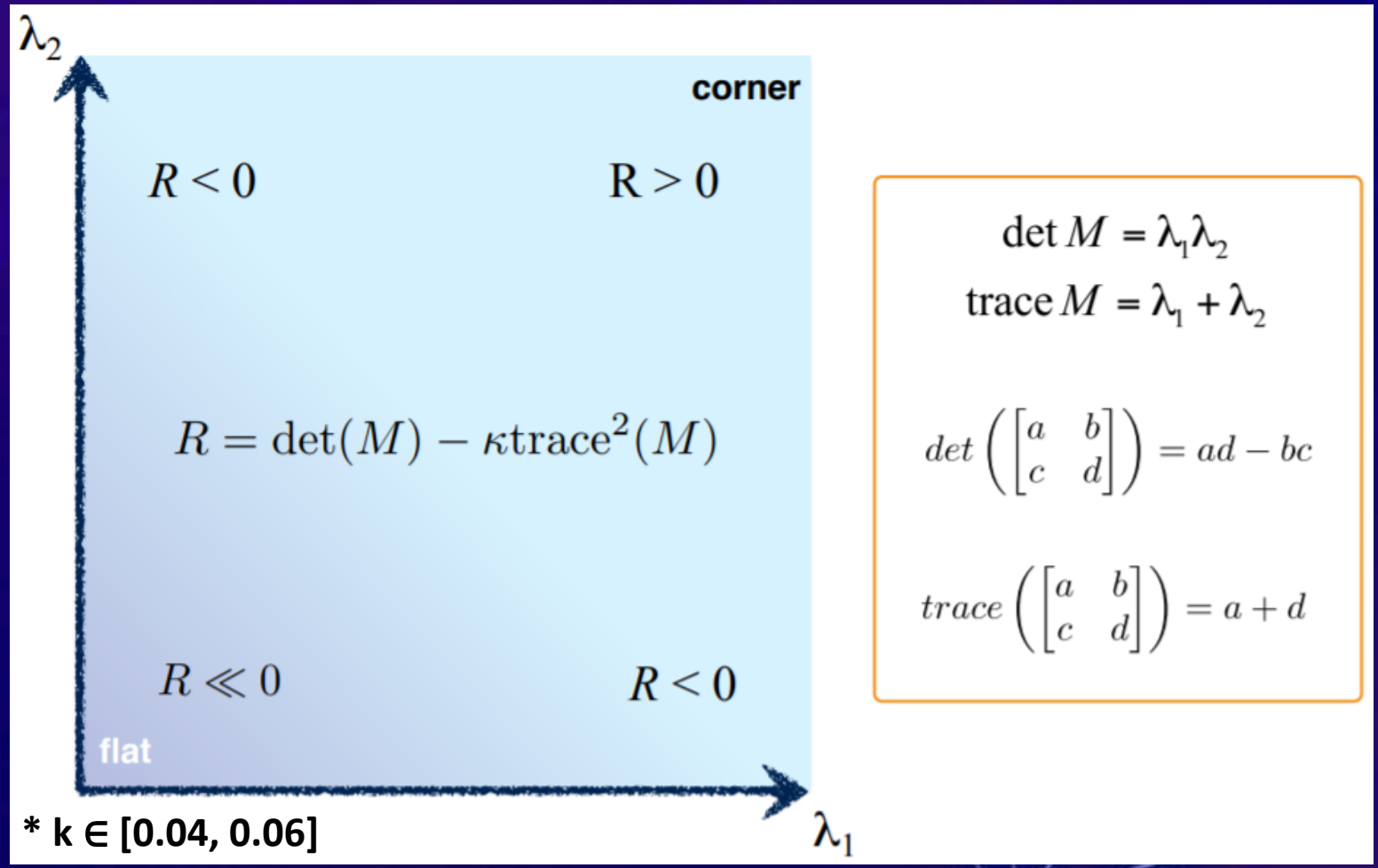
Used to compute the score matrix R.

$$M = \begin{bmatrix} m_{11}(i,j) & m_{12}(i,j) \\ m_{21}(i,j) & m_{22}(i,j) \end{bmatrix}$$

$m_{11} = F \otimes G_x^2$
$m_{12} = F \otimes G_{xy}$
$m_{21} = F \otimes G_{xy}$
$m_{22} = F \otimes G_y^2$

6. For each pixel compute the score R:

R holds a ‘probability’ for each pixel to be classified as a corner.



* k ∈ [0.04, 0.06]

$$\det M = \lambda_1 \lambda_2$$

$$\text{trace}\, M = \lambda_1 + \lambda_2$$

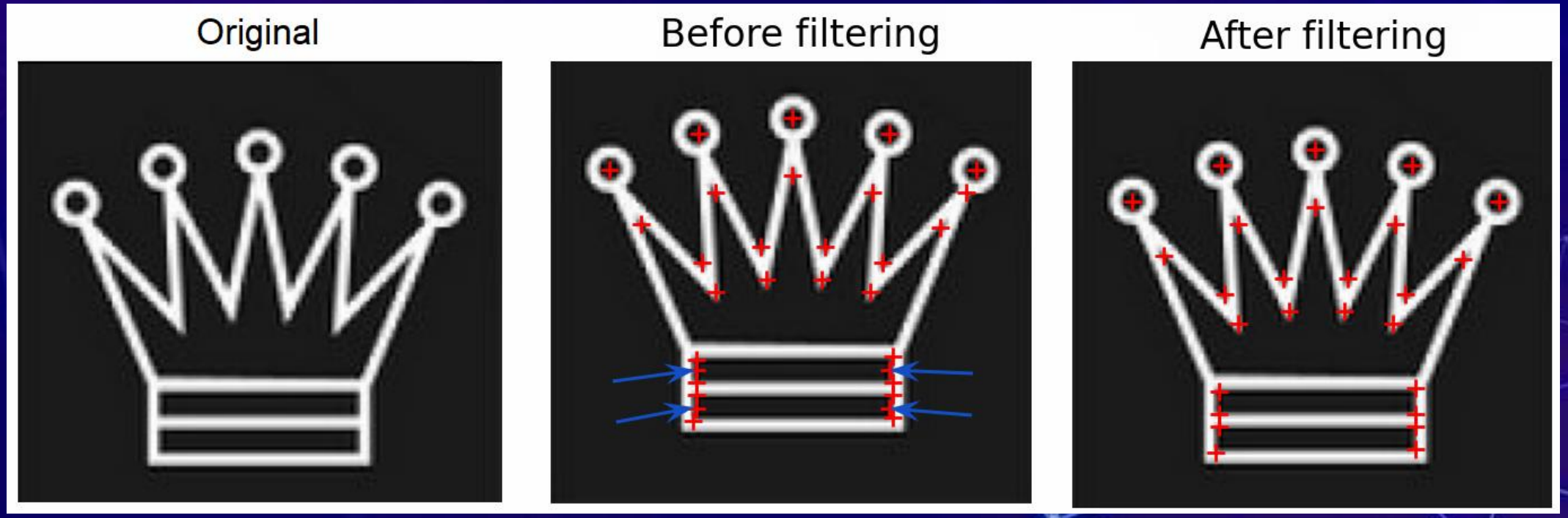
$$det\left(\begin{bmatrix} a & b \\ c & d \end{bmatrix}\right) = ad - bc$$

$$trace\left(\begin{bmatrix} a & b \\ c & d \end{bmatrix}\right) = a + d$$

7. Threshold R and perform NMS (Non-Maxima Suppression):

After setting a threshold, each R that surpasses it and is the local maximum within a radius of 1, will be contended as a corner.

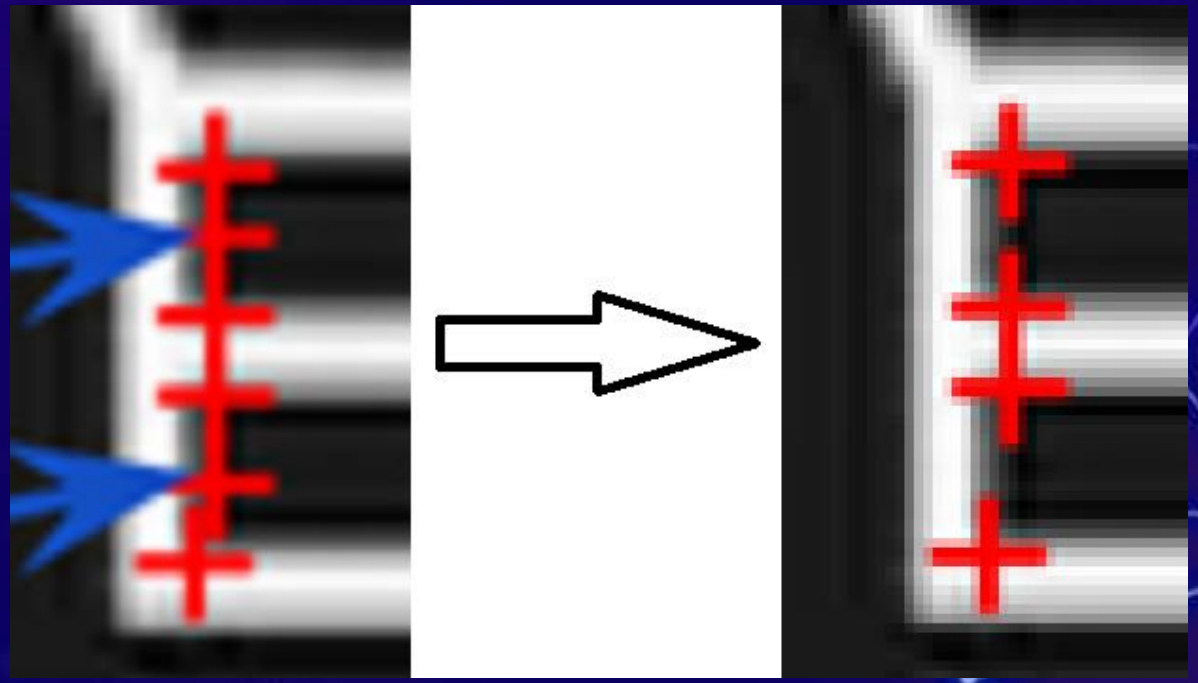
* Originally, each R that fulfills the last requirement would be considered a corner. Our edition to the algorithm adds another step that filters the contended corners, and returns a more accurate result.



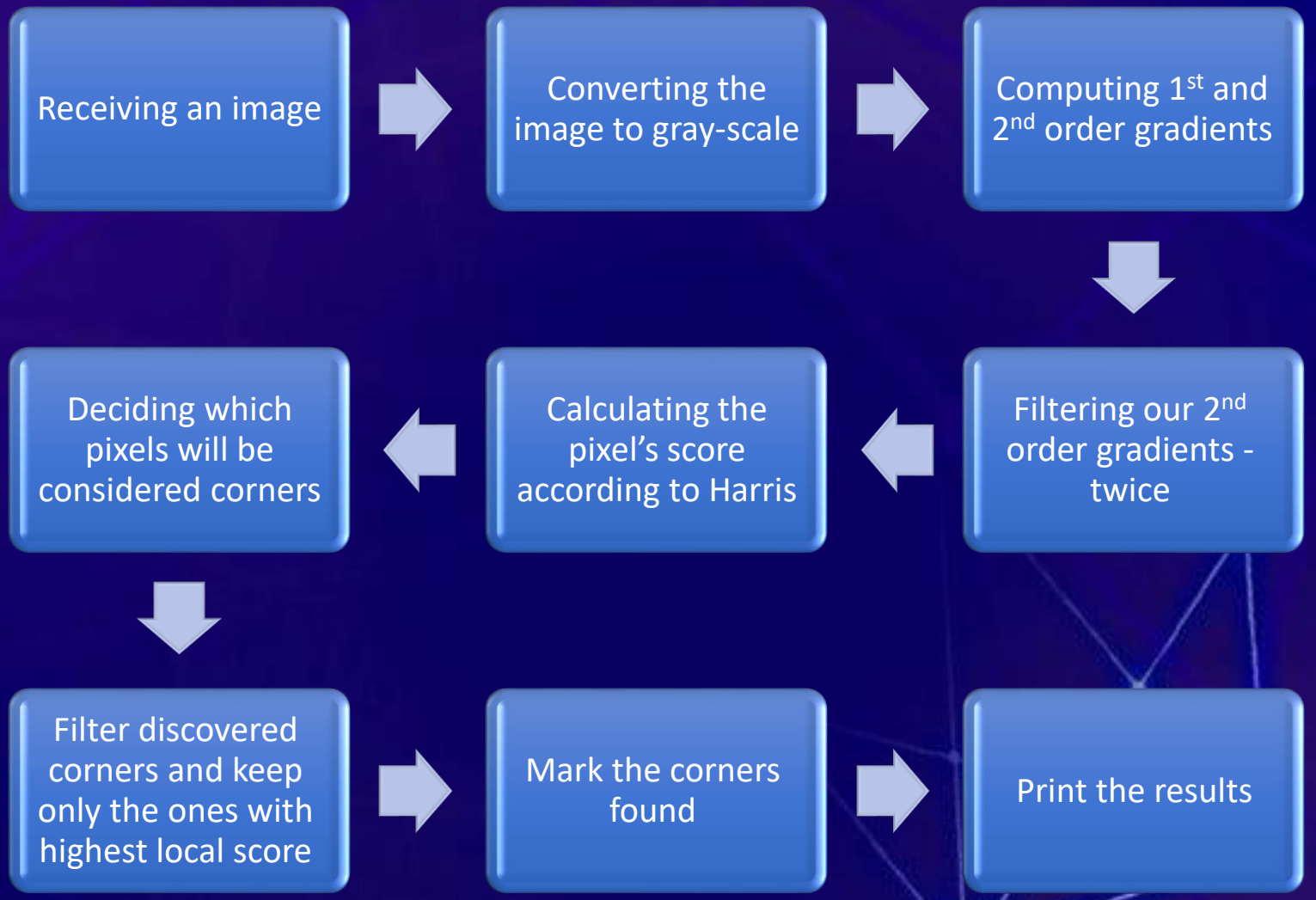
7. Threshold R and perform NMS (Non-Maxima Suppression):

After setting a threshold, each R that surpasses it and is the local maximum within a radius of 1, will be contended as a corner.

* Originally, each R that fulfills the last requirement would be considered a corner. Our edition to the algorithm adds another step that filters the contended corners, and returns a more accurate result.



Process diagram



Our contribution to the algorithm

- Changed the Gaussian mask to Moving Average mask.
- Filtered the 2nd order gradients twice.
- Added a threshold and checked the surrounding indices in radius of 1.
- Added another routine to reduce inaccurate corners.

- By doing all of the above, we improved the original Harris Corner Detection algorithm and got more accurate results.

Pros and Cons of Harris Corner Detection

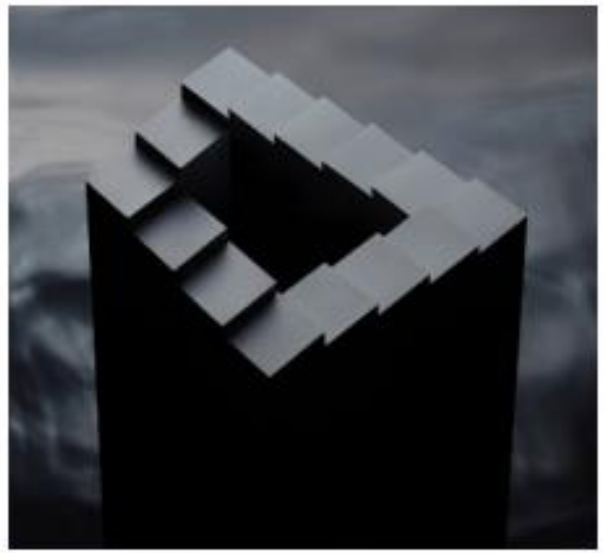
Pros:

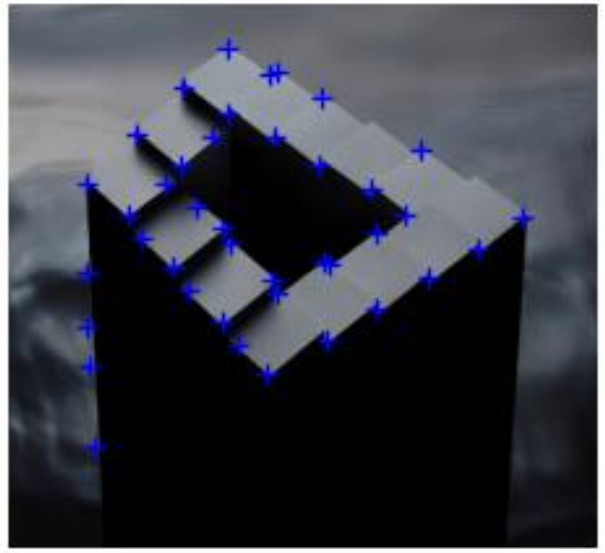
- Uses mathematical equations in order to rate the chances of a pixel being a corner
- Rotation invariance
- Illumination invariance

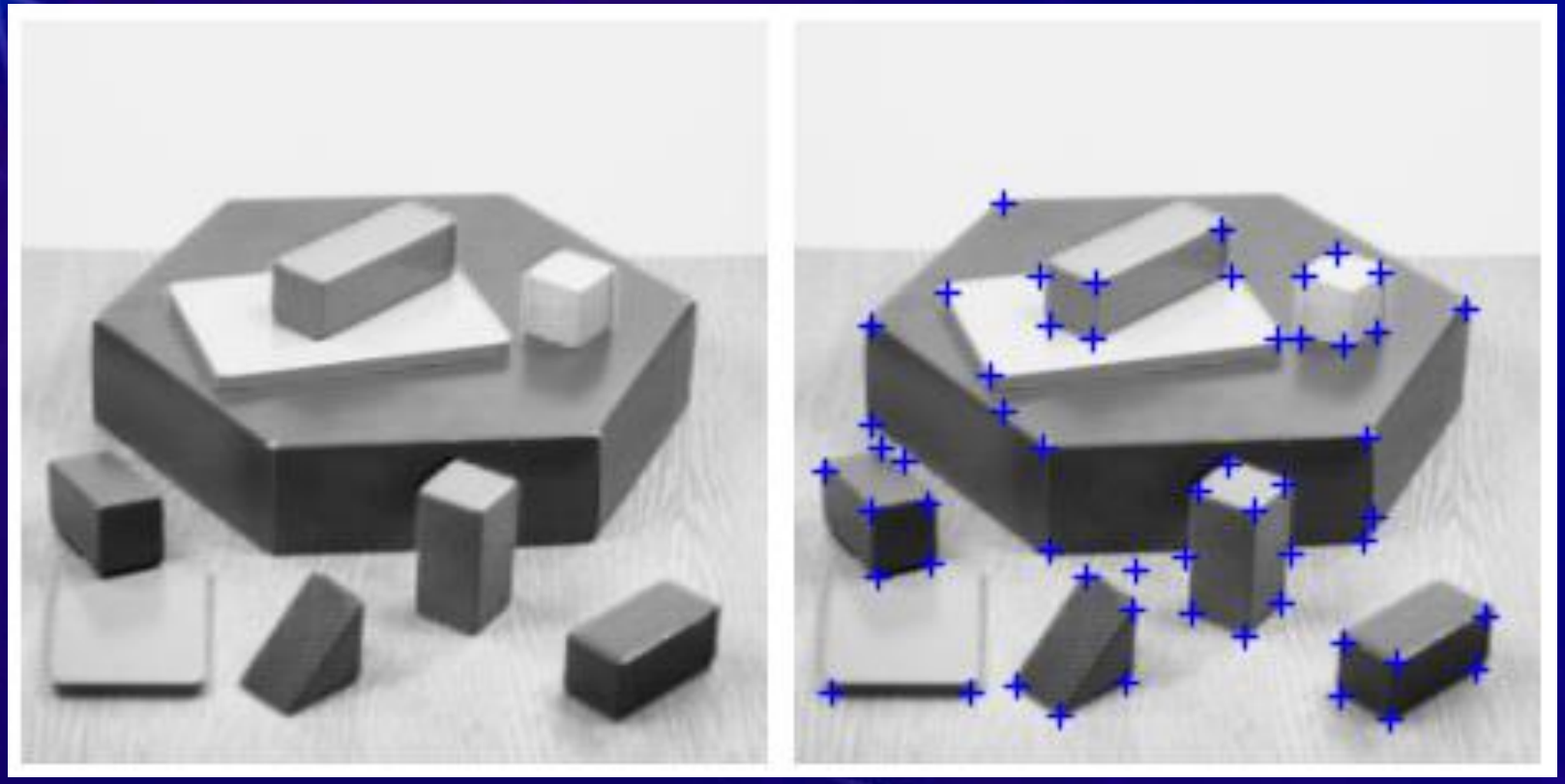
Cons:

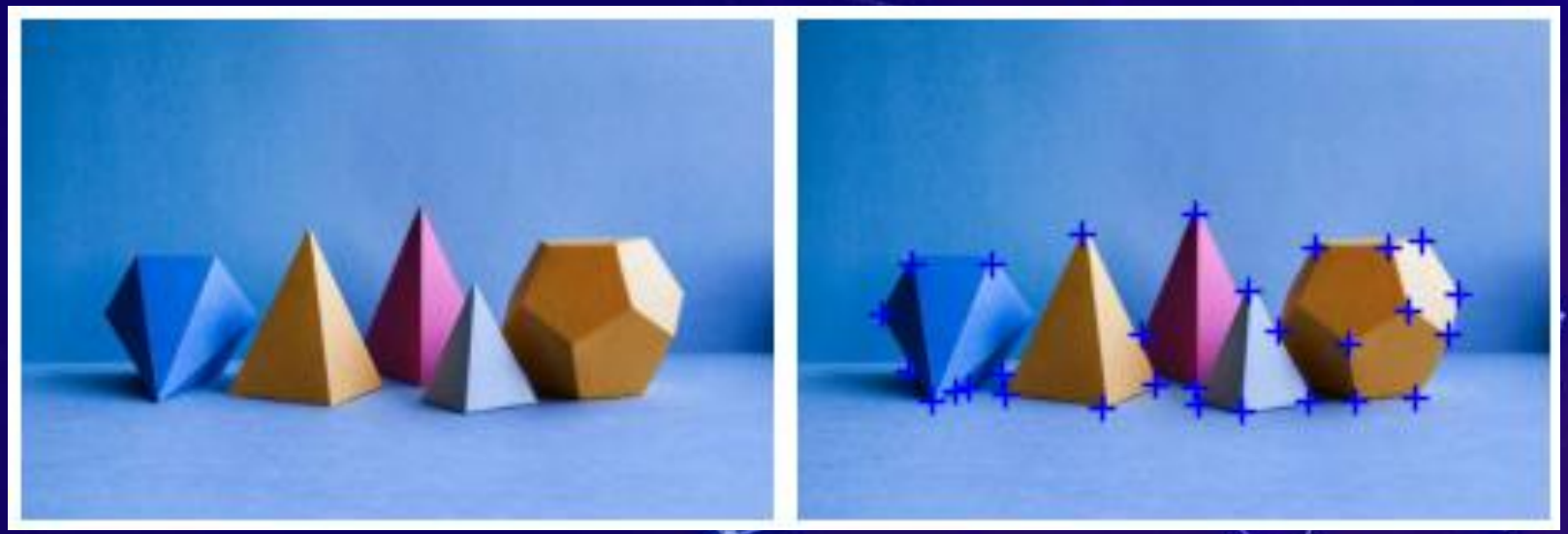
- Having hard time with lower contrast corners
- Noisy textures might affect the accuracy
- Does not always bring the ideal results

Final products:



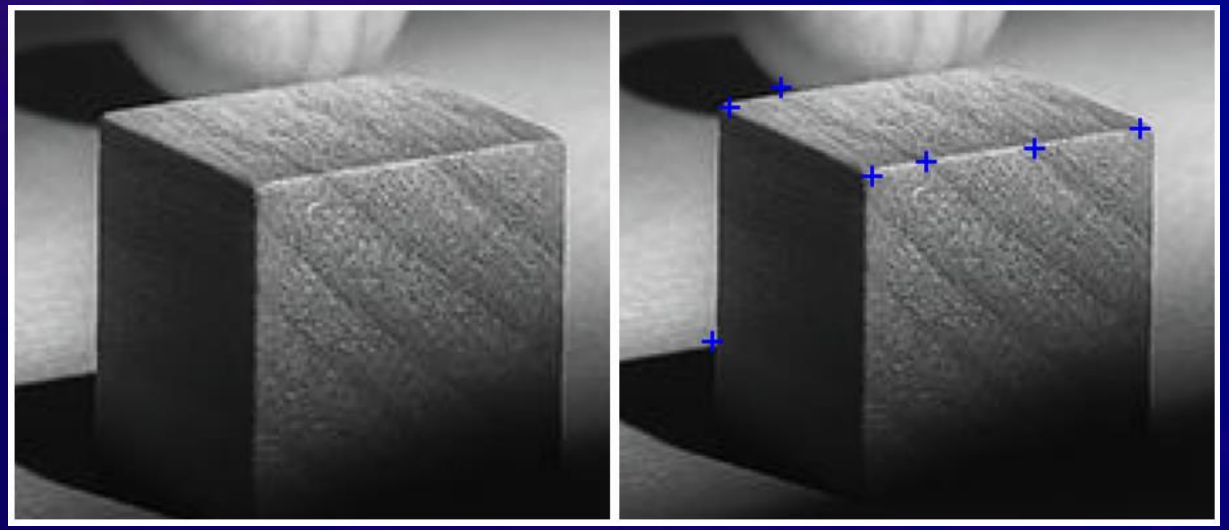




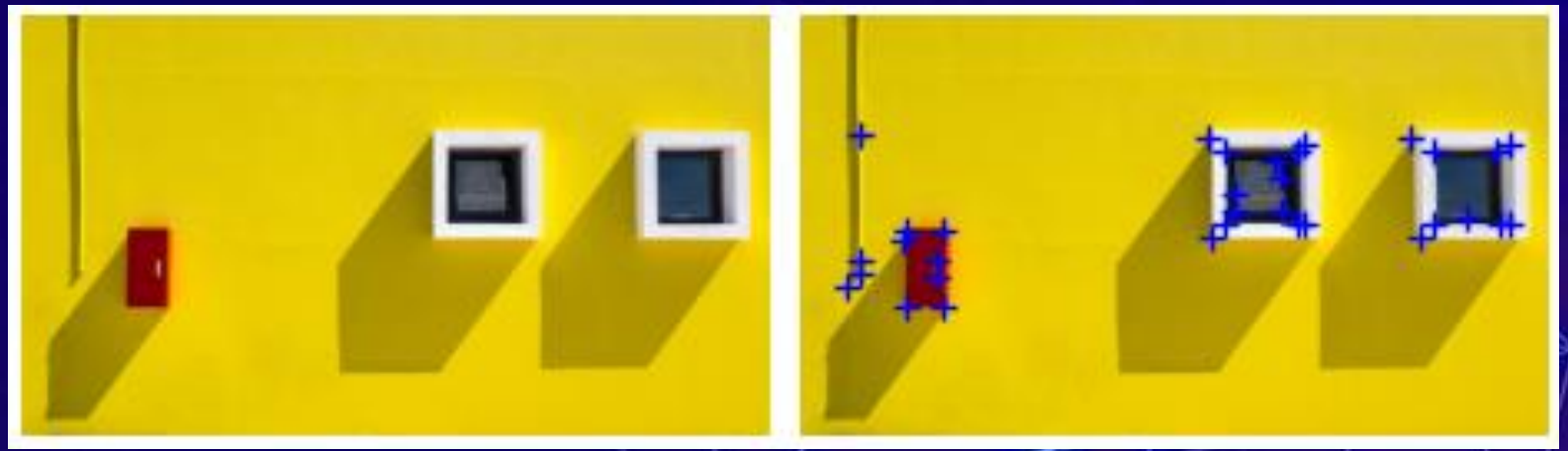


Our limits:

- Noisy texture causes areas with high contrast to be considered as a corner



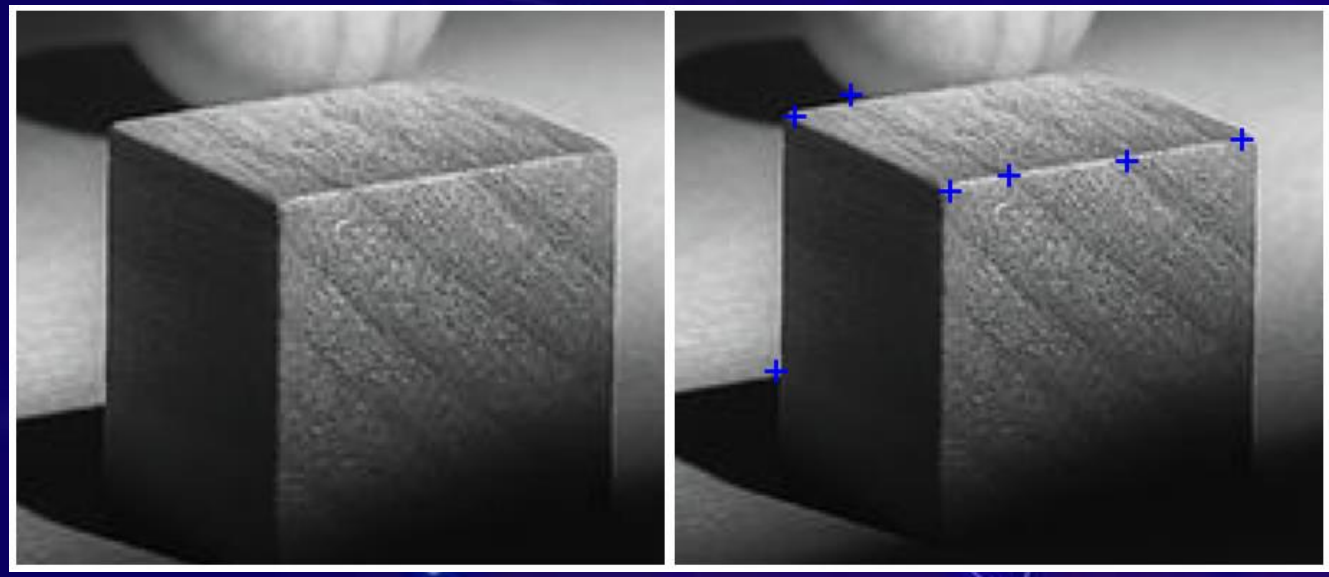
- The contrast in the shaded area is low in relation to the threshold set by the highest score pixel



Future discussions

Our solution to noisy texture problems:

- Using median-filter
- Using K-Means to unite similar colors
- Using histogram equalization



Thank You !