



[Like](#) 3.5K people like this. Be the first of your friends.

Fundamentals of Features and Corners

A lot of computer vision algorithms use features as their backbone. But what exactly is a feature?

Harris Corner Detector

The Harris Corner Detector is a mathematical operator that finds features (what are features? (/tutorials/features/)) in an image. It is simple to compute, and is fast enough to work on computers. Also, it is popular because it is rotation, scale and illumination variation independent. However, the Shi-Tomasi corner detector, the one implemented in OpenCV, is an improvement of this corner detector.

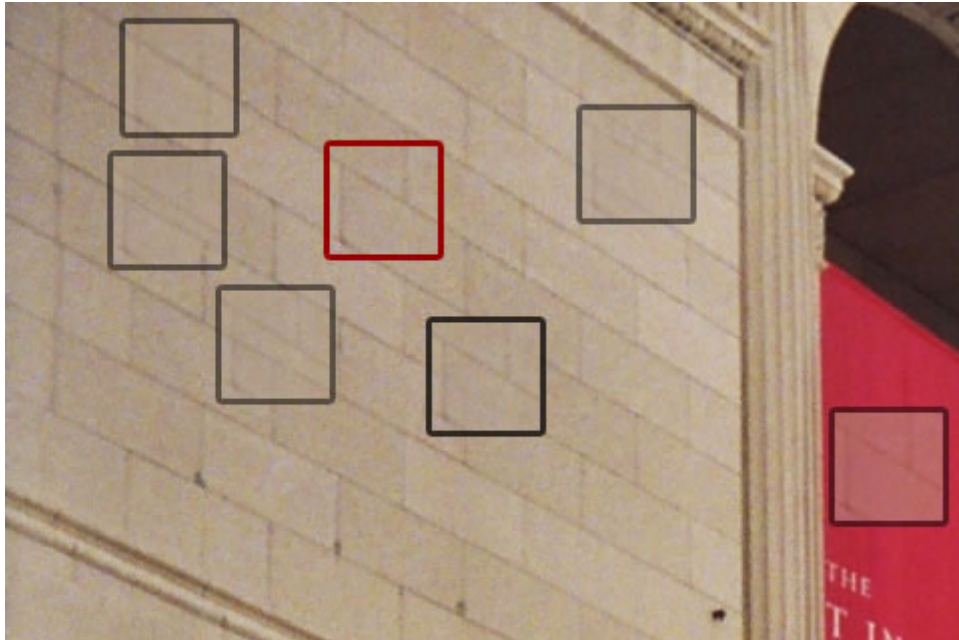
The mathematics

To define the Harris corner detector, we have to go into a bit of math. We'll get into a bit of calculus, some matrix math, but trust me, it won't be tough. I'll make everything easy to understand!

Our aim is to find little patches of image (or "windows") that generate a large variation when moved around. Have a look at this image:

Series: Fundamentals of Features and Corners:

1. Features: What are they? (/tutorials/features/)
2. **Harris Corner Detector**
3. Interesting windows in the Harris Corner Detector (/tutorials/windows-harris-corner-detector/)
4. The Shi-Tomasi Corner Detector (/tutorials/shitomasi-corner-detector/)
5. Subpixel Corners: Increasing accuracy (/tutorials/subpixel-corners-increasing-accuracy/)

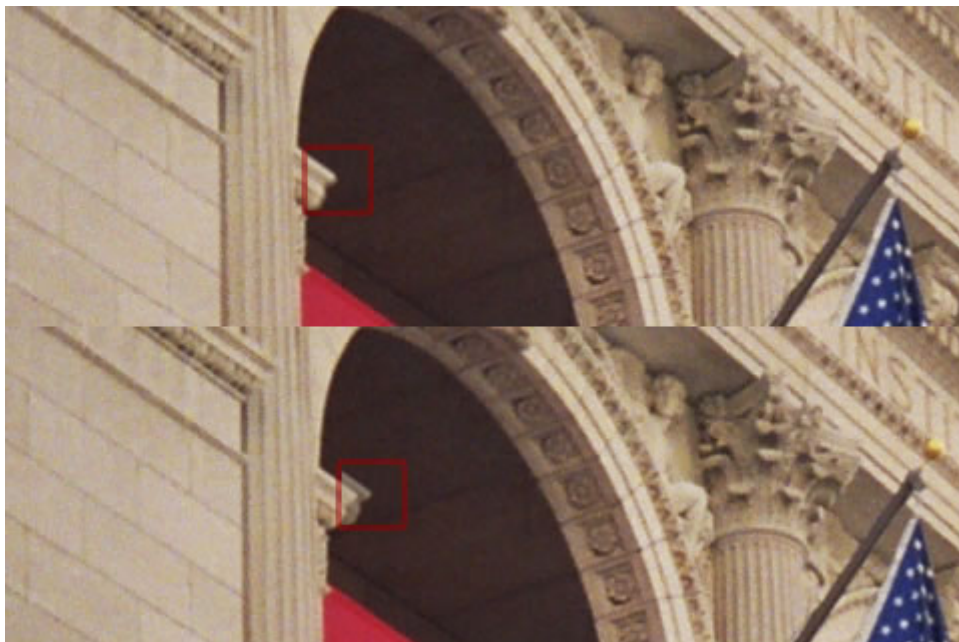


Marked areas have a lot variation

The red square is the window we've chosen. Moving it around doesn't show much of variation. That is, the difference between the window, and the original image below it is very low. So you can't really tell if the window "belongs" to that position.

Of course, if you move the window too much, like onto the reddish region, you're bound to see a big difference. But we've moved the window too much. Not good.

Now have a look at this:



Regions with extremely high variation

See? Even the little movement of the window produces a noticeable difference. This is the kind of window we're looking for. Here's how it translates mathematically:

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

The equation

- E is the difference between the original and the moved window.
- u is the window's displacement in the x direction
- v is the window's displacement in the y direction
- w(x, y) is the window at position (x, y). This acts like a mask. Ensuring that only the desired window is used.
- I is the intensity of the image at a position (x, y)
- I(x+u, y+v) is the intensity of the moved window
- I(x, y) is the intensity of the original

We've looking for windows that produce a large E value. To do that, we need to high values of the terms inside the square brackets.

(Note: There's a little error in these equations. Can you figure it out? Answer below!)

So, we maximize this term:

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

Then, we expand this term using the Taylor series. Whats that? It's just a way of rewriting this term in using its derivatives.

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

See how the I(x+u, y+v) changed into a totally different form (I(x,y)+uI_x + vI_y)? Thats the Taylor series in action. And because the Taylor series is infinite, we've ignored all terms after the first three. It gives a pretty good approximation. But it isn't the actual value.

Next, we expand the square. The I(x,y) cancels out, so its just two terms we need to square. It looks like this:

$$E(u, v) \approx \sum_{x,y} u^2 I_x^2 + 2uv I_x I_y + v^2 I_y^2$$

Now this messy equation can be tucked up into a neat little matrix form like this:

$$E(u, v) \approx [u \quad v] \left(\sum \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \right) \begin{bmatrix} u \\ v \end{bmatrix}$$

See how the entire equation gets converted into a neat little matrix!

(The error: There's no w(x, y) in these errors :P)

Now, we rename the summed-matrix, and put it to be M:

$$M = \sum w(x, y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \text{ So the equation now becomes:}$$

$$E(u, v) \approx \begin{bmatrix} u & v \end{bmatrix} M \begin{bmatrix} u \\ v \end{bmatrix}$$

Looks so neat after all the clutter we did above.

Interesting windows

It was figured out that eigenvalues of the matrix can help determine the suitability of a window. A score, R , is calculated for each window:

$$R = \det M - k(\text{trace } M)^2$$

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

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

All windows that have a score R greater than a certain value are corners. They are good tracking points.

Summary

The Harris Corner Detector is just a mathematical way of determining which windows produce large variations when moved in any direction. With each window, a score R is associated. Based on this score, you can figure out which ones are corners and which ones are not.

OpenCV implements an improved version of this corner detector. It is called the Shi-Tomasi corner detector (</tutorials/shitomasi-corner-detector/>).

More in the series

This tutorial is part of a series called **Fundamentals of Features and Corners**:

1. Features: What are they? (</tutorials/features/>)
2. **Harris Corner Detector**
3. Interesting windows in the Harris Corner Detector (</tutorials/windows-harris-corner-detector/>)
4. The Shi-Tomasi Corner Detector (</tutorials/shitomasi-corner-detector/>)
5. Subpixel Corners: Increasing accuracy (</tutorials/subpixel-corners-increasing-accuracy/>)

Like 3.5K people like this. Be the first of your friends.



(<http://utkarshsinha.com/>)


Utkarsh Sinha created AI Shack in 2010 and has since been working on computer vision and related fields. He is currently at Microsoft working on computer vision.

3 Comments

AI Shack

 Login ▾

 Recommend

 Share

Sort by Best ▾



Join the discussion...

LOG IN WITH

OR SIGN UP WITH DISQUS 

Name

Jannis • 10 months ago

Hey Utkarsh,

Thanks for this great website! Could you explain what I_x and I_y refer to? I am currently implementing SIFT, but I have struggles computing the matrix M to approximate the eigenvalue analysis to discard IPs. Mathematically, I see that I_x and I_y come from Taylor expansion but I have no idea what I_x and I_y mean and how I could infer this from my image I .

Thanks!

^ | ▾ • Reply • Share ›

DonaldAlan • a year ago

The explanation is unclear. "That is, the difference between the window, and the original image below it is very low." I didn't know what "the original window below it" referred to. There are two windows below it. I can see now that the one in the dark square is probably the original window. But you don't mention that anywhere.

^ | ▾ • Reply • Share ›

Utkarsh Sinha Mod  DonaldAlan • a year ago

In that line, the "original image" is the sub-patch marked in red. If you try moving the patch around a bit (a few instances of this are shown by the sub-patches with black borders), the differences aren't that stark. The colors look similar, the brick lines are faint, etc. I hope it makes sense now.

^ | ▾ • Reply • Share ›



About AI Shack

Learn about the latest in AI technology with in-depth tutorials on vision and learning!

Follow [@utkarshsinha](#)

[More... \(/about/\)](#)

Get started

Get started with OpenCV ([/tracks/opencv-basics/](#))

Track a specific color on video ([/tutorials/tracking-colored-objects-opencv/](#))

Learn basic image processing algorithms ([/tracks/image-processing-algorithms-level-1/](#))

How to build artificial neurons? ([/tutorials/single-neuron-dictomizer/](#))

Look at some source code (<http://github.com/aishack/>)



AI Shack

Like Page

3.5K likes

Be the first of your friends to like this



Created by **Utkarsh Sinha** (<http://utkarshsinha.com/>)