SIFT: Finding keypoints

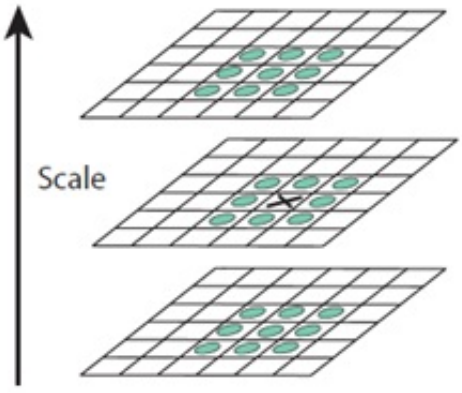
Up till now, we have generated a scale space (/tutorials/sift-scale-invariant-feature-transform-scale-space/) and used the scale space to calculate the Difference of Gaussians (/tutorials/sift-scale-invariant-feature-transform-keypoints/). Those are then used to calculate Laplacian of Gaussian approximations that is scale invariant. I told you that they produce great key points. Here's how it's done!

Finding key points is a two part process

1. Locate maxima/minima in DoG images
2. Find subpixel maxima/minima

Locate maxima/minima in DoG images

The first step is to coarsely locate the maxima and minima. This is simple. You iterate through each pixel and check all it's neighbours. The check is done within the current image, and also the one above and below it. Something like this:



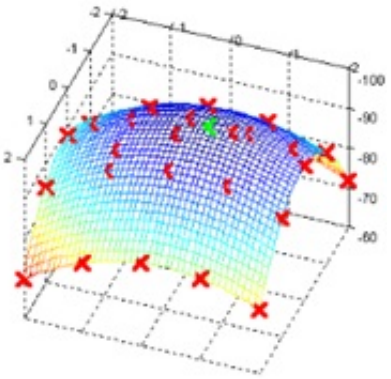
X marks the current pixel. The green circles mark the neighbours. This way, a total of 26 checks are made. **X is marked as a "key point" if it is the greatest or least of all 26 neighbours.**

Usually, a non-maxima or non-minima position won't have to go through all 26 checks. A few initial checks will usually sufficient to discard it.

Note that keypoints are not detected in the lowermost and topmost scales. There simply aren't enough neighbours to do the comparison. So simply skip them!

Once this is done, the marked points are the approximate maxima and minima. They are "approximate" because the maxima/minima almost never lies exactly on a pixel. It lies somewhere between the pixel. But we simply cannot access data "between" pixels. So, we must mathematically locate the subpixel location.

Here's what I mean:



The red crosses mark pixels in the image. But the actual extreme point is the green one.

Find subpixel maxima/minima

Using the available pixel data, subpixel values are generated. This is done by the Taylor expansion of the image around the approximate key point.

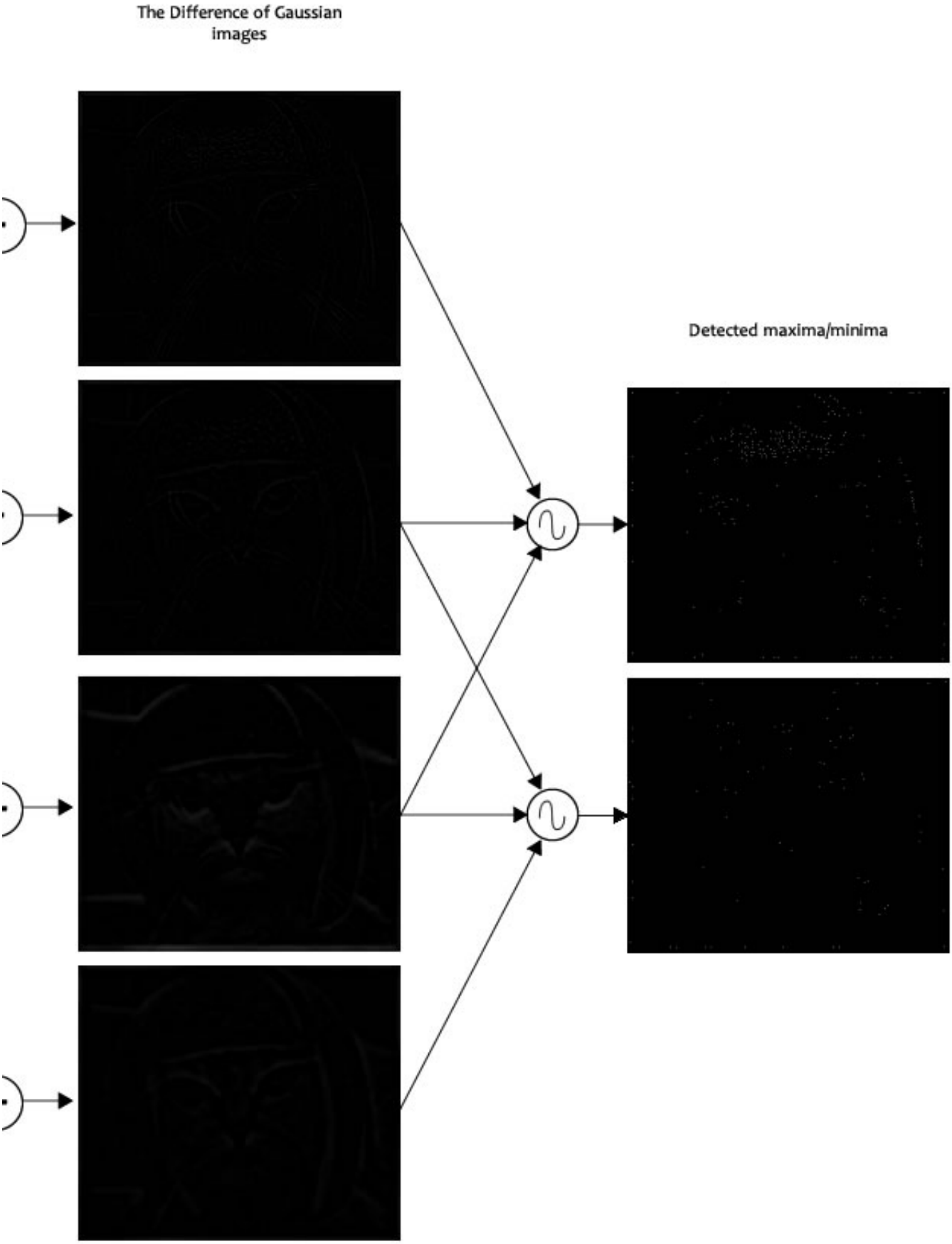
Mathematically, it's like this:

$$D(\mathbf{x}) = D + \frac{\partial D^T}{\partial \mathbf{x}}\mathbf{x} + \frac{1}{2}\mathbf{x}^\mathbf{T}\frac{\partial^2 D}{\partial \mathbf{x}^2}\mathbf{x}$$

We can easily find the extreme points of this equation (differentiate and equate to zero). On solving, we'll get subpixel key point locations. These subpixel values increase chances of matching and stability of the algorithm.

Example

Here's a result I got from the example image I've been using till now:



The

author of SIFT recommends generating two such extrema images. So, you need exactly 4 DoG images. To generate 4 DoG images, you need 5 Gaussian blurred images. Hence the 5 level of blurs in each octave.

In the image, I've shown just one octave. This is done for all octaves. Also, this image just shows the first part of keypoint detection. The Taylor series part has been skipped.

Summary

Here, we detected the maxima and minima in the DoG images generated in the previous step. This is done by comparing neighbouring pixels in the current scale, the scale "above" and the scale "below".

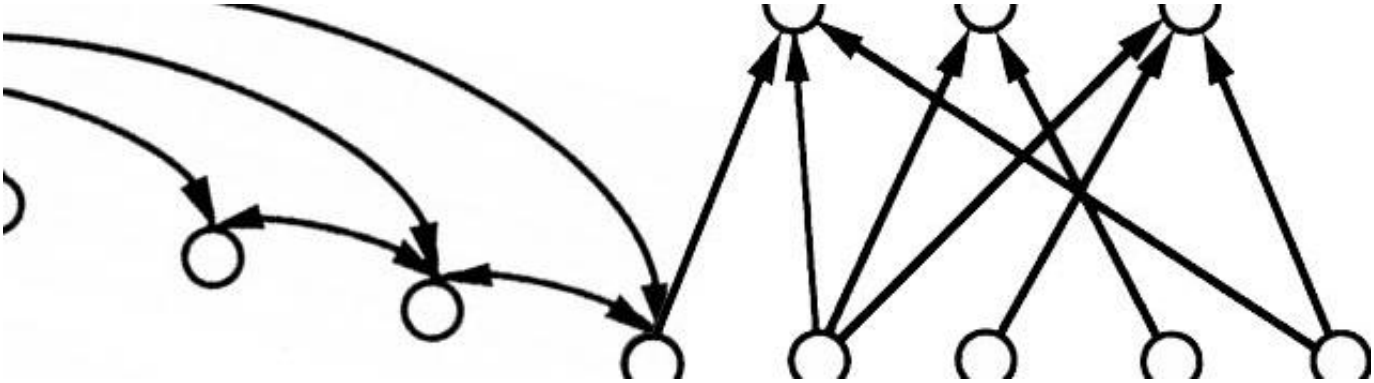
Next, we'll reject some keypoints detected here. This is because they either don't have enough contrast or they lie on an edge

---

This tutorial is part of a series called **SIFT: Theory and Practice**:

1. SIFT: Introduction (/tutorials/sift-scale-invariant-feature-transform-introduction/)
2. SIFT: The scale space (/tutorials/sift-scale-invariant-feature-transform-scale-space/)
3. SIFT: LoG approximations (/tutorials/sift-scale-invariant-feature-transform-log-approximation/)
4. **SIFT: Finding keypoints**
5. SIFT: Getting rid of low contrast keypoints (/tutorials/sift-scale-invariant-feature-transform-eliminate-low-contrast/)
6. SIFT: Keypoint orientations (/tutorials/sift-scale-invariant-feature-transform-keypoint-orientation/)
7. SIFT: Generating a feature (/tutorials/sift-scale-invariant-feature-transform-features/)

Related posts



7 unique neural network architectures (/tutorials/7-unique-neural-network-architectures/)

A single neuron Dictomizer (/tutorials/a-single-neuron-dictomizer/)

Capturing images with DirectX (/tutorials/capturing-images-with-directx/)

(http://utkarshsinha.com/)

Utkarsh Sinha created AI Shack in 2010 and has since been working on computer vision and related fields as a hobby! Currently he works as a writer of software in Bangalore, India.

The bottom sidebar

AI Shack on Facebook

AI Shack

Like

2,756 people like AI Shack.

Facebook social plugin

Contact

You can contact me through the internet over the following channels:

Facebook (http://facebook.com/aishack/)
Twitter (http://twitter.com/aishack)
Github (http://github.com/aishack)
Email (mailto:utkarsh@utkarshsinha.com)

Where to start?

So you want to get started with AI but you're not sure where. Here are some links to get you started:

- Get started with OpenCV (/tracks/opencv-basics/)
- Track a specific color on video (/tutorials/tracking-colored-objects-in-opencv/)
- Learn basic image processing algorithms (/tracks/image-processing-algorithms-level-1/)
- How to build artificial neurons? (/tutorials/a-single-neuron-dictomizer/)
- Look at some source code (http://github.com/aishack/)