SIFT: LoG approximations

In the previous step , we created the scale space of the image (/tutorials/sift-scale-invariant-feature-transform-scale-space/). The idea was to blur an image progressively, shrink it, blur the small image progressively and so on. Now we use those blurred images to generate another set of images, the Difference of Gaussians (DoG). These DoG images are a great for finding out interesting key points in the image.

Laplacian of Gaussian

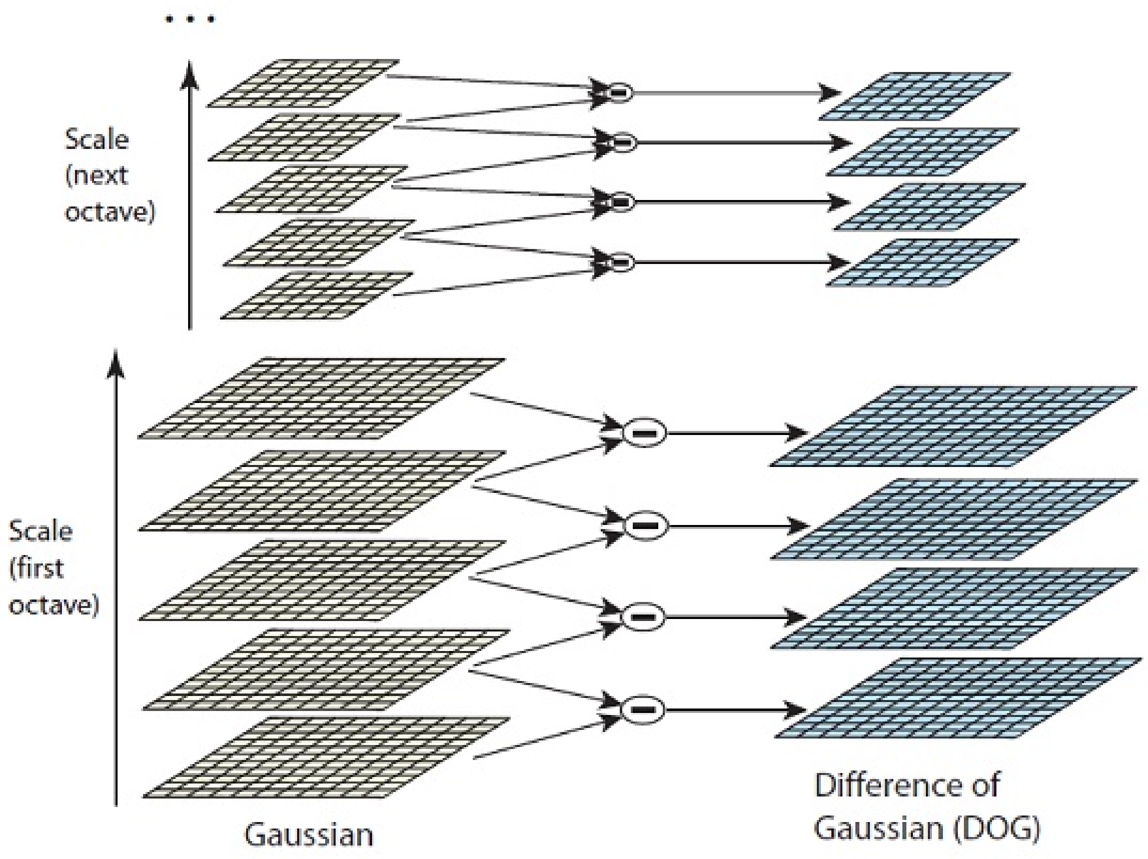
The Laplacian of Gaussian (LoG) operation goes like this. You take an image, and blur it a little. And then, you calculate second order derivatives on it (or, the "laplacian"). This locates edges and corners on the image. These edges and corners are good for finding keypoints.

But the second order derivative is extremely sensitive to noise. The blur smoothes it out the noise and stabilizes the second order derivative.

The problem is, calculating all those second order derivatives is computationally intensive. So we cheat a bit.

The Con

To generate Laplacian of Guassian images quickly, we use the scale space. We calculate the difference between two consecutive scales. Or, the Difference of Gaussians. Here's how:



These Difference of Gaussian images are approximately equivalent to the Laplacian of Gaussian. And we've replaced a computationally intensive process with a simple subtraction (fast and efficient). Awesome!

These DoG images comes with another little goodie. These approximations are also "scale invariant". What does that mean?

The Benefits

Just the Laplacian of Gaussian images aren't great. They are not scale invariant. That is, they depend on the amount of blur you do. This is because of the Gaussian expression. (Don't panic ;))

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2}$$

See the σ^2 in the demonimator? That's the scale. If we somehow get rid of it, we'll have true scale independence. So, if the laplacian of a gaussian is represented like this:

$$\nabla^2 G$$

Then the scale invariant laplacian of gaussian would look like this:

$$\sigma^2 \nabla^2 G$$

But all these complexities are taken care of by the Difference of Gaussian operation. The resultant images after the DoG operation are already multiplied by the σ^2. Great eh!

Oh! And it has also been proved that this scale invariant thingy produces much better trackable points! Even better!

Side effects

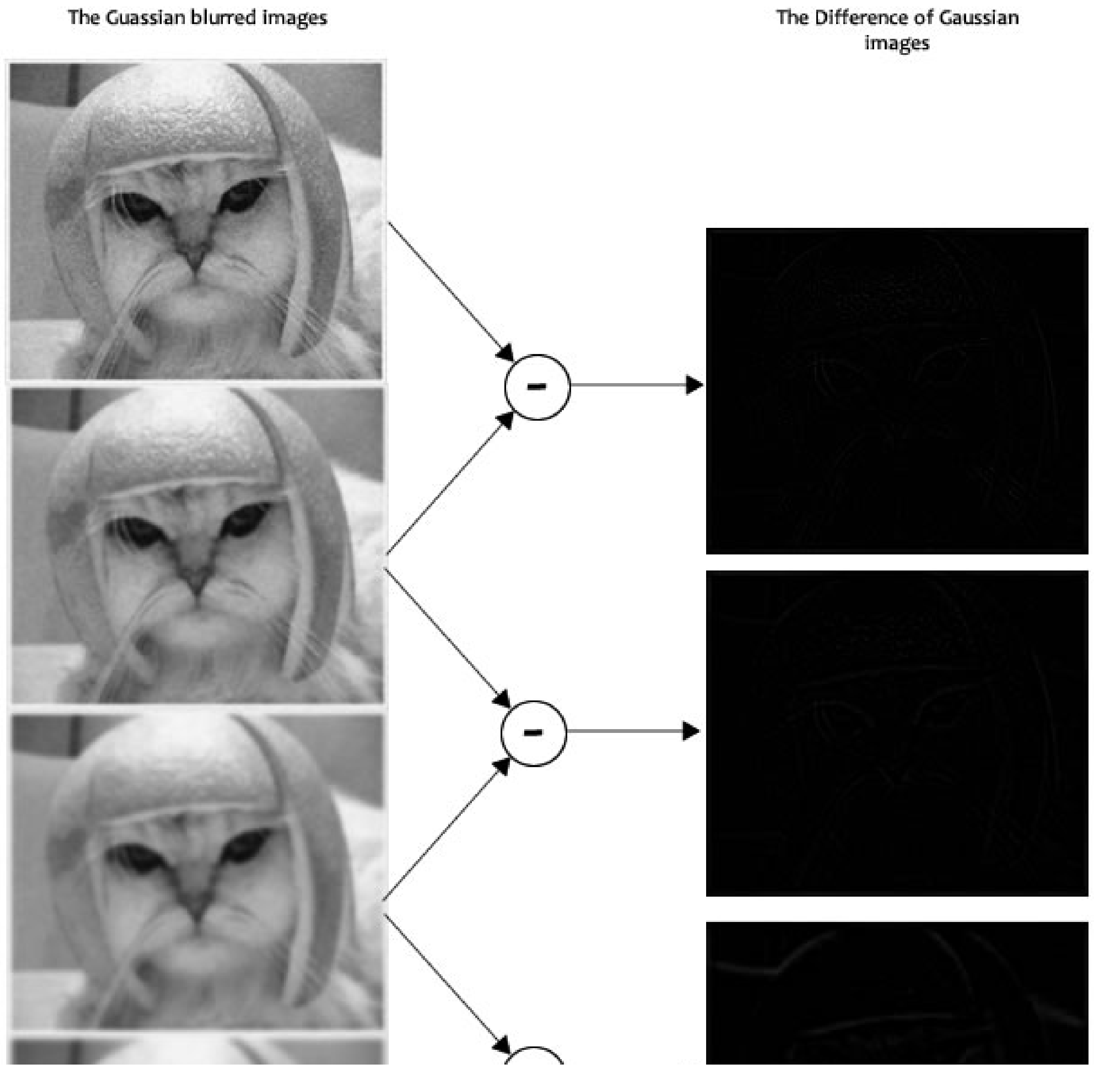
You can't have benefits without side effects >.<

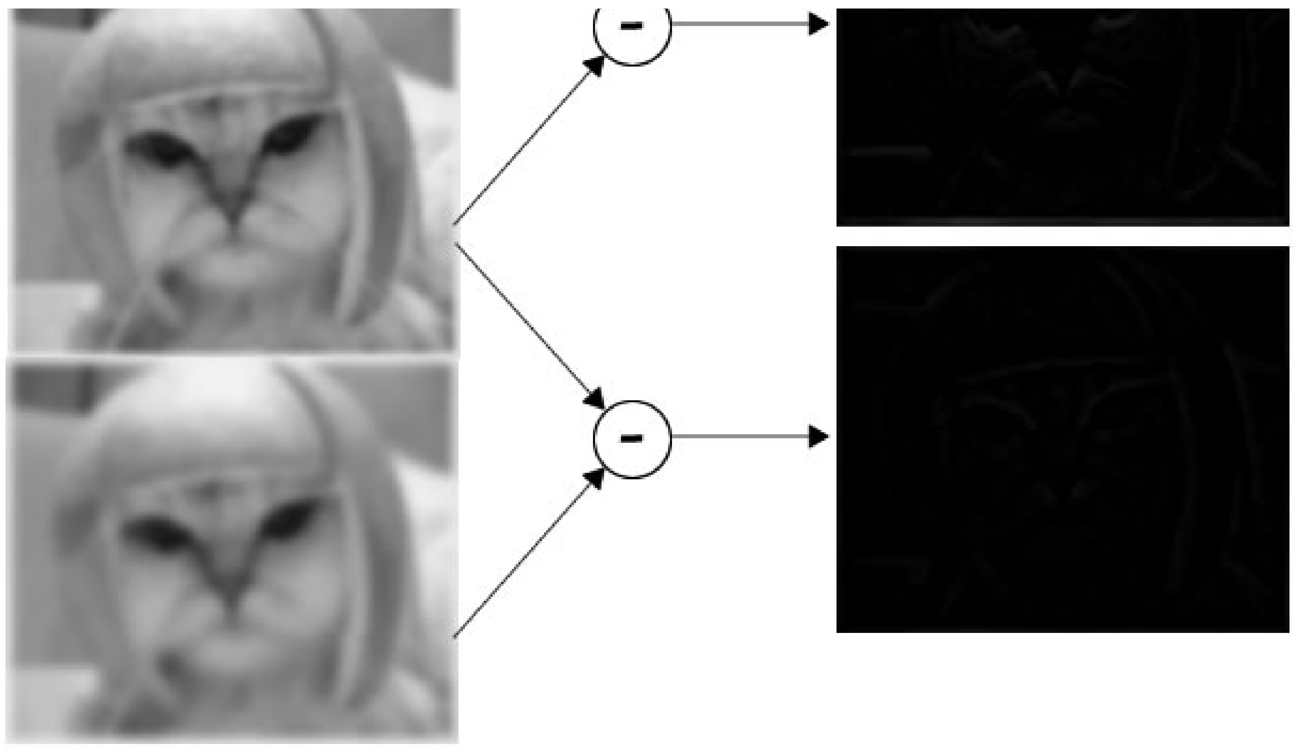
You know the DoG result is multiplied with σ^2. But it's also multiplied by another number. That number is (k-1). This is the k we discussed in the previous step (/tutorials/sift-scale-invariant-feature-transform-scale-space/).

But we'll just be looking for the location of the maximums and minimums in the images. We'll never check the actual values at those locations. So, this additional factor won't be a problem to us. (Even if you multiply throughout by some constant, the maxima and minima stay at the same location)

Example

Here's a gigantic image to demonstrate how this difference of Gaussians works.


The Guassian blurred images
The Difference of Gaussian images
-
-



In the image, I've done the subtraction for just one octave. The same thing is done for all octaves. This generates DoG images of multiple sizes.

Summary

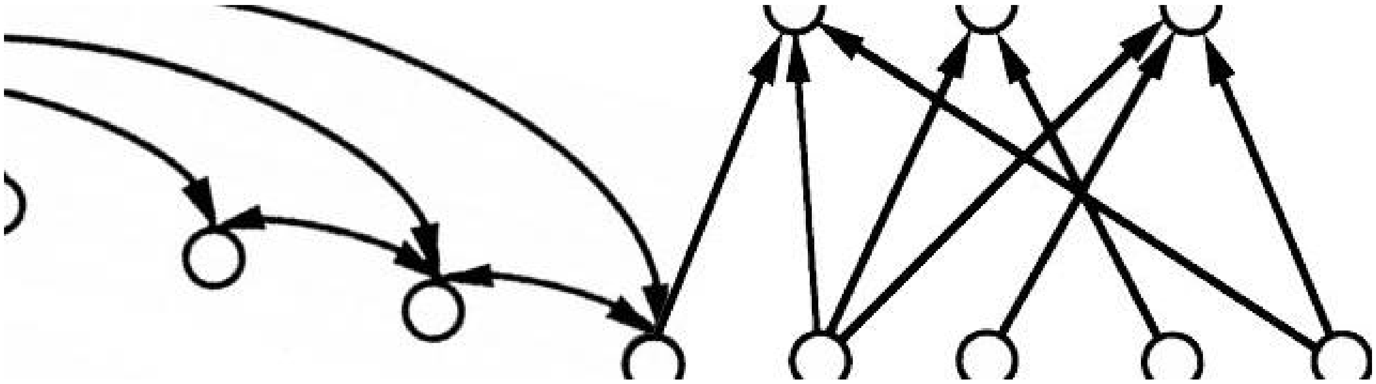
Two consecutive images in an octave are picked and one is subtracted from the other. Then the next consecutive pair is taken, and the process repeats. This is done for all octaves. The resulting images are an approximation of scale invariant laplacian of gaussian (which is good for detecting keypoints). There are a few "drawbacks" due to the approximation, but they won't affect the algorithm.

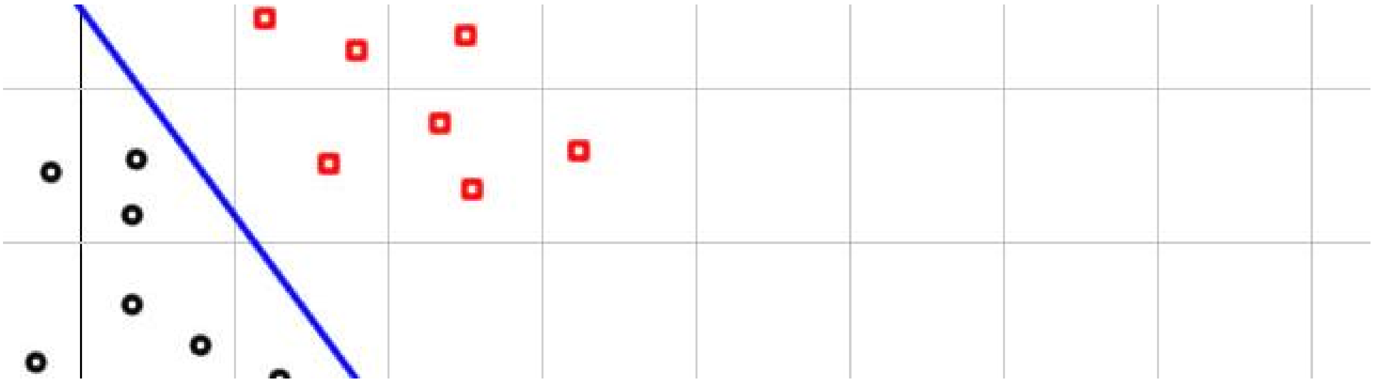
Next, we'll actually find some interesting keypoints. Maxima and Minima. Or, Maximums and Minimums of the image.

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**
4. SIFT: Finding keypoints (/tutorials/sift-scale-invariant-feature-transform-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/)