SIFT: The scale space

Real world objects are meaningful only at a certain scale. You might see a sugar cube perfectly on a table. But if looking at the entire milky way, then it simply does not exist. This multi-scale nature of objects is quite common in nature. And a scale space attempts to replicate this concept on digital images.

Scale spaces

Do you want to look at a leaf or the entire tree? If it's a tree, get rid of some detail from the image (like the leaves, twigs, etc) intentionally.

While getting rid of these details, you must ensure that you do not introduce new false details. The only way to do that is with the Gaussian Blur (it was proved mathematically, under several reasonable assumptions).

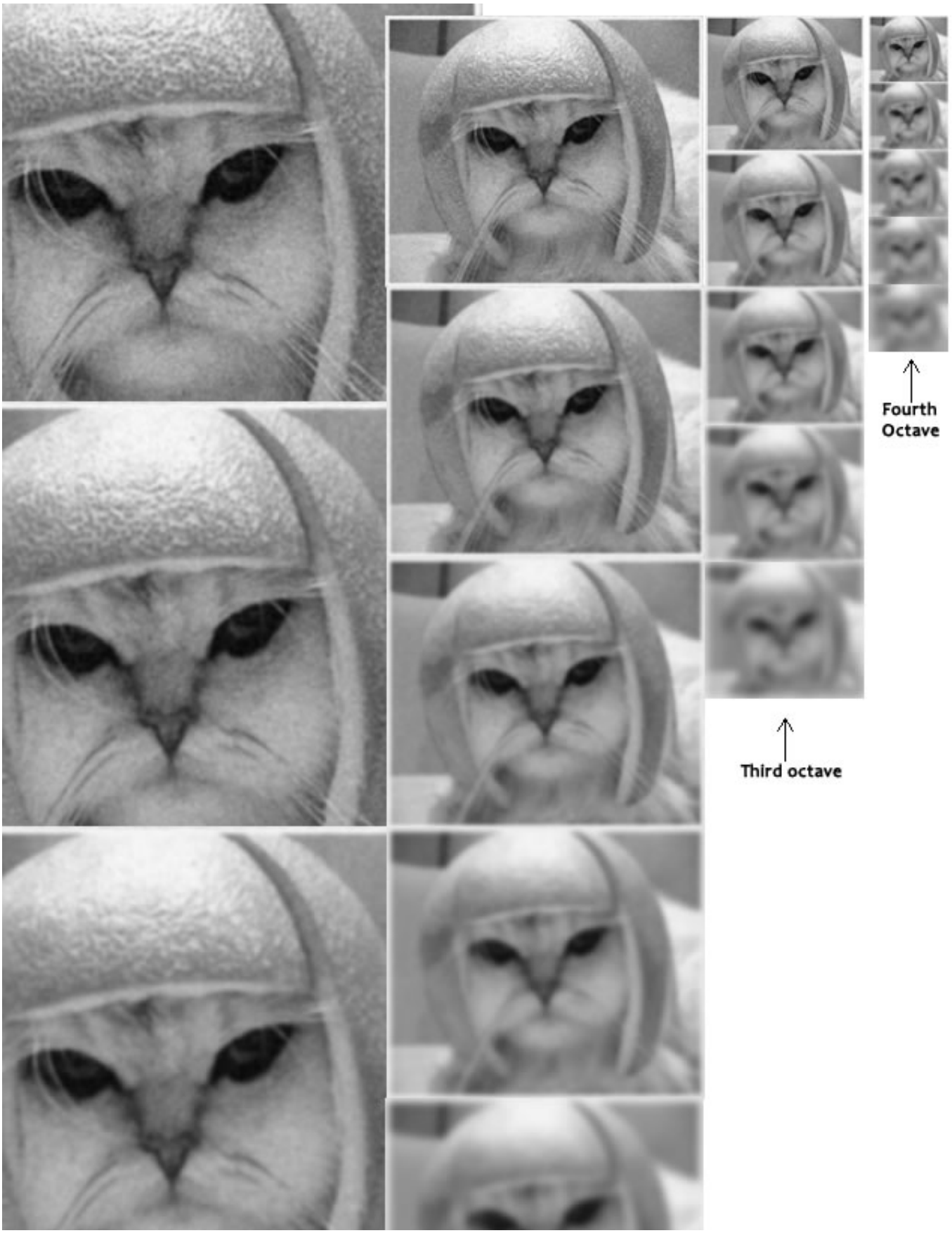
So to create a scale space, you take the original image and generate progressively blurred out images. Here's an example:

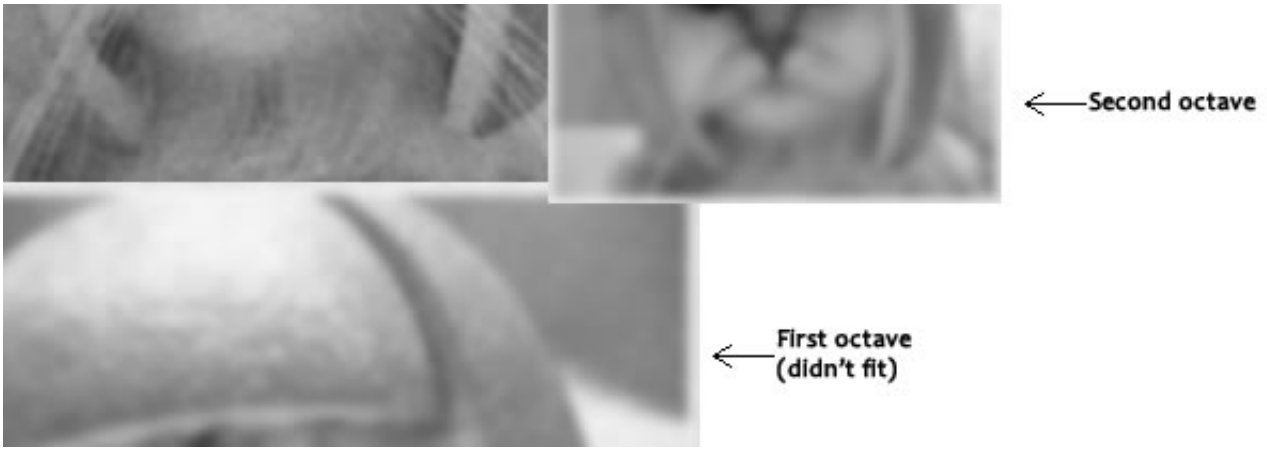
Look at how the cat's helmet loses detail. So do it's whiskers.

Scale spaces in SIFT

SIFT takes scale spaces to the next level. You take the original image, and generate progressively blurred out images. Then, you resize the original image to half size. And you generate blurred out images again. And you keep repeating.

Here's what it would look like in SIFT:





Images of the same size (vertical) form an octave. Above are four octaves. Each octave has 5 images. The individual images are formed because of the increasing "scale" (the amount of blur).

The technical details

Now that you know things the intuitive way, I'll get into a few technical details.

Octaves and Scales

The number of octaves and scale depends on the size of the original image. While programming SIFT, you'll have to decide for yourself how many octaves and scales you want. However, the creator of SIFT suggests that 4 octaves and 5 blur levels are ideal for the algorithm.

The first octave

If the original image is doubled in size and antialiased a bit (by blurring it) then the algorithm produces more four times more keypoints. The more the keypoints, the better!

Blurring

Mathematically, "blurring" is referred to as the convolution of the gaussian operator and the image. Gaussian blur has a particular expression or "operator" that is applied to each pixel. What results is the blurred image.

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y)$$

The symbols:

- L is a blurred image
- G is the Gaussian Blur operator
- I is an image
- x,y are the location coordinates
- σ is the "scale" parameter. Think of it as the amount of blur. Greater the value, greater the blur.
- The * is the convolution operation in x and y. It "applies" gaussian blur G onto the image I.

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

This is the actual Gaussian Blur operator.

Amount of blurring

The amount of blurring in each image is important. It goes like this. Assume the amount of blur in a particular image is σ. Then, the amount of blur in the next image will be k*σ. Here k is whatever constant you choose.

	scale →				
octave	0.707107	1.000000	1.414214	2.000000	2.828427
	1.414214	2.000000	2.828427	4.000000	5.656854
	2.828427	4.000000	5.656854	8.000000	11.313708
	5.656854	8.000000	11.313708	16.000000	22.627417

This is a table of σ's for my current example. See how each σ differs by a factor sqrt(2) from the previous one.

Summary

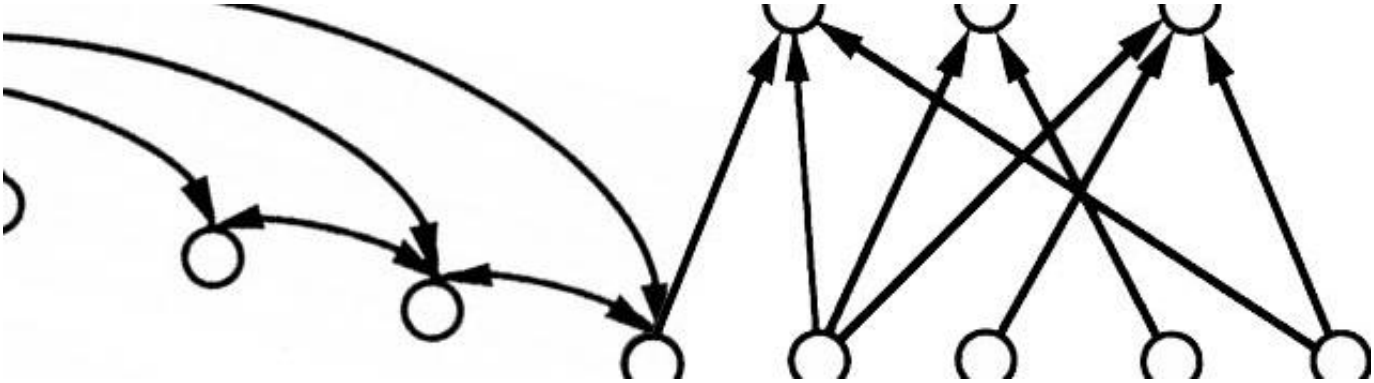
In the first step of SIFT, you generate several octaves of the original image. Each octave's image size is half the previous one. Within an octave, images are progressively blurred using the Gaussian Blur operator.

In the next step, we'll use all these octaves to generate Difference of Gaussian images.

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**
3. SIFT: LoG approximations (/tutorials/sift-scale-invariant-feature-transform-log-approximation/)
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/)