Algorithm for finding similar images

Asked 16 years, 3 months ago Modified 2 years, 6 months ago

Viewed 66k times

93

I need an algorithm that can determine whether two images are 'similar' and recognizes similar patterns of color, brightness, shape etc.. I might need some pointers as to what parameters the human brain uses to 'categorize' images. ..

I have looked at hausdorff based matching but that seems mainly for matching transformed objects and patterns of shape.

algorithm math image-comparison

Share

Improve this question

Follow

edited Aug 12, 2014 at 15:14

Robert Harvey

181k ● 48 ● 346 ● 512

asked Sep 16, 2008 at 19:15

kitsune

11.6k ● 15 ● 59 ● 78

There are some good answers in this other similar question: stackoverflow.com/questions/25977/… – blak Jul 2, 2012 at 20:17

2 a lot of 'coulds' and 'mights'. Anyone try all these suggestions, and know what is best? – john k Jan 27, 2013 at 4:01

It's been 13 years, but I would update here that ML-based image feature vectors can be robust and easily compared with cosine similarity. You could search for img2vec project or something like latentvector.space for an easier API integration (disclaimer: I run that service). – Christian Safka Oct 18, 2021 at 19:27

15 Answers

Sorted by: Highest score (default)

64

I have done something similar, by decomposing images into signatures using wavelet transform.

My approach was to pick the most significant *n* coefficients from each transformed channel, and recording their location. This was done by sorting the list of (power,location) tuples according to abs(power). Similar images will share similarities in that they will have significant coefficients in the same places.

I found it was best to transform in the image into YUV format, which effectively allows you weight similarity in shape (Y channel) and colour (UV channels).

You can in find my implementation of the above in mactorii, which unfortunately I haven't been working on as much as I should have :-)

Another method, which some friends of mine have used with surprisingly good results, is to simply resize your image down to say, a 4x4 pixel and store that as your signature. How similar 2 images are can be scored by say, computing the Manhattan distance between the 2 images, using corresponding pixels. I don't have the details of how they performed the resizing, so you may have to play with the various algorithms available for that task to find one which is suitable.

Share Improve this answer

Follow

edited Dec 10, 2021 at 2:16

WSBT

36.2k ● 18 ● 135 ● 150

answered Sep 17, 2008 at 13:56

freespace

16.7k ● 4 ● 39 ● 59

7 The resize to 4x4 method is a awesome idea (not that your method isn't great too) but the first is simpler. – Alix Axel Oct 23, 2009 at 20:39

@freespace, could you please explain this "computing the Manhattan distance between the 2 images, using corresponding pixels" – Ambika Mar 24, 2016 at 6:24

1 @Ambika: treat the colour of each pixel as a vector of length 3, and compute the Manhattan distance between corresponding pixels in the images being compared. That gives you 4 Manhattan distances. How you derive a single measure from that is up to you. The most obvious is to sum them together. – freespace Mar 25, 2016 at 15:59

48

+50

pHash might interest you.

> perceptual hash n. a fingerprint of an audio, video or image file that is mathematically based on the audio or visual content contained within. Unlike cryptographic hash functions which rely on the avalanche effect of small changes in input leading to drastic changes in the output, perceptual hashes are "close" to one another if the inputs are visually or auditorily similar.

Share Improve this answer

Follow

answered Apr 21, 2009 at 22:23

Alvis

606 ● 6 ● 5

11 Just checked out pHash's website. They currently have this feature on their site that allows you to upload two images and it tells you if they're similar or not. I tried around 10 images that were similar and 10 that were not. Success rate wasn't that impressive, unfortunately. – rodrigo-silveira May 21, 2012 at 1:52

3 pHash is actually pretty strict, you may want to use 'ahash' or average hash, which tends to be less strict. You can find a python implementation here github.com/JohannesBuchner/imagehash/blob/master/… – Rohit Apr 27, 2016 at 0:03

I've used SIFT to re-detect te same object in different images. It is really powerfull but rather complex, and

13

might be overkill. If the images are supposed to be pretty similar some simple parameters based on the difference between the two images can tell you quite a bit. Some pointers:

- Normalize the images i.e. make the average brightness of both images the same by calculating the average brightness of both and scaling the brightest down according to the ration (to avoid clipping at the highest level)) especially if you're more interested in shape than in colour.
- Sum of colour difference over normalized image per channel.
- find edges in the images and measure the distance betwee edge pixels in both images. (for shape)
- Divide the images in a set of discrete regions and compare the average colour of each region.
- Threshold the images at one (or a set of) level(s) and count the number of pixels where the resulting black/white images differ.

Share Improve this answer Follow

edited Apr 12, 2012 at 11:04

answered Sep 17, 2008 at 22:41

jilles de wit

7,138 ● 3 ● 28 ● 50

can you point to the code that use sift-like fetures to compute image similarity? – mrgloom Aug 3, 2012 at 10:40

I'm sorry, I'm sure there is publicly available code, but none that I am aware of. There are some examples on this site. For example here: stackoverflow.com/questions/5461148/… – jilles de wit Aug 6, 2012 at 14:02

The Accord Framework for .Net (accord-framework.net) has some great classes for doing SURF, BagOfVisualWords, Harris Corner Detection, etc with a slew of various kernels and clustering algorithms. – dynamichael Aug 5, 2018 at 5:27

6

My lab needed to solve this problem as well, and we used Tensorflow. Here's a full app implementation for visualizing image similarity.

For a tutorial on vectorizing images for similarity computation, check out this page. Here's the Python (again, see the post for full workflow):

```
from __future__ import absolute_import, division,
print_function

"""

This is a modification of the classify_images.py
script in Tensorflow. The original script produces
string labels for input images (e.g. you input a
picture
of a cat and the script returns the string "cat");
this
modification reads in a directory of images and
generates a vector representation of the image
using
the penultimate layer of neural network weights.
```

```
Usage: python classify_images.py
"../image_dir/*.jpg"

"""

# Copyright 2015 The TensorFlow Authors. All
Rights Reserved.
#
# Licensed under the Apache License, Version 2.0
(the "License");
# you may not use this file except in compliance
with the License.
# You may obtain a copy of the License at
#
#     http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to
in writing, software
# distributed under the License is distributed on
an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND,
```

Share Improve this answer

Follow

answered Mar 28, 2018 at 12:23

duhaime

27.6k ● 21 ● 191 ● 242

Is there a ways to improve this technique? when I add a new file, I have to redo everything to find similarities. is there a way to speed up the process of adding new fiels? @duhaime – mostafa8026 Sep 25, 2021 at 14:05

I appreciate your help with providing some keyword or some resources to read, how to speedup the process of classification of new files. – mostafa8026 Sep 25, 2021 at 14:17

@mostafa8026 any time! We'd certainly be grateful if you'd want to try sending a pull request to the pixplot repo! – duhaime Sep 26, 2021 at 0:49

5

You could use Perceptual Image Diff

It's a command line utility that compares two images using a perceptual metric. That is, it uses a computational model of the human visual system to determine if two images are visually different, so minor changes in pixels are ignored. Plus, it drastically reduces the number of false positives caused by differences in random number generation, OS or machine architecture differences.

Share Improve this answer Follow

edited Sep 16, 2008 at 19:46

answered Sep 16, 2008 at 19:19

Alejandro Bologna

823 ● 6 ● 8

4

It's a difficult problem! It depends on how accurate you need to be, and it depends on what kind of images you are working with. You can use histograms to compare colours, but that obviously doesn't take into account the spatial distribution of those colours within the images (i.e. the shapes). Edge detection followed by some kind of segmentation (i.e. picking out the shapes) can provide a pattern for matching against another image. You can use coocurence matrices to compare textures, by considering the images as matrices of pixel values, and comparing those matrices. There are some good books out there on

image matching and machine vision -- A search on Amazon will find some.

Hope this helps!

Share Improve this answer

Follow

answered Sep 16, 2008 at 19:20

Ben

68.5k ● 38 ● 87 ● 108

3

Some image recognition software solutions are actually not purely algorithm-based, but make use of the **neural network** concept instead. Check out http://en.wikipedia.org/wiki/Artificial_neural_network and namely NeuronDotNet which also includes interesting samples: http://neurondotnet.freehostia.com/index.html

Share Improve this answer

Follow

answered Sep 17, 2008 at 13:54

petr k.

8,100 ● 7 ● 43 ● 52

1 neurondotnet.freehostia.com link is dead – Monica Heddneck May 6, 2017 at 0:53

3

There is related research using Kohonen neural networks/self organizing maps

Both more academic systems (Google for PicSOM) or less academic

(http://www.generation5.org/content/2004/aiSomPic.asp ,

(possibly not suitable for all work enviroments)) presentations exist.

Share Improve this answer

Follow

answered Sep 17, 2008 at 17:54

EPa

41 ● 1

3

Calculating the sum of the squares of the differences of the pixel colour values of a drastically scaled-down version (eg: 6x6 pixels) works nicely. Identical images yield 0, similar images yield small numbers, different images yield big ones.

The other guys above's idea to break into YUV first sounds intriguing - while my idea works great, I want my images to be calculated as "different" so that it yields a correct result - even from the perspective of a colourblind observer.

Share Improve this answer

Follow

edited Nov 3, 2013 at 9:06

zero323

330k ● 108 ● 975 ● 948

answered Oct 14, 2011 at 15:44

chris

31 ● 1

2

This sounds like a vision problem. You might want to look into Adaptive Boosting as well as the Burns Line Extraction algorithm. The concepts in these two should

help with approaching this problem. Edge detection is an even simpler place to start if you're new to vision algorithms, as it explains the basics.

As far as parameters for categorization:

- Color Palette & Location (Gradient calculation, histogram of colors)
- Contained Shapes (Ada. Boosting/Training to detect shapes)

Share Improve this answer

Follow

answered Sep 16, 2008 at 19:30

willasaywhat

2,372 ● 20 ● 23

2

Depending on how much accurate results you need, you can simply break the images in n x n pixels blocks and analyze them. If you get different results in the first block you can't stop processing, resulting in some performance improvements.

For analyzing the squares you can for example get the sum of the color values.

Share Improve this answer

Follow

answered Sep 17, 2008 at 13:47

JValente

You could perform some sort of block-matching motion estimation between the two images and measure the

1

overall sum of residuals and motion vector costs (much like one would do in a video encoder). This would compensate for motion; for bonus points, do affine-transformation motion estimation (compensates for zooms and stretching and similar). You could also do overlapped blocks or optical flow.

Share Improve this answer

Follow

answered Sep 16, 2008 at 19:16

Dark Shikari

8,009 ● 4 ● 28 ● 38

1

As a first pass, you can try using color histograms. However, you really need to narrow down your problem domain. Generic image matching is a very hard problem.

Share Improve this answer

Follow

answered Sep 16, 2008 at 19:19

Dima

39.3k ● 14 ● 78 ● 116

1

Apologies for joining late in the discussion.

We can even use ORB methodology to detect similar features points between two images. Following link gives direct implementation of ORB in python

http://scikit-image.org/docs/dev/auto_examples/plot_orb.html

Even openCV has got direct implementation of ORB. If you more info follow the research article given below.

https://www.researchgate.net/publication/292157133_Image_Matching_Using_SIFT_SURF_BRIEF_and_ORB_Performance_Comparison_for_Distorted_Images

Share Improve this answer

Follow

answered Oct 17, 2016 at 5:39

Vivek Srinivasan

2,887 ● 3 ● 18 ● 18

0

There are some good answers in the other thread on this, but I wonder if something involving a spectral analysis would work? I.e., break the image down to it's phase and amplitude information and compare those. This may avoid some of the issues with cropping, transformation and intensity differences. Anyway, that's just me speculating since this seems like an interesting problem. If you searched http://scholar.google.com I'm sure you could come up with several papers on this.

Share Improve this answer

Follow

answered Sep 16, 2008 at 19:26

neuroguy123

1,365 ● 10 ● 14

spectral analysis is wth Fourier Transform, there isn't a color-histogram since you can reconstruct the image from the two parts --imaginary and real. (don't know if it'll work, just letting you know it isn't in that category). – nlucaroni Sep 16, 2008 at 19:35

Yes, a Fourier Transform is what I meant. – neuroguy123
Sep 16, 2008 at 20:59