Simple and fast method to compare images for similarity

Asked 14 years, 1 month ago Modified 2 years, 9 months ago

Viewed 177k times

236

I need a simple and fast way to compare two images for similarity. I.e. I want to get a high value if they contain exactly the same thing but may have some slightly different background and may be moved / resized by a few pixel.

(More concrete, if that matters: The one picture is an icon and the other picture is a subarea of a screenshot and I want to know if that subarea is exactly the icon or not.)

I have **OpenCV** at hand but I am still not that used to it.

One possibility I thought about so far: Divide both pictures into 10x10 cells and for each of those 100 cells, compare the color histogram. Then I can set some made up threshold value and if the value I get is above that threshold, I assume that they are similar.

I haven't tried it yet how well that works but I guess it would be good enough. The images are already pretty much similar (in my use case), so I can use a pretty high threshold value.

I guess there are dozens of other possible solutions for this which would work more or less (as the task itself is quite simple as I only want to detect similarity if they are really very similar). What would you suggest?

There are a few very related / similar questions about obtaining a signature/fingerprint/hash from an image:

- OpenCV / SURF How to generate a image hash / fingerprint / signature out of the descriptors?
- Image fingerprint to compare similarity of many images
- Near-Duplicate Image Detection
- OpenCV: Fingerprint Image and Compare Against Database.
- more, more, more, more, more, more, more

Also, I stumbled upon these implementations which have such functions to obtain a fingerprint:

- pHash
- imgSeek (GitHub repo) (GPL) based on the paper Fast Multiresolution Image Querying
- image-match. Very similar to what I was searching for. Similar to pHash, based on An image signature for any kind of image, Goldberg et al. Uses Python and Elasticsearch.
- iqdb

- ImageHash. supports pHash.
- Image Deduplicator (imagededup). Supports CNN, PHash, DHash, WHash, AHash.

Some discussions about perceptual image hashes: here

A bit offtopic: There exists many methods to create audio fingerprints. MusicBrainz, a web-service which provides fingerprint-based lookup for songs, has a good overview in their wiki. They are using AcoustID now. This is for finding exact (or mostly exact) matches. For finding similar matches (or if you only have some snippets or high noise), take a look at Echoprint. A related SO question is here. So it seems like this is solved for audio. All these solutions work quite good.

A somewhat more generic question about fuzzy search in general is here. E.g. there is locality-sensitive hashing and nearest neighbor search.

image-processing opencv computer-vision

Share

Improve this question

Follow

edited Oct 5, 2019 at 13:35

asked Nov 16, 2010 at 16:31

Albert

67.9k ● 67 ● 251 ● 400

1 Maybe image fingerprinting could help? stackoverflow.com/questions/596262/… – GWW Nov 16, 2010 at 16:43

1 The Wasserstein metric, also known as Earth Mover's Distance (EMD), is something people seem to not know about, but would give pretty much what you want here. – mmgp Jan 14, 2013 at 14:06

3 possible duplicate of Image comparison - fast algorithm – sashoalm Oct 23, 2013 at 6:23

Hi, I came up with improved dHash -- I called it IDHash: github.com/Nakilon/dhash-vips – Nakilon Nov 10, 2017 at 5:53

Are there any libraries like image-match that are still maintained? – James Parker Mar 26, 2021 at 2:22

9 Answers

Sorted by: Highest score (default)

128

Can the screenshot or icon be transformed (scaled, rotated, skewed ...)? There are quite a few methods on top of my head that could possibly help you:

- **Simple euclidean distance** as mentioned by @carlosdc (doesn't work with transformed images and you need a threshold).
- **(Normalized) Cross Correlation** - a simple metrics which you can use for comparison of image areas. It's more robust than the simple euclidean distance but doesn't work on transformed images and you will again need a threshold.

- **Histogram comparison** - if you use normalized histograms, this method works well and is not affected by affine transforms. The problem is determining the correct threshold. It is also very sensitive to color changes (brightness, contrast etc.). You can combine it with the previous two.
- **Detectors of salient points/areas** - such as MSER (Maximally Stable Extremal Regions), SURF or SIFT. These are very robust algorithms and they might be too complicated for your simple task. Good thing is that you do not have to have an exact area with only one icon, these detectors are powerful enough to find the right match. A nice evaluation of these methods is in this paper: Local invariant feature detectors: a survey.

Most of these are already implemented in OpenCV - see for example the cvMatchTemplate method (uses histogram matching): http://dasl.mem.drexel.edu/~noahKuntz/openCVTut6.html . The salient point/area detectors are also available - see OpenCV Feature Detection.

Share Improve this answer

Follow

answered Nov 17, 2010 at 9:59

Karel Petranek

15.2k ● 4 ● 46 ● 69

1 It can be scaled or moved slightly. Also the background of the icon will be different. I tried histogram comparison but I got many false positives. I also tried euclidean distance but that also gives too many false positives (but maybe I can make that a bit better some handling for the alpha value in the

icon). I will try that a bit further, otherwise I will check out MSER, SURF or SIFT. – Albert Nov 17, 2010 at 15:09

1 Another idea - wouldn't it work if you used histogram comparison of the images after applying a sobel operator? That would only compare similarity of edges. Might or might not work, depending on how "edgy" the background is. – Karel Petranek Nov 17, 2010 at 17:32

Hey, I wonder, with all the recent neural network development, is there a better neural approach nowadays? Maybe also even faster. – Albert Mar 26, 2021 at 9:31

1 I am not aware of any off-the-shelf solution but you can always train a siamese network on a synthetic dataset that contains the transformations you want to treat as "same". That is, you can generate images with noise, slight rotations and shifts and train the network to give the same output for the original and transformed image. – Karel Petranek Mar 26, 2021 at 21:04

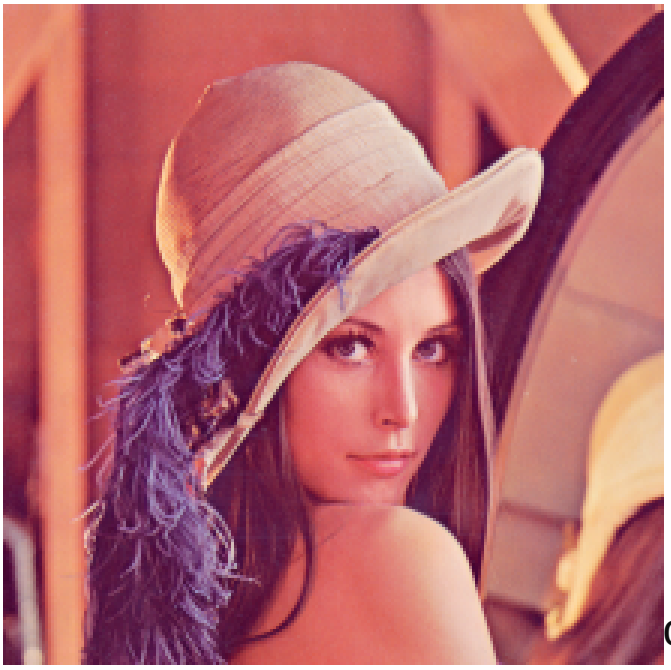
2 It'd be good to also mention opencv's `structural_similarity` function, which is an SSIM implementation, and pretty much exactly what you need to flag "diffs". Which can then be paired with `findContours` to get the bounding boxes around regions with diffs, and then further built out using template matching to see if any of the resulting bboxes represents modified content vs. relocated content. – Mike 'Pomax' Kamermans May 6, 2021 at 16:14

58

I face the same issues recently, to solve this problem(simple and fast algorithm to compare two images) once and for all, I contribute an img_hash module to opencv_contrib, you can find the details from this link.

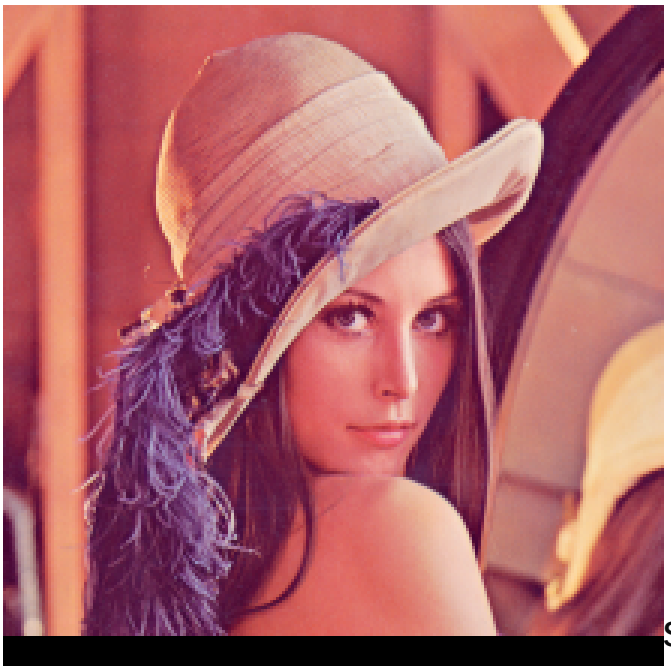
img_hash module provide six image hash algorithms, quite easy to use.

Codes example


origin lena


blur lena


resize lena

 shift lena

```
#include <opencv2/core.hpp>
#include <opencv2/core/ocl.hpp>
#include <opencv2/highgui.hpp>
#include <opencv2/img_hash.hpp>
#include <opencv2/imgproc.hpp>

#include <iostream>

void compute(cv::Ptr<cv::img_hash::ImgHashBase>
algo)
{
    auto input = cv::imread("lena.png");
    cv::Mat similar_img;

    //detect similiar image after blur attack
    cv::GaussianBlur(input, similar_img, {7,7}, 2,
2);
    cv::imwrite("lena_blur.png", similar_img);
    cv::Mat hash_input, hash_similar;
    algo->compute(input, hash_input);
    algo->compute(similar_img, hash_similar);
    std::cout<<"gaussian blur attack : "<<
               algo->compare(hash_input,
hash_similar)<<std::endl;

    //detect similar image after shift attack
    similar_img.setTo(0);
    input(cv::Rect(0,10, input.cols,input.rows-
```

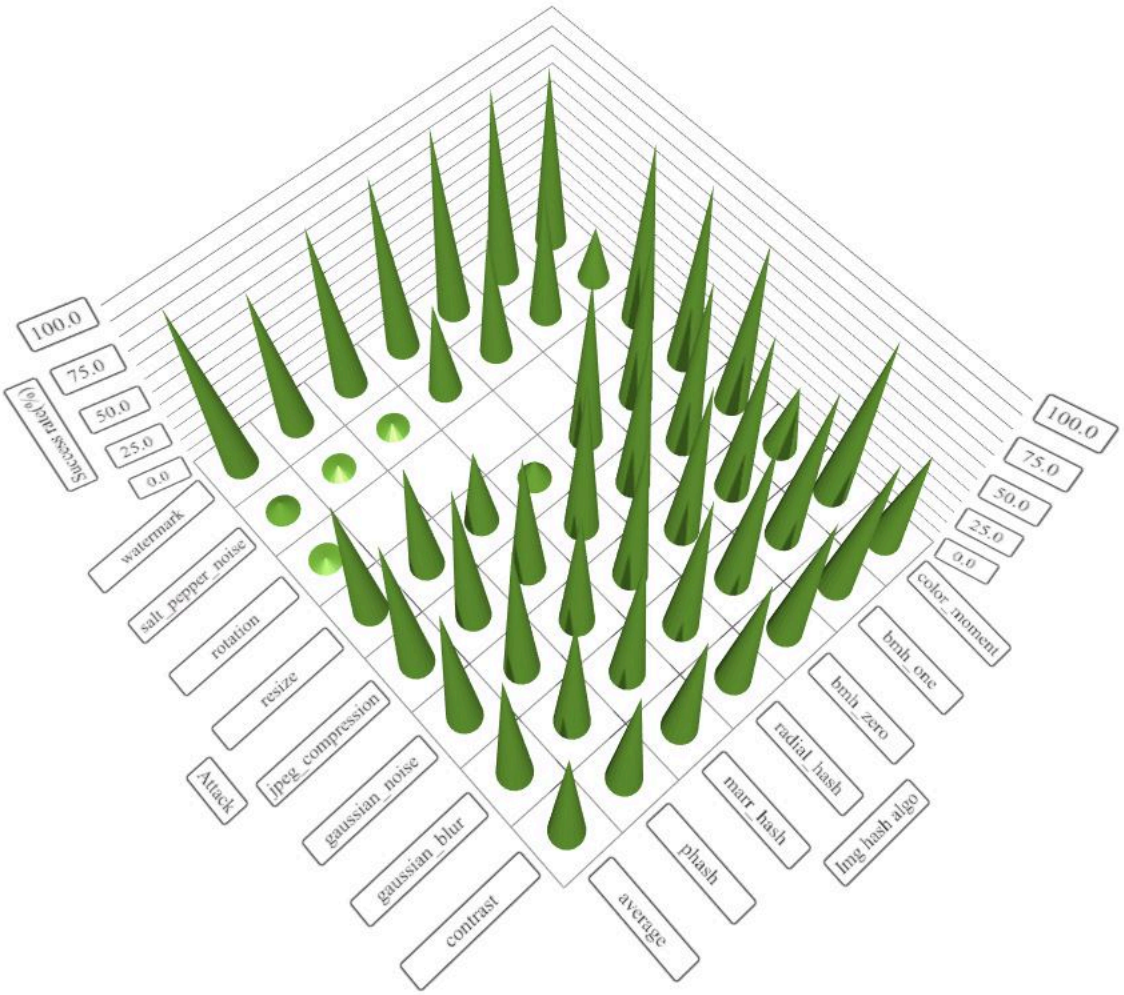
```
10)).

copyTo(similar_img(cv::Rect(0,0,input.cols,input.row
10)));
    cv::imwrite("lena_shift.png", similar_img);
    algo->compute(similar_img, hash_similar);
    std::cout<<"shift attack : "<<
               algo->compare(hash_input,
hash_similar)<<std::endl;
```

In this case, ColorMomentHash give us best result

- gaussian blur attack : 0.567521
- shift attack : 0.229728
- resize attack : 0.229358

Pros and cons of each algorithm



The performance of img_hash is good too

Speed comparison with PHash library(100 images from ukbench)

	Average hash	PHash	Marr hash	Radial hash	BMH zero	BMH one	Color hash
opencv(us)	739	3927	787125	180618	75263	97342	424210
PHash(us)			10993897	876526	280651	322545	
PHash/opencv			13.96715515	4.852927172	3.728937194	3.313523453	

Times	Average hash	PHash	Marr hash	Radial hash	BMH zero	BMH one	Color hash
opencv(us)	30	31	40	502	42	45	74
PHash(us)			68	688	42	134	
			1.7	1.370517928	1	2.977777778	

If you want to know the recommend thresholds for these algorithms, please check this post(http://qtandopencv.blogspot.my/2016/06/introduction-to-image-hash-module-of.html). If you are interesting about how do I measure the performance of img_hash modules(include speed and different attacks), please check this link(http://qtandopencv.blogspot.my/2016/06/speed-up-image-hashing-of-opencvimghash.html).

Share Improve this answer Follow

edited Jun 27, 2017 at 14:25

answered Jul 3, 2016 at 8:09

StereoMatching
5,019 ● 7 ● 42 ● 72

Great work! In your experience, which is the best and fast algorithm for shift attack with different aspect ratios?
– Luis A. Florit Mar 28, 2021 at 13:11

11

Does the screenshot contain only the icon? If so, the L2 distance of the two images might suffice. If the L2 distance doesn't work, the next step is to try something simple and well established, like: Lucas-Kanade. Which I'm sure is available in OpenCV.

Share Improve this answer Follow

edited Nov 16, 2010 at 23:19

answered Nov 16, 2010 at 21:55

carlosdc

12.1k ● 4 ● 47 ● 64

> The subarea contains either exactly only the icon (with some random background) or something different. I want to see which case it is. Though, it may be very slightly shifted or resized, that's why I was not sure if I could just look at the distance (in whatever norm). But I will try with a scaled down version. – Albert Nov 16, 2010 at 22:23

9

If you want to get an index about the similarity of the two pictures, I suggest you from the metrics the SSIM index. It is more consistent with the human eye. Here is an article about it: Structural Similarity Index

It is implemented in OpenCV too, and it can be accelerated with GPU: OpenCV SSIM with GPU

Share Improve this answer Follow

answered Sep 12, 2013 at 16:49

Milan Tenk
2,673 ● 1 ● 19 ● 32

the link for the article is broken. Could you update it or provide more recent publication? Thanks – bit_scientist Nov 8, 2022 at 0:16

5

If you can be sure to have precise alignment of your template (the icon) to the testing region, then any old sum of pixel differences will work.

If the alignment is only going to be a tiny bit off, then you can low-pass both images with cv::GaussianBlur before finding the sum of pixel differences.

If the quality of the alignment is potentially poor then I would recommend either a Histogram of Oriented Gradients or one of OpenCV's convenient keypoint detection/descriptor algorithms (such as SIFT or SURF).

Share Improve this answer

Follow

answered Nov 17, 2010 at 23:28

rcv
6,288 ● 10 ● 45 ● 64

4

If for matching identical images - code for L2 distance

```
// Compare two images by getting the L2 error
(square-root of sum of squared error).
double getSimilarity( const Mat A, const Mat B ) {
if ( A.rows > 0 && A.rows == B.rows && A.cols > 0
&& A.cols == B.cols ) {
    // Calculate the L2 relative error between
```

```
images.
    double errorL2 = norm( A, B, CV_L2 );
    // Convert to a reasonable scale, since L2
error is summed across all pixels of the image.
    double similarity = errorL2 / (double)( A.rows
* A.cols );
    return similarity;
}
else {
    //Images have a different size
    return 100000000.0;  // Return a bad value
}
```

Fast. But not robust to changes in lighting/viewpoint etc.
[Source]

Share Improve this answer

Follow

edited Apr 4, 2014 at 8:28

answered Apr 4, 2014 at 8:23

kiranpradeep

11.2k ● 4 ● 53 ● 82

1

If you want to compare image for similarity,I suggest you to used OpenCV. In OpenCV, there are few feature matching and template matching. For feature matching, there are SURF, SIFT, FAST and so on detector. You can use this to detect, describe and then match the image. After that, you can use the specific index to find number of match between the two images.

Share Improve this answer

Follow

answered Jan 19, 2013 at 5:17

Hua Er Lim

231 ● 2 ● 4 ● 11

1 you said "After that, you can use the specific index to find number of match between the two images." what can be the minimum number of matches between the two images to say that they "contais" the same object? – Inês Martins Nov 3, 2015 at 11:44

0

Hu invariant moments is very powerful tool to compare two images

Share Improve this answer

Follow

answered Mar 25, 2021 at 8:30

Leox

370 ● 4 ● 10

0

Hash functions are used in the undouble library to detect (near-)identical images (*disclaimer: I am also the author*). This is a simple and fast way to compare two or more images for similarity. It works using a multi-step process of pre-processing the images (grayscaling, normalizing, and scaling), computing the image hash, and the grouping of images based on a threshold value.

Share Improve this answer

Follow

edited Mar 7, 2022 at 16:02

answered Feb 1, 2022 at 19:16

erdogant

1,694 16 24