State of The Art

Vincent FALCONIERI

February 2019 - August 2019

Contents

0.1 Introduction

A general overview was made through standard web lookup. [Bupe, 2017] A look was given to libraries, which also provide detailed and useful information. [Fea,]

In the following, we expose :

- The main steps of a Image Matching algorithm
- Few of the most popular Image Matching algorithms

Please, be sure to consider this document is under construction, and it can contain mistakes, structural errors, missing areas .. feel free to ping me if you find such flaw. (Open a PR/Issue/...)

Chapter 1

Classic Computer vision techniques - White box algorithms

1.1 Step 1 - Key Point Detection

- Corner detectors to find easily localizable points.

Harris

[Harris and Stephens, 1988]
Distinctive features :

- Rotation-invariant
- NOT scaling invariant

FAST

Distinctive features :

- Not rotation-invariant (no orientation calculation)
- ? scaling invariant

1.2 Step 2 - Descriptor Extraction

Extract a small patch around the keypoints, preserving the most relevant information and discaring necessary information (illumination ..)

Can be :

- Pixels values
- Based on histogram of gradient
- Learnt

Usually :

- Normalized
- Indexed in a searchable data structure

Example Vector descriptors based on our keypoints, each descriptor has size 64 and we have 32 such, so our feature vector is 2048 dimension.

Descriptor's quality A good descriptor code would be, according to [Spe,] :

- easily computed for a novel input
- requires a small number of bits to code the full dataset
- maps similar items to similar binary codewords
- require that each bit has a 50

We should be aware that a smaller code leads to more collision in the hash.

1.3 Step 3 - Matching

Linked to correspondence problem ?

1.3.1 Distance

Hamming distance

Bruteforce

- $O(N^2)$, N being the number of descriptor per image
- One descriptor of the first picture is compared to all descriptor of a second candidate picture. A distance is needed. The closest is the match.
- Ratio test
- CrossCheck test : list of "perfect match" (TO CHECK)

Best match

- Returns only the best match
- Returns the K (parameter) best matchs

FLANN – Fast Library for Approximate Nearest Neighboors

- Collections of algorithm, optimized for large dataset/high dimension
- Returns the K (parameter) best matchs
- [Ope,]

1.3.2 Compression of descriptors before matching

LSH – Locally Sensitive Hashing

- O(˜N)
- Returns the K (parameter) best matchs
- [Ope,]
- Convert descriptor (floats) to binary strings. Binary strings matched with Hamming Distance, equivalent to a XOR and bit count (very fast with SSE instructions on CPU)

BBF – Best bin first Kd-tree

- O(˜N)
- Example : SIFT – Scale Invariant Feature Tranform

1.4 Step 4 - Model Fitting

- Identify inliers and outliers ˜ Fitting a homography matrix ˜ Find the transformation of (picture one) to (picture two)
- Inliers : "good" points matching that can help to find the transformation
- outliers : "bad" points matching
- See [Fea,]

RANSAC – Random Sample Consensus

Estimation of the homography

Least Meadian

Chapter 2

Standard algorithms

Goal is to transform visual information into vector space

2.1 SIFT- Scale Invariant Feature Transform

From the original paper [Lowe, 2004] and a concise explanation from [Int, 2014]

Pro

- test

Con

- Patented algorithm and not included in OpenCV (only non-free module)
- Slow (HOW MUCH TO CHECK)

Steps of the algorithm

1. Extrema detection
   Uses an approximation of LoG (Laplacian of Gaussian), as a Difference of Gaussion, made from difference of Gaussian blurring of an image at different level of a Gaussian Pyramid of the image. Kept keypoints are local extrema in the 2D plan, as well as in the blurring-pyramid plan.
2. Keypoint localization and filtering
   Two threesholds has to be set :
   - Contract Threeshold : Eliminate low contract keypoint (0.03 in original paper)
   - Edge Threeshold : Eliminate point with a curvature above the threeshold, that could match edge only. (10 in original paper)
3. Orientation assignement
   Use an orientation histogram with 36 bins covering 360 degrees, filled with gradient magnitude of given directions. Size of the windows on which the gradient is calculated is linked to the scale at which it's calculated. The average direction of the peaks above 80% of the highest peak is considered to calculate the orientation.
4. Keypoint descriptors
   A 16x16 neighbourhood around the keypoint is divided into 16 sub-blocks of 4x4 size, each has a 8 bin orientation histogram. 128 bin are available for each Keypoint, represented in a vector of float, so 512 bytes per keypoint. Some tricks are applied versus illumination changes, rotation.
5. Keypoint Matching
   Two distance calculation :
   - Finding nearest neighboor.
   - Ratio of closest distance to second closest is taken as a second indicator when second closest match is very near to the first. Has to be above 0.8 (original paper) (TO CHECK what IT MEANS)

2.2 SURF – Speeded-Up Robust Features

[Bay et al., 2006]

Pro

- Faster than SIFT (x3) : Parralelization, integral image ..
- Tradeoffs can be made :
  - Faster : no more rotation invariant, lower precision (dimension of vectors)
  - More precision : **extended** precision (dimension of vectors)
- Good for blurring, rotation

Con

- Patented algorithm
- Not good for illumination change, viewpoint change

Steps of the algorithm

1. Extrema detection
   Approximates Laplacian of Guassian with a Box Filter. Computation can be made in parrallel at different scales at the same time, can use integral images . . .
   Roughly, does not use a gaussian approximation, but a true "square box" for edge detection, for example.
   The sign of the Laplacian (Trace of the Hessian) give the "direction" of the contrast : black to white or white to black. So a negative picture can match with the original ? (TO CHECK)
2. Keypoint localization and filtering
3. Orientation assignement
   Dominant orientation is computed with wavlet responses with a sliding window of 60°
4. Keypoint descriptors
   Neighbourhood of size 20sX20s is taken around the keypoint, divided in 4x4 subregions. Wavelet response of each subregion is computed and stored in a 64 dimensions vector (float, so 256 bytes), in total. This dimension can be lowered (less precise, less time) or extended (e.g. 128 bits ; more precise, more time)
5. Keypoint Matching

2.3 U-SURF – Upright-SURF

Rotation invariance can be "desactivated" for faster results, by bypassing the main orientation finding, and is robust up to 15°rotation.

2.4 BRIEF – Binary Robust Independent Elementary Features

Extract binary strings equivalent to a descriptor without having to create a descriptor

See BRIEF [BRI,]

Pro

- Solve memory problem

Con

- Only a keypoint descriptor method, not a keypoint finder
- Bad for large in-plan rotation

Steps of the algorithm

1. Extrema detection
2. Keypoint localization and filtering
3. Orientation assignement
4. Keypoint descriptors
   Compare pairs of points of an image, to directly create a bitstring of size 128, 256 ou 512 bits. (16 to 64 bytes)
   Each bit-feature (bitstring) has a large variance ad a mean near 0.5 (TO VERIFY). The more variance it has, more distinctive it is, the better it is.
5. Keypoint Matching Hamming distance can be used on bitstrings.

2.5 R-BRIEF – Rotation (?) BRIEF

Variance and mean of a bit-feature (bitstring) is lost if the direction of keypoint is aligned (TO VERIFY : would this mean that there is a preferential direction in the pair of point selection ?)

Uncorrelated tests (TO CHECK WHAT IT IS) are selected to ensure a high variance.

2.6 CenSurE

Pro

Con

2.7 ORB – Oriented FAST and Rotated BRIEF

From [Rublee et al., 2011] which is rougly a fusion of FAST and BRIEF. See also [ORB, 2014]

Pro

- Not patented

Con

Steps of the algorithm

1. Extrema detection
   FAST algorithm (no orientation)
2. Keypoint localization and filtering
   Harris Corner measure : find top N keypoints
   Pyramid to produce multi scale features
3. Orientation assignement
   The direction is extracted from the orientation of the (center of the patch) to the (intensity-weighted centroid fo the patch). The region/patch is circular to improve orientation invariance.
4. Keypoint descriptors
   R-BRIEF is used, as Brief Algorithm is bad at rotation, on rotated patches of pixel, by rotating it accordingly with the previous orientation assignement.
5. Keypoint Matching
   Multi-probe LSH (improved version of LSH)

2.8 KASE -

Shipped in OpenCV library. Example can be found at [Nikishaev, 2018]

Pro

Con

Steps of the algorithm

2.9 FAST – Features from Accelerated Segment Test

Pro

Con

Steps of the algorithm

1. Extrema detection
2. Keypoint localization and filtering
3. Orientation assignement
4. Keypoint descriptors
5. Keypoint Matching

2.10 Delaunay Graph Matching

Algorithm from 2012, quite advanced. Would need some tests or/and review See M1NN [Fang, 2012] that is presenting 3 algorithms :

- **M1NN Agglomerative Clustering**

Different types of data,robust to noise, may 'over' cluster. Better clustering performance and is extendable to many applications, e.g. data mining, image segmentation and manifolding learning.

- **Modified Log-likelihood Clustering**

Measure and compare clusterings quantitatively and accurately. Energy of a graph to measure the complexity of a clustering.

- **Delaunay Graph Characterization and Graph-Based Image Matching**

Based on diffusion process and Delaunay graph characterization, with critical time. Graph-based image matching method. SIFT descriptors also used. Patent problem ? Outperforms SIFT matching method by a lower error rate.

Pro

- Lower error
- Extensible to 3D (but not done yet ?)

Con

- Lower number of matches

Steps of the algorithm

Chapter 3

Hash algorithms

The following algorithms does not intend to match pictures with common part, but to match pictures which are rougly the same. To be clear : If the hashes are different, then the data is different. And if the hashes are the same, then the data is likely the same. There is a possibility of a hash collision, having the same hash values then does not guarantee the same data.

3.1 A-HASH : Average Hash

From ... [Loo, 2011]

"the result is better than it has any right to be."

relationship between parts of the hash and areas of the input image = ability to apply "masks" (like "ignore the bottom 25

8 bits for a image vector.

Idea to be faster (achieve membw-bound conditions) : Batch search (compare more than one vector to all others) = do X search at the same time

More than one vector could be transformation of the initial image (rotations, mirrors)

Javascript Implementation : [Aluigi, 2019]

Pro

- Masks and transformation available
- Ability to look for modified version of the initial picture
- Only 8 bits for a image vector.

Con

- Nd

Steps of the algorithm

1. Ce

3.2 D-HASH

From [Hahn, 2019], DHash is a very basic algorithm to find nearly duplicate pictures. The hash can be of length 128 or 512 bits. The delta between 2 ”matches” is a Hamming distance (# of different bits.)

Pro

- Detecting near or exact duplicates : slightly altered lighting, a few pixels of cropping, or very light photoshopping

Con

- Not for similar images
- Not for duplicate-but-cropped

Steps of the algorithm

1. Convert the image to grayscale
2. Downsize it to a 9x9 thumbnail
3. Produce a 64-bit “row hash”: a 1 bit means the pixel intensity is increasing in the x direction, 0 means it’s decreasing
4. Do the same to produce a 64-bit “column hash” in the y direction
5. Combine the two values to produce the final 128-bit hash value

3.3 P-HASH

From ... [Loo, 2011] Exist in mean and median flavors

8 bits for a image vector. Java implementation : [PHa, 2011]

Pro

- Robustness to gamma
- Robustness to color histogram adjustments

Con

- Nd

Steps of the algorithm

1. Reduce size of the input image to 32x32 (needed to simplify DCT computation)
2. Reduce color to grayscale (same)
3. Compute the DCT : convert image in frequencies, a bit similar to JPEG compression
4. Reduce the DCT : keep the top-left 8x8 of the DCT, which are the lowest frequencies
5. Compute the average DCT value : without the first term (i.e. solid colors)
6. Further reduce the DCT : Set the 64 hash bits to 0 or 1 depending on whether each of the 64 DCT values is above or below the average value.
7. Construct the hash : create a 64 bits integer from the hash
8. Comparing with Hamming Distance (threeshold = 21)

3.4 R-HASH

From ... [Loo, 2011]

Equivalent to A-Hash with more granularity of masks and transformation. Ability to apply "masks" (color channel, ignoring (f.ex. the lowest two) bits of some/all values) and "transformations" at comparison time. (color channel swaps)

48 bits for a rgb image vector

Pro

- Masks and transformation available
- More precise masks (than A-hash)
- More precise transformations (than A-hash)

Con

- Larger memory footprint

Steps of the algorithm

1. Image scaled to 4x4
2. Compute vector
3. Comparison = sum of absolute differences: abs(a[0]-b[0]) + abs(a[1]-b[1]) + ... + abs(a[47]-b[47]) = 48 dimensional manhattan distance

3.5 Spectral-HASH

From [Spe,]. A word is given in [Loo, 2011]

The bits are calculated by thresholding a subset of eigenvectors of the Laplacian of the similarity graph

Similar performance to RBM

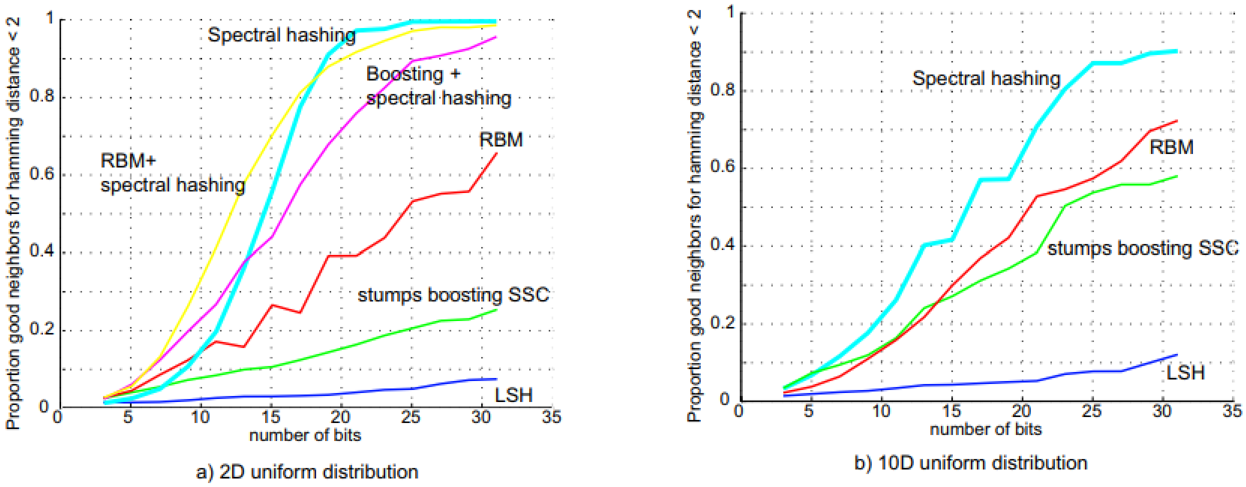

Figure 3.1: Spectral Hashing comparison from [Spe,]

Pro

- D

Con

- Nd

Steps of the algorithm

1. Ce

3.6 E2LSH - LSH - Locality Sensitve Hashing

From ... A word is given in [Spe,] The code is calculated by a random linear projection followed by a random threshold, then the Hamming distance between codewords will asymptotically approach the Euclidean distance between the items.

Not so far from Machine Learning Approaches, but outperformed by them.

Pro

- Faster than Kdtree

Con

- Very inefficient codes (512 bits for a picture (TO CHECK))

Steps of the algorithm

1. Ce

Chapter 4

Neural networks – Black box algorithms

4.1 RBM - Restricted Boltzmann machine

From ... A word is given in [Spe,]

To learn 32 bits, the middle layer of the autoencoder has 32 hidden units Neighborhood Components Analysis (NCA) objective function = refine the weights in the network to preserve the neighborhood structure of the input space.

Pro

- More compact outputs code of picture than E2LSH = Better performances

Con

- e

Steps of the algorithm

1. Ce

4.2 RPA - Robust Projection Algorith

From ... [Igor, 2011]

Pro

- M

Con

- e

Steps of the algorithm

1. R

. Like Average Hash, pHash starts with a small image. However, the image is larger than 8x8; 32x32 is a good size. This is really done to simplify the DCT computation and not because it is needed to reduce the high frequencies. Reduce color . The image is reduced to a grayscale just to further simplify the number of computations. Compute the DCT . The DCT separates the image into a collection of frequencies and scalars. While JPEG uses an 8x8 DCT, this algorithm uses a 32x32 DCT. Reduce the DCT . This is the magic step. While the DCT is 32x32, just keep the top-left 8x8. Those represent the lowest frequencies in the picture. Compute the average value . Like the Average Hash, compute the mean DCT value (using only the 8x8 DCT low-frequency values and excluding the first term since the DC coefficient can be significantly different from the other values and will throw off the average). Thanks to David Starkweather for the added information about pHash. He wrote: "the dct hash is based on the low 2D DCT coefficients starting at the second from lowest, leaving out the first DC term. This excludes completely flat image information (i.e. solid colors) from being included in the hash description." Further reduce the DCT . This is the magic step. Set the 64 hash bits to 0 or 1 depending on whether each of the 64 DCT values is above or below the average value. The result doesn't tell us the actual low frequencies; it just tells us the very-rough relative scale of the frequencies to the mean. The result will not vary as long as the overall structure of the image remains the same; this can survive gamma and color histogram adjustments without a problem. Construct the hash . Set the 64 bits into a 64-bit integer. The order does not matter, just as long as you are consistent. To see what this fingerprint looks like, simply set the values (this uses +255 and -255 based on whether the bits are 1 or 0) and convert from the 32x32 DCT (with zeros for the high frequencies) back into the 32x32 image: = 8a0303769b3ec8cd At first glance, this might look like some random blobs... but look closer. There is a dark ring around her head and the dark horizontal line in the background (right side of the picture) appears as a dark spot.

4.3 Boosting SSC

From ... A word is given in [Spe,]

Pro

- Better than E2LSH

Con

- Worst than RBM

Steps of the algorithm

1. Ce

4.4 ConvNet - Convolutional Neural Networks

Learn a metric between any given two images. The distance can be threesholded to decide if images match or not.

Training phase

Goal :

- Minimizing distance between “same image” examples
- Maximizing distance between “not same image” examples

Evaluation phase

Apply an automatic threeshold.

SVM - Support Vector Machine

Bibliography

[BRI,] BRIEF (Binary Robust Independent Elementary Features) — OpenCV 3.0.0-dev documentation. https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_feature2d/py_brief/py_brief.html#brief.

[Fea,] Feature Matching + Homography to find Objects — OpenCV 3.0.0-dev documentation. https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_feature2d/py_feature_homography/py_feature_homography.html#py-feature-homography.

[Ope,] OpenCV: Feature Matching. https://docs.opencv.org/3.4.3/dc/dc3/tutorial_py_matcher.html.

[Spe,] Spectralhashing.pdf. http://people.csail.mit.edu/torralba/publications/spectralhashing.pdf.

[Loo, 2011] (2011). Looks Like It - The Hacker Factor Blog. http://www.hackerfactor.com/blog/index.php?/archives/432-Looks-Like-It.html.

[PHa, 2011] (2011). pHash-like image hash for java. https://pastebin.com/Pj9d8jt5.

[Int, 2014] (2014). Introduction to SIFT (Scale-Invariant Feature Transform) — OpenCV 3.0.0-dev documentation. https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_feature2d/py_sift_intro/py_sift_intro.html#sift-intro.

[ORB, 2014] (2014). ORB (Oriented FAST and Rotated BRIEF) — OpenCV 3.0.0-dev documentation. https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_feature2d/py_orb/py_orb.html#orb.

[Aluigi, 2019] Aluigi, V. (2019). JavaScript implementation of the Average Hash using HTML5 Canvas.

[Bay et al., 2006] Bay, H., Tuytelaars, T., and Van Gool, L. (2006). SURF: Speeded Up Robust Features. In Leonardis, A., Bischof, H., and Pinz, A., editors, *Computer Vision – ECCV 2006*, volume 3951, pages 404–417. Springer Berlin Heidelberg, Berlin, Heidelberg.

[Bupe, 2017] Bupe, C. (2017). What algorithms can detect if two images/objects are similar or not? - Quora. https://www.quora.com/What-algorithms-can-detect-if-two-images-objects-are-similar-or-not.

[Fang, 2012] Fang, Y. (2012). Data Clustering and Graph-Based Image Matching Methods.

[Hahn, 2019] Hahn, N. (2019). Differentiate images in python: Get a ratio or percentage difference, and generate a diff image - nicolashahn/diffimg.

[Harris and Stephens, 1988] Harris, C. and Stephens, M. (1988). A Combined Corner and Edge Detector. In *Procedings of the Alvey Vision Conference 1988*, pages 23.1–23.6, Manchester. Alvey Vision Club.

[Igor, 2011] Igor (2011). Nuit Blanche: Are Perceptual Hashes an instance of Compressive Sensing ?

[Lowe, 2004] Lowe, D. G. (2004). Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 60(2):91–110.

[Nikishaev, 2018] Nikishaev, A. (2018). Feature extraction and similar image search with OpenCV for newbies.

[Rublee et al., 2011] Rublee, E., Rabaud, V., Konolige, K., and Bradski, G. (2011). ORB: An efficient alternative to SIFT or SURF. In *2011 International Conference on Computer Vision*, pages 2564–2571, Barcelona, Spain. IEEE.