**Abstract:**

This project is mainly about detecting and measure the movements of nerves from Ecography Videos by using SIFT *(Scale-invariant feature transform)* method. The Ecography Videos contains the movements of hip tissue when the leg is stretch or curl. Since the image contain both muscles and nerves, my goal is to distinguish nerves from muscles and record the movements of the nerve.

**Background:**

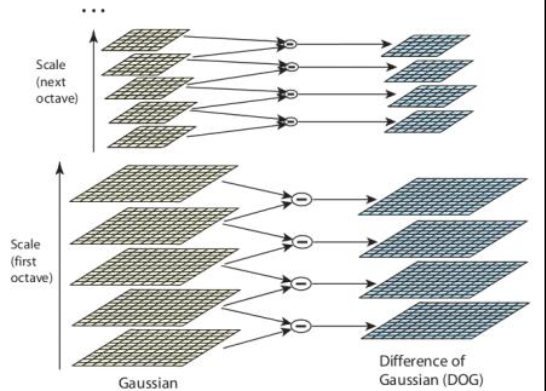
The ecography videos are recorded for the research about physical injury. And when the researchers examine the videos, they surprisingly find out that the nerves in the video are moving elastically along the muscles and some point of the nerve seems connected with the muscle.

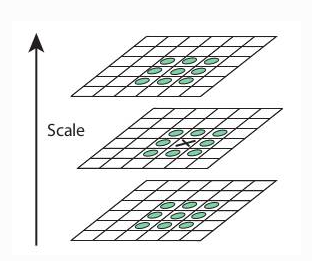
This feature has never been noticed before and if we can record the movement of the nerve then we can see if it is changed when the injury occurs and find a way to reduce the pain.

**About SIFT:**

SIFT algorithm is a method which can detect and describe the local features in an image. By using this method, we can know which part of the points on the image is Scale-invariant, which means nonmatter the movements of the camera, we can always know which part is our interested part.

I choose this method because the task is to trace the movement of the nerve in a video, which means we must know the correspondence between two frame and measure the movement between them. With SIFT, we can match the correspondent points even they are different in scale.

SIFT algorithm uses Difference of Gaussians which is obtained as the difference of Gaussian blurring of an image with two different σ, let it be σ and kσ. This process is done for different octaves of the image in Gaussian Pyramid. It is represented in below image:

Once this DoG are found, images are searched for local extrema over scale and space. For example, one pixel in an image is compared with its 8 neighbors as well as 9 pixels in next scale and 9 pixels in previous scales. If it is a local extremum point, it is a potential key-point. It basically means that key-point is best represented in that scale. It is shown in below image:

Regarding different parameters, the paper gives some empirical data which can be summarized as, number of octaves = 4, number of scale levels = 5, initial σ=1.6, k= etc as optimal values.

Indexing consists of storing SIFT keys and identifying matching keys from the new image. Lowe used a modification of the k-d tree algorithm called the Best-bin-first search method that can identify the nearest neighbors with high probability using only a limited amount of computation. The BBF algorithm uses a modified search ordering for the k-d tree algorithm so that bins in feature space are searched in the order of their closest distance from the query location. This search order requires the use of a heap-based priority queue for efficient determination of the search order. The best candidate match for each key-point is found by identifying its nearest neighbor in the database of key-points from training images. The nearest neighbors are defined as the key-points with minimum Euclidean distance from the given descriptor vector. The probability that a match is correct can be determined by taking the ratio of distance from the closest neighbor to the distance of the second closest.

Hough Transform is used to cluster reliable model hypotheses to search for keys that agree upon a particular model pose. Hough transform identifies clusters of features with a consistent interpretation by using each feature to vote for all object poses that are consistent with the feature. When clusters of features are found to vote for the same pose of an object, the probability of the interpretation being correct is much higher than for any single feature. An entry in a hash table is created predicting the model location, orientation, and scale from the match hypothesis. The hash table is searched to identify all clusters of at least 3 entries in a bin, and the bins are sorted into decreasing order of size.

Each of the SIFT key-points specifies 2D location, scale, and orientation, and each matched key-point in the database has a record of its parameters relative to the training image in which it was found. The similarity transform implied by these 4 parameters is only an approximation to the full 6 degree-of-freedom pose space for a 3D object and also does not account for any non-rigid deformations. Therefore, Lowe used broad bin sizes of 30 degrees for orientation, a factor of 2 for scale, and 0.25 times the maximum projected training image dimension (using the predicted scale) for location. The SIFT key samples generated at the larger scale are given twice the weight of those at the smaller scale. This means that the larger scale is in effect able to filter the most likely neighbors for checking at the smaller scale. This also improves recognition performance by giving more weight to the least-noisy scale. To avoid the problem of boundary effects in bin assignment, each key-point match votes for the 2 closest bins in each dimension, giving a total of 16 entries for each hypothesis and further broadening the pose range.

Lowe rejected all matches in which the distance ratio is greater than 0.8, which eliminates 90% of the false matches while discarding less than 5% of the correct matches. To further improve the efficiency of the best-bin-first algorithm search was cut off after checking the first 200 nearest neighbor candidates. For a database of 100,000 keypoints, this provides a speedup over exact nearest neighbor search by about 2 orders of magnitude, yet results in less than a 5% loss in the number of correct matches.

Once potential key-points locations are found, they have to be refined to get more accurate results. They used Taylor series expansion of scale space to get more accurate location of extrema, and if the intensity at this extremum is less than a threshold value (0.03 as per the paper), it is rejected. This threshold is called contrastThreshold in OpenCV

DoG has higher response for edges, so edges also need to be removed. For this, a concept similar to Harris corner detector is used. They used a 2x2 Hessian matrix (H) to compute the principal curvature. We know from Harris corner detector that for edges, one edge value is larger than the other. So here they used a simple function, If this ratio is greater than a threshold, called edge Threshold in OpenCV, that key-point is discarded. It is given as 10 in paper. So it eliminates any low-contrast key-points and edge key-points and what remains is strong interest points.

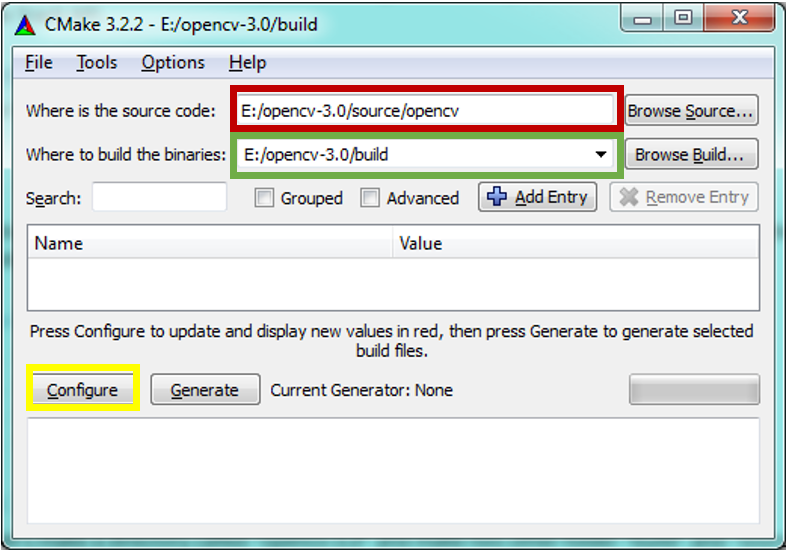
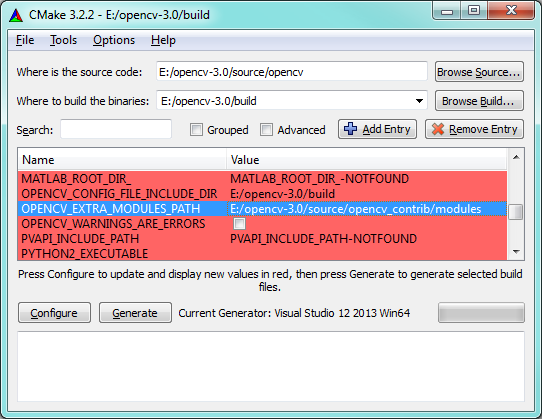
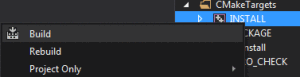
Now key-point descriptor is created. A 16x16 neighborhood around the key-point is taken. It is divided into 16 sub-blocks of 4x4 size. For each sub-block, 8 bin orientation histogram is created. So a total of 128 bin values are available. It is represented as a vector to form key-point descriptor. In addition to this, several measures are taken to achieve robustness against illumination changes, rotation etc.

By using SIFT, we must do a lot of calculation base of the image, such as firstly we must find the Key Location of the image. Key locations are defined as maxima and minima of the result of difference of Gaussians function applied in scale space to a series of smoothed and resampled images. Low contrast candidate points and edge response points along an edge are discarded. Dominant orientations are assigned to localized key-points. These steps ensure that the key-points are more stable for matching and recognition.

And secondly we must match the points we have with a new image. Here we will use a method called *Best-bin-first* search method that can identify the nearest neighbors with high probability using only a limited amount of computation.

**Approach:**

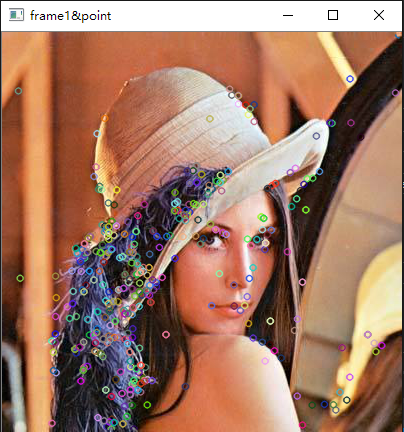
As this project is using C++, I tried an open source library called OpenCV. It contains a class called *xfeatures2d* which allows us to use SIFT algorithm in a very convenient way. But first we must link our C++ project to the OpenCV library. I’m using Visual Studio 2015 and the version of OpenCV library is 3.1.0. 1. This version of OpenCV (3.1.0) is not contain the SIFT class because it’s considered not stable. So we have to install something called opencv\_contrib first.

1. Install the cmake-3.4.3-win32-x86.exe in the folder and open it, You will see as the following picture The red box must be filled with the directory path of OpenCV source, and the green box must be filled with the directory path of designated build folder.
2. Then click configure and follow the instruction until the content said configuring done!
3. Next, we need to specify the extra modules path which is opencv\_contrib /modules as depicted in figure below: 
4. Then click the configure button again and wait until it’s done. Now you will find a solution file in you OpenCV/build folder called ‘OpenCV.sln’ (not the opencv\_contrib one).
5. Use the VS2015 to open the OpenCV.sln file and buid the INSTALL program in release mode.
6. Then in the path of OpenCV.sln, there will be a new install folder. It is now a new OpenCV with SIFT in it.
7. In the Project→Properties→VC++ Directories add include and lib path of OpenCV to the project and add the lib file name we use(opencv\_world310.lib; opencv\_xfeatures2d310.lib) in the linker.

**Experiments:**

1. We first test SIFT on its most widely used way: Match two picture when the content is different in scale. Here we use the famous girl in image processing as the origin picture, and make a zoom and rotation as the second picture:

and the key-points find by using SIFT when the minimum Hessian is 400 the key-points are as below:

cv::Ptr<cv::Feature2D> f2d = cv::xfeatures2d::SIFT::create(400);

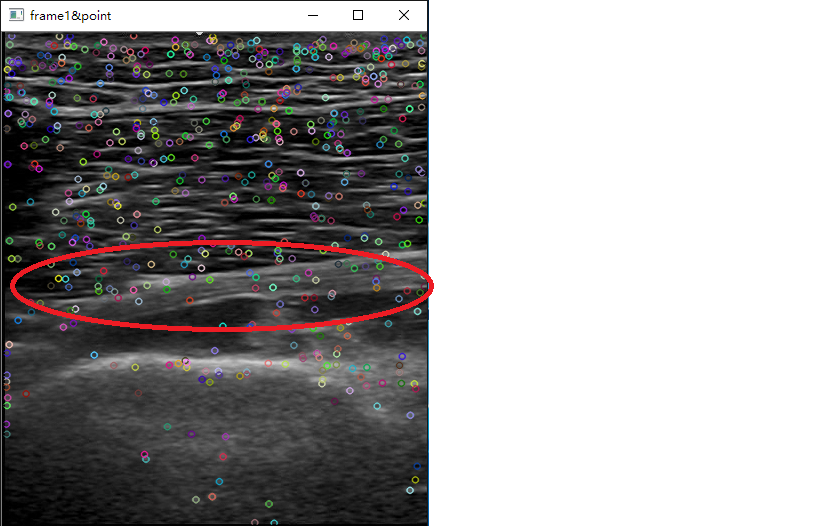
If we don’t set the minimum Hessian we will get all the key-points we can get. And the smaller the minimum Hessian we set, the less point we can get. But the points will perform better when we use it.

Not all the key points that we find can draw a good match, by using some selection we will get a more accurate result, in this test I only consider the distance between two match is less than twice the length of the minimum distance of all matches. You can see that the points in the above picture is a lot, but bellow the matches are not so much, it is because some of the matches is not showed because they are considered “bad”

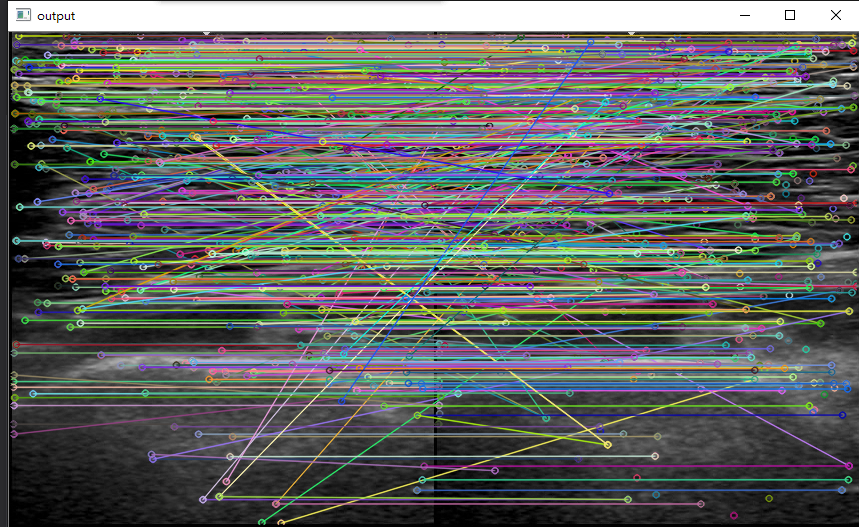


And if we select the key-points more strictly by lower the minimum Hessian value to 20, we will get a more obvious result as blow and we can see the matches are very good and easy to see.

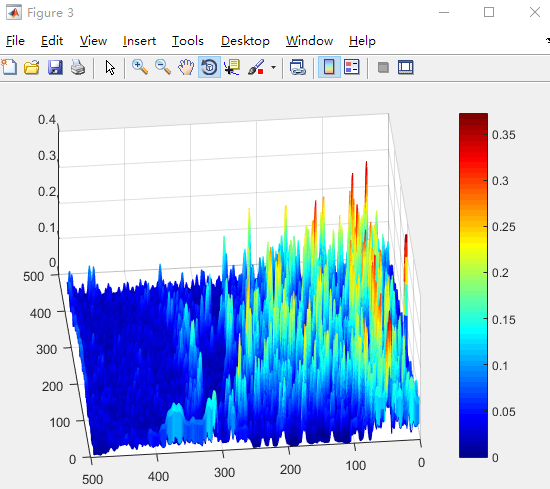
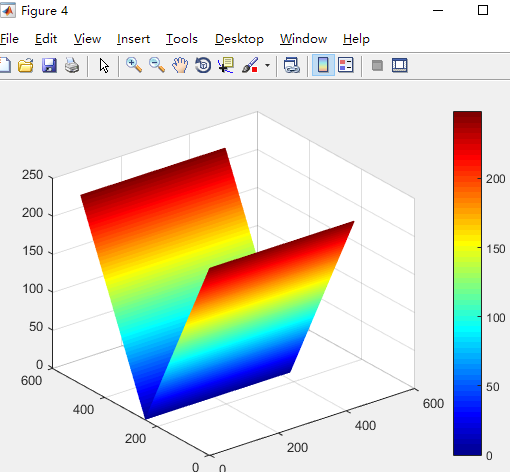


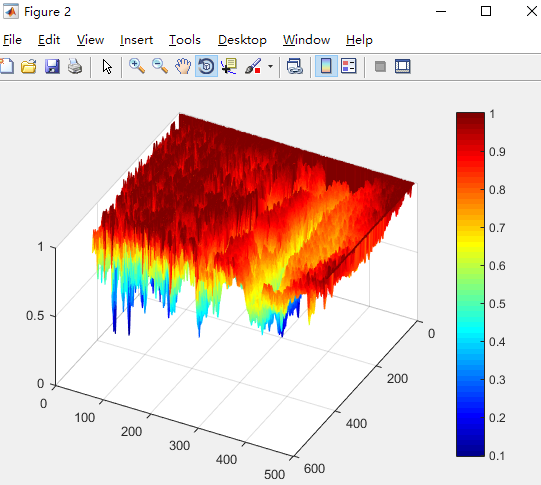
1. Then I tried to use SIFT on the echography video. By using the SIFT method in OpenCV we can easily get the key-points and the match result like above, by using the first frame of a test video we can see the result for finding key-points is like this:

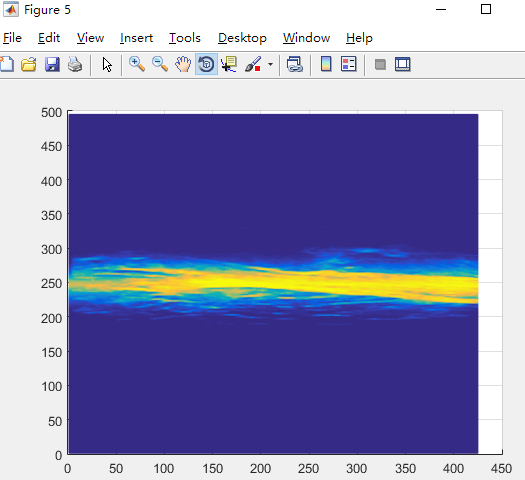
The points are showed on the above picture are only 1/10 of the whole points detected by SIFT detector, and the white part in red circle is the nerve. So here is the bad part of SIFT: we can’t decide which part of the picture is the ROI*(region-of-interest)* and we can’t decide which point is the real key-point.

And if we try to match the points between this frame and the following frame, here is the result:

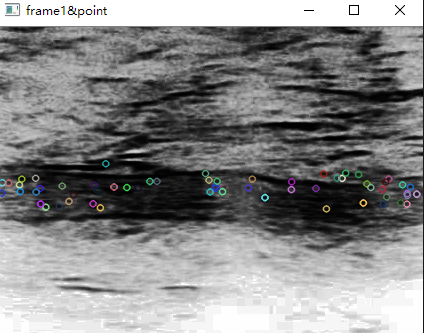
We can see the matching is a mess because of the amount of the points and some bad matching. Of course we can select some of the points like we did to the girls picture, but the problem is the girls picture have some good features but in this frame of ecography video, most of the picture look the same. So there are some key-points get form SIFT are actually not good and will cause ‘bad’ matches easily with other points. So our next goal is to reduce the amount of the key-points, make sure we only get the key-points on the nerve region and remove bad match.

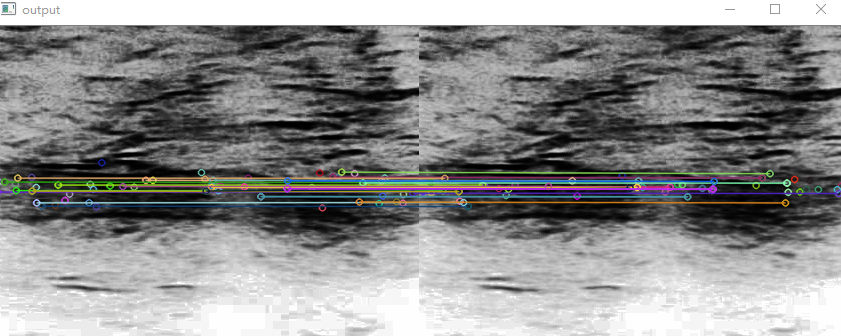
1. To solve the problem, we first must make sure all the points are on the region of interest we use three filter to select the points.
2. Select the color of the nerve (in this case, the nerve is in black because we use a negative effect) which means if a key-point weight more if it is darker.
3. Select the center part of the image since the nerve is the interested part and it’s usually in the center part (here we can’t find a better solution because we have to decide which part of the image is nerve but we can’t use some make to find nerve in a picture because actually all the nerves are quite different and the muscles nearby is also confusing a lot)
4. Select the part of the image where it’s neighbor looks like it (the nerve is a big part; this will help us getting the boarder correctly) in this step we use standard deviation of a square near the key-point to define the difference.



We trained the three filter through Matlab using a mask to make sure all the nerve points from the test picture can be detected, and get the result like the following picture

When apply on the OpenCV, I just take the parameter into the program and test with another video with a nerve in the center and the result is like this, most of the points left is on the ROI and now it’s time to select the match.

1. When selecting the matches, the problem is how to define a match is good or bad. Because in the video the motion of the nerve is slow so the difference between two frame is slight. So we consider that if the match result of the two point is too far, it is a bad match. Also because all the part of the nerve will move in one direction, so we must remove the result that moves in wrong direction (because they are miss matched).



1. After successfully get matches between two frame, the next target is to record all the movement between frames in a video and see the result if it is correct. But here we meet a problem, it is because of the feature of SIFT, so we can’t trace a point throughout the video. That means we can’t make sure the movement is always recorded.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 153.838 | 175.137 | 153.479 | 176.32 | 149.213 | 320.091 | 144.981 | 331.878 |
| 153.888 | 176.271 | 153.712 | 178.24 | 150.078 | 320.468 | 144.804 | 333.466 |
| 1.51701 |  |  |  |  |  |  |  |

Above is one line of the result, the first line is the coordinate of the points on the first frame, and the second line is for the following frame, the third like is the average motion of all the point above in this case is 1.5pixel.

**Results:**

The detect and the match job of this project is complete quite good, but the record job failed because SIFT itself actually doesn’t suitable for this job. From the data we record, the motion of the nerve can be detected most of the time but between some of the frame (about 10 in 150) the motion will be lost or miscalculate, as the total movement is not very large, this slight mistake will cause the result not accurate enough for medical study.

**Related work:**

Also we tried some of the improve method base on SIFT, such as SURF and PCA-SIFT, they all improve SIFT in some way so it can use in some special occasion, such as SURF can get the key-points faster than SIFT but the match result by using SURF is not as good as SIFT. But since the main problem is we can’t trace point has not been solved, they are also not suitable for this task.

**Conclusion:**

The SIFT algorithm is more suitable for getting the matches between two different picture about one object when the scale of the object is different. Here in this case, we need a method that can trace a point or an object through a whole video when the scale is not changed so obvious and the picture is not so clear, the result show that although SIFT is able to get the key-points and make some good matches, it is not accurate enough for this kind of video. But I think SIFT still work best in its own area.

**Quote:**

1. *Lowe, David G. (1999). "Object recognition from local scale-invariant features". Proceedings of the International Conference on Computer Vision. pp. 1150–1157. doi:10.1109/ICCV.1999.790410*
2. *Lowe, David G. (2004). "Distinctive Image Features from Scale-Invariant Keypoints". International Journal of Computer Vision 60 (2): 91–110. doi:10.1023/B:VISI.0000029664.99615.94*
3. *Lowe, D.G., Local feature view clustering for 3D object recognition. IEEE Conference on Computer Vision and Pattern Recognition,Kauai, Hawaii, 2001, pp. 682-688.*
4. *U.S. Patent 6,711,293, "Method and apparatus for identifying scale invariant features in an image and use of same for locating an object in an image", David Lowe's patent for the SIFT algorithm, March 23, 2004*
5. *Lindeberg, Tony (2012). "Scale invariant feature transform". Scholarpedia 7 (5): 10491. doi:10.4249/scholarpedia.10491*
6. [*https://en.wikipedia.org/wiki/Scale-invariant\_feature\_transform#cite\_ref-Lowe1999\_1-0*](https://en.wikipedia.org/wiki/Scale-invariant_feature_transform#cite_ref-Lowe1999_1-0)
7. [*http://docs.opencv.org/2.4/doc/tutorials/features2d/feature\_flann\_matcher/feature\_flann\_matcher.html*](http://docs.opencv.org/2.4/doc/tutorials/features2d/feature_flann_matcher/feature_flann_matcher.html)
8. [*http://opencv-python-tutroals.readthedocs.io/en/latest/py\_tutorials/py\_feature2d/py\_sift\_intro/py\_sift\_intro.html*](http://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_feature2d/py_sift_intro/py_sift_intro.html)