# Detecting and Identifying Salient Components in Images

**COMP-558**

**Prof. Kaleem Siddiqui**

**McGill University**

**Fall 2013**

by

Navjot Singh - 260565243

Priya Sidhaye - 260589025

## Abstract

The problem of tagging images by identifying the objects in an image is a deep learning problem. The problem combines Computer Vision and Machine Learning fields. The challenge is to identify the objects in spite of scale, orientation and the natural variety of objects in the real world. This project aims to detect in images certain specific components of images, like logos, that have a defined colour and layout which does not change much. This can be achieved with the help of SIFT(Scale Invariant Feature Transformation) algorithm and search algorithms for matching to help identify the tags.

## Contents

# I. Introduction

## *A] Basic Idea*

The idea for the project was to explore how images can be described using natural language. The process would include processing images to identify the components in it, gauging their positions in the image, and preparing a description of them in simple language. This is done using natural language tools.

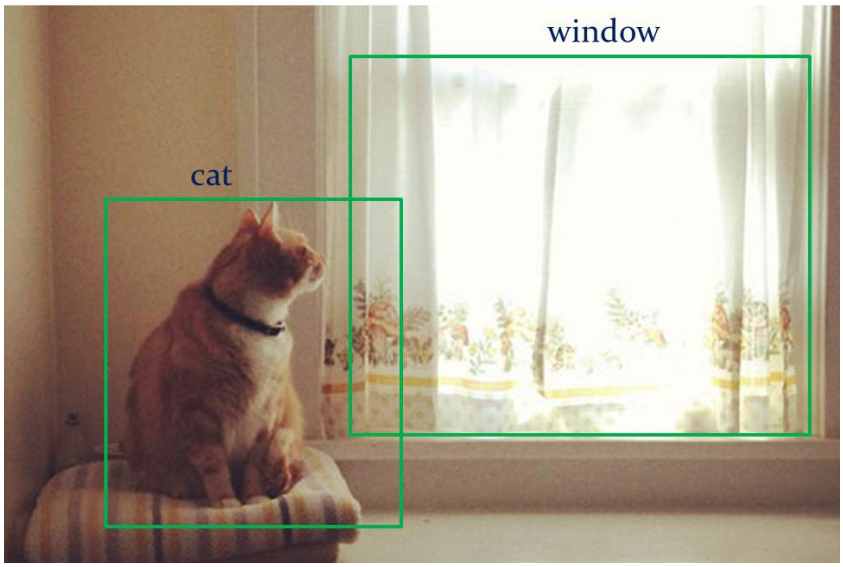


Figure 1. Example. The natural language description would be "A cat to the left of a window".

This problem, however, is a complex deep learning problem. It will require the collection of a database, learning various objects and tagging them. It also includes a significant natural language component in translating the tags from the learning process and translating them into a more readable form. Objects in the real world itself contain a large variety of shapes and sizes. Learning all these variations can become a huge project. So, a small subset of this problem is chosen for this project.

## *B] Problem Definition*

We define the problem as detecting whether a logo or an icon is present in a test query image. The logos will be predefined, and the algorithm will tag the query image as containing a logo from the selected set or not and tagging them accordingly. For example,

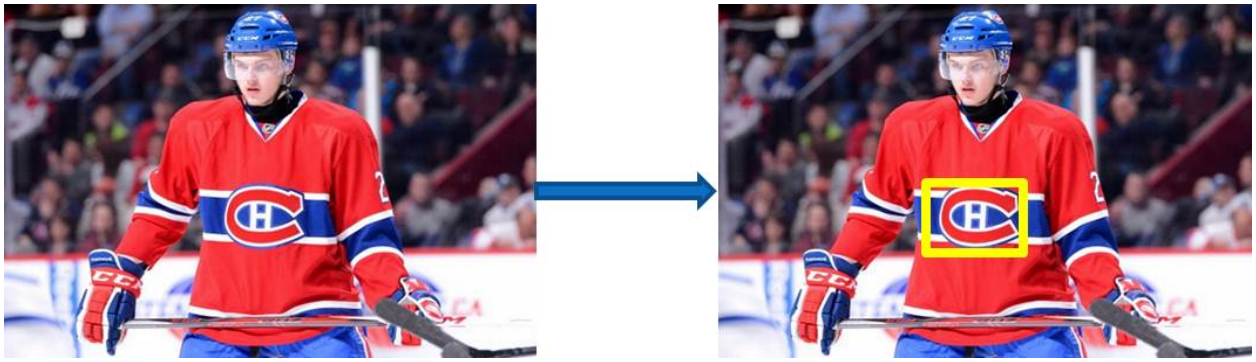

Figure 2. The image on the left is the query image and the image on the right shows the logo of Montreal Canadiens highlighted as it should be detected in the result.

Logos have a defined shape and colour. Thus the variations can occur in scaling, orientation, or viewpoint of the logo. These problems can be addressed using Computer Vision algorithms. Logos of sports teams were chosen to be tagged in test images.

### *C] Approach*

The approach to solving this problem is :

1. Decide the sports teams for the dataset and collect images for their logos. : The sports teams used were a combination of hockey, cricket and football teams and logos were collected that represented the teams and current and past logos were combined.
2. Use a feature detection tool on these images and store these features in an optimum way. : The feature detection algorithm used is SIFT - Scale Invariant Feature Transform. SIFT provides descriptors for features in an image and has a high accuracy.
3. When there is an incoming query image, generate feature descriptors for this query image.
4. Use an effective search algorithm to search through the dataset of feature descriptors and find a match.
5. Output the tag associated with the matched logo as the tag of the feature descriptors in the query image.

The output is the location of the logo in the query image, and it will be tagged with the name of the team the logo belongs to.

## II. Methodology

SIFT and k-d trees for SIFT matching are the major methodologies used in solving the defined problem.

### *A] SIFT*

SIFT, Scale Invariant Feature Transform is an algorithm for feature extraction. The SIFT algorithm can detect features irrespective of scaling, rotation, illumination and change in viewpoint in images. Features in SIFT are detected as feature vectors, called *frames*, or *key points*. A feature is a selected image region. It is represented as four numbers - ones which define the boundary of the frame(in pixels) and the angle to which it is tilted with respect to the image edges.

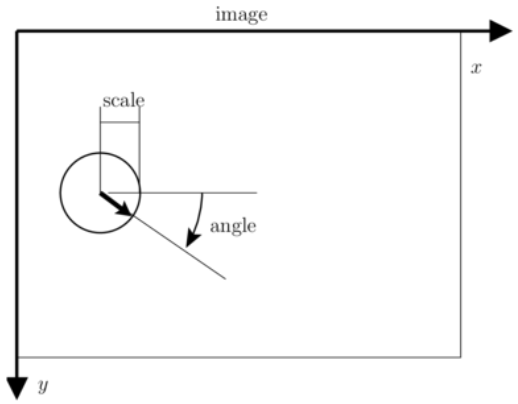


Figure 3 : The values stored for the frame keypoint.[5]

Thus, the frame descriptors are oriented and scaled corresponding to the particular component in the image. Various levels of smoothing or scales are considered. Increasing the smoothing factors affects how much the image is blurred. At each iteration the image blurring is increased to twice the amount it

was in the earlier iteration to save computations. All the convolved images at different scales/blur level forms an octave. Several octaves can be formed by resizing the image. The keypoints are detected based on the local extremas in the image after applying the DoG filter (Difference of Gaussian) to the images in each octave. The keypoints thus obtained are filtered out to eliminate trivial results such as the ones close to the edges, or ones that are small and could have been generated by noise, etc.

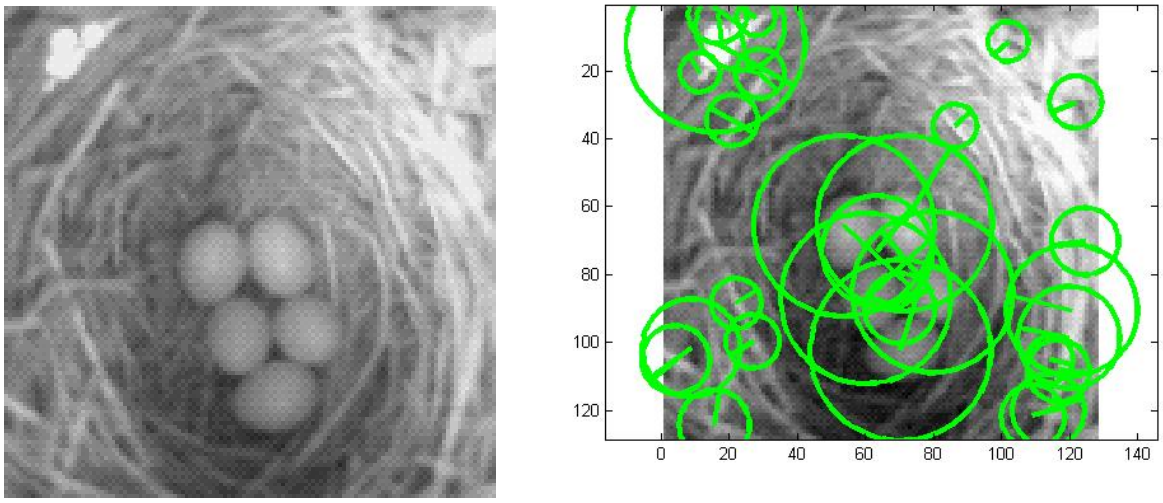


Figure 4. Image on the right shows the keypoints detected by SIFT enclosed in circles for the original image on the left. These are the frame keypoints and the descriptors.

These features are returned by SIFT as a vector of length 128 for each feature. Thus for each image, and for every feature in the image, SIFT gives *k* features, each an array of 128 scalars. These vectors are called the frame descriptors. A SIFT descriptor is the spatial histogram of the gradient of the image at that keypoint. An additional weighting function is applied on this histogram, so that the points towards the edge of the frame descriptor count less than the ones near the centre. The 128 size vector of the descriptor is obtained as follows.

The gradient is calculated at the keypoint of the frame. It is calculated as follows : the orientation is divided into 8 directions, called bins, and the spatial coordinates into 4 each, that is 4 x 4 = 16 bins.

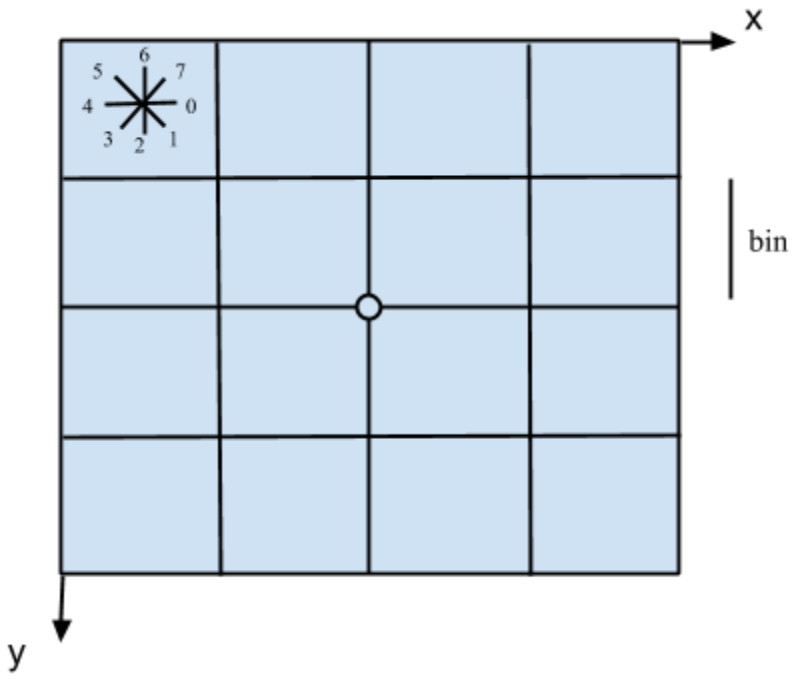


Figure 5: The bins into which the keypoint is divided to generate a histogram of the gradient. The circle at the centre represents the keypoint.

The angles are measured clockwise and Y axis points downward. Thus, the total gradient gives 8x16 = 128 values. These values are stacked in a single vector to give the feature descriptor.

## *B] Feature Matching*

### B.1] Using Linear Search

In this approach, each descriptor is matched with all the descriptors found from the training dataset and the descriptors having the lowest euclidean distances with the query descriptors would give the correct tag.

### B.2] Using K-d trees

To search all descriptors for a big training set, exhaustive linear search would be slow. SIFT descriptor matching is done best with the help of k-dimensional tree with best bin first search method. The

algorithm finds the nearest neighbour based on Euclidean distance from the search image and returns the tag associated with the result. A threshold can be defined on the accuracy for matches to remove the false positives.

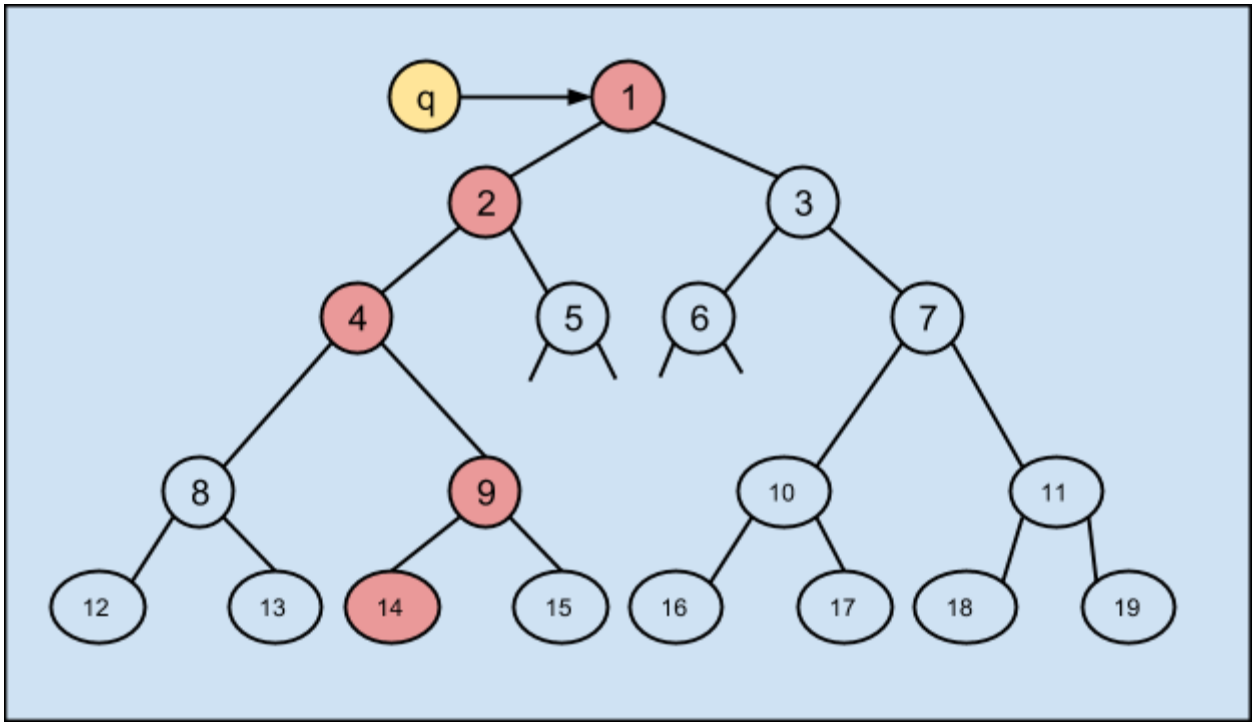


Figure 6. The figure represents a k-d tree's traversal. The path taken is indicated by the nodes in red.

The figure 6 shows a k-dimensional binary tree. The node q represents the query image descriptor. Each node is a 128- dimensional vector, the size of a feature descriptor obtained from SIFT.

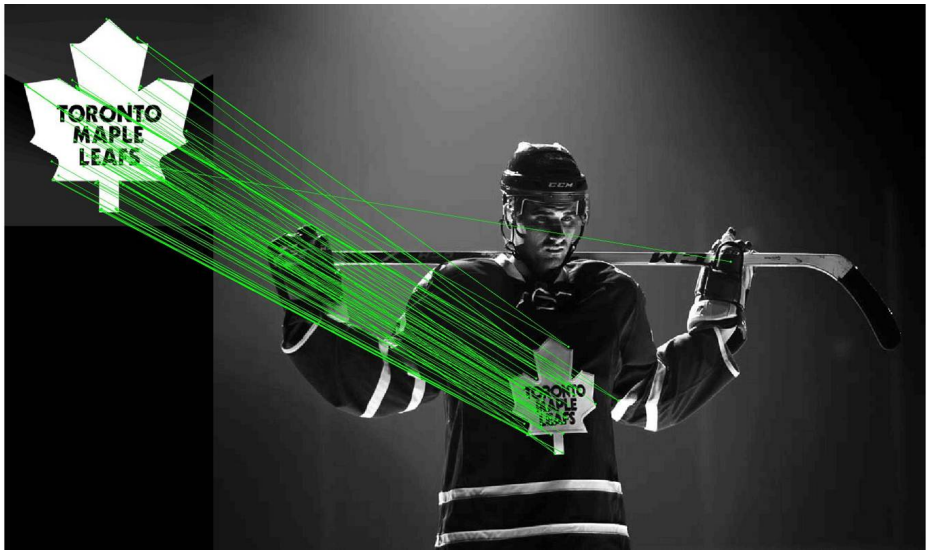


Figure 7. The figure shows a basic SIFT[5] matching

### *C] Challenges*

One of the challenges faced by this method is how to detect the area that is specific to the part where the logo is in the query image. This becomes an issue when the query image has a small logo and there are a lot of false matches(which should not be considered). In this case, the descriptors of the image in general are far greater in number compared to the descriptors of the logo and hence this does not lead to good results. A couple of approaches to solve this problem are :

1. Use the smoothing in SIFT descriptor detection algorithm to extract rough outlines of components
2. Divide the image into smaller patches and use the descriptors of these patches to search in the dataset.

## III. Algorithm

The defined problem can be solved by going through the following steps :

1. Collect a dataset of images of logos.
2. Extract features of these images using the SIFT algorithm and store the descriptors thus

obtained. The descriptors have the length of 128. They will be stored as :

$[d_{11}\ d_{12}\ d_{13}\ \ldots\ldots\ d_{1,128}]_1$

$[d_{21}\ d_{22}\ d_{23}\ \ldots\ldots\ d_{2,128}]_2$

.

.

.

$[d_{N1}\ d_{N2}\ d_{N3}\ \ldots\ldots\ d_{N,128}]_N$

3. Build a balanced k-dimensional tree using all the descriptors from the training dataset.
4. Extract features of query image for searching.
5. Use the SIFT matching algorithm to find the nearest neighbour to the query descriptors.

| Query descriptors | Database of Feature descriptors |
|---|---|
| $[x_{11}\ x_{12}\ x_{13}\ \ldots\ldots\ x_{1,128}]_1$ | $[d_{11}\ d_{12}\ d_{13}.....d_{1,128}]$ :tag 1 |
| $[x_{21}\ x_{22}\ x_{23}\ \ldots\ldots\ x_{2,128}]_2$ | $[d_{11}\ d_{12}\ d_{13}.....d_{1,128}]2$ :tag 2 |
| . | . |
| . | . |
| . | . |
| $[x_{n1}\ x_{n2}\ x_{n3}\ \ldots\ldots\ x_{n,128}]_n$ | $[d_{11}\ d_{12}\ d_{13}.....d_{1,128}]$ N :tag N |

5.1 *Using linear search*

Each query descriptor is compared to all the descriptors in the dataset. This approach is slower than the k-d tree approach but would give the most accurate results as we are considering the whole dataset. There is a trade off between accuracy and performance which can be varied by adjusting the patch size and the overlap size of the patch.

5.2 *Using k-d tree*

Each node in the tree is a 128 dimensional vector. A bin size is defined before traversing the tree. We keep the information regarding the nodes in this bin which have the lowest euclidean distance with the query descriptor. The number of these nodes saved in the bin are defined by

the bin size. At each point, the Euclidean distance is calculated and if this distance is lower than any distance in the bin, we put the current node in the bin by replacing it with the node with highest distance.

During traversal, for each level, a dimension is selected on which the tree node and the query node value is to be compared and tree is traversed in a similar way a one dimensional binary tree is traversed. We save the lowest distances in the bin till we reach a leaf node. Then we pop the node with the lowest distance and traverse it's child which was not taken during the previous traversal. This process is repeated till all the nodes from the bin are traversed. As with the linear search, we can vary the patch size and the overlap size of the patch.

5.3 *Comparing the two search approaches*

The trade off is between performance and correctness. While the linear search would give the most accurate results depending on the patch and overlap sizes, the k-d tree approach might give similar results in lesser time. We can increase the bin size in k-d tree approach so that we cover more nodes on the tree which would give better results while increasing the time of comparisons. The input dataset would also make a considerable difference for both approaches. We would get better results for both approaches if for each logo we have multiple images in our dataset.

6. Return the tag and the patch of the detected logo.

## IV. Implementation

### *A] Collection of dataset*

A number of freely available images of standard logos of various sports teams were collected from google images and wikipedia and saved in individual directories with team name as the directory name.

### *B] Extraction of Features using SIFT*

The SIFT features were extracted using the matlab implementation of sift algorithm provided on D. Lowe's UBC web-page[3]. The extracted features are stored in a k-by-129 matrix. The extra column in the matrix is for saving the index of the logo.

### *C] Implementation of k-d tree*

The challenge here is to make a *balanced* 128 dimensional tree. The tree might not be complete balanced tree because the number of extracted features might not be a power of 2. .From the matrix, we find the median value on the first dimension and we select this descriptor row to be the root. Now we divide the matrix into two parts. The first having the values of first dimension less than the root and the other having values greater than the root. Again we find the medians for the second dimension in both parts and assign them to be the children of the root. This process is repeated till all the descriptors have been parsed. We save the tree in a k-by-3 matrix form. The first column is for index to the original k-by-129 matrix and the other two columns are for children.

### *D] Matching features and thresholding*

The user can select matching approach, patch size, overlap size and bin size for k-d tree. To eliminate false results various thresholds are used. Firstly, we do not consider matches where the ratio of the least distance to the second least distance is greater than 0.8. This threshold was proposed by D. Lowe to eliminate false matches. Moreover, we neglect matches where the ratio of number of highest matches to the number of second highest matches is greater than 0.6. Fair results could only be expected if difference between the number of highest matches and the second highest matches is considerable. Furthermore, we consider a patch based approach in which the descriptors in that patch are considered. The patch is moved over entire image with considerable overlap. The largest ratio of number of highest descriptor matches to the total descriptors of the entire image, is selected as the winner.

## V. Results

Tests were performed for a dataset of up to 120 images (22 teams). Due to memory and time constraints, we are showing here the results for a smaller dataset which includes 6 hockey teams having 1 to 3 logo images each. Best results were returned by a linear search as expected. Certain values were fixed as:

patch size = 90

patch increment size = 9 (90% overlap)

Note that the second and third highest patches were the neighbouring patches to the winning patch but we only show here the winning patch. Moreover, there are few similarities between these logos but the algorithm performed quite well distinguishing each team.


Figure 8: The green patches show the winning patch from linear searching. From left to right and top to bottom the identified teams are ‘Calgary Flames, Vancouver Canucks, Ottawa Senators, Toronto Maple Leafs and Montreal Canadiens’

By using k-d tree for searching, the winning patch heavily depends on the bin size and the number of image logos in the dataset. The larger bin sizes were able to detect the correct patches similar to the linear searching. For example the *ottawa senators* player could not be identified when bin size was 3 but for bin size 9, the patch was correctly identified.

## VI. Future Scope

A simple extension of this project would be by using SURF(Speeded Up Robust Features) and FREAK(Fast Retina Keypoint) for feature detection and matching and compare results with SIFT.
The project can be extended to include more images types and an image based search. By moving towards biological feature detectors and a better way to represent descriptors for an efficient matching algorithm, this method could be extended to include more real life objects in varying conditions of lighting, orientation, and scaling. A further extension is to be able to identify people too with the help of face recognition algorithms along with objects and combine the generated tags using natural language tools to describe the components/actions in an image according to their placement in it, in a simple language.

## References

[1] Lowe, David G. "Distinctive image features from scale-invariant keypoints." *International journal of computer vision* 60.2 (2004): 91-110.

[2] Beis, Jeffrey S., and David G. Lowe. "Shape indexing using approximate nearest-neighbour search in high-dimensional spaces." *Computer Vision and Pattern Recognition, 1997. Proceedings., 1997 IEEE Computer Society Conference on*. IEEE, 1997.

[3] Lowe, David G. "Object recognition from local scale-invariant features." *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*. Vol. 2. Ieee, 1999.

[4] Lowe, David G. SIFT demo program (Version 4, July 2005)
http://www.cs.ubc.ca/~lowe/keypoints

[5] Vedaldi, Andrea, ”Scale Invariant Feature Transform (SIFT)”, 14 Nov. 2013, http://www.vlfeat.org/api/sift.html