A Mini Project on

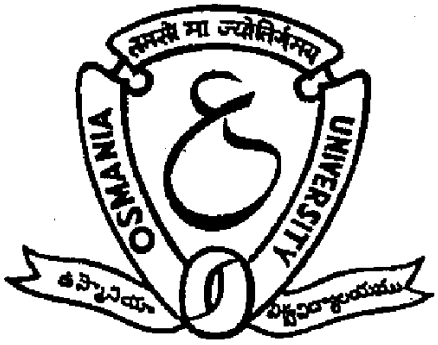
FEATURE EXTRACTION AND MATCHING USING BRISK

By

S.MOUNICA (1005-10-735025)

V.S.V.RAMA MADHURI (1005-10-735033)

S.SNEHA (1005-10-735038)



DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

UNIVERSITY COLLEGE OF ENGINEERING

OSMANIA UNIVERSITY

MAY-JULY 2013

CERTIFICATE

This is to certify that the mini project entitled "FEATURE EXTRACTION AND MATCHING USING BRISK" is a bonafide work done and submitted by S. MOUNICA (1005-10-735025), V.S.V. RAMA MADHURI (1005-10-735033), S. SNEHA (1005-10-735038), as a part of Industrial Attachment Programme during the summer vacation from MAY-JULY 2013.

Certified further, that to the best of our knowledge the work presented in this report has not been submitted to any other University or Institution for the award of any degree.

Dr. P. Chandra Sekhar
Head,
Department of ECE,
UCE, OU,
Hyderabad.

Mr. P. Balakrishna Reddy
Computer Vision Architect,
Thinc Innovations,
Hyderabad.

ACKNOWLEDGEMENTS

This project work entitled "FEATURE EXTRACTION AND MATCHING USING BRISK" has been carried out due to our special interest while pursuing Bachelor of Engineering(B.E) degree at Dept. of Electronics and Communication Engineering, University College of Engineering, Osmania University during the summer vacation. We would like to convey our regards to the special people for their guidance and encouragement in motivating us to take up this project.

We convey our sincere thanks to **Mr. P. Balakrishna Reddy**, Computer Vision Architect, Thinc Innovations, Hyderabad, for his masterly supervision through all the phases of our project with his valuable suggestions.

We would like to extend our sincere gratitude to Dr. P. Chandra Sekhar, Head, Department of Electronics and Communication Engineering, UCE, OU and Dr. P. Laxminarayana, Senior Scientist, NERTU, OU, for their support and encouragement throughout the project.

ABSTRACT

Effective and efficient generation of keypoints from an image is a well-studied problem in the literature and forms the basis of numerous Computer Vision applications. The aim of our project is to develop a code for detecting, extracting and matching image keypoints effectively using BRISK in OpenCV and Matlab.

The frames from a video sequence are read. The detection of keypoints is done using **FAST** (Features from Accelerated Segment Test) detector. It classifies a point as a corner based on its intensity value. Once features have been detected, extraction is done. Local image patch around the features are extracted and are shortlisted. The shortlisted features are also called as descriptors. This is carried out by **BRISK** extractor. It is speed, efficient and compact. BRISK (Binary Robust Invariant Scalable Keypoints) extractor relies on an easily configurable circular sampling pattern from which it computes brightness comparisons to form a binary descriptor string.From the extracted features, Feature Matching is to be done. It is the comparison of two sets of feature descriptors obtained from different images. Hamming distance method is used, where the Euclidian distance between the points is taken into consideration.

BRISK solves the classic Computer Vision problem of detecting, describing and matching image keypoints with minimum computational effort. This concept can also be extended for the pose estimation of the camera in Augmented Reality applications where there is no prior knowledge of the scene. It offers the quality of high-end features and can also be applied for tasks with hard real-time constraints.

CONTENTS

1. INTRODUCTION

1.1. Introduction

Decomposing an image into local regions of interest or 'features (keypoints)' is a widely applied technique in Computer Vision and is used to alleviate complexity while exploiting local appearance properties. Many applications rely on the presence of stable, representative features in the image, driving research and yielding a plethora of approaches to this problem.

The ideal keypoint detector finds salient image regions such that they are repeatedly detected despite change of viewpoint; more generally it is robust to all possible image transformations. Similarly, the ideal keypoint descriptor captures the most important and distinctive information content enclosed in the detected salient regions, such that the same structure can be recognized if encountered. Moreover, on top of fulfilling these properties to achieve the desired quality of keypoints, the speed of detection and description needs also to be optimized to fit within the time constraints of the task at hand.

The inherent difficulty in extracting suitable features from an image lies in balancing two competing goals: high quality description and low computational requirements. The method befitting the above requirements is BRISK. As suggested by the name, the method is rotation as well as scale invariant to a significant extent, achieving performance comparable to the state-of-the-art while dramatically reducing computational cost.

1.2. Project Description

This paper outlines a method for generating keypoints from an image, structured as follows:

- ***Keypoint detection***: Points of interest are identified across both the image and scale dimensions using saliency criterion. In order to boost efficiency of computation, keypoints are detected in octave layers of the image pyramid as well as in layers in-between. The location and the scale of each keypoint are obtained in the continuous domain via quadratic function fitting.

- ***Keypoint description***: A sampling pattern consisting of points lying on appropriately scaled concentric circles is applied at the neighborhood of each keypoint to retrieve gray values:

processing local intensity gradients, the feature characteristic direction is determined. Finally, the oriented BRISK sampling pattern is used to obtain pairwise brightness comparison results which are assembled into the binary BRISK descriptor.

- ***Keypoint Matching***: Once generated, the BRISK keypoints can be matched very efficiently using Hamming distance method for binary descriptors.

1.3. Organization of the report

This mini project presents the details of "Feature Extraction and Matching using BRISK". It also portrays the background information needed to understand the design and the results of the Project. The project report is organized in chapters as follows:

Chapter 2 presents the first phase of this project, i.e. the detection of keypoints by FAST algorithm.

Chapter 3 deals with keypoint extraction by BRISK algorithm and construction of feature descriptors or vectors. The corresponding results of this stage are presented.

Chapter 4 gives a detailed explanation of feature matching technique, the last phase in the project. Furthermore, a flowchart and the results are presented.

Chapter 5 gives the conclusion of the project. It also specifies the future scope of this mini project.

2. FEATURE DETECTION

In the detection process, our main aim is to detect the keypoints with robust corners and scale invariant points. With the focus of efficiency to computation, FAST detector is used rather than other detectors like SIFT, SURF etc.

2.1. FAST

Features from Accelerated Segment Test (FAST) is an algorithm defined for identifying interest points in an image. An interest point in an image is a pixel which has a well-defined position and can be robustly detected. Interest points have high local information content.

There are several well established algorithms like Harris & Stephens corner detection algorithm, SUSAN corner detector. The reason behind the work of the FAST algorithm was to develop an interest point detector for use in real time frame rate applications which have limited computational resources.FAST corner detector is very suitable for real-time video processing application because of high-speed performance.Keypoints are detected based on the intensity at each pixel using Scale Space Pyramid.

Feature detection using FAST:

The FAST corner detection algorithm is given below:

- Given an image,select a pixel ‘p’ in the image. Assume the intensity of this pixel to be Ip. This is the pixel which is to be identified as an interest point or not.
- Set a threshold intensity value T, (say 20% of the pixel under test).
- Consider a circle of 16 pixels surrounding the pixel p.
- ‘N’ contiguous pixels out of the 16 need to be either above or below Ip by the value T, if the pixel needs to be detected as an interest point.

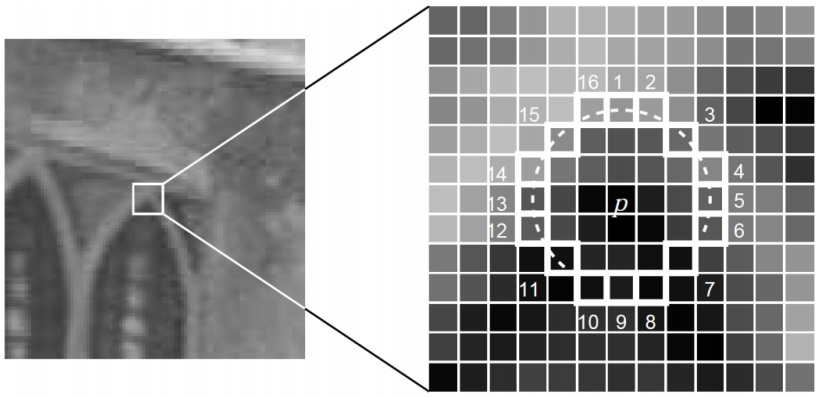


Figure 1

Figure 1 represents the interest point under test and the 16 pixels on the circle.

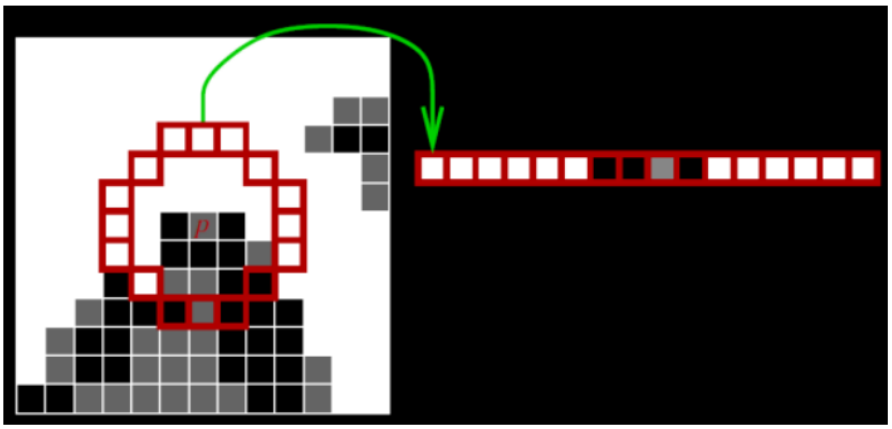
In order to make the algorithm fast, first compare the intensity of pixels 1,5,9,13of the circle with Ip. As evident from the figure above, at least three of these four pixels should satisfy the threshold criterion so that the interest point will exist.

- If at least three of the four pixel values - 1,5,9,13 are not above or below Ip + T, then p isnot an interest point (corner). In this case reject the pixel p as a possible interest point. Else if atleast three of the pixels are above or below Ip + T, then check for all 16 pixels and check if 12 contiguous pixels fall in the criterion.
- Repeat the procedure for all the pixels in the image.

There are a few limitations to the algorithm.For N<12, the algorithm does not work very well in all cases because when N<12, the number of interest points detected are very high.

In order to deal with these issues, a machine learning approach has been added to the algorithm.

Machine learning approach:



- Select a set of images for learning.Select a set of images for training.
- In every image run the FAST algorithm to detect the interest points by taking one pixel at a time and evaluating all the 16 pixels in the circle.
- For every pixel 'p', store the 16 pixels surrounding it, as a vector. The 16 values surrounding pixel p stored in vector form.
- Repeat this for all the pixels in all the images. This is the vector p which contains all the data for training.
- Each value (one of the 16 pixels, say x) in the vector, can take three states. Darker than p, lighter than p or similar to p.
- Mathematically,

$$S_{p\to x} = \begin{cases} d, & I_{p\to x} \le I_p - t \quad \text{(darker)} \\ s, & I_p - t < I_{p\to x} < I_p + t \quad \text{(similar)} \\ b, & I_p + t \le I_{p\to x} \quad \text{(brighter)} \end{cases}$$

Where Sp-x is the state, Ip-x is the intensity of the pixel x and t is a threshold.

- Depending on the states the entire vector P will be subdivided into three subsets, Pd, Ps, Pb.
- Define a variable Kp which is true if p is an interest point and false if p is not an interest point.

Scale space keypoint detection:

In order to achieve invariance to scale which is crucial for high-quality keypoints, we implement this technique by searching for maxima not only in the image plane, but also in scale-space using the FAST scores as a measure for saliency. Despite discretizing the scale axis at coarser intervals than in alternative high-performance detectors, the BRISK detector estimates the true scale of each keypoint in the continuous scale-space.

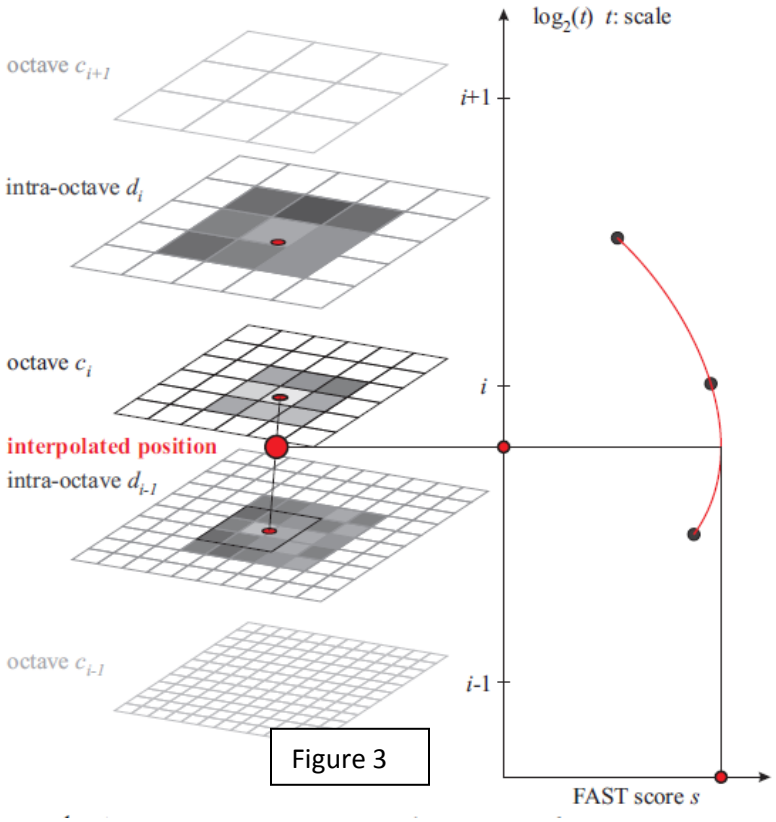


Figure 3

In the BRISK framework, the scale-space pyramid layers consist of n octaves c_i and n intra octaves d_i, for I ={0,1,...,n− 1} and typically n=4. The octaves are formed by progressively half sampling the original image (corresponding to c_0). Each intra-octave d_i is located between layers c_i and c_{i+1}.

The first intra octave d_o is obtained by down sampling the original image Co by a factor of 1.5, while the rest of intra octave layers are derived by successive half sampling. Therefore, if t denotes scale then $t(c_i)=2^i$ and $t(d_i)=2^i \cdot 1.5$.

In BRISK, we mostly use the 9-16 mask, which requires atleast 9 consecutive pixelsinthe16-pixel circle to either be sufficiently brighter or darker than the central pixel for the FAST criterion to be fulfilled.

Initially, the FAST 9-16 detector is applied on each octave and intra octave separately using the same threshold T to identify potential regions of interest. Next, the points belonging to these regions are subjected to non-maxima suppression in scale-space.

Considering the image saliency as a continuous quantity across the image and also along the scale dimension, we perform a sub pixel and continuous scale refinement for each detected maximum. In order to limit complexity of the refinement process, we first fit a 2D quadratic function in the least squares sense to each of the three score patches (as obtained in the layer of the keypoint, the one above and the one below) resulting in the three sub-pixel refined saliency maxima. In order to avoid re-sampling, we consider a 3x3 score patch on each layer. Next, these refined scores are used to fit a 1D parabola along the scale axis yielding the finalscore estimate and scale estimate at its maximum. As a final step, were interpolate the image coordinates between the patches in the layers next to the determined scale.

3. FEATURE EXTRACTION

Feature extraction is a method in which the desired portions or shapes (features) of a digitalized image or video stream are detected using various algorithms. One of the most efficient algorithms is the BRISK.

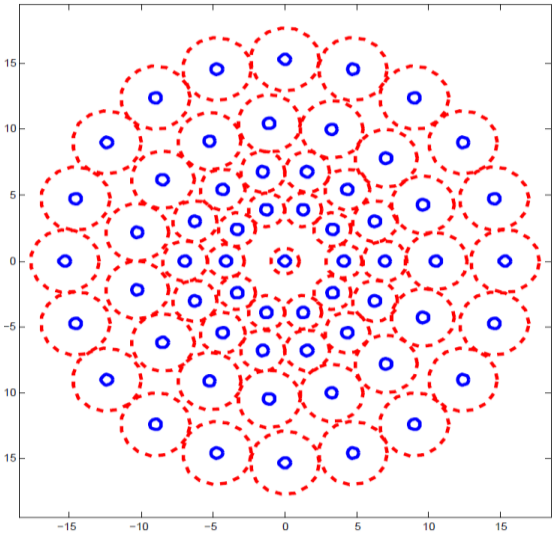
BRISK(Binary Robust Invariant Scalable Keypoints)

The BRISK descriptor consists of a binary string by concatenating the results of simple brightness comparison tests.

Here, we identify the characteristic direction of each keypoints to allow for orientation-normalized descriptors and hence achieve rotation invariance which is the key to general robustness. Also, we carefully select the brightness comparisons with the focus on maximizing descriptiveness.

3.1. Sampling pattern and Rotation estimation

The key concept of BRISK detector makes use of a pattern used for the sampling the neighborhood of the keypoint. The sampling pattern with N locations, equally spaced on circles concentric with the keypoints is shown below.



This is the BRISK sampling pattern with N=60 points. The small blue circles denote the sampling locations. The bigger red dashed circles are drawn at a radius σ corresponding to the Gaussian kernel used to smooth the intensity values at the sampling points. This pattern applies to a scale of t=1.

Consider a small patch centered about a keypoint, which is described as a binary string. First thing, take a sampling pattern around the keypoint. For example the points spread on the concentric circles.

Next, choose say 512 pairs of points on this sampling pattern. Now for all the pairs of points, compare the intensity value of the first point in the pair with the intensity value of the second value in the pair. If the first value is greater than the second, write '1' in the string otherwise '0'. After going over all 512 pairs, we'll have 512 characters string composed of '1' and '0' that encoded the local information around the keypoint.

In order to avoid aliasing effects when sampling the image intensity of a point $\mathbf{p_i}$ in the pattern, we apply Gaussian smoothing with standard deviation $\boldsymbol{\sigma_i}$ which is proportional to the distance between the points on the respective circle. The Gaussian smoothing operator is a 2-D convolution operator that is used to 'blur' images and remove detail and noise. In 2-D, an isotropic (i.e. circularly symmetric) Gaussian has the form:

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

There are N points on the circular pattern. The total number of sampling point pairs will be $N\cdot(N-1)/2$. For positioning and scaling the pattern accordingly for a particular key point k in the image, let us consider one of thesampling point pairs, say $(\mathbf{p_i},\mathbf{p_j})$. The smoothed intensity values at these points are $\mathbf{I}(\mathbf{p_i},\boldsymbol{\sigma_i})$ and $\mathbf{I}(\mathbf{p_j},\boldsymbol{\sigma_j})$ respectively. They are used to estimate the local gradient.

$$g(p_i,p_j) = (p_j - p_i)\cdot\frac{I(p_j,\sigma_j) - I(p_i,\sigma_i)}{\|p_j - p_i\|^2}$$

Considering the set A with all sampling pattern pairs,

$$\mathcal{A} = \{(\mathbf{p}_i, \mathbf{p}_j) \in \mathbb{R}^2 \times \mathbb{R}^2 \mid i < N \wedge j < i \wedge i, j \in \mathbb{N}\}$$

We define a set of short distance pairings 'S' and another subset of long distance pairings 'L'

$$S = \{(p_i, p_j) \in A \mid \|p_j - p_i\| < \delta_{max}\} \subseteq A$$

$$L = \{(p_i, p_j) \in A \mid \|p_j - p_i\| > \delta_{min}\} \subseteq A$$

The threshold distances are set to $\delta_{max} = 9.75t$ and $\delta_{min} = 13.67t$ (t is the scale of k). The long distance pairs are used for computing overall characteristic pattern direction. This is done based on the assumption that local gradients annihilate each other and are thus not necessary in the global gradient determination. Iterating the point pairs in L, we estimate the overall characteristic pattern direction of the keypoint k to be:

$$\mathbf{g} = \begin{pmatrix} g_x \\ g_y \end{pmatrix} = \frac{1}{L} \cdot \sum_{(\mathbf{p}_i, \mathbf{p}_j) \in \mathcal{L}} \mathbf{g}(\mathbf{p}_i, \mathbf{p}_j).$$

This gradient value is used to find the angle by which the sampling pattern is rotated around the keypoint 'k'. Let it be α.

$$\alpha = \arctan2(g_y, g_x)$$

3.2. Building Descriptor:

The rotated short distance pairings are used to build the binary descriptor 'd_k'.The bit vector 'd_k' is assembled by performing all short distance intensity comparisons of point pairs, $(p^a,p^a) \in S$ such that each bit 'b' corresponds to:

$$b = \begin{cases} 1, & I(\mathbf{p}_j^\alpha, \sigma_j) > I(\mathbf{p}_i^\alpha, \sigma_i) \\ 0, & \text{otherwise} \end{cases}$$
$$\forall (\mathbf{p}_i^\alpha, \mathbf{p}_j^\alpha) \in \mathcal{S}$$

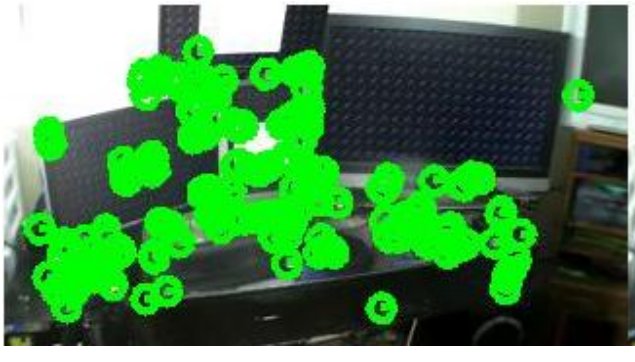
Firstly, BRISK uses a deterministic sampling pattern resulting in a uniform sampling-point density at a given radius around the keypoint.Furthermore, BRISK uses dramatically fewer sampling-points than pairwise comparisons (i.e. a single point participates in more comparisons), limiting the complexity of looking- up intensity values. Finally, the comparisons here are restricted spatially such that the brightness variations are only required to be locally consistent. With the sampling pattern and the distance thresholds as shown above, we obtain a bit-string of length 512.

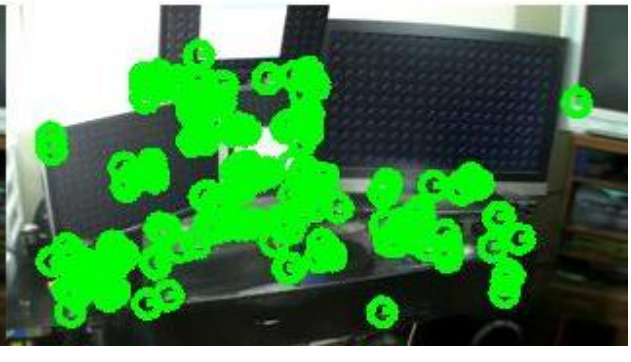
The bit string of BRIEF64 also contains 512 bits. Thus matching for a descriptor pair will be performed equally fast by definition.

Original frames 1&2 from video sequence

Frame 1

Frame 2

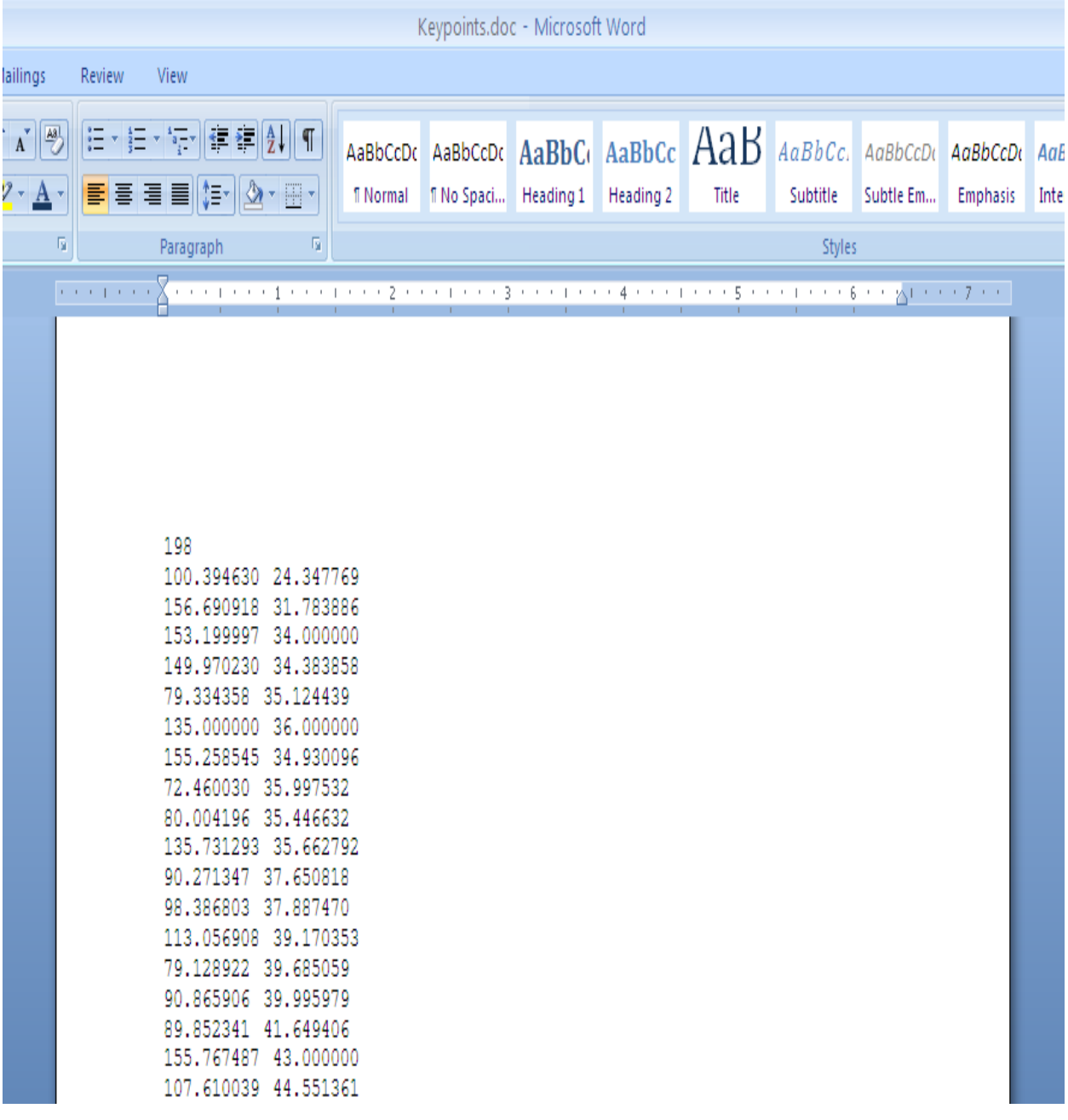




Number of keypoints in frame1 = 198

Number of keypoints in frame2 = 187

3.3. Result of Feature Extraction



4. FEATURE MATCHING

Feature matching is the comparison of two sets of feature descriptors obtained from different images to provide point correspondences between images.

4.1. Types of Feature Matchers

- Brute Force
- FlannBased
- KnnMatch
- Hamming distance for binary features

Brute force matcher:

For each descriptor in the first set, this matcher finds the closest descriptor in the second set by trying each one. This descriptor matcher supports masking permissible matches of descriptor sets.

*class***BFMatcher**: *public***DescriptorMatcher**

The **BFMathcher** will only return consistent pairs. Such technique usually produces best results with minimal number of outliers when there are enough matches.

Flann Based Matcher:

FLANN stands for Fast Library for Approximate Nearest Neighbors. This matcher trains **flann::Index_** on a train descriptor collection and calls its nearest search methods to find the best matches. So, this matcher may be faster when matching a large collection than the brute force matcher. FlannBasedMatcher does not support masking permissible matches of descriptor sets because **flann::Index** does not support this.

class **FlannBasedMatcher** : *public* **DescriptorMatcher**

K-NN MATCHER:

K-nearest neighbor algorithm (*k*-NN) matches objects based on the k closest training examples in the feature space. The function is only approximated locally and all computation is deferred until classification. An object is classified by a majority vote of its neighbors, with the object being assigned to the class most common amongst its *k* nearest neighbors.

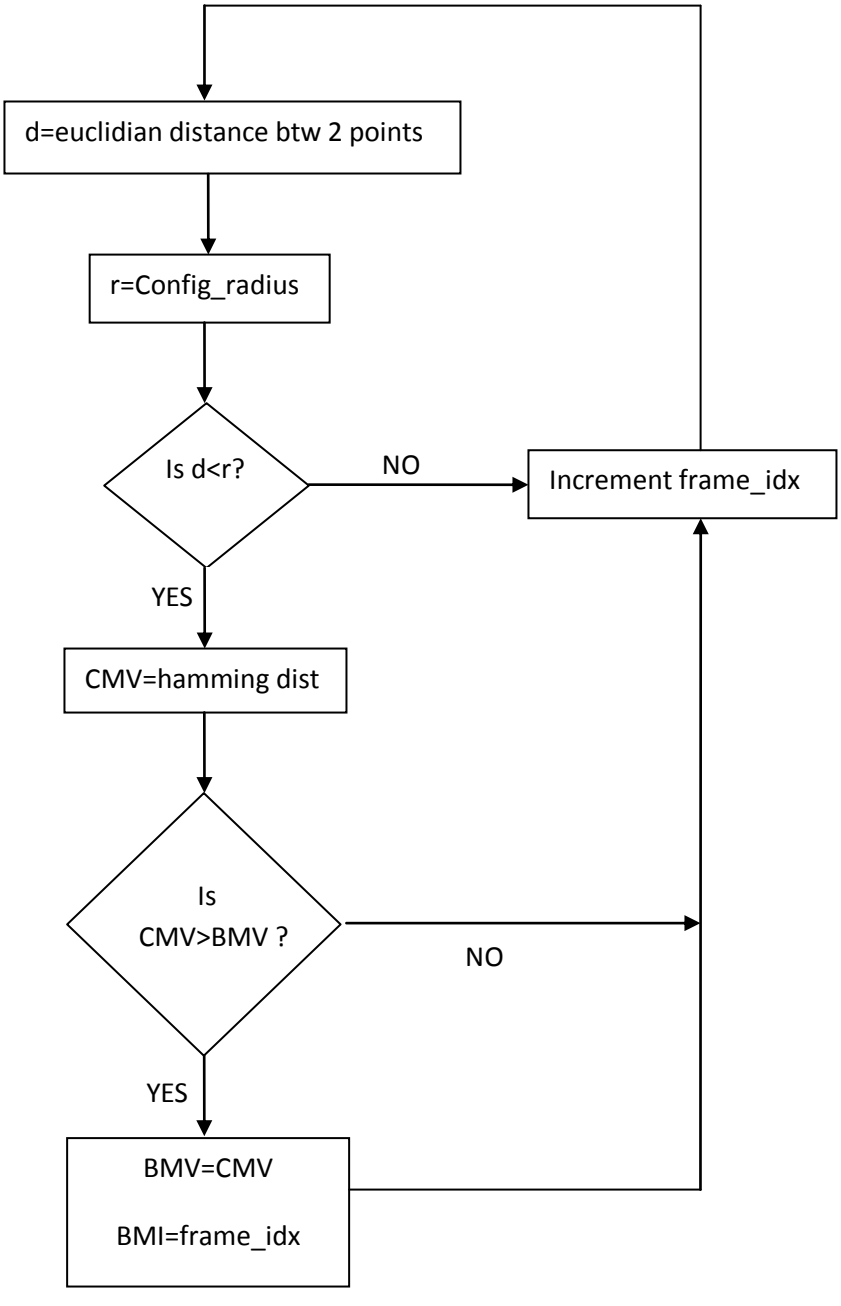
DescriptorMatcher: :knnMatch

Hamming Distance for binary features:

Hamming distance between two strings of equal length is the number of positions at which corresponding signals are different. The function hamming_distance() computes the Hamming distance between two strings of equal length, by creating a sequence of Boolean values indicating mismatches and matches between corresponding positions in the two inputs, and then summing the sequence with False and True values being interpreted as zero and one. For binary strings *a* and *b* the Hamming distance is equal to the number of ones (population count) in *a* XOR *b*.

Since Hamming distance method is efficient, it has been implemented.

4.2. Flowchart for Matching process



4.3. Explanation

In feature matching, keypoints from frame1 are matched with those in frame2.

Create a file keypoints.doc and store all the keypoints from feature extraction step into it. Bestmatchindex (BMI), currentmatchvalue (CMV) and bestmatchvalue (BMV) are initialized where bestmatchindex indicates the best matching keypoint index of frame1 in frame2, currentmatchvalue indicates hamming distance between keypoints and bestmatchvalue indicates the maximum hamming distance.

Euclidean distance (d) between keypoints of frame 1 and 2 are calculated using the formula

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} .$$

This Euclidean distance is compared with configuration radius (r) to ensure that matched keypoints lie within the locality. Here configuration radius is 30 and is constant during the whole process.

If the condition is satisfied, then hamming distance between the keypoints is found out which will be explained later. Since the aim of this step is to find out the best match, this hamming distance i.e. the current match value will be compared with best match value and based on that condition, the step will be repeated and best match of keypoint of frame1 is identified in frame2.

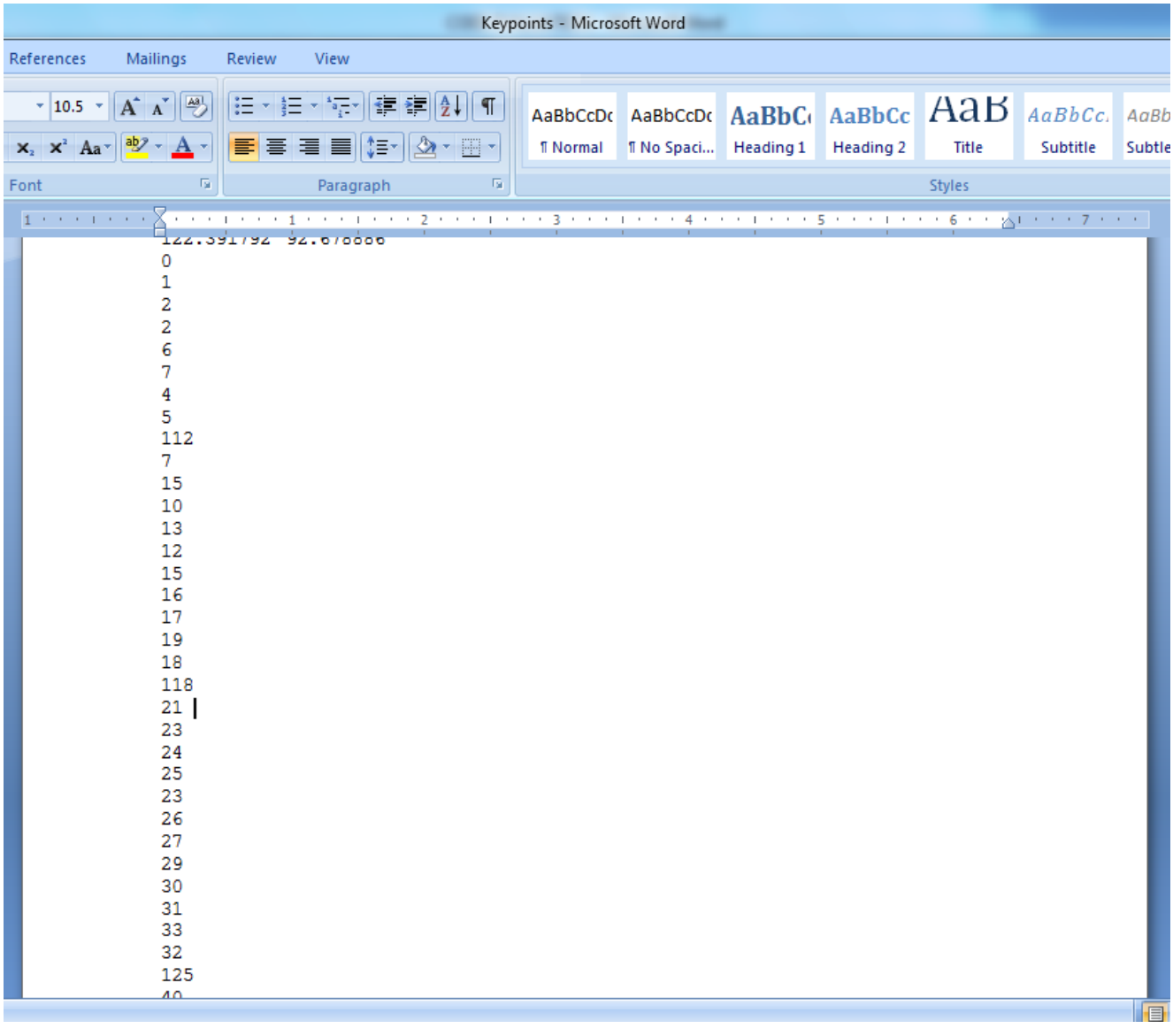
Now, this process will be repeated for all the frames and corresponding match indices will be identified and stored in keypoints.doc.

Hamming Distance:

To find the matching between two images, the keypoint descriptors of 1st and 2nd images are XORed and the number of similar bits is obtained. This hamming distance helps in identifying the best match for each keypoint. When initial conditions are satisfied, maximum hamming distance between two keypoints tells that they are the best matches.

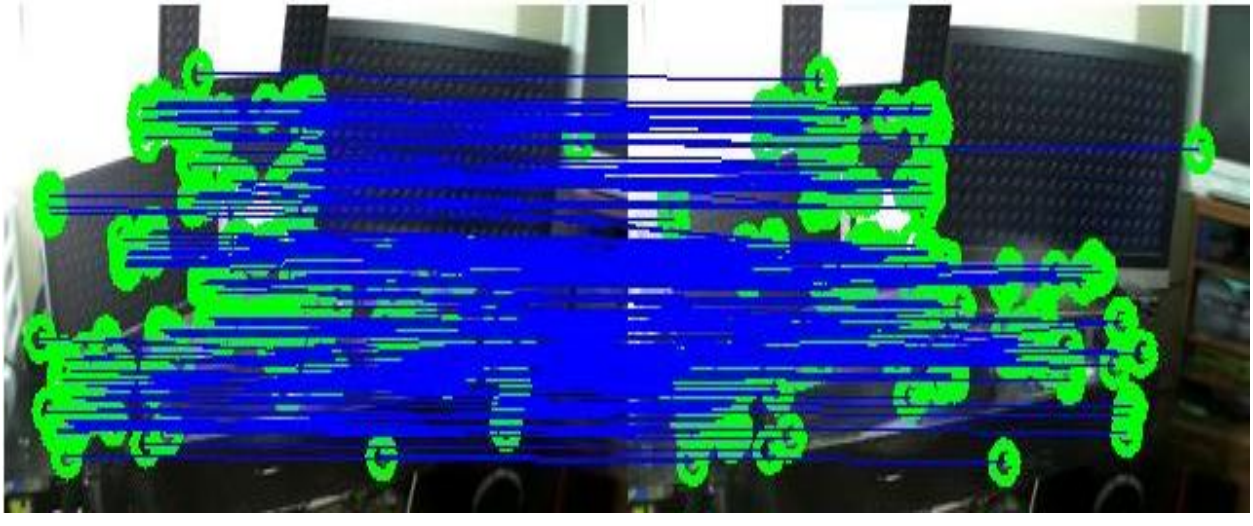
For example, let us consider a keypoint in frame1 and find its best match in frame2. In the figure, the 2 keypoints in frame2 satisfy the initial conditions of Euclidean distance with keypoint in frame1. But to find the best match, we will find the hamming distance and the set of keypoints of frame1 and frame2 which has the maximum hamming distance will be considered as the best match.

4.4. Result of Feature Matching



Here, it can be seen that keypoint1 of frame1 is not matched with any keypoint in frame2 and is given a value ‘0’. Keypoint2 of frame1 is matched with keypoint1 of frame2 and similarly keypoint3 to keypoint2 and so on.

Frame 1 Frame 2



The bluelines in the above figure indicate the matched points in the two consecutive frames of the video sequence.

5. CONCLUSION

5.1. Summary:

Effective and efficient generation of keypoints from an image is a well-studied problem in the literature and forms the basis of numerous Computer Vision applications. In this paper, we have presented a novel method named BRISK, which tackles the classic Computer Vision problem of detecting, describing and matching image keypoints for cases without sufficient a priori knowledge on the scene and camera poses. In contrast to well-established algorithms with proven high performance, such as SIFT and SURF, the method at hand offers a dramatically faster alternative at comparable matching performance – a statement which we base on an extensive evaluation using an established framework. BRISK relies on an easily configurable circular sampling pattern from which it computes brightness comparisons to form a binary descriptor string. The unique properties of BRISK can be useful for a wide spectrum of applications, in particular for tasks with hard real-time constraints or limited computation power: BRISK finally offers the quality of high-end features in such time-demanding applications.

5.2. Future Scope:

This paper will be extended for our main project Augmented Reality, which is a field of computer research that combines the actual scene, viewed by the user, and a virtual scene, generated by the computer, that augments the scene with additional information. Using the results of BRISK, camera pose of real world video sequence will be estimated and a virtual 3D object will be studied in detail and augmented with the video based on the pose estimated.

BIBLIOGRAPHY

- http://gilscvblog.wordpress.com/2013/08/26/tutorial-on-binary-descriptors-part-1/
- Stefan Leutenegger, Margarita Chli and Roland Y. Siegwart, Autonomous Systems Lab, ETH Z¨urich "***BRISK: Binary Robust Invariant Scalable Keypoints***"
- Francesc Moreno-Noguer, Vincent Lepetit, Pascal Fua, Computer Vision Laboratory, EcolePolytechniqueFédérale de Lausanne – 1015 Lausanne, Switzerland "***Accurate Non-Iterative O(n) Solution to the PnP Problem***"
- Vincent Lepetit and Pascal Fua, Ecole Polytechnique Fédérale de Lausanne (EPFL) "***Keypoint Recognition using Randomized Trees***"
- http://www.asl.ethz.ch/people/lestefan/personal/BRISK
- http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.371.1343
- http://miksik.co.uk/papers/miksik2012icpr.pdf