UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

Universitat Politècnica de Catalunya

Advanced Topics in Computer Vision

Practice 1: Local Descriptors

Leonardo Palacios Fidalgo
Ce Xu Zheng

March 30, 2022
Academic year 2021-2022

Contents

1 Introduction

In this document we present an implementation of the BRIEF descriptor described by Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua in their work ”BRIEF: Binary Robust Independent Elementary Features” [1]. The goal of this work is understanding the methodology's underlying principles along with its advantages and limitations. The work done follows the Local Descriptors Practice proposed by the Advanced Topics in Computer Vision course at the Universitat Politécnica de Catalunya.

We first begin by presenting a brief introduction to the methodology described in [1] in section 2. We then continue our work by presenting our implementation and characterizing BRIEF's descriptive power for image key-points in section 3. Finally, in section 5, we evaluate the BRIEF descriptor as a general patch descriptor.

2 Background: BRIEF Descriptor

Feature point descriptors have become essential in many computer vision technologies due to their capacity of uniquely and successfully describing key-points in an image. Applications such as image retrieval, object recognition, 3D reconstruction and camera localization leverage the power of descriptors to achieve their goals. With the ever growing amount of data that these application are expected to process, along with the desire to port these application to devices with more limited power and to run them in real time; academics in the computer vision community have noted the growing need for local descriptors that are fast to compute, fast to match, and that are memory efficient.

Binary Robust Independent Elementary Features, or BRIEF, are binary strings used as efficient feature point descriptors. Presented by Calonder et al in [1], BRIEF descriptors are highly discriminative and can be computed using simple intensity difference tests. In the remainder of these section, we will briefly discuss the key aspects of this methodology

2.1 Method

The approach followed by Calonder et al [1] is inspired by earlier work referenced in [2] and [3]. Calonder et al's method leverages a bit vector as a descriptor generated from an intensity test, which is computed after smoothing the image patch, by the following test τ on a patch $\mathbf{p}$ of size $S \times S$:

$$\tau(\mathbf{p}; \mathbf{x}, \mathbf{y}) := \begin{cases} 1 & \text{if } \mathbf{p}(\mathbf{x}) < \mathbf{p}(\mathbf{y}) \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

Where $\mathbf{p(x)}$ is the pixel intensity in a smoothed version of $\mathbf{p}$ at $\mathbf{x} = (u, v)^\top$.

The BRIEF descriptor is then the n_d-dimensional bitstring:

$$f_{n_d}(\mathbf{p}) := \sum_{1 \le i \le n_d} 2^{i-1} \tau(\mathbf{p}; \mathbf{x}_i, \mathbf{y}_i) \tag{2}$$

In this work, we will follow Calonder et al [1] recommendations and consider $n_d = 128$, 256, and 512. These yield good compromises between speed, storage efficiency, and recognition rate [1]. When creating these descriptors, the only choices we have to make are those relating to the kernels used to smooth the image, and the spatial arrangement of the (**x**,**y**)-pairs of the test in (1).

2.1.1 Smoothing Kernels

As previously mentioned, with the goal of reducing noise sensitivity, and to increase stability and repeatability of the descriptors, the images will be first smoothed with a Gaussian filter of $\sigma = 2$ and a corresponding discrete kernel window of size 9×9 pixels (As recommended in [1]).

2.1.2 Spatial Arrangement of the Binary Tests

Beyond the parameters chosen for the smoothing kernels, the only open choices we have in designing a BRIEF descriptor is the spatial arrangements of the binary tests. This can be achieved with an ordered and/or symmetric arrangement of pair points, e. g. a randomly sampled pairs from discrete locations of a coarse polar grid with a spatial quantization, or it can be created from random sampling form the patch. As it is observed in **??**, and supported by our results, unordered random sampling yields stronger discriminative power.

2.1.3 Hamming Distance

In addition to the ease of calculation of BRIEF descriptors, another factor that improves the speed of this method is how descriptor similarity is evaluated. In non-binary descriptor implementations the distance between to descriptors is often determined by the sum of square distance, or something to that end. BRIEF descriptors, being binary vectors, allow us to evaluate their distance to each other, or similarity, by simply applying the efficient and simple XOR operation with the two descriptors and adding the amount of ones in the resulting vector. Using this hamming distance allows satisfying real time constraints on computer vision applications.

3 BRIEF Arrangements as a Key-point Descriptor

In this exercise, we implemented the BRIEF descriptor to identify the some points in a set of known images of the same structure (a wall).

3.1 Methodology

The methodology will consist in the selection of some interesting points in one of the images and the posterior identification of those in the following images. The BRIEF descriptor should be able to identify and match most of the points of the first image with the same points in the others, despite the change of orientation and perspective.

Once the points are chosen, we have to select a window which will contain the points that describe the point. Then we will generate a random pattern for the pairs that we are going to compare and finally make the descriptor of each point that we selected from the image that we can.

The matching algorithm will consist in looking for the Hamming distance between the descriptor of one point to each point of a second image. The one of the latter image that has the lowest distance will be considered the descriptor's prediction and marked as a True positive or a False positive depending on what we previously know. I we proceed in this way for each one of the points in the first image, we will obtain the distribution of the True positive and False positive's distance.

3.1.1 Wall Dataset

The wall dataset is conformed by a set of 6 images of the same wall from different perspectives. The first image is taken from the front of the wall, and will be taken as reference to obtain the key points that we are going to match. The other 5 images are pictures taken with progressively change in the perspective, and they come with the homography transformation matrix from the first image. This is useful to know the exact position where one given keypoint from the first image falls in each one of the others, and also to know if they even belong to the latter one.

The knowledge of the exact position of the correspondent points between images will allow us to evaluate the quality of the descriptors, as we only need to prove that the descriptor match the points that we previously know that are the same.

3.1.2 Key-point Extraction

Firstly, we have to obtain the keypoints in the first wall image. To do so, we are going to utilize the Matlab *detectHarrisFeatures()* function, then we have to filter the points that lies near the edges of the image, as we will not be able to perform a BRIEF descriptor around those points. And finally we will take the one thousand Stronger points as keypoints that we are going to test 1.

Once we obtained the keypoints, we can obtain the position of the same points in each one of the other images, as long as they belong to the same plane, with the homography matrix.

The homography matrix is a 3 by 3 matrix that contains the information not only for rotations and translations of the real points inside the image, but also codifies the perspective in which the picture is taken. We can use it to transform points of the first image to points of the second one by multiplying the matrix to the point extended in the projective space (adding a one as third dimension):

$$\begin{bmatrix} x' \\ y' \\ n \end{bmatrix} = \mathcal{H} \times \begin{bmatrix} x^1 \\ y^1 \\ 1 \end{bmatrix}, \tag{3}$$

where the $\mathcal{H}$ represents the holographic matrix, (x^1, y^1) are the coordinates of the point in the first image and (x', y', n) is a non normalized point in the projective space that represent the coordinates of the point in the second image. To obtain the position of the point in the second image, we have to

Figure 1: Selection of keypoints form the front view of the wall

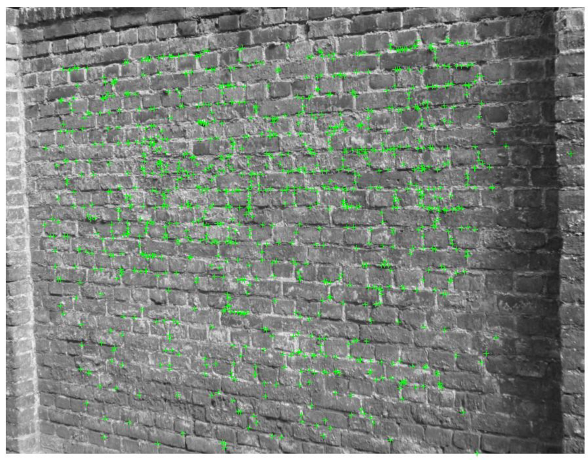


Figure 2: The same keypoints in the third photo of the wall

normalize the vector –divide the three coordinates by the third component to obtain a one in this last position, it is called a reprojection to the affine plane, and note that we are never going to obtain a zero in this last dimension as they only corresponds to points in the infinite in the projective space.

$$\begin{bmatrix} x^2 \\ y^2 \end{bmatrix} = \begin{bmatrix} x'/n \\ y'/n \end{bmatrix}, \tag{4}$$

This will finally return the position of the same key points in each one of the other pictures of the wall, for example the third one

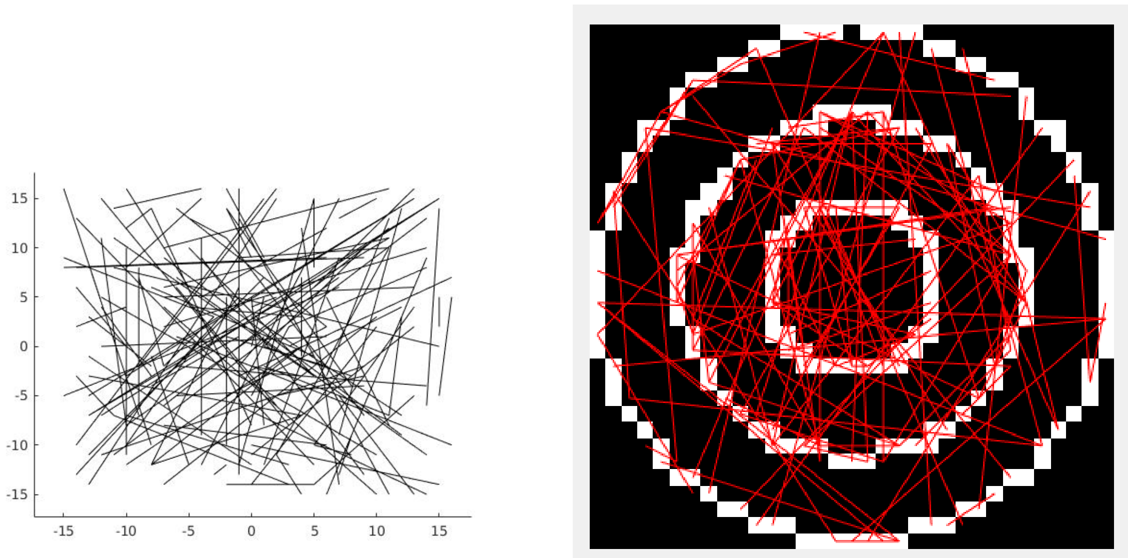
3.1.3 Gaussian Filtering

Once we have chosen the keypoints of the first picture and localized them in the other images, we can start to perform the BRIEF descriptor. As stated in the section 2.1.1, we have to perform a Gaussian filtering before performing the BRIEF descriptor. We have to decide wether we improve the sensitivity to the noise with a greater variance of the Gaussian filter or we prefer to preserve the pixel information with a smaller variance.

In the BRIEF paper[1], they proposed a $\sigma = 2$, and we also checked that increasing this value will progressively throw worse results (lower hit rate) in the identification of the points of the first image

Gaussian σ	2	3	4	1.5	1
hit_rate%	95.08	95.58	94.68	95.08	91.97

Table 1: Hit rate with the same descriptor(pairs of points, and number of them) and different Gaussian variance.



(a) Random descriptor

(b) New selected descriptor

Figure 3: Descriptors

with respect to the second one.

As we can see in the Table1, the proposed σ lies next to the sweet spot for maximum acuracy – note that the higher value in one single example does not mean that $\sigma = 3$ could be a better value in the general case.

3.1.4 BRIEF Descriptor Definition

For this exercise, we have chosen a random generated pattern of pairs with a uniform distribution along all the patch like the first proposal of the BRIEF authors [1], and the second pattern proposal will be given by choosing some random points in three different concentric circles.

The outer circle will have the radius of a half of the patch size and one third of the points, the middle circle is of a radius of a third of the patch size and half of the points (as we consider it the best circle), and the inner circle with a radius of a sixth of the patch size and the rest of the points.

We can see here the two descriptors' pairs 3.

We hypothesize that if we compare points with the same distance to the center, we can obtain a better robustness to rotations and therefore a pretty good descriptor.

3.1.5 BRIEF Descriptor Evaluation

Once we have chose the pattern of the descriptor (a number of points pairs in a patch around a given keypoint), we compare for each pair if the first pixel is brighter than the second (or equivalently the opposite). This set of booleans will be the feature BRIEF descriptor of the particular keypoint in the image.

3.1.6 Descriptor Similarity and Recognition Rate Testing

After obtaining the descriptor of each valid point, we can compare the distances between them with the Hamming distance, that computes the number of differences between two sets of booleans as distance.

Which point of the second image matches with on point of the first image, we are going to compute the hamming distance of the descriptor of this particular point to each one of the points of the second picture and pair this point to the first one. I the prediction is correct, we are going to consider it a True positive and otherwise a False positive.

If we perform this algorithm for each point of the chosen keypoints in the first image to each one of the other images, we are going to obtain the measure of all the True positive and False positive predictions and therefore compute the hit/cognition rate $= TP/N$ where N stand for the number of possible total matches.

4 Results

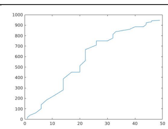
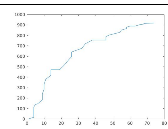
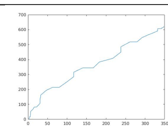
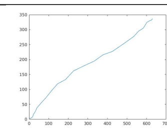
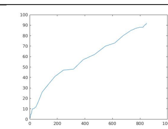
BRIEF descriptor	1 to 2	1 to 3	1 to 4	1 to 5	1 to 6
Correct hits	947	919	623	337	92
Correct hits	49	74	348	630	846
Hit rate %	95.08	92.55	64.16	34.85	09.81
Roc curve					

Table 2: New descriptor's performance

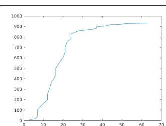
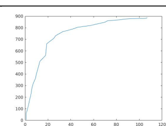
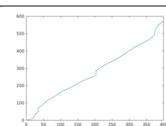
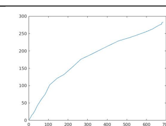
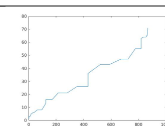
New descriptor	1 to 2	1 to 3	1 to 4	1 to 5	1 to 6
Correct hits	933	886	572	283	71
Correct hits	63	107	399	684	867
Hit rate %	93.67	89.22	58.91	29.27	07.57
Roc curve					

Table 3: New descriptor's performance

In the Tables 2 and 1, we can observe that the the given distribution also gives a pretty good hit rate, but falls always behind the uniform random distribution. As we expected, the greater the change of perspective, the lower the hit rate.

The Roc curves that we have presented are not very representative due to the lack of True positives in the lower precision examples and the lack of False negatives in the higher precision image. This lack of statistics along side with the scaling factor makes difficult to see correctly the distribution of the matching points, but we can observe in the distribution of the distances of True positives are smaller than the False positives 4.

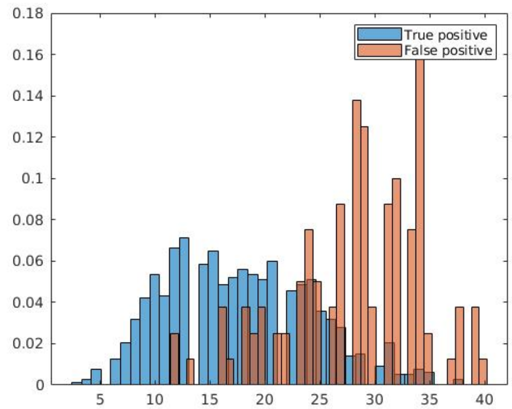


Figure 4: Distribution of distances of the descriptors by True and False positives of the random descriptor between the first and the third wall image.

5 BRIEF Arrangements as a General Patch Descriptor

We would now like to evaluate the descriptive power of BRIEF arrangements as a general patch descriptors. To this end, we will compute associated BRIEF descriptors of different test models and evaluate the similarity of the descriptors across models.

5.1 Methodology

5.1.1 Traffic Sign Models

For this test, we will use the set of traffic model patterns shown in fig [**?**]. We will then use BRIEF descriptors to evaluate how these descriptors would work for describing and discriminate between such model. As in section 3, we will also study how the descriptor size and the spatial arrangement of our pair points fair in the overall descriptor performance.

Figure 5: Traffic sign models.

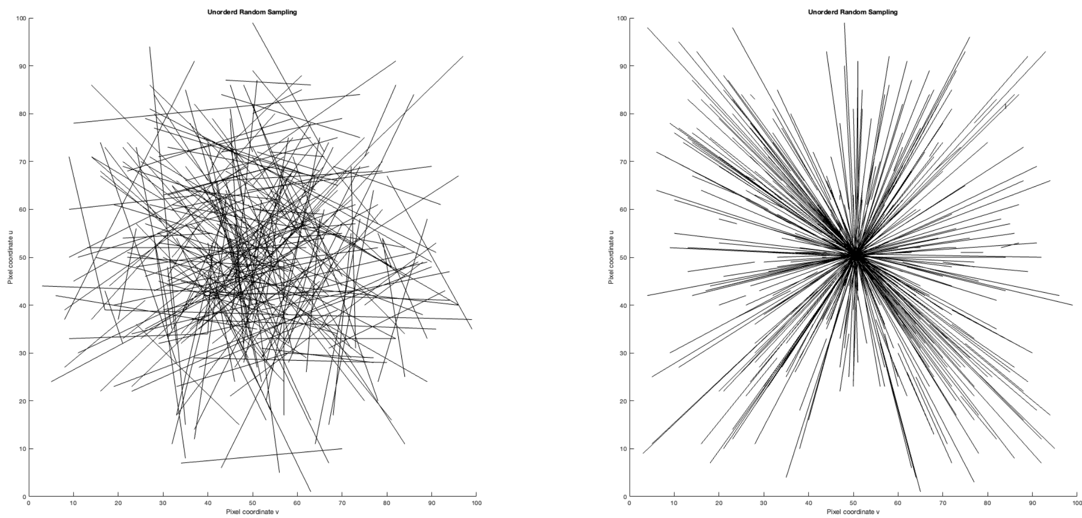
Note that a handful of the original models were not the 100x100 pixels size most models are, but instead 88x199 or 87x100 pixels in size. These models were resized to 100x100 for convenience.

5.1.2 Gaussian Filtering

As in the previous section, the models were smoothed with a Gaussian filter with the goal of reducing noise sensitivity, and to increase stability and repeatability of the descriptors. The models wee smoothed with a Gaussian filter of $\sigma = 2$ and a corresponding discrete kernel window of size 9×9 pixels (As recommended in [1]).

5.1.3 BRIEF Descriptor Definition

As mentioned before, the design aspects of a BRIEF descriptor that are flexible, are the spatial arrangement of the pair points and the size of the descriptor. With this in mind, we will evaluate the BRIEF arrangements according to their size and pair point selection. In particular we will evaluate the distance between the model descriptors for descriptor sizes of 128, 256, 512, 1024 with a random ordered pair points, and with pair points sampled from $(\mathbf{X}, \mathbf{Y}) \sim$ i.i.d. Gaussian $\left(0, \frac{1}{25}S^2\right)$. The random ordered pair points consist on pair of points that lie in the same line and include the model center. Examples of the obtained set of pair points can be visualized in figure **??**.



(a) Random Unordered Sampling. (b) Random Ordered Sampling.

Figure 6: Examples of the two distributions sampled to select the pair points of a BRIEF-256 descriptor.

5.1.4 Model BRIEF Descriptor Evaluation

After having designed our descriptor, i.e selected the descriptor size and the spatial arrangement of the (**x**,**y**)-pairs of the test in (1), we next calculate the descriptor (bit vector) for each model in fig 5. After having done this, we can evaluate the hamming distance among model descriptors and draw conclusions on the efficacy of the BRIEF descriptor for model characterization, at least with this test data set. Results are presented in the next subsection.

5.2 Results

We now present the results obtained after performing the steps discussed in the previous subsection. We would like to evaluate the discriminitave/classification power of the BRIEF descriptor for characterizing entire image patches.

5.2.1 Hamming Distance Matrix

A good way of evaluating the utility of BRIEF arrangements as a general patch descriptor is to create a hamming distance matrix among the traffic sign models in fig 5. This way, we can evaluate BRIEF's discriminative power among different models of traffic signs. An example of a distance matrix is given in figure 7 for a BRIEF-512 descriptor with an unordered Gaussian sample for the pair points.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43
1	0	32	50	52	54	51	132	156	161	165	157	241	226	263	315	206	141	303	269	254	252	248	314	269	235	260	277	282	255	287	269	248	184	275	272	243	281	271	301	289	189	192	185
2	32	0	32	42	52	35	112	152	159	155	155	261	220	259	313	204	137	303	277	248	252	252	314	265	235	270	283	284	261	287	283	254	192	269	270	243	285	277	303	285	191	182	181
3	50	32	0	38	68	43	118	156	155	153	159	253	220	247	307	200	141	309	281	238	252	258	314	259	237	270	287	276	257	281	277	256	192	257	264	233	289	285	303	283	195	180	191
4	52	42	38	0	70	25	104	164	159	155	153	263	228	265	309	208	129	311	275	256	258	264	320	267	243	274	283	282	273	297	283	258	188	271	270	253	291	291	299	287	189	176	175
5	54	52	68	70	0	77	148	160	169	167	169	255	230	267	331	190	145	287	265	246	250	250	298	249	229	266	271	270	255	285	277	248	182	261	282	247	287	261	291	263	191	202	197
6	51	35	43	25	77	0	85	165	166	154	148	262	229	268	310	209	130	316	278	257	255	261	327	268	240	269	286	287	270	296	280	259	189	270	261	252	288	288	300	290	180	171	170
7	132	112	118	104	148	85	0	204	211	207	199	291	232	309	289	258	161	303	305	298	278	302	350	283	261	290	315	306	297	327	303	290	128	303	256	263	293	301	291	313	213	128	133
8	156	152	156	164	160	165	204	0	65	139	153	219	234	233	335	212	133	327	217	224	224	240	282	223	217	238	231	236	243	279	251	234	214	251	254	247	245	267	259	273	195	174	193
9	161	159	155	159	169	166	211	65	0	162	178	218	231	224	330	207	144	312	212	239	235	249	285	252	218	233	224	221	256	276	242	217	219	248	249	266	252	264	260	278	206	189	212
10	165	155	153	155	167	154	207	139	162	0	122	212	235	228	320	191	134	352	228	185	199	239	271	198	208	225	232	231	226	258	238	221	237	232	241	226	270	268	276	266	198	115	174
11	157	155	159	153	169	148	199	153	178	122	0	250	229	252	308	205	156	344	264	231	241	267	295	250	252	275	264	285	266	288	276	255	241	256	269	238	312	248	300	248	210	159	104
12	241	261	253	263	255	262	291	219	218	212	250	0	241	240	278	211	248	264	96	97	107	101	169	118	104	79	98	115	98	130	58	87	255	194	183	246	216	210	254	236	188	259	286
13	226	220	220	228	230	229	232	234	231	235	229	241	0	217	245	172	235	245	223	234	232	228	188	237	253	238	211	244	239	215	233	232	230	261	260	245	247	235	261	249	269	236	241
14	263	259	247	265	267	268	309	233	224	228	252	240	217	0	268	167	274	244	242	229	223	207	169	236	238	243	224	239	220	192	232	233	323	224	227	164	202	182	210	206	296	281	304
15	315	313	307	309	331	310	289	335	330	320	308	278	245	268	0	223	320	134	280	285	291	277	229	286	312	275	282	265	276	248	246	287	257	228	197	252	224	250	218	224	286	303	294
16	206	204	200	208	190	209	258	212	207	191	205	211	172	167	223	0	211	199	199	222	222	222	188	227	233	216	203	220	227	219	219	218	212	201	196	197	213	203	203	187	205	266	271
17	141	137	141	129	145	130	161	133	144	134	156	248	235	274	320	211	0	344	242	253	257	297	315	262	258	265	248	271	284	332	286	261	175	290	251	268	256	298	272	304	178	155	172
18	303	303	309	311	287	316	303	327	312	352	344	264	245	244	134	199	344	0	252	271	277	233	191	270	264	245	246	239	252	212	242	251	245	244	225	262	196	214	186	202	290	355	342
19	269	277	281	275	265	278	305	217	212	228	264	96	223	242	280	199	242	252	0	117	125	155	133	132	134	111	34	103	142	150	124	99	265	226	241	256	180	188	220	222	220	281	298
20	254	248	238	256	246	257	298	224	239	185	231	97	234	229	285	222	253	271	117	0	100	114	134	83	87	130	111	132	113	107	119	106	304	201	226	211	203	219	265	211	225	258	285
21	252	252	252	258	250	255	278	224	235	199	241	107	232	223	291	222	257	277	125	100	0	114	128	79	71	108	123	102	107	123	113	114	262	227	214	207	233	179	217	255	233	228	253
22	248	252	258	264	250	261	302	240	249	239	267	101	228	207	277	222	297	233	155	114	114	0	126	151	95	116	141	138	101	85	93	80	292	209	222	209	223	183	245	223	237	288	309
23	314	314	314	320	298	327	350	282	285	271	295	169	188	169	229	188	315	191	133	134	128	126	0	127	145	138	111	138	143	95	151	140	314	241	236	197	181	155	183	187	283	322	331
24	269	265	259	267	249	268	283	223	252	198	250	118	237	236	286	227	262	270	132	83	79	151	127	0	82	115	124	109	114	122	128	127	263	222	217	202	208	196	242	232	210	239	266
25	235	235	237	243	229	240	261	217	218	208	252	104	253	238	312	233	258	264	134	87	71	95	145	82	0	95	128	105	96	120	108	91	257	214	217	210	232	196	248	256	204	241	276
26	260	270	270	274	266	269	290	238	233	225	275	79	238	243	275	216	265	245	111	130	108	116	138	115	95	0	113	104	113	141	85	110	246	215	200	237	193	203	237	251	201	264	299
27	277	283	287	283	271	286	315	231	224	232	264	98	211	224	282	203	248	246	34	111	123	141	111	124	128	113	0	115	138	128	124	95	287	236	239	238	166	184	212	216	238	291	306
28	282	284	276	282	270	287	306	236	221	231	285	115	244	239	265	220	271	239	103	132	102	138	138	109	105	104	115	0	113	129	107	106	248	213	230	231	213	201	203	231	229	262	305
29	255	261	257	273	255	270	297	243	256	226	266	98	239	220	276	227	284	252	142	113	107	101	143	114	96	113	138	113	0	90	86	133	283	206	219	190	214	204	232	228	228	265	292
30	287	287	281	297	285	296	327	279	276	258	288	130	215	192	248	219	332	212	150	107	123	85	95	122	120	141	128	129	90	0	112	115	319	218	227	186	210	184	216	194	284	305	326
31	269	283	277	283	277	280	303	251	242	238	276	58	233	232	246	219	286	242	124	119	113	93	151	128	108	85	124	107	86	112	0	103	269	202	187	226	220	200	246	230	202	271	298
32	248	254	256	258	248	259	290	234	217	221	255	87	232	233	287	218	261	251	99	106	114	80	140	127	91	110	95	106	133	115	103	0	276	225	224	239	221	201	247	225	215	278	293
33	184	192	192	188	182	189	128	214	219	237	241	255	230	323	257	212	175	245	265	304	262	292	314	263	257	246	287	248	283	319	269	276	0	293	234	293	263	293	259	311	193	148	169
34	275	269	257	271	261	270	303	251	248	232	256	194	261	224	228	201	290	244	226	201	227	209	241	222	214	215	236	213	206	218	202	225	293	0	203	232	268	198	260	154	228	281	298
35	272	270	264	270	282	261	256	254	249	241	269	183	260	227	197	196	251	225	241	226	214	222	236	217	217	200	239	230	219	227	187	224	234	203	0	231	205	257	183	257	221	244	273
36	243	243	233	253	247	252	263	247	266	226	238	246	245	164	252	197	268	262	256	211	207	209	197	202	210	237	238	231	190	186	226	239	293	232	231	0	186	188	192	190	278	243	258
37	281	285	289	291	287	288	293	245	252	270	312	216	247	202	224	213	256	196	180	203	233	223	181	208	232	193	166	213	214	210	220	221	263	268	205	186	0	246	172	244	234	285	306
38	271	277	285	291	261	288	301	267	264	268	248	210	235	182	250	203	298	214	188	219	179	183	155	196	196	203	184	201	204	184	200	201	293	198	257	188	246	0	234	156	270	295	280
39	301	303	303	299	291	300	291	259	260	276	300	254	261	210	218	203	272	186	220	265	217	245	183	242	248	237	212	203	232	216	246	247	259	260	183	192	172	234	0	234	298	263	292
40	289	285	283	287	263	290	313	273	278	266	248	236	249	206	224	187	304	202	222	211	255	223	187	232	256	251	216	231	228	194	230	225	311	154	257	190	244	156	234	0	262	321	298
41	189	191	195	189	191	180	213	195	206	198	210	188	269	296	286	205	178	290	220	225	233	237	283	210	204	201	238	229	228	284	202	215	193	228	221	278	234	270	298	262	0	223	228
42	192	182	180	176	202	171	128	174	189	115	159	259	236	281	303	266	155	355	281	258	228	288	322	239	241	264	291	262	265	305	271	278	148	281	244	243	285	295	263	321	223	0	91
43	185	181	191	175	197	170	133	193	212	174	104	286	241	304	294	271	172	342	298	285	253	309	331	266	276	299	306	305	292	326	298	293	169	298	273	258	306	280	292	298	228	91	0

Figure 7: BRIEF descriptor distance matrix for the traffic sign models in fig 5.

From this matrix we can see the BRIEF descriptor has acceptable results when models are not too similar to each other. For example, among the circular speed signs, the discriminative power is not as good as we can see low descriptor hamming distance among models (shown in red). Of course, an optimized classifier would try to balance the threshold as to achieve higher precision among similar models.

5.2.2 BRIEF Descriptors Comparison

With the goal of studying the effect of the descriptor's size and an ordered vs unordered distribution when sampling the pair points, a simulation consisting on 1000 trials was performed and the highest and lowest distances among model descriptor obtained. This distance was later normalized as a percentage of the size of the descriptor, e.g. If the descriptor size is 512, and the hamming distance was 256, we would obtain a value of 50.

	Ordered Random Samling								Unordered Random Sampling							
	Random polar pairs - 128		Random polar pairs - 256		Random polar pairs - 512		Random polar pairs - 1024		i.i.d. Gaussian - 128		i.i.d. Gaussian - 256		i.i.d. Gaussian - 512		i.i.d. Gaussian - 1024	
	MIN	MAX	MIN	MAX	MIN	MAX	MIN	MAX	MIN	MAX	MIN	MAX	MIN	MAX	MIN	MAX
0	6.20	63.92	6.27	62.43	6.29	61.50	6.32	61.01	6.35	65.36	6.49	64.30	6.52	63.61	6.50	63.35
1	5.11	64.46	5.48	62.89	5.72	61.93	5.91	61.47	4.96	65.59	5.40	64.55	5.73	63.89	5.87	63.69
2	5.69	64.86	5.86	63.34	6.03	62.32	6.14	61.88	5.86	65.07	6.22	64.07	6.42	63.35	6.59	63.12
3	4.72	65.46	4.82	63.89	5.00	63.06	5.11	62.65	4.79	65.48	5.04	64.39	5.11	63.74	5.14	63.42
4	10.79	64.48	10.86	63.62	10.90	63.38	11.02	63.32	10.72	67.26	10.93	67.09	10.99	66.94	11.09	66.94
5	4.48	65.69	4.79	64.24	4.97	63.41	5.10	62.96	4.48	65.87	4.81	64.71	4.99	64.04	5.09	63.78
6	16.18	71.84	16.85	70.69	17.24	69.99	17.46	69.86	16.09	67.87	16.23	66.60	16.33	66.12	16.33	66.20
7	16.96	60.59	16.96	59.39	16.99	58.68	17.12	58.39	11.92	65.67	11.85	65.08	11.81	64.61	11.92	64.51
8	17.06	63.82	16.97	63.07	16.99	62.77	17.12	62.59	11.93	65.07	11.85	64.55	11.81	64.45	11.92	64.44
9	20.57	64.60	21.77	63.64	22.25	63.34	22.58	63.37	20.42	70.14	21.46	69.99	21.90	70.04	22.27	70.08
10	19.32	65.44	20.24	64.37	20.59	63.95	20.87	63.75	18.60	67.52	19.05	66.96	19.29	66.99	19.39	67.00
11	9.92	60.62	10.55	59.48	10.88	58.83	11.02	58.61	12.30	58.51	12.84	56.55	13.03	55.49	13.04	54.70
12	24.23	57.80	24.34	55.90	24.34	54.64	24.45	53.93	29.58	55.93	29.87	53.97	30.03	52.70	29.99	52.00
13	22.97	67.61	23.58	66.91	24.02	66.23	24.24	66.15	28.81	63.67	29.85	62.27	30.46	61.66	30.71	61.18
14	29.04	66.49	29.61	65.16	29.54	64.42	29.48	63.95	26.02	69.29	25.94	68.14	25.79	67.47	25.87	67.17
15	22.23	56.91	22.97	56.18	23.53	55.51	23.87	55.52	28.13	55.65	29.13	54.60	29.70	54.14	29.87	53.89
16	21.63	73.02	22.79	72.20	23.61	72.03	24.03	72.05	20.64	71.15	21.78	70.63	22.38	70.43	22.70	70.46
17	22.50	75.55	23.89	75.17	24.57	74.84	24.96	74.83	25.81	73.57	25.93	72.72	25.79	72.18	25.87	71.84
18	9.81	60.59	10.35	59.67	10.52	59.10	10.72	59.10	7.31	59.01	7.41	57.27	7.44	56.21	7.49	55.61
19	11.85	59.89	12.38	58.34	12.78	57.59	13.00	57.27	15.17	59.55	15.91	58.06	16.42	56.97	16.72	56.33
20	10.72	58.82	11.37	57.47	11.90	56.38	12.18	55.93	13.71	57.89	14.34	56.18	14.70	55.31	14.97	54.74
21	12.07	64.46	12.23	63.63	12.31	63.47	12.26	63.58	14.36	60.73	14.83	59.37	15.14	58.58	15.20	58.19
22	15.97	71.60	16.17	70.74	16.27	70.14	16.27	69.96	18.18	67.94	18.38	66.93	18.38	66.39	18.46	66.35
23	11.58	59.25	12.02	57.61	12.24	56.58	12.36	55.93	14.90	58.20	15.35	56.68	15.73	55.71	15.98	55.13
24	13.69	58.38	14.20	56.87	14.49	55.95	14.58	55.49	13.83	58.21	14.57	56.76	14.92	55.95	15.13	55.58
25	12.83	58.90	13.38	57.34	13.64	56.27	13.78	55.62	15.70	58.71	16.75	56.78	17.15	55.75	17.49	55.26
26	10.68	59.74	10.76	58.25	10.61	57.43	10.74	57.07	7.31	59.78	7.41	58.37	7.44	57.73	7.49	57.72
27	13.67	59.42	14.26	57.90	14.55	56.77	14.70	56.21	17.25	59.59	18.04	58.12	18.42	57.51	18.63	57.19
28	15.74	60.93	16.48	59.81	16.83	59.14	17.12	58.90	15.83	58.87	16.59	57.44	16.78	56.71	16.90	56.35
29	11.78	68.58	12.14	67.84	12.29	67.43	12.26	67.51	14.32	64.19	14.77	63.11	15.10	62.43	15.19	62.11
30	11.01	59.51	11.14	57.66	11.06	56.50	11.08	55.80	12.55	59.96	12.89	58.56	13.03	57.88	13.04	57.46
31	13.30	57.82	13.82	56.08	14.18	55.01	14.36	54.44	15.32	58.23	16.46	56.54	17.15	55.60	17.65	54.97
32	22.40	65.53	22.91	64.13	23.23	63.42	23.50	62.88	25.45	65.38	25.91	64.00	26.19	63.50	26.20	63.20
33	28.55	59.48	28.81	58.01	28.70	56.88	28.79	56.32	30.38	61.22	31.08	59.79	31.47	59.10	31.50	58.60
34	28.60	60.04	28.84	58.54	28.70	57.76	28.79	57.50	31.91	60.15	33.04	58.92	33.63	58.17	34.02	57.67
35	23.42	68.55	24.11	67.43	24.58	66.85	24.83	66.57	30.96	60.02	32.47	58.24	33.04	57.67	33.30	57.30
36	29.22	68.34	30.46	67.38	30.99	67.00	31.30	66.90	30.47	63.37	31.81	62.13	32.50	61.56	32.90	61.34
37	29.39	66.84	30.39	65.78	31.02	65.44	31.34	65.25	29.97	63.58	30.89	62.54	31.41	61.94	31.69	61.63
38	23.95	68.67	24.93	67.34	25.32	66.83	25.58	66.66	30.86	62.65	31.86	61.00	32.45	60.23	32.95	59.74
39	23.66	69.17	24.26	68.24	24.68	67.66	24.91	67.37	29.14	66.30	29.87	65.03	30.49	64.44	30.86	63.90
40	28.47	63.46	28.83	62.07	29.16	61.19	29.05	60.77	30.07	62.30	31.41	60.80	31.94	60.02	32.25	59.46
41	15.00	74.93	15.30	74.85	15.37	74.70	15.38	74.80	19.77	71.79	20.42	71.49	20.76	71.54	20.97	71.52
42	15.10	72.17	15.40	71.41	15.39	71.04	15.38	71.19	18.21	68.42	18.73	67.57	19.07	67.57	19.24	67.37

Figure 8: Average normalized maximum and minimum hamming distance between models' BRIEF descriptors.

As we can see from the figure above, increasing the descriptor size does increase the normalized minimum hamming distance for the models. Nonetheless, the increase is not substantial indicating that a small descriptor can perform relatively more efficiently as it requires less memory and computation time. Moreover, the unordered random sampling yielded better results for all sizes tested with a couple of exceptions. For example, models 18 and 26 benefit from the polar pairs, indicating it somehow leverages the symmetrical properties of said models. The models in green benefit from a stronger discriminative power from this BRIEF descriptor, of course this can be a result of the circular speed sign having more similar models in the set.

6 References

[1] Calonder, Michael and Lepetit, Vincent and Strecha, Christoph and Fua, Pascal. (2010). BRIEF: Binary Robust Independent Elementary Features. Eur. Conf. Comput. Vis.. 6314. 778-792. 10.1007/978-3-642-15561-1-56.

[2] Ozuysal, M., Calonder, M., Lepetit, V., Fua, P.: Fast Keypoint Recognition Using Random Ferns. IEEE Transactions on Pattern Analysis and Machine Intelligence 32 (2010) 448–461

[3] Lepetit, V., Fua, P.: Keypoint Recognition Using Randomized Trees. IEEE Trans- actions on Pattern Analysis and Machine Intelligence 28 (2006) 1465–1479

Appendix A BRIEF Arrangements as a Key-point Descriptor Code

Listing 1: BRIEF Arrangements as a Key-point Descriptor Code

```
clear; clc; close all;

S = 32;                     % patch size

Ims = {};            % library for imaes
for i = 1:6
    Im = imread(strcat("wall/img", int2str(i), ".ppm"));
    Ims{i} = rgb2gray(Im);
end

A_mat = {};          % library of homographic matrices
for i = 2:6
    FileName = strcat('wall/H1to', int2str(i), 'p');
    fid = fopen(FileName, 'r');
    C = textscan(fid, '%s');
    C = C{1};
    A = zeros(3,3);
    for j = 1:9
        A(floor((j-1)/3)+1, mod(j-1,3)+1) = str2num(C{j});
    end
    A_mat{i} = A;
end

%% select points

N_points = 1000;            % Number of corners selected

points1 = detectHarrisFeatures(Ims{1});


valid_points1 = and(and(points1.Location(:,1) > S/2+1, points1.Location(:,2) > ...
    S/2+1),  ...
            and(points1.Location(:,1) < size(Ims{1},2)-S/2, points1.Location(:,2) < ...
                size(Ims{1},1)-S/2));
% valid points are those that have some distance to the edge

points1 = points1(valid_points1);               % subset of valid points
points1 = points1.selectStrongest(N_points);    % choose the strongest

points_matrix = zeros(6,2,N_points);            % dimensions are (which_im, (x,y), ...
    Points)
points_matrix(1,:,:) = points1.Location';       % save the ones from the first image
valid_points_mat = zeros(6,N_points);           % valid points for (which_im, ...
    which_point)
valid_points_mat(1,:) = ones(1,N_points);       % we have already made sure that the ...
    first image where valid

p_use = ones(3,N_points);
p_use(1:2,:) = points_matrix(1,:,:);            % p_use are the points (x,y,1)' in a ...
    matrix
for i = 2:6
    p_aux = A_mat{i}*p_use;                     % hom transf for the image i
    points_matrix(i,1,:) = p_aux(1,:)./p_aux(3,:);
    points_matrix(i,2,:) = p_aux(2,:)./p_aux(3,:);
    valid_points_mat(i,:) = and(and(points_matrix(i,1,:) > S/2+1, ...
        points_matrix(i,2,:) > S/2+1),  ...
            and(points_matrix(i,1,:) < size(Ims{i},2)-S/2, points_matrix(i,2,:) < ...
                size(Ims{i},1)-S/2));
    % check which are valid in the image i
end


% r_ind = randi(N_points);
%
figure()
```

```
imshow(Ims{1})
hold on
plot( squeeze(points_matrix(1,1,:)), squeeze(points_matrix(1,2,:)), '+g' )
% plot(points_matrix(1, 1, r_ind), points_matrix(1, 2, r_ind), 'ro')
hold off


figure()
imshow(Ims{3})
hold on
plot( squeeze(points_matrix(3,1,:)), squeeze(points_matrix(3,2,:)), '+g' )
% plot(points_matrix(3, 1, r_ind), points_matrix(3, 2, r_ind), 'ro')
hold off


%% start exercise
n = 128;                          % number of descriptors pairs

X_descriptor_points = randi(S, [2,n])-S/2; % each column [p1(1), p1(2)]'
Y_descriptor_points = randi(S, [2,n])-S/2; % each column [p2(1), p2(2)]'

% from [-S/2, +S/2]
figure; hold on;
for i = 1:n
    plot( [X_descriptor_points(1,i);Y_descriptor_points(1,i)], ...
        [X_descriptor_points(2,i);Y_descriptor_points(2,i)], '-k')
end
%%

% First filter the image
for i = 1:6
    Im = imread(strcat("wall/img", int2str(i), ".ppm"));
    Ims{i} = rgb2gray(Im);
    Ims{i} = imgaussfilt(Ims{i}, 2);
end

% Obtain the desciptors of each valid point in each image
Im1_descriptor_points = zeros(6, N_points, n);    % (Im, point, description)
for i = 1:6
    for j = 1:N_points
        if(valid_points_mat(i,j))
            c_point = round(squeeze(points_matrix(i,:,j)));
            idx1 = sub2ind(size(Ims{i}), c_point(2)+X_descriptor_points(2,:), ...
                c_point(1)+X_descriptor_points(1,:));
            idx2 = sub2ind(size(Ims{i}), c_point(2)+Y_descriptor_points(2,:), ...
                c_point(1)+Y_descriptor_points(1,:));
            Im1_descriptor_points(i,j,:) = Ims{i}(idx1) > Ims{i}(idx2);
        else
            Im1_descriptor_points(i,j,:) = -1;
        end
    end
end
% descriptor comparison
predictions_correct = zeros(6,N_points);            % save in a matrix if it hited ...
    right
predictions_distance = zeros(6,N_points);           % save in a matrix the distances
for j = 1:N_points % for all points in the first image
    xor_dist = sum(xor(Im1_descriptor_points(1,j,:), Im1_descriptor_points( ...
        logical(valid_points_mat(:,j)),:,:)), 3);       % xor=> matrix of size ...
            (valid_points, 1, n) thend add third dimension
    [m,m_ind] = min(xor_dist ,[],2 );                   % find the minimum
    predictions_correct(logical(valid_points_mat(:,j)),j) = (m_ind == j); % did it hit?
    idx1 = sub2ind(size(xor_dist), [1:size(xor_dist, 1)], m_ind');        % indices ...
        of the descriptor
    predictions_distance(logical(valid_points_mat(:,j)),j) = xor_dist(idx1); % ...
        compute distance
end

%% results

which_image = 3;
```

```
True_pos_dist = ...
    predictions_distance(which_image,and(logical(predictions_correct(which_image,:)), ...
    logical(valid_points_mat(which_image,:)))   ) ;
False_pos_dist  = predictions_distance(which_image,and(¬ ...
    logical(predictions_correct(which_image,:)), ...
    logical(valid_points_mat(which_image,:)))   ) ;


figure;
h1 = histogram(True_pos_dist,40,'Normalization','pdf')
hold on
h2 = histogram(False_pos_dist,40,'Normalization','pdf')
legend('True positive', 'False positive')
hold off

correct_hits = sum( and(logical(predictions_correct(which_image,:)), ...
    logical(valid_points_mat(which_image,:)))  )
bad_hits = sum(  and(¬logical(predictions_correct(which_image,:)), ...
    logical(valid_points_mat(which_image,:)))  )

TPR = hist(True_pos_dist , 40, 'Normalization', 'pdf');
FPR = hist(False_pos_dist , 40, 'Normalization', 'pdf');

hit_rate = correct_hits/(correct_hits+bad_hits)

figure()
title('Roc curve')
plot(cumsum(FPR), cumsum(TPR))






%% new descriptor proposed

theta = linspace(0,2*pi, 100);
circle_matrix = zeros(33,33);

r = S/2;
circle_points = [r*cos(theta);r*sin(theta)];            % radius S/2 outer circle
circle_points = unique( round(circle_points)' , 'rows' )';% round and eliminate the ...
    repeated points

circle_matrix( sub2ind([33,33] , circle_points(1,:)+S/2+1, circle_points(2,:)+S/2+1 ...
    ) ) = 1;
n_left = n-round(n/3);
X_descriptor_points_new = circle_points(:,randi(size(circle_points,2),round(n/3),1));
Y_descriptor_points_new = ...
    circle_points(:,randi(size(circle_points,2),round(n/3),1)); % one third to the ...
    outer circle

% midle circle
r = S/3;
circle_points = [r*cos(theta);r*sin(theta)];            % radius S/3 outer circle
circle_points = unique( round(circle_points)' , 'rows' )';% round and eliminate the ...
    repeated points

circle_matrix( sub2ind([33,33] , circle_points(1,:)+S/2+1 , ...
    circle_points(2,:)+S/2+1) ) = 1;
n_left = n_left - round(n/2);
X_descriptor_points_new = [X_descriptor_points_new, ...
    circle_points(:,randi(size(circle_points,2),round(n/2),1))];
Y_descriptor_points_new = [Y_descriptor_points_new, ...
    circle_points(:,randi(size(circle_points,2),round(n/2),1))]; % one half to the ...
    middle circle

% inner circle
r = S/6;
circle_points = [r*cos(theta);r*sin(theta)];            % radius S/3 outer circle
```

```
circle_points = unique( round(circle_points)' , 'rows' )';% round and eliminate the ...
    repeated points

circle_matrix( sub2ind([33,33] , circle_points(1,:)+S/2+1 , circle_points(2,:)+S/2+1 ...
    ) ) = 1;
X_descriptor_points_new = [ X_descriptor_points_new, ...
    circle_points(:,randi(size(circle_points,2),n_left,1) ) ];
Y_descriptor_points_new = [ Y_descriptor_points_new, ...
    circle_points(:,randi(size(circle_points,2),n_left,1) ) ]; % one half to the ...
    middle circle


figure
imshow(circle_matrix)
hold on
for i = 1:n
    plot( [X_descriptor_points_new(1,i);Y_descriptor_points_new(1,i)]+S/2+1, ...
        [X_descriptor_points_new(2,i);Y_descriptor_points_new(2,i)]+S/2+1, '-r')
end
%%

% First filter the image
for i = 1:6
    Im = imread(strcat("wall/img", int2str(i), ".ppm"));
    Ims{i} = rgb2gray(Im);
    Ims{i} = imgaussfilt(Ims{i}, 2);
end

% Obtain the desciptors of each valid point in each image
Im1_descriptor_points = zeros(6, N_points, n);    % (Im, point, description)
for i = 1:6
    for j = 1:N_points
        if(valid_points_mat(i,j))
            c_point = round(squeeze(points_matrix(i,:,j)));
            idx1 = sub2ind(size(Ims{i}), c_point(2)+X_descriptor_points_new(2,:), ...
                c_point(1)+X_descriptor_points_new(1,:));
            idx2 = sub2ind(size(Ims{i}), c_point(2)+Y_descriptor_points_new(2,:), ...
                c_point(1)+Y_descriptor_points_new(1,:));
            Im1_descriptor_points(i,j,:) = Ims{i}(idx1) > Ims{i}(idx2);
        else
            Im1_descriptor_points(i,j,:) = -1;
        end
    end
end
% descriptor comparison
predictions_correct = zeros(6,N_points);            % save in a matrix if it hited ...
    right
predictions_distance = zeros(6,N_points);           % save in a matrix the distances
for j = 1:N_points % for all points in the first image
    xor_dist = sum(xor(Im1_descriptor_points(1,j,:), Im1_descriptor_points( ...
        logical(valid_points_mat(:,j)),:,:)), 3);       % xor=> matrix of size ...
            (valid_points, 1, n) thend add third dimension
    [m,m_ind] = min(xor_dist ,[],2 );                   % find the minimum
    predictions_correct(logical(valid_points_mat(:,j)),j) = (m_ind == j); % did it hit?
    idx1 = sub2ind(size(xor_dist), [1:size(xor_dist, 1)], m_ind');        % indices ...
        of the descriptor
    predictions_distance(logical(valid_points_mat(:,j)),j) = xor_dist(idx1); % ...
        compute distance
end

%% results

which_image = 2;

True_pos_dist = ...
    predictions_distance(which_image,and(logical(predictions_correct(which_image,:)), ...
    logical(valid_points_mat(which_image,:)))   ) ;
False_pos_dist  = predictions_distance(which_image,and(¬ ...
    logical(predictions_correct(which_image,:)), ...
    logical(valid_points_mat(which_image,:)))   ) ;

```

```

figure;
h1 = histogram(True_pos_dist,40,'Normalization','pdf')
hold on
h2 = histogram(False_pos_dist,40,'Normalization','pdf')
legend('True positive', 'False positive')
hold off

correct_hits = sum( and(logical(predictions_correct(which_image,:)), ...
    logical(valid_points_mat(which_image,:)))  )
bad_hits = sum(  and(¬logical(predictions_correct(which_image,:)), ...
    logical(valid_points_mat(which_image,:)))  )

TPR = hist(True_pos_dist , 33, 'Normalization', 'pdf');
FPR = hist(False_pos_dist , 33, 'Normalization', 'pdf');

hit_rate = correct_hits/(correct_hits+bad_hits)

figure()
title('Roc curve')
plot(cumsum(FPR), cumsum(TPR))
```

Appendix B BRIEF Arrangements as a General Patch Descriptor Code

Listing 2: BRIEF Arrangements as a General Patch Descriptor Code

```
%% DOUBTS AND OBSERVATIONSforma

% No key Points detection, using whole patch...
%% SET UP --------------------------------------------------------------------------
clear; clc;
displaying   = 1;
currentPath = pwd;
f           = filesep;
ImagesPath  = strcat(currentPath,f,"traffic models/Meta/",f);

% Load Traffic Images
nImages = 43;
Ims = cell(1,nImages);           % library for images
for i = 1:nImages
    Im = imread(strcat(ImagesPath,int2str(i-1), ".png"));
    Ims{i} = Im;
end

%% RESIZE IMAGES TO 100x100 PIXELS ---------------------------------------------------

for i = 1:nImages
    Ims{i} = imresize(Ims{i},[100 100]);
end

%% SMOOTH - GAUSSIAN FILTER ----------------------------------------------------------
% Parameters
sigma = 2; %(Filter Size is then 9x9)

% Gaussian Filter Images

Ims_G = cell(1,nImages);
for i = 1:nImages
    Ims_G{i} = imgaussfilt(rgb2gray(Ims{i}),sigma);
end
%% HERE FOR TESTING AND AVERAGING ACROSS RANDOM PATTERNS FOR EVALUATION
clc
nTrials = 1000;
displaying = 0;
DISTminmax_norm_mean = zeros(nImages,2);
for i =1:nTrials
%% BRIEF PATTERN SELECTION/CREATION ----------------------------------------------------

% Parameters
nBits = 512; %128,256,512,1024
im_center = 50;
S = 100;        % Entire Image is our patch

%% (A) i.i.d. Gaussian(0, 1/25*S^2) (From paper: sigma = 1/5*S)
POINTS = zeros(nBits,4);
nCoord = 4*nBits;   % For each bite in the descriptor, 2 points of 2 coordinates are ...
    used
count = 1;
pointMin = 1;       % Select points inside image
pointMax = 100;      % Smallest image dimension, could have also resized images
while count≤nCoord
    point = im_center + round(randn(1)*1/5*S);
    if point ≥ pointMin && point ≤ pointMax
        POINTS(count) = point;
        count = count +1;
    end
end
% Change from coordinates (u,v) to linear index. New Point MTX is then nBits by 2
POINTS_lin = zeros(nBits,2);
```

```
for pair = 1:nBits
    POINTS_lin(pair,1) = sub2ind([S S],POINTS(pair,1),POINTS(pair,2));
    POINTS_lin(pair,2) = sub2ind([S S],POINTS(pair,3),POINTS(pair,4));
end



%% (B) Random polar pairs
nBits = 256; %128,256,512,1024
Psize = 100;
POINTS = zeros(nBits,4);
for i=1:nBits
    theta       = rand()*360;
    r1          = rand()*100-50;
    r2          = rand()*100-50;
    POINTS(i,1) = ceil(50+cosd(theta)*r1); % x1
    POINTS(i,2) = ceil(50+sind(theta)*r1); % y1
    POINTS(i,3) = ceil(50+cosd(theta)*r2); % x2
    POINTS(i,4) = ceil(50+sind(theta)*r2); % y2
end

% Change from coordinates (u,v) to linear index. New Point MTX is then nBits by 2
POINTS_lin = zeros(nBits,2);
for pair = 1:nBits
    POINTS_lin(pair,1) = sub2ind([Psize Psize],POINTS(pair,1),POINTS(pair,2));
    POINTS_lin(pair,2) = sub2ind([Psize Psize],POINTS(pair,3),POINTS(pair,4));
end

%
close all;
if displaying
    figure; hold on;
    for i = 1:nBits
        plot( [POINTS(i,1);POINTS(i,2)], [POINTS(i,3);POINTS(i,4)], '-k')
    end
    title('Unorderd Random Sampling')
   % title('Orderd Random Sampling')
    xlabel('Pixel coordinate v')
    ylabel('Pixel coordinate u')
end


%% IMAGE DESCRIPTORS ------------------------------------------------------------

Ims_D = zeros(nImages,nBits); % Descriptor Array: Row for Image, Columns for bits
for i = 1:nImages
    Ims_D(i,:) = Ims_G{i}(POINTS_lin(:,1)) > Ims_G{i}(POINTS_lin(:,2));
end


%% EVALUATION ------------------------------------------------------------------

DistanceMTX = zeros(nImages,nImages);

for i= 1:nImages
    % Image Descriptor = Ims_D(i,:)
    DescriptorMTX    = repmat(Ims_D(i,:),nImages,1);
    ComparisonMTX    = xor(Ims_D,DescriptorMTX);
    DistanceVec      = sum(ComparisonMTX,2)';
    DistanceMTX(i,:) = DistanceVec;
end
DistanceMTX_norm     = round(DistanceMTX./nBits*100);
if displaying
    % Display compactly (Taken from google)
    fprintf([repmat(sprintf('%% %dd', max(floor(log10(abs(DistanceMTX(:))))) ...
    +2+any(DistanceMTX(:)<0)), 1,size(DistanceMTX,2)) '\n'],DistanceMTX');
    fprintf([repmat(sprintf('%% %dd', max(floor(log10(abs(DistanceMTX_norm(:))))) ...
        ... +2+any(DistanceMTX_norm(:)<0)),1,size(DistanceMTX_norm,2)) '\n'], ...
    DistanceMTX_norm');
end
%% RESULTS ------------------------------------------------------------
```

```

DISTminmax = zeros(nImages,2); % Rows: model, Columns: Min Dist, Max Dist

for i = 1:nImages
    tempResult    = DistanceMTX(i,:);
    % Convert 0 into NaN, hard fix to avoid selecting same image
    tempResult(tempResult==0)=NaN;
    [closesDist, closestIm]    = min(tempResult);
    [furthestDist, furthestIm] = max(tempResult);
    DISTminmax(i,:) = [closesDist, furthestDist];
    if displaying
        disp(['Image ', num2str(i-1), ':'])

        disp(['Closest classification is image ', num2str(closestIm-1), ...
            ', with a hamming distance of ', num2str(closesDist)])
        disp(['Furthest classification is image ', num2str(furthestIm-1), ...
            ', with a hamming distance of ', num2str(furthestDist)])
        disp(' ')
    end
end
DISTminmax_norm = round(DISTminmax./ nBits*100,2);


DISTminmax_norm_mean = DISTminmax_norm_mean + DISTminmax_norm;

%% END of TESTING
end
DISTminmax_norm_mean = DISTminmax_norm_mean./nTrials;
nBits
DISTminmax_norm_mean
```