TREAT: Terse Rapid Edge-Anchored Tracklets

Remi Trichet
remi.trichet@gmail.com

Noel E. O'Connor
noel.oconnor@dcu.ie

Insight centre for data analytics, Dublin City University, Glasnevin, Ireland

Abstract

Fast computation, efficient memory storage, and performance on par with standard state-of-the-art descriptors make binary descriptors a convenient tool for many computer vision applications. However their development is mostly tailored for static images. To respond to this limitation, we introduce TREAT (Terse Rapid Edge-Anchored Tracklets), a new binary detector and descriptor, based on tracklets. It harnesses moving edge maps to perform efficient feature detection, tracking, and description at low computational cost. Experimental results on 3 different public datasets demonstrate improved performance over other popular binary features. These experiments also provide a basis for benchmarking the performance of binary descriptors in video-based applications.

1. Introduction

Binary descriptors have demonstrated comparable performance to some of their floating-point counterparts **[17]** while significantly boosting application speed. This has led to a flourishing literature on the topic **[6][7][8][9]**. More recently, some effort has been made to extend these type of descriptors to the temporal dimension **[1]** **[2]**, and therefore to robotic and event recognition applications. This is a challenging task as mining the temporal dimension requires significant extra computation. As of today, the state of the art has demonstrated the effectiveness of some general characteristics in the design of motion descriptors. However, current binary motion features **[1]** **[2]** still lack some of those. Namely:

1. Long term motion patterns. As the work reported in **[11][12][13][14]** has shown, fine representation of motion patterns is paramount. These descriptors, typically based on tracks or tracklets, need to be robust to camera motion, noise and data variability.

2. Dense point extraction. Static feature **[10]** and trajectories **[11]** dense sampling have been shown to outperform sparse sampling for video activity recognition. However, this strategy implies extensive computation, which may not always be practically feasible for large datasets.

3. Separation or compensation of background features. Being able to differentiate background features from object-of-interest features **[15][16]** has always been a major problem in video based applications. Background features should be isolated, or pruned, depending on whether the background information is treated as complementary information or outliers. Camera motion compensation **[11]** was recently shown to be the best solution to this issue. However, reliable camera motion modelling is a time consuming operation that cannot be performed in the context of real-time applications.

To address these shortcomings, we developed a novel descriptor that incorporates these three characteristics. TREAT (Terse Rapid Edge-Anchored Tracklets) features harness moving edges to extract, track and represent features.

Our contribution is twofold. First, we provide a new tracklet-based binary feature suited for real-time applications. Its extraction and description runs in real-time and results obtained on an event recognition task outperform other concurrent descriptors. The key idea to perform extraction and description in real-time is to resort to the same cue in each case, namely moving edges. Second, results are provided on 3 public datasets of increasing difficulty: UCFsports **[27]**, Hollywood **[26]**, and Hollywood2 **[16]**. To the best of our knowledge, this is the first time binary features are evaluated on such challenging datasets. TREAT code is publicly available **[30]**.

The rest of this paper is organized as follows. Section 2 covers the related work. The third section presents the TREAT descriptor extraction pipeline. Sections 4 to 6 detail each part of the process, namely the video processing, the tracklets extraction and description respectively. Section 7 is dedicated to experimental results and parameter discussion. The final section.

2. Related work

Histogram-of-Gradient (HoG) based methods are the counterparts of binary descriptors. SIFT **[4]** features detect maxima and minima of the result of difference of Gaussians and represents with a gradient orientation histogram. PHOW **[19]** extend them by extracting them densely at different scales. Efficient computation of similar features is proposed by SURF **[5]**. Its 128 bit version, e-SURF **[29],** extends it to the temporal dimension. The possibility to compress HoG into a binary representation has been explored with convolution chains **[22]** or product quantization **[25]**.

978-1-5090-3811-4/16/$31.00 ©2016 European Union IEEE AVSS 2016, August 2016, Colorado Springs, CO, USA

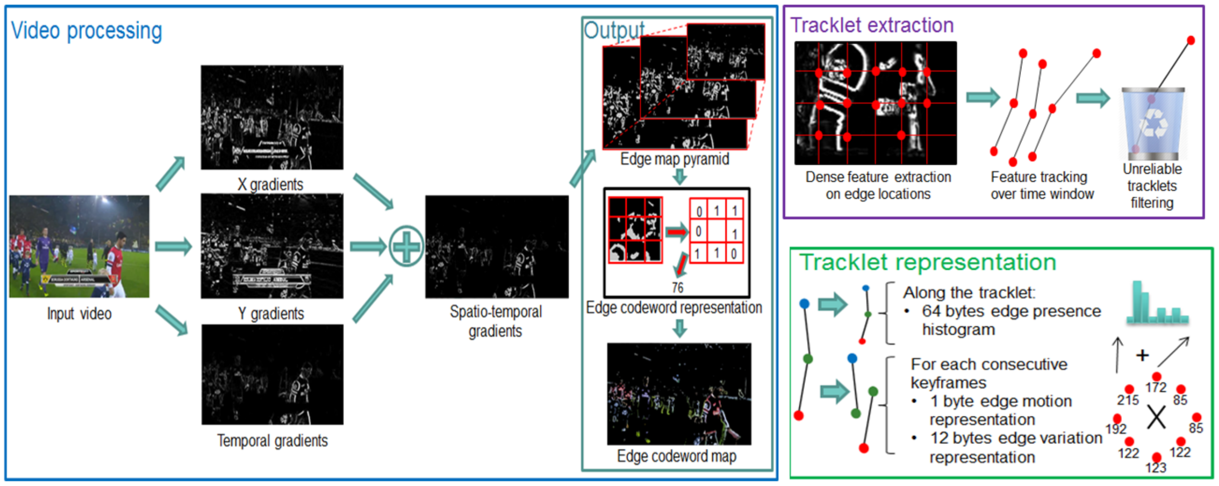

Video processing
Input video
X gradients
Y gradients
Temporal gradients
Spatio-temporal gradients
Output
Edge map pyramid
Edge codeword representation
76
Edge codeword map
Tracklet extraction
Dense feature extraction on edge locations
Feature tracking over time window
Unreliable tracklets filtering
Tracklet representation
Along the tracklet:
• 64 bytes edge presence histogram
For each consecutive keyframes
• 1 byte edge motion representation
• 12 bytes edge variation representation
172
215
85
192
85
122
122
123

Figure 1: **TREAT descriptor extraction pipeline.**

Binary descriptors are an alternative to HoG-based features. They rely on n simple binary intensity tests to produce a corresponding n-bits description. The precursor was the BRIEF descriptor **[6]**, featuring randomly selected pairs without orientation compensation. Its successors all improved this approach in some way. D-BRIEF **[23]** extended it by using box filtering. ORB **[7]** provided orientation compensation based on the intensity centroid moment and learned uncorrelated sampling pairs. FAST **[20]** detector compared pixels on a ring centred at a feature point. BRISK **[8]** further extended FAST by searching for maxima in a 3D scalespace. Rotation invariance was also investigated in MRRID and MROGH descriptors **[20]**. Finally, the main contribution of FREAK **[9]** is the use of a circular pattern. Points are equally spaced on circles concentric, similarly to DAISY **[21]**. Recently, **[31]** employed deep learning. An evaluation of the performance of static binary detectors and descriptors in the context of image recognition was provided in **[17][32].**

Binary motion features extend typical binary features to the temporal dimension. **[3]** developed a descriptor fusing intensity comparison over successive frames of a spatio-temporal volume. However, the method is not robust to camera motion. **[1]** proposed MoFREAK features, extending FREAK **[9]**. It concatenates 8 core bits from the FREAK descriptor with 8 bits of interchange patterns **[3]** computed over the temporal dimension.

3. TREAT Extraction and Description Overview

TREAT feature extraction and description, illustrated in figure 1, proceeds as follows. For each frame, gradients are extracted along the X, Y and temporal axes, and combined to form a spatio-temporal map of the image. This map represents moving edges. We then build up on it a three-level moving edge pyramid. An edge codeword is then associated with each pixel calculated based on 8 moving edges values coarsely depicting its 12×12 neighbourhood. Edge and codeword maps are stored for further use. We densely extract keypoints over moving edge locations and track them according to their neighbourhood edge patterns. Finally, each tracklet is described according to a 64-byte edge presence histogram along the tracklet, 12 bytes for edge variations over successive keyframes, and 1 byte for its motion over each successive keyframe. Features are extracted at several scales determined by a resolution pyramid.

4. Video processing

Most of our tracklet-based video representation is based on a video pre-processing step. First of all, incoming frames are reduced to 640 pixels in width if necessary. Smaller resolutions are possible for optimal speed but we found this size to give reasonable computation time without adversely affecting the performance.

The core cues used by our descriptor are the moving edge locations. They provide a natural way to distinguish foreground objects from background and have shown reliable performance in the past **[11]**. A key idea of this approach is to extract, track and describe features based on this same cue in order to limit computation.

To produce them, we extract X, Y and temporal gradients by applying a 3-pixel wide Sobel filter and subtracting subsequent frames. A threshold over a weighted combination of these three gradient inputs is applied to obtain the final spatio-temporal gradients. The threshold controls the desired magnitude of the edge shift. In our experiments, we assume no a priori knowledge and set it low to extract moving edges of any magnitude. Note that gradient magnitudes, typically used in gradient based descriptors, are lost in the process.

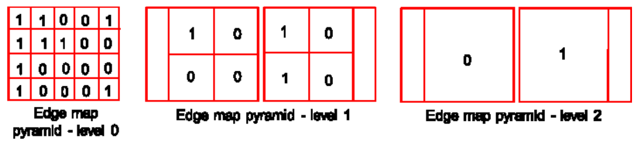

Edge map pyramid - level 0
Edge map pyramid - level 1
Edge map pyramid - level 2

Figure 2: **Toy example of a 3-level edge map representation build over an initial edge map (i.e. the level 0) of 5×4 pixels.**

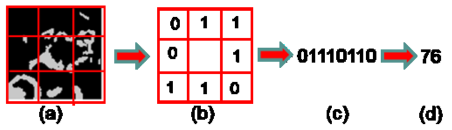


Figure 3: **Example of edge codeword determined over a 12×12 pixel patch. a) Image patch. b) Corresponding last level of the edge map representation c) Resulting binary code d) Edge codeword.**

We then build a three-level moving edge map representation on this initial map, by summing and thresholding to the average each 2×2 patch of the previous level. Further operations only use the last level of the edge map. A toy example is provided in figure 2.

Lastly, we represent each lpixel *(x,y)* 12×12 vicinity at frame *f* with an edge codeword $C_f(x,y)$. This patch is first divided in 9 4×4 areas. The bits corresponding to the 8 outer parts on the last level of the edge map representation are stacked to form a 1-byte encoding. The converted integer value, ranging from 0 to 255, is the final edge codeword. This codeword value is assigned to the patch's central pixel. Figure 3's example illustrates the process.

The edge codeword map is the core component of TREAT extraction. Indeed, this feature has the advantage of being discriminative and widely present across the video. Further feature detection, tracking, and description will mostly be based on this cue, allowing efficient video characterization while keeping the computation load low.

The multi-level edge map representation and edge codeword map are stored for further use.

5. Tracklet extraction

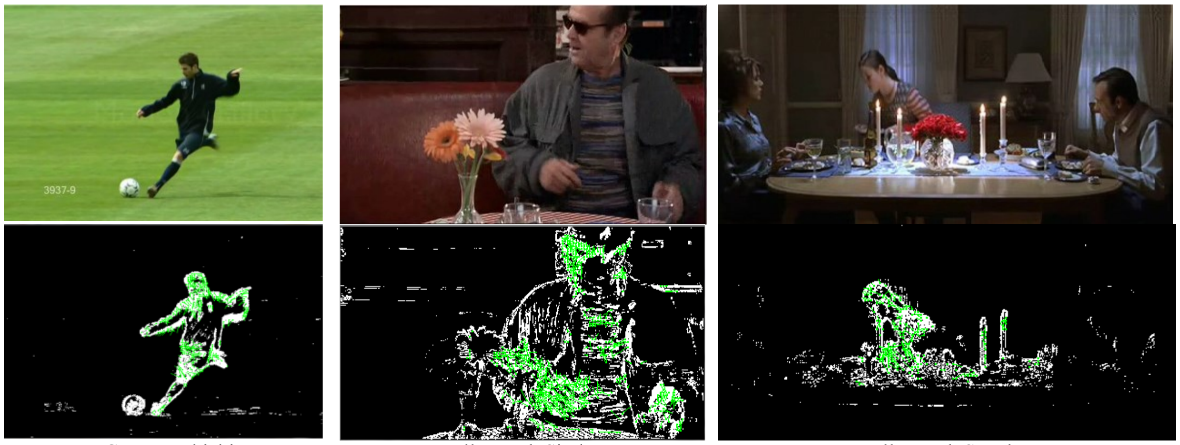
TREATs are densely computed on a regular grid. Computation is restricted to edge locations as the descriptor is based on edge presence and motion. This is a crucial constraint as it allows us to focus the feature extraction process on a limited set of distinctive cues, therefore limiting the computation while yielding robust features. We used a 3-pixel stride for all our experiments.

Each selected edge location is further tracked over a time window of temporal size *L* according to a simple Kalman filter. This step allows us to exploit the longer-term motion information that is lacking in existing motion binary features **[1][3]**.

Matching compares moving edges in an 18×18 neighbourhood around the point to track, giving more weight to the central area. This is achieved by comparing the 4 edge codewords located at the corners of the 6×6 pixel patch centred on the point to track, utilizing the Hamming distance. This representation allows efficient matching based on only 4 integer values. Substantial feature filtering, based on edge presence, is performed to offer a discriminative set of tracks computed in real-time. Practically, we:

- Eliminate tracks of "low edgeness". Tracks centred on patches with few moving edges are considered neither sufficiently discriminative nor robust. These noisy artefacts are discarded.
- Prune out tracks for which the surrounding edges do not vary (i.e. *H(.)* too low). Such tracks are assimilated to object textured parts or the background.
- Similarly, tracks with very high variation (i.e. *H(.)* too high) are considered lost, and discarded.

To avoid drifting, the last two tests are performed between consecutive frames as well as consecutive keyframes. Keyframes are evenly spaced along the track.



UCFsports, kicking Hollywood, Sit down Hollywood, Stand up

Figure 4: **Example of final extracted feature locations as well as their detected TREATs. First row: Original image. Second row: detected moving edges in white and TREAT features in green (the stroke representing the flow direction). Corresponding database and event information are below the images. Best viewed in colour.**

Their number *K* *(2 ≤ K ≤ L)* is a predefined parameter. The influence of *K* and *L* on the feature extraction process is further discussed in section 7.3. Loose thresholding is used to avoid overfitting and guarantee a dense set of tracks (see implementation for threshold values). Figure 4 shows examples of final extracted feature locations as well as their detected TREATs.

To yield a multiscale feature, we extract TREAT at various scales. The frame doesn't undergo the typical blurring performed while reducing its size, as it impairs edge detection. We use a 3 scale pyramid with a $\sqrt{2}$ shrinking factor for all our experiments. In practice, this last step leads to a 2% to 6% performance boost while doubling the computation time.

6. Tracklet description

Our descriptor representation is based on 3 complementary cues: edge variation, edge motion, and edge presence. Edge presence and variation are extracted along the tracklet to compensate for its displacement. Motion is represented separately. In order to avoid redundancy and limit computation, a set of key frames are selected at regular intervals along the tracklet for descriptor calculation.

The edge variation broadly describes the moving edge variation around the tracklet. Learning from the experiences of previous work **[6][7][8][9]**, bitwise differences between the edge codewords of consecutive keyframes are utilized for this purpose.

$$d(C(i,j,t_1), C(i,j,t_2)) = C(i,j,t_1)\ XOR\ C(i,j,t_2) \qquad (1)$$

with *C(i,j,t)* edge codewords at spatial locations *(i, j)* and keyframe *t*. These comparisons are performed at 12 spatial locations *(i+x, j+y)* with $i, j \in \{-6, -3, 0, 3, 6\}$. Locations are chosen to avoid overlap of the corresponding edge codeword patches. It yields a 12-byte descriptor per consecutive keyframes.

The edge motion represents the tracklet optical flow between the selected keyframes. The intensity of the displacement in each of the four possible directions (up, down, left, right) is downsampled to 2 bits. The four

direction representations are further concatenated to form a 1-byte motion description per consecutive keyframes.

Histograms have demonstrated robustness and summarization capabilities (**[11][19]**…). We extend here the principle to binary descriptors. The edge presence histogram describes the edge presence along the tracklet according to the calculated edge codewords. We utilize a 256 bin histogram for this purpose, each bin representing one of the possible edge codeword encoding values. Each edge codeword in a 12×12 neighbourhood centred on the tracklet at each frame increments its corresponding bin. Finally, the histogram is binarized. Each bin is encoded with 2 bits. The first one is set to 1 if the bin value is positive. Then, as peaks are an important component of histograms, the second one is thresholded by the average bin value. More formally, the binarization of the bin b_i in $b_i^j, j = \{0,1\}$ translates as:

$$b_i^0 = \begin{cases} 1 & if\ b_i > 0 \\ 0 & else \end{cases} \quad b_i^1 = \begin{cases} 1 & if\ b_i > 144 * L/256 \\ 0 & else \end{cases} \qquad (2)$$

with *L* the tracklet length. The total histogram size is 64 bytes. With *K* the number of keyframes, the total descriptor size is *64+13(K-1)*.

This representation statically and dynamically describes every single tracklet according to the moving edges patterns in its vicinity. It differs, in spirit, from existing binary descriptor techniques that use binary intensity comparisons.

7. Experiments

In this section, we compare our binary features to state-of-the-art competitors. Also, by evaluating them on datasets of various difficulties, we aim to assess the performance of current binary descriptor and thereby provide a benchmark for event recognition applications. Parameter influence for our approach is also discussed.

7.1. Experimental setup

We performed tests on the Hollywood **[26]**, hollywood2 **[16]**, and UCFsports **[27]** datasets. Performance on the latter is evaluated according to a 5-fold cross validation scheme. We restricted our comparisons to real-time or near-real time features. However, we also added one commonly used and well performing feature for each dataset, as reference. As the purpose of this experiment is to evaluate and compare the descriptor raw potential in the context of event recognition, we utilized the most common encoding and normalization techniques, that is, *k*-means clustering, hard-assignment, and *L2*-histogram normalization. The codebook size is 1000. A linear SVM is employed for all runs. Hamming distance is utilized for comparing binary descriptors, χ^2 for their floating-value counterparts. No PCA, spatial or temporal pooling was performed. SIFT and PHOW features were obtained using the VlFeat toolbox **[28]**. We used the author's implementations of MoFREAK **[1]** and MBP **[3]**. Other features were based on OpenCV code. When the descriptor is available along with its detector, we used it. We employed the grid FAST detector **[20]** for others, which has shown good and fast performance **[17]**. As MBP **[3]** directly produces histograms for each video clip, we utilized a histogram size of 1024 for fair comparison with the codebook size of other methods. TREAT performance is provided for various settings of the two main parameters, the tracklet length *L* and the number of

	STATIC FEATURES								MOTION FEATURES					
Detector	Dense	SIFT	SURF	grid FAST	grid FAST	ORB	BRISK	grid FAST	BRISK	MBP	TREAT	TREAT	TREAT	TREAT
descriptor	PHOW	SIFT	SURF	BRIEF32	BRIEF64	ORB	BRISK	FREAK	Mo-FREAK (L=2)	MBP (L=3)	TREAT (L=7 K=4)	TREAT (L=9 K=3)	TREAT (L=5 K=2)	TREAT (L=5 K=3)
descriptor size (byte)	128	128	64	32	64	32	64	64	16	n.a.	103	90	77	90
Diving	**100%**	**100%**	93.56%	98.29%	**100%**	**100%**	97.32%	**100%**	80.78%	90.01%	**100%**	**100%**	**100%**	**100%**
Golfing	78.00%	63.12%	65.31%	67.19%	70.93%	68.05%	74.29%	69.59%	69.14%	45.49%	**81.24%**	74.63%	79.06%	78.98%
Kicking	**65.43%**	49.87%	58.35%	57.83%	58.79%	51.02%	45.96%	49.87%	58.72%	49.87%	51.02%	53.11%	52.84%	51.71%
Lifting	**100%**	**100%**	**100%**	**100%**	**100%**	**100%**	**100%**	**100%**	96.66%	96.66%	**100%**	**100%**	**100%**	**100%**
Horse Riding	69.91%	**83.21%**	68.77%	71.50%	77.24%	52.44%	73.74%	73.16%	54.44%	52.68%	71.18%	73.18%	77.20%	71.66%
Running	68.48%	62.31%	65.08%	70.73%	69.02%	70.84%	72.29%	73.84%	47.94%	61.39%	**75.39%**	72.61%	74.01%	75.75%
Skateboarding	83.25%	**86.16%**	84.31%	83.25%	83.25%	83.25%	83.01%	83.25%	83.25%	83.25%	83.25%	83.25%	83.25%	83.55%
Swinging	96.17%	89.78%	**97.26%**	95.52%	97.04%	91.56%	95.86%	90.10%	55.71%	63.96%	88.58%	87.95%	87.36%	88.23%
Walking	59.44%	41.42%	74.26%	46.51%	47.00%	62.50%	65.06%	66.39%	66.01%	51.62%	70.21%	74.61%	**77.66%**	77.73%
mAP	80.08%	75.10%	78.54%	76.76%	78.14%	75.52%	78.61%	78.47%	68.07%	66.10%	80.10%	79.93%	**81.26%**	80.85%
time (ms/fr)	***2078.76***	***178.52***	***40.91***	***4.43***	***5.31***	***8.06***	***393.32***	***47.28***	***67.73***	***6.72***	***31.57***	***34.27***	***27.56***	***27.48***

Table 1: AP performance comparison of various descriptors on the UCFsports dataset. Best scoring ones are in bold; mAP over the 9 classes is displayed in red. Computational time is provided in blue.

	STATIC FEATURES								MOTION FEATURES					
Detector	HoG **[26]**	SIFT	SURF	grid FAST	grid FAST	ORB	BRISK	grid FAST	BRISK	grid FAST	TREAT	TREAT	TREAT	TREAT
descriptor	HoG **[26]**	SIFT	SURF	BRIEF32	BRIEF64	ORB	BRISK	FREAK	Mo-FREAK (L=2)	MBP (L=3)	TREAT (L=7 K=4)	TREAT (L=9 K=3)	TREAT (L=5 K=2)	TREAT (L=5 K=3)
descriptor size (byte)	96	128	64	32	64	32	64	64	16	n.a.	103	90	77	90
AnswerPhone	13.40%	16.95%	15.76%	22.51%	19.46%	15.74%	17.20%	16.92%	18.08%	23.57%	20.11%	**24.67%**	19.37%	18.24%
GetOutCar	21.90%	22.03%	22.10%	31.12%	**34.41%**	29.74%	28.83%	27.28%	49.50%	19.46%	31.87%	26.71%	20.20%	26.23%
HandShake	18.60%	15.43%	16.60%	13.91%	15.45%	17.35%	17.16%	12.29%	11.31%	**29.36%**	27.84%	27.47%	25.15%	27.02%
HugPerson	29.10%	22.81%	25.97%	16.22%	18.21%	17.34%	20.26%	27.38%	12.31%	20.32%	36.18%	**36.95%**	28.26%	33.24%
Kiss	52.00%	36.41%	39.09%	33.06%	28.93%	31.75%	35.44%	34.34%	43.45%	35.43%	**55.88%**	55.12%	53.71%	54.06%
SitDown	**29.10%**	21.48%	21.48%	21.48%	21.48%	21.48%	23.08%	21.87%	23.63%	24.65%	22.57%	22.57%	22.57%	22.57%
SitUp	6.50%	8.36%	13.70%	6.27%	6.90%	5.88%	**23.63%**	8.14%	22.68%	9.43%	9.08%	9.16%	9.12%	9.21%
StandUp	**45.40%**	36.99%	34.20%	34.71%	35.64%	35.26%	29.76%	29.44%	27.21%	28.66%	37.72%	38.31%	39.13%	40.56%
mAP	27.00%	22.56%	23.61%	22.41%	22.56%	21.82%	24.42%	22.21%	26.02%	23.86%	**30.16%**	30.12%	27.19%	28.89%
time (ms/fr)	***?***	***84.24***	***14.63***	***2.01***	***3.73***	***3.56***	***385.63***	***32.74***	***39.18***	***2.21***	***22.13***	***24.83***	***19.25***	***19.45***

Table 2: AP performance comparison of various descriptors on the Hollywood dataset. Best scoring ones are in bold; mAP over the 8 classes is displayed in red. Computational time is provided in blue.

selected keyframes K. The performance is measured by mean average precision (mAP) over all classes. Computation time is is performed on frames of 640 pixels in width, and measured in milliseconds per frame, using only one core of an Intel i7-4770 3.4GHz.

7.2. Comparison with state-of-the-art features

Detailed results are provided in tables 1 to 3. The best real-time static descriptor for event recognition is SURF. Surprisingly, the best binary static descriptor for event recognition is, despite its simplicity, BRIEF. The overall best performing real-time descriptor for event recognition is TREAT. Its computation load is a factor of the scene complexity and the video resolution. Tracking features across time takes the bulk of the computation. Hence, the tracklet length L has a significant impact on the computation time. Other motion-based binary descriptors score lower than their static counterparts on UCFsports, probably due to limited camera motion. However, they achieve on par or better mAP on the more complex Hollywood and Hollywood2 datasets.

However, although they are faster, binary features still achieve significantly lower results than non-binary ones on event recognition tasks. As reference, dense trajectories **[11]** report (with a 4000 codeword dictionary) 58.3% mAP on Hollywood2, 88.2% on UCFsports, and run at 1081 ms/frame with our setting. Hence, when faced with challenging datasets, performance drop remains a challenge for binary descriptors.

7.3. Parameter influence and discussion

As shown on tables 1 to 3, TREAT descriptors yield good results across a wide range of parameters. The best tracklet length L ranges between 3 and 11 frames, depending on motion pattern velocity and scene complexity. Shorter tracks lack information.

	STATIC FEATURES								MOTION FEATURES					
Detector	HoG/HoF **[16]**	SIFT	SURF	grid FAST	grid FAST	ORB	BRISK	grid FAST	BRISK	grid FAST	TREAT	TREAT	TREAT	TREAT
descriptor	HoG/HoF **[16]**	SIFT	SURF	BRIEF32	BRIEF64	ORB	BRISK	FREAK	MoFREAK (L=2)	MBP (L=3)	TREAT(L=7 K=4)	TREAT(L=9 K=3)	TREAT(L=5 K=2)	TREAT(L=5 K=3)
descriptor size (byte)	204	128	64	32	64	32	64	64	16	n.a.	103	90	77	90
AnswerPhone	8.80%	10.50%	17.11%	13.21%	13.21%	11.55%	9.97%	12.28%	10.40%	11.38%	15.69%	14.57%	**16.65%**	15.31%
DriveCar	**74.90%**	31.30%	51.96%	45.05%	45.41%	47.02%	44.11%	21.91%	39.93%	25.54%	43.11%	42.32%	44.79%	43.11%
Eat	26.30%	8.20%	12.37%	14.58%	12.23%	20.22%	19.42%	19.26%	**37.91%**	10.97%	27.79%	19.32%	33.52%	34.76%
FightPerson	**67.50%**	8.10%	41.85%	57.78%	57.22%	42.54%	42.84%	43.89%	27.03%	11.67%	39.74%	39.76%	41.55%	37.25%
GetOutCar	9.00%	19.10%	22.62%	**26.59%**	27.87%	18.80%	14.15%	22.05%	13.64%	7.80%	24.85%	24.16%	23.61%	24.01%
HandShake	11.60%	12.30%	11.06%	17.32%	16.27%	9.33%	6.36%	16.61%	11.79%	7.50%	21.49%	17.85%	**26.31%**	23.66%
HugPerson	13.50%	12.90%	15.77%	19.65%	20.25%	9.88%	13.31%	20.99%	10.88%	9.90%	25.82%	23.05%	21.81%	**24.64%**
Kiss	**49.60%**	34.80%	37.02%	35.87%	36.20%	33.07%	32.85%	31.92%	19.04%	14.13%	36.15%	38.57%	36.70%	35.94%
Run	**53.70%**	45.80%	52.99%	50.96%	50.92%	41.90%	40.95%	53.19%	38.51%	37.57%	50.81%	48.62%	53.45%	50.05%
SitDown	**31.60%**	16.10%	19.35%	22.02%	18.40%	19.33%	17.25%	17.18%	19.73%	13.73%	31.26%	25.55%	24.89%	25.70%
SitUp	7.20%	**14.20%**	6.05%	6.29%	6.71%	6.15%	8.61%	6.05%	6.06%	9.23%	8.53%	7.83%	8.24%	7.86%
StandUp	**35.00%**	26.20%	24.11%	24.08%	23.86%	19.41%	20.22%	24.97%	29.56%	19.21%	28.69%	29.36%	29.99%	30.12%
mAP	**32.39%**	19.96%	26.02%	27.78%	27.38%	23.27%	22.50%	24.19%	22.04%	14.89%	29.49%	27.58%	30.13%	29.37%
time (ms/fr)	***?***	***98.51***	***15.21***	***2.48***	***3.71***	***5.47***	***382.23***	***41.63***	***43.57***	***5.13***	***31.87***	***33.28***	***28.54***	***27.11***

Table 3: AP performance comparison of various descriptors on the Hollywood2 dataset. Best scoring ones are in bold; mAP over the 12 classes is displayed in red. Computational time is provided in blue.

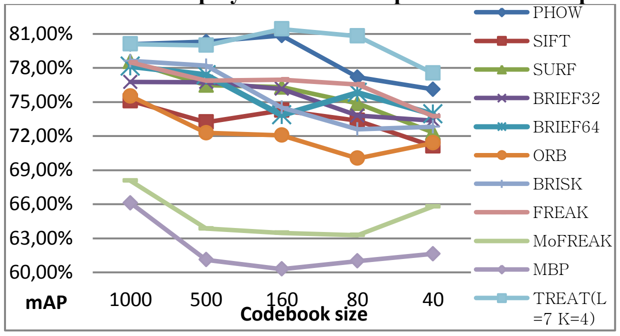


Figure 5: Codebook size impact on the UCF sports dataset classification results. Best viewed in colour.

Longer ones may summarise too much information. Setting up the number of keyframes *K* is straightforward: the more, the better. Nevertheless, a minimum temporal gap of 2 frames is often needed to allow for significant motion pattern changes between 2 consecutive keyframes. Increasing this parameter also increases the descriptor size. We also state that changing the tracklet length and the number of keyframe parameters leads to different class performance. Therefore, stacking TREATs of various lengths and keyframe densities might lead to better results. Nevertheless, this modification will be at the cost of the real-time performance. More sophisticated keyframe selection might be another direction to investigate. However, considering the relatively small length of tracklets, and therefore the small changes it may induce, we didn't prioritize it.

As most binary descriptors are utilized in the context of real-time applications, one might be interested in reducing their associated codebook to speed-up the application. Therefore, we have also evaluated descriptors according to their performance loss as the codebook size dwindles. Results on the UCF sports dataset are displayed in figure 5 (note: quantization experiments are assumed consistent across datasets).

The ability of a descriptor to maintain performance despite a reduction in codebook size is related to its global performance. In other words, better descriptors better resist to a drop in performance. More importantly, binary and floating-point descriptors behave the same way. The average performance drop, compared to the original one and calculated over the 9 best features is 1.4% for 500 codewords, 5.66% for 40 codewords. TREATs show the best resilience to a reduction in codebook size, maintaining performance for a codebook size as low as 80 codewords.

8. Conclusion

In this paper, we have presented TREAT, a new motion binary descriptor that successfully harnesses moving edges to detect, track, and describe motion patterns in real-time. Results on three public datasets significantly outperform existing binary descriptors and even show comparable performance to some state-of-the-art floating value references. These experiments also establish a baseline for the current capacities of binary descriptors on event recognition tasks.

Stacked TREATs using various set of tracklet parameters as well as fusion with existing static descriptors will be investigated in future work.

9. Acknowledgment

This publication has emanated from research conducted with the financial support of Science Foundation Ireland (SFI) under grant number SFI/12/RC/2289.

References

[1] C. Whiten, R. Laganière and G.-A. Bilodeau. Efficient Action Recognition with MoFREAK, CRV, page 319-325. IEEE, 2013.

[2] F. Baumann, J. Liao, A. Ehlers, B. Rosenhahn, Motion Binary Patterns for Action Recognition, 3rd International Conference on Pattern Recognition Applications and Methods, 2014.

[3] O. Kliper-Gross, Y. Gurovich, T. Hassner, and L. Wolf, Motion Interchange Patterns for Action Recognition in Unconstrained Videos, ECCV, 2012.

[4] D. G. Lowe, Object recognition from local scale-invariant features. ICCV, 1999.

[5] B. Herbert, T. Tuytelaars, and L. Van Gool. Surf: Speeded up robust features. ECCV pp. 404-417, 2006.

[6] M. Calonder, et al. Brief: Binary robust independent elementary features. ECCV. pp. 778-792. 2010.

[7] E. Rublee, et al. ORB: an efficient alternative to SIFT or SURF. ICCV, 2011.

[8] S. Leutenegger, M. Chli, and R. Y. Siegwart. BRISK: Binary robust invariant scalable keypoints, ICCV, 2011.

[9] A. Alahi, R. Ortiz, and P. Vandergheynst. Freak: Fast retina keypoint.CVPR, 2012.

[10] H. Wang; M. M. Ullah; A. Klaser; I. Laptev; C. Schmid, Evaluation of local spatio-temporal features for action recognition, BMVC, 2009.

[11] H. Wang, A. Kläser, C. Schmid, and L. Cheng-Lin. Action Recognition by Dense Trajectories, CVPR, 2011.

[12] R. Messing, Christopher J. Pal, Henry A. Kautz. Activity recognition using the velocity histories of tracked keypoints, ICCV, 2009.

[13] B. D. Lucas and T. Kanade, An iterative image registration technique with an application to stereo vision. Proceedings of Imaging Understanding Workshop, pages 121-130, 1981.

[14] J. Sun, X. Wu, S Yan, L.-F. Cheong, T.S. Chua, and J. Li. Hierarchical spatio-temporal context modeling for action recognition, CVPR, 2009.

[15] G. Carmichael, R. Laganière, and P. Bose. Global Context Descriptors for SURF and MSER Feature Descriptors, CRV, 2010.

[16] M. Marszałek, I. Laptev, C. Schmid, Actions in context, CVPR, 2009.

[17] Miksik O. and Mikolajczyk K. Evaluation of Local Detectors and Descriptors for Fast Feature Matching, ICPR, Tsukuba Science City, Japan, 2012.

[18] E. Shechtman and M. Irani, Matching Local Self-Similarities across Images and Videos. CVPR, 2007.

[19] Mikolajczyk, Krystian, and Cordelia Schmid. A performance evaluation of local descriptors, PAMI, vol. 27, no. 10, p1615-1630, 2005.

[20] B. Fan, F. Wu, and Z. Hu, Rotationally invariant descriptors using intensity order pooling, T-PAMI, 2011.

[21] E. Tola, V. Lepetit, and P. Fua, DAISY: An Efficient Dense Descriptor Applied to Wide Baseline Stereo, PAMI, 2010.

[22] E. Tola, V. Lepetit, and P. Fua, A fast local descriptor for dense matching, CVPR, 2008.

[23] T. Trzcinski and V. Lepetit, Efficient Discriminative Projections for compact Binary Descriptors, ECCV, 2012.

[24] V. Chandrasekhar, G. Takacs, D. Chen, S. Tsai, R. Grzeszczuk, and B. Girod. CHoG: Compressed Histogram of Gradients a Low Bit-Rate Feature Descriptor, CVPR, 2009.

[25] H. Jegou, M. Douze, C. Schmid, and P. Perez. Aggregating Local Descriptors into a Compact Image Representation, CVPR, 2010.

[26] I. Laptev, M. Marszałek, C. Schmid and B. Rozenfeld, Learning realistic human actions from movies, CVPR, 2008.

[27] M. Rodriguez, J. Ahmed, and M. Shah. Action MACH: A spatio-temporal maximum average correlation height filter for action recognition, CVPR, 2008.

[28] A. Vedaldi and B. Fulkerson. VLFeat - An open and portable library of computer vision algorithms, ACM Multimedia, 2010.

[29] G. Willems, T. Tuytelaars, and L. Van Gool, An efficient dense and scale-invariant spatio-temporal interest point detector, ECCV, 2008.

[30] Please contact authors to obtain the source code.

[31] K. Lin, J. Lu, C.-S. Chen, & J. Zhou, Learning Compact Binary Descriptors With Unsupervised Deep Neural Networks, CVPR 2016.

[32] Persson, Andreas, and Amy Loutfi. Fast Matching of Binary Descriptors for Large-scale Applications in Robot Vision, IJARS, 2016.