INTRODUCING MOTION INFORMATION IN DENSE FEATURE CLASSIFIERS

Claudiu Tănase, Bernard Mérialdo

EURECOM
Campus SophiaTech
450 Route des Chappes
06410 Biot France

ABSTRACT

Semantic concept detection in large scale video collections is mostly achieved through a static analysis of selected keyframes. A popular choice for representing the visual content of an image is based on the pooling of local descriptors such as Dense SIFT. However, simple motion features such as optic flow can be extracted relatively easy from such keyframes. In this paper we propose an efficient addition to the DSIFT approach by including information derived from optic flow. Based on optic flow magnitude, we can estimate for each DSIFT patch whether it is static or moving. We modify the bag of words model used traditionally with DSIFT by creating two separate occurrence histograms instead of one: one for static patches and one for dynamic patches. We further refine this method by studying different separation thresholds and soft assignment, as well as different normalization techniques. Classifier score fusion is used to maximize the average precision of all these variants. Experimental results on the TRECVID Semantic Indexing collection show that by means of classifier fusion our method increases overall mean average precision of the DSIFT classifier from 0.061 to 0.106.

1. INTRODUCTION

The TRECVID[1] evaluation campaign is a major challenge evaluating state of the art research on large scale video collection indexing. In particular, the Semantic Indexing (SIN) track of TRECVID is a concept classification task where concepts are high level (”mountain”,”Female Person”), mostly but not exclusively visual (e.g. ”joy”, ”weather”,”amateur video”). We are investigating ways to improve concept detection in the TRECVID Semantic Indexing Task. In this paper, we are working with the video data and annotations from the 2011 edition of the TRECVID Semantic Indexing challenge. We are evaluating 50 concepts, with sparse training annotations available on a development set containing 119685 sequences, and applied on a test set of 146788 sequences. We also use the MAP (Mean Average Precision) as performance measure, as mentioned in [1]. The traditional Bag-of-Words approach involves extracting local visual features such as SIFT from a video keyframe. More recent research suggest that dense extraction of visual features ultimately gives better performance[2] than interest points. The extracted features are clustered using K-means, with a large enough value of K (typically hundreds or thousands) and the resulting cluster centroids form a codebook. Ultimately, videos are described by quantizing the extracted features into a histogram of codewords by assigning each feature to its nearest codeword. If the chosen descriptor is SIFT, this method is informally known as Dense SIFT or DSIFT. The overlapping zones around keypoints are referred to as patches. In this paper, we employ simple motion estimation techniques to compute a motion mask at keyframe level. This motion mask allows us to separate static SIFT keypoints from SIFT keypoints in motion and deal which each of these classes separately. We propose a method based on the construction of two separate Bag-of-Words histograms for the static and dynamic classes with concatenation of the resulting feature vectors.
The contribution of this paper is twofold. Firstly, we are proposing an enrichment of the DSIFT descriptor by adding motion information. In the process, 3 static-dynamic separation methods and 3 normalization techniques are tested. Secondly, we show that linear fusion of the newly obtained descriptor in several flavors can outperform individual descriptor performance and thus help increase the performance of the retrieval system.

2. RELATED WORK

Traditionally concept detection in TRECVID is done as a fusion of numerous classifiers, mostly visual, but also audio and metadata-derived[1]. One visual descriptor used in the overwhelming majority of TRECVID submissions is David Lowe’s SIFT[3]. In its original iteration SIFT was both an interest point detector and descriptor. The extrema values of a difference of gaussians at different scales was used to detect keypoints. However, more recent results[4, 2] in semantic indexing show that dense extraction may give better results than keypoints, especially in a large data context such as TRECVID. The extracted features are then processed ac-

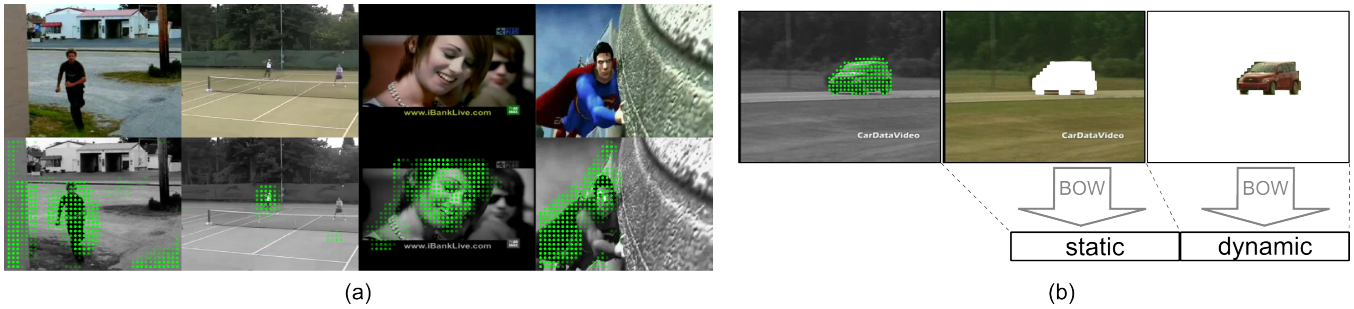

Fig. 1. (a) Examples of motion masks extracted using optic flow. Green dots are placed at the position of the SIFT keypoint. The dots color intensity is proportional to the amount of motion. (b) Separation between static and dynamic patches followed by the Bag of Words model applied separately on each of the 2 resulting sets of patches. The resulting vectors are concatenated into the final feature.

cording to the Bag of Words paradigm. Adoption of spatio-temporal and motion features is low but steadily growing since recently. One notable examples is Laptev's Spatio-Temporal Interest Point descriptor[5] (STIP), which detects 3D interest points using a 3D extension of the Harris operator and describes them using histograms of oriented gradients and histograms of flow (HOGHOF). Wang's dense trajectories descriptor [6] extracts and tracks dense features throughout the entire video volume. This approach is however more suitable to human action recognition than concept detection, and has a much higher computational cost than the proposed approach. Our work also bears some resemblance to SIFT Flow[7], in that displacements of SIFT patches are extracted, but while their method estimates the apparent motion between images representing different scenes, we use "real" motion information from video for describing the image content. Chen's MoSIFT[8] is an extension of SIFT that adds in a similar feature where the gradient is replaced by optic flow. We are not aware of any published methods that try to combine local 2D descriptor information with motion information by separating features into motion categories.

Basic methods[9] for obtaining a foreground mask are frame differencing, optic flow thresholding and background modeling. Frame differencing is fast but shows very little robustness to uniform objects, textureless zones and illumination changes. Background model methods are far more reliable but demand a stationary camera and require a longer sequence. Optic flow is a good compromise in speed and performance because it requires only a pair of frames instead of a full sequence and gives relatively accurate estimates on the displacement.

3. OBTAINING THE MOTION MASK

One way to simulate a stationary camera at frame level is to compensate for camera motion between frames. We use a camera stabilization function similar to the one in [10]. This method does dominant motion compensation by estimating a homography with RANSAC over detected feature correspondences. This homography is then used to produce a synthetic motion vector field that models the camera movement which is used as an initial estimate for the full-frame optic flow using Farneback's method[11]. The displacement between the synthetic background motion field and the actual motion field can then be used as an estimation of foreground objects. One simple way of doing this is by thresholding the flow magnitude in each pixel. If the flow magnitude is higher than a predetermined threshold, the pixel is in motion and is considered foreground (see figure 1(a)).

4. CONSTRUCTION OF THE NEW FEATURE

4.1. Codebook construction

The standard approach when classifying using bag-of-features involves the construction of a codebook. This is usually achieved by clustering the features using the K-means algorithm using euclidean distance. Each resulting cluster centroid is considered a codeword. In this paper, we follow the standard approach in computing the codebook. We empirically choose a codebook size of k=500, which we found as a reasonable compromise between performance and speed.

4.2. Static-dynamic separation

The core of our approach is in the separation between static and dynamic patches. We do this separation based on the corresponding value of compensated optic flow, which we obtain earlier for the motion mask. Figure 1 (b) shows how this separation influences the final feature. We experimented with different variants of separation.

A collection-wide statistic on flow magnitude in all keypoint positions gives us a distribution of flow velocities. We exclude from this statistic points with zero velocity flow since these points dominate the distribution without providing any information, and because failed optic flow estimation would result in zero flow. The θ_1 median of this distribution would

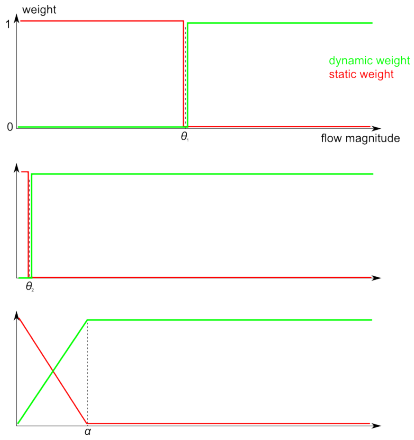

Fig. 2. The functions defining the static and dynamic weight with respect to flow magnitude in the 3 strategies: fixed threshold θ_1, fixed threshold θ_2 and soft assignment with parameter α

be the value that divides keypoints with slower motion and keypoints with faster motion into two equal sets. When constructing our feature vector, instead of quantizing all DSIFT features into one occurrence histogram, we separate the features based on this threshold and construct two histograms separately: one for the "slow" flow, which we call static, and one for the "fast" flow which we call dynamic. This is the first separation strategy.

A second separation strategy used the same rule, but with the threshold at a minimal level θ_2. This causes more patches to be considered dynamic, and only the patches that are certain to be stationary as static.

The third strategy involves soft assignment. Using a fixed threshold value means that patches with a velocity close to threshold level may fall either on the static or dynamic part, which creates noise. This can easily be avoided by making a soft assignment instead of a binary static/dynamic choice. We achieve this by assigning each patch a *static weight* w_s and a *dynamic weight* w_d with $w_s + w_d = 1$. The flow velocity v is used to directly determine the dynamic weight by using a clipped ramp function (see figure 2). The value of the α parameter has been empirically set.

4.3. Normalization

Since we use dense feature sampling, this involves that the number of features sampled in each keyframe is the same (assuming all videos have the same resolution). This makes for an implicit L1 normalization of the resulting histograms: the sum of the histogram will be equal to the total number of features in the shot, which will be constant for all shots. However, since the principle of our approach is to separate between static and dynamic zones in the image, the total number of features in each zone will vary, thus a special normalization technique is required. We compare 3 normalization strategies:

The first and most simple is the L1 normalization of the static-dynamic concatenated vector. Each element of the feature vector is divided by the total sum of the vector.

The second normalization method normalizes the static and dynamic histograms separately. Each histogram is separately normalized using L1 norm and the resulting histograms are then concatenated.

The third method is svmscale[12]. This method works by normalizing components instead of features. Each component of the feature vector is divided by the total sum of that component over all vectors.

The separation and normalization described are reasonably fast, taking in average 1.48 seconds CPU time per shot.

5. CLASSIFICATION AND FUSION

We use LibSVM to train concept classifiers. We use exponential $\chi 2$ kernels, that have proven in practice to give state of the art results in visual feature based classification. Specifically, we use a modification of the binary C-SVM implementation in LibSVM. We optimize the learning parameters C (main parameter of a C-SVM), γ ($\chi 2$ kernel width) and $w1$ (weight of the positive class) through brute force search. Each run is evaluated by measuring the Average Precision (AP). The final score of the method is the mean average precision across all concepts (mAP). For comparison we add the "default" DSIFT approach, which simply bypasses the separation stage.

We also experiment with a simple linear fusion technique. This is done by combining the score of each classifier using a weighted sum, with weights summing to one. Since the computation of weighted sums and of the mean average precision are almost instantaneous, a grid search on the weight values is possible. We experiment with the fusion of 4 best performing variants, as described in the next section. Each weight is tested in 0.1 increments.

6. EXPERIMENTAL RESULTS

In the TRECVID training dataset, the threshold values for flow are $\theta_1 = 14$, $\theta_2 = 1$, $\alpha = 5$. As described in section 4.2, θ_1 is a corpus-specific median value that splits the set of displacements into equal static and dynamic sets. Lowering the α parameter would essentially lower performance by approaching the $threshold = \theta_2$ situation, whereas excessively increasing it would lead to unbalancing between the static and dynamic classes. θ_2 should function as a noise threshold, and is chosen based on the precision of optic flow detection.

If we count the 'default' run, there are 11 valid combinations between separation 4 strategies: default (no separation), medium threshold, low threshold, and soft assignment and 3 normalization methods: global L1, separate L1 and svmscale. We exemplify the resulting combinations on 2 concepts: predominantly static "Mountain" and a predominantly dynamic

Mountain	default	trhesh=θ_1	thresh=θ_2	soft
globalL1	0.13164	0.0297	0.0581	0.0316
sepL1	N/A	0.0045	0.0173	0.0003
svmscale	0.2699	0.2866	0.1808	0.2605
Running	default	trhesh=θ_1	thresh=θ_2	soft
globalL1	0.02351	0.001	0.0235	0.0134
sepL1	N/A	0.001	0.0235	0.0009
svmscale	0.01811	0.03843	0.0198	0.0474

Table 1. Average precision of the different separation and normalization techniques for two concepts

"Running". Table 1 shows these results.

The general impact on performance is measured by averaging precision over all 50 concepts. Since thresholding methods and the sepL1 consistently gave lower performance (see table 1) we have done this only with two of the above 4 separation strategies: 'default' and 'soft' and two out of 3 normalization methods: 'globalL1' and 'svmscale'. The final run is the best performing linear fusion of the 4. The resulting average precisions are summarized in the following table:

run	default-globalL1	soft-globalL1	default-svmscale	soft-svmscale	fusion
MAP	0.0615	0.0637	0.0651	0.0701	0.1067

Table 2. Mean average precision of 4 runs and fusion

Note that "default globalL1" actually means the standard DSIFT baseline. It is clear to see that both soft assignment and svmscale normalization improve on the initial approach. The late fusion experiment improves performance significantly: in average each concept is improved by 10.05% over the best single classifier out of the 4 variants. These improved classifiers give a final mean average precision of 0.106 which is an improvement of 68.33% over the baseline DSIFT.

7. CONCLUSIONS

It is easy to observe from table 1 that separate L1 normalization is not beneficial. The explanation could be that keyframes containing very little motion have few dynamic patches. Separate L1 normalization artificially boosts the weight of these dynamic patches so that overall, static and dynamic end up having the same weight.

Results confirm that the best combination is soft assignment separation with svmscale normalization. We conclude that this is due to the fact that component based normalization deals with static and dynamic features indiscriminately. Also, svmscale mitigates the problem of having very little motion information.

In this paper we have shown that straightforward motion analysis methods can significantly improve the performances of established visual descriptors. We have proposed the usage of 3 static-dynamic feature separation strategies, as well as 3 normalization methods for the resulting features. Thus, using simple and fast motion estimation methods and with the help of efficient linear classifier fusion we increase the MAP of DSIFT from 0.061 to 0.106.

8. REFERENCES

[1] P. Over, G. Awad, J. Fiscus, B. Antonishek, M. Michel, A.F. Smeaton, W. Kraaij, G. Quénot, et al., "An overview of the goals, tasks, data, evaluation mechanisms and metrics," in *TRECVID 2011-TREC Video Retrieval Evaluation Online*, 2011.

[2] H. Wang, M.M. Ullah, A. Klaser, I. Laptev, C. Schmid, et al., "Evaluation of local spatio-temporal features for action recognition," in *BMVC 2009-British Machine Vision Conference*, 2009.

[3] D.G. Lowe, "Object recognition from local scale-invariant features," in *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*. Ieee, 1999, vol. 2, pp. 1150–1157.

[4] David Gorisse and Frédéric Precioso, "IRIM at TRECVID 2010: Semantic Indexing and Instance Search," in *TREC online proceedings*, Gaithersburg, United States, Nov. 2010, pp. – , GDR ISIS.

[5] I. Laptev and T. Lindeberg, "Space-time interest points," in *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, oct. 2003, pp. 432 –439 vol.1.

[6] H. Wang, A. Klaser, C. Schmid, and C.L. Liu, "Action recognition by dense trajectories," in *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*. IEEE, 2011, pp. 3169–3176.

[7] Ce Liu, Jenny Yuen, and Antonio Torralba, "Sift flow: Dense correspondence across scenes and its applications," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 33, no. 5, pp. 978–994, 2011.

[8] M. Chen and A. Hauptmann, "Mosift: Recognizing human actions in surveillance videos," Tech. Rep. CMU-CS-09-161, Carnegie Mellon University, 2009.

[9] Massimo Piccardi, "Background subtraction techniques: a review," in *Systems, Man and Cybernetics, 2004 IEEE International Conference on*. IEEE, 2004, vol. 4, pp. 3099–3104.

[10] N. Ikizler-Cinbis and S. Sclaroff, "Object, scene and actions: Combining multiple features for human action recognition," *Computer Vision–ECCV 2010*, pp. 494–507, 2010.

[11] Gunnar Farnebäck, "Two-frame motion estimation based on polynomial expansion," *Image Analysis*, pp. 363–370, 2003.

[12] Chih-Chung Chang and Chih-Jen Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, pp. 27:1–27:27, 2011.