PRISMA-ORAND team: Instance Search Based on Parallel Approximate Searches

Juan Manuel Barrios
ORAND S.A.
Santiago, Chile
juan.barrios@orand.cl

Benjamin Bustos
PRISMA Research Group, Department of Computer Science, University of Chile
bebustos@dcc.uchile.cl

ABSTRACT

ORAND is a Chilean company focused on developing applied research in Computer Science. This paper describes the participation of the PRISMA-ORAND team for the Instance Search task (INS) at TRECVID 2012.

Currently, the most common technique used to address the Instance Search problem is the codebook approach or some extension of it. In our participation we preferred a different approach which consisted in retrieving k nearest neighbors (k-NN) for each query descriptor on the full set of descriptors instead of computing a codebook. In order to efficiently solve the k-NN searches in large datasets we perform several parallel approximate searches using the open source project P-VCD and the Amazon Elastic Compute Cloud (EC2) service.

We submitted four Runs to TRECVID 2012 INS task, namely: prisma-two180px, prisma-four90px, prisma-two90px, and prisma-one180px. In general, these Runs achieve satisfactory performance as their MAP are higher than the median of all the Runs in most of the topics. Additionally, we tested a fifth Run prisma-two280px by increasing both number of descriptors and search accuracy. This last Run demanded higher computational power to solve the k-NN searches but it highly improved the effectiveness compared with initial four Runs.

The performance achieved by the presented approach shows some promising results. It presents an alternative to codebook approaches that can achieve satisfactory results in the Instance Search problem. The parallelization and approximation enable to solve k-NN searches in large datasets achieving high effectiveness. However, more research is needed in order to determine the potential performance that can be reached by k-NN searches compared to codebook approaches.

1. INTRODUCTION

ORAND is a Chilean software company focused on developing applied research in Computer Science. This paper describes our work for the Instance Search task (INS) at TRECVID 2012. This work is the progression of the research developed by PRISMA team (University of Chile) since TRECVID 2010 [5, 7].

TRECVID is an evaluation sponsored by the National Institute of Standards and Technology (NIST) with the goal of encouraging research in video information retrieval [15]. Instance Search task (INS) is part of TRECVID since 2010. Given some visual examples of an entity (person, object or location), a system must return a list of videos where the entity appears. Each visual example contains a mask outlining the relevant zone of the image (the region where the entity appears in the example). INS 2012 evaluated 21 topics (one person, 15 objects and 5 locations) with an average of 4.9 images per topic, on a reference video collection of 74,958 videos with a total extension of 188 hours (19 million frames). The object topics consisted in 6 trademark logos and 9 buildings. This differs with last year's evaluation which considered more persons (6), wider variety of objects (e.g., tortoise, fork, plane, car, etc.), and less visual examples (an average of 3.8 images per topic) [13].

Currently, the most common approach used to address the Instance Search problem is the well-known Bag-of-Visual-Words or codebook approach. It was introduced as a technique to perform efficient similarity searches in large video collections [14]. This approach is grounded on constructing a codebook or "visual vocabulary" in order to represent the local descriptors in videos. Briefly, the approach follows these steps: first, it extracts local descriptors from a sample of video frames. Second, it defines the codebook as the set of k centroids computed by a clustering algorithm on the extracted descriptors. In this step, the k-means algorithm is usually preferred due to its high efficiency at processing large sets of vectors. Finally, the codebook is used to produce a brief description for each frame by quantizing each local descriptor to its nearest centroid.

Using the codebook, the efficiency can be highly improved by constructing an inverted index. The inverted index enables the perform similar searches in "immediate run-time" by retrieving collisions (local descriptors assigned to the same centroid) between frames. Many systems based on the codebook approach have achieved high performance at Instance Search and other TRECVID's tasks as well as other related problems like video classification, copy detection, event detection, object recognition, etc.

Two main issues arise when following the codebook approach: the high computational cost required by the codebook creation, and the loss of information due to quantization. Many techniques has been developed to improve the performance of the codebook computation and/or to increase the information calculated from the local descriptors in a frame, e.g. soft assignment (instead of hard assignment to the nearest centroid) [16], hamming embedding [10], spatial pyramids [11], histogram of distances by codeword [4], hierarchical k-means [12], and many others.

Unlike the codebook approach, in this work we follow the k-NN approach on the full set of descriptors without computing any codebook nor quantizing local descriptors. We are interested in studying the effectiveness that can be reached when no quantization is applied to local descriptors. In this case, the main issue that we needed to address is to efficiently solve a large amount of k-NN searches in a large set of vectors. In order to address that issue, we perform several parallel approximate searches using P-VCD [2] and Amazon Elastic Compute Cloud (EC2) [1].

P-VCD is an open source project that implements different feature extraction methods, interfaces with external feature extraction methods, and implements exact [8] and approximate [6] k-NN searches following the metric space approach [17]. Amazon EC2 is a service that enables to easily create an arbitrary number of virtual machines, called instances or nodes.

2. INSTANCE SEARCH PARTICIPATION

In our participation we studied the detection performance that can be achieved by a system using local descriptors without applying any quantization nor codebook. We control the amount of local descriptors to search in by sampling video frames at a fixed rate and by shrinking each video frame to a fixed height. Visual samples are also scaled to the same height.

The set of local descriptors for the visual samples corresponds to the query set $\mathcal{Q}$, and the set of local descriptors for the reference video collection corresponds to the search space $\mathcal{R}$. Therefore, the search process aims to retrieve for each vector in $\mathcal{Q}$ the k nearest neighbors in $\mathcal{R}$. However, due to the size of the video collection, comparing every vector in $\mathcal{Q}$ with every vector in $\mathcal{R}$ drives to an unaffordable search time.

The efficiency (search time) can be improved by reducing $\mathcal{Q}$ and $\mathcal{R}$ and by using approximate searches. However, as shown below, in order to spend a reasonable search time in a single machine, the parameter values for sampling rate, frame size, and search approximation produce a setup which effectiveness (MAP) is practically ruined. In order to increase effectiveness, we solved similarity searches using many temporary virtual machines rented from Amazon EC2. This allowed us to reach a setup with higher effectiveness without compromising the efficiency.

As a general overview, our approach follows these steps: the video dataset is partitioned into subsets of videos and each one is assigned to a different machine in order to process them in parallel. Each machine computes the set of descriptors by extracting local descriptors from video frames sampled at a regular-step, and then computes an index (pivots) for its local dataset. For each local descriptor in a query image, each machine resolves in parallel a partial approximate k-NN search using the P-VCD software. The partial results are merged and the video candidates are calculated according to the number of nearest neighbors located for each query image. Finally, the results of different descriptors are combined by score aggregation of candidates.

We submitted four Runs, namely: prisma-two180px, prisma-four90px, prisma-two90px, and prisma-one180px. Additionally, we create an extra Run (not submitted to evaluation) called prisma-two280px, which is essentially identical to prisma-two180px but with higher sampling rate and frame size. A summary for these Runs is shown in Table 1 and Table 2.

2.1 Sampling and Feature Extraction

Every video in the reference video collection was processed in order to extract representative keyframes. We tested three sampling rates:

- One frame every 5 seconds. This rate selects 142,749 frames for the whole reference video collection.
- One frame every 1.5 seconds. This rate selects 479,492 frames for the whole reference video collection.
- One frame every 0.5 seconds. This rate selects 1,398,505 frames for the whole reference video collection.

For each selected frame we extracted local descriptors using the FeatureSpace software [3]. The keypoints detectors were Hessian-Laplace and MSER (Maximally Stable Extremal Regions). In order to control de number of keypoints, each frame was scaled down to height 90, 180 or 280 pixels, and the width was fixed accordingly to the aspect ratio. At each keypoint we extracted SIFT (Scale-Invariant Feature Transform, 128-d) descriptors and/or CSIFT (Color SIFT, 192-d) descriptors:

- SH: SIFT descriptor at Hessian-Laplace keypoints.
- SM: SIFT descriptor at MSER keypoints.
- CH: CSIFT descriptor at Hessian-Laplace keypoints.
- CM: CSIFT descriptor at MSER keypoints.

The sampling rate, frame size, and feature extraction method used by each Run are summarized in Table 1.

2.2 Similarity Searches

On the query side ($\mathcal{Q}$), we extracted descriptors from query images and flipped versions. On the reference side ($\mathcal{R}$), we extracted descriptors at sampled frames from reference videos. Then, for each vector in $\mathcal{Q}$ we performed a k-NN search to retrieve its k=50 nearest neighbors in $\mathcal{R}$ according to distance:

$$L_1(\vec{x}, \vec{y}) = \sum_{i=0}^{d} |x_i - y_i|$$

Table 2 summarizes the number of vectors for $\mathcal{Q}$ and $\mathcal{R}$ involved for each Run. For instance, prisma-one180px required to solve 75,236 searches in a dataset of 166,431,407 vectors of 192-d, and prisma-two280px required to solve 155,138 searches in 973,691,517 vectors plus 94,050 searches in 542,942,624 vectors of 192-d. Given these sizes, a brute-force search (i.e., linear scan) becomes unfeasible.

The reference set $\mathcal{R}$ is partitioned into n subsets $\{\mathcal{R}_1, ..., \mathcal{R}_n\}$, where:

$$\mathcal{R} = \bigcup_{i=1}^{n} \mathcal{R}_i$$

$$\forall\, i \neq j, \mathcal{R}_i \bigcap \mathcal{R}_j = \emptyset$$

Each subset is assigned to an Amazon EC2 node, and each node performed the approximate search between $\mathcal{Q}$ and $\mathcal{R}_i$.

Run	Sampling Rate [sec.]	Frame Height [pix.]	Descriptor	MAP
*prisma-two280px	0.5	280	CH+CM	0.210
prisma-two180px	1.5	180	CH+CM	0.155
prisma-one180px	1.5	180	CH	0.140
prisma-two90px	5	90	CH+SH	0.103
prisma-four90px	5	90	CH+CM+SH+SM	0.094

* extra run (not submitted).

Table 1: Summary of features extracted to reference videos.

Run	$\mathcal{Q}$ [$*10^3$]	$\mathcal{R}$ [$*10^6$]	EC2 nodes	T	MAP
*prisma-two280px	155 + 94	973 + 543	120	1%	0.210
prisma-two180px	75 + 44	166 + 95	20	0.5%	0.155
prisma-one180px	75	166	10	0.5%	0.140
prisma-two90px	22 + 22	15 + 15	1	1%	0.103
prisma-four90px	22 + 15 + 22 + 15	15 + 9 + 15 + 9	1	0.5%	0.094

* extra run (not submitted).

Table 2: Summary of vectors involved in the approximate searches.

Once completed all the partial searches, the final result is created by merging the partial results.

The approximate search is an adaptation of the approximate search with pivots [6] to local descriptors. This search first selects $\mathcal{P}$ vectors from $\mathcal{R}_i$ and calculates their distance to every other vectors in $\mathcal{R}_i$. During the search, for any two vectors $q \in \mathcal{Q}$ and $r \in \mathcal{R}_i$ an estimation of $L_1(q,r)$ is calculated using the triangle inequality:

$$L_1(q,r) \approx \max_{p \in \mathcal{P}} |L_1(q,p) - L_1(p,r)|$$

The search used $|\mathcal{P}|$=5 pivots (selected by the SSS algorithm [9]) and the approximation parameter T shown in Table 2. Each distance estimation cost only 5 operations (instead of the 192 operations for CSIFT vectors). Thereafter, for the $T \cdot |\mathcal{R}_i|$ vectors with lowest estimations the real $L_1(q,r)$ is calculated, and the k=50 nearest neighbors are located.

2.3 Locating candidates

Given a topic, a similarity search is performed for each descriptor in each query image. Given the query descriptor q, its k-NN in $\mathcal{R}$ are located. Every NN sums one vote to the reference video that owns it. The vote is weighted by 2 if q resides inside the mask, and is weighted by 1.5 is q is close to the mask (i.e., q resides in the mask dilated by a 5x5 kernel). The candidate videos with more votes produce the final result sorted by score (total sum of votes).

For Runs with multiple descriptors, an independent search is performed for each kind of descriptor, and the candidate videos from each modality are combined by score aggregation (i.e., late fusion).

The final candidates are sorted from maximum to minimum score and the top 1000 scores are reported.

3. RESULTS ANALYSIS

The evaluation of a Run relies on calculating the precision at the ranks where each correct answers is located. For each topic the average precision (AP) is computed, and the mean average precision (MAP) for the whole run is calculated. Additionally, each submission must report the time spent in processing each topic. Two kinds of submissions were evaluated: interactive (where a user can correct the results and give feedback to the system), and automatic (where the system gives the results without user interaction). Our submissions are automatic, hence we compare the performance against the other automatic submissions.

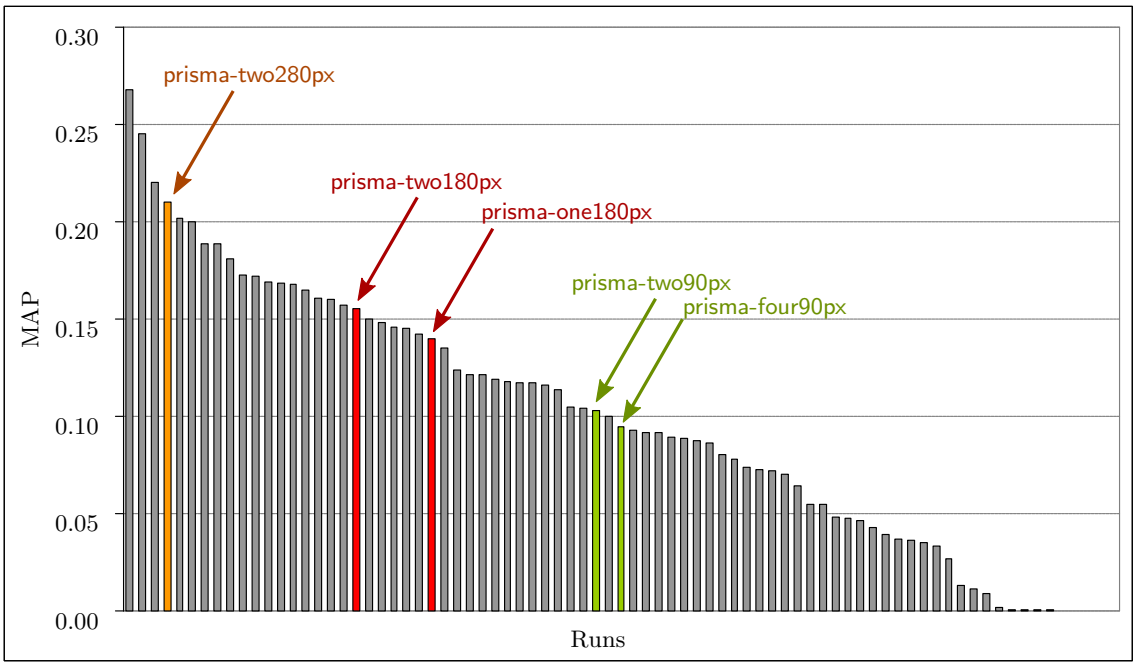
Twenty-four teams participated in the evaluation. Each team submitted at most four Runs, which resulted in 85 submissions: 6 interactive and 79 automatic. The results show this year the task was really difficult. In fact, MAP for the top performing system dropped from more than 0.53 last year to less than 0.27 this year.

Table 1 and Table 2 show the MAP four our submissions, which range from 0.094 up to 0.210. Figure 1 compares these MAP values with all the automatic Runs submitted to evaluation. The results show the high impact of sampling rate and frame size in the MAP, which is directly correlated with the resources needed to solve similarity searches. In fact, prisma-two90px and prisma-four90px have small sets $\mathcal{Q}$ and $\mathcal{R}$, thus the search can be solved on a single machine, but they both achieve a MAP near the median (ranks 37^{th} and 39^{th}, respectively). Using parallelism it is possible to tackle larger search spaces: prisma-one180px and prisma-two180px achieve higher MAP (ranks 24^{th} and 18^{th}, respectively) but they require more computational resources (10 and 20 nodes in parallel were used, respectively).

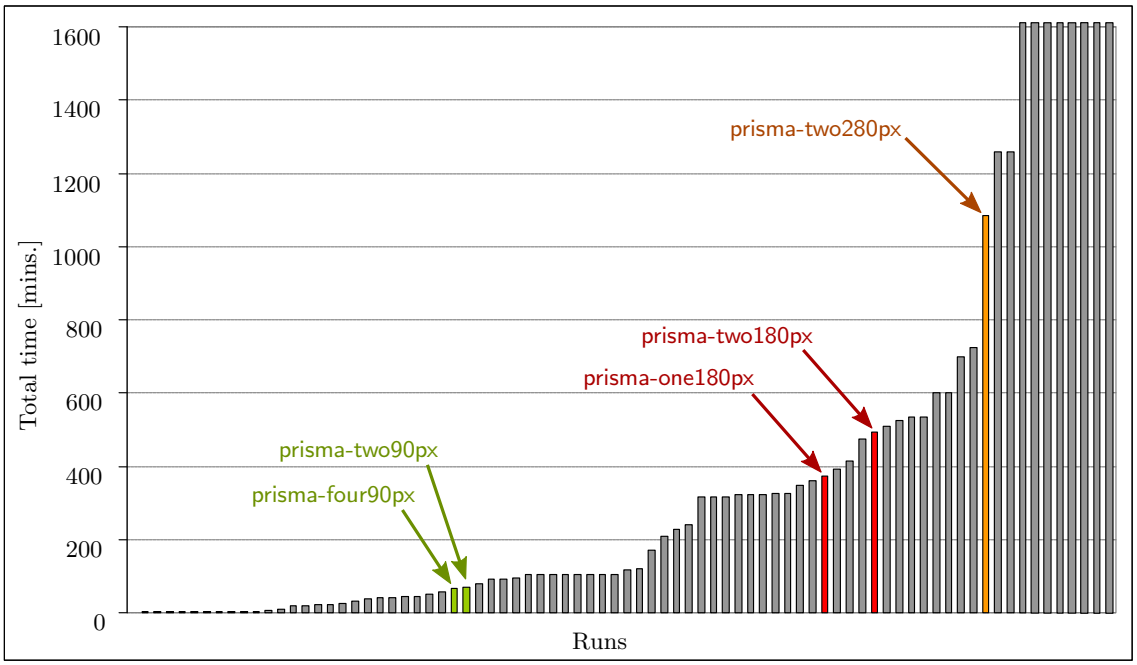
An interesting result is that prisma-two280px achieve near the highest MAP (4^{th} of 79). This result proves that k-NN searches can indeed achieve as high effectiveness as codebook approach. However, this Run demanded even more computational resources (120 nodes in parallel were used).

Regarding results by topic, Table 3 and Figure 2 compares the AP for our best Runs with the maximum AP and median AP at every topic. We achieved top result for 9061 (Pepsi logo), a near-top for 9063 (Prague Castle), and above or in the median for the others.

In general, the approach shows promising results. The results prove it is possible to address the Instance Search



(a) Effectiveness (MAP) achieved by all Runs.



(b) Efficiency (processing time) achieved by all Runs.

Figure 1: Global results for all the automatic submissions.

Topic	Max	Median	prisma-one180px		prisma-two180px		prisma-two280px*	
	AP	AP	AP	#	AP	#	AP	#
9048 - Mercedes star	0.115	0.000	0.003	7th	0.001	17th	0.000	30th
9049 - Brooklyn bridge tower	0.319	0.004	0.036	14th	0.038	12th	0.077	8th
9050 - Eiffel tower	0.370	0.033	0.123	17th	0.136	15th	0.188	11th
9051 - Golden Gate Bridge	0.633	0.107	0.053	51th	0.249	21th	0.292	18th
9052 - London Underground logo	0.310	0.009	0.053	21th	0.130	11th	0.309	2th
9053 - Coca-cola logo - letters	0.500	0.005	0.015	31th	0.066	18th	0.185	10th
9054 - Stonehenge	0.166	0.000	0.019	14th	0.022	13th	0.016	15th
9055 - Sears/Willis Tower	0.172	0.042	0.121	14th	0.086	17th	0.121	15th
9056 - Pantheon interior	0.820	0.463	0.591	28th	0.588	29th	0.679	16th
9057 - Leshan Giant Buddha	0.383	0.204	0.248	28th	0.265	23th	0.363	2th
9058 - US Capitol exterior	0.556	0.043	0.058	33th	0.074	27th	0.175	15th
9059 - baldachin in Saint Peter's Basilica	0.717	0.009	0.009	40th	0.014	38th	0.157	14th
9060 - Stephen Colbert	0.761	0.529	0.625	24th	0.583	30th	0.583	31th
9061 - Pepsi logo - circle	0.042	0.000	0.004	5th	**0.042**	**1th**	0.045	1th
9062 - One World Trade Center building	0.360	0.001	0.041	15th	0.046	11th	0.043	14th
9063 - Prague Castle	0.254	0.022	0.253	2th	0.241	3th	0.258	1th
9064 - Empire State Building	0.084	0.001	0.002	20th	0.001	24th	0.001	42th
9065 - Hagia Sophia interior	0.588	0.043	0.478	7th	0.459	8th	0.578	2th
9066 - Hoover Dam exterior	0.318	0.001	0.002	35th	0.005	27th	0.010	25th
9067 - MacDonald's arches	0.200	0.000	0.000	12th	0.000	12th	0.000	12th
9068 - PUMA logo animal	0.491	0.155	0.208	32th	0.212	31th	0.329	10th
MAP	0.268	0.093	**0.140**	**24th**	**0.155**	**18th**	**0.210**	**4th**

Table 3: Results per topic for prisma-one180px, prisma-two180px, **and** prisma-two280px*.
* extra run (not submitted).

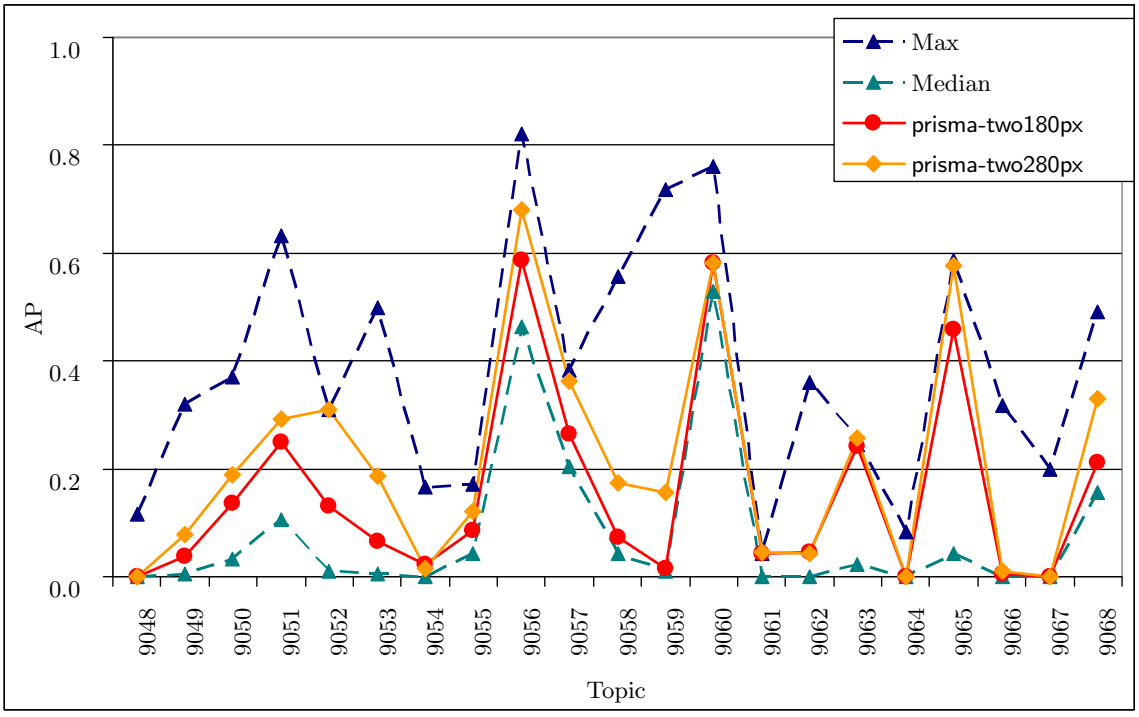


Figure 2: Results per topic for prisma-two180px **and** prisma-two280px* **compared with maximum and median AP.**

problem without using codebooks. Moreover, resolving k-NN searches in large search spaces it is possible to outperform many codebook-based systems.

4. CONCLUSIONS

In this work we present an alternative approach to codebooks that can achieve high effectiveness at the Instance Search problem. It is based on performing k-NN searches for local descriptors without applying any quantization. In order to achieve high efficiency and effectiveness we processed a large video collection using many virtual machines and performing several parallel approximate searches. We tested our approach using the Amazon Elastic Compute Cloud (EC2).

The achieved results prove the presented approach can achieve high effectiveness, hence it is a valid alternative to address the Instance Search problem. The main issue that arises with k-NN searches is that it requires high computational resources during the online phase, unlike codebook approach which requires high computational resources during the offline phase.

After analyzing our results a research question directly arises: if we had sufficiently high computational resources, would k-NN searches outperform codebooks in large datasets? or would codebook approach still be preferable? Unfortunately, the results shown in this work are not conclusive and the dataset was not ideal to answer this question. Hence, more research is needed in order to understand the features that are captured and the topics that can be detected by following the k-NN searches and codebook approaches.

P-VCD is an open source software with GPL license written in C. It was originally designed as an engine for content-based video copy detection system, and now we have extended it to address the Instance Search problem. Its development is currently supported by ORAND. The project is still immature, but we encourage researchers and advanced users to test its performance.

5. REFERENCES

[1] Amazon EC2. `http://aws.amazon.com/es/ec2/`.

[2] P-VCD. `http://sourceforge.net/projects/p-vcd/`.

[3] Feature Detection Code., 2010. `http://www.featurespace.org/`.

[4] S. Avila, N. Thome, M. Cord, E. Valle, and A. Araujo. Bossa: Extended bow formalism for image classification. In *Proc. of the int. conf. on Image Processing (ICIP)*, pages 2909–2912. IEEE, 2011.

[5] J. M. Barrios and B. Bustos. Content-based video copy detection: Prisma at trecvid 2010. In *Proc. of TRECVID*. NIST, USA, 2010.

[6] J. M. Barrios and B. Bustos. Competitive content-based video copy detection using global descriptors. *Multimedia Tools and Applications*, 62(1):75–110, 2013.

[7] J. M. Barrios, B. Bustos, and X. Anguera. Combining features at search time: Prisma at video copy detection task. In *Proc. of TRECVID*. NIST, USA, 2011.

[8] J. M. Barrios, B. Bustos, and T. Skopal. Snake table: A dynamic pivot table for streams of k-nn searches. In *Proc. of the int. workshop on Similarity Search and Applications (SISAP)*. Springer, 2012. To appear.

[9] B. Bustos, O. Pedreira, and N. Brisaboa. A dynamic pivot selection technique for similarity search. In *Proc. of the int. workshop on Similarity Search and Applications (SISAP)*, pages 105–112. IEEE, 2008.

[10] H. Jégou, M. Douze, and C. Schmid. Hamming embedding and weak geometric consistency for large scale image search. In *Proc. of the european conf. on Computer Vision (ECCV)*, pages 304–317. Springer, 2008.

[11] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *Proc. of the intl. conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 2169–2178 Vol.2. IEEE, 2006.

[12] D.-D. Le, C.-Z. Zhu, S. Poullot, V. Q. Lam, D. A. Duong, and S. Satoh. National institute of informatics, japan at trecvid 2011. In *Proc. of TRECVID*. NIST, USA, 2011.

[13] P. Over, G. Awad, M. Michel, J. Fiscus, G. Sanders, B. Shaw, W. Kraaij, A. F. Smeaton, and G. Quéenot. Trecvid 2012 – an overview of the goals, tasks, data, evaluation mechanisms and metrics. In *Proc. of TRECVID*. NIST, USA, 2012.

[14] J. Sivic and A. Zisserman. Video google: A text retrieval approach to object matching in videos. In *Proc. of the IEEE int. conf. on Computer Vision (ICCV)*, pages 1470–1477. IEEE, 2003.

[15] A. F. Smeaton, P. Over, and W. Kraaij. Evaluation campaigns and TRECVid. In *Proc. of the int. workshop on Multimedia Information Retrieval (MIR)*, pages 321–330. ACM, 2006.

[16] J. van Gemert, J.-M. Geusebroek, C. Veenman, and A. Smeulders. Kernel codebooks for scene categorization. In *Proc. of the european conf. on Computer Vision (ECCV)*, pages 696–709. Springer, 2008.

[17] P. Zezula, G. Amato, V. Dohnal, and M. Batko. *Similarity Search: The Metric Space Approach (Advances in Database Systems)*. Springer, 2005.