Indexing Motion Detection Data for Surveillance Video

Vind, Søren Juhl; Bille, Philip; Gørtz, Inge Li

Published in:
Proceedings of the IEEE International Symposium on Multimedia (ISM2014)

Link to article, DOI:
10.1109/ISM.2014.36

Publication date:
2014

Document Version
Peer reviewed version

Indexing Motion Detection Data for Surveillance Video ⋆

Søren Vind, Philip Bille †, and Inge Li Gørtz †
Department of Applied Mathematics and Computer Science
Technical University of Denmark
Copenhagen, Denmark
sovi@dtu.dk, phbi@dtu.dk, inge@dtu.dk

***Abstract*—We show how to compactly index video data to support fast *motion detection* queries. A query specifies a time interval T, a area A in the video and two thresholds v and p. The answer to a query is a list of timestamps in T where $\geq p\%$ of A has changed by $\geq v$ values.**

Our results show that by building a small index, we can support queries with a speedup of two to three orders of magnitude compared to motion detection without an index. For high resolution video, the index size is about 20% of the compressed video size.

***Keywords*-motion detection index; motion detection data structure; surveillance video; video analysis; data structure;**

I. Introduction

Video data require massive amounts of storage space and substantial computational resources to subsequently analyse. For motion detection in video surveillance systems, this is particularly true, as the video data typically have to be stored (in compressed form) for extended periods for legal reasons and motion detection requires time-consuming decompressing and processing of the data. In this paper, we design a simple and compact index for video data that supports efficient motion detection queries. This enables fast motion detection queries on a selected time interval and area of the video frame without the need for decompression and processing of the video file.

A. Problem & Goal

A *motion detection query* $\texttt{MD}(\texttt{T}, \texttt{A}, \texttt{v}, \texttt{p})$ specifies a time range T, an area A, and two thresholds $v \in [0, 255]$ and $p \in [0, 100]$. The answer to the query is a list of timestamps in T where the amount of motion in A exceeds thresholds v and p, meaning that $\geq p\%$ of the pixels in A changed by $\geq v$ pixel values. Our goal is build an index for video data that supports motion detection queries. Ideally, the index should be small compared to the compressed size of the video data and should support queries significantly faster than motion detection without an index.

⋆ Supported by a grant from the Danish National Advanced Technology Foundation.

† Supported by a grant from the Danish Council for Independent Research | Natural Sciences.

B. Related Work

Several papers have considered the problem of *online* motion detection, where the goal is to efficiently identify movement in the video in real time, see e.g. [1], [2], [3], [4], [5]. Previous papers [6], [7] mentions indexing movement of objects based on motion trajectories embedded in video encoding. However, to the best of our knowledge, our solution is the first to show a highly efficient index for motion detection queries on the raw video.

C. Our Results

We design a simple index for surveillance video files, which support motion detection queries efficiently. The performance of the index is tested by running experiments on a number of surveillance videos that we make freely available for use. These test videos capture typical surveillance camera scenarios, with varying amounts of movement in the video.

Our index reduces the supported time- and area-resolution of queries by building summary *histograms* for the number of changed pixels in a number of *regions* of frames succeeding each other. Histograms for a frame are compressed and stored using an off-the-shelf compressor. Queries are answered by decompressing the appropriate histograms and looking up the answer to the query in the histograms.

The space required by the index varies with the amount of motion in the video and the region resolution supported. The query time only varies slightly. Compared to motion detection without an index we obtain:

- A query time speedup of several orders of magnitude, depending on the resolution of the original video. The choice of compressor has little influence on the time required to answer a query by the index.
- A space requirement which is $10 - 90\%$ of the compressed video. The smallest relative space requirement occur for high resolution video. Quadrupling the region resolution roughly doubles the space use.

Furthermore, as the resolution of the video increases, the time advantage of having an index grows while the additional space required by the index decreases compared to the compressed video data. That is, the index performs increasingly better for higher resolution video.

II. The Index

A $\texttt{MD}(\texttt{T}, \texttt{A}, \texttt{v}, \texttt{p})$ query spans several dimensions in the video file: The time dimension given by T and two spatial dimensions given by A. However, as high-dimensional data structures for range queries typically incur high space cost, we have decided to not implement our index using such data structures. Instead, we create a large number of two-dimensional data structures for the pixel value difference for each successive pair of frames, called a *difference frame*. Answering a query then involves querying the data structures for all difference frames in T.

We restrict the query area A to always be a collection of *regions*, $r_1, \ldots, r_k$. The height and width of a region is determined by the video resolution and the number of regions in each dimension of the video (if other query areas are needed, the index can be used as a filter). For simplicity, we assume that the pixel values are grey-scale.

The index stores the following. For each region r and difference frame F, we store a histogram $H_{F,r}$, counting for each value $0 \leq c \leq 255$ the number of pixels in the region changed by at least c pixel values. Clearly a histogram can be stored using 256 values only. While this may exceed the number of pixels in a region when storing many regions per frame, it generally does not. However, because modern video encoding is extremely efficient, the raw histograms may take more space than the compressed video (especially for low video resolutions). Thus, we compress the histograms using an off-the-shelf compressor before storing them to disk.

To answer a $\texttt{MD}(\texttt{T}, \texttt{A}, \texttt{v}, \texttt{p})$ query, we decompress and query the histograms for each region in A across all difference frames in T. Let $|r|$ denote the number of pixels in region r. For a specific difference frame F, we calculate $p' = \sum_{r \in A} H_{F,r}[v] / \sum_{r \in A} |r|$, which is exactly the percentage of pixels in A changed by $\geq v$ pixel values. Thus, if $p' \geq p$, frame F is a matching timestamp.

III. Experiments

A. Experimental setup

All experiments ran on an Apple Macbook Pro with an Intel Core i7-2720QM CPU, 8GB ram and a 128GB Apple TS128C SSD disk, plugged into the mains power. All reported results (both time and space) were obtained as the average over three executions (we note that the variance across these runs was extremely low).

B. Data sets

We tested our index on the following three typical video surveillance scenarios, encoded at 29.97fps using H264/MP4 (reference [8]). We use different video resolutions (1920×1080, 1280×720 and 852×480 pixels). See Table I.

1) Office: Recording of typical workday activities in a small well-lit office with three people moving. The image is almost static, only containing small movements by the people. There is very little local motion in the video.

Table I
Surveillance video recording samples used for testing. Videos were encoded at 29.97fps using H264/MP4.

Scenario	Length (s)	Motion amount	Size (MB) 1080p	720p	480p
Office	60	Low	9.0	3.0	1.2
Students	60	Medium	27.3	7.8	3.3
Rain	60	High	67.6	18.1	4.6

2) Students: Recording of a group of students working in small groups, with trees visible through large windows that give a lot of reflection. People move about, which gives a medium amount of motion across most of the frame.

3) Rain: A camera mounted on the outside of a building, recording activities occurring along the building and looking towards another building. It is windy and raining, which combined with many trees in the frame creates a high amount of motion across the entire frame.

C. Implementation

The system was implemented in Python using bindings to efficient C/C++ libraries where possible. In particular, OpenCV and NumPy were used extensively, and we used official python bindings to the underlying C/C++ implementations of the compressors. The implementation uses a number of tunable parameters, see Table II. The source code can be found at [8].

IV. Main Results

We now show the most significant experimental results for our index compared to the trivial method. We show results on both query time and index space when applicable. Unless otherwise noted, the index was created for a video size of 1080p, storing 1024 regions/frame, 3 frames/second, 1 frame/file, using linear packing and zlib-6 compression. We will only give detailed results for the students scenario, as the index performs relatively worst in this case.

A. Regions Queried

The first set of experiments show the influence of the number of regions queried in the image on the total query time and also check if one scenario diverts significantly from

Table II
Tunable parameters for use in experiments.

Name	Description
Frames/Second	The frame rate of the difference frames to index.
Regions/Frame	The number of regions to divide a frame into.
Compressor	The compressor used to compress the histograms.
Frames/File	The number of frames for which the histograms should be stored in the same file on disk
Packing	Which strategy should be used when storing histograms for more than one frame in same file on disk

Table III
INDEX QUERY TIME VERSUS VIDEO QUERY TIME FOR 1 AND 1024 REGIONS QUERIED. SIZE OF INDEX COMPARED TO THE VIDEO SIZE.

		Time (s)			
Scenario	**Query Reg.**	**Index**	**Video**	**Speedup**	**Size**
Office	1	0.17	463.51	2726×	24.4%
	1024	0.81	467.17	576×	2.2 MB
Students	1	0.20	249.76	1248×	20.1%
	1024	0.83	253.86	305×	5.5 MB
Rain	1	0.20	351.40	1757×	8.0%
	1024	0.83	355.98	428×	5.4 MB

Table IV
SPEEDUP FOR INDEX QUERY TIME COMPARED TO VIDEO QUERY TIME FOR STUDENTS SCENARIO, WITH VARYING INPUT VIDEO RESOLUTIONS AND NUMBER OF REGIONS QUERIED.

	Speedup / Query Reg.			
Resolution	**1**	**64**	**1024**	**Size**
480p	454×	368×	102×	75.8%
720p	903×	745×	219×	47.4%
1080p	1248×	1060×	305×	20.1%

the others in query time performance. The number of regions queried was both extremes (1 and 1024).

Table III summarises the results, with the index size shown relative to the video size. The query time of the index does not vary with the video input, while that is the case for the video compression approach. Observe that though the total time spent answering a query using the index scales with the number of regions queried, it never exceeds 1 s, while the video approach spends at least 250 s in all cases. Thus, the index answers queries at least two orders of magnitude faster than the video compression approach.

The very small difference in query time for both extremes is surprising, since it directly influences the number of pixels to analyse in each difference frame. The relative increase is much larger for the index query than the video, meaning that the time spent performing the actual query is a larger fraction of the total query time for the index (as shown in Section V-B). We believe that the reason the office scenario has the worst performance is that the video compression is most efficient here (and thus harder to decompress).

Table IV shows that the relative index query time increases when fewer regions are queried. However, even when querying all regions the index has a performance which is at least two orders of magnitude quicker than the video.

B. Resolution Comparison

Table IV show the influence of the video resolution on the speedup obtained for the students scenario. The difference in space required for the index with varying resolutions is shown in Table V. Note that the index times are almost always the same (varies between 0.2s and 1s), while the video query times decrease from around 250s at 1080p to 75s at 480p, and we thus only report the relative speedup for different numbers of regions queried. The relative index performance compared to the video approach improves in both space and time with larger video resolution. The index query time varies very little with the resolution (which is as expected, since the number of histogram values to check does not change).

Table V
SIZE OF INDEXES FOR VARYING INPUT VIDEO RESOLUTIONS.

	Index Size (MB)			Index Size (%)		
Scenario	**1080p**	**720p**	**480p**	**1080p**	**720p**	**480p**
Office	2.2	1.5	1.1	24.4%	50.0%	91.7%
Students	5.5	3.7	2.5	20.1%	47.4%	75.8%
Rain	5.4	3.7	2.2	8.0%	20.4%	47.8%

Table VI
INDEX SIZE FOR VARYING NUMBER OF STORED REGIONS AND RESOLUTIONS IN THE STUDENTS SCENARIO.

	Index Size (MB)			Index Size (%)		
Store Reg.	**1080p**	**720p**	**480p**	**1080p**	**720p**	**480p**
64	1.1	0.8	0.6	4.0%	10.2%	18.2%
256	2.3	1.7	1.2	8.4%	21.8%	36.4%
1024	5.5	3.7	2.5	20.1%	47.4%	75.8%

C. Regions Stored

Clearly, the number of regions stored by the index has an influence on the index size (as this directly corresponds to the number of histograms to store). However, from Table VI, it is clear that the influence is smaller than would be expected. In fact, due to the more efficient compression that is achieved, quadrupling ($4\times$) the number of regions only causes the index to about double ($2\times$) in size. That is, it is relatively cheap to increase the resolution of regions.

V. OTHER RESULTS

In this section, we review the index performance when varying the different parameters listed in Table II. Changing the index parameters had insignificant influence on query times, so we only show results for the index space when varying the parameters. The largest contributor to the index advantage over the video decompression method is the idea of storing an index, and thus we only briefly review the results when varying the parameters.

A. Compressor

Table VII shows the size of the index when the histograms are compressed using a number of different compressors, and Table VIII shows the time spent compressing the index in total (remember the input is a 60s video file). In all of the tests in the previous section, we have used the `zlib-6`

Table VII
INDEX SIZE COMPARISON FOR DIFFERENT COMPRESSORS.

	Index Size (MB) / Compressor				
Scenario	`lz4`	`snappy`	`zlib`	`lzma`	`bzip2`
Office	2.9	6.6	2.2	1.7	1.5
Students	7.5	10.9	5.5	4.1	3.5
Rain	7.4	10.6	5.4	4.2	3.4

Table VIII
INDEX COMPRESSION TIME FOR DIFFERENT COMPRESSORS.

	Compression Time (s) / Compressor				
Scenario	`lz4`	`snappy`	`zlib`	`lzma`	`bzip2`
Office	0.04	0.07	1.01	29.67	30.67
Students	0.07	0.11	1.29	32.69	31.80
Rain	0.07	0.11	1.41	31.89	31.92

compressor, as it gives a good tradeoff between compression time and space use. If one can spare the computing resources, there is hope for almost halving the index space if switching to `bzip2` compression. Note, however, that the query time increases with a slower decompressor (especially when querying few regions, as shown in Section V-B).

B. Query Time Components

To see the influence of the compressor used, we determined how much of the total query time is spent checking the decompressed values, compared to the amount of time spent decompressing the video frames or histograms. The results are in Table IX. It is evident that the video resolution and our chosen index compressor only has a small influence on the total query time when the number of regions queried is large: Around 75% of the time is spent on the actual query. As for the video query time, $> 95\%$ of the total time is spent decompressing the video, with the actual query time being very insignificant in all cases. In absolute terms, the query times for the index and the decompressed video approach are comparable (difference around $10\times$) after decompression.

C. Frames/File & Histogram Packing Strategies

We tested the influence on the index size if storing more than one difference frame per file on disk. We tested four different value packing strategies: linear, binned, reg-linear, reg-binned. Consider a frame F with two region histograms r_1, r_2. In the linear strategy, we just write all values from r_1 followed by all values from r_2. In the binned strategy, we interleave the value for the same histogram index from all regions in a frame, i.e. we wite $r_1[0]r_2[0]r_1[1]r_2[1]\ldots r_1[255]r_2[255]$ on disk. When storing multiple frames F_1, F_2 in a file, assume r_1, r_2 has the same spatial coordinate in both frames. Then the reg-linear strategy writes r_1 followed by r_2, while the reg-binned strategy interleaves the values as before.

Table IX
FRACTION OF TIME SPENT ANALYSING REGIONS OF TOTAL QUERY TIME FOR STUDENTS SCENARIO (REMAINDER IS DECOMPRESSION).

	Index Query Time		Video Query Time		
Resolution	**1**	**1024**	**1**	**1024**	**Size**
480p	2.63%	76.73%	0.01%	1.48%	75.8%
720p	3.27%	76.84%	0.01%	1.84%	47.4%
1080p	3.08%	73.79%	0.02%	3.84%	20.1%

The hope is that this would result in more efficient compression, since the histogram for the same region may be assumed to be very similar across neighbouring frames. However, our results show that storing more frames per file and changing the packing strategy had very little effect on the index efficiency for storing many regions. One exception is when storing few regions (less than 64), increasing the number of frames per file decreases the index size due to the added redundancy available for the compressor.

VI. CONCLUSION

We have shown an index for motion detection data from surveillance video cameras that provides a speedup of at least two orders of magnitude when answering motion detection queries in surveillance video. The size of the index is small compared to the video files, especially for high resolution video.

REFERENCES

[1] T. S. Sachs, C. H. Meyer, B. S. Hu, J. Kohli, D. G. Nishimura, and A. Macovski, "Real-time motion detection in spiral mri using navigators," *Magnetic resonance in medicine*, vol. 32, no. 5, pp. 639–645, 1994.

[2] Y.-L. Tian and A. Hampapur, "Robust salient motion detection with complex background for real-time video surveillance," in *WACV 2005*.

[3] W. Hu, T. Tan, L. Wang, and S. Maybank, "A survey on visual surveillance of object motion and behaviors," *IEEE Trans. SMC*, vol. 34, no. 3, pp. 334–352, 2004.

[4] R. Cutler and L. S. Davis, "Robust real-time periodic motion detection, analysis, and applications," *IEEE Trans. PAMI*, vol. 22, no. 8, pp. 781–796, 2000.

[5] S.-C. Huang, "An advanced motion detection algorithm with video quality analysis for video surveillance systems," *IEEE Trans. CSVT*, vol. 21, no. 1, pp. 1–14, 2011.

[6] S. Du, C. A. Rahman, S. Sharmeen, and W. Badawy, "Event detection by spatio-temporal indexing of video clips." *IJCTE*, vol. 6, no. 1, 2014.

[7] W. C. Kao, S. H. Chiu, and C. Y. Wen, "An effective surveillance video retrieval method based upon motion detection," *IEEE ISI 2008*, pp. 261–262.

[8] https://github.com/sorenvind/phd-motiondetectionindex.