

A Compact Representation of Raster Time Series

Nataly Cruces*, Diego Seco^{*†}, and Gilberto Gutiérrez[‡]

*Universidad de Concepción [†]IMFD - Millennium Institute [‡]Universidad del Bío-Bío
Concepción, Chile for Foundational Research Chillán, Chile
`{natalycruces,dseco}@udec.cl` on Data, Chile `ggutierr@ubiobio.cl`

Abstract

The raster model is widely used in Geographic Information Systems to represent data that vary continuously in space, such as temperatures, precipitations, elevation, among other spatial attributes. In applications like weather forecast systems, not just a single raster, but a sequence of rasters covering the same region at different timestamps, known as a raster time series, needs to be stored and queried. Compact data structures have proven successful to provide space-efficient representations of rasters with query capabilities. Hence, a naive approach to save space is to use such a representation for each raster in a time series. However, in this paper we show that it is possible to take advantage of the temporal locality that exists in a raster time series to reduce the space necessary to store it while keeping competitive query times for several types of queries.

Introduction

Geographic Information Systems (GIS) [1] facilitate the management of spatial data in digital format, which allow representing features of a certain surface at a time instant, such as rainfall, temperature, population density, among others. Vector and raster are the two main models used in GIS, each of them tailored for different applications. The latter consists in dividing a surface in a regular set of cells, forming a two-dimensional matrix with numeric values that represent some feature of such surface at a given timestamp (e.g. temperature in Snowbird at 2019-03-26 9 am).

The raster model allows representing large areas of land, thus generating huge spatial data volumes. Two factors affect the space needed to store a raster. One is the spatial resolution, the smaller the cell size of a raster, the higher the precision of the model, but also the larger the space consumption. In several domains, not just a single raster, but a sequence of rasters covering the same region at different timestamps has to be stored and queried. This sequence is known in the literature as a raster time series and it is used, for example, in weather forecast systems and in data mining on *Satellite Image Time Series* [2]. The temporal resolution, defined as the distance in time between two consecutive rasters in the series also impacts the

Funded in part by European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement 690941, by Millennium Institute for Foundational Research on Data (IMFD), by Fondecyt-Conicyt grant number 1170497, and by CYTED 519RT0579. Gilberto Gutiérrez was supported by the research group ALgoritmos y BAses de Datos (ALBA) 160119 GI/EF and by the research project 171319 4/R, both funded by Universidad del Bío-Bío (Chile).

space consumption. For example, going up from a precision of days to a precision of hours requires 24 times more space.

Data compression has been used to reduce the storage and transmission time of raster data [3, 4]. Even the widely used format to represent rasters, GeoTIFF, supports compression techniques based on Lempel-Ziv-Welch [5]. Recently, compact data structures [6] that not only reduce the space, but also provide index capabilities on the rasters, have been proposed [7–9]. Although they do not reduce the space as much as compression techniques, they support several interesting operations without decompression, such as the retrieval of a specific zone or filtering the cells in a zone which values are restricted to a range. These operations are useful in many domains, for example, to zoom-in to a specific area or to detect zones with high flood risk.

A naive approach to store and query raster time series is to use one of the aforementioned compact data structures for each raster in the series. However, in [9] we hypothesized that more compact solutions were possible because of the temporal locality existing in raster time series. Temporal locality produces that the same cell, which represents a spatial region, has similar values in consecutive rasters. For example, the temperature in Snowbird at 14 pm and at 15 pm is often similar. This is even more evident if we increase the temporal resolution, for example, to blocks of 15 minutes. In this paper, we propose a compact data structure that validates the hypothesis in [9]. Our solution is based on the 3D2D mapping described in [9] and a k^3 -tree [10], which is a 3D version of the k^2 -tree [11].

Background and Related Work

We refer to [6] for an overview of compact data structures and to the SDSL library [12] for practical implementations of many of them. Our proposal is based on a 3D version of the k^2 -tree [11], called the k^3 -tree [10]. This data structure has been used in several domains, being the most related to our work the representation of time-evolving region data, a.k.a. moving regions. The k^2 -tree is a compact representation of a quadtree [13] that stores a sparse binary matrix in compact space by subdividing it into k^2 submatrices, which are processed in Morton order storing a 1-bit for non-empty submatrices and a 0-bit, otherwise. This procedure is recursive, but just on non-empty submatrices and up to a minimum submatrix size. This partitioning strategy is generalized by the k^3 -tree, for 3D, and by the k^n -tree, in general. It is important to notice that each 1-bit in the compact representation produces a new (implicit) node with k^n -bits, hence degrading compactness with the dimensionality. In a nutshell, both the k^2 -tree and the k^3 -tree use few space when the ones in the binary grid are clustered, but non-clustered ones have a more negative impact in the k^3 -tree than in the k^2 -tree. This is one of the reasons why there are more examples of success for the k^2 -tree than for its 3D version.

Our other main building block is one of the compact data structures for rasters presented in [9], specifically the 3D2D-mapping. This mapping is based on space-filling curves, which are mathematical functions providing a mapping from a multidimensional space to one dimension [14]. The 3D2D-mapping uses the Morton or Z-order [15] curve, which preserves spatial locality and it is suitable for efficient com-

putations based on bit interleaving. Next, we briefly describe this data structure.

Let $\langle x, y, z \rangle$ be a 3D tuple, with (x, y) a cell in the raster and z its value. These tuples are mapped into a 2D binary grid, in which one axis represents Morton order and the other represents values. Hence, a tuple $\langle x, y, z \rangle$ induces a 1-bit at row $Z(x, y)$ and column z , where $Z(x, y)$ represents the Morton code of cell (x, y) . Then, this binary grid is stored using a k^2 -tree, which exploits the clustering of the 1-bits that is produced by the locality of the Morton order. Three types of queries, *access*, *windowQuery* and *rangeQuery*, are supported by this data structure by reduction to primitives of the k^2 -tree, mainly range queries. However, some of them also require a decomposition of a query into maximal quadboxes [16, 17].

Besides the 3D2D-mapping there are some other compact representations of rasters. For example, in the same paper [9], another structure based on space-filling curves and succinct trees is proposed, which usually achieves slightly better performance than the 3D2D-mapping, however, its generalization to raster time series is not easy. Previously, in [7] several variants of the k^2 -tree [11] were also proposed and proved successful to outperform classical approaches such as GeoTIFF. A more recent work [8] proposed the k^2 -raster, which is based on an augmented k^2 -tree that scales better with the number of different values in the raster. Earlier this year [18], a generalization of such structure to temporal rasters was also proposed.

Our solution

In this section we describe our compact representation of raster time series with support for the following operations:

- *access*(x, y, t) retrieves the value of the cell (x, y) at t -th time instant. E.g. temperature in Snowbird¹ at 2019-03-26 9 am.
- *windowQuery*($x_1, y_1, x_2, y_2, t_1, t_2$) retrieves all the values (a 3D submatrix) registered in a rectangular zone defined by its corners (x_1, y_1) and (x_2, y_2) during a time interval $[t_1, t_2]$. E.g. temperatures in Utah from march to may.
- *rangeQuery*($x_1, y_1, x_2, y_2, t_1, t_2, rMin, rMax$) retrieves all the cells contained in a rectangular zone defined by (x_1, y_1) and (x_2, y_2) during a time interval $[t_1, t_2]$ which values are in the range $[rMin, rMax]$. E.g. zones (cells) in Utah from march to may with *moderate rain*, i.e. greater than 0.5 mm per hour, but less than 4.0 mm per hour.
- *valuesRangeQuery*($rMin, rMax$) retrieves all the cells which values are in the range $[rMin, rMax]$. Although this is a particular case of the previous type, we study it independently given its application in several domains, for example, to detect zones with high flood risk.

As we mentioned above, our solution is based on the 3D2D-mapping proposed in [9]. Let $TS = [R_1, R_2, \dots, R_n]$ be a raster time series composed of n rasters. We

¹In our examples, we use a toponym for easy readability, but queries actually receive geographic coordinates as parameters, either to define a point or a window.

first read each raster R_i in Morton order, which produces a sequence T_i . Then, we transform each T_i to a binary grid B_i that contains a 1-bit at cell $B_i[x][y]$ if $T_i[x] = y$ and a 0-bit, otherwise. Finally, we concatenate all these binary grids into a 3D cube $\{B_1, B_2, \dots, B_n\}$ and store it using a k^3 -tree. Figure 1 shows an example of the mappings used by our method.

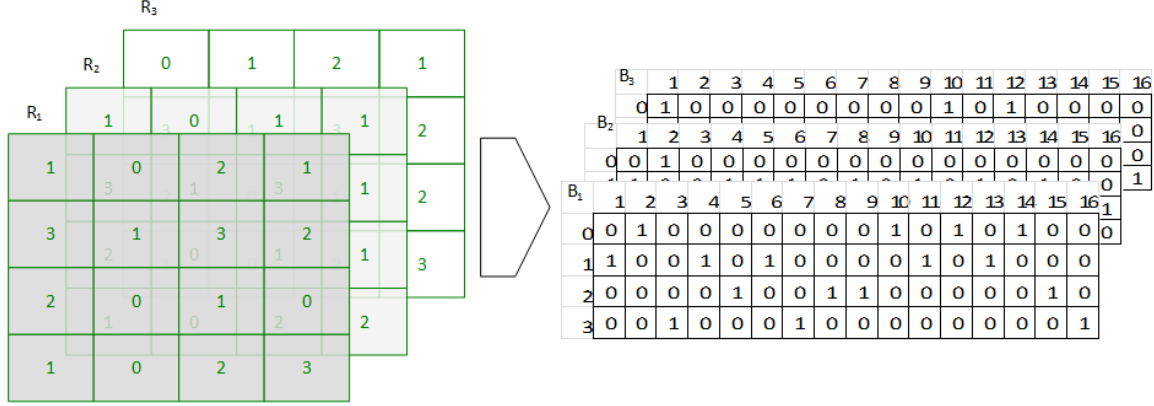


Figure 1: Mappings to construct the data structure

In the example, let us focus on the first raster R_1 , which is transformed into the binary grid B_1 . The first four values of R_1 in Morton order are 1, 0, 3, and 1, which induce a 1-bit at row 1 of the first column, at row 0 of the second column, at row 3 of the third column, and at row 1 of the fourth column, respectively. All the other bits in these four columns are 0-bits. Similarly, the first value in R_2 is also a 1, which induces a 1-bit at $B_2[1][1]$.

There are two important characteristics in the domain for the k^3 -tree to perform well, which are the spatial and the temporal locality. Spatial locality is due to Tobler's first law of geography [19] "*everything is related to everything else, but near things are more related than distant things*", and to the use of space-filling curves in the mapping, which preserve such property. In other words, Tobler's law implies that cells that are close in a raster of the time series should contain similar values. In our transformation to a binary grid, we use Morton order, and thus cells that are close in the original raster are also close in one of the axis of the grid (w.l.o.g. let us say x-axis). For each of such cells, we set a 1-bit in the row corresponding to its value. As their values are similar, these 1-bits will be close in the binary grid. Hence, the 1-bits in each binary grid are clustered.

Temporal locality is not related to a specific raster, but to consecutive rasters. This property states that if $R_i[x][y] = v$ and $R_{i+1}[x][y] = v'$, then v and v' should be similar. Given our mapping, both B_i and B_{i+1} will have a 1-bit set in column $Z(x, y)$, and row v and v' , respectively. Hence, the 1-bits are not only clustered in each binary grid, but also through consecutive binary grids. The temporal resolution of the raster time series has a great impact on this property, in most domains. This can be easily explained with the following example. Consider two temporal resolutions,

one of 1-hour blocks and another of 6-hours blocks, and the domain of temperatures. Obviously, the temperature in Snowbird at 9 am is similar to the temperature at 10 am, but not necessarily similar to the temperature at 3 pm. In the next section, we illustrate this influence with some experiments.

To complete the description of our proposal, we explain the query algorithms to solve the four types of queries explained above. The k^3 -tree supports operations *getCell*, which checks the value of a cell, and *generalRange*, which performs a 3-dimensional range query, as explained in [10, Section 6.2.1]. Then, *access* can be easily implemented using *getCell*, and all the other queries using *generalRange*. An important consideration is about the spatial dimension of the queries which, due to the use of Morton order, can not be directly transformed to a contiguous range in one dimension of the k^3 -tree. However, a maximal quadbox decomposition [16, 17] can be used to convert a general spatial query, to a set of maximal quadboxes, and each of them becomes a contiguous range in the k^3 -tree, hence allowing the use of *generalRange*. This is the same solution used in [9].

Experiments

In this section, we present our preliminary experiments that show the viability of the proposal. We compare the solution described above, with a baseline based on the use of the 3D2D-mapping for each raster of the time series using a k^2 -tree. This baseline is a representative of the compact data structures existing for rasters [7–9], which exploit the spatial locality but not the temporal one. All the experiments presented here were performed in an Intel Core i7-3820@3.60GHz, 32GB RAM, running Ubuntu server (kernel 3.13.0-35). We compiled with gnu/g++ version 4.6.3 using -O3 directive as both the baseline and the proposed data structure were implemented in C++.

Regarding the datasets, we created several real datasets of temperatures from the data provided by the National Weather Service [20]. Among other products, this service provides hourly information of temperatures for Guam, Hawaii and Puerto Rico. As these data are just available for 48 hours, we downloaded them for two months from September to October 2017. Also, we cropped some parts of these datasets to experiment with datasets of different sizes. Finally, to evaluate the precision of the temporal resolution, we averaged the daily temperatures of the Guam dataset. Table 1 summarizes the characteristics of all the datasets we collected. Also, to illustrate how data vary with time, we prepared a video for two days of the Guam dataset (<http://www.inf.udec.cl/~dseco/guam.mp4>).

We first present a space comparison of our proposal and the baseline, which is summarized in Table 2. The first column shows the size of the dataset. We show both total space in MB, including the size of the original dataset, and the bits per cell required by both our proposal and the baseline.

The main conclusion of this evaluation is that our proposal usually requires less than half the space of the baseline, thus validating the hypothesis that it is possible to exploit the temporal locality in raster time series to design more space-efficient data structures for them. A second conclusion is the influence of the temporal resolution in our proposal, which can be observed in the last two rows. The original Guam dataset

Dataset	Columns	Rows	#Rasters	Resolution
Guam-64	64	64	456	3-hours
Hawaii-64	64	64	1,368	Hourly
Puerto-Rico-64	64	64	1,368	Hourly
Guam-128	128	128	456	3-hours
Hawaii-128	128	128	1,368	Hourly
Puerto-Rico-128	128	128	1,368	Hourly
Guam-193	193	193	456	3-hours
Guam-daily-193	193	193	62	Daily

Table 1: Characteristics of the datasets

Dataset	Total space (MB)			Bits per cell	
	plain	baseline	proposal	baseline	proposal
Guam-64	5.44	0.65	0.17	2.9	0.8
Hawaii-64	16.33	2.90	1.12	4.3	1.7
Puerto-Rico-64	16.34	4.70	2.20	7.0	3.3
Guam-128	21.50	1.33	0.65	1.5	0.7
Hawaii-128	64.50	7.24	4.01	2.7	1.5
Puerto-Rico-128	64.52	8.29	4.82	3.1	1.8
Guam-193	48.77	2.46	2.09	1.6	1.4
Guam-daily-193	5.37	0.75	0.81	2.7	2.9

Table 2: Space evaluation

has a temporal resolution of blocks of 3 hours, unlike the others that provide an hourly resolution. Despite this lower resolution, our proposal still slightly outperforms the baseline in this dataset. However, in the last dataset, we averaged the temperatures of each day, generating a dataset with daily resolution. As it can be seen in the last row, our proposal requires slightly more space than the baseline in this case, which is due to the lower temporal locality.

We now present an evaluation of the query time performance for the queries introduced above. First, Figure 2 shows the results for *access*. We show average time per query for a set of 10,000 random queries.

Results are consistent through the different datasets and show that the proposal is about 4 times slower than the baseline, but still in the order of few microseconds per query. This is not a surprising result as the underlying data structure of the proposal is more complex than that of the baseline, and this type of query just involves one raster in the baseline.

For the other types of queries, we just show the results for one of the datasets, as

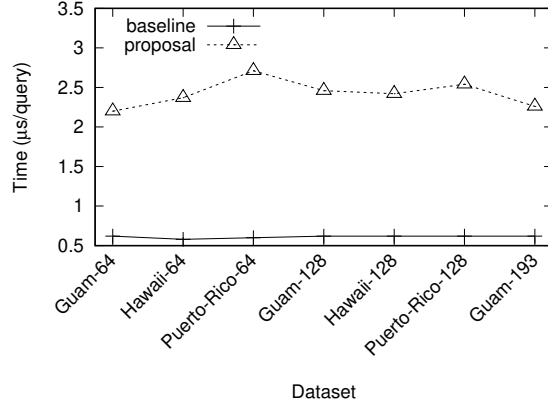


Figure 2: Query time for *access*

results are similar in all of them and, in this way, we can focus on the influence of the parameters of such query types. We selected Hawaii-128 as a representative dataset. Recall that this dataset has a spatial extent of 128×128 with 1,368 snapshots and an hourly temporal resolution. Figure 3 illustrates the results of *windowQuery* for different window sizes. These results show that the proposal is also competitive for this type of queries as the differences between the proposal and the baseline are not significant, being our proposal slightly faster than the baseline for most window sizes (note the logarithmic scale).

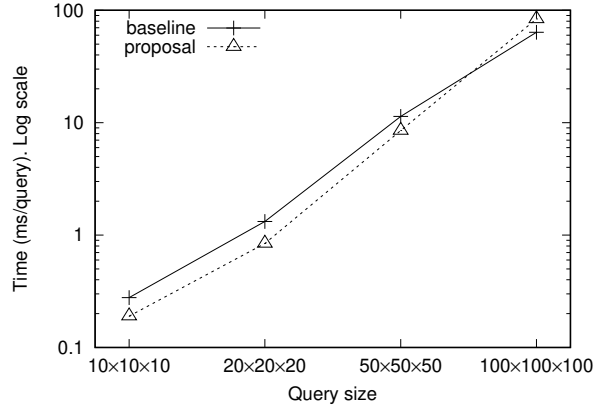


Figure 3: Query time for *windowQuery*

For the most general type of queries, *rangeQuery*, we show the influence of both the query size, given a fixed range size of 10, and the influence of the range size, given a fixed size of query of $10 \times 10 \times 10$. Figures 4 and 5 show these results.

Similar to previous experiments, the proposal is slightly faster than the baseline. Another conclusion is that the range size has less influence in the query time than the

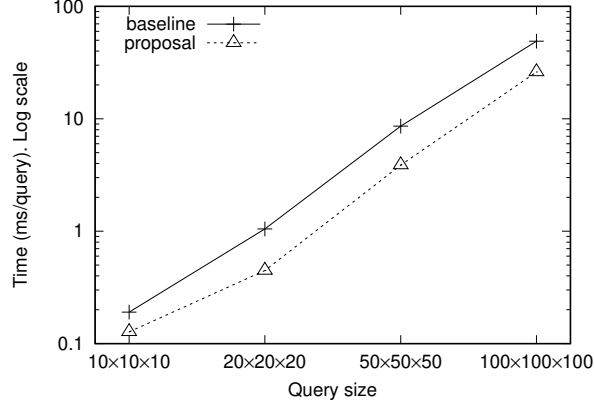


Figure 4: Query time for *rangeQuery*, given a fixed range size of 10

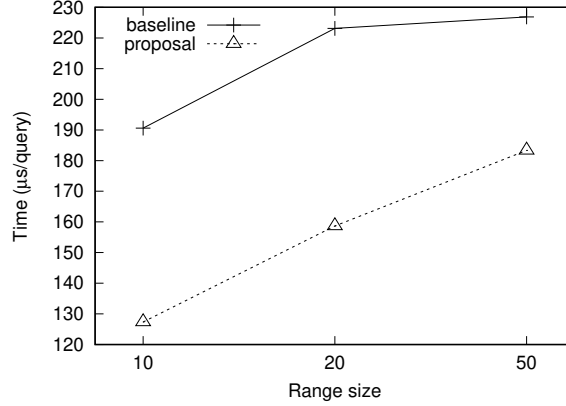


Figure 5: Query time for *rangeQuery*, given a fixed window size of $10 \times 10 \times 10$

window size (note the logarithmic scale in Figure 4). This is not only because range size represents just one of the dimensions of the k^3 -tree, but also because it does not require the maximal quadbox decomposition needed for the spatial dimension.

Conclusions

We proposed a new space-efficient data structure that takes advantage of the spatial and temporal locality existing in a raster time series. Our experimental evaluation, performed using a real dataset of temperatures that has both spatial and temporal locality, shows that our structure requires half the space of a baseline that uses a compact representation for each raster in the time series but does not take advantage of the temporal locality in the time series. However, we also showed that our proposal is dependant on the temporal resolution and, in the case of temperatures, it requires similar space than the baseline for daily averaged datasets. Regarding

query times, our structure is competitive, and even slightly faster, than the baseline for several types of queries. As future work, a more detailed experimental evaluation is necessary to compare with all the compact data structures for rasters existing in the literature [7–9]. Also, our work has to be evaluated against a recent proposal for temporal rasters presented in [18]. Finally, to deal with the curse of dimensionality of the k^3 -tree a 3D version of the heavy path tree in [21] may improve the space usage.

References

- [1] M. Worboys and M. Duckham, *GIS: A Computing Perspective*. CRC Press, Inc., 2004.
- [2] F. Petitjean, P. Gançarski, F. Masegla, and G. Forestier, *Analysing Satellite Image Time Series by Means of Pattern Mining*. Springer Berlin Heidelberg, 2010, pp. 45–52.
- [3] W. Kou, *Digital Image Compression: Algorithms and Standards*. Kluwer Pub., 1995.
- [4] G. K. Wallace, “The jpeg still picture compression standard,” *Commun. ACM*, vol. 34, no. 4, pp. 30–44, 1991.
- [5] D. Salomon, *Data Compression: The Complete Reference*. Springer, 2006.
- [6] G. Navarro, *Compact Data Structures: A Practical Approach*. Cambridge University Press., 2016.
- [7] G. de Bernardo, S. Álvarez-García, N. R. Brisaboa, G. Navarro, and O. Pedreira, “Compact queriable representations of raster data,” in *Proc. 20th SPIRE*, 2013, pp. 96–108.
- [8] S. Ladra, J. R. Paramá, and F. Silva-Coira, “Scalable and queryable compressed storage structure for raster data,” *Information Systems*, vol. 72, pp. 179–204, 2017.
- [9] A. Pinto, D. Seco, and G. Gutiérrez, “Improved queryable representations of rasters,” in *Proc. DCC*, 2017, pp. 320–329.
- [10] G. de Bernardo, “New data structures and algorithms for the efficient management of large spatial datasets,” Ph.D. dissertation, Universidade da Coruña, 2014.
- [11] N. R. Brisaboa, S. Ladra, and G. Navarro, “Compact representation of web graphs with extended functionality,” *Information Systems*, pp. 152–174, 2014.
- [12] S. Gog, T. Beller, A. Moffat, and M. Petri, “From theory to practice: Plug and play with succinct data structures,” in *Proc. 13th SEA*, 2014, pp. 326–337.
- [13] H. Samet, *Foundations of multidimensional and metric data structures*. Morgan Kaufmann, 2006.
- [14] H. Sagan, *Space-Filling Curves*. Springer, 1994.
- [15] G. M. Morton, “A computer oriented geodetic data base and a new technique in file sequencing,” Tech. Rep., 1966.
- [16] G. Proietti, “An optimal algorithm for decomposing a window into maximal quadtree blocks,” *Acta Informatica*, vol. 36, no. 4, pp. 257–266, 1999.
- [17] Y.-H. Tsai, K.-L. Chung, and W.-Y. Chen, “A strip-splitting-based optimal algorithm for decomposing a query window into maximal quadtree blocks,” *IEEE Trans. on Knowl. and Data Eng.*, vol. 16, no. 4, pp. 519–523, Apr. 2004.
- [18] A. Cerdeira-Pena, G. de Bernardo, A. Fariña, J. R. Paramá, and F. Silva-Coira, “Towards a compact representation of temporal rasters,” in *Proc. 25th SPIRE*, 2018, pp. 117–130.
- [19] W. R. Tobler, “A computer model simulation of urban growth in the detroit region,” 1970, *economic Geography*. p.236.
- [20] N. C. Operations. (2017) Real-time mesoscale analysis (RTMA) products.
- [21] T. Gagie, J. I. González-Nova, S. Ladra, G. Navarro, and D. Seco, “Faster compressed quadtrees,” in *Proc. DCC*, 2015, pp. 93–102.