16th WSEAS Int.Conf. on Computers ICCOMP12, Kos, Greece, pp.143-148, 2012

A Unified Approach for Textual and Geometrical Information Retrieval

VACLAV SKALA
Department of Computer Science and Engineering
University of West Bohemia, Faculty of Applied Sciences
Univerzitni 8, CZ 306 14 Plzen
CZECH REPUBLIC
skala@kiv.zcu.cz http://www.VaclavSkala.eu

Abstract: Textual and geometrical algorithms have been considered as two separate fields. This was caused by the fact that textual data are discrete in principal and interpolation is not defined as there is no metric in general, while geometrical data are considered discrete samples of continuous phenomena, geometrical surface etc.

In this paper we present a unified approach to textual and geometrical data is hashing technique is used. We summarize principles, knowledge gain and experimental results obtained recently. Hashing techniques enable to answer a query with *O(1)* complexity that is very important if large textual or geometrical data are to be processed. This approach has lead to a new construction of hash functions for textual and geometrical data as well.

Key-Words: Hash function, textual data, geometrical data, triangular mesh reconstruction, data retrieval

1 Introduction

Fast information retrieval is a crucial aspect of many applications as today's data sets are becoming larger and larger. There is a lot of different data structures [4], [6], [7], like binary tree, Kd-tree, index-sequential etc. However the only one, hashing, offers an answer to a query with *O(1)* complexity. It means that the time needed for any item retrieval is constant if the hashing function is "perfect" one [2], i.e. it gives a different address for a different value. This is not generally true and computational complexity and response time heavily depend on properties of the hash function used.

Techniques based on hashing are heavily used in many applications, e.g. information retrieval [1], [18], geometry processing [9], [19], chemical [20] and medical applications etc. and even in cryptography. Typical applications in the textual are well known, in the case of geometrical data the hashing can be used for a fast elimination of duplicated points, which is of *O(N)* complexity instead of *O(N lgN)* or $O(N^2)$, for triangular mesh reconstruction from the given set of triangles etc.

Hash functions are considered in the form of $index = f(v)\ \mathbf{mod}\ m$ where *m* is a prime number and *f(v)* is a function over an element *v*, which is generally of „unlimited" dimensionality and/or of „unlimited" range of values.

In the text processing case, the dimensionality is "unlimited", i.e. a string can be very very long, e.g. the titin protein is described by the word *Methionyl-threonylthreonylglutaminyla.....isoleucine* which consists of 189,819 characters [8], [20], but the range of values for each dimension is limited to the size of the actual alphabet.

In the case of geometric data processing we need to process $10^6 - 10^{10}$ of points consisting usually of three coordinates, i.e. $< x, y, z >$ coordinates, or more. The situation is even more complicated by the fact that the range of values for each coordinate is generally "unlimited", i.e. the interval $(-\infty, \infty)$ has to be considered.

It can be seen that there are significantly different requirements from those two application fields to the hash function construction and hashing methods in general. The basic data classification is presented by the Tab.1.

		Dimensionality	
		Limited	Unlimited
Interval	Limited	All trivial data sets	Textual data
	Unlimited	Geometrical data	Special data sets

Table 1: Data classification

However the situation is even more complex, see "Special data" in the Tab.1. There are a lot of applications where we need to represent a continuous signal and find if the signal is already

stored. Such signals are usually represented by infinite series, e.g. DCT, FT etc.

$$X_k = \sum_{n=0}^{N-1} x_i \cos\frac{(\pi(2n+1)k}{2N} \quad k = 0, \dots, N-1$$

It can be seen that the sampled continuous signal is represented by X_k values. It means that we have "unlimited" interval of values and "unlimited" dimensionality as *N* is high in general.

In the following we explain how the hash function can be constructed in general and how the hash function is constructed for textual and geometric data.

2 Hash Function

Hashing technique is well known and several modifications have been designed and used. Hashing is considered as a fundamental data structure, now.

The hashing principle is presented in Fig.1.

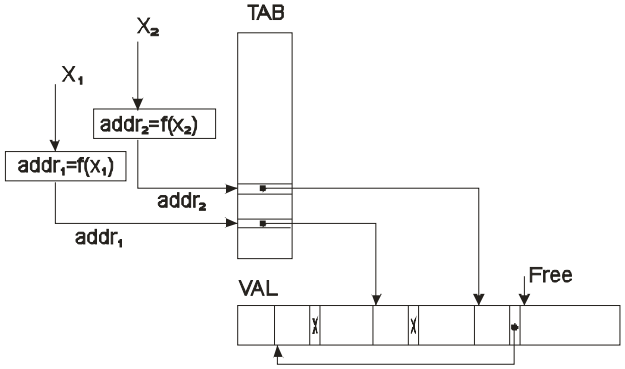

Figure 1: Hash function principle

Data structures based on hashing have the following properties:

- Storing and retrieval complexities *O(1)*
- Hash table length is more than twice of number of items stored
- No data spatial coherency in a memory
- If less memory is available, the already build data structure has to be totally recomputed.

It is necessary to note that standard hashing techniques require modulo operation with a prime that is very expensive.

The modulo operation is defined as:

$$a \; \boldsymbol{mod} \; p = a - p * \left\lfloor \frac{a}{p} \right\rfloor$$

where $\lfloor . \rfloor$ is defined as a "floor" function. It means that each access to the data structure, i.e. in storing or retrieval requires one division and multiplication at least, if division is made in the integer representation. However, today's processors are optimized for operations with a floating point representation and floating point operations are faster.

As we are expected to process data with "unlimited" dimensionality or values, the integer representation cannot be used for a representation of the value a, if uncontrolled overflow is to be avoided. It means that the value a has to be represented in the floating point representation and many conversion operations float to integer or vice versa have to be used.

3 New Hash Function Construction

The above mentioned arguments and different requirements for the textual and geometrical data led to new specifications for hashing.

The following requirements were taken into account:

- Floating point representation is to be used
- Division operation should not be used
- Conversions float to integer and vice versa are to be limited as much as possible
- Hash table should be easily recomputed, if less memory is available.

These requirements lead to understanding that the hash table TAB, see Fig.1, is to be virtually made as long as possible with the length 2^k, i.e. the *addr* address will be in the interval $\{0,\dots,2^k-1\}$, usually the full range of available integer values.

The operation ***mod*** is replaced by masking, i.e. by bitwise ***and*** operation. This will map the "virtual" address to the address for the physically allocated hash table of the length 2^N, i.e. mask will be given by the value $2^N - 1$. As a consequence, if the physical length of the hash table is ½ of the original, the new hash table is easily made by "repagination" of the original hash table.

However there are some important points to be mentioned.

Geometrical case

In the case of geometrical data the "unlimited" interval of values is to be converted to a limited interval <-*1,1*> using a formula:

$$x' = \frac{1}{2}\left(\frac{x}{|x| + k} + 1\right)$$

where *k* is a parameter $0 < k < \infty$ and $x \in < 0,1 >$. Similarly for *y* and *z* coordinates. If the <*min,max*> interval is known, the conversion is simple.

The argument for the hash function is computed as

$$val = Q(\alpha x + \beta y + \gamma z)$$

where: α, β, γ are coefficients, $Q = 1/(\alpha + \beta + \gamma)$ in order to have $val \in < 0,1 >$.

Originally the ***mod*** operation with a prime was used to get a better spread of values. In our approach we do not use the ***mod*** operation with a prime, but we use bitwise **and** operation with a mask of 2^N*-1* value. For better spread "irrational" values for the α, β, γ coefficients have to be selected, like $\frac{1}{3}, \pi, e, \sqrt{2}, \sqrt{3}$ etc.

Textual case

In the case of textual data we know that the dimensionality is "unlimited" while values are limited by the given alphabet. In this case the argument for the hash function is computed as

$$val = Q \sum_{i-1}^{L} q^i x_i$$

where: L is the length of the given string if known, x_i is the i-th character of the given string, $0 < q < 1$ so the sum above is convergent. The value Q is a multiplicative factor so that $0 \leq val \leq 1$.

Again, as we do not use **mod** operation with a prime but bitwise **and** operation with a mask of 2^N*-1* value we need "irrational" values for the q for better spread. By the term "irrational" we mean that the best is a value which cannot be represented by finite number of bits and $0 < q < 1$.

Both cases

It can be seen that the value $val \in < 0,1 >$ in the both cases. Now, the value val has to be multiplied by a constant C so that $\lfloor C * val \rfloor \in < 0, 2^k - 1 >$, where 2^k is a less or equal to a maximum value given by the number of bits for integer representation, e.g. 2^{23} if single precision used, 2^{52} if double precision used or 2^{112} if quadruple precision used (number of bits used depends on the number of mantissa bits in the floating point representation used). Then the address to the virtual table address space is given as:

$$VirualAddress = \lfloor C * val \rfloor$$

the *addr* address to the physical hash table space:

$$addr = VirtualAddress \; \mathbf{and} \; (2^N - 1)$$

The above described approach has to be optimized in coding for the speed as the principles are described only.

4 Experimental Results

The hash function can be used e.g. for duplicates elimination in the both cases, i.e. in the geometrical and textual data. This seems to be a simple task, but generally it is of the $O(N^2)$ complexity, if a brute force approach is used, or *O(N lgN)* complexity, if some ordering is applied as preprocessing, e.g. sorting etc. The hash function offers *O(1)* complexity for storing and retrieval, if the hash function is **properly** designed.

The application of a hash function for geometrical data was originally used by Glassner [3] for triangular mesh reconstruction and the "standard" hash function was used, i.e. with ***mod*** operation with primes and Fig.2 - Fig.4 present selected examples of data sets used in experiments.

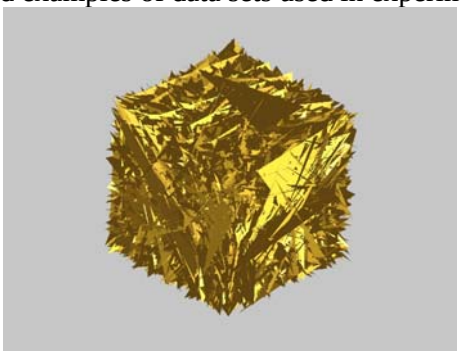

Figure 2: Gener - synthetically generated data

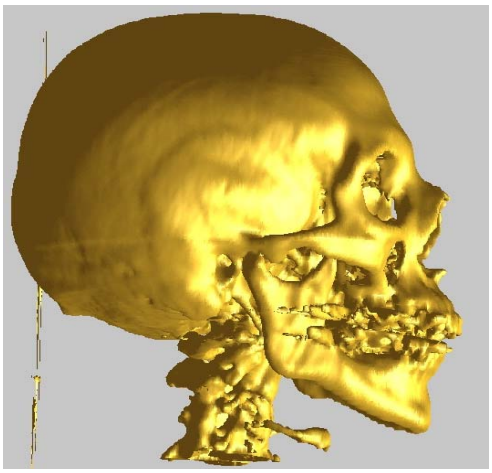

Figure 3: CT Head – iso-surface generation

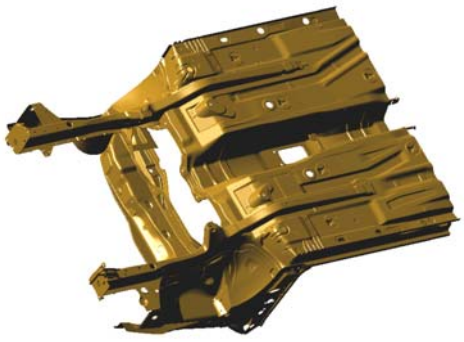

Figure 4: Car chassis data set

For effective data retrieval and storing it is necessary to have cluster length for all points stored close to one.

Geometrical case

Distribution of the cluster lengths for the synthetically generated data and for the Car chassis data, if standard hash function with ***mod*** operation with a prime is used, is shown in Fig.5 and Fig.6.

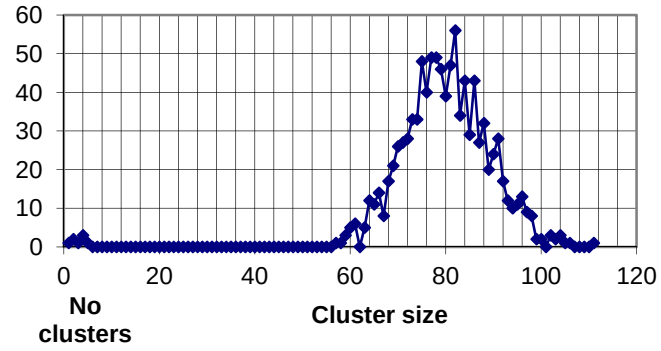

Figure 5: Cluster lengths for the Gener data set

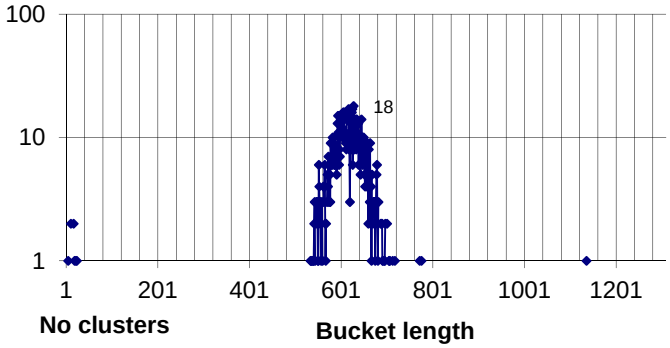

Figure 6: Cluster lengths for the Car chassis data set

It can be seen that for the Car chassis data set, the average cluster length is approx. 600, i.e. there is approx. 300 tests to answer whether the item is already stored or not. Also the distribution of cluster lengths depends significantly on data sets.

If the proposed hash function construction is used, significantly better results are obtained.

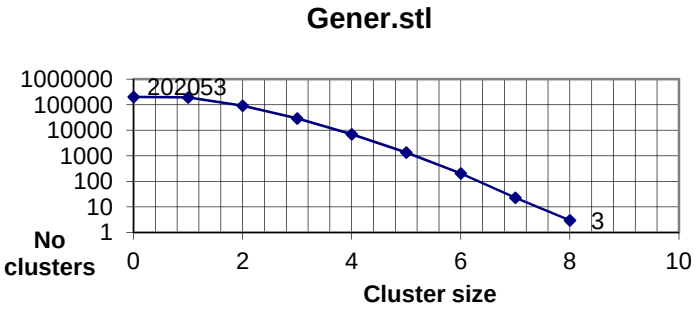

Figure 7: Cluster lengths for the Gener data set

Typical experimental results are presented at Fig.7.- Fig.9., if the proposed hash function construction is used. It can be seen that in all the cases the average cluster length is significantly smaller, cluster lengths decrease monotonically and behavior is stable. Detailed information can be found in [5], [13], [15], [16], [17].

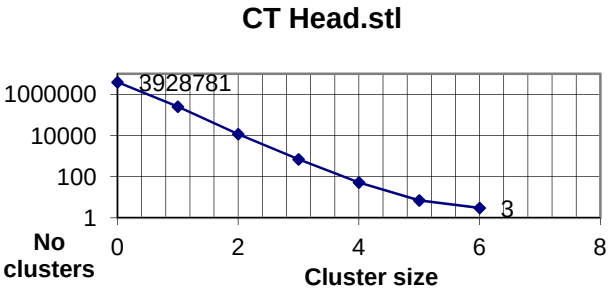

Figure 8: Cluster lengths for the CT Head data set

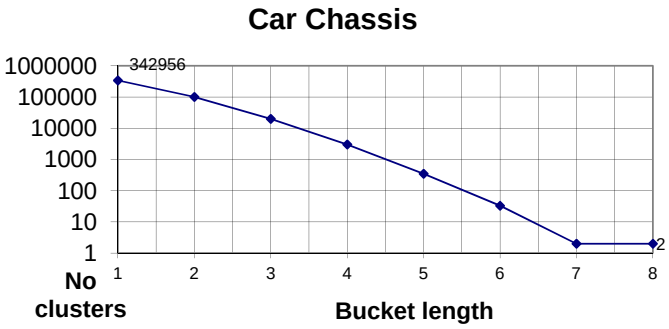

Figure 9: Cluster lengths for the Car chassis data set

Textual case

The principle of the proposed hash function for the textual case is the same and modification of the hash function construction was described above.

The experiments were based on duplicates of words elimination with different languages using Ispell dictionary [21]. For the hash function property evaluation the following criterion was derived:

$$Q' = \frac{3}{2N}\sum_{I=1}^{I_m} I^2 C_I$$

where: N is a number of items stored, I_m is a maximum cluster length, I number of items in the cluster, C_I number of clusters of the length I, see [11], [12] for details.

The experiments were made with the English, Czech, French, German, Hebrew, Russian and other dictionaries. It is necessary to note that for the Slavonic languages, i.e. Czech, Russian etc., slight modification improves the results even more (if string is processed from the end of the string).

The data sets for English and Czech languages were taken from the Ispell package. The Czech dictionary contains approx. $2.5\ 10^6$ words and the English dictionary contains approx. $1.3\ 10^5$ words.

Fig.10 and Fig.11 present how the relative criterion Q' depends on the parameter q – only "irrational" values of q were tested. If value q would be something like ½ or ¼ cluster lengths are

extremely high. It can be seen that the behavior of the proposed hash function is stable.

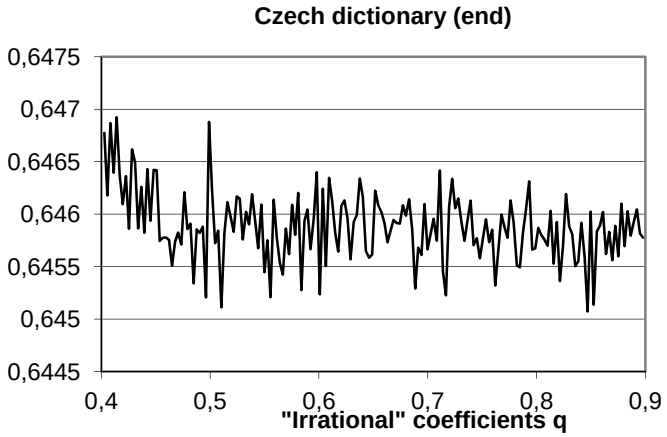

Figure 10: Relative criterion *Q*' for the Czech dictionary

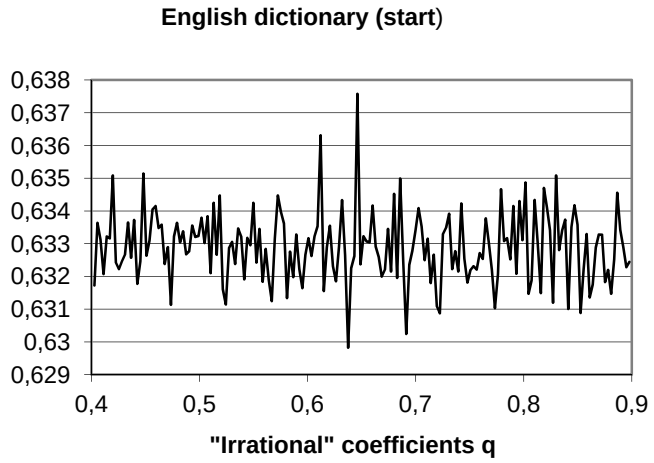

Figure 11: Relative criterion *Q*' for the English dictionary

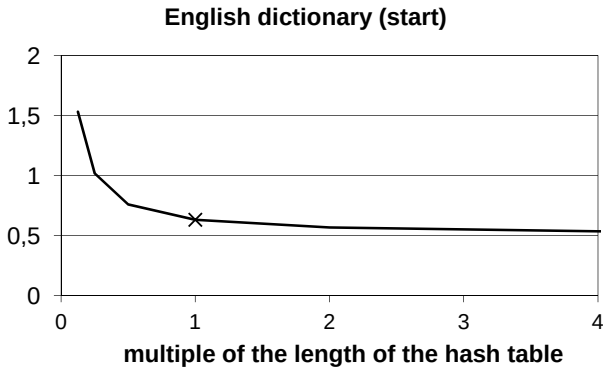

Figure 12: Relative criterion *Q*' for the English dictionary for different hash table sizes

If the table length is extended or shorten, the hash function proposed behaves as expected, see Fig.12 (value 1 means that the table length is approx. 2*number of items stored).

6 Conclusion

A new classification of data for hashing is presented. The hashing technique enables not only fast processing of textual data but also fast processing of geometrical data. The newly presented classification enables also to store and retrieve representation of sampled continuous signals using DCT or FT transformations or harmonic analysis approach. This group of data was presented in the classification in Tab.1 as "Special data".

Due to the unified design of the hash function the properties of hash functions, the principles of the hash function for textual data and geometrical data can be combined also for processing of the "Special data" group.

Experiments made proved that hash functions for both textual and geometrical data are stable with very short clusters (buckets). The proposed method eliminates the ***mod*** operation with a prime number. Computations are made in the floating point representation that offers additional speed up due to current hardware architecture. Experiments also showed that combinations of "irrational" coefficients can slightly improve the efficiency and this can be further explored.

We summarized principles, knowledge gain and experimental results obtained recently and detailed description can be found in [11]-[17].

Future work will be concentrated on modifications of the hash function for the "Special data" case.

7 Acknowledgments

The author would like to express his thanks to students at the University of West Bohemia, especially to Martin Kuchar, Jan Hradek, Martin Sinko, Ondrej Neved and Peter Citriak for additional implementations and verifications of the proposed hash function constructions. Thanks also belong to colleagues at the University of West Bohemia, to Dr.Rongjian Pan from the Shandong University, China and anonymous reviewers for recommendations and constructive hints that helped to finish this work and improve the manuscript.

This work was supported by the Ministry of Education of the Czech Republic – Projects No. ME10060 and LH12181.

References:

[1] D'hondt,J., Verhaegen,P.A., Vertommen,J., et al.: Near-Duplicate Detection Based on Text Coherence Quantification, *10th European conference on Knowledge Management*, Vols.1&2, pp. 238-246, 2009

[2] Gettys,T.: Generating perfect hash function, *Dr. Dobb's Journal*, Vol. 26. No.2, pp.151-155, 2001.

[3] Glassner,A.: Building Vertex Normals from an Unstructured Polygon List, *Graphics Gems IV*,. Academic Press, Inc., Cambridge, pp.60 – 73, 1994.

[4] Horowitz,E., Sahni,S.: *Fundamentals of Data Structures*, Pitman Publ.Inc., 1976.

[5] Hradek,J., Skala,V.: Hash Function and Triangular Mesh Reconstruction, *Computers and Geosciences*, Elsevier, Vol.29, No.6., pp.741-751, 2003

[6] Knuth,D.,E.: *The Art of Computer Programming*, vol. 3, Searching and sorting, Addison-Wesley, 1990.

[7] Korfhage,R.,R., Gibbs,N.E.: *Principles of Data Structures and Algorithms with Pascal*, Wm.C.Brown Publishers, 1978.

[8] Mughal,M.S., Nawaz,M., Ahmad,F., Shahzad,S., Bhatti,A.K., Mohsin,S: 3D Hash Function for Fast Image Indexing and Retrieval, Computer Graphics, *Imaging and Visualization 2007*, IEEE 0-7695-2928-3, 2007.

[9] Qu,X., Stucker,B.: A 3D surface offset method for STL-format models, *Rapid Prototyping Journal*, Vol.9, No.3., pp.133-141, ISSN 1355-2546, 2003.

[10] Skala,V., Hradek,J., Kuchar,M.: New Hash Fuction Construction for Textual and Geometrical Data Retrieval, *Latest Trends on Computers*, ISBN 978-960-474-213-4, ISSN 1792-4251, CSCC conference, Corfu, Greece, Vol.2, pp.483-489, 2010.

[11] Skala,V., Hradek,J.: Efficient Hash Function for Duplicate Elimination in Dictionaries, *Algoritmy 2009 proceedings*, Slovak University of Technology, Bratislava, pp.382-391, ISBN 978-80-227-3032-7, 2009.

[12] Skala,V., Hradek,J., Kuchar,M.: New Hash Fuction Construction for Textual and Geometrical Data Retrieval, *Applied Mathematics, Simulation and Modeling - ASM 2010 proceedings*, NAUN, Corfu, Greece, ISSN 1792-4332, ISBN 978-960-474-210-3, pp.209-219, 2010.

[13] Skala,V., Hradek,J., Kuchar,M.: Hash Function for Triangular Mesh Reconstruction, *13th International Conference on COMPUTERS, WSEAS*, , ISBN: 978-960-474-099-4, pp. 233-238, 2009.

[14] Skala,V.: Hash Function Use for Triangular Mesh Reconstruction, *2nd International Conference on Image and Graphics, SPIE*, Vol.4875, pp.39-46, ISSN 0277-786X/02, 2002.

[15] Skala,V., Kuchar,M., Hradek,J.: Geometry Reconstruction in Rapid Prototyping with Hash Function, *International Conference on Computer Graphics and Imaging CGIM2001*, Honolulu, USA, ISBN 0-88986-303-2, pp.240-245, 2001

[16] Skala,V., Kuchar,M.: The Hash Function and the Principle of Duality, *Computer Graphics International 2001*, Honk Kong, China, IEEE Proceedings, ISSN 1530-1052, pp.167-174, 2001.

[17] Skala,V., Kuchar,M.: Hash Function for Geometry Reconstruction in Rapid Prototyping, *Algoritmy'2000 proceedings*, Slovakia, ISBN 80-227-1391-0, pp.379-387, 2000

[18] Stein,B.: Principles of Hash-based Text Retrieval, *ACM SIGIR 07*, pp. 527-534, 2007.

[19] Yazdi,M.S., Najafzade,K., Moghaddam,M.E.: A Fast Symbolic Image Indexing and Retrieval Method Based on TSR and Linear Hashing, *Signal Processing and Information Technology*, pp.469 - 473, 2008.

[20] Wipke,W.T., Krishnan,S., Ouchi,G.I.: Hash Function for Rapid Storage and Retrieval of Chemical Structures, *J.Chem.Inf.Compu.Sci*., Vol.18., No.1, pp.32-37, 1978.

WEB references

[21] Dictionaries for International Ispell, *http://ficus-www.cs.ucla.edu/geoff/ispell-dictionaries.html*
<Retrieved on 2012-03-24>