

Gábor Misi

An algebraic representation of Labanotation for retrieval and other operations

Introduction

Computer applications for Kinetography Laban/Labanotation are mostly simple graphic editors and generally do not contain search facilities. The first experimental software for retrieval was DanceStruct (Fügedi, 1995), developed at the Institute for Musicology of the Hungarian Academy of Sciences. The next generation application in Hungary is Labanatory (Misi, 2002); its latest version has the capability to search for almost all Labanotation signs or sign groups in a score, and its retrieval function is unique (Calvert–Wilke–Ryman–Fox, 2005).

The development of these systems was inspired by the need to support analyses of Central European traditional dances, based on the assumption that recurring movements correspond to repeating patterns in Labanotation, since similar movements are indicated in a similar way. This means that dance elements can be found with pattern matching in a purely formal way (Misi, 2005; Misi, 2006). These searches can be performed in any dance style in which movement repetitions exist. Searches are useful for spell-checking scores or checking the uniformity of sign usage.

To implement a retrieval system in a computer, a digital representation has to be defined for it. This paper describes the algebraic model and data structure used by Labanatory, which can be used for search as well as other operations. First of all, however, the general problems associated with searches and the related solutions will be demonstrated with a simple case, the case of English texts and text editors.

Writing represents speech with a sequence of letters. Handling texts written by hand and scanned from paper is a tedious exercise for the purposes of a computerized search. (Image processing algorithms are statistics-based, thus they cannot guarantee hits with 100% certainty.) Therefore, a special device, the keyboard is used for text input, where a text is typed character by character. Characters are stored in a computer as unique identifiers; a sequence of characters is represented with a sequence of integers¹. For example, the following text is represented with three integers in the standard ASCII coding:

s i t

115 105 116

Finding a text in another, longer text is achieved as a pattern search in the corresponding integer sequence representation: while the first integer pattern is gradually being slid on the second one, a match is examined between them². Where the two patterns match, a hit is indicated, and the place of the match in the representation shows the place of the hit in the text. For example:

s i t is to be searched in this text:

She sat, he did not s i t.

hit

that is in the representation,

115 105 116

is to be searched in this integer sequence:

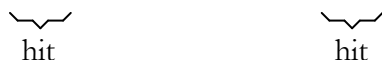
83 104 101 32 115 97 116 44 32 104 101 32 100 105 100 32 110 111 116 32 115 105 116 46

hit

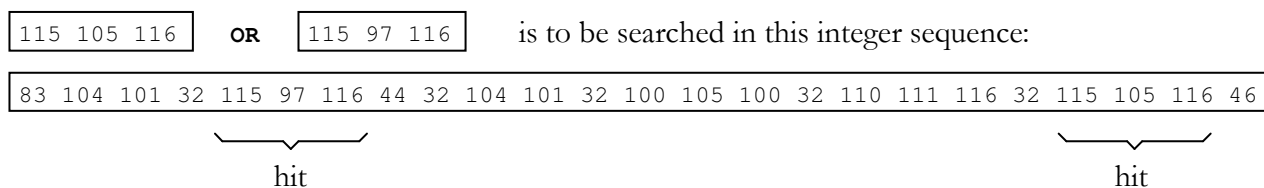
In the above search, the past tense of the word “sit” has not been found of course, because the three-character pattern did not match at the position of the text “sat”. Various forms of a word can be found with several consecutive searches or with a so-called complex search where logical operators (OR, NOT) are used in one query:

sit OR sat is to be searched in this text:

She sat, he did not sit.



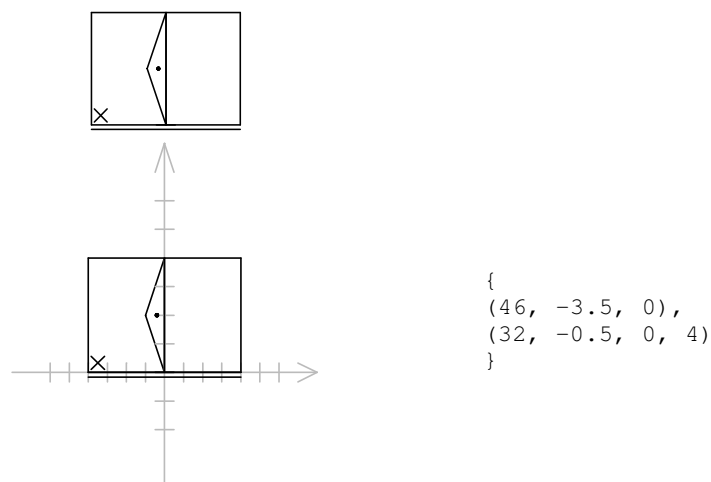
that is in the representation,



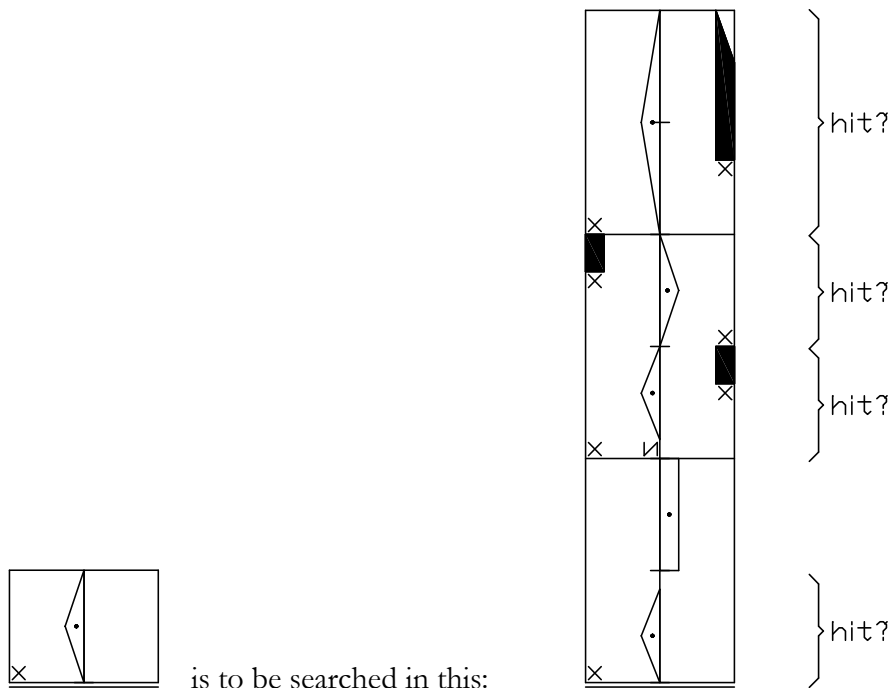
Labanotation and search

Analogously to written texts, processing scanned images of hand-drawn Labanotation scores is an inefficient exercise. Using a keyboard and a mouse as input devices, the user can select Labanotation signs and place them into a staff. As soon as a sign is selected, it is assigned an integer that is unique for its particular sign form. Besides these sign identifiers, the coordinates of the signs and the vertical length of stretchable signs (that is path, direction, revolution signs, action strokes, motion toward/away signs, vertical bows, inclusion bows, addition bows) have to be stored for each sign element of a Laban-pattern³.

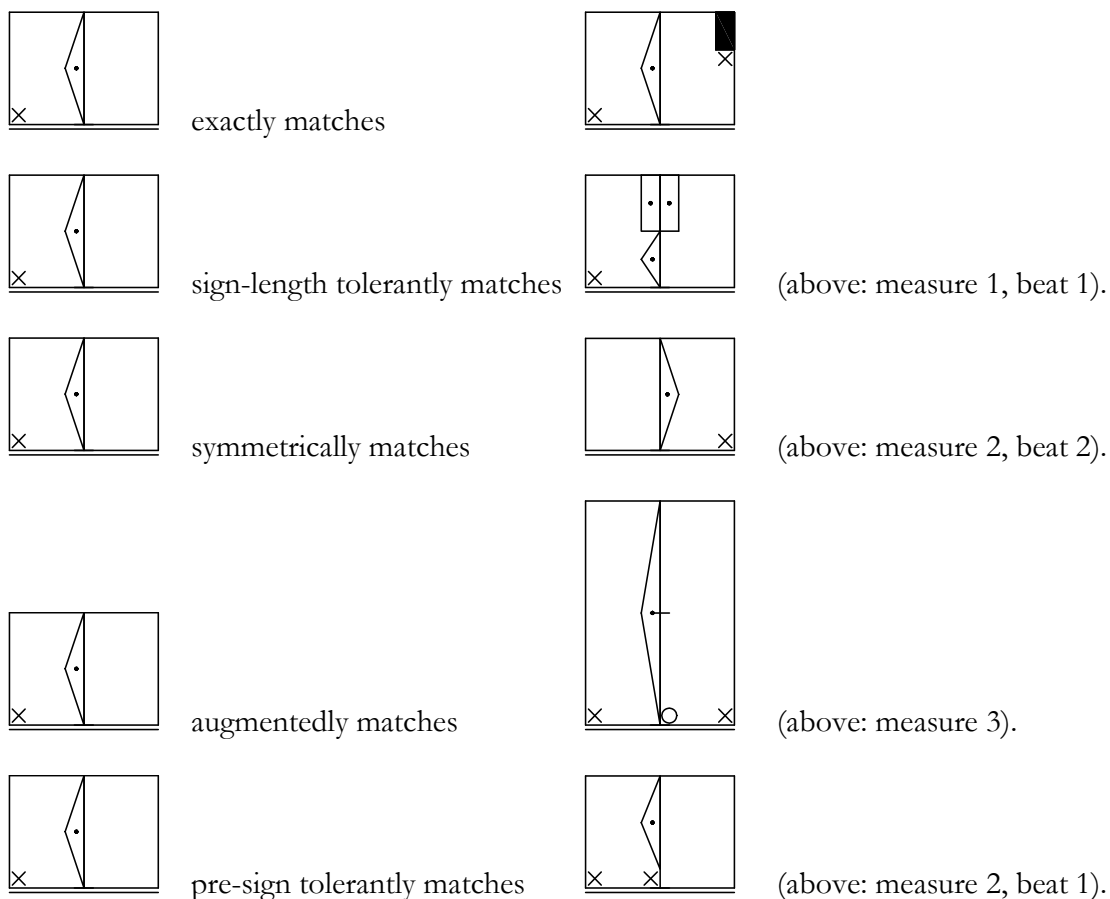
The kinetogram in the next figure contains two signs. In the second drawing, two coordinate axes are added so that the coordinates of the signs can be determined. (More precisely, the coordinates of the bottom line center point of the minimum bounding rectangle of each sign. The staff width is 8 units and the measure length is 4 units.) Next to the kinetogram, the representation of the Laban-pattern is shown, which lists the pattern elements: first the identifier of the space measurement sign (46) and its 2D coordinates, then the identifier of the direction sign (32), its 2D coordinates and its length.



To what extent is this representation suitable for retrieval purposes? Before answering this, other basic questions must be answered first. What general requirements can be set for searches on kinetograms? Where are the hits expected for example in the next search?



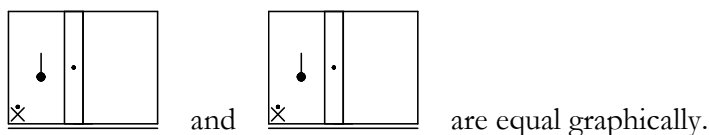
It is a normal expectation that hits should be detected at the places indicated with question marks, where the match of the sign-patterns is not perfect, but one of the different types of matches shown below applies. The definitions of these matches were introduced before (Misi, 2005; Misi, 2006), and only examples for them are shown here.



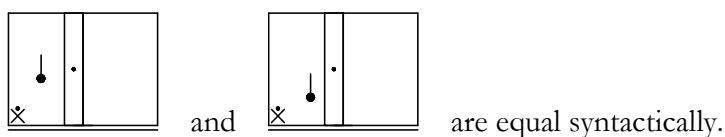
The 2D-coordinate-enumerating representation shown above allows the examination of all five matches, four of them quite easily (the exact match: with checking all the sign data; the sign-length tolerant match: with eliminating sign-length data; the symmetric match: with a certain horizontal mirroring; and the augmented match: with a certain vertical lengthening)⁴, but a simple operation is not sufficient for the fifth match. In the case of the pre-sign tolerant match, there is a vertical shift in the coordinate system a) by the length of a pre-sign (space measurements, body parts, joint signs, keys), b) but into only a certain direction (up, if there is no pre-sign and down, if a pre-sign is placed), since the shift cannot cross the measure line – and this is the point where understanding of time indication in Labanotation comes into play.

The match examples presented above are valid at the graphic level. Beyond the graphic level, however, syntactic and semantic levels also exist, and retrieval operating on these levels might also be expected. The concepts of graphic, syntactic and semantic levels were introduced in previous papers (Misi, 2005; Misi, 2006) and they are illustrated below with the following equalities.

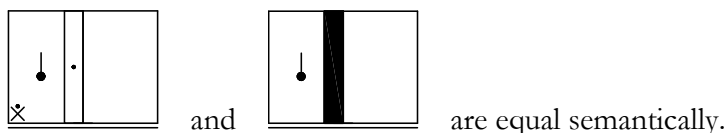
Two kinetograms are graphically equal if they describe movements with the same signs, each in the same 2D position. Example:



Two kinetograms are syntactically equal if they describe movements with the same signs, each with the same meaning. Example:



Two kinetograms are semantically equal if they describe the same movement. Example:



Labanotation users can rightly expect that a retrieval program can perform at least syntactic searches. As with the pre-sign tolerant match, syntactic operations require the recognition of a sign in a 2D environment: the environment where its meaning is still the same (under the definition of syntactic equality).

The aim is to have a representation that supports syntactic and pre-sign tolerant search, while allowing examination of all previously mentioned matches and their combinations. (It is possible to examine a match that is syntactical, symmetrical, augmented, sign-length tolerant and pre-sign tolerant at the same time.) This representation will be a 2D matrix described below.

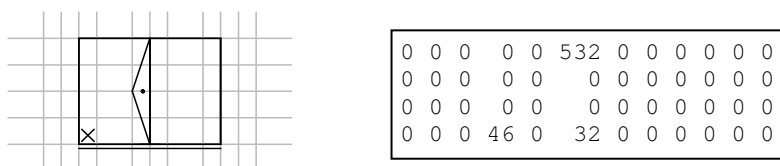
The basic idea is very simple. If an imaginary grid is placed on the kinetogram, coordinate intervals appear and can discretize x and y coordinate values. (If the signs are focused in two kinetograms, the corresponding signs will be put into the corresponding cells even if the signs are placed near but not exactly in the same 2D position. Matching these cells instead of examining coordinate values will decrease the computational effort.)

The first question is where the grid should be placed on kinetograms and how dense it should be. Horizontally, there is natural grid size that comes from Labanotation: the staff column width. Vertically, it is worth considering the start of beats (measure lines or tick marks) or subdivisions of the beats. The appropriate vertical grid size depends on the general rhythm of the dance in

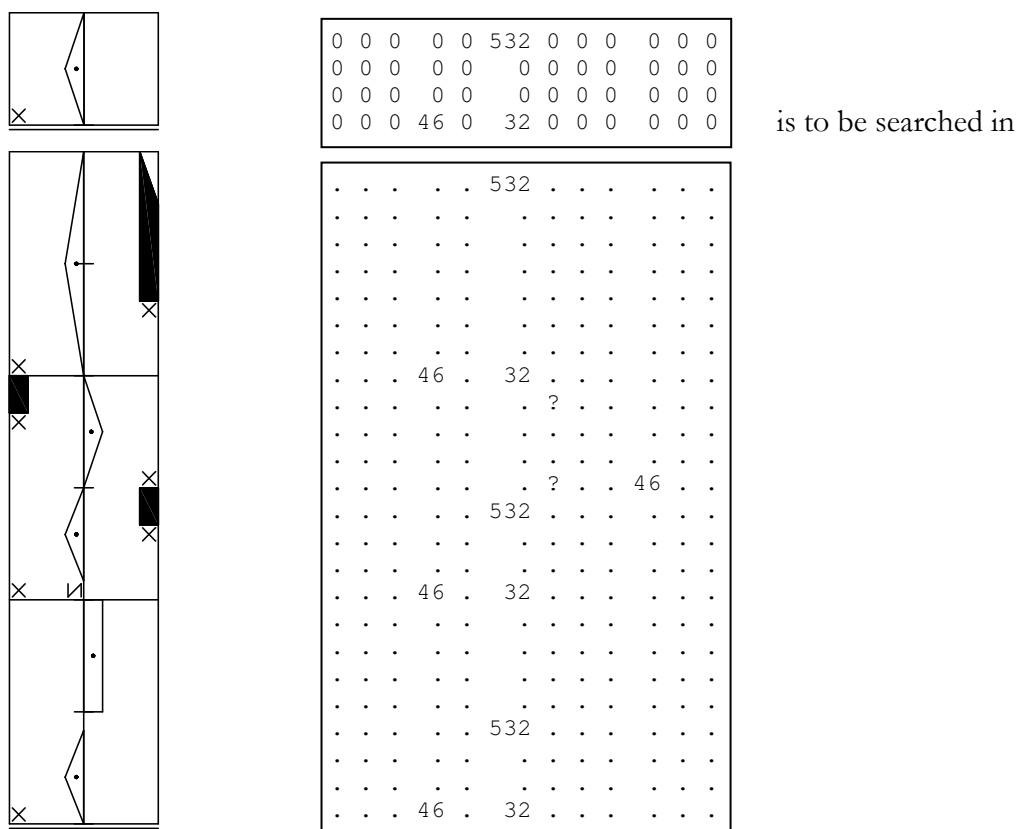
question. Its suggested value is the length of the shortest direction signs used in the score. (In the case of the analyzed Transylvanian dance (Misi, 2005), a measure was divided into two or four parts. The examples in this paper will show the quarter-divided representation.) There is no point in choosing a vertical grid size that is less than the length of a pre-sign, because, as pointed out later, only cells that are vertically long enough allow the examination of pre-sign tolerant matches.

Therefore, the representation of a kinetogram is a two-dimensional matrix. The elements of the matrix are identifiers of Labanotation signs. Each sign identifier is inserted into the matrix element that belongs to the grid cell in which the sign is placed. The end of each stretchable sign is represented in a separate cell, after the execution of an invertible mathematical operation on the given sign identifier. (In this way it is known whether an identifier represents a sign-end or not, and if yes, which sign end). The matrix elements are all zeros where there are no signs in the grid cell⁵.

In the next figure, the grid on the kinetogram determines 12 cell columns and 4 cell rows per measure. Next to the kinetogram, its matrix representation is shown. Three integers are inserted into the matrix: 46 is the identifier of the first degree of the space measurement sign, 32 means the left direction sign, and 532 indicates its sign-end (the value is calculated by adding 500).



With a well-constructed matrix representation, Labanotation search can be performed as follows: while a matrix pattern is being gradually slid on another matrix, a pattern match has to be examined for the non-zero elements⁶.



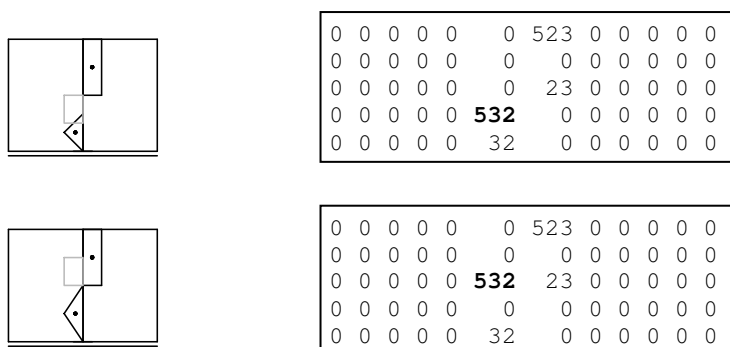
For the time being, question marks replace the codes of the symmetrical sign of the left direction sign; the integers will be given later. The dotted elements are not interesting in this search; regardless of the dots, which can be either zeros or any sign identifiers, the 2D pattern containing three integers can be found by sliding and matching it. Obviously, searches for any Laban pattern

can be executed in any other Laban pattern. General rules have to be established for the creation of matrices, and the rules have to ensure that the matrices, as integer patterns, will match if the corresponding Laban-patterns match.

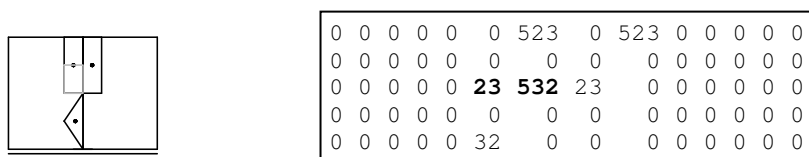
The task is to find general rules that describe the insertion of sign identifiers into the matrix. The aim is to insert all sign identifiers, and every one of them to a definite place in the matrix⁷. The matrix has to be well-defined: so that after the construction of two matrices, it can be examined in the same place in the matrices whether the identifiers of two corresponding signs match. On the other hand, the matrix has to allow searches for either individual signs or notation parts.

When implementing the idea of the matrix representation, the matrix will be defined more and more precisely. Certain problems are encountered during the implementation, and possible solutions for them are described below. The grey lines in the kinetograms show the problematic grid cell, and the bold numbers show the corresponding sign identifiers.

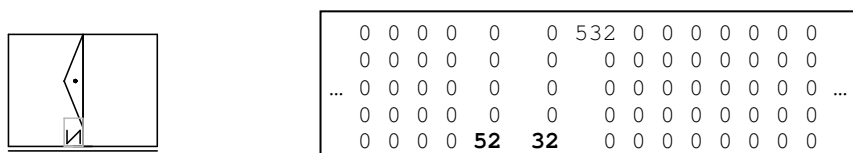
Problem 1: how to represent a step and a jump so that they can be distinguished? Solution: the vertical discretizing coordinate intervals will have to include the lower limit but exclude the upper limit. (This way a sign-end in the position of a cell border line – e.g. the end of a direction sign that indicates a step – will be inserted into the next time cell. Since this can be placed at the end of the staff, the matrix will contain one more row.)



Problem 2: how to represent a sign-end, if it is placed in a cell where another sign is also placed? Solution: new matrix columns will have to be introduced for sign-ends next to the staff columns of stretchable signs. (E.g. a direction sign in a support column will be represented in two matrix columns.)

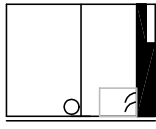


Problem 3: how to represent various signs if they are placed in the same cell? Solution: new matrix columns will have to be introduced for all the signs that can be placed in the relevant staff column. (E.g. since the support column can include both direction and space measurement signs, they will be represented in different matrix columns.)



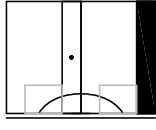
This example illustrates that having cells that are vertically long enough means that the start of a directions sign is placed into the same cell, whether the sign is preceded by a pre-sign or not. Therefore, the match of the representing matrices will induce a pre-sign tolerant match of the kinetograms. (This and the following figures do not show all the columns of the representing matrices, which is indicated with suspension points.)

Problem 4: how to represent doubled signs? Solution: new, secondary matrix columns will have to be introduced for signs that can be doubled. (One of the matrix column pairs has to be appointed as the primary column so that the sign identifier will be inserted into the primary one, if the relevant sign is not doubled.)



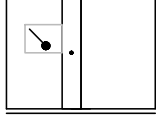
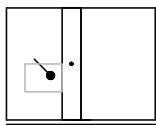
0	0	0	0	0	0	0	0	0	0	0	514
0	0	0	0	0	0	0	0	0	0	0	0
...	0	0	0	0	0	...	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	300	0	0	161	161	14	0

Problem 5: at which cell should a contact bow be represented? Solution: in order to have information about both body parts touching each other, two ends of a contact bow will have to be represented with two identifiers in the matrix. (More precisely, the left and the right end will be identified with different integers to distinguish them.)



0	0	0	0	0	532	0	0	0	0	0	512
0	0	0	0	0	0	0	0	0	0	0	0
...	0	0	0	0	...	0	0	0	0	...	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	139	23	0	0	0	-139	0	12	0

Problem 6: how to represent signs that have no time value, just modify the meaning of other signs? Idea (not developed solution): if an auxiliary sign does not have a standard place in Labanotation, and therefore its cell is not fixed, a matrix transformation should be performed: the identifiers of the auxiliary signs will be 'projected down' to the row of the identifier of its main sign. (E.g. a pin beside a direction sign will be represented at the start of the direction sign.)

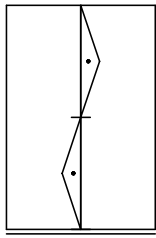



0	0	0	0	0	0	523	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0
...	0	0	94	0	0	...	0	0	0	0	...	0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	23	0	0	0	0	0	0	0

0	0	0	0	0	0	523	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0
...	0	0	0	0	0	...	0	0	0	0	...	0
0	0	94	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	23	0	0	0	0	0	0	0

0	0	0	0	0	0	523	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0
...	0	0	0	0	0	...	0	0	0	0	...	0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	94	0	0	23	0	0	0	0	0	0	0

Remark: ticks and measure lines can be represented as imaginary identifiers in a matrix column, for measure line relation searches. For symmetric searches, not only the symmetrical pairs of matrix columns, but the symmetrical pairs of sign identifiers have to be stored as well.



0	0	0	0	0	0	998	0	527	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0
...	0	0	0	0	0	532	997	27	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	32	0	998	0	0	0	0	0

↑
↑
↑
↑

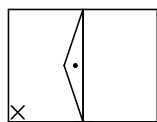
As the representation becomes more elaborate, the number of matrix columns increases quite rapidly. Nevertheless, it is worth representing signs separately, sign category by sign category. Notating a turn in support columns is a good example. Since it is indicated with a complex sign that has two graphical components, a rotation sign and a pin, and these are represented in different matrix columns, it is possible to search for the direction or the degree of the turn separately. Additionally, having matrix columns that belong to sign categories allows searches with wildcard signs (Misi, 2002): the identifier of a wildcard sign will be inserted into the matrix column that belongs to the sign category in question.

The representation that was improved during the development of Labanatory is a matrix containing 166 columns. 26 sign categories were defined⁸. One identifier is stored in the case of 14 sign categories, two identifiers are stored for 9 categories where the signs can be stretched (because of sign-start and sign-end), two identifiers are stored for 2 categories (hooks and dynamics signs) where the signs can be doubled, and four identifiers are stored for 1 category (the contact bows, which can also be doubled). This is $14+9\cdot2+2\cdot2+1\cdot4=40$ different matrix column types. 17 discretizing cells were defined horizontally on the basis of staff columns⁹, and therefore a matrix should contain $40\cdot17=680$ columns. There is no need for 680 columns, since certain signs are never placed in certain staff columns, and given signs can be placed in only one of a given group of neighboring columns. Insertion rules describe how to insert sign identifiers into a matrix representing a staff¹⁰, and the 17 staff columns are mapped to 166 (instead of 680) matrix columns under these rules.

The Labanatory software creates the matrices from digitized scores on the basis of sign data (Fügedi, 1999) and the grid size set by the user, then performs the requested search by sliding and matching the matrices.

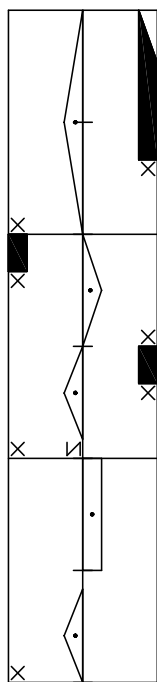
This figure shows the relevant matrix columns in the search of the example discussed above.

Searching



0	0	0	0	0	532	998	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0
...	0	0	0	...	0	0	0	0	0	0	0	...
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	46	0	32	0	998	0	0	0	0	0	0

in

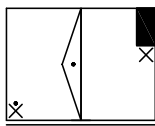


0	0	0	0	0	532	998	0	0	0	0	0	522
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	997	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	46	22	0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	512	46	0	32	0	998	0	527	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0
12	0	46	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0
...	0	0	0	...	0	0	532	997	27	0	0	...
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	46	12
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	46	52	32	0	998	0	523	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	997	23	0	0	0	0
0	0	0	0	0	532	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	46	0	32	0	998	0	0	0	0	0	0

Other operations

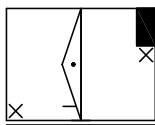
The matrix representation can also be used for operations other than searches. Intersection and subtraction of Laban-patterns¹¹ can be created by examining matrix cells rather than 2D coordinates, in a method that approaches the syntactic level. The next figure is an example for an intersection (common parts of kinetograms); the operation is performed with the matrices at right¹².

Creating the intersection of



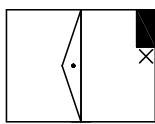
0	0	0	0	0	532	998	0	0	0	0	0	512			
0	0	0	0	0	0	0	0	0	0	0	0	0			
...	0	0	0	0	...	0	0	0	0	0	...	46	12	0	...
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	47	0	32	0	998	0	0	0	0	0	0	0	0

and



0	0	0	0	0	532	998	0	0	0	0	0	0	512		
0	0	0	0	0	0	0	0	0	0	0	0	0	0		
...	0	0	0	0	...	0	0	0	0	0	...	46	12	0	...
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	46	153	32	0	998	0	0	0	0	0	0	0	0

The result is



0	0	0	0	0	532	998	0	0	0	0	0	0	512		
0	0	0	0	0	0	0	0	0	0	0	0	0	0		
...	0	0	0	0	...	0	0	0	0	0	...	46	12	0	...
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	32	0	998	0	0	0	0	0	0	0	0

Additionally, the matrix representation can serve as the basis for methods of multivariate statistics and data mining that can reveal hidden relationships between elements of the notated dance.

Results

This paper described a representation of Labanotation that allows operations between kinetograms at a quasi-syntactic level, above the graphic level. This level is not purely syntactic, because whether the operations work on the syntactic level depends on how coordinates are discretized (on the selected grid size) and how matrices are transformed before the operations (see problem 6).

One may ask whether the user should really have such an in-depth knowledge of software representation when he simply wants to perform a search and expects hits from the computer. The answer is yes: the user can only use a complex system effectively if he has an understanding of the background processes. This is true in the case of searches in relational databases or in Internet contents. These systems – just as Labanotation – are more complex than the text search mentioned in the Introduction.

The matrix representation shown in this paper is not the only possible representation of Labanotation. It has no great importance in itself. It is more important that the paper defined matches on kinetograms, and described general requirements of searches (and other operations) for Labanotation systems in the future. The exact description of these requirements is necessary in order to meet the needs of Labanotation users.

Notes

¹ With mathematical description:

Let C be the set of characters (lowercase, uppercase letters, digits, punctuation marks and space).

Let $a_{n \times 1}^T = (a_1, \dots, a_n)$ and $b_{m \times 1}^T = (b_1, \dots, b_m)$ be two character row vectors, $a_i, b_j \in C$ ($1 \leq i \leq n, 1 \leq j \leq m, i, j, n, m \in \mathbb{N}$), $n \leq m$.

Let $f: C \rightarrow \mathbb{N}$ be an invertible character coding function. Coding the elements of the vectors above $c_i = f(a_i)$, $d_j = f(b_j)$, and now $c_{n \times 1}^T = (c_1, \dots, c_n)$ and $d_{m \times 1}^T = (d_1, \dots, d_m)$ vectors can be examined.

² Continuing the previous note, $a_{n \times 1}^T$ occurs in $b_{m \times 1}^T$ is said, if $\exists k \in \mathbb{N}_0$ that $\forall i$ ($1 \leq i \leq n$): $b_{i+k} = a_i$.

Similarly, $c_{n \times 1}^T$ vector occurs in $d_{m \times 1}^T$ is said, if $\exists k \in \mathbb{N}_0$ that $\forall i$ ($1 \leq i \leq n$): $c_{i+k} = d_i$.

It is provable that $c_{n \times 1}^T$ occurs in $d_{m \times 1}^T$, if and only if $a_{n \times 1}^T$ occurs in $b_{m \times 1}^T$.

³ As a mathematical structure: Let J be the set of Labanotation signs, $I \subset J$ the set of the graphically stretchable signs.

A Laban-pattern is defined as a finite set of (ordered) quadruples: $L \subset \{(j, x, y, h) \mid j \in J, x, y, h \in \mathbb{R}, h=0, \text{ if } j \in J \setminus I\}$.

⁴ Let L_1 and L_2 be two Laban-patterns. Definitions:

L_1 exactly matches L_2 is said, if $L_1 \subseteq L_2$, that is if

$\forall l_1 = (j_1, x_1, y_1, h_1) \in L_1 \exists l_2 = (j_2, x_2, y_2, h_2) \in L_2: j_1 = j_2, x_1 = x_2, y_1 = y_2, h_1 = h_2$.

L_1 sign-length tolerantly matches L_2 is said, if

$\forall l_1 = (j_1, x_1, y_1, h_1) \in L_1 \exists l_2 = (j_2, x_2, y_2, h_2) \in L_2: j_1 = j_2, x_1 = x_2, y_1 = y_2$.

L_1 symmetrically matches L_2 is said, if

L_{1sz} matches L_2 , where L_{1sz} is from L_1 with transformation $(j, x, y, h) \rightarrow (j, -x, y, h)$ on its elements.

L_1 augmentedly matches L_2 is said, if

L_{1a} matches L_2 , where L_{1a} is from L_1 with transformation $(j, x, y, h) \rightarrow (j, 2x, y, 2h)$ on its elements.

L_1 occurs in L_2 is said, if

$\exists t \in \mathbb{R}$ that L_{1t} matches L_2 , where L_{1t} is from L_1 with transformation $(j, x, y, h) \rightarrow (j, x, y+t, h)$ on its elements.

⁵ Retaining the above indications, let $f: J \rightarrow \mathbb{N}$ be a Labanotation sign coding invertible function.

Let $r \in \mathbb{N}$, $t_1, t_2, \dots, t_r \in \mathbb{R}$ and $d \in \mathbb{R}$ be given constants as parameters for discretization.

The matrix representation of the Laban-pattern L is calculated as follows: $g(L) = M_{n \times m} = (c_{ki})$, where $c_{ki} =$

$f(j)$, if $\exists (j, x, y, h) \in L: t_k \leq x < t_{k+1}, d \cdot (n-k) \leq y < d \cdot (n-k+1)$.

$f(j) + z$, $z \in \mathbb{N}$ is a fixed constant, $z > |J|$, if $\exists (j, x, y, h) \in L: j \in I, t_k \leq x < t_{k+1}, d \cdot (n-k) \leq y + h < d \cdot (n-k+1)$.

$= 0$ else.

⁶ Let L_1, L_2 be two Laban-patterns ($|L_1| \leq |L_2|$), their matrix representation: $g(L_1) = A_{n \times k} = (a_{ij})$, $g(L_2) = B_{m \times k} = (b_{ij})$.

Definition: $A_{n \times k}$ occurs in $B_{m \times k}$ is said, if $\exists l \in \mathbb{N}_0$ that $\forall i, j: b_{(i+l)j} = a_{ij}$, if $a_{ij} \neq 0$.

⁷ The target is to define $g()$ so that $g()$ should be a function where the next statement is true:

$A_{n \times k}$ occurs in $B_{m \times k}$, if and only if L_1 occurs in L_2 .

⁸ Enumerating in C programming language:

```
enum ESrchSignCategories {SRCH_CATG_NO_LEVEL_DIRECTION, SRCH_CATG_DIRECTION, SRCH_CATG_SPACE_MEASUREMENT,
SRCH_CATG_B_PINS, SRCH_CATG_T_PINS, SRCH_CATG_W_PINS,
SRCH_CATG_RETENTION, SRCH_CATG_CANCELLATION, SRCH_CATG_BOW_CARETS,
SRCH_CATG_BOW_INCLUSION, SRCH_CATG_BOW_VERTICALS_LEFT, SRCH_CATG_BOW_VERTICALS_RIGHT,
SRCH_CATG_WAVES, SRCH_CATG_TIME_EX, SRCH_CATG_STROKE_TAIL,
SRCH_CATG_DYNAMICS, SRCH_CATG_HOOKS, SRCH_CATG_CONTACT,
SRCH_CATG_CONTACT_BOW_HUN_P, SRCH_CATG_TORSO, SRCH_CATG_JOINTS,
SRCH_CATG_ROTATION, SRCH_CATG_PATH_STRAIGHT, SRCH_CATG_PATH_CIRCULAR,
SRCH_CATG_FACING, SRCH_CATG_TOOLS};
```

9

```
enum ESrchStaffColumns {SRCH_STAFF_COL__LEFT_OUTER_OTHER, SRCH_STAFF_COL__LEFT_FOREARM, SRCH_STAFF_COL__LEFT_ARM,
SRCH_STAFF_COL__LEFT_OUTER_AUX, SRCH_STAFF_COL__LEFT_TRUNK_PART, SRCH_STAFF_COL__LEFT_LEG_GESTURE,
SRCH_STAFF_COL__LEFT_INNER_AUX, SRCH_STAFF_COL__LEFT_SUPPORT, SRCH_STAFF_COL__SUPPORT_LINE,
SRCH_STAFF_COL__RIGHT_SUPPORT, SRCH_STAFF_COL__RIGHT_INNER_AUX, SRCH_STAFF_COL__RIGHT_LEG_GESTURE,
SRCH_STAFF_COL__RIGHT_TRUNK_PART, SRCH_STAFF_COL__RIGHT_OUTER_AUX, SRCH_STAFF_COL__RIGHT_ARM,
SRCH_STAFF_COL__RIGHT_FOREARM, SRCH_STAFF_COL__RIGHT_OUTER_OTHER};
```

¹⁰ The insertion rules for space measurements, direction signs and hooks are listed here:

```
static SSrchInsertionRule a_st_rules[]=
/*
{ sign_category,
  {{ sign_id_type, x_coord_inside_sign,
    {
      {staff_column, matrix_column,
        second_matrix_column},
      ...
    }, y_coord_inside_sign }
  ...
}},
*/
{SRCH_CATG_SPACE_MEASUREMENT,
  {{ SRCH_INS_ID, 0.5, {
    {SRCH_STAFF_COL__LEFT_OUTER_OTHER, SRCE_MATRIX_COL__LEFT_OTHER_X},
    {SRCH_STAFF_COL__LEFT_FOREARM, SRCE_MATRIX_COL__LEFT_ARM_X},
    {SRCH_STAFF_COL__LEFT_ARM, SRCE_MATRIX_COL__LEFT_ARM_X},
    {SRCH_STAFF_COL__LEFT_OUTER_AUX, SRCE_MATRIX_COL__LEFT_ARM_X},
    {SRCH_STAFF_COL__LEFT_TRUNK_PART, SRCE_MATRIX_COL__LEFT_TRUNK_X},
    {SRCH_STAFF_COL__LEFT_LEG_GESTURE, SRCE_MATRIX_COL__LEFT_LEG_GESTURE_X},
    {SRCH_STAFF_COL__LEFT_INNER_AUX, SRCE_MATRIX_COL__LEFT_LEG_GESTURE_X},
    {SRCH_STAFF_COL__LEFT_SUPPORT, SRCE_MATRIX_COL__LEFT_LEG_SUPPORT_X},
    {SRCH_STAFF_COL__RIGHT_SUPPORT, SRCE_MATRIX_COL__RIGHT_LEG_SUPPORT_X},
    {SRCH_STAFF_COL__RIGHT_INNER_AUX, SRCE_MATRIX_COL__RIGHT_LEG_GESTURE_X},
    {SRCH_STAFF_COL__RIGHT_LEG_GESTURE, SRCE_MATRIX_COL__RIGHT_LEG_GESTURE_X},
    {SRCH_STAFF_COL__RIGHT_TRUNK_PART, SRCE_MATRIX_COL__RIGHT_TRUNK_X},
    {SRCH_STAFF_COL__RIGHT_OUTER_AUX, SRCE_MATRIX_COL__RIGHT_ARM_X},
    {SRCH_STAFF_COL__RIGHT_ARM, SRCE_MATRIX_COL__RIGHT_ARM_X},
    {SRCH_STAFF_COL__RIGHT_FOREARM, SRCE_MATRIX_COL__RIGHT_ARM_X},
    {SRCH_STAFF_COL__RIGHT_OUTER_OTHER, SRCE_MATRIX_COL__RIGHT_OTHER_X},
  }, 0.01 }}}},
{SRCH_CATG_DIRECTION,
  {{ SRCH_INS_ID, 0.5, {
    {SRCH_STAFF_COL__LEFT_OUTER_OTHER, SRCE_MATRIX_COL__LEFT_OTHER_DIRECTION},
    {SRCH_STAFF_COL__LEFT_FOREARM, SRCE_MATRIX_COL__LEFT_ARM_GESTURE_FORE},
    {SRCH_STAFF_COL__LEFT_ARM, SRCE_MATRIX_COL__LEFT_ARM_GESTURE},
    {SRCH_STAFF_COL__LEFT_TRUNK_PART, SRCE_MATRIX_COL__LEFT_TRUNK_DIRECTION},
    {SRCH_STAFF_COL__LEFT_LEG_GESTURE, SRCE_MATRIX_COL__LEFT_LEG_GESTURE},
    {SRCH_STAFF_COL__LEFT_INNER_AUX, SRCE_MATRIX_COL__LEFT_LEG_GESTURE_LOW},
    {SRCH_STAFF_COL__LEFT_SUPPORT, SRCE_MATRIX_COL__LEFT_LEG_SUPPORT},
    {SRCH_STAFF_COL__SUPPORT_LINE, SRCE_MATRIX_COL__ANY_ADLIB},
    {SRCH_STAFF_COL__RIGHT_SUPPORT, SRCE_MATRIX_COL__RIGHT_LEG_SUPPORT},
    {SRCH_STAFF_COL__RIGHT_INNER_AUX, SRCE_MATRIX_COL__RIGHT_LEG_GESTURE_LOW},
    {SRCH_STAFF_COL__RIGHT_LEG_GESTURE, SRCE_MATRIX_COL__RIGHT_LEG_GESTURE},
    {SRCH_STAFF_COL__RIGHT_TRUNK_PART, SRCE_MATRIX_COL__RIGHT_TRUNK_DIRECTION},
    {SRCH_STAFF_COL__RIGHT_ARM, SRCE_MATRIX_COL__RIGHT_ARM_GESTURE},
    {SRCH_STAFF_COL__RIGHT_FOREARM, SRCE_MATRIX_COL__RIGHT_ARM_GESTURE_FORE},
    {SRCH_STAFF_COL__RIGHT_OUTER_OTHER, SRCE_MATRIX_COL__RIGHT_OTHER_DIRECTION},
  }, 0.01 }},
  { SRCH_INS_END, 0.5, {
    {SRCH_STAFF_COL__LEFT_OUTER_OTHER, SRCE_MATRIX_COL__LEFT_OTHER_DIRECTION_END},
    {SRCH_STAFF_COL__LEFT_FOREARM, SRCE_MATRIX_COL__LEFT_ARM_GESTURE_FORE_END},
    {SRCH_STAFF_COL__LEFT_ARM, SRCE_MATRIX_COL__LEFT_ARM_GESTURE_END},
    {SRCH_STAFF_COL__LEFT_TRUNK_PART, SRCE_MATRIX_COL__LEFT_TRUNK_DIRECTION_END},
    {SRCH_STAFF_COL__LEFT_LEG_GESTURE, SRCE_MATRIX_COL__LEFT_LEG_GESTURE_END},
    {SRCH_STAFF_COL__LEFT_INNER_AUX, SRCE_MATRIX_COL__LEFT_LEG_GESTURE_LOW_END},
    {SRCH_STAFF_COL__LEFT_SUPPORT, SRCE_MATRIX_COL__LEFT_LEG_SUPPORT_END},
    {SRCH_STAFF_COL__SUPPORT_LINE, SRCE_MATRIX_COL__ANY_ADLIB_END},
    {SRCH_STAFF_COL__RIGHT_SUPPORT, SRCE_MATRIX_COL__RIGHT_LEG_SUPPORT_END},
    {SRCH_STAFF_COL__RIGHT_INNER_AUX, SRCE_MATRIX_COL__RIGHT_LEG_GESTURE_LOW_END},
    {SRCH_STAFF_COL__RIGHT_LEG_GESTURE, SRCE_MATRIX_COL__RIGHT_LEG_GESTURE_END},
    {SRCH_STAFF_COL__RIGHT_TRUNK_PART, SRCE_MATRIX_COL__RIGHT_TRUNK_DIRECTION_END},
    {SRCH_STAFF_COL__RIGHT_ARM, SRCE_MATRIX_COL__RIGHT_ARM_GESTURE_END},
    {SRCH_STAFF_COL__RIGHT_FOREARM, SRCE_MATRIX_COL__RIGHT_ARM_GESTURE_FORE_END},
    {SRCH_STAFF_COL__RIGHT_OUTER_OTHER, SRCE_MATRIX_COL__RIGHT_OTHER_DIRECTION_END},
  }, 1 }}}},
{SRCH_CATG_HOOKS,
  {{ SRCH_INS_ID, 0.5, {
    {SRCH_STAFF_COL__LEFT_LEG_GESTURE, SRCE_MATRIX_COL__LEFT_SOLE},
    {SRCH_STAFF_COL__LEFT_INNER_AUX, SRCE_MATRIX_COL__LEFT_SOLE_2ND},
    {SRCH_STAFF_COL__LEFT_SUPPORT, SRCE_MATRIX_COL__LEFT_SOLE_2ND},
    {SRCH_STAFF_COL__RIGHT_SUPPORT, SRCE_MATRIX_COL__RIGHT_SOLE},
    {SRCH_STAFF_COL__RIGHT_INNER_AUX, SRCE_MATRIX_COL__RIGHT_SOLE_2ND},
    {SRCH_STAFF_COL__RIGHT_LEG_GESTURE, SRCE_MATRIX_COL__RIGHT_SOLE_2ND},
  }, 0.5 }}}},
/*
...
*/
};
```

¹¹ Size, intersection and subtraction of Laban-patterns are defined as set operations ($|L|, L_1 \setminus L_2, L_1 \cap L_2$).

¹² Let $A_{n \times m} = (a_{ij})$ and $B_{n \times m} = (b_{ij})$ be two matrices, $a_{ij}, b_{ij} \in \mathbb{N}_0$ ($n, m, i, j \in \mathbb{N}, 1 \leq i \leq n, 1 \leq j \leq m$).

Projected intersection of $A_{n \times m}$ and $B_{n \times m}$ is $C_{n \times m} = (c_{ij})$, where $\forall i, j: c_{ij} =$

$= a_{ij}$, if $a_{ij} = b_{ij}$.

$= 0$, if $a_{ij} \neq b_{ij}$.

Bibliography

Calvert, Tom – Wilke, Lars – Ryman, Rhonda – Fox, Ilene (2005): Applications of Computers to Dance. IEEE Computer Graphics and Applications XXV/2. pp. 6-12.

Fügedi, János (1995): Applying dance structure and motive collections by computer - DanceStruct 1.0 Proceedings of the 19th Biennial Conference of ICKL, Paris, 1995. pp. 71-82.

Fügedi, János (1999): “Labanatorium” szimbólum analízis. Manuscript. Hungarian Academy of Sciences, ZTI Akt. 1488.

Misi, Gábor (2002): Labanatory. A computer program to analyse dance. Booklet of Abstracts for the 22nd Symposium of the ICTM Study Group on Ethnochoreology, Szeged, 2002. p. 18. <www.labanatory.com/common/download/2002ICTM/ICTM02abs.pdf>

Misi, Gábor (2005): Az erdélyi férfitáncok formális elemzéséről. A tévesztések, korrigálások, illetve általában a táncelem-kapcsolatok vizsgálatával. Manuscript. Hungarian Academy of Sciences, ZTI Akt. 1653. <www.labanatory.com/common/download/2006MGYE/Formalis.pdf>

Misi, Gábor (2006): Formal methods in form analysis of Transylvanian male solo dances. Manuscript. Hungarian Academy of Sciences, ZTI Akt. 1660. <www.labanatory.com/common/download/2006ICTM/Formalm.pdf>