

# Generation of Moment Invariants by Tensor Method with the Software “Afinvtensors”

Tomáš Suk

The Czech Academy of Sciences,  
Institute of Information Theory and Automation.  
Pod vodárenskou věží 4, 182 08 Praha 8, Czech Republic  
`suk@utia.cas.cz`  
`https://www.utia.cas.cz`

February 13, 2020

## 1 Procedure

The software “Afinvtensors” is intended for generation of moment invariants both to rotation and to affine transformation of 2D and 3D images, vector fields and tensor fields. It uses the tensor method combined with the graph method for generation of individual invariants.

The software is written in language C++ in the Microsoft Visual Studio Community 2017. The screenshot of the main dialog window is in Fig. 1.

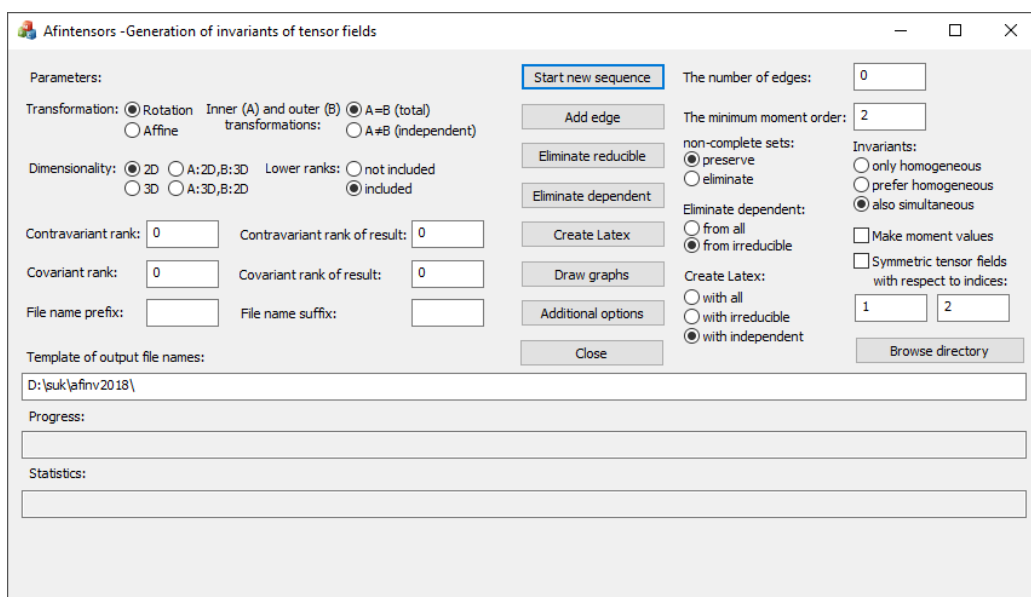


Figure 1: Main dialog window.

## 1.1 Start

The root directory is called “afinvtensors”. The executable file “afinvtensors.exe” is then in the subdirectory “\afinvtensors\x64\Release” or “\afinvtensors\x64\Debug”. The subdirectories “\afinvtensors\Release” and “\afinvtensors\Debug” contain 32-bit versions. After launching, this process makes a dialog window for starting other operations. The standard order of operations is following:

1. Fill the parameters (transformation, dimensionality, tensor field ranks, and also output directory).
2. Press the “Start new sequence” button.
3. Press the “Add edge” button repeatedly, until the time of computation is too long.
4. When the number of generated invariants is sufficient, press the “Eliminate reducible” button.
5. Press the “Eliminate dependent” button.
6. Press the “Create Latex” button.
7. Press the “Draw graphs” button.
8. Translate the made file with the suffix “indep” and extension “tex” by some translator from Latex to dvi, postscript or pdf, e.g. Miktex.

The algorithm has exponential computing complexity (very approximately  $\mathcal{O}(n_e!)$ , where  $n_e$  is the number of graph edges), therefore it is initially very fast, but the computing time quickly increases with  $n_e$ . Other limits are memory consumption (The error message: “Cannot allocate memory for *variable*”) and precision.

As an illustration of the exponential computing complexity in practice, see Tab. 1.

Table 1: Statistics for rotation invariants of 2D images from the minimum moment order 0.

edges	graphs	invariants	time
0	1	1	0.002 s
1	2	2	0.003 s
2	10	7	0.007 s
3	62	23	0.038 s
4	442	79	0.199 s
5	3458	274	1.214 s
6	28882	998	12.015 s
7	253138	3722	2 min 55.681 s
8	2301842	14399	59 min 53.889 s
9	21545762	57433	1 day 3 h 42 min 2.139 s

Now description of the individual parameters in more details follows.

## 1.2 Parameters

At the left part of the dialog window, there are basic parameters. The default values are set to generate rotation invariants of 2D images from the second order. In all other cases, the parameters must be set.

### 1.2.1 Transformation

The software produces invariants to linear transformations of coordinates

$$\mathbf{x}' = \mathbf{A}\mathbf{x}, \text{ where } \mathbf{x} = (x^1 \dots x^d)^T. \quad (1)$$

Selection “Rotation” means the matrix  $\mathbf{A}$  is orthogonal with  $\det \mathbf{A} = 1$  and the transformation is just rotation. The selection “Affine” means there are no constraints on the matrix  $\mathbf{A}$ . The default is rotation.

### 1.2.2 Dimensionality

The number of dimensions of the original space. The possible choices are “2D” (then  $d = 2$ ) or “3D” (then  $d = 3$ ). The default is 2D.

There are also choices “A:2D,B:3D” and “A:3D,B:2D”. It means that dimensions of the original space and the space of tensor values differ, see the Sec. 1.2.3. In the case “A:2D,B:3D”, the dimension of the original space  $d = 2$  and the dimension of the tensor values is 3. The matrix of the inner transformation  $\mathbf{A}$  has the size  $2 \times 2$  and the matrix of the outer transformation  $\mathbf{B}$  has the size  $3 \times 3$ . We can take a color image, what is 2D object, where we have three channels RGB in each point. The case “A:3D,B:2D” is rather theoretical, it means 3D data with two channels.

The choices “A:2D,B:3D” and “A:3D,B:2D” bring some limitations on other parameters. The inner (A) and outer (B) transformations cannot equal, when they have different sizes, so the choice “A $\neq$ B (independent)” is automatic, the choice “A=B (total)” is not possible. When the matrix B should have some prescribed size, it must exist, i.e. we must choose some original rank greater than zero, otherwise the error message “If the dimensions of the inner and outer transformations are different, the contravariant rank or the covariant rank must be greater than zero.” is used. E.g. in the case of color images, we choose the contravariant rank  $r_i = 1$ .

### 1.2.3 Inner (A) and outer (B) transformations

The inner transformation deals with the coordinates, the outer transformation works with the tensor values. If we have a tensor field in the narrow sense, then inner transformation equals the outer one and we talk about total transformation. Therefore the option A=B is default. If sometimes we need invariants for the case of different transformations (we talk about independent transformations), we can select A $\neq$ B. Then the graphs with inadmissible bi-color edges are not evaluated and only the invariants to the independent transformation are generated. If we work with images (input rank is zero), this option has no meaning.

#### 1.2.4 Lower ranks

When we generate partial invariants with some output ranks greater than 0, then the algorithm for elimination of reducible invariants needs also all lower output ranks. So, the default option is “included”. When we choose “not included”, only partial invariants with the defined output ranks are generated, but then we must omit “Eliminate Reducible” and use directly “Eliminate dependent” with option “from all”. When both output ranks are zero, this option has no influence.

#### 1.2.5 Contravariant rank

The original contravariant rank  $r_i$  of the tensor field.

#### 1.2.6 Covariant rank

The original covariant rank  $o_i$  of the tensor field. If both  $r_i = 0$  and  $o_i = 0$  (default), we generate invariants of images. If  $r_i = 1$  and  $o_i = 0$ , we generate invariants of vector fields. If  $r_i = 0$  and  $o_i = 1$ , it would be the purely theoretical case, when the inner transformation is inverse of the outer transformation. If  $r_i + o_i \geq 2$ , we obtain invariants of tensor fields.

#### 1.2.7 Contravariant rank of result

The number  $r_o$  of contravariant indices that do not come under the contraction. Therefore  $r_o \leq r_i$  must be valid.

#### 1.2.8 Covariant rank of result

The number  $o_o$  of covariant indices that do not come under the contraction. Therefore  $o_o \leq o_i$  must be valid. When  $r_o + o_o = 1$ , the results are partial invariants in the form of vectors that are multiplied once by the transformation matrix during the transformation. They are suitable for normalization, because we can compute the normalizing transformation from the values of two partial invariants (three in 3D).

#### 1.2.9 The number of edges

The current number  $n_e$  of edges of the generated graphs. The operation “Add edge” increases  $n_e$  by one automatically. It must be set manually, when some return to previous results is asked. In the case of affine invariants of images,  $n_e$  (usually) equals the order of the generated invariants. In the case of rotation, it equals the order of the complete set of the invariants, the maximum order is double  $n_e$ . In the case of the tensor fields, some edges are consumed to the tensor field indices and the order is  $n_e - r_i - o_i$ .

#### 1.2.10 The minimum moment order

The zeroth-order moments are usually used for the normalization to scaling and the first-order moments for normalization to translation, therefore the default value is 2. Sometimes, e.g. when we are about to derive the normalization to scaling and translation, we need also the zeroth- and first-order invariants. Then it is suitable to set this parameter to zero.

### 1.2.11 Homogeneous and simultaneous invariants

The homogeneous invariants are only composed from the moments of the same order, while the simultaneous invariants contain the moments of more than one order. The default option “also simultaneous” leads to preference of usually simpler simultaneous invariants. The option “prefer homogeneous” leads to the same result of the operations “Start new sequence” and “Add edges”, but during operations “Eliminate reducible” and “Eliminate dependent”, the simultaneous invariants are removed first. The option “only homogeneous” leads to removing all simultaneous invariants immediately in the operations “Start new sequence” and “Add edges”.

### 1.2.12 Symmetric tensor fields

Some tensor fields in physics, e.g. stress tensor, are symmetric, i.e.

$$\mathbf{T}^{i_1 i_2}(x^1 \dots x^d) = \mathbf{T}^{i_2 i_1}(x^1 \dots x^d). \quad (2)$$

The indices  $i_1$  and  $i_2$  can be both contravariant and covariant. Then the moments are also symmetric

$$m_{p_1 \dots p_d}^{(i_1 i_2)} = m_{p_1 \dots p_d}^{(i_2 i_1)} \quad (3)$$

and we can simplify the invariants. From the two moments in (3) only  $m_{p_1 \dots p_d}^{(i_1 i_2)}$  with  $i_1 \geq i_2$  is used, the other moment is substituted by the first one.

It works well for the tensor fields of the common rank (contravariant plus covariant) equaling two. Checking of this box has no meaning for images and vector fields. When the common rank is greater than two, we must use the two edit boxes below the check box for definition, what index is symmetric with what index.

### 1.2.13 Template of output names

Fill only the directory with the output files here. It is possible to use the button “Browse directory”. The names of the output files are composed automatically from the following abbreviations:

- “af” affine transformation,
- “rot” rotation,
- “3D” three-dimensional (two-dimensional is default without an abbreviation),
- “2D3D” inner three-dimensional and outer two-dimensional transformations,
- “3D2D” inner two-dimensional and outer three-dimensional transformations,
- “adifb” = “A differ from B”, the inner transformation can differ from the outer one,
- “ts” = “total skew”, the inner and outer transformations must equal,
- “img” = “images”, the numbers of both contravariant and covariant indices  $r_i$  and  $o_i$  equal zero.
- “vect” = “vector fields”,  $r_i + o_i = 1$ ,

- when  $r_i + o_i \geq 2$ , the notation “ $r_i\text{-}o_i\text{tens}$ ” is used,
- “norm” means  $r_o = 1$  and  $o_o = 0$ , else the notation “ $r_o\text{-}o_o\text{norm}$ ” is used,
- “inv” = “invariant” means  $r_o + o_o = 0$  and full invariants are generated,
- “hmg”, only homogeneous invariants are generated,
- “enc”, eliminate non-complete sets,
- “sym”, symmetric tensor fields,
- “zo” = “also zero and one”, the minimum moment order equals zero,
- “o” = “also one”, the minimum moment order equals one,
- when the minimum moment order  $m_o$  is greater than 2, the notation “ $\text{from}m_o$ ” is used,
- “lr” = “lower ranks”, the partial invariants with lower output ranks than  $r_o$  and  $o_o$  are also generated,
- the number of edges,
- “g”, the file with graphs,
- “irred”, the file with irreducible invariants only,
- “indep”, the file with independent invariants only,
- “deriv”, the file with derivatives of the invariants,
- “labels”, the file with node labels of the independent invariants,
- “\_statistics” the file with statistics of the computations.
- “\_momval” the file with prescribed moment values for the dependence test.

The parameter “File name prefix” is inserted before the file name and the parameter “File name suffix” is inserted behind the file name (more precisely between “lr” and the number of edges), so, it is possible to distinguish a few sequences of the invariants with the same parameters.

The Latex command ‘\graphicspath’ cannot work with non-standard characters (that with ASCII code higher than 127). If you work with Postscript graphs in the Latex files, avoid such characters in the prefix and suffix, please.

The operation “Start new sequence” make new directory with the name composed from the abbreviations from “af” to “lr”, i.e. without the number of edges. The new sequence of files with the increasing number of edges is saved into this new directory.

### 1.3 Operations

Here, we add some other comments to individual operations.

### 1.3.1 Start new sequence

It generates the graphs of the given parameters and the invariants from them. Before it, the old files with invariants, with graphs and with statistics are deleted, if they exist. It does not increase the number of edges in the graphs.

### 1.3.2 Add edge

It generates the graphs of the given parameters and the invariants from them. Before it, it increases the number of edges in the graphs by one. It opens the files for append and inserts the results to the files from less numbers of edges.

Similarly as “Start new sequence”, the invariant generation according to an individual graph can lead to four possible results: the graph does not pass through the test of admissibility, the result is zero, the result is the same as some previous invariant or a new invariant is generated. The statistics does not count the graphs that have not passed through the test of admissibility, but it is possible to calculate them as the number of graphs minus zeros, identical and written.

There is an option “Make moment values” with the default state unchecked. When this check box is checked, a special file with the suffix “\_momval” is created. It contains a matrix of moment values filled by the letters ‘r’ for each channel of the tensor field. In 3D, it is a few matrices. When the letter ‘r’ is substituted by some number, the number is used as static moment value during the dependence test. It works also during “Start new sequence”.

### 1.3.3 Eliminate reducible

It removes the products and the linearly dependent invariants from the file. It has one option: non-complete sets “preserve” (default) or “eliminate”. When “eliminate non-complete sets” is chosen, the numbers of the invariants of the individual orders are counted and compared with the theoretical number. When the actual number is less than the theoretical one, it is sure the set of this order is not complete and the invariants are eliminated directly before the multiplication test, what can save some time. The option “preserve non-complete sets” means all the input invariants are tested and when pass, they are included to the result.

### 1.3.4 Eliminate dependent

It removes (also) the polynomially dependent invariants from the file. It has one option: the default “from irreducible”, it means the input is the file with the irreducible invariants (the result of the operation “Eliminate reducible”). When the option “from all” is chosen, the input is the file from “Start new sequence” or “Add edge” is used. The operation has additional options.

When the “Make moment values” check box is checked and the special file with the suffix “\_momval” is found, the static numbers in this file are used as the moment values during the dependence test.

### 1.3.5 Create Latex

It transcripts the chosen txt file to Latex. The options are “with all” for transcription from the result of “Start new sequence” or “Add edge” operations. The option “with irreducible” means transcription from “Eliminate reducible”, and the option “with independent” means transcription from “Eliminate dependent”.

Implicitly, the file contains the references to images with the graphs. The graphs can be made by “Draw graphs” operation or the references can be removed, typically by operation “substitute `\includegraphics`’ by `%\includegraphics`’ ” in some text editor. The operation has additional options.

### 1.3.6 Draw graphs

It makes a special subdirectory with the graphs in Postscript format for the tex files. The operation has additional options.

### 1.3.7 Additional options

It creates a new dialog window making accessible some additional options for operations “Eliminate dependent”, “Create Latex”, and “Draw graphs”. It is these parameters:

- “Eliminate dependent”
  - “Number of tests” – the number of sets of random values of moments for the test of independence (default 5)
  - “Tolerance SVD” – tolerance of zero values in singular value decomposition. The default 0 means it is computed according to the formula  $\epsilon_d \cdot \max(n_v \cdot n_p, n_t) \cdot v$ , where  $\epsilon_d = 2.2204460492503131 \cdot 10^{-16}$  is the smallest number, whose sum with 1 is greater than 1 in double precision floating point arithmetics,  $n_v$  is the number of tested invariants,  $n_p$  is the number of parts of the partial invariants,  $n_t$  is the number of different moments occurring in the tested invariants and  $v$  is the maximum singular value in the tested matrix. The specific value is written in the statistic file. If the test “Eliminate dependent” returns repeatedly different rank of the tested matrix, it is useful to increase this value.
  - “Tolerance RREF” – tolerance of zero pivots in reduced row echelon form. The default 0 means it is computed according to the formula  $\epsilon_d \cdot \max(n_v \cdot n_p, n_t) \cdot w$ , where  $w$  is the maximum absolute value in the tested matrix. The specific value is written in the statistic file. If the test “Eliminate dependent” returns repeatedly different list of independent invariants, it is useful to increase this value typically to the square root of the value from the statistic file.
  - “Derivatives on disk” – “none” (default), the derivatives of the invariants are neither written to disk nor read from disk, “write”, the derivatives are newly computed and written to disk, “read”, the derivatives are read from disk.
- “Create Latex”
  - “Characters in line” – the number of characters in one row of the Latex file (default 83). The average subscript size is used, average standard size is calculated as  $1.5 \times$  greater.



- “Lines in page” – the number of rows in one page of the Latex file (default 38).
- “Draw graphs”
  - “Include border” – if checked (default), the bounding box of the graph image is set “Horizontal size”  $\times$  “Vertical size”, if not, the tight bounding box is computed. It saves space in a paper, but the graphs have various size, otherwise you must set the figure size in the `\includegraphics` command manually.
  - “Describe nodes” – if checked, the nodes are labeled by numbers from 1, if not (default), no description is used.
  - “Horizontal size” – the horizontal size of the graph image in pt (default 256 pt). One typographic point (pt) equals  $0.3527 = 127/360$  mm.
  - “Vertical size” – the vertical size of the graph image in pt (default 256 pt).
  - “Radius of a node” – the radius of the disk representing the node in pt (default 5 pt).
  - “Distance of edges” – the distance between adjacent multiple edges in pt (default 8 pt).
  - “Edge thickness” – the thickness of the edges in pt (default 1 pt).
  - “Color of edges” – the red, green and blue components of the colors of individual edge types. The default is black coordinate edges (0,0,0), magenta contravariant edges (255,0,255), and green covariant edges (0,255,0).

The screenshot of the dialog window with the additional options is in Fig. 2.

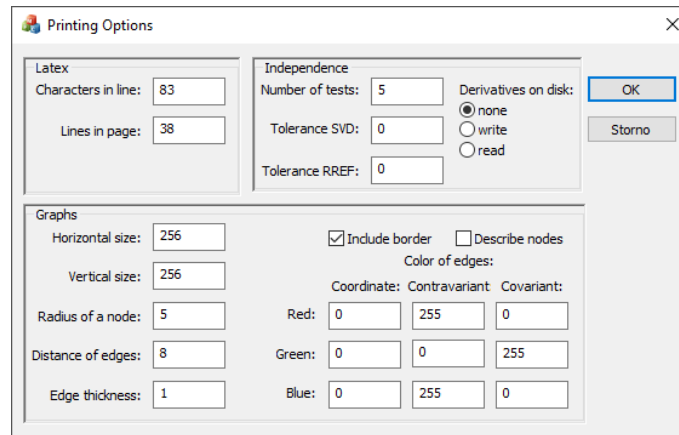


Figure 2: Dialog window with additional parameters.

### 1.3.8 Close

It closes the dialog window. Be careful not to use it before the last operation finishes.

## 2 Formats of the Generated Files

The files intended for the translation by Latex have the extension “.tex”, the graphs have the extension “.eps”, and other files have extension “.txt”.

## 2.1 Files with Invariants

The file contains a few (sometimes a lot of) invariants separated by an empty row. The first row of each invariant consists of three numbers  $n_e$   $n_c$   $n_r$ , where  $n_e$  is the number of edges of the generating graph,  $n_c$  is the number of terms and  $n_r$  is the number of moments in one term. There is one exception. If the graph is

A1

A0,

it is just a node without any edge, but  $n_e = 1$  in this case. So,  $n_e$  is rather the number of columns necessary for storage of the graph to the memory.

Then,  $n_c$  rows with individual terms follow. The first number in the row is a coefficient and the rest of the row contains the indices of the moments in order: coordinate indices, contravariant indices and covariant indices. The entire number of indices  $n_i$  of one moment can be calculated from the number of integers in one row  $n_n$  as  $n_i = (n_n - 1)/n_r$ . The precise numbers of specific types of indices can be found from the file name. E.g. the file “rot3D2\_0tensinvzo4indep.txt” contains  $c_i = 3$  coordinate indices at each moment (“3D” in the file name),  $r_i = 2$  contravariant indices and  $o_i = 0$  covariant indices (“2\_0tens” in the file name). If there is an invariant

```

2  9  1
1  2  0  0  0  0
1  0  2  0  0  0
1  0  0  2  0  0
1  2  0  0  1  1
1  0  2  0  1  1
1  0  0  2  1  1
1  2  0  0  2  2
1  0  2  0  2  2
1  0  0  2  2  2

```

that means the invariant

$$m_{200}^{(00)} + m_{020}^{(00)} + m_{002}^{(00)} + m_{200}^{(11)} + m_{020}^{(11)} + m_{002}^{(11)} + m_{200}^{(22)} + m_{020}^{(22)} + m_{002}^{(22)}$$

has 2 columns of the generating graph, 9 terms and one moment in each term. The contravariant and covariant indices are written in parenthesis here to distinguish them from the coordinate indices.

The files with the partial invariants differ, the first row of each invariant includes eight numbers  $n_e$   $n_c$   $n_r$   $c_i$   $r_i$   $o_i$   $r_o$   $o_o$ , where  $c_i$ ,  $r_i$ , and  $o_i$  are the same in the entire file,  $r_o$  or  $o_o$  can differ. The invariant

```

2  4  1  2  1  0  1  0
1  1  1  0
-1  2  0  1
1  0  2  0
-1  1  1  1

```

is then a partial invariant with  $r_o = 1$

$$P^{(1)} = m_{11}^{(0)} - m_{20}^{(1)}$$

$$P^{(2)} = m_{02}^{(0)} - m_{11}^{(1)}$$

and with the one-based output index.

The files with derivatives have two particularities. The first row consists of seven numbers defining the size of the file  $n_v$   $n_t$   $c_i$   $r_i$   $o_i$   $r_o$   $o_o$ , where  $n_v$  is the number of derived invariants and  $n_t$  is the number of moments in the denominators of the derivatives, so, the file contains  $n_v \times n_t$  derivatives. It includes also zero derivatives in the form of three zeros in one row.

## 2.2 Files with Graphs

We use the following notation for description of the graphs. The nodes are numbered from 1 to  $n_d$ , each edge is written in one column. In them, 0 is imaginary node that does not correspond to any moment tensor, 'V' ('Vector Value') means the part of the edge is plotted as magenta (it is contravariant index), 'T' ('The oTher index', i.e. covariant) is plotted green and 'A' means the edge part is not plotted 'At All'. So, A0 means that the edge does not connect any second or third node. E.g. the graph from Fig. 3 is described as

```

1    1  T1
V1  V2  T2
V3    3  T3
```

and the example from Fig. 4 is described as

```

T1  T2  3  V3  T4  V4    1
V2  T3  4    4    4  V1  A0.
```

## 2.3 Files with prescribed moment values

The name of the file includes the suffix “\_momval”. The first row is e.g. “dimensions=3, contravariant=2, covariant=2, maxorder=4”, where the numbers are the parameters of the invariants, namely 2D or 3D, the number of contravariant and covariant indices and the maximum moment order. The rows in the form e.g. “i0=1 i1=0 j0=1 j1=2” are here just for orientation. The 'i' are contravariant indices, the 'j' are covariant. In 'ik=v', the 'k' means  $k$ th contravariant index and 'v' is its value; 0 means x-component, 1 y-component and 2 z-component. The matrices in form

```

r  r  r  r  r  r
r  r  r  r  r  r
r  r  r  r  r  r
r  r  r  r  r  r
r  r  r  r  r  r
r  r  r  r  r  r
```

follows. The 'r' means random. The arrangement in 2D is

```

m00  m10  ...  md0
m01  m11  ...  md1
      :
m0d  m1d  ...  mdd,
```

where  $d$  is the maximum index value (maxorder). In 3D, it is

$$\begin{array}{cccc}
m_{000} & m_{100} & \cdots & m_{d00} \\
m_{010} & m_{110} & \cdots & m_{d10} \\
& & \vdots & \\
m_{0d0} & m_{1d0} & \cdots & m_{dd0} \\
& & & \\
m_{001} & m_{101} & \cdots & m_{d01} \\
m_{011} & m_{111} & \cdots & m_{d11} \\
& & \vdots & \\
m_{0d1} & m_{1d1} & \cdots & m_{dd1} \\
& & & \\
\vdots & & & \\
& & & \\
m_{00d} & m_{10d} & \cdots & m_{d0d} \\
m_{01d} & m_{11d} & \cdots & m_{d1d} \\
& & \vdots & \\
m_{0dd} & m_{1dd} & \cdots & m_{ddd}
\end{array}$$

The matrices for different value of the z-component are separated by an empty row. It is possible to substitute the letter 'r' by some prescribed value at the specific position, e.g.

$$\begin{array}{cccccc}
0 & 0 & 1 & r & r & r \\
0 & 0 & r & r & r & r \\
0 & r & r & r & r & r \\
r & r & r & r & r & r \\
r & r & r & r & r & r \\
r & r & r & r & r & r
\end{array}$$

The value 1 will be used as  $m_{20}$  and the zeros as other moments up to the second order in the dependence test. The values of the higher-order moments stay random.

### 3 Theory

An introduction to tensors can be found in [1] or in [2]. A good explanation can also be found in the English translation [3] of the Russian original [4].

#### 3.1 Tensors

Intuitively speaking, a tensor is an array of numbers and the rank of a tensor determines the dimensionality of this array. Special cases include scalars, which are tensors of rank zero, vectors, which are tensors of rank one, and matrices, which are tensors of rank two. Unlike usual arrays, the tensors have two types of indices, contravariant and covariant. They differ in behavior under affine transformation.

**Definition 1** A multidimensional array  $\mathbf{T}_{j_1 \dots j_m}^{i_1 \dots i_n}$  that behaves under an affine transformation by the invertible matrix  $\mathbf{A}_j^i \in \mathbb{R}^{d \times d}$  like

$$\mathbf{T}_{j_1 \dots j_m}^{i_1 \dots i_n} = |\det(\mathbf{A}^{-1})|^w \mathbf{A}_{k_1}^{i_1} \cdots \mathbf{A}_{k_n}^{i_n} (\mathbf{A}^{-1})_{j_1}^{l_1} \cdots (\mathbf{A}^{-1})_{j_m}^{l_m} \mathbf{T}_{l_1 \dots l_m}^{k_1 \dots k_n} \quad (4)$$

is called a (relative, axial) **tensor** of covariant rank  $m$ , contravariant rank  $n$ , and weight  $w$ . An (absolute) tensor has weight zero.

The basic operations acting on tensors, are addition, multiplication, and contraction. Only addition of tensors with the same rank and weight is possible.

**Definition 2** Let  $\mathbf{T}$  and  $\mathbf{T}'$  be two relative tensors of covariant rank  $m$ , contravariant rank  $n$ . Then, the sum  $\mathbf{T} + \mathbf{T}'$  is defined

$$(\mathbf{T} + \mathbf{T}')_{j_1 \dots j_m}^{i_1 \dots i_n} := \mathbf{T}_{j_1 \dots j_m}^{i_1 \dots i_n} + \mathbf{T}'_{j_1 \dots j_m}^{i_1 \dots i_n}. \quad (5)$$

The sum is a relative tensor of the same covariant rank  $m$ , contravariant rank  $n$ , and weight  $w$  as the terms. The multiplication can be performed with tensors of different rank and weight

**Definition 3** Let  $\mathbf{T}$  and  $\mathbf{T}'$  be two relative tensors of covariant rank  $m$ , contravariant rank  $n$ , and weight  $w$  and  $m', n', w'$  respectively. Then, the product  $\mathbf{T} \otimes \mathbf{T}'$  (also called outer product or tensor product)

$$(\mathbf{T} \otimes \mathbf{T}')_{j_1 \dots j_m j'_1 \dots j'_{m'}}^{i_1 \dots i_n i'_1 \dots i'_{n'}} := \mathbf{T}_{j_1 \dots j_m}^{i_1 \dots i_n} \mathbf{T}'_{j'_1 \dots j'_{m'}}^{i'_1 \dots i'_{n'}} \quad (6)$$

is a relative tensor of covariant rank  $m + m'$ , contravariant rank  $n + n'$ , and weight  $w + w'$ .

In other words, we multiply each component of the first tensor by each component of the second tensor.

The contraction is a sum over one index used twice, once as contravariant index, once as covariant index. The good example of the contraction is vector dot product, where both indices are used for the contraction, or matrix multiplication, where two from four indices are contracted. The result has two remaining indices. Formally

**Definition 4** Let  $\mathbf{T}$  be a relative tensor of covariant rank  $m$ , contravariant rank  $n$ , and weight  $w$ . Then, the contraction  $\sum_{(i_k, j_l)} \mathbf{T}$  of a covariant index  $i_k$  and a contravariant index  $j_l$

$$\left( \sum_{(i_k = j_l = \lambda)} \mathbf{T} \right)_{j_1 \dots j_{l-1} j_{l+1} \dots j_m}^{i_1 \dots i_{k-1} i_{k+1} \dots i_n} := \mathbf{T}_{j_1 \dots j_{l-1} \lambda j_{l+1} \dots j_m}^{i_1 \dots i_{k-1} \lambda i_{k+1} \dots i_n} \quad (7)$$

is a relative tensor of covariant rank  $m - 1$ , contravariant rank  $n - 1$ , and weight  $w$ .

The total contraction is performed over all indices. The contraction over one index causes direct and an inverse matrix of the affine transformation in Def. 1 are multiplied and canceled. The total contraction cancels all the matrices and we obtain a relative affine invariant that is multiplied by certain power of determinant of the transformation. Therefore, the total contraction can be used for the construction of affine invariants.

If we leave one index without contraction, we obtain a vector that behaves linearly under the linear transformation of coordinates. It can be used for normalization of tensor fields.

Besides tensors, we can have also tensor fields. It means we have defined the tensor in each point of a space.

**Definition 5** When a tensor is defined in each point of  $d$ -dimensional space and it behaves under an affine transformation like

$$\begin{aligned} \mathbf{T}_{j_1 \dots j_m}^{i_1 \dots i_n} ((x')^1 \dots (x')^d) &= |\det(\mathbf{A}^{-1})|^{w_A} |\det(\mathbf{B}^{-1})|^{w_B} \mathbf{A}_{k_1}^{i_1} \dots \mathbf{A}_{k_n}^{i_n} \\ &\quad (\mathbf{A}^{-1})_{j_1}^{l_1} \dots (\mathbf{A}^{-1})_{j_m}^{l_m} \mathbf{T}_{l_1 \dots l_m}^{k_1 \dots k_n} (x^1 \dots x^d) \end{aligned} \quad (8)$$

where  $((x')^1 \dots (x')^d)^T = \mathbf{B}^{-1} (x^1 \dots x^d)^T$

is called a (relative, axial) **tensor field** of covariant rank  $m$ , contravariant rank  $n$ , and weight  $w_A + w_B$ . The affine transformation with the matrix  $\mathbf{A}$  is called outer and the affine transformation with the matrix  $\mathbf{B}$  is called inner.

The tensor field in strict sense has ever  $\mathbf{A} = \mathbf{B}$ . We can imagine the case  $\mathbf{A} \neq \mathbf{B}$  as a color image, where one affine transformation deals with the space coordinates and the other affine transformation deals with the colors. Our software can work even with this special case.

## 3.2 Rotation

The linear transformation of coordinates can be described as

$$((x')^1 \dots (x')^d)^T = \mathbf{A} (x^1 \dots x^d)^T. \quad (9)$$

If there are no constraints on the matrix  $\mathbf{A}$  of the size  $d \times d$ , it is called affine transformation. If the matrix  $\mathbf{A} = \{a_{ij}\}$  is orthogonal, i.e.  $\sum_{i=1}^d a_{ij} a_{ik} = 0$  and  $\sum_{i=1}^d a_{ji} a_{ki} = 0$  for  $j \neq k$ , then the transformation is called orthogonal. If in addition  $\det(\mathbf{A}) = 1$ , then it is rotation, at least in 2D and 3D. Our software does not work with spaces of higher dimensions.

The inverse of orthogonal matrix equals its transposition, i.e.  $\mathbf{A}^{-1} = \mathbf{A}^T$ . It implies that then we need not distinguish contravariant and covariant indices. We can perform contraction over arbitrary two indices.

## 3.3 Moment Tensors

Geometric moments of real valued functions have been introduced to pattern recognition in [5]. The geometric moment of an image is defined as follows.

**Definition 6** For a scalar function  $f : \mathbb{R}^d \rightarrow \mathbb{R}$  with compact support, the **geometric moment**  $m_{p_1 \dots p_d}$  of order  $o = \sum_{i=1}^d p_i$  is

$$m_{p_1 \dots p_d} = \int_{\mathbb{R}^d} (x^1)^{p_1} \dots (x^d)^{p_d} f(x) \, d^d x, \quad (10)$$

where  $(x^i)^{p_i}$  refers to the  $p_i$ -th power of  $i$ th coordinate.

Dirilten and Newman suggest the use of moment tensors for the construction of moment invariants with respect to orthogonal transformations in [6]. They construct the moment tensors by arranging the moments of each order in a way such that they obey the tensor transformation property (4). The method is also well described in [7].

**Definition 7** For a scalar function  $f : \mathbb{R}^d \rightarrow \mathbb{R}$  with compact support, the **moment tensor**  ${}^o\mathbf{M}$  of order  $o \in \mathbb{N}$  takes the form

$${}^o\mathbf{M}^{k_1 \dots k_o} = \int_{\mathbb{R}^d} x^{k_1} \dots x^{k_o} f(x) d^d x, \quad (11)$$

with  $k_l \in \{1, \dots, d\}$ ,  $l \in \{1, \dots, o\}$  and  $x^{k_l}$  representing the  $k_l$ -th component of  $x \in \mathbb{R}^d$ .

The component  ${}^o\mathbf{M}^{k_1 \dots k_o}$  of a moment tensor equals the geometric moment  $m_{p_1 \dots p_d}$  iff the number of indices  $k_1, \dots, k_o$  equal  $i$  is  $p_i$  for  $i \in \{1, \dots, d\}$ .

Langbein and Hagen [8] have generalized the definition of the moment tensor to tensor valued functions.

**Definition 8** For a tensor field  $\mathbf{T} : \mathbb{R}^d \rightarrow \mathbb{R}^{d^n \times d^m}$  with compact support, the **moment tensor**  ${}^o\mathbf{M}$  of order  $o \in \mathbb{N}$  takes the form

$${}^o\mathbf{M}_{j_1 \dots j_m}^{k_1 \dots k_o i_1 \dots i_n} = \int_{\mathbb{R}^d} x^{k_1} \dots x^{k_o} \mathbf{T}_{j_1 \dots j_m}^{i_1 \dots i_n} (x^1 \dots x^d) d^d x. \quad (12)$$

Similarly to the scalar case, the component  ${}^o\mathbf{M}_{j_1 \dots j_m}^{k_1 \dots k_o i_1 \dots i_n}$  of a moment tensor equals the geometric moment

$$m_{p_1 \dots p_d(j_1 \dots j_m)}^{(i_1 \dots i_n)}$$

of the component  $\mathbf{T}_{j_1 \dots j_m}^{i_1 \dots i_n}$  of the tensor field iff  $p_l$  of the indices  $k_1, \dots, k_o$  equals  $l$  for all  $l = 1, \dots, d$ . The moment tensor of order  $o$  of a tensor field of covariant rank  $m$ , contravariant rank  $n$  and weight  $w$  is a tensor of covariant rank  $m$ , contravariant rank  $n + o$  and weight  $w - 1$ , as shown by Bujack and Hagen in [9].

The moment tensor  ${}^o\mathbf{M}_{j_1 \dots j_m}^{k_1 \dots k_o i_1 \dots i_n}$  has  $o$  coordinate indices,  $n$  contravariant indices, and  $m$  covariant indices, i.e.  $o + n$  upper and  $m$  lower indices. The upper left index  $o$  is written to distinguish the coordinate indices and the contravariant indices of the original tensor field.

It is often not possible to perform the total contraction directly because of the different number of covariant and contravariant indices of the tensor fields. Dirilten and Newman suggested already in [6] contractions with the permutation tensor  $\varepsilon$ .

**Definition 9** The component with indices  $1 \dots d$  of the **permutation tensor** equals one and also each component with an even permutation of the indices equal one. The components with the odd permutation of indices  $1 \dots d$  equal -1. If some index is repeated, then the sequence of the indices is not any permutation of  $1 \dots d$  and the corresponding component of the permutation tensor equals zero.

In the even permutation, we must swap an even number of indices to convert the current ordering of indices to  $1 \dots d$  or back; in the odd permutation, it is the odd number of swaps. In 2D, the permutation tensor  $\varepsilon_{ij}$  takes the form

$$\varepsilon = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} \quad (13)$$

In 3D, 132, 213 and 321 are odd permutations of 123, while 231 and 312 are even permutations. So, these 6 components equal 1 or -1, other 21 components equal 0.

The permutation tensor can be used as covariant  $\varepsilon_{1\dots d}$ , then it has contravariant rank zero, covariant rank  $d$  and weight  $-1$ . Then, we can perform it for contraction of the contravariant indices of the moment tensors. The permutation tensor can also be used as the contravariant  $\varepsilon^{i_1\dots i_d}$  with contravariant rank  $d$  and covariant rank zero. It can be used for contraction of the covariant indices.

We can use the permutation tensors for the total contraction of a tensor product of moment tensors of a tensor field with both contravariant and covariant ranks equaling one

$$I = {}^2M_\ell^{ijk} {}^1M_o^{mn} {}^0M_q^p \varepsilon_{ikn} \varepsilon_{jmp} \varepsilon^{\ell oq}. \quad (14)$$

According to Einstein's notation, the sum symbols over all indices from 1 to  $d$  are omitted.

### 3.4 Graph Method

In the following, we use graph theory to generate all possible contractions of the given moment tensor products. Every product contraction can be expressed by a graph, where each moment tensor corresponds to a node and each index corresponds to a connection edge-node. We use edges of several types. The basic type is direct contraction over an index used twice at two moment tensors. When the index is used once at a permutation tensor, we should use auxiliary node for the permutation tensor. These nodes are omitted and we use one edge for the whole permutation tensor, e.g. the contraction  ${}^1M^i {}^1M^j \varepsilon_{ij}$  is expressed by one edge. In 3D, it means we have triple hyperedges connecting three nodes.

The type of the edge is distinguished by color of the edge, we use black color for coordinate indices, magenta for contravariant indices and green for covariant indices. So, if we perform contraction over one coordinate and one covariant index, the edge color is changed in the middle of the edge. If there are some non-contracted indices, they can be expressed by special "half-edges" leading to nowhere.

An example of a graph corresponding to the contraction (14) is in Fig. 3.

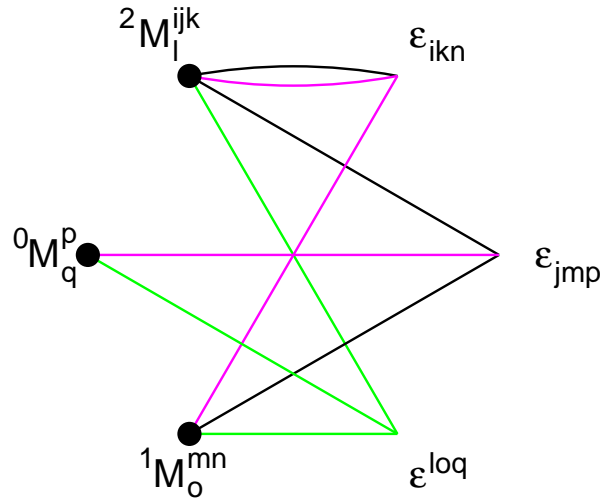


Figure 3: Graph example generating a partial invariant with the zero-rank contraction  ${}^2M_\ell^{ijk} {}^1M_o^{mn} {}^0M_q^p \varepsilon_{ikn} \varepsilon_{jmp} \varepsilon^{\ell oq}$  of a vector field.



Another example can be contraction

$$P^r = {}^3M_i^{ijkl} {}^1M_o^{mn} {}^0M_p^q {}^1M_q^{rs} \varepsilon_{jn} \varepsilon_{km} \varepsilon^{op} \varepsilon_{\ell s} \quad (15)$$

with graph in Fig. 4.

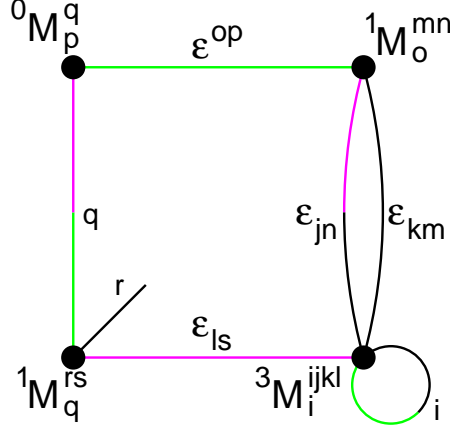


Figure 4: Graph example generating a partial invariant with the first-rank contraction  ${}^3M_i^{ijkl} {}^1M_o^{mn} {}^0M_p^q {}^1M_q^{rs} \varepsilon_{jn} \varepsilon_{km} \varepsilon^{op} \varepsilon_{\ell s}$  of a tensor field with one covariant index and one contravariant index.

### 3.4.1 Generation of the Graphs

The graphs are described by a list of edges. The main parameter is the number of edges  $n_e$ , it is given by a user, the number of nodes  $n_d$  is computed from the graph. The graph generation starts with the first graph

$$\begin{array}{ccccc} 1 & 1 & \cdots & 1 & 1 \\ 1 & 1 & \cdots & 1 & 1. \end{array}$$

Then, the edges are colored, the graph is evaluated and the next graph is generated. If the last graph is reached, the algorithm finishes. The coloring is based on the similar principle. If there is another possibility of the coloring, it is evaluated, otherwise next form of the edges is generated. The last graph is

$$\begin{array}{ccccc} 1 & 3 & \cdots & 2n_e - 3 & 2n_e - 1 \\ 2 & 4 & \cdots & 2n_e - 2 & 2n_e. \end{array}$$

In some cases, there are modifications. If we generate 3D affine invariants, triple hyperedges are generated from

$$\begin{array}{ccccc} 1 & 1 & \cdots & 1 & 1 \\ 1 & 1 & \cdots & 1 & 1 \\ 1 & 1 & \cdots & 1 & 1 \end{array}$$

to

$$\begin{array}{ccccc} 1 & 4 & \cdots & 2n_e - 5 & 2n_e - 2 \\ 2 & 5 & \cdots & 2n_e - 4 & 2n_e - 1 \\ 3 & 6 & \cdots & 2n_e - 3 & 2n_e \end{array}$$

for the rotations and to

1	2	$\dots$	$n_e - 1$	$n_e$
2	3	$\dots$	$n_e$	$n_e + 1$
3	4	$\dots$	$n_e + 1$	$n_e + 2$

for the affine transformation.

If partial invariants with  $r_i$  contravariant indices and  $o_o$  covariant indices are generated, then we need  $r_i + o_o$  half-edges. They are created so the last  $r_i + o_o$  edges is filled by 1 in the first row and zeros in other rows in the first graph.

The generation of the next graph is divided to two parts. First, the next half-edge is tried to generate. Only if it not possible, next standard edges and triple hyperedges are generated by the Algorithm 1.

---

**Algorithm 1** Generation of the next graph from the current one.

---

```

1: Row  $S \leftarrow$  last row
2: while Row  $\geq$  first row & no new graph found do
3:   Search Row of the edge list from behind.
4:   if Can we increase any node label? then
5:     Increase it to  $v_1$ .
6:     if Row  $\neq$  the first row then
7:       Fill the rest of Row with  $\max(v_1, a_i)$ .
8:        $a_i$  is the node label above it (in Row-1).
9:     else
10:      Fill the rest of Row with  $v_1$ .
11:    end if
12:    Fill the node labels below and rest of these rows with  $v_1$ .
13:  end if
14:  Decrease Row by one to the beginning.
15: end while
16: if Was any node label increased? then
17:   Return the new graph.
18: else
19:   Stop.
20: end if

```

---

The algorithm is illustrated in Fig. 5. On the position  $v_1$ , there was value  $v_1 - 1$ . It was the first value from the end that can be increased to  $v_1$ . The new value  $v_n$  is maximum from the values  $v_1$  and  $a_i$ . The value  $v_1$  fills also the right bottom corner of the graph.

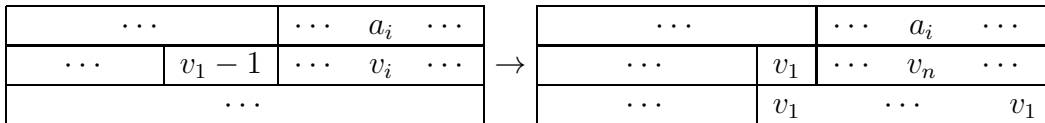


Figure 5: Filling the rest of the graph after increasing a node label. The new value  $v_n = \max\{v_1, a_i\}$  does not depend on the previous value  $v_i$ .

The algorithm for the half-edges is a loop that insert the half-edges successively to all nodes. Similarly we move the colors of the edges going from the individual nodes, so each node would have  $r_i$  magenta edges and  $r_o$  green edges. The remaining  $o$  black edges define the moment order.

### 3.5 Removing of Reducible Invariants

The graph method produces many dependent invariants that should be removed. We can distinguish a few types of dependencies. Some invariants are zero, there are pairs of the same invariants, some invariants are products of the others, some are linear combinations and there are also polynomial dependencies among the invariants. There is a standard procedure for elimination of linearly dependent invariants, see e.g. [10]. Recently, Langbein and Hagen [8] published a numerical test for elimination of dependent invariants that works also with the polynomial dependencies.

The zero invariants and the identical invariants are removed immediately after their generation. The removing of the products is based on multiplication of each invariant with each and search on the product in our list. If e.g. product of invariants  $I_{i_1}$  and  $I_{i_2}$  is an invariant  $I_{i_3}$  and we find it in our list, we know  $I_{i_3} = I_{i_1} \cdot I_{i_2}$ . If it is not in our list, we add it there. Finally, the products are removed, but before, they are used for the test of linear dependencies.

When some group of invariants have the same moment orders in each term (we call them the invariants with the same structure), they can be potentially linearly dependent. We compose a matrix of the coefficients; one coefficient from each term. The rank of this matrix equals the number of linearly independent (irreducible) invariants from this group. The other (reducible) invariants are finally removed together with the products. We use singular value decomposition for the rank computation and reduced row echelon form to find a basis of invariants.

When partial invariants with non-zero output rank are generated, the algorithm for multiplication must be slightly modified. While the full invariants can be multiplied freely and the result is always full invariant, the partial invariants cannot. During multiplication, the contraction ranks are added and we need to generate only the products with the same contraction ranks. In practice, we need to include all the invariants of lower ranks to the multiplication.

In some cases, especially in 3D, the number of the products is enormous and this algorithm is slow (sometimes slower than the invariant generation). Then it is possible to try to remove all dependencies together by the Langbein's test.

### 3.6 Elimination of Dependent Invariants

If some invariants  $I_1, I_2, \dots, I_n$  are dependent, it means there is a function  $f$  that  $f(I_1, I_2, \dots, I_n) = 0$ . Since the invariants are polynomials of the moments, there can be no other dependence than also polynomial, i.e. smooth. When the function  $f$  is smooth, we can differentiate this equation.

$$\frac{\partial f(I_1, I_2, \dots, I_n)}{\partial m_j} = \sum_{i=1}^n \frac{\partial f(I_1, I_2, \dots, I_n)}{\partial I_i} \frac{\partial I_i}{\partial m_j} = 0, \quad (16)$$

where  $m_j$  is some moment occurring in the invariants. It can be understood as a system of linear equations with known matrix  $\mathbf{S} = \{s_{ij}\} = \partial I_i / \partial m_j$  and the unknown vector  $\mathbf{b} = \{b_i\} = \partial f(I_1, I_2, \dots, I_n) / \partial I_i$ . If  $\mathbf{S}$  has the full rank, the system has only one solution full of zeros. The invariants are then independent, otherwise the rank of it equals the number of independent invariants.

Analytic solution in whole space would be too demanding, therefore we compute the system only at one point. We choose some random numbers with uniform distribution from 0 to 1 as values of the moments. We evaluate the derivatives for these moment values and compute a basis of independent invariants by the reduced row echelon form algorithm. The singular value decomposition is used here just for the check of the matrix rank.

We need not only to solve the equation  $\mathbf{S}\mathbf{b} = 0$ , but also to find the indices of the independent invariants. It is advantageous to use Gauss-Jordan elimination for conversion of the matrix  $\mathbf{S}$  to reduced row echelon form. The indices of the used pivots then equal the indices of the independent invariants. In some cases, we can choose an arbitrary invariant from a group of dependent invariants, therefore the invariants should be sorted in  $\mathbf{S}$  according to their complexity. The elimination then selects the set of independent invariants as simple as possible.

When we have partial invariant  $P^{(i)}$  with  $n_p$  components (i.e.  $i = 1, \dots, n_p$ ), we would obtain  $n_p$  such systems of linear equations. In this case, we concatenate their matrices to one  $\mathbf{S} = [\mathbf{S}^{(1)}, \dots, \mathbf{S}^{(n_p)}]$ . Then, we solve the system of linear equations

$$\mathbf{S}\mathbf{b}^T = [\mathbf{S}^{(1)}, \dots, \mathbf{S}^{(n_p)}] \begin{bmatrix} \mathbf{b}^{(1)} \\ \vdots \\ \mathbf{b}^{(n_p)} \end{bmatrix} = 0. \quad (17)$$

As a result, some invariants can be contained more than once in the parts of the solution corresponding to the different components. We must remove these duplicities and the number of independent partial invariants is then less than the rank of the matrix  $\mathbf{S}$ .

The test is numerical, therefore we choose randomly five sets of moment values and repeat the test five times. If the results of all five sets are the same, we consider it to be the correct result. If they differ, it usually means that default tolerances of non-zero singular values in rank computation and pivots in reduced row echelon form computation must be changed.

## 4 Conclusion

This software is intended for the generation of the invariants of the tensor fields. When we want to use these generated invariants, we need additional functions for reading the invariants in the txt files, moment computation and evaluation of the invariants. These functions are not included in this software, but they can be delivered separately.

Good luck with “Afinvtensors”!

## References

- [1] R.M. Bowen and C. Wang. *Introduction to Vectors and Tensors*. Dover books on mathematics. Dover Publications, 2008.
- [2] P. Grinfeld. *Introduction to Tensor Analysis and the Calculus of Moving Surfaces*. Springer New York, 2013.

- [3] Grigorii Borisovich Gurevich. *Foundations of the Theory of Algebraic Invariants*. Nordhoff, Groningen, The Netherlands, 1964.
- [4] Grigorii Borisovich Gurevich. *Osnovy teorii algebraicheskikh invariantov*. OGIZ, Moskva, The Union of Soviet Socialist Republics, 1937.
- [5] Ming-Kei Hu. Visual pattern recognition by moment invariants. *IRE Transactions on Information Theory*, 8(2):179–187, 1962.
- [6] H. Dirilten and T. G. Newman. Pattern matching under affine transformations. *IEEE Transactions on Computers*, 26(3):314–317, 1977.
- [7] David Cyganski and John A. Orr. Object recognition and orientation determination by tensor methods. In T. S. Huang, editor, *Advances in Computer Vision and Image Processing*, pages 101–144, Greenwich, Connecticut, USA, 1988. JAI Press.
- [8] Max Langbein and Hans Hagen. A generalization of moment invariants on 2D vector fields to tensor fields of arbitrary order and dimension. In *Proceedings of 5th International Symposium Advances in Visual Computing, ISVC'09, Part II*, volume 5876 of *Lecture Notes in Computer Science*, pages 1151–1160. Springer, 2009.
- [9] Roxana Bujack and Hans Hagen. Moment Invariants for Multi-Dimensional Data. In Evren Ozerslan, Thomas Schultz, and Ingrid Hotz, editors, *Modelling, Analysis, and Visualization of Anisotropy*, Mathematica and Visualization, Basel, 2017. Springer.
- [10] Jan Flusser, Tomáš Suk, and Barbara Zitová. *2D and 3D Image Analysis by Moments*. Wiley, Chichester, U.K., 2016.