# Hidden functional dependencies found by the technique of F–transform

**Jiří Kupka Iva Tomanová**

Centre of Excellence IT4Innovations - Division University of Ostrava - IRAFM,
30. dubna 22, 701 03 Ostrava, Czech Republic

**Abstract**

In this contribution we provide an application of the technique of F–transform. We demonstrate that by using a simple density-based preprocessing, the applicability of F–transform in data analysis can be significantly improved. Despite of the fact that our procedure is demonstrated by a well-known DBSCAN algorithm and the technique of F–transform, the ideas are general enough to be applied for other density-based clustering algorithms and regression techniques, respectively.

**Keywords**: Density-based cluster analysis, F–transform, fuzzy approximation, DBSCAN.

## 1. Introduction and motivation

A fuzzy transform is an approximation technique introduced by I. Perfilieva in [1]. This soft computing method has been recently used in many applications (e.g. in data analysis [2], image processing [3], [4], [5], partial differential equations [6], fuzzy control [7], time series processing [8] etc.). Additionally, in [2] the use of the F–transform in data analysis has been demonstrated. The use of this technique was twofold - first, it can be used to mine linguistically expressible associations between attributes and, secondly, it can be considered as some kind of regression analysis.

In this contribution we demonstrate that, in the latter case, the procedure can be further generalized. While, by using the original method, one can approximate any functional dependence with arbitrary precision (see Theorem 1), the same procedure can fail for a slightly more complex data set despite of the fact that some dependence between attributes can be easily seen. For a toy example, depicted on Figure 1, the data set might be represented by two linear maps, but cannot be represented by a single continuous function (e.g. by the dashed line). In this contribution we introduce a procedure which is capable to find arbitrary dependence expressible by finitely many continuous functions (see Theorem 2).

Several useful side effects of our simple procedure can be easily obtained. For instance, our preprocessing acts as a simple noise reduction. Moreover, parts of the data set which cannot be represented by a continuous function can be found and specified

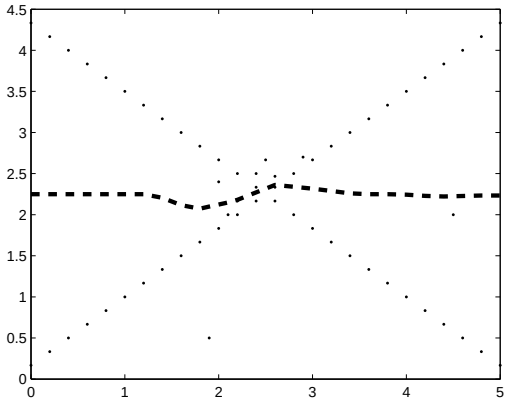


Figure 1: Experimental data 2 (dots).

and, finally, output of our procedure can be easily used as, say, a probabilistic classifier.

Within our procedure probably the best known density-based clustering technique (the DBSCAN algorithm) with the standard two dimensional Euclidean metric is used. Consequently, users of our procedure should be conscious of the well-known "curse of dimensionality", i.e. of the fact that results need not be so convincing for high-dimensional data sets. We used the DBSCAN algorithm and the F–transform technique just for demonstration purposes. Our simple ideas might be used for other density-based clustering and regression techniques.

The structure of this contribution is common: within the next section we recall some basic notions, then the procedure itself is in Section 3 followed by a simple theoretical result in Section 4 and illustrative examples in Section 5. As usual, the paper is finished by several concluding remarks.

## 2. Preliminaries

Below the set of natural numbers is denoted by $\mathbb{N}$, the set of real numbers is denoted by $\mathbb{R}$. Further, $I$ denotes the closed unit interval $[0, 1]$. We also need some basic notions from fuzzy mathematics. A *fuzzy set* $A$ on $[a, b] \subseteq \mathbb{R}$ (resp. universum, universal space) is a map $A : [a, b] \to I$. As a fuzzy set $A$ is defined as a map, all notions related to maps (such as continuity, uniform continuity, upper semicontinuity etc.) may be considered for fuzzy sets as

well. Further, for any $\alpha \in (0,1]$, an *$\alpha$-cut* $[A]_\alpha$ of $A$ is defined by

$$[A]_\alpha := \{x \in [a,b] \mid A(x) \geq \alpha\}.$$

It should be mentioned that if $A$ is upper semi-continuous then every $\alpha$-cut is a closed subset of $[a,b]$. A *support* $supp(A)$ of a given fuzzy set $A$ is usually defined as

$$supp(A) = \overline{\{x \in [a,b] \mid A(x) > 0\}}$$

where $\overline{\{\ldots\}}$ denotes a topological closure.

A data set $D$ is in the form of a table (see Table 1) consisting of two columns $X_1$ and $X_2$, where elements $f_i$ and $e_i$ are real numbers, i.e. $f_i, e_i \in [a,b]$ for every $i$. In this contribution $X_1$ represents an independent and $X_2$ represents the dependent variable. Occasionally, we will use the fact that $D$ can be considered as a set of points in $\mathbb{R}^2$. Then it makes sense to work with canonical projections $\pi_1, \pi_2$ defined by

$$\pi_1(z) = e_i,\ \pi_2(z) = f_i$$

for an $i$-th row $z \in D$ and $i \in \{1, 2, \ldots, N\}$.

| $X_1$ | $X_2$ |
|---|---|
| $e_1$ | $f_1$ |
| $e_2$ | $f_2$ |
| $\vdots$ | $\vdots$ |
| $e_N$ | $f_N$ |

Table 1: A data set $D$.

The best known algorithm based on the notion of density (namely, DBSCAN) was first suggested in [9]. Its main idea is very simple and natural. Roughly speaking, for a given metric on $\mathbb{R}^2$ (we consider the standard Euclidean metric $d_{\mathbb{R}^2}$) and $\varepsilon > 0$, this algorithm searches for objects $z := (e_i, f_i) \in D$ such that the number of its neighbors lying $\varepsilon$-close to $z$ is sufficient enough. Such points are then put to the same cluster if they are *mutually density-connected*, i.e. for any pair $z_{k_1}, z_{k_w} \in D$ there exists a finite sequence of points $\{z_{k_i}\}_{i=1}^{w} \in D$ such that $d_{\mathbb{R}^2}(z_{k_i}, z_{k_{i+1}}) < \varepsilon$ for any $i = 1, 2, \ldots, w-1$. Points of $D$, which do not have enough neighboring points, are marked as NOISE.

A nicely written pseudocode of DBSCAN can be found e.g. in [10]. In order to make this pseudocode more legible, we explain some abbreviations:

- *regionQuery* returns the points of $D$ contained in the $\varepsilon$–neighborhood of a point $p \in D$,
- $\varepsilon > 0$ specifies the maximal distance between two neighboring points,
- *min_points* $\in \mathbb{N}$ specifies how many neighbors must have any cluster point in its $\varepsilon$-neighborhood.

**algorithm DBSCAN**

```
DBSCAN(D, ε, min_points)
C = 0
for each unvisited point p ∈ D mark p as visited
NeighborPts = regionQuery(p, ε)
if size(NeighborPts) < min_points
   mark p as NOISE
   else
   C = next cluster
   expandCluster(p, NeighborPts, C, ε, min_points)
end

expandCluster(p, NeighborPts, C, ε, min_points)
add p to cluster C
for each point p' ∈ NeighborPts
if p' is not visited
   mark p' as visited
   NeighborPts' = regionQuery(p', ε)
end
if size(NeighborPts') ≥ min_points
   NeighborPts = NeighborPts ∪ NeighborPts'
end
if p' is not yet member of any cluster
   add p' to cluster C
end

regionQuery(p, ε)
return all points within p's ε-neighborhood
```

Let $a = x_0 < x_1 < \cdots < x_n = b,\ n \geq 2$, be fixed nodes within $[a,b]$ such that $x_k = x_0 + (k-1)h$, $k = 1, \ldots, n$, where the number $h = (b-a)/(n)$ specifies the distance between nodes. Fuzzy sets $A_0, \ldots, A_n : [a,b] \to [0,1]$ establish an *h-uniform fuzzy partition* of $[a,b]$ if they fulfill the following conditions:

1. for every $k = 0, \ldots, n$, $A_k(x) = 0$ if $x \in [a,b] \setminus [x_{k-1}, x_{k+1}]$ where $x_0 = x_{-1}$, $x_{n+1} = x_n$;
2. for every $k = 0, \ldots, n$, $A_k$ is continuous on $[x_{k-1}, x_{k+1}]$ where $x_0 = x_{-1}$, $x_{n+1} = x_n$;
3. for every $x \in [a,b]$, $\sum_{k=0}^{n} A_k(x) = 1$;
4. for every $k = 1, \ldots, n$, $A_k(x_k) = 1$;
5. for every $k = 1, \ldots, n-1$, $A_k$ is symmetrical with respect to the line $x = x_k$.

Let a function $f$ be given at finitely many points $p_1, \ldots, p_N \in [a,b]$ and $A_0, \ldots, A_n$ be basic functions which form a fuzzy partition of $[a,b]$. The following vector of real numbers

$$\mathbf{F}[f] = [F_0, \ldots, F_n] \tag{1}$$

is the *(direct) F–transform* of $f$ w.r.t. $A_0, \ldots, A_n$ where the $k-$th component $F_k$ is defined by

$$F_k = \frac{\sum_{i=1}^{N} A_k(p_i) f(p_i)}{\sum_{i=1}^{N} A_k(p_i)},\ k = 0, \ldots, n. \tag{2}$$

The F-tranform with respect to a fixed fuzzy partition is, in fact, a linear map from the space of continuous maps on $[a,b]$ to $\mathbb{R}^n$ satisfying

$$F'_n[\alpha f + \beta g] = \alpha F'_n[f] + \beta F'_n[G]$$

for any $\alpha, \beta \in \mathbb{R}$ and continuous maps $f, g$ on $[a, b]$. For more details and basic properties we again refer to [11]. Now we only recall that, under some assumptions, components of F–transform can be considered as weighted mean values of $f$ with weights given by the basic functions.

The inverse F–transform (with respect to a partition $A_0, \ldots, A_n$) represents the inverse process of $F'_n$. It takes an $n$-dimensional vector $v$ of real numbers and creates a linear combination of basic functions with coefficients given by $v$. In such a way a continuous map on the interval $[a, b]$ can be constructed. Thus, by using an inversion formula (i.e. an *inverse F–transform* $f_{F,n}(x)$)

$$f_{F,n}(x) = \sum_{k=0}^{n} F_k A_k(x),\ i = 1, \ldots, N,\ x \in [a, b]. \tag{3}$$

we can approximately reconstruct the function $f$ from the vector of components of its direct F–transform. The following theorem was shown in [11].

**Theorem 1** *Let $f(x) : [a, b] \to \mathbb{R}$ be a continuous function. For every $\varepsilon > 0$ there exists an integer $n(\varepsilon)$ and a related fuzzy partition $A_0, A_1, \ldots, A_{n(\varepsilon)}$ of $[a, b]$ such that the inverse F–transform $f_{F,n(\varepsilon)}$ of the function $f$ with respect to the partition $A_0, A_1, \ldots, A_{n(\varepsilon)}$ satisfies*

$$|f(x) - f_{F,n(\varepsilon)}(x) < \varepsilon|.$$

## 3. Algorithm

Before we present the main algorithm, some other notions should be introduced for any cluster $C$. We say that a cluster $C$ is *inappropriate* if there are two points $z_1, z_2 \in C$ and an interval $J_i$ such that are not density-connected on $C \cap (J_i \times \mathbb{R})$. Otherwise we say that the cluster is *appropriate*.

Finally, an *accuracy coefficient* can be specified for any interval $J_i$ and a cluster $C$. Let $\tilde{h}_i$ denotes the diameter of the set $\{\pi_2(z) \in \mathbb{R} \,|\, z \in (C \cap (J_i \times \mathbb{R}))\}$. Then this coefficient is defined as a ratio $c_a(J_i) := \tilde{h}_i / h$, which is still acceptable in further approximations.

**Steps of the algorithm:**

Let us describe the algorithm linguistically first:

1. INPUT: ($D$, *min_points*, $\varepsilon$, $h$), where
   (a) $D$ is a data set,
   (b) *min_points* is an integer which denotes minimal number of points in a particular cluster,
   (c) $\varepsilon > 0$ is a number representing the required precision,
   (d) $h$ specifies the $h$-uniform fuzzy partition of the interval $[a, b]$.
2. The algorithm DBSCAN is used to determine the number of clusters in $D$ (Let $K \in \mathbb{N}$ be this number and $C[i]$ denote $i$-th cluster, $i = 1, 2, \ldots, K$.) and the set $\mathcal{C}$ of clusters in $D$.
3. For every $i \in \{1, 2, \ldots, K\}$, the appropriateness of the cluster $C[i]$ is checked by the algorithm *AppChecking* and two sets of clusters are constructed - namely the set $AC$ of appropriate clusters and its complement $NC := \mathcal{C} \setminus AC$. In this step, the algorithm *AppChecking* is used. (Without loss of generality, we may assume that there exists $K' \leq K$ such that $NC$ consists of the first $K'$ clusters.)
4. For every cluster $C[i] \in NC$, i.e. $i = 1, 2, \ldots, K'$, the algorithm *FindSubclusters* finds a finite number $k_i \in \mathbb{N}$ of appropriate subclusters in the cluster $C[i]$. Let, for every $i$ as above, $C_j[i]$ denote such clusters for $j = 1, 2, \ldots, k_i$.
5. For every cluster $C[i]$, $i = K' + 1, \ldots, K$, and for every subcluster $C_j[i]$, for $i = 1, 2, \ldots, K'$, $j = 1, 2, \ldots, k_i$, approximations by using F–transform are calculated by the algorithm *FTransform*.
6. OUTPUT: a finite number of $\varepsilon$-approximations of $D$.

**Algorithm**

**Input:** $D$, *min_points*, $\varepsilon$, $h$

**1.** *DBSCAN* ($D$, *min_points*, $\varepsilon$)
specifies $K$ clusters of $D$.

**2.** For each cluster $C[i] \in \mathcal{C}$,
*AppChecking($C_i$)* checks the appropriateness
of $C[i]$, the sets $AC$ and $NC$ are specified.

**3.** For every cluster $C[i] \in NC$,
*FindSubclusters* ($C[i]$) finds the set $ASC_i$
of all appropriate subclusters of $C[i]$.

**4.** For any $C := C[i] \in AC$ and
for any $C \in ASC_i$ of $C \in NC$,
*(i)* check the accuracy of $C$,
(if necessary, specify a part of $D$ which cannot be approximated by a continuous function (see Figure 7))
modify $C$ if necessary,
*(ii)* approximate $C$ by the inverse F–transform;
*FTransformsCLUSTER* ($C[i]$)

**Outputs:**
*(i)* appropriate parts (clusters) of $D$
approximated by F–transform,
*(ii)* noise removement,
*(iii)* parts of $D$ without functional
dependence are detected.

**Algorithm AppChecking**
For any cluster $C[i]$ the set $J[i] := \{J_i \mid J_i \cap \pi_1(z) \neq \emptyset$ for some $z \in C[i]\}$ is nonempty and the union of its elements forms a closed subinterval of $[a, b]$ by the choice of $\varepsilon$. Thus, if $J[i]$ consists of $s$ intervals $J_i$, then there exists a node $x_j$ such that

$$[x_j, x_{j+s}] = \bigcup_{J_i \in J[i]} J_i. \tag{4}$$

Then for any $k = j, \ldots, j+s-1$ the number $l_k \in \mathbb{N}$ of clusters of $C[i]$ over $J_k$ is calculated by the algorithm DBSCAN$(C[i] \cap (J_k \times \mathbb{R}), min_points, \varepsilon)$. Then the following decision is made:
If $\max\{l_k | k = j, \ldots, j+s-1\} > 1$ then
$C[i] \in NC$ and $i = i+1$
else $C[i] \in AC$ and $i = i+1$.

**Algorithm FindSubclusters**
For an inappropriate cluster $C[i]$ we can use the notation introduced above - namely, for $C \in NC$ there exists an interval $[x_j, x_{j+s}]$ by (4). Furthermore, for any $k = j, \ldots, j+s-1$ there exist $l_k$ clusters of $C[i]$ over $J_k$. We use the following notation - $C_{k,l}$ denotes the $l$-th cluster of $C[i]$ over the interval $J_k$.

It is easy to see that there exists an oriented graph representation of the cluster $C[i]$ where the oriented graph $G_{C[i]}$ is defined by:

(i) $\tilde{C}_{k,l}$ denotes a vertex of $G_{C[i]}$ representing the $l$-th cluster of $C[i]$ over the interval $J_k$,
(ii) if there exist $z \in C_{k,l}, z' \in C_{k+1,l'}$ so that $d_{\mathbb{R}^2}(z, z') < \varepsilon$, there exists an oriented arrow $\tilde{C}_{k,l} \to \tilde{C}_{k+1,l'}$. Thus the oriented arrows specify whether or not the subclusters $C_{k,l}, C_{k+1,l'}$ are neighboring.

By using this graph notation, the rest of this algorithm searches for all maximal oriented paths in $G_{C[i]}$. Just to recall, an oriented path of $G_{C[i]}$ is a sequence of vertices $\{\tilde{C}_{t,j}, \tilde{C}_{t+1,j+1}, \ldots, \tilde{C}_{t+k,t+k}\}$ such that there exists an oriented arrow from $\tilde{C}_{t+q,t+q}$ to $\tilde{C}_{t+(q+1),t+(q+1)}$ for every $q = 0, 1, \ldots, k-1$. A *maximal oriented path* is an oriented path which is not a proper subset of another oriented path.

Directly from the definition of the graph $G_{C[i]}$ it can be seen that every maximal oriented path $P = \{\tilde{C}_{t,j}, \tilde{C}_{t+1,j+1}, \ldots, \tilde{C}_{t+k,t+k}\}$ represents a subcluster $C \in ASC_i$ of $C[i]$ defined by

$$C := \bigcup_{\tilde{C}_{t,j} \in P} C_{t,j},$$

and consisting of mutually density-connected points. The output of this algorithm is the set $ASC_i$ of subclusters of $C[i]$.

**Algorithm FTransformsCLUSTER**
Let either $C := C[i] \in AC$ or $C \in ASC_i$ for some $i = 1, 2, \ldots, K'$. As above, $C$ lies over the interval $L := [x_j, x_{j+s}]$ defined by (4) and $C_k$ denotes the subcluster of $C$ over the interval $J_k$.

(i) For any $J_k$, $k = 1, 2, \ldots, j+s-1$, the accuracy of $C_k$ is calculated by

$$c_a(J_k) = \frac{\max_{z \in C_k}\{\pi_2(z)\} - \min_{z \in C_k}\{\pi_2(z)\}}{h}.$$

If $c_a(J_k)$ is greater than some user-specified threshold then $C := C \backslash C_k$ and $L := L \backslash int(J_k)$, where $int(J_k)$ means the interior of the interval.
(ii) The F–transform can be computed for the $h$-uniform fuzzy partition restricted to $L$ and points from $C$. Without loss of generality we may assume that $L$ is still a closed interval. Then, the vector of components $F = [F_j, F_{j+1}, \ldots, F_{j+s}]$ is computed for each subinterval according to (2). By the inverse F–transform (see (3)) an approximate value $(F_{f,n}(e_i))$ of each point $e_i$ can be computed.
OUTPUT: Approximations by F–transform of relevant points $e_i \in D$.

## 4. Results

In this section we show a result demonstrating that our procedure extends applicability of the technique of F–transform in data analysis. Before stating this result we describe outputs of our procedure. We would like to emphasize that neither crisp partition $\{J_k\}$ nor fuzzy partition $\{A_k\}$ of the interval $[a, b]$ is changed within our procedure. Therefore, although F–transforms are calculated for finitely many clusters, they are always calculated over the same fuzzy sets of the same fuzzy partition (possibly restricted to some subinterval) of $[a, b]$.

Consequently, because the ordinary F–transform is represented as $n$-dimensional vector (1), the result of our procedure can be written as

$$\tilde{\mathbf{F}}[f] = [\tilde{F}_0, \ldots, \tilde{F}_n], \tag{5}$$

where $\tilde{F}_i$ is a finite number of $i$-th components $F_i$ of F–transforms calculated for some appropriate clusters or subclusters of $D$.

**Remark 1** *It is obvious that this output may include some additional information about the data set $D$ - for instance, a relative cardinality of subclusters over the interval $J_i$ etc.*

**Theorem 2** *Let $D$ be a subset of $\mathbb{R}^2$ which can be expressed as the union of finitely many continuous functions $f_j : [a, b] \to [a, b]$. Then for every $\varepsilon > 0$ there exists an integer $n(\varepsilon)$, a related fuzzy partition $A_0, A_1, \ldots, A_{n(\varepsilon)}$ of $[a, b]$ and such that the inverse F–transforms of F–transforms given by (5) are $\varepsilon$-close to the graphs of $f_j$'s*[1].

[1] We realize that this statement is not as mathematically precise as it should be. However, the idea and all the constructions are so straightforward that we expect that we can afford it in this contribution.

**Proof.** The sketch of this proof is natural and follows the procedure from Section 3. Because graphs of continuous functions are topologically connected in $\mathbb{R}^2$, the whole graph of each function $f_j$ is always (i.e. for any $\varepsilon > 0$) found as a part of some cluster by the algorithm DBSCAN. Consequently, if the number of found clusters is strictly smaller than the number of graphs of $f_j$'s, then clearly two (or more) graphs have a nonempty intersection.

This feature is checked by the algorithm *AppChecking*. Then, for all inappropriate clusters $C$'s their oriented graphs $G_C$'s can be constructed and maximal oriented paths can be found by the algorithm *FindSubclusters*. Without loss of generality we can assume that maximal oriented paths coincide with graphs of $f_j$'s. Because there is no need to check the accuracy of graphs of $f_j$'s, the algorithm *FTransformsCLUSTER* finds approximations which are $\varepsilon$-close to graphs of relevant functions $f_j$'s. □

## 5. Illustrative examples

Two short demonstrations of our approach are prepared in this section.

The experimental data and their approximation by the F–transform are shown in Fig. 2. The inverse F–transform is highlighted as the dashed line.

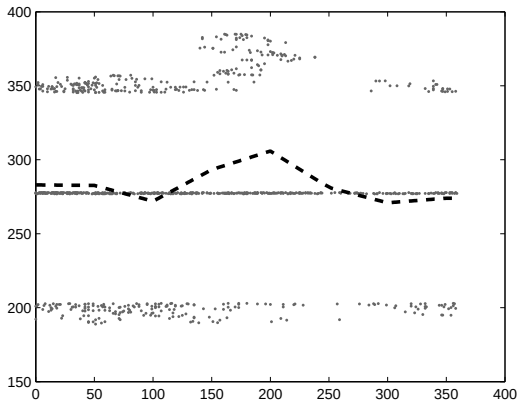


Figure 2: Experimental data 1.

By using the procedure from Section 3 with parameters $min\_points = 30$, $\varepsilon = 50$ and $h = 10$ we obtain approximations represented in Fig. 3. Here, particular clusters are denoted by different marks (stars etc.). All the clusters are appropriate and can then approximated by inverse F–transforms. All approximations are highlighted by dashed lines. Remaining points marked as circles denote a noise.

In Figure 4 one can see another experimental data set that is approximated by the "ordinary" F–transform represented by the dashed line. By using our procedure ($min\_points = 3$, $\varepsilon = 1$ and $h = 0,65$) just one inappropriate cluster $\tilde{C}$ is obtained.

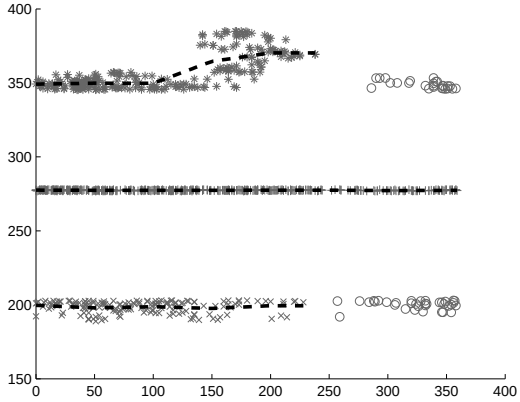


Figure 3: Approximation of Experimental data.

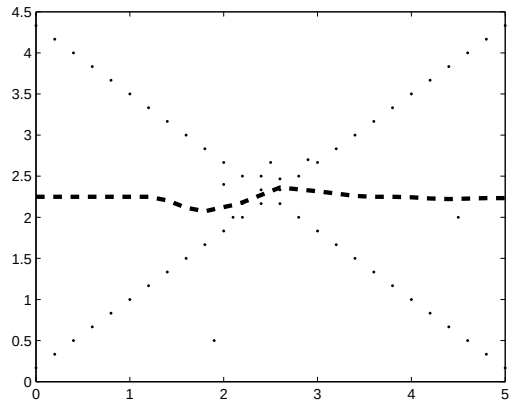


Figure 4: Experimental data 2.

Consequently, the proposed algorithm provides an oriented graph approximation of the cluster $\tilde{C}$ shown in Figure 5. Then two subclusters (i.e. two maximal oriented paths of the graph in Figure 5) of the cluster $\tilde{C}$ are found and approximated by dashed lines in Figure 6. The noise is again denoted by circles.

Finally, just to illustrate that fact, in Figure 7 one can see an example of a single cluster for which the subcluster over the interval $(143, 192)$ is not suitable for any approximation for some accuracy coefficient.

## 6. Conclusion

Within this contribution we presented a simple procedure expanding applicability of the technique of F–transform in data analysis. It is known that the original technique can approximate any continuous function with arbitrary precision. Our procedure acts as a preprocessing step as well as automatically provides several advanced features. Namely, our

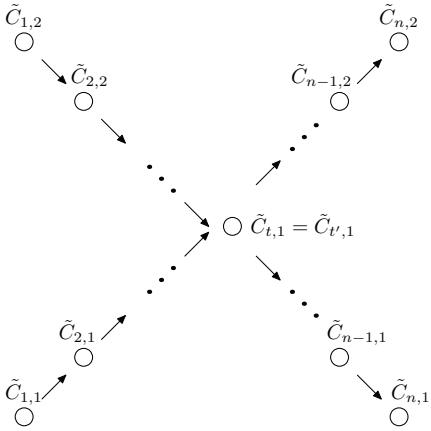


Figure 5: Graph representation of Experimental data 2.

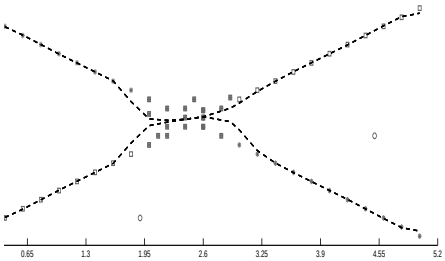


Figure 6: Approximation of Experimental data 2.

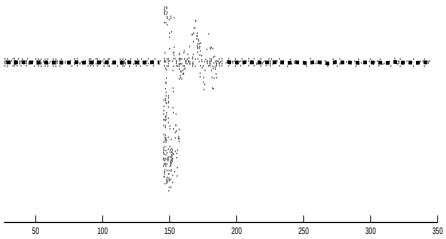


Figure 7: Cluster which cannot be well approximated by a single function.

procedure provides a fast outliers detection (resp. denoising) and also detects functional dependencies which can be represented by a finitely many continuous functions, and finally, such functional dependencies can be represented in the form which allows simpler classification than ordinary regression methods. Linguistic expressions of found results of the form mentioned in [11] are also available. Just for completeness, due to the compactness of the product space it does not make any sense to consider data sets represented by infinitely many continuous functions.

As we highlighted in the introduction of this contribution, the users should be careful about using this method for high-dimensional data sets due to the dimensionality problem. And we recall once more - we demonstrate this procedure for the most known density-based clustering algorithm and the technique of F–transform, but our ideas can be easily used for other density-based clustering and regression techniques. Our future research in this topic will lead especially to classification tasks for high-dimensional data sets.

## Acknowledgements

This work was supported by the European Regional Development Fund in the IT4Innovations Centre of Excellence project (CZ.1.05/1.1.00/02.0070). Furthermore, we gratefully acknowledge partial support of project SGS06/Přf/2013.

## References

[1] I. Perfilieva and E. Chaldeeva. Fuzzy transformation and its applications. pages 116–124, 2001.

[2] I. Perfilieva, V. Novák, and A. Dvořák. Fuzzy transform in the analysis of data. *International Journal of Approximate Reasoning*, 48(1):36–46, 2008.

[3] Martina Daňková and Radek Valášek. Full fuzzy transform and the problem of image fusion. *Journal of Electrical Engineering*, 12:82–84, 2006.

[4] I. Perfilieva, M. Daňková, P. Hoďáková, and M. Vajgl. F-transform based image fusion. In Osamu Ukimura, editor, *Image Fusion*, pages 3–22. InTech, 2011.

[5] M. Vajgl, I. Perfilieva, and P. Hoďáková. Advanced f-transform-based image fusion. *Advances in Fuzzy Systems*, 2012.

[6] M. Štěpnička and R. Valášek. Numerical solution of partial differential equations with help of fuzzy transform. In *Proc. FUZZ-IEEE2005 The international Conference on FUZZY Systems*, pages 1104–1109, Reno, Nevada, 2005.

[7] I. Perfilieva and V Kreinovich. Probabilistic interpretation of fuzzy transforms and fuzzy control. *Departmental Technical Reports (CS)*, 44, 2009.

[8] I. Perfilieva, V. Novák, V. Pavliska, A. Dvořák, and M. Štěpnička. Analysis and prediction of time series using fuzzy transform. In *Proc. of WCCI 2008, IEEE Int. Conf. on Neural Networks*, pages 3875–3879, Hong Kong, 2008.

[9] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96)*, pages 226–231.

[10] Dbscan. http://en.wikipedia.org/wiki/DBSCAN, 2013.

[11] I. Perfilieva. Fuzzy transforms: Theory and applications. *Fuzzy Sets and Systems*, 157(8):993–1023, 2006.