Monotone Centroid Flow Algorithm for Type-Reduction of General Type-2 Fuzzy Sets

Ondrej Linda, *Student Member, IEEE*, Milos Manic, *Senior Member, IEEE*

Abstract— Recently, Type-2 Fuzzy Logic Systems (T2 FLSs) received increased research attention due to their potential to model and cope with the dynamic uncertainties ubiquitous in many engineering applications. However, because of the complex nature and the computational intensity of the inference process, only the constrained version of T2 FLSs, the Interval T2 FLSs, were typically used. Fortunately, the very recently introduced concepts of α *-planes* and *zSlices* allow for efficient representation as well as computationally fast inference process with General T2 (GT2) FLSs. This paper addresses the type-reduction phase in GT2 FLSs, using GT2 Fuzzy Sets (FSs) represented in the α *-planes* framework. The monotone property of centroids of a set of α *-planes* is derived and leveraged towards developing a simple to implement, but fast algorithm for type-reduction of GT2 FSs – the Monotone Centroid Flow (MCF) algorithm. When compared to the Centroid Flow (CF) algorithm previously developed by Zhai and Mendel, the MCF algorithm features the following advantages: 1) the MCF algorithm computes numerically identical centroid as the Karnik-Mendel (KM) iterative algorithms, unlike the approximated centroid obtained with CF algorithm, 2) the MCF algorithm is faster than the CF algorithm as well as the independent application of the KM algorithms, 3) the MCF algorithm is simple to implement, unlike the CF algorithm, which requires computation of the derivatives of the centroid, 4) the MCF algorithm completely eliminates the need to apply the KM iterative procedure to any α *-planes* of the GT2 FS. The performance of the algorithm is presented on benchmark problems and compared to the other type-reduction techniques available in literature.

***Index Terms*— General Type-2 Fuzzy Sets, Type-reduction, Centroid, α *-planes* Representation**

I. Introduction

TYPE-2 Fuzzy Logic Systems (T2 FLSs) become the scope of work for many researchers in recent years [1] –[5]. The concept of T2 Fuzzy Sets (FSs) was originally proposed by Zadeh [6]. Since then, T2 FLSs have been successfully applied in many engineering areas, demonstrating the ability of T2 FLSs to perform better than T1 FLSs when confronted with various sources of dynamic uncertainties, frequently associated with real world engineering problems [7]-[13]. Unlike the T1 FLSs, the T2 FLSs use individual fuzzy sets with membership grades that are themselves fuzzy sets. This additional dimension of uncertainty in T2 FLSs provides additional degrees of freedom for modeling and coping with dynamic input uncertainties when compared to T1 FLS with equivalent number of fuzzy sets.

However, the early representations of General T2 (GT2) FSs, namely the *vertical-slice* and the *wavy-slice* representations, did not provide inference algorithms that were fast enough for real-time processing. This was primarily due to the immense computational complexity of the model of individual fuzzy sets as well as the inference process itself. In particular, the type-reduction phase required enumeration of an astronomically large number of embedded fuzzy sets, impractical even in case of large discretization levels [14]. Due to its complexity the type-reduction used to be one of the major limiting factors of applications of GT2 FSs and GT2 FLSs. For these reasons, the most typically used variant of T2 FLSs were the Interval T2 (IT2) FLSs, which use constrained secondary membership functions [15]. The IT2 FLSs offer a compromise between the computational inexpensiveness of T1 FLSs and the uncertainty modeling capability of GT2 FLSs.

Fortunately, the very recently introduced representations of α *-planes* [16], [17] and *zSlices* [18] offer a computationally efficient and viable framework for representing GT2 FSs as well as for the inferencing process with GT2 FLSs. In both cases, the α *-planes* and the *zSlices* representation theorems allow for treating of the GT2 FSs as a composition of multiple IT2 FSs, each raised to the respective level of α or *z*. Hence, the operations on GT2 FSs become a multiple application of the computational efficient arithmetic of IT2 FLSs. It should be noted here that both representations of α *-planes* and *zSlices* are very similar concepts, developed independently at the same time [19]. In the sequel of this paper the notation of the α *-planes* representation is followed.

This paper focuses on the type-reduction of GT2 FSs represented using the α *-planes* framework. Recently, the independent type-reduction of each α *-plane* was suggested in [16]. Such approach uses the established tools of IT2 FLSs, namely the Karnik-Mendel (KM) iterative algorithm [14] [20], and considers the entire GT2 FS as a composition of independent α *-planes*. The first authors to recognize the dependency of neighboring α *-planes* were Zhai and Mendel with their Centroid Flow (CF) algorithm [21], [22]. The CF algorithm starts by applying the KM algorithm to the lowest α *-planes* in order to compute the base of the type-reduced centroid. Next, the derivatives of the secondary membership

Ondrej Linda is with the Computer Science Department, University of Idaho, Idaho Falls, ID 83402 USA phone: 208-227-3919; e-mail: olinda@uidaho.edu.
Dr. Milos Manic is with the Computer Science Department, University of Idaho, Idaho Falls, ID 83402 USA

functions are used to propagate the centroid of one α *-plane* to the next one. Despite the elimination of the need to apply the iterative KM algorithm to each α *-plane* independently, which made the CF algorithm substantially faster, several deficiencies of this approach must be recognized: 1) the CF algorithm provides only an approximate result when compared to the baseline centroid computed using the KM algorithm, 2) the CF algorithm requires computation of the derivatives of all secondary membership functions, which might be considered as potential computational bottleneck.

This paper proposes the Monotone Centroid Flow (MCF) algorithm for computation of the centroid of GT2 FSs represented by α *-planes*. The algorithm is based on the monotone property of the α *-plane* representation and the centroid of the GT2 FS. The primary contributions of the presented MCF algorithm are as follows: 1) the MCF algorithm computes numerically identical centroid as the KM algorithms, unlike the approximated centroid obtained with the CF algorithm, 2) the MCF algorithm is faster than both the CF algorithm and the independent application of the KM or the Enhanced KM (EKM) algorithms, 3) the MCF algorithm is simple to implement, not requiring computation of the derivatives of the centroid and secondary membership functions, 4) the MCF algorithm completely eliminates the need to apply the KM iterative procedure to any α *-planes* of the GT2 FS. The accuracy and the computational time of MCF algorithm was demonstrated on several benchmark problems and compared to the CF algorithm and the independent application of the KM and the EKM algorithms.

The rest of the paper is organized as follows. Section II discusses the background of GT2 FSs and their α *-plane* representation. The type-reduction of GT2 FSs is reviewed in detail in Section III. Section IV first derives the monotone property of the α *-plane* representation and then uses this property to introduce the MCF algorithm. The experimental results and comparisons are presented in Section V. Finally, the paper is concluded in Section VI.

II. General Type-2 Fuzzy Sets

This section provides background overview of GT2 FSs and the fundamentals of their α *-plane* representation.

A. General Type-2 Fuzzy Sets

A GT2 FS $\tilde{A}$ can be expressed on the universe of discourse *X* using its Type-2 fuzzy membership function $\mu_{\tilde{A}}(x,u)$, where $x \in X$ and $u \in J_x$ [1]:

$$\tilde{A} = \int_{x \in X} \int_{u \in J_x} \mu_{\tilde{A}}(x,u)/(x,u) \quad J_x \subseteq [0,1] \tag{1}$$

Here, variable *x* and *u* are the primary and the secondary variables and J_x denotes the support of the secondary membership function also called the primary membership of *x*. Operator $\int\int$ denotes union over all possible values of *x* and *u*, and $\mu_{\tilde{A}}(x,u) \in [0,1]$. Two representations of GT2 FSs are commonly adopted; the *vertical-slice* representation and the *wavy-slice* representations.

First, the *vertical-slice* representation is considered. By instantiating a specific value for $x = x'$, a vertical slice $\mu_{\tilde{A}}(x',u)$ of the fuzzy membership function $\mu_{\tilde{A}}(x,u)$ can be obtained. This vertical slice defines a secondary membership function $\mu_{\tilde{A}}(x = x',u)$ for $x' \in X$ and $\forall u \in J_{x'} \subseteq [0,1]$:

$$\mu_{\tilde{A}}(x = x',u) \equiv \mu_{\tilde{A}}(x') = \int_{u \in J_{x'}} f_{x'}(u)/u \quad J_{x'} \subseteq [0,1] \tag{2}$$

Here, $f_{x'}(u)$ denotes the secondary grade or the amplitude of the secondary membership function and $f_{x'}(u) \in [0,1]$. Assuming that the domain of primary variable *x* is discretized using *N* samples, the GT2 FS $\tilde{A}$ can be represented as a composition of all its vertical slices:

$$\tilde{A} = \sum_{i=1}^{N} \left[\int_{u \in J_{x_i}} f_{x_i}(u)/u \right] \Big/ x_i \tag{3}$$

Next, consider the *wavy-slice* representation. The GT2 FS $\tilde{A}$ can be also constructed as a composition of embedded FSs $\tilde{A}_e$. Again, for a discrete universe of discourse with *N* elements, the embedded FS $\tilde{A}_e$ can be described as:

$$\tilde{A}_e = \sum_{i=1}^{N} \left[f_{x_i}(\theta_i)/\theta_i \right] \Big/ x_i \quad \theta_i \in J_{x_i} \subseteq U \in [0,1] \tag{4}$$

According to the Mendel and John representation theorem [23], the GT2 FS $\tilde{A}$ can be described as a union of all of its *n* embedded T2 FSs:

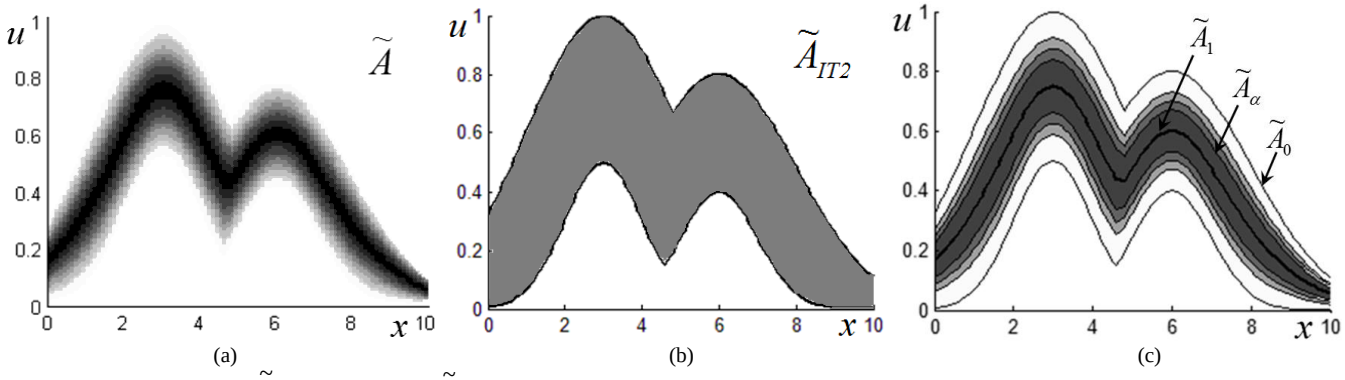
$$\tilde{A} = \bigcup_{j=1}^{n} \tilde{A}_e^j \tag{5}$$

For the discretized primary domain *X*, the centroid $C_{\tilde{A}}$ of a GT2 FS $\tilde{A}$ can be calculated using the *Extension Principle* and by enumerating all of the embedded fuzzy sets [6]:

$$C_{\tilde{A}} = \int_{\theta_1 \in J_{x_1}} \cdots \int_{\theta_N \in J_{x_N}} \left[f_{x_1}(\theta_1) \wedge \ldots \wedge f_{x_N}(\theta_N) \right] \Bigg/ \frac{\sum_{i=1}^{N} x_i \theta_i}{\sum_{i=1}^{N} \theta_i} \tag{6}$$

Here, every possible combination of variables $\theta_1, \ldots, \theta_N$ forms an embedded FS, which has a secondary grade of $f_{x_1}(\theta_1) \wedge \ldots \wedge f_{x_N}(\theta_N)$. Operator $\wedge$ is the specific t-norm used (e.g. the minimum operator).

Assuming that each primary membership J_{x_i} was discretized into M_i points, the number of embedded fuzzy sets that have to be enumerated in (6) is $n = \prod_{i=1}^{N} M_i$. Already for small number of discretization steps, *n* reaches inadmissibly large values.



(a) (b) (c)

Fig. 1 General T2 fuzzy set $\tilde{A}$ (a), its IT2 variant $\tilde{A}_{IT2}$ (b) and its α -plane representation (c). (The shade of grade illustratively depicts the distribution of secondary grades).

The crisp output value *y* can be obtained by applying one of the available defuzzification methods to the type-reduced centroid $C_{\tilde{A}}$. As an example, the centroid defuzzifier is commonly used [1]:

$$y = \frac{\sum_{i=1}^{n} x_i C_{\tilde{A}}(x_i)}{\sum_{i=1}^{n} C_{\tilde{A}}(x_i)} \tag{7}$$

B. α -Plane Representation for GT2 Fuzzy Sets

The following notation for the α *-plane* representation was adopted from [17], [21]. The α *-plane* representation was originally developed by several authors [16], [18], [24]. It constitutes a horizontal slice representation for GT2 FSs.

An α *-plane* $\tilde{A}_\alpha$ of a GT2 Fs $\tilde{A}$ can be defined as the union of all primary memberships of $\tilde{A}$ with secondary grades that are greater than or equal to α :

$$\tilde{A}_\alpha = \int_{\forall x \in X} \int_{\forall u \in J_x} \{(x,u) | f_x(u) \geq \alpha\} \tag{8}$$

An α -cut of the secondary membership function $\mu_{\tilde{A}}(x)$ can be denoted as $S_{\tilde{A}}(x|\alpha)$ and expressed as:

$$S_{\tilde{A}}(x|\alpha) = [s_L(x|\alpha), s_R(x|\alpha)] \tag{9}$$

Hence, an α *-plane* $\tilde{A}_\alpha$ can be seen as a composition of all individual α -cuts of all of its secondary membership functions:

$$\tilde{A}_\alpha = \int_{\forall x \in X} S_{\tilde{A}}(x|\alpha) = \int_{\forall x \in X} \left(\int_{\forall u \in [s_L(x|\alpha), s_R(x|\alpha)]} u \right) \Big/ x \tag{10}$$

It is apparent that the well known Footprint of Uncertainty (FOU) of the GT2 FS $\tilde{A}$ can now be denoted as:

$$FOU(\tilde{A}) = \tilde{A}_0 \tag{11}$$

Each α *-plane* $\tilde{A}_\alpha$ is bounded from above by its upper membership function $\overline{\mu}_{\tilde{A}}(x|\alpha)$ and from below by its lower membership function $\underline{\mu}_{\tilde{A}}(x|\alpha)$. Using the α -cuts boundaries of each vertical slice (9), these bounds can be expressed as:

$$\overline{\mu}_{\tilde{A}}(x|\alpha) = \int_{\forall x \in X} s_R(x|\alpha) \tag{12}$$

$$\underline{\mu}_{\tilde{A}}(x|\alpha) = \int_{\forall x \in X} s_L(x|\alpha) \tag{13}$$

By raising the α *-plane* $\tilde{A}_\alpha$ to the level of α , a special IT2 FS is created. This FS was named α *-level* T2 FS $R_{\tilde{A}_\alpha}(x,u)$ in [16], [19] and expressed as follows:

$$R_{\tilde{A}_\alpha}(x,u) = \alpha / \tilde{A}_\alpha, \ \forall x \in X, \ \forall u \in J_x \tag{14}$$

Finally, according to Liu's representation theorem, the GT2 FS $\tilde{A}$ can be constructed as a composition of all of its individual α *-level* T2 FSs [16]:

$$\tilde{A} = \bigcup_{\alpha \in [0,1]} \alpha / \tilde{A}_\alpha \tag{15}$$

It should be noted here, that the $\bigcup$ sign denotes the union set-theoretic operations, which for each point computes the maximum membership grade for all α *-planes*. Furthermore, Liu used the α *-plane* representation theorem to express the centroid $C_{\tilde{A}}(x)$ of a GT2 FS $\tilde{A}$ as a composition of individual centroids $C_{\tilde{A}_\alpha}(x)$ of the respective α *-level* T2 FS $R_{\tilde{A}_\alpha}(x,u)$:

$$C_{\tilde{A}}(x) = \bigcup_{\alpha \in [0,1]} \alpha / C_{\tilde{A}_\alpha}(x) \tag{16}$$

Because each α *-level* T2 FS $R_{\tilde{A}_\alpha}(x,u)$ is an interval-valued set, centroid $C_{\tilde{A}_\alpha}(x)$ will become an interval set completely determined by its left and right boundaries $C_{\tilde{A}_\alpha}(x) = [c_l(\tilde{A}|\alpha), c_r(\tilde{A}|\alpha)]$. Hence, it follows that:

$$C_{\tilde{A}}(x) = \bigcup_{\alpha \in [0,1]} \alpha / [c_l(\tilde{A}|\alpha), c_r(\tilde{A}|\alpha)] \tag{17}$$

Equation (16) showed a new way for computing the centroid $C_{\tilde{A}_\alpha}(x)$ of a GT2 FS $\tilde{A}$ via type-reducing each α-*plane* $\tilde{A}_\alpha$ and then merging the results together.

The different discussed variants of GT2 FSs are depicted in Fig. 1.

III. Type-Reduction of General Type-2 Fuzzy sets

This section provides an overview of type-reduction techniques for GT2 FSs represented using the α*-plane* framework.

A. Karnik Mendel Iterative Algorithms

Initially, the GT2 FS represented in the α*-plane* framework was type-reduced using entirely independent applications of the Karnik-Mendel (KM) iterative algorithms applied to each α*-plane* [16], [17], [18]. Recall that for an IT2 FS $\tilde{A}$, the KM algorithm iteratively converges towards the left and right switch points *L* and *R*, which are then used to compute the left and right centroid boundary points $c_l(\tilde{A})$ and $c_r(\tilde{A})$ as follows [20]:

$$c_l(\tilde{A}) = \frac{\sum_{i=1}^{L} x_i \, \overline{\mu}_{\tilde{A}}(x_i) + \sum_{i=L+1}^{N} x_i \, \underline{\mu}_{\tilde{A}}(x_i)}{\sum_{i=1}^{L} \overline{\mu}_{\tilde{A}}(x_i) + \sum_{i=L+1}^{N} \underline{\mu}_{\tilde{A}}(x_i)} \tag{18}$$

$$c_r(\tilde{A}) = \frac{\sum_{i=1}^{R} x_i \, \underline{\mu}_{\tilde{A}}(x_i) + \sum_{i=R+1}^{N} x_i \, \overline{\mu}_{\tilde{A}}(x_i)}{\sum_{i=1}^{R} \underline{\mu}_{\tilde{A}}(x_i) + \sum_{i=R+1}^{N} \overline{\mu}_{\tilde{A}}(x_i)} \tag{19}$$

The details of the KM iterative algorithm are presented in *Appendix A*.

Assume that a GT2 FS $\tilde{A}$ is represented using *k* α*-planes*, where $k \geq 2$. The algorithm for computing the centroid $C_{\tilde{A}}$ can be described in several steps as follows [16], [17]:

Step 1: For each α*-plane* $\tilde{A}_\alpha$ create the α*-level* T2 FS $R_{\tilde{A}_\alpha}(x,u)$.

Step 2: Compute the boundary points $c_l(\tilde{A}|\alpha)$ and $c_r(\tilde{A}|\alpha)$ of centroid $C_{\tilde{A}_\alpha}$ using the KM algorithm (details in *Appendix A*).

Step 3: Repeat steps 2 and 1 for all *k* α*-planes* of $\tilde{A}$.

Step 4: Construct the final centroid $C_{\tilde{A}}$ using (16).

Following the computational analysis of the KM iterative algorithm in *Appendix A*, the above method for type-reduction of GT2 FS $\tilde{A}$ requires time $O(Nkn)$, where *N* is the number of discretization steps of the primary domain and *n* is the number of iterations required by the KM algorithm.

B. Enhanced Karnik Mendel Iterative Algorithm

In an effort to speed up the computational time of the KM algorithm, the Enhanced KM (EKM) algorithm was proposed in [25]. The major differences were as follows: 1) improved initialization of the switch points *L* and *R*, 2) recursive update of the computed centroid boundary, and 3) improved convergence test.

The EKM algorithm was applied to independent type-reduction of each α*-plane* in [21], [22], leading to a faster computation of the centroid in the presented implementation.

C. Centroid Flow Algorithm

The previous KM and EKM based approaches used completely independent type-reduction of each α*-plane*. There, the entire GT2 FS was considered as a composition of independent α*-level* T2 FSs $R_{\tilde{A}_\alpha}(x,u)$. The first algorithm to leverage the interconnected structure of the α*-level* T2 FSs was the Centroid Flow (CF) algorithm developed by Zhai and Mendel [21], [22].

The CF algorithm starts by applying the KM/EKM iterative algorithm to the lowest α*-plane* $\tilde{A}_0$ in order to compute the base of the type-reduced centroid. Next, the derivatives of the secondary membership functions are used to propagate the centroid coordinate from one α*-plane* to the next. The fundamental idea of the CF algorithm is to use the α*-planes* connection equations, rather than independently type-reducing each α*-plane* [21], [22]:

$$s_L(x|\alpha + T_s) = s_L(x|\alpha) + \left[T_s / g'(s_L(x|\alpha))\right] \tag{20}$$

$$s_R(x|\alpha + T_s) = s_R(x|\alpha) + \left[T_s / h'(s_R(x|\alpha))\right] \tag{21}$$

Here, T_S denotes the difference of the value of α between two neighboring α*-planes* and $g'(x)$ and $h'(x)$ are the derivatives of the left and the right shoulders of the secondary membership function of a vertical slice at coordinate *x*.

The CF algorithm requires only a single application of the iterative KM/EKM algorithm applied to the base α*-plane* $\tilde{A}_0$, which can be computed in time $O(Nn)$. The centroid of the subsequent α*-planes* is then computed using the previous location and the centroid derivative following (17) and (18). The computation of the centroid derivative requires $O(N)$ steps for each α*-plane*. Thus, the computational time of the CF algorithm is $O(Nn + kN)$.

A simplified version of the CF algorithm can be applied to GT2 FSs with triangular or trapezoidal secondary membership functions. Here, the constant derivative of the secondary functions at each vertical slice can be pre-computed saving computation of several integrals at each α*-plane*.

D. Other Applicable Type-reduction Techniques

A GT2 FS represented in the α*-plane* framework can be viewed as a composition of many individual IT2 FSs. Hence, in general any algorithm for type-reduction of IT2 FSs can be also used to compute the centroid of the GT2 FS. In the available literature, the authors are aware of the work of Greenfield who used the sampling defuzzifier for type-reduction of a GT2 FS represented in the α*-plane* framework [26], [27]. However, in [27] only the defuzzified value is presented, without commenting on the properties of the

calculated centroid. Other applicable type-reduction techniques are for example the Collapsing defuzzifier [28].

IV. Monotone Centroid Flow Algorithm

In this section the monotone property of the α *-plane* representation of GT2 FS and the type-reduced centroid is first derived. Next, the MCF algorithm is introduced.

As pointed out by one of the reviewers: "*…centroid of general type-2 fuzzy set is 'hot' right now*." As it sometimes happens with "hot" topics, different research groups might arrive at similar results independently. Some of the ideas behind the introduced MCF algorithm have been independently developed by Yeh et al. [29]. However, there are several significant differences between the Enhanced Type-Reduction algorithm proposed by Yeh et al. and the MCF algorithm. These differences are summarized at the end of this section.

A. Monotonicity of α - planes

Initially, the following assumption is made: at any value of primary variable *x*, the vertical slice $\mu_{\tilde{A}}(x)$ is a convex T1 FS defined by the secondary membership function $f_x(u)$ with the following properties:

$$f_x(u)=\begin{cases} g_x(u) & u\in\left[s_L(x|0),s_L(x|1)\right], \quad g_x(u)\in[0,1] \\ 1 & u\in\left[s_L(x|1),s_R(x|1)\right] \\ h_x(u) & u\in\left[s_R(x|1),s_R(x|0)\right], \quad h_x(u)\in[0,1] \\ 0 & Otherwise \end{cases} \tag{22}$$

where $g_x(u)$ and $h_x(u)$ are monotonically non-decreasing and monotonically non-increasing functions in their respective domains.

This assumption on the nature of secondary membership functions might seem as a limiting factor for the applicability of the proposed algorithm. Examples of convex T1 FS FSs are triangular, trapezoidal and Gaussian T1 FSs, which are the most commonly used types of secondary membership functions for GT2 FSs. Karnik and Mendel proved in [30] that under the maximum t-conorm and the minimum t-norm, the set-theoretic operations join and meet on GT2 FSs with convex and normal secondary membership functions also result in GT2 FSs with convex and normal secondary membership functions. Because maximum t-conorm and minimum t-norm are commonly used in fuzzy logic system, the proposed MCF algorithm is widely applicable.

Property 1: *Containment of α -planes*:

$$\tilde{A}_{\alpha_1}\subseteq\tilde{A}_{\alpha_2} \quad \text{if} \quad \alpha_1\geq\alpha_2 \tag{23}$$

This containment property was stated without a proof by Liu in [16] and it also appeared in [17], [31]. For completeness sake the proof of **Property 1** is given in *Appendix B*.

Property 2: *Containment of Centroids*:

$$C_{\tilde{A}_{\alpha_1}}(x)\subseteq C_{\tilde{A}_{\alpha_2}}(x) \quad \text{if} \quad \alpha_1\geq\alpha_2 \tag{24}$$

Property 2 appeared as an observation without a proof in [17]. For completeness sake, its proof is included in *Appendix B*.

Corollary 1: *Monotonicity of Centroids*:

$$\begin{matrix} c_l(\tilde{A}|\alpha_1)\geq c_l(\tilde{A}|\alpha_2) \\ \\ c_r(\tilde{A}|\alpha_1)\leq c_r(\tilde{A}|\alpha_2) \end{matrix} \quad \text{if} \quad \alpha_1\geq\alpha_2 \tag{25}$$

Corollary 1 is a direct consequence of **Property 2**. For completeness sake its proof is presented in *Appendix B*.

B. Monotone Centroid Flow Algorithm

The MCF algorithm leverages **Corollary 1** to propagate the boundary values $[c_l(\tilde{A}|\alpha_i), c_r(\tilde{A}|\alpha_i)]$ of centroid $C_{\tilde{A}_{\alpha_i}}$ to the type-reduction process of a nearby α *-plane* $\tilde{A}_{\alpha_j}$. Hence, similarly to the CF algorithm, the type-reduction of each α *-plane* does not run independently, but rather proceeds in a sequential manner taking advantage of the previously computed results.

1) Initialization

Unlike the CF algorithm that uses the approximated gradients of the centroid [21], [22], the MCF algorithm utilizes a monotone iteration through the discretized domain of the primary variable *x*. However, an initial starting point for the iteration procedure must be first provided. This initial point is computed at a selected starting α *-plane*. For instance, the CF algorithm begins at $\tilde{A}_0$ and thus starts by applying the KM/EKM algorithm to the actual $FOU(\tilde{A})$ [21]. The proposed MCF algorithm uses a different approach starting its way down from the highest α *-plane* $\tilde{A}_1$.

Recall that a core $core(A)$ of a T1 fuzzy set *A* is described as a set of elements of *X* with a membership degree of 1. For the secondary membership function $f_x(u)$, the core can be defined as [32]:

$$core(f_x(u))=\left\{u\in J_x \mid f_x(u)=1\right\} \tag{26}$$

For T1 FSs such as triangular fuzzy sets, the core is composed of a single point at the position of the apex of the triangle. For T1 FSs such as trapezoidal fuzzy sets, the core becomes an interval bounded by the coordinates of the apex of the trapezoid.

When observing the properties of the highest α *-plane* $\tilde{A}_1$ for an arbitrary secondary membership functions $f_x(u)$, two cases can be encountered. In the first case, the highest α *-plane* $\tilde{A}_1$ reduces to a single line. This can happen when the core of all vertical slice of fuzzy set $\tilde{A}$ is a singleton (e.g.

when all $f_x(u)$ are triangular). Such fuzzy set $\tilde{A}$ fulfills the following condition:

$$\forall x \in X \qquad \bar{\mu}_{\tilde{A}}(x|1) = \underline{\mu}_{\tilde{A}}(x|1) \tag{27}$$

In the second case, the highest α *-plane* $\tilde{A}_1$ is an actual plane, which might only locally reduce to a line. This can happen when there exists a vertical slice of fuzzy set $\tilde{A}$ with an interval core (e.g. when some $f_x(u)$ are trapezoidal). For such GT2 FS $\tilde{A}$, the following condition holds:

$$\exists x \in X \qquad \bar{\mu}_{\tilde{A}}(x|1) \neq \underline{\mu}_{\tilde{A}}(x|1) \tag{28}$$

The initial point for the MCF algorithm is set to be the defuzzified value of the principal membership function of the GT2 fuzzy set $\tilde{A}$, denoted as A^*. In the first case (e.g. all $f_x(u)$ are triangular), the principal membership function is simply equal to the highest α *-plane* $\tilde{A}_1$. In the second case (e.g. when some $f_x(u)$ are trapezoidal), the principal membership function can be computed as the average of the lower and the upper boundary membership functions of to the highest α *-plane* $\tilde{A}_1$. In summary:

$$A^*(x) = \begin{cases} \tilde{A}_1(x) = \bar{\mu}_{\tilde{A}}(x|1) = \underline{\mu}_{\tilde{A}}(x|1) \\ \qquad if \quad \bar{\mu}_{\tilde{A}}(x|1) = \underline{\mu}_{\tilde{A}}(x|1) \\ \dfrac{\bar{\mu}_{\tilde{A}}(x|1) + \underline{\mu}_{\tilde{A}}(x|1)}{2} \\ \qquad if \quad \bar{\mu}_{\tilde{A}}(x|1) \neq \underline{\mu}_{\tilde{A}}(x|1) \end{cases} \tag{29}$$

Because the principal membership function A^* is a T1 FS, it can be defuzzified using the standard centroid defuzzifier as follows [1]:

$$C_{A^*} = \frac{\sum_{i=1}^{N} x_i \mu_{A^*}(x_i)}{\sum_{i=1}^{N} \mu_{A^*}(x_i)} \tag{30}$$

Here, N denotes the number of discretized steps in the primary domain. The defuzzified value C_{A^*} of the principal membership function A^* defines the initial point for the MCF algorithm. Note, that when the highest α *-plane* $\tilde{A}_1$ does not reduce to a single line, then $C_{A^*} \neq C_{A_1}$. Therefore the value of C_{A^*} can only be used to initialize the monotone search, which must first compute the accurate centroid of top α *-plane* $\tilde{A}_1$.

Theorem 1: *Initialization of the MCF algorithm:*

$$c_l(\tilde{A}|\alpha) \leq C_{A^*} \quad \text{and} \quad c_r(\tilde{A}|\alpha) \geq C_{A^*} \qquad \forall \alpha \in [0,1] \tag{31}$$

The proof of **Theorem 1** is provided in *Appendix B*.

2) Incremental Step

Assume that the centroid $C_{\tilde{A}_t}$ for an α *-plane* $\tilde{A}_{\alpha_t}$ was computed as $[c_l(\tilde{A}|\alpha_t), c_r(\tilde{A}|\alpha_t)]$. This result was obtained by calculating the left and right switching points $L(\tilde{A}|\alpha_t)$ and $R(\tilde{A}|\alpha_t)$. The estimated boundaries $[\hat{c}_l(\tilde{A}|\alpha_{t-1}), \hat{c}_r(\tilde{A}|\alpha_{t-1})]$ of the centroid of the neighboring α *-plane* $\tilde{A}_{\alpha_{t-1}}$ can be computed using the switch points from the preceding α *-plane* $\tilde{A}_{\alpha_t}$. Hence, the auxiliary variables $\hat{L}$ and $\hat{R}$ are initialized to the values of switch points $L(\tilde{A}|\alpha_t)$ and $R(\tilde{A}|\alpha_t)$ and the estimated centroid is computed as follows:

$$\hat{c}_l(\tilde{A}|\alpha_{t-1}) = \frac{\sum_{i=1}^{\hat{L}} x_i \, \bar{\mu}_{\tilde{A}}(x_i|\alpha_{t-1}) + \sum_{i=\hat{L}+1}^{N} x_i \, \underline{\mu}_{\tilde{A}}(x_i|\alpha_{t-1})}{\sum_{i=1}^{\hat{L}} \bar{\mu}_{\tilde{A}}(x_i|\alpha_{t-1}) + \sum_{i=\hat{L}+1}^{N} \underline{\mu}_{\tilde{A}}(x_i|\alpha_{t-1})} \tag{32}$$

$$\hat{c}_r(\tilde{A}|\alpha_{t-1}) = \frac{\sum_{i=1}^{\hat{R}} x_i \, \underline{\mu}_{\tilde{A}}(x_i|\alpha_{t-1}) + \sum_{i=\hat{R}+1}^{N} x_i \, \bar{\mu}_{\tilde{A}}(x_i|\alpha_{t-1})}{\sum_{i=1}^{\hat{R}} \underline{\mu}_{\tilde{A}}(x_i|\alpha_{t-1}) + \sum_{i=\hat{R}+1}^{N} \bar{\mu}_{\tilde{A}}(x_i|\alpha_{t-1})} \tag{33}$$

Denote the numerator and the denumerator in (32) and (33) as $E_{1,\hat{L}}$, $E_{2,\hat{L}}$ and $D_{1,\hat{R}}, D_{2,\hat{R}}$, respectively. An elementary operations of the MCF algorithm are monotone decrement and increment of the auxiliary variables $\hat{L}$ and $\hat{R}$ and the subsequent update of the centroid estimate $[\hat{c}_l(\tilde{A}|\alpha_{t-1}), \hat{c}_r(\tilde{A}|\alpha_{t-1})]$.

Theorem 2: *Recursive update*:

Given the values of $E_{1,\hat{L}}$, $E_{2,\hat{L}}$ and $D_{1,\hat{R}}, D_{2,\hat{R}}$ the updated values for switch points $\hat{L}-1$ and $\hat{R}+1$ can be computed recursively as follows:

$$E_{1,\hat{L}-1} = E_{1,\hat{L}} - x_{\hat{L}}(\bar{\mu}_{\tilde{A}}(x_{\hat{L}}|\alpha_{t-1}) - \underline{\mu}_{\tilde{A}}(x_{\hat{L}}|\alpha_{t-1})) \tag{34}$$

$$E_{2,\hat{L}-1} = E_{2,\hat{L}} - \bar{\mu}_{\tilde{A}}(x_{\hat{L}}|\alpha_{t-1}) + \underline{\mu}_{\tilde{A}}(x_{\hat{L}}|\alpha_{t-1}) \tag{35}$$

$$D_{1,\hat{R}+1} = D_{1,\hat{R}} - x_{\hat{R}}(\underline{\mu}_{\tilde{A}}(x_{\hat{R}}|\alpha_{t-1}) - \bar{\mu}_{\tilde{A}}(x_{\hat{R}}|\alpha_{t-1})) \tag{36}$$

$$D_{2,\hat{R}+1} = D_{2,\hat{R}} - \underline{\mu}_{\tilde{A}}(x_{\hat{R}}|\alpha_{t-1}) + \bar{\mu}_{\tilde{A}}(x_{\hat{R}}|\alpha_{t-1}) \tag{37}$$

The proof of **Theorem 2** is provided in *Appendix B*. Note that this idea of incrementally stepping through the domain of primary variable x is similar to the idea of the IASCO algorithm presented by Duran et al [33]. However, the IASCO

algorithm was performing an exhaustive search for both switch points in the primary domain starting from its lower end-point and was only applied to IT2 FSs. The MCF algorithm uses the incremental stepping procedure through the primary domain of the GT2 FS iterating from the initial starting point outwards and descending from the top α *-plane* to the lowest one.

3) The MCF Algorithm

The MCF algorithm starts with the defuzzification of the principal membership function of the GT2 FS $\tilde{A}$. The algorithm further proceeds sequentially through individual α *-planes* in a top-down direction. The switch points from one α *-plane* are used to compute the estimate of the centroids of the subsequent α *-plane*. In this manner, the left and right switch points are monotonically decremented and incremented until the lowest α *-plane* $\tilde{A}_0$ is reached. The MCF algorithm can be described in several steps as follows:

Input: A GT2 FS $\tilde{A}$, decomposed into $k \geq 2$ α *-planes*. The domain of the primary variable *x* is discretized into *N* samples.
Output: T1 FS defining the centroid $C_{\tilde{A}}(x)$ represented as a set of coordinates pairs $[c_l(\tilde{A}\,|\,\alpha), c_r(\tilde{A}\,|\,\alpha)]$ for each considered value of α.

Step 1: Compute the principal membership function $A^*(x)$ using (29).

Step 2: Calculate the centroid C_{A^*} of the primary membership function (30). Initialize the global switch points $\hat{L}$ and $\hat{R}$ as follows:

$$\hat{L} = \lceil C_{A^*} \rceil, \;\; \hat{R} = \lfloor C_{A^*} \rfloor \tag{38}$$

where operators $\lceil\ \rceil$ and $\lfloor\ \rfloor$ compute the discretized index of the value C_{A^*} rounded to the nearest value of the discretized variable *x* in the increasing and in the decreasing direction, respectively.

Step 3: Repeat for all α *-planes* starting from the highest α *-plane* $\tilde{A}_{\alpha_k}$ ($\tilde{A}_1$) until the lowest α *-plane* $\tilde{A}_{\alpha_0}$ ($\tilde{A}_0$) is reached.

Step 3.1: Compute values $E_{1,\hat{L}}$, $E_{2,\hat{L}}$ and $D_{1,\hat{R}}$, $D_{2,\hat{R}}$ and calculate the estimated centroid $[\hat{c}_l(\tilde{A}|\alpha_{t-1}), \hat{c}_r(\tilde{A}|\alpha_{t-1})]$ of α *-planes* $\tilde{A}_{\alpha_{t-1}}$ for the global switch points $\hat{L}$ and $\hat{R}$ using (32) and (33).

Step 3.2: Pre-compute the location of the estimated left centroid boundary $\hat{c}_l(\tilde{A}|\alpha_{t-1})$ for a switch point $\hat{L}-1$ as follows. Recursively calculate the values of $E_{1,\hat{L}-1}$, $E_{2,\hat{L}-1}$ using (34) and (35). Compute the estimated left centroid boundary as:

$$c_l' = \frac{E_{1,\hat{L}-1}}{E_{2,\hat{L}-1}} \tag{39}$$

Note that (39) resembles (32) with the switch point $\hat{L}-1$, where the numerator and the denumerator were recursively pre-computed according to (34) and (35).

Step 3.3: If $c_l' < \hat{c}_l(\tilde{A}|\alpha_{t-1})$, set $\hat{c}_l(\tilde{A}|\alpha_{t-1}) = c_l'$, decrement the global switch point $\hat{L} = \hat{L} - 1$ and go back to **Step 3.2**. Else, store the estimated left centroid boundary as the true centroid boundary $c_l(\tilde{A}|\alpha_{t-1}) = \hat{c}_l(\tilde{A}|\alpha_{t-1})$ and proceed to **Step 3.4**.

Step 3.4: Pre-compute the location of the estimated right centroid boundary $\hat{c}_r(\tilde{A}|\alpha_{t-1})$ for a switch point $\hat{R}+1$ as follows. Recursively calculate the values $D_{1,\hat{R}+1}$, $D_{2,\hat{R}+1}$ using (36) and (37). Compute the estimated right centroid boundary as:

$$c_r' = \frac{D_{1,\hat{R}+1}}{D_{2,\hat{R}+1}} \tag{40}$$

Note that (40) resembles (33) with the switch point $\hat{R}+1$, where the numerator and the denumerator were recursively pre-computed using (36) and (37).

Step 3.5: If $c_r' > \hat{c}_r(\tilde{A}|\alpha_{t-1})$, set $\hat{c}_r(\tilde{A}|\alpha_{t-1}) = c_r'$, increment the global switch point $\hat{R} = \hat{R} + 1$ and go back to **Step 3.4**: Else, store the estimated right centroid boundary as the true centroid boundary $c_r(\tilde{A}|\alpha_{t-1}) = \hat{c}_r(\tilde{A}|\alpha_{t-1})$ and proceed to the next α *-planes* below.

An illustration of the MCF algorithm for a GT2 FSs with two α *-planes* is depicted in Fig. 2. First, the search for the left centroid boundary $c_l(\tilde{A}|\alpha_1)$ of α *-plane* $\tilde{A}_{\alpha_1}$ is illustrated in Fig. 2(a). The label *Init* marks the initial position of the global switch point $\hat{L}$ based on the center of gravity C_{A^*} of the principal membership function A^*. Next, the global switch point $\hat{L}$ is decremented two times in *Step* 1 and *Step* 2 (denoted as $\hat{L}-$) together with the recursive computation of the estimated left centroid boundary according to (39). Note that calculation of the estimated left centroid boundary in *Step* 2 is needed in order to confirm that the calculated left boundary $c_l(\tilde{A}|\alpha_1)$ is the correct result (**Step 3.3**). Subsequently, the MCF algorithm proceeds to the neighboring α *-plane* $\tilde{A}_{\alpha_2}$ seeking the new left centroid boundary $c_l(\tilde{A}|\alpha_2)$ (Fig. 2(c)). Here, the value of the final switch point from α *-plane* $\tilde{A}_{\alpha_1}$ $L(\tilde{A}|\alpha_1)$ is used to initialize the iterative stepping procedure on α *-plane* $\tilde{A}_{\alpha_2}$ in *Step* 3. Next, two more decrements of the global left switch points $\hat{L}$ in *Step* 4 and *Step* 5 are performed to find the left centroid boundary

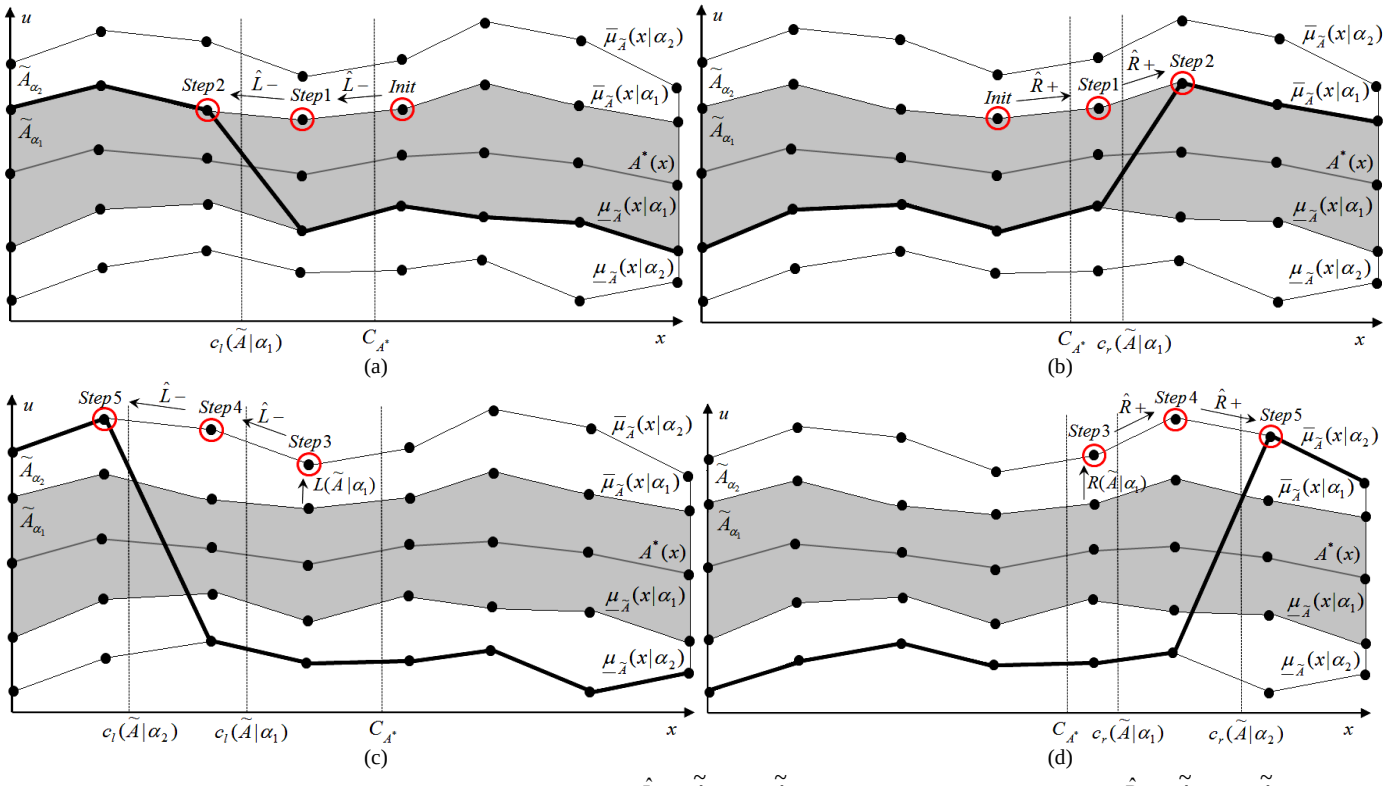


Fig. 2 Schematic view of the monotone iteration of the left switch point $\hat{L}$ at $\tilde{A}_{\alpha_1}$ and $\tilde{A}_{\alpha_2}$ (a), (c) and of the right switch point $\hat{R}$ at $\tilde{A}_{\alpha_1}$ and $\tilde{A}_{\alpha_2}$ (b), (d).

$c_l(\tilde{A}|\alpha_2)$. The MCF algorithm proceeds in a similar manner in case of the right centroid boundary incrementing the right global switch point $\hat{R}$ as shown in Fig. 2(b) and 2(d).

Theorem 3: *Stopping of the MCF algorithm*:

The MCF stops after at most N updates of either global switching points $\hat{L}$ and $\hat{R}$.

Proof: Both global switching points $\hat{L}$ and $\hat{R}$ can only take on values from the discretized primary domain $\{x_1,..., x_N\}$. Further, variables $\hat{L}$ and $\hat{R}$ can be only monotonically decremented or incremented in **Step 3.3** and **Step 3.5**, respectively. Hence, they can be updated at most N times, which completes the proof.

C. Computational Complexity

One of the interesting consequences of the monotone property of the α*-plane* representation of GT2 FSs is that individual α*-planes* do not have to be separately type-reduced using the KM/EKM algorithms. Rather, they are processed in a sequential manner. Furthermore, because of the initialization of the monotone iterative stepping procedure using the principal membership function $A^*(x)$ and proceeding in a top-down manner, the KM/EKM algorithms have been completely eliminated from the type-reduction process using the MCF algorithm.

The calculation of the primary membership function $A^*(x)$ and its subsequent defuzzification in **Step 1** and **2** requires $2N$ steps.

The main body of the algorithm will be repeated k times for each α*-plane*. At each α*-plane* the most computational effort is required for computing the values of $E_{1,\hat{L}}$, $E_{2,\hat{L}}$ and $D_{1,\hat{R}}$, $D_{2,\hat{R}}$ used to calculate the estimated centroid boundaries $\hat{c}_l(\tilde{A}|\alpha_{t-1})$ and $\hat{c}_r(\tilde{A}|\alpha_{t-1})$. This phase requires $2N$ steps.

For each α*-plane*, at least one centroid location for switching points $\hat{L}-1$ and $\hat{R}+1$ will be computed in **Steps 3.3** and **3.4**. Theoretically, either of the sub-loops between **Step 3.2 - 3.3** and **Step 3.4 - 3.5** can be evaluated up to N times for a single α*-plane*. However, according to **Theorem 2** there can be at most N updates of either global switch points $\hat{L}$ and $\hat{R}$, resulting in an overall at most N pre-computed centroid positions in **Steps 3.3** and **3.4**.

In summary, the computational complexity of the MCF algorithm can be described as:

$$O(2N + k2N + k + N) \tag{41}$$

This can be further simplified into:

$$O(Nk + \max(N, k)) \tag{42}$$

The second term in (42) was not excluded despite its apparent lower asymptotical order since it might play a significant role during performance comparison against other algorithms such as the independent application of KM/EKM algorithms or the CF algorithm.

As stated at the beginning of this Section, the Enhanced Type-Reduction (ETR) algorithm independently developed by Yeh et al. features some similarities with the presented MCF algorithm [29]. Both algorithms propose to start the type-reduction process from the highest α *-plane* $\tilde{A}_1$ and proceed in a top-down fashion. Also, both approaches are based on the idea to initialize the search for switch points at current α *-plane* with the switch points computed for the previous α *-plane*. However, the ETR algorithm uses the original KM algorithm to obtain the initial centroid at the highest α *-plane* $\tilde{A}_1$ and then also uses the KM algorithm to iteratively compute the switch points at each consecutive α *-plane* with the new initial values. Hence, the ETR algorithm still relies on the iterative KM algorithms and does not take full advantage of the monotone property of centroid as derived in this paper. On the other hand, the MCF algorithm is in its nature more similar to the IASCO algorithm, which monotonically iterates through the domain from the initial starting value until the final centroid is obtained [33]. In summary, the MCF algorithm starts from the defuzzified value of the principal membership function of the highest α *-plane* $\tilde{A}_1$ and then alternates between monotonically traversing outwards through the domain of the discretized primary variable and descending to the consecutive α *-plane* when the current switch points are computed.

V. Experimental Results

This section presents the experimental testing of the proposed MCF algorithm. The MCF algorithm was compared to the KM/EKM algorithms and the CF algorithm in terms of both accuracy and computational time.

For the purpose of experimental testing two benchmark GT2 FSs commonly used in literature were implemented [16], [17]. Fuzzy set $\tilde{F}$ has a piece-wise linear FOU, which can be described in terms of the lower and upper membership functions as follows:

$$\overline{\mu}_{\tilde{F}}(x) = \max\left\{\begin{bmatrix} (x-1), & x \in \langle 1,3\rangle \\ (7-x)/4, & x \in \langle 3,7\rangle \\ 0, & x \notin \langle 1,7\rangle \end{bmatrix}, \begin{bmatrix} (x-2)/5, & x \in \langle 2,6\rangle \\ (16-2x)/5, & x \in \langle 6,8\rangle \\ 0, & x \notin \langle 2,8\rangle \end{bmatrix}\right\} \tag{43}$$

$$\underline{\mu}_{\tilde{F}}(x) = \max\left\{\begin{bmatrix} (x-1)/6, & x \in \langle 1,4\rangle \\ (7-x)/6, & x \in \langle 4,7\rangle \\ 0, & x \notin \langle 1,7\rangle \end{bmatrix}, \begin{bmatrix} (x-3)/6, & x \in \langle 3,5\rangle \\ (8-x)/9, & x \in \langle 5,8\rangle \\ 0, & x \notin \langle 3,8\rangle \end{bmatrix}\right\} \tag{44}$$

Fuzzy set $\tilde{G}$ is composed of two Gaussian membership functions defined as follows:

$$\overline{\mu}_{\tilde{G}}(x) = \max\left\{\exp\left[-\frac{(x-3)^2}{8}\right], 0.8\exp\left[-\frac{(x-6)^2}{8}\right]\right\} \tag{45}$$

$$\underline{\mu}_{\tilde{G}}(x) = \max\left\{0.5\exp\left[-\frac{(x-3)^2}{2}\right], 0.4\exp\left[-\frac{(x-6)^2}{2}\right]\right\} \tag{46}$$

Fuzzy set $\tilde{F}$ maintains trapezoidal secondary membership functions. The position of the left and the right boundary points of the core of the trapezoid can be adjusted using parameter *w* [16], [17]:

$$core_L(x) = \underline{\mu}_{\tilde{F}}(x) + 0.6\,w\,(\overline{\mu}_{\tilde{F}}(x) - \underline{\mu}_{\tilde{F}}(x)) \tag{47}$$

$$core_R(x) = \overline{\mu}_{\tilde{F}}(x) - 0.6(1-w)\,(\overline{\mu}_{\tilde{F}}(x) - \underline{\mu}_{\tilde{F}}(x)) \tag{48}$$

The secondary membership function of fuzzy set $\tilde{G}$ was created using the non-linear spline-based curve with a single-point core computed as [16], [17]:

$$core(x) = \underline{\mu}_{\tilde{G}}(x) + w\,(\overline{\mu}_{\tilde{G}}(x) - \underline{\mu}_{\tilde{G}}(x)) \tag{49}$$

Fuzzy sets $\tilde{F}$ and $\tilde{G}$ represented using 5 α *-planes* are

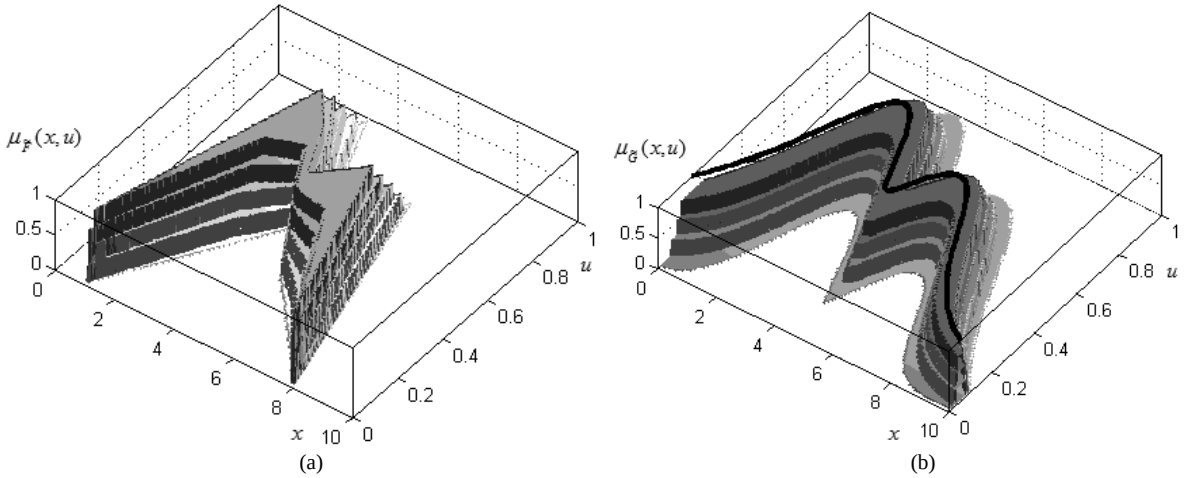


Fig. 3 Benchmark GT2 fuzzy sets $\tilde{F}$ (a) and $\tilde{G}$ (b) represented using 5 α *-planes*. The bold-faced line in (b) depicts the top α *-plane* $\tilde{G}_1$.

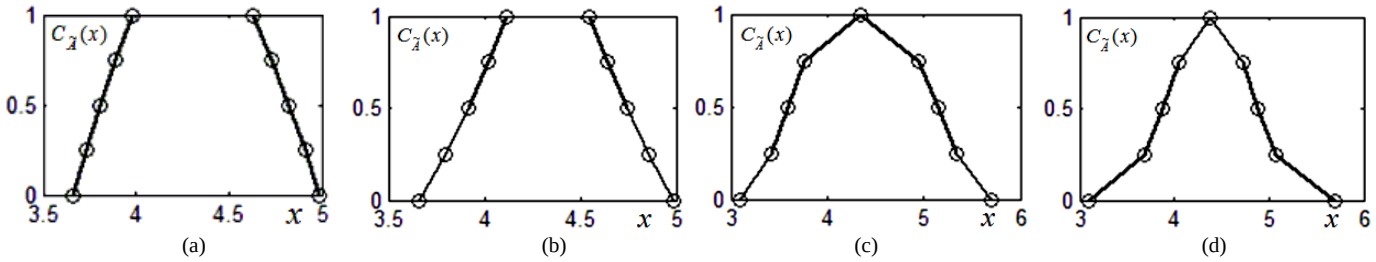


Fig. 4 The centroid calculated using the MCF algorithm (circles) and using the independent application of the KM algorithm (solid line) with N=20, k=5 for fuzzy set $\tilde{F}$ with w=0.1 (a), $\tilde{F}$ with w=0.9 (b), $\tilde{G}$ with w=0.1 (c) and $\tilde{G}$ with w=0.9 (d).

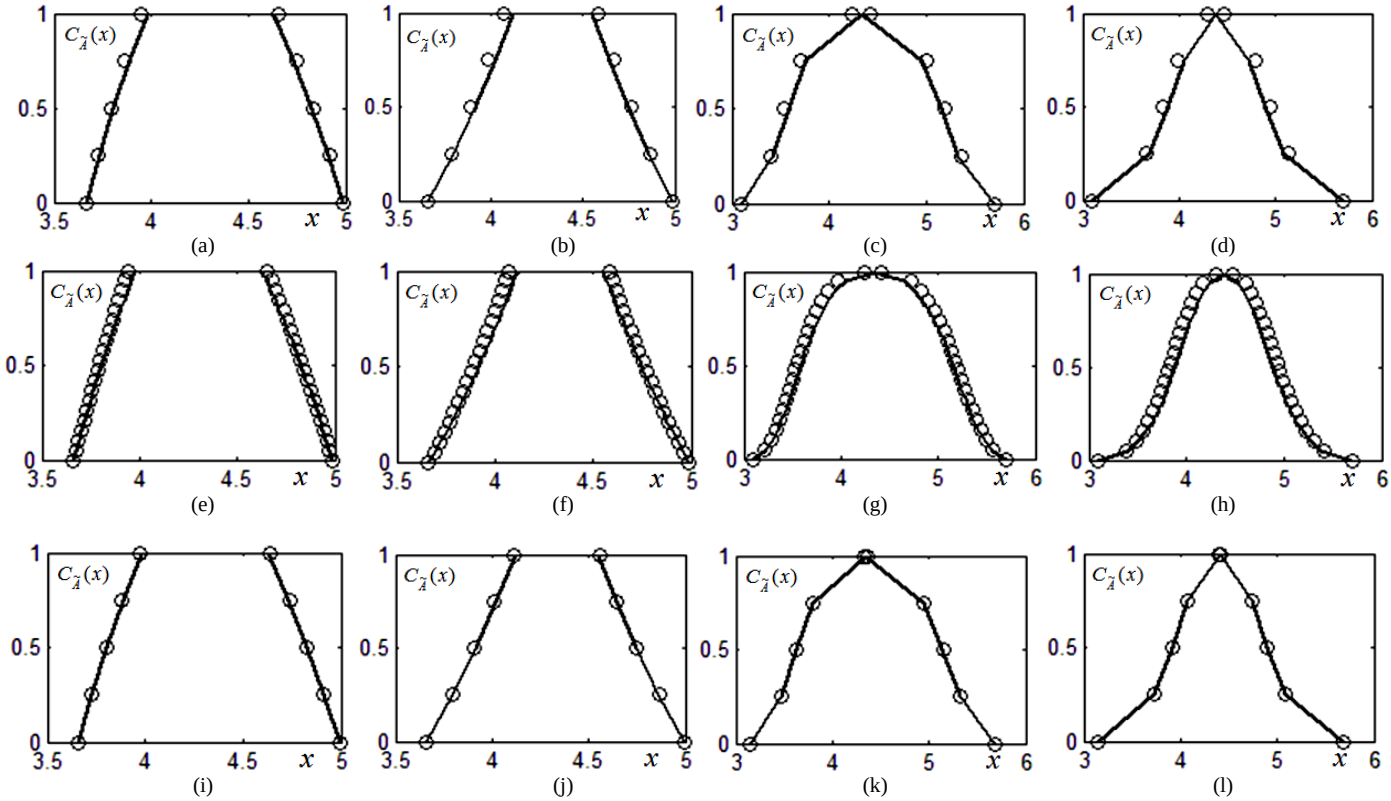


Fig. 5 The centroid calculated using the CF algorithm (circles) and using the MCF algorithm (solid line) with N=20, k=5 (a)-(d), N=20, k=20 (e)-(h) and N=100, k=5 (i)-(l). In columns from left to right the original GT2 fuzzy set was $\tilde{F}$ with w=0.1, $\tilde{F}$ with w=0.9, $\tilde{G}$ with w=0.1 and $\tilde{G}$ with w=0.9.

shown in Fig. 3. Note that the α *-plane* $\tilde{G}_1$ is only depicted as a line in Fig. 3(b).

A. Accuracy Testing

First, the KM[1], CF and the MCF algorithm were compared in terms of accuracy. Fig. 4 depicts the centroid of fuzzy sets $\tilde{F}$ and $\tilde{G}$ with secondary membership functions constructed with $w = \{0.1, 0.9\}$ using the KM (solid line) and the MCF (circles) algorithms. It can be observed that the MCF and the KM algorithms compute numerically identical solution.

Next, Fig. 5 compares the centroid computed using the MCF (solid line) and the CF (circles) algorithms. Fig. 5(a)-(d) clearly demonstrate the approximation of the centroid obtained with the CF algorithm for values of k=5 and N=20. N=20 is equivalent to sampling rate of 0.5. Fig. 5(e)-(h) show the computed centroid with an increased number of α *-planes*. It can be observed that the accuracy of the CF algorithm does not improve as more α *-planes* are added for the considered sampling rate of 0.5. Fig. 5(i)-(l) show a substantial improvement of the accuracy of the CF algorithm when increased number of discretized samples is applied to the primary domain. In [22], it was demonstrated that when small sampling rate of 0.01 (N=1000) was used, the results of the CF algorithm could be considered approximately the same when compared to the result of the baseline KM algorithm. The proposed MCF algorithm does not impose any such limits on the sampling rate and computes numerically identical solution when compared to the KM algorithm for all values of k or N.

In addition to the plotted results, Table I numerically quantifies the differences between CF and MCF algorithms. First, the calculated centroids for the highest α -planes $\tilde{F}_1$ and $\tilde{G}_1$ are compared. It can be seen, that the calculated relative error of the CF algorithm is greater for the fuzzy set $\tilde{G}$, which

[1] Only the KM is considered since the EKM algorithm is known to produce numerically identical results.

TABLE I
COMPARISONG OF THE CENTROIDS COMPUTED USING THE CF AND THE MCF ALGORITHMS

Fuzzy Set	$C_{\tilde{A}_1}$ with MCF algorithm	$C_{\tilde{A}_1}$ with CF algorithm	Relative error of the CF algorithm for $\tilde{A}_1$	Distance Measure $d_{\tilde{A}}$	Dissimilarity Measure $s_{\tilde{A}}$
$\tilde{F}$ with w=0.1, k=5, N=20	[3.980, 4.628]	[3.946, 4.655]	[-0.853%, 0.583%]	0.0025	0.0257
$\tilde{F}$ with w=0.9, k=5, N=20	[4.119, 4.551]	[4.074, 4.583]	[-1.092%, 0.703%]	0.0061	0.0449
$\tilde{G}$ with w=0.1, k=5, N=20	[4.338]	[4.228, 4.428]	[-2.536%, 2.075%]	0.0074	0.0555
$\tilde{G}$ with w=0.9, k=5, N=20	[4.386]	[4.302, 4.470]	[-1.915%, 1.915%]	0.0014	0.0846
$\tilde{F}$ with w=0.1, k=20, N=20	[3.980, 4.628]	[3.945, 4.656]	[-0.879%, 0.605%]	0.0025	0.0258
$\tilde{F}$ with w=0.9, k=20, N=20	[4.119, 4.551]	[4.074, 4.584]	[-1.092%, 0.725%]	0.0057	0.0456
$\tilde{G}$ with w=0.1, k=20, N=20	[4.338]	[4.240, 4.421]	[-2.259%, 1.913%]	0.0033	0.0527
$\tilde{G}$ with w=0.9, k=20, N=20	[4.386]	[4.297, 4.468]	[-2.029%, 1.869%]	0.0017	0.0926
$\tilde{F}$ with w=0.1, k=5, N=100	[3.978, 4.628]	[3.971, 4.634]	[-0.176%, 0.129%]	0.0004	0.0058
$\tilde{F}$ with w=0.9, k=5, N=100	[4.118, 4.551]	[4.109, 4.557]	[-0.219%, 0.132%]	0.0013	0.0089
$\tilde{G}$ with w=0.1, k=5, N=100	[4.345]	[4.331, 4.355]	[-0.322%, 0.230%]	0.0058	0.0049
$\tilde{G}$ with w=0.9, k=5, N=100	[4.409]	[4.404, 4.414]	[-0.113%, 0.113%]	0.0005	0.0026

maintains non-linear secondary membership functions. Furthermore, the centroid calculated with the CF algorithm has an interval core, despite all secondary membership functions having a single-point core. This result contradicts some of the conclusions about the centroid of GT2 FSs drawn in [17], where it is stated that when all secondary membership functions are triangles then the centroid will be triangle-looking with its apex will be a single point. In addition, this observation might lead to incorrect conclusions, should the geometry of the centroid be interpreted as a measure of output uncertainty for an associated GT2 FLS.

Next, two measures for comparing the type-reduced centroids have been calculated and reported in Table I. First, the distance measure $d_{\tilde{A}}$ was used, which calculates the difference between the defuzzified values of both centroids [22]:

$$d_{\tilde{A}} = \left| c(C_{\tilde{A}}^{CF}) - c(C_{\tilde{A}}^{MCF}) \right| \tag{50}$$

Here, $c(C_{\tilde{A}}^{CF})$ and $c(C_{\tilde{A}}^{MCF})$ denote the center of gravity of the centroids computed by the CF and MCF algorithms respectively. Second, the dissimilarity measure was computed, which expresses the difference in shapes of both centroids. The dissimilarity measure $s_{\tilde{A}}$ was calculated based on the discrete version of Jaccard similarity measure for T1 FSs [34]:

$$s_{\tilde{A}} = 1 - \frac{\sum_{i=1}^{N} \min(C_{\tilde{A}}^{CF}(x_i), C_{\tilde{A}}^{MCF}(x_i))}{\sum_{i=1}^{N} \max(C_{\tilde{A}}^{CF}(x_i), C_{\tilde{A}}^{MCF}(x_i))} \tag{51}$$

Note that $s_{\tilde{A}} \in [0,1]$ and that the higher its value the more dissimilar both centroids are.

The calculated values of the distance measure $d_{\tilde{A}}$ presented in Table I reveals that the there is only very small difference between the final defuzzified values of the centroids computed by the CF and the MCF algorithms. However, the dissimilarity measure $s_{\tilde{A}}$ shows that there is a substantial difference between the shapes of both centroids, mainly for smaller number of α -planes *k*=5 and higher sampling rate of 0.5.

In summary, it can be concluded that the MCF algorithm calculates numerically identical geometry of the centroid of the GT2 FS when compared to the baseline KM algorithm, while the CF algorithm only calculates an approximate result.

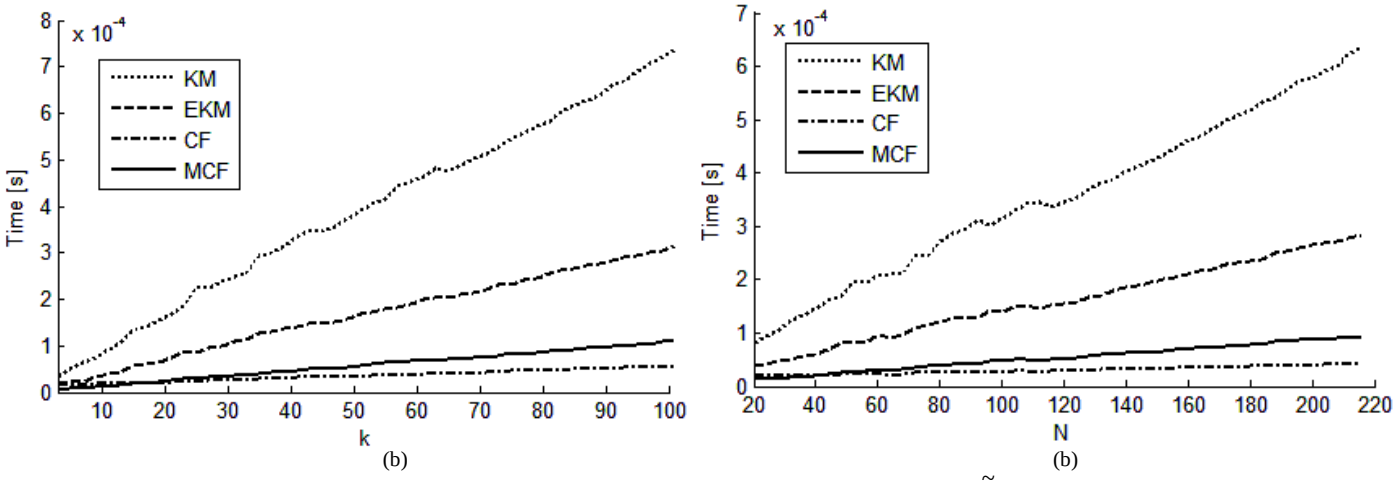
B. Computational Time

Next, the KM, EKM, CF and MCF algorithms have been compared in terms of computational speed. In order to achieve maximally objective comparison, all algorithms were implemented in C++ programming language[2]. The authors believe that C++ provides more robust environment for unbiased computation time comparison as opposed to the highly implementation-sensitive Matlab environment. The algorithms have been executed on Dell Precision M4500, Intel Core i7 CPU Q720@1.60 GHz with 8.00 GB RAM and running Windows 7.

The computational time was measured for both fuzzy sets $\tilde{F}$ and $\tilde{G}$ first with varying number *k* of α *-planes* and next with varying number *N* of discretized steps in the primary domain of variable *x*. The fuzzy sets $\tilde{F}$ and $\tilde{G}$ were constructed using $w = \{0.0, 0.25, 0.5, 0.75, 1.0\}$. Each experiment was repeated 1,000 times and the average time was recorded.

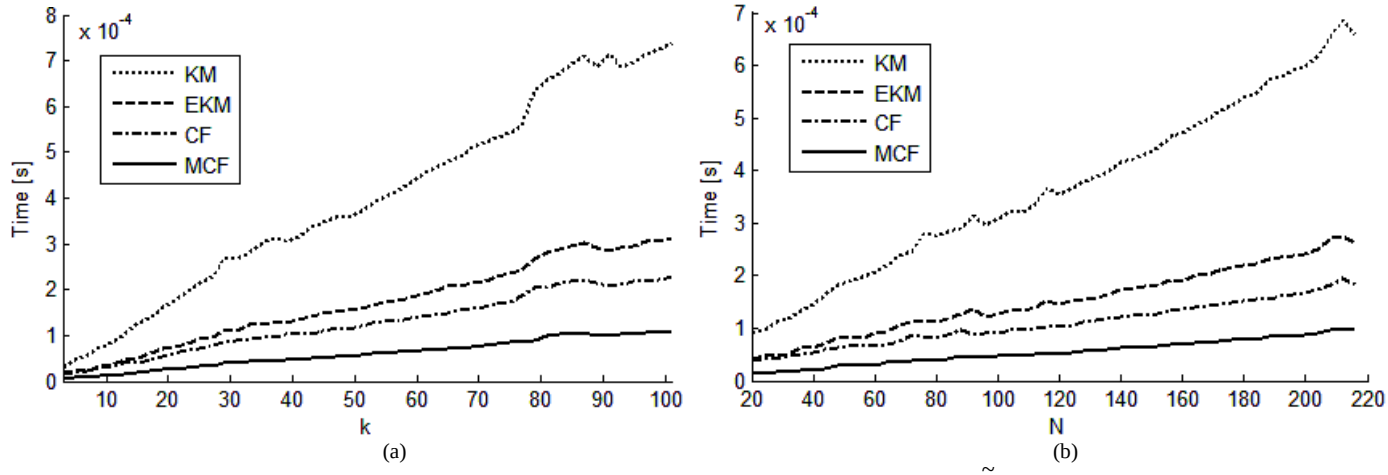
Fig. 6 shows the average computational time aggregated for all values of parameter *w* for both varying values of *k* and *N* for fuzzy set $\tilde{F}$. Several observations can be made. Both the CF and MCF algorithms are substantially faster than the independent application of the KM/EKM algorithms. Because of the trapezoidal secondary membership functions of fuzzy set $\tilde{F}$, the faster version of the CF algorithm can be used, which takes advantage of the constant derivative of the

[2] The authors rewrote the original Matlab code for the CF algorithm kindly provided by D. Zhai into C++ programming language.



(b) (b)

Fig. 6 Computational time for the KM, EKM, CF and MCF algorithms for GT2 fuzzy set $\tilde{F}$ with varying *k* and *N*.



(a) (b)

Fig. 7 Computational time for the KM, EKM, CF and MCF algorithms for GT2 fuzzy set $\tilde{G}$ with varying *k* and *N*.

TABLE II
AVERAGE COMPUTATIONAL SPEED-UP RELATIVE TO THE KM ALGORITHM

Fuzzy Set	Average Relative Speed-Up		
	EKM	CF	MCF
$\tilde{F}$ with varying *k*	56.81%	**88.51%**	85.23%
$\tilde{F}$ with varying *N*	55.10%	**90.23%**	85.10%
$\tilde{G}$ with varying *k*	56.94%	66.75%	**84.85%**
$\tilde{G}$ with varying *N*	58.00%	69.04%	**85.17%**

secondary membership function. Hence, the CF algorithm outperforms all other algorithms in terms of computational speed.

Fig. 7 shows the average computational time aggregated for all values of parameter *w* for both varying values of *k* and *N* for fuzzy set $\tilde{G}$. Due to the non-linear secondary membership function, the CF algorithm must re-compute the derivatives at each α*-plane*. This results in increased computational load and the CF algorithm only slightly outperforms the independent application of the EKM algorithm. This fact is most likely due to the more complicated calculations of the centroid derivatives. On the other hand, irrespective of the nature of the secondary membership functions, the MCF algorithm outperforms all other techniques yielding the fastest computational time for both varying number of *k* and *N*.

In addition, Table II summarizes the average relative speed-up with respect to the KM algorithm over all values of *k* or *N* for the respective fuzzy sets. As highlighted in the table, for fuzzy set $\tilde{F}$ the CF algorithm provides nearly 90% speed-up, with the MCF algorithm placing second with 85% speed-up. However, in case of the fuzzy set $\tilde{G}$, the speed-up of the CF algorithm drops under 70%, while the MCF algorithm still maintains the 85% speed-up, irrespective of the nature of the secondary membership functions.

Nevertheless, examining the actual computational time not exceeding 1*ms*, it can be said that the α*-plane* representation framework finally allows for real-time real-world applications of GT2 fuzzy logic.

VI. CONCLUSION

This paper addressed the type-reduction of GT2 FSs represented using the α*-planes* framework. The novel Monotone Centroid Flow algorithm was introduced, which leverages the derived monotone properties of the α*-plane* representation and the centroid of the GT2 FSs. When compared to other available approaches, the main advantages of the MCF algorithm are as follows: 1) the MCF algorithm calculates numerically identical centroid when compared to the baseline KM algorithm, as opposed to the approximated

centroid obtained with CF algorithm, 2) the MCF algorithm features faster computational speed when compared to the CF algorithm or the independent application of the EKM or KM algorithms, 3) the MCF algorithm does not require computation of the derivative of the centroid and the secondary membership function and is thus simpler to implement, 4) the MCF algorithm completely eliminates the need to apply the KM iterative procedure to any α *-planes* of the GT2 FS.

The accuracy and the computational time of the MCF algorithm was tested on benchmark problems and compared to other available type-reduction techniques for GT2 FSs. It was demonstrated that the MCF algorithm computes numerically identical geometry of the centroid when compared to the baseline KM algorithm. The MCF algorithm provided 85% relative computational speed-up when compared to independent application of the KM algorithm, irrespective of the nature of the secondary membership functions.

In summary, the presented MCF algorithm together with the independently developed and different approach of Yeh et al. [29] constitute the only available algorithms for computing the centroid of GT2 FSs using fast methods which take advantage of the interconnected structure of the GT2 FSs and produce numerically identical solution when compared to the EKM/KM algorithms.

APPENDIX A

The review of the KM algorithm below was adopted from [14]. The algorithm consists of two phases, which independently compute the values $c_l(\tilde{A})$ and $c_r(\tilde{A})$. The algorithm for computing the left boundary $c_l(\tilde{A})$ can be described in several steps as follows:

Step 1: Initialize a vector of weights w_i as follows:

$$w_i = \frac{1}{2}\left[\overline{\mu}_{\tilde{A}}(x_i) + \underline{\mu}_{\tilde{A}}(x_i)\right] \qquad i = 1,\ldots N \tag{A1}$$

And compute the value of *y*:

$$y = \frac{\sum_{i=1}^{N} x_i w_i}{\sum_{i=1}^{N} w_i} \tag{A2}$$

Step 2: Find switching point k ($1 \le k \le N-1$) such that

$$x_{k+1} \le y \le x_{k+1} \tag{A3}$$

Step 3: Set

$$w_i = \begin{cases} \overline{\mu}_{\tilde{A}}(x_i) & i \le k \\ \underline{\mu}_{\tilde{A}}(x_i) & i \ge k+1 \end{cases} \tag{A4}$$

And compute the value of y' as:

$$y' = \frac{\sum_{i=1}^{N} x_i w_i}{\sum_{i=1}^{N} w_i} \tag{A5}$$

Step 4: If $y' = y$, stop and set $c_l(\tilde{A}) = y$ and $L = k$. Otherwise, go to **Step 5**.

Step 5: Set $y = y'$ and go to **Step 2**.

The procedure for computing the value of $c_r(\tilde{A})$ is identical to computing $c_l(\tilde{A})$ except that in **Step 3** different update of weights w_i is used as follows:

$$w_i = \begin{cases} \underline{\mu}_{\tilde{A}}(x_i) & i \le k \\ \overline{\mu}_{\tilde{A}}(x_i) & i \ge k+1 \end{cases} \tag{A6}$$

The final output value is assigned to $c_r(\tilde{A})$ and $R=k$.

Despite Mendel and Liu proving the super-exponential convergence of the KM algorithm [35], its asymptotical complexity still remains $O(Nn)$, where N is the discretization level of the primary variable x and n denotes the beforehand unknown number of iterations of the KM algorithm.

APPENDIX B

In this section, proofs of properties, theorems and corollaries stated in this paper are presented.

A. Proof of Property 1

The proof of the containment property of α *-planes* is based on a decomposition of the GT2 FS $\tilde{A}$ into its vertical slices and then taking advantage of the interval calculus of α-cuts of the secondary membership functions. It should be recalled here, that an assumption that all secondary membership functions are convex T1 FSs have been made in Section IV.A.

Assuming that $\alpha_1 \ge \alpha_2$ the property of $\tilde{A}_{\alpha_1} \subseteq \tilde{A}_{\alpha_2}$ can be rewritten using the vertical slice representation as follows:

$$\int_{\forall x \in X} S_{\tilde{A}}(x|\alpha_1) \subseteq \int_{\forall x \in X} S_{\tilde{A}}(x|\alpha_2) \tag{B1}$$

Further decomposition yields:

$$\int_{\forall x \in X} \left(\int_{\forall u \in [s_L(x|\alpha_1), s_R(x|\alpha_1)]} u \right) \Big/ x \;\subseteq\; \int_{\forall x \in X} \left(\int_{\forall u \in [s_L(x|\alpha_2), s_R(x|\alpha_2)]} u \right) \Big/ x \tag{B2}$$

Hence, for all values of variable *x* it must hold that:

$$\int_{\forall u \in [s_L(x|\alpha_1), s_R(x|\alpha_1)]} u \subseteq \int_{\forall u \in [s_L(x|\alpha_2), s_R(x|\alpha_2)]} u \tag{B3}$$

Using the interval calculus, (B3) is equivalent to the following set of inequalities:

$$s_L(x|\alpha_1) \geq s_L(x|\alpha_2) \quad \text{(B4)}$$

$$s_R(x|\alpha_1) \leq s_R(x|\alpha_2) \quad \text{(B5)}$$

Here, recall one of the fundamental properties of α *-cuts* of T1 FS from [32]:

$$\alpha_1 \geq \alpha_2 \;\Rightarrow\; A_{\alpha_1} \subseteq A_{\alpha_2} \quad \text{(B6)}$$

It is easy to see, that by applying this property to the secondary membership functions at each vertical slice, the inequalities (B4) and (B5) are proven. This completes the proof of Property 1.

B. Proof of Property 2

The proof of the containment property of two α *-planes* centroids $C_{\tilde{A}_{\alpha_1}}(x)$ and $C_{\tilde{A}_{\alpha_2}}(x)$ takes advantage of the fact the centroid can be computed by applying the tools for type-reduction of IT2 FSs to the α *-level* T2 FSs $R_{\tilde{A}_{\alpha_1}}(x,u)$ and $R_{\tilde{A}_{\alpha_2}}(x,u)$. Assuming that $\alpha_1 \geq \alpha_2$, Property 2 can be restated as follows:

$$[c_l(\tilde{A}|\alpha_1),\, c_r(\tilde{A}|\alpha_1)] \subseteq [c_l(\tilde{A}|\alpha_2),\, c_r(\tilde{A}|\alpha_2)] \quad \text{(B7)}$$

This can be broken down into the following two inequalities:

$$c_l(\tilde{A}|\alpha_1) \geq c_l(\tilde{A}|\alpha_2) \quad \text{(B8)}$$

$$c_r(\tilde{A}|\alpha_1) \leq c_r(\tilde{A}|\alpha_2) \quad \text{(B9)}$$

First, inequality (B8) will be proven. Recall from the derivations performed by Karnik and Mendel that finding the left centroid boundary c_l for a given IT2 FS is equivalent to minimizing the value of c_l treated as a function of parameters $w_1,..., w_N$ [14]:

$$c_l(w_1,..., w_N) = \frac{\sum_{i=1}^{N} x_i\, w_i}{\sum_{i=1}^{N} w_i} \quad \text{(B10)}$$

The value of each parameter w_i is selected from the primary membership J_{x_i} of the primary variable x_i, thus subject to the following constraint:

$$w_i \in \left[s_L(x_i), s_R(x_i)\right] \quad \text{(B11)}$$

Next, following the work of Karnik and Mendel, the partial derivative of (B10) with respect to weight w_k can be expressed as [14]:

$$\frac{\partial}{\partial w_k}\left[\frac{\sum_{i=1}^{N} x_i\, w_i}{\sum_{i=1}^{N} w_i}\right] = \frac{x_k - c_l(w_1,..., w_N)}{\sum_{i=1}^{N} w_i} \quad \text{(B12)}$$

Since $\sum_{i=1}^{N} w_i > 0$, the following observation can be made [14]:

$$\frac{\partial}{\partial w_k} c_l(w_1,..., w_N) \begin{matrix} \geq \\ \leq \end{matrix} 0 \quad \text{if} \quad x_k \begin{matrix} \geq \\ \leq \end{matrix} c_l(w_1,..., w_N) \quad \text{(B13)}$$

This leads to the final observation for adjusting parameter w_k for minimizing $c_l(w_1,..., w_N)$:

If $x_k > c_l(w_1,..., w_N)$:
$c_l(w_1,..., w_N)$ **decreases** as w_k **decreases** (B14)

If $x_k < c_l(w_1,..., w_N)$:
$c_l(w_1,..., w_N)$ **decreases** as w_k **increases** (B15)

The minimum value of c_l is thus achieved by maximizing all parameters w_i to the left of c_l and minimizing all parameters w_i to the right of c_l. Using the constraint stated in (B11), the solution can be written as:

$$w_i = \begin{cases} s_R(x_i), & if \;\; x_i < c_l(w_1,..., w_N) \\ s_L(x), & if \;\; x_i > c_l(w_1,..., w_N) \end{cases}, \quad i \in \{1,..., N\} \quad \text{(B16)}$$

Next, (B16) is rewritten for the left centroid boundary $c_l(\tilde{A}|\alpha_1)$ of α *-plane* $\tilde{A}_{\alpha_1}$:

$$w_i = \begin{cases} s_R(x_i|\alpha_1), & if \;\; x_i < c_l(w_1,..., w_N|\tilde{A}_{\alpha_1}) \\ s_L(x_i|\alpha_1), & if \;\; x_i > c_l(w_1,..., w_N|\tilde{A}_{\alpha_1}) \end{cases}, i \in \{1,..., N\} \quad \text{(B17)}$$

Hence, the left centroid boundary $c_l(\tilde{A}|\alpha_1)$ cannot be further minimized because all parameters w_i are all constrained by the boundaries of α *-plane* $\tilde{A}_{\alpha_1}$ according to (B11). However, assuming that $\alpha_1 \geq \alpha_2$, the previously proven inequalities (B4) and (B5) show that for α *-plane* $\tilde{A}_{\alpha_2}$ the constraints imposed on parameters w_i can be relaxed, since:

$$[s_L(x|\alpha_1), s_R(x|\alpha_1)] \subseteq [s_L(x|\alpha_2), s_R(x|\alpha_2)] \quad \text{(B18)}$$

Hence, by following the direction of change equations (B14) and (B15), it can be observed that the left centroid boundary $c_l(\tilde{A}|\alpha_1)$ can be further minimized. Consequently $c_l(\tilde{A}|\alpha_2) \leq c_l(\tilde{A}|\alpha_1)$, which completes the proof.

The proof of inequality (B9) is identical, except that (B10) is maximized, which leads to reversed directions of the update rules in (B14), (B15).

C. Proof of Corollary 1

Corollary 1 is a direct consequent of **Property 2** and its proof is included in the preceding section in (B8) and (B9).

D. Proof of Theorem 1

For the specific case of the highest α *-plane* $\tilde{A}_1$ **Corollary 1** can be rewritten as follows:

$$c_l(\tilde{A}|\alpha) \le c_l(\tilde{A}|1) \text{ and } c_r(\tilde{A}|\alpha) \ge c_r(\tilde{A}|1) \quad \forall \alpha \in [0,1] \qquad \text{(B19)}$$

Hence, in order to prove **Theorem 1** the following set of inequalities must be verified:

$$c_l(\tilde{A}|1) \le C_{A^*} \le c_r(\tilde{A}|1) \qquad \text{(B20)}$$

Recall that according to (29), for the general case of any secondary membership functions having an interval core, the value of the principal membership function $A^*(x)$ at coordinate *x* is computed as:

$$A^*(x) = \frac{\overline{\mu}_{\tilde{A}}(x|1) + \underline{\mu}_{\tilde{A}}(x|1)}{2} \qquad \text{(B21)}$$

Hence, using the α-cuts notations introduced in (9) the above equation (B21) can be rewritten as:

$$A^*(x) \in [s_L(x|1), s_R(x|1)] \qquad \text{(B22)}$$

Equation (B22) shows that the principal membership function $A^*(x)$ is fully contained within the highest α*-plane* $\tilde{A}_1$:

$$A^* \subseteq \tilde{A}_1 \qquad \text{(B23)}$$

Upon deriving (B23), the proof of **Corollary 2** can be applied leading to verifying the inequality in (B20) and thus proving **Theorem 1**.

E. Proof of Theorem 2

The proofs of (34), (35), (36) and (37) in Theorem 2 are very similar. Therefore only proof of (34) is provided here. Recall that $E_{1,\hat{L}}$ in (34) stands for the numerator in (32). Hence:

$$E_{1,\hat{L}} = \sum_{i=1}^{\hat{L}} x_i \, \overline{\mu}_{\tilde{A}}(x_i|\alpha_{t-1}) + \sum_{i=\hat{L}+1}^{N} x_i \, \underline{\mu}_{\tilde{A}}(x_i|\alpha_{t-1}) \qquad \text{(B24)}$$

For a decremented left switch point value $\hat{L}-1$ it is:

$$E_{1,\hat{L}-1} = \sum_{i=1}^{\hat{L}-1} x_i \, \overline{\mu}_{\tilde{A}}(x_i|\alpha_{t-1}) + \sum_{i=\hat{L}}^{N} x_i \, \underline{\mu}_{\tilde{A}}(x_i|\alpha_{t-1}) \qquad \text{(B25)}$$

By subtracting and adding the $\hat{L}^{th}$ element to and from both the left and the right sum in (B25), the value of $E_{1,\hat{L}-1}$ will remain unchanged:

$$\begin{aligned} E_{1,\hat{L}-1} = &\sum_{i=1}^{\hat{L}-1} x_i \, \overline{\mu}_{\tilde{A}}(x_i|\alpha_{t-1}) + x_{\hat{L}} \, \overline{\mu}_{\tilde{A}}(x_{\hat{L}}|\alpha_{t-1}) - x_{\hat{L}} \, \overline{\mu}_{\tilde{A}}(x_{\hat{L}}|\alpha_{t-1}) \\ &+ \sum_{i=\hat{L}}^{N} x_i \, \underline{\mu}_{\tilde{A}}(x_i|\alpha_{t-1}) + x_{\hat{L}} \, \underline{\mu}_{\tilde{A}}(x_{\hat{L}}|\alpha_{t-1}) - x_{\hat{L}} \, \underline{\mu}_{\tilde{A}}(x_{\hat{L}}|\alpha_{t-1}) \end{aligned} \qquad \text{(B26)}$$

The above expression can be simplified as:

$$\begin{aligned} E_{1,\hat{L}-1} = &\sum_{i=1}^{\hat{L}} x_i \, \overline{\mu}_{\tilde{A}}(x_i|\alpha_{t-1}) - x_{\hat{L}} \, \overline{\mu}_{\tilde{A}}(x_{\hat{L}}|\alpha_{t-1}) \\ &+ \sum_{i=\hat{L}+1}^{N} x_i \, \underline{\mu}_{\tilde{A}}(x_i|\alpha_{t-1}) + x_{\hat{L}} \, \underline{\mu}_{\tilde{A}}(x_{\hat{L}}|\alpha_{t-1}) \end{aligned} \qquad \text{(B27)}$$

Note that the initial and the final indexes in the first and the second sums changed. By reordering (B27) and further simplifying the expression, it can be concluded that:

$$\begin{aligned} E_{1,\hat{L}-1} = &\sum_{i=1}^{\hat{L}} x_i \, \overline{\mu}_{\tilde{A}}(x_i|\alpha_{t-1}) + \sum_{i=\hat{L}+1}^{N} x_i \, \underline{\mu}_{\tilde{A}}(x_i|\alpha_{t-1}) \\ &- x_{\hat{L}}(\overline{\mu}_{\tilde{A}}(x_{\hat{L}}|\alpha_{t-1}) - \underline{\mu}_{\tilde{A}}(x_{\hat{L}}|\alpha_{t-1})) \end{aligned} \qquad \text{(B28)}$$

By substituting (B24) into (B28) the proof of (34) is completed:

$$E_{1,\hat{L}-1} = E_{1,\hat{L}} - x_{\hat{L}}(\overline{\mu}_{\tilde{A}}(x_{\hat{L}}|\alpha_{t-1}) - \underline{\mu}_{\tilde{A}}(x_{\hat{L}}|\alpha_{t-1})) \qquad \text{(B29)}$$

ACKNOWLEDGMENT

The authors would like to acknowledge the help of Daoyuan Zhai who generously provide the Matlab code for the Centroid Flow algorithm. The authors would like to also thank the anonymous reviewers who significantly helped to improve the quality of this manuscript.

REFERENCES

[1] J. M. Mendel, *Uncertain Rule-Based Fuzzy Logic Systems: Introduction and New Directions*, Prentice-Hall, Upper Saddle River, NJ, 2001.

[2] J. M. Mendel, "Advances in type-2 fuzzy sets and systems," in *Information Sciences*, vol. 177, pp. 84-110, 2007.

[3] H. Hagras, "Type-2 FLCs: A New Generation of Fuzzy Controllers," in *IEEE Computational Intelligence Magazine*, pp. 30-43, February 2007.

[4] N. N. Karnik, J. M. Mendel, "Type-2 Fuzzy Logic Systems," in *IEEE Trans. on Fuzzy Systems*, vol. 7, no. 6, pp. 643-658, December 1999.

[5] S. Coupland, R. John, "Geometric Type-1 and Type-2 Fuzzy Logic Systems," in *IEEE Trans. on Fuzzy Systems*, vol. 15, no. 1, pp. 3-15, February 2007.

[6] L. A. Zadeh, "The Concept of a Linguistic Variable and its Approximate Reasoning - II," in *Information Sciences*, No. 8, pp. 301-357, 1975.

[7] H. A. Hagras, "A Hierarchical Type-2 Fuzzy Logic Control Architecture for Autonomous Mobile Robots," in *IEEE Trans. Fuzzy Systems*, vol. 12, no. 4, pp. 524-539, 2004.

[8] J. Figueroa, J. Posada, J. Soriano, M. Melgarejo, S. Rojas, "A type-2 fuzzy logic controller for tracking mobile objects in the context of robotic soccer games," in *Proc. IEEE Intl' Conf. on Fuzzy Systems*, pp. 359-364, 2005.

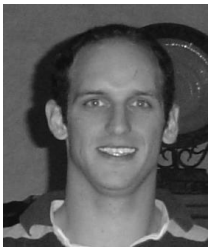
[9] Ch. Lynch, H. Hagras, "Using Uncertainty Bounds in the Design of an Embedded Real-Time Type-2 Neuro-Fuzzy Speed Controller for Marine Diesel Engines," in *Proc. IEEE Intl' Conf. on Fuzzy Systems*, pp. 1446-1453, Vancouver, Canada, 2006.

[10] O. Linda, M. Manic, "Uncertainty-Robust Design of Interval Type-2 Fuzzy Logic Controller for Delta Parallel Robot," in *IEEE Trans. On Industrial Informatics*, vol. 7, no. 4, pp. 661-671, Nov. 2011.

[11] O. Linda, M. Manic, "Interval Type-2 Fuzzy Voter Design for Fault Tolerant Systems," in *Information Sciences*, vol. 181, issue: 14, pp. 2933-2950, July 2011..

[12] A. Niewiadomski, "On Finity, Countability, Cardinalities, and Cylindric Extensions of Type-2 Fuzzy Sets in Linguistic Summarization of Databases, " in *IEEE Trans. on Fuzzy Systems*, vol. 18, no. 3, pp. 532-545, June 2010.

[13] D. Wu, J. M. Mendel, "On the Continuity of Type-1 and Interval Type-2 Fuzzy Logic Systems," in *IEEE Trans. on Fuzzy Systems*, vol. 19, no. 1, pp. 179-192, Feb. 2011.
[14] N. Karnik, J. M. Mendel, "Centroid of a type-2 fuzzy set," in *Information Sciences*, vol. 132, pp. 195-220, 2001.
[15] J. M. Mendel, R. John, F. Liu, "Interval Type-2 Fuzzy Logic Systems Made Simple," in *IEEE Trans. on Fuzzy Systems*, vol. 14, no. 6, pp. 808-821, 2006.
[16] F. Liu, "An efficient centroid type-reduction strategy for general type-2 fuzzy logic system," in *Information Sciences*, vol. 178, pp. 2224-2236, 2008.
[17] J. M. Mendel, F. Liu, D. Zhai, "α -Plane Representation for Type-2 Fuzzy Sets: Theory and Applications," in *IEEE Transaction on Fuzzy Systems*, vol. 17, no. 5, pp. 1189-1207, October 2009.
[18] Ch. Wagner, H. Hagras, "Toward General Type-2 Fuzzy Logic Systems Based on zSlices," in *IEEE Transaction of Fuzzy Systems*, vol. 18, no. 4, pp. 637-660, August, 2010.
[19] J. M. Mendel, "Comments on "α -Plane Representation for Type-2 Fuzzy Sets: Theory and Applications"," in *IEEE Trans. on Fuzzy Systems*, vol. 18, no. 1, pp. 229-230, Feb. 2010.
[20] X. Liu, J. M. Mendel, "Connect Karnik-Mendel Algorithms to Root-Finding for Computing the Centroid of an Interval Type-2 Fuzzy Set," in *IEEE Trans. on Fuzzy Systems*, vol. 19, no. 4, pp. 652-665, Aug. 2011.
[21] D. Zhai, J. M. Mendel, "Centroid of a General Type-2 Fuzzy Set Computed by Means of the Centroid-Flow Algorithm," in *Proc. WCCI 2010*, Barcelona, Spain, pp. 895-902, July, 2010.
[22] D. Zhai, J. M. Mendel, "Computing the Centroid of a General Type-2 Fuzzy Set by Means of the Centroid-Flow Algorithm," in *IEEE Trans. on Fuzzy Systems*, vol: 19, issue: 3, pp. 401-422, June 2011.
[23] J. M. Mendel, R. I. John, "A fundamental decomposition of type-2 fuzzy sets," in *Proc. of IFSA World Congress and 20th NAFIPS International Conference*, pp.1896-1901, Canada, July, 2001.
[24] H. Tahayori, A. G. B. Tettamanzi, G. D. Antoni, "Approximated Type-2 Fuzzy Set Operations," in *Proc. IEEE International Conference in Fuzzy Systems*, Canada, pp. 1910-1917, 2006.
[25] D. Wu, J. M. Mendel, "Enhanced Karnik-Mendel Algorithms", in *IEEE Transaction on Fuzzy Systems*, vol. 17, No. 4, pp. 923-934, August, 2009.
[26] S. Greenfield, R John, S. Coupland, "A Novel Sampling Method for Type-2 Defuzzification," in *Proc. UKCI 06*, pp. 120-127, 2005.
[27] S. Greenfield**,** F. Chiclana, S. Coupland, R. John, "Type-2 Defuzzification: Two Contrasting Approaches." in *Proc. of IEEE World Congress on Computational Intelligence*, Barcelona, Spain, pp. 3183-3189, June 2010.
[28] S. Greenfield, F. Chiclana, S. Coupland, R. John, "The collapsing method of defuzzification for discretized interval type-2 fuzzy sets," in *Information Sciences*, vol. 179, issue: 13, pp. 2055-2069, June 2009.
[29] C.Y. Yeh, W. H. Jeng, S. J. Lee, "An Enhanced Type-Reduction Algorithm for Type-2 Fuzzy Sets, " in *IEEE Trans. on Fuzzy Systems*, vol. 19, issue: 2, pp. 227-240, April 2011.
[30] N. N. Karnik, J. M. Mendel, "Operations on type-2 fuzzy sets," *in Fuzzy Sets and Systems*, vol. 122, issue: 2, pp. 327-348, Sep. 2001.
[31] H. Hamrawi, S. Coupland, "Non-specificity Measures for Type-2 Fuzzy Sets," in *Proc. IEEE-FUZZ*, Korea, pp. 732-737, August 2009
[32] G. Klir, B. Yuan, *Fuzzy Sets and Fuzzy Logic: Theory and Applications*, Prentice Hall, Upper Saddle River, NJ, 1995.
[33] K. Duran, H. Bernal, M. Melgarejo, "Improved iterative algorithm for computing the generalized centroid of an interval type-2 fuzzy set," in *Proc. Annual Meeting of the North American Fuzzy Information Processing Society – NAFIPS*, pp. 1-5, May 2008.
[34] P. Jaccard, "Nouvelles recherches sur la distribution florale," in *Bulletin de la Societe de Vaud des Sciences Naturelles*, vol. 44, pp. 223, 1908.
[35] J. M. Mendel, F. Liu, "Super-Exponential Convergence of the Karnik-Mendel Algorithms for Computing the Centroid of an Interval Type-2 Fuzzy Set," in *IEEE Trans. on Fuzzy Systems*, vol. 15, no. 2, pp. 309-320, April, 2007.



Ondrej Linda, (S'09) received his M.Sc. in Computer Graphics from Czech Technical University in Prague in 2010, and M.Sc. in Computational Intelligence from the University of Idaho at Idaho Falls in 2009. He received his B.Sc. in Electronic Engineering and Informatics from Czech Technical University in Prague in 2007. He is currently a Doctoral student at the University of Idaho in Idaho Falls. His research experience includes research assistant positions at Kansas State University and at the University of Idaho, and an internship with the Robotics Group at the Idaho National Laboratory. His fields of interest included machine learning, pattern recognition, intelligent control systems, data mining and computer graphics.

Dr. Milos Manic, (S'95-M'05-SM'06), Dr. Milos Manic, IEEE Senior Member, has been leading Computer Science Program at Idaho Falls and is a Director of Modern Heuristics Research Group. He received his Ph.D. degree in Computer Science from University of Idaho, Computer Science Dept. He received his M.S. and Dipl.Ing. in Electrical Engineering and Computer Science from the University of Nis, Faculty of Electronic Engineering, Serbia. He has over 20 years of academic and industrial experience, including an appointment at the ECE Dept. and Neuroscience program at University of Idaho. As university collaborator or principal investigator he lead number of research grants with the Idaho National Laboratory, NSF, EPSCoR, Dept. of Air Force, and Hewlett-Packard, in the area of data mining and computational intelligence applications in process control, network security and infrastructure protection. Dr. Manic is an Administrative Committee Member for the IEEE Industrial Electronics Society and member of several technical committees and boards of this Society. Dr. Manic has published over hundred refereed articles in international journals, books, and conferences.