Geometric Logical Operations for Type-2 Fuzzy Sets

Simon Coupland
Centre for Computational Intelligence
De Montfort University
The Gateway
Leicester, LE1 9BH
United Kingdom
simonc@dmu.ac.uk

Robert John
Centre for Computational Intelligence
De Montfort University
The Gateway
Leicester, LE1 9BH
United Kingdom
rij@dmu.ac.uk

Abstract

This paper presents a series of geometric definitions and algorithms that together form a type-2 geometric fuzzy inference system which can operate over a truly continuous domain, with no need for discretisation at any stage.

Keywords: Type-2 fuzzy logic, Computational geometry, Geometric inference.

1 Introduction

In all fuzzy logic applications there is an (usually unwritten) assumption that the final implementation will use discrete fuzzy sets. This assumption is a direct consequence of the method by which fuzzy system are implemented on computer systems. The authors have previously defined methods rooted in geometry, to define type-1 fuzzy systems over continuous domains that may be easily implemented on a computer system [2]. This paper extends this previous work by exploring new ways of defining and computing the logical operations on a type-2 fuzzy set. The definitions and algorithms given here result from the combination of the geometric approach to fuzzy logic and the optimised join and meet operations. The result is a series of methods that allow logical operations for type-2 fuzzy sets to be defined for the entire membership function of that set, rather than at discrete points. The outcome of this paper (in combination with previous work [2, 3, 1]) is a type-2 fuzzy inference methodology that can operate over truly continuous universes of discourse.

The remainder of this paper is structured as follows: Section 2 presents type-2 fuzzy sets, Section 3 presents geometric type-2 fuzzy sets, Section 4 presents novel geometric inference techniques and Section 5 presents the conclusions and outcomes of this paper.

2 Type-2 Fuzzy Sets

Type-2 fuzzy sets come in two flavours, generalised and interval. Throughout this paper we only ever deal with generalised type-2 fuzzy sets. Both generalised and interval type-2 fuzzy sets are extensions of type-1 fuzzy sets. The membership grade of a generalised type-2 fuzzy set is a type-1 fuzzy set with a support bounded by the interval $[0, 1]$. The membership grade of an interval type-2 fuzzy set is an interval of real numbers $\in [0, 1]$. We are interested in generalised type-2 fuzzy sets as we believe they offer greater expressive power that interval type-2 fuzzy set. However, type-2 fuzzy logic has traditionally suffered from a significantly high level of computational complexity. The geometric defuzzifier [1] has overcome these problems and additionally forms part of the geometric approach, which this work forms part of.

2.1 Inference Operations of Type-2 Fuzzy Sets

The inference engine in a type-2 fuzzy logic system uses only two logical operators, the join and the meet, to perform all inference operations. Antecedent combination, rule implication and the combination of consequents can all be defined with join and meet alone. The logical connectives, the ‘or’ and ‘and’ of type-2 fuzzy sets were given by Zadeh [10]. Mizumoto and Tanaka [7] renamed these operations the ‘join’ and ‘meet’ and were the first to look at the properties of these operations. Mizumoto and Tanaka [7], along with Dubois and Prade [4], also discuss the use of different t-norm and t-conorm operators. Karnik and Mendel [6] defined more computationally efficient methods for calculating the join and meet of secondary membership functions that are normal and convex. This paper uses Karnik and Mendel’s operations as the basis for geometric inference operations presented in this paper.

The join ($\sqcup$) operation finds the conjunction of two secondary membership functions $\mu_{\widetilde{A}}(x)$ and $\mu_{\widetilde{B}}(x)$. Let $\mu_{\widetilde{A}}(x) = \sum_{i=1}^{M} \alpha_i/v_i$ and let $\mu_{\widetilde{B}}(x) = \sum_{j=1}^{N} \beta_j/w_j$. The conjunction of $\mu_{\widetilde{A}}(x)$ and $\mu_{\widetilde{B}}(x)$ is given by

$$\mu_{\widetilde{A}\sqcup\widetilde{B}}(x) = \sum_{i=1}^{M}\sum_{j=1}^{N}(\alpha_i \star \beta_j)/(v_i \vee w_j) \quad (1)$$

where $\vee$ is the t-conorm, generally taken to be maximum and $\star$ is a t-norm such as minimum or product.

The meet ($\sqcap$) operation finds the disjunction of two secondary membership functions $\mu_{\widetilde{A}}(x)$ and $\mu_{\widetilde{B}}(x)$. The disjunction of $\mu_{\widetilde{A}}(x)$ and $\mu_{\widetilde{B}}(x)$ is given by

$$\mu_{\widetilde{A}\sqcap\widetilde{B}}(x) = \sum_{i=1}^{M}\sum_{j=1}^{N}(\alpha_i \star \beta_j)/(v_i \star w_j) \quad (2)$$

where again $\star$ is a t-norm.

It was mentioned earlier that the computational overheads that come with type-2 fuzzy logic are much greater than type-1. The origins of some of these issues are now discussed. The basic logical operations are far more complex than the type-1 equivalent. Assume that t-norms and t-conorms require an equal amount of processing resource t, then cost of finding the ‘and’ or ‘or’ of two discrete type-1 fuzzy sets at a point x is t. The cost of finding the ‘and’ or ‘or’ of two discrete type-2 fuzzy sets at a point x is $2MNt$, where M and N are the number of discrete points in the domain of the respective secondary memberships. Karnik and Mendel [6] gave methods to reduce this overhead significantly. These methods rely on the secondary membership functions being both normal and convex. If the condition of normality and convexity are not met then incorrect results are produced. The optimised join and meet operations [6] are now given. Let there be n convex, normal, type-1 real fuzzy sets $F_1, \ldots, F_n$ characterized by membership functions $f_1, \ldots, f_n$, respectively. Let $v_1, v_2, \ldots, v_n$ be real numbers such that $v_1 \leq v_2 \leq \ldots v_n$ and $f_1(v_1) = f_2(v_2) = \ldots = f_n(v_n) = 1$. Then restricting the t-conorm to maximum ($\vee$) and the t-norm to minimum ($\wedge$) gives,

$$\mu_{\sqcup_{i=1}^{n} F_i}(\theta) = \begin{cases} \wedge_{i=1}^{n} f_i(\theta), & \theta < v_1, \\ \wedge_{i=k+1}^{n} f_i(\theta), & v_k \leq \theta < v_{k+1}, \\ & \quad 1 \leq k \leq n-1, \\ \vee_{i=1}^{n} f_i(\theta), & \theta \geq v_n \end{cases} \quad (3)$$

and

$$\mu_{\sqcap_{i=1}^{n} F_i}(\theta) = \begin{cases} \vee_{i=1}^{n} f_i(\theta), & \theta < v_1, \\ \wedge_{i=k+1}^{n} f_i(\theta), & v_k \leq \theta < v_{k+1}, \\ & \quad 1 \leq k \leq n-1, \\ \wedge_{i=1}^{n} f_i(\theta), & \theta \geq v_n \end{cases} \quad (4)$$

These definitions significantly reduce the computational cost of the join and meet operations. The reduction in computational cost is achieved by reducing the number of times the domain of the sets have to be traversed to one. Prior to this definition the number of times that the domain of each secondary membership function had to be traversed was equal to the number of points in other secondary membership function it is being combined with. Novel extensions to the Karnik and Mendel optimisations which reduce the limitations of these operations were given by the authors [3]. This work underpins the geometric inference process given in this paper. The geometric operations presented in 3 are based on equations 3 and 4.

Rule antecedents can be combined using the join and meet operators. In a Mamdani system, these computed antecedents can then be used to find the implication of the antecedent to a rule consequent for each of the rules. Implication is performed by finding the meet of the antecedent with every point in the consequent type-2 fuzzy set. Let the antecedent value be $\mu_{\widetilde{A}}(x_1) \sqcap \mu_{\widetilde{B}}(x_2)$ and the consequent be $\widetilde{C}$ over the domain Y. The value of

$\mu_{\widetilde{A}}(x_1) \sqcap \mu_{\widetilde{B}}(x_2) \Rightarrow \widetilde{C}$ is given below.

$$\mu_{\widetilde{A}}(x_1) \sqcap \mu_{\widetilde{B}}(x_2) \Rightarrow \widetilde{C} = \int_{y \in Y} \left(\mu_{\widetilde{A}}(x_1) \sqcap \mu_{\widetilde{B}}(x_2)\right) \sqcap \mu_{\widetilde{C}}(y) \tag{5}$$

Each rule will produce an inferred consequent type-2 fuzzy set. To combine these sets the 'or' operation, the join operation, is applied at every point in the domain of the sets. Let consequent sets be $\widetilde{C_1}, \widetilde{C_2}, \ldots, \widetilde{C_n}$ and the final combined set be $\widetilde{F}$, all over domain X. The value of $\widetilde{F}$ is given below.

$$\widetilde{F} = \int_{x \in X} \bigsqcup_{i=1}^{n} \widetilde{C_i}(x) \tag{6}$$

This operation results in a single type-2 fuzzy set which represents the decision of the fuzzy logic system. A crisp output must now be derived which is representative of this final fuzzy set. This work sets out how to arrive at this result using purely geometric techniques, preserving the accuracy of a continuous domain.

3 Geometric Type-2 Fuzzy Sets

A type-2 fuzzy set is characterised by a type-2 fuzzy membership function. A geometric type-2 fuzzy set [1] is simply a type-2 fuzzy set where the membership function is modelled using geometric primitives. A generalised type-2 fuzzy set will require 3-dimensional geometric primitives. For reasons for computational simplicity, we strongly advocate the use of triangles for geometrically modelling a generalised type-2 fuzzy membership function.

Definition 1 *A geometric type-2 fuzzy set is defined as a collection of n triangles in 3D space where the edges of these triangles connect to form a 3D polyhedron, i.e.:*

$$\widetilde{A} = \bigcup_{i=1..n} \begin{bmatrix} x_1^i & y_1^i & z_1^i \\ x_2^i & y_2^i & z_2^i \\ x_3^i & y_3^i & z_3^i \end{bmatrix} \tag{7}$$

where x_1^i, x_2^i and $x_3^i \in X$ and y_1^i, y_2^i, y_3^i, z_1^i, z_2^i and $z_3^i \in [0,1]$. In this geometric model, values on the y axis represent primary membership grades and values on the z axis represent secondary membership grades.

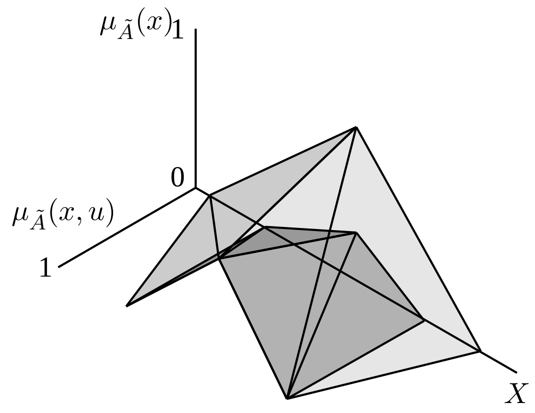

Figure 1: The Geometric Type-2 Fuzzy Set $\widetilde{Moderate}$.

An example geometric type-2 $\widetilde{Moderate}$ is depicted in Figure 1. The membership function of $\widetilde{Moderate}$ is a polyhedron, in this case made up of eight triangles. These eight triangles approximate the membership function of $\widetilde{Moderate}$ over a continuous domain X. The polyhedron provides an approximation as the actual membership function of $\widetilde{Moderate}$ is a non-planar surface that is being modelled by planar triangles. The accuracy of this approximation will depend upon the configuration of the triangles that make up the polyhedron.

4 Type-2 Geometric Inference

We now present the *and* and *or* logical operations for type-2 fuzzy sets using three dimensional geometric techniques which are equivalent to the discrete operations using only minimum and maximum.

4.1 Geometry Primer

Geometric type-2 fuzzy sets, presented in Section 3, are defined as a collection of 3D triangles. TO make the simplify the notation, when defining logical operation we consider the set of triangles that define a given set $\widetilde{A}$ to be separated in to two subsets. The set of triangle that form the upper surface denoted $\overline{\widetilde{A}}$ and the triangles that form the lower surface denoted $\underline{\widetilde{A}}$. Since all triangles have counter-clockwise ordered vertices it is relatively simple to identify which surface a triangle belongs to. If the normal of the triangle has a pos-

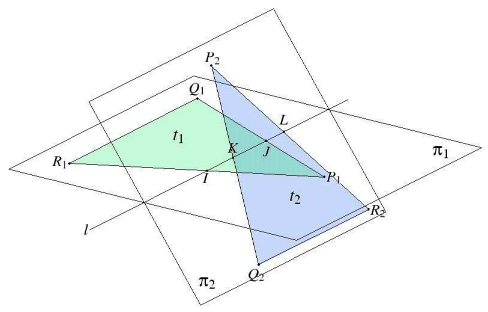

Figure 2: Intersecting Triangles and The Planes in Which They Lie. Adapted from [5]

itive y component then it is in the upper surface, if negative then it is in the lower surface.

The 2D geometric logic operations given by the authors in [2, 3] were based on the ability to identify points where two line segments intersect. These lines could then be 'clipped' to give a new geometric fuzzy set, the result of a logical operation. The type-2 geometric logic operations use the same principle only with 3D surfaces (formed by triangles) instead of 2D line segments. Such an approach will require a geometric operation for identifying the line where two triangles intersect and an operation for clipping the triangles accordingly. The notion of 'clipping' is taken from computational geometry [3] where one geometric primitive may be clipped against another, that is, one primitives form is truncated by another.

The Guigue and Devillers triangle-triangle overlap test [5], an extension of Möllers intersection test [8], provides a computationally fast method for testing for and calculating any points where two triangles intersect. Consider the two triangles t_1 with vertices P_1, Q_1 and R_1, and t_2 with vertices P_2, Q_2 and R_2 on the respective planes π_1 and π_2 depicted in Figure 2. The algorithm begins by testing whether t_1 intersects with π_2 and whether t_2 intersects with π_1. This is done by comparing the distance from each vertex to the plane of the opposing triangle. First take the distances from P_1, Q_1 and R_1 to the plane π_2, denoted dp_1, dq_1 and dr_1. When the signs of dp_1, dq_1 and dr_1 are compared three distinct situations can be identified. If all three have the same sign, then t_1 lies in one of the half spaces of π_2. If all three are zero then the t_1 and t_2 lie on the same plane and can therefore by handled using 2-dimensional methods. If one of the distances has a different sign then t_1 intersects the plane π_2 and the algorithm continues. The same is done for the distances dp_2, dq_2 and dr_2. If one of the distances has a different sign then the algorithm continues as each triangle must intersect the plane of the opposing triangle. The next stage involves rotating the vertices of each triangle in such a way that P_1 lies on the opposite side of π_2 than Q_1 and R_1 and P_2 lies on the opposite side of π_1 than q_2 and R_2. Also P_2, Q_2 and R_2 are ordered counter clockwise with respect to P_1 and P_1, Q_1 and R_1 are ordered counter clockwise with respect to P_2. This is the vertex ordering depicted in Figure 2. The line l depicts the line where the planes π_1 and π_2 intersect. There must now be a point along each of vectors $\vec{p_1q_1}$, $\vec{p_1r_1}$, $\vec{p_2q_2}$ and $\vec{p_2r_2}$ that intersects the line l. These points are denoted I, J, K and L respectively. A series of boolean operations then finds the order of I, J, K and L along l. If this ordering confirms that the triangles do intersect then intersection points are calculated. In the case of t_1 and t_2 the points follow the ordering I, J, K and L giving the intersection points K and J. The calculations to give the points K and J are given below.

$$K = P_2 - (\vec{p_2} - \vec{q_2})\frac{(\vec{p_2} - \vec{p_1}) \cdot \vec{n_1}}{(\vec{p_2} - \vec{q_2}) \cdot \vec{n_1}} \quad (8)$$

$$J = P_1 - (\vec{p_1} - \vec{p_2})\frac{(\vec{p_1} - \vec{p_2}) \cdot \vec{n_2}}{(\vec{p_1} - \vec{q_1}) \cdot \vec{n_2}} \quad (9)$$

Where $\vec{n_1}$ and $\vec{n_2}$ are the normals to the respective planes π_1 and π_2, $\cdot$ is the dot product of two vectors and $\vec{p_i}$ is the position vector of P_i. This algorithm provides a method for testing for and calculating the intersection points of two triangles in 3-dimensions. The use of a series of single geometric tests to ascertain the way the two triangles are interacting simplifies the intersection point calculation and reduces the possibility of errors from floating point calculations.

Having found a way to test for and identify intersecting triangles we need an algorithm to clip the triangles at the intersection lines. No suitable algorithm could be found in the literature, instead we present the following novel surface clipping algorithm. The algorithm is based on the principle that maximum (in the y dimension) of any two surfaces is given by the maximum (in the

y dimension) of all the points that make up that surface. It is therefore possible to find the minimum of maximum of two surfaces by calculating the segments where the two surfaces intersect. This is analogous to the Weiler-Atherton [9] algorithm which calculates all points where two polygons intersect in order clip one polygon against the other.

The Surface Clipping Algorithm

```
Inputs: the surfaces $subject$ and $clip$. $clip$ is the surface that is being clipped against.
Let there also be two lists of triangles - $currentsubject$ and $currentclip$.
These lists hold triangles where intersections are possible.
Let the minimum $x$ component of the three vertices of the triangle $t$ be $min.x$ and maximum $max.x$.
order all triangles in both the surfaces by $min.x$.
while not off both the surfaces $subject$ and $clip$. do
  take a triangle $t$ from the head of $subject$ or $clip$, whichever has the lower value of $min.x$; add $t$ to respective current list.
  check whether any triangles can be removed from current lists i.e, check whether the $max.x$ value for that triangle is less than the $min.x$ of $t$.
  if $t$ is from $subject$ then

    if $t$ is below all triangles in the clip list then
      append $t$ to $clipped$.
    end if
    if $t$ intersects with any of the triangles in the clip list then
      area below the intersection line must be output:
      if only one vertex is below the intersection line then
        only one triangle is output. This triangle consist of that vertex and the end points of the triangle intersection line.
      end if
      if two vertices lie below the intersection line then
        two triangles need to be added. One triangle consists the intersection line endpoints and one of the vertices. The other triangle consists of one of the intersection line vertices and the vertices that are below the intersection line.
      end if
      if either of the intersection line endpoints is inside $t$ then
        $t$ must be replaced by triangles that cover the area of $t$ that did not intersect.
      end if
    end if
  end if
end while
Outputs: the surface $clipped$.
```

This algorithm gives the minimum of two surfaces. The *below* operations can be replaced with *above* operations and this will give the maximum of two surfaces.

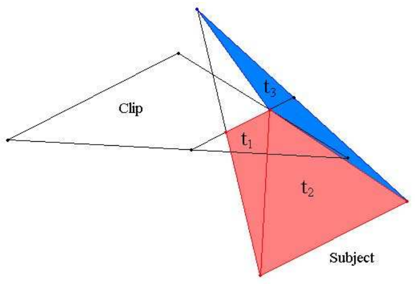

Figure 3: A Clipped Triangle and the Resultant Triangles t_1, t_2 and t_3.

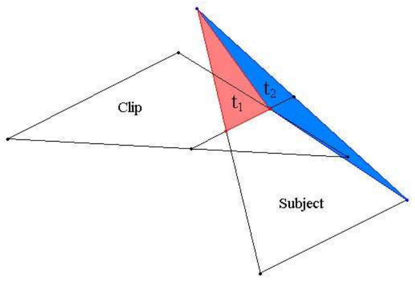

Figure 4: A Clipped Triangle and the Resultant Triangles t_1 and t_2.

Figure 3 depicts a clipping operation using *below* where two intersecting triangles and the resultant

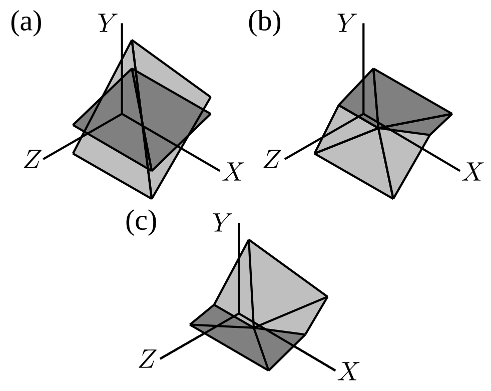

Figure 5: (a) Two Surfaces. (b) The Minimum of those Two Surfaces. (c) The Maximum of those Two Surfaces.

triangles t_1 and t_2 which are appended to $clipped$ and t is replaced by t_3 in the respective lists. Figure 4 depicts a clipping operation using *above* where two intersecting triangles and the resultant triangles t_1 and t_2 which are appended to $clipped$ and t is replaced by t_3 in the respective lists. Each of these operations can be applied to collections of triangles, giving clipping operations over entire surfaces Figure 5 shows how clipping operation may be used to find either the minimum or maximum of two surfaces.

These clipping operations can implement the join and meet operations as defined in equations (3) and (4), where minimum and maximum are the only permitted t-norm and t-conorm. The surfaces depicted in Figure 5 (a) could easily be from the upper surface of a geometric type-2 fuzzy set. With reference to the join and meet operations, these surfaces capture all points where $x < \upsilon_1$ and $\upsilon_1 \leq x < \upsilon_2$. Equation (4) tells us that to perform meet operation on these points we need to take the minimum secondary grade at each point in the sets where $x < \upsilon_1$ and $\upsilon_1 \leq x < \upsilon_2$. This operation is depicted in Figure 5 (b), where the minimum of the two surfaces is found using the surface clipping algorithm.

This operation also gives the meet for all points less than the first apex point, which are captured by the lower surface of a geometric type-2 fuzzy set. Equation (3) tells us that to perform join operation on these points we need to take the maximum secondary grade at every point in the sets where $x < \upsilon_1$ and $\upsilon_1 \leq x < \upsilon_2$. This operation is depicted in Figure 5 (c), where the maximum

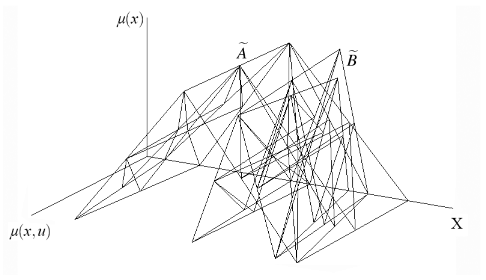

Figure 6: The Geometric Type-2 Fuzzy Sets $\widetilde{A}$ and $\widetilde{B}$.

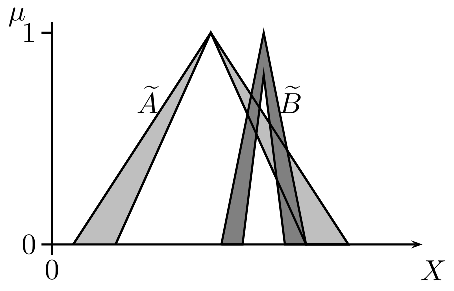

Figure 7: The FOU of the Geometric Type-2 Fuzzy Sets $\widetilde{A}$ and $\widetilde{B}$.

of the two surfaces is found using the surface clipping algorithm. This operation is also used to find the join for the lower surfaces of a pair of geometric type-2 fuzzy sets.

The logical *and*, *or* and *implies* will now be defined using as examples the geometric type-2 fuzzy sets $\widetilde{A}$ and $\widetilde{B}$ depicted in full in Figure 6 and the associated FOUs in Figure 7.

4.2 The Geometric And Operator

The *and* of two type-2 fuzzy sets is defined as the result of taking the meet of the secondary membership functions of the two sets at each point in

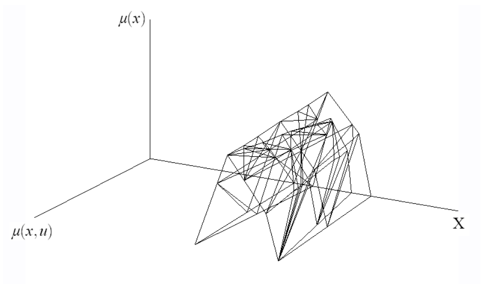

Figure 8: The Geometric Type-2 Fuzzy Set $\widetilde{C} = \widetilde{A} \cap \widetilde{B}$.

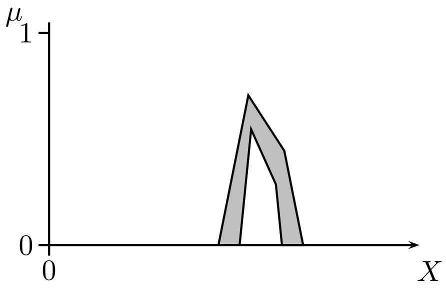

Figure 9: The FOU of the Geometric Type-2 Fuzzy Set $\widetilde{C} = \widetilde{A} \cap \widetilde{B}$.

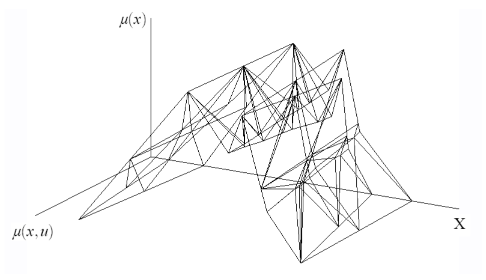

Figure 10: The Geometric Type-2 Fuzzy Set $\widetilde{C} = \widetilde{A} \cup \widetilde{B}$.

the domain of the sets. The geometric clipping operation is used to give the meet, not at every discrete point, but a every point along the continuous domain of the two geometric type-2 fuzzy sets.

Definition 2 *Let $\widetilde{A}$ and $\widetilde{B}$ be two geometric type-2 fuzzy sets, each with membership functions defined by lower and upper surface over the continuous domain X. Let the logical* and *of $\widetilde{A}$ and $\widetilde{B}$ be a third geometric type-2 fuzzy set $\widetilde{C}$.*

The lower surface of $\widetilde{C}$ = the maximum, as given by the surface clipping algorithm, of the lower surfaces of $\widetilde{A}$ and $\widetilde{B}$.

The upper surface of $\widetilde{C}$ = the minimum, as given by the surface clipping algorithm, of the upper surfaces of $\widetilde{A}$ and $\widetilde{B}$.

This performs the meet across the entire domain of $\widetilde{A}$ and $\widetilde{B}$ giving the logical and.

The logical *and* of the example geometric type-2 fuzzy sets $\widetilde{A}$ and $\widetilde{B}$ is depicted in full in Figure 8 and just the FOU in Figure 9

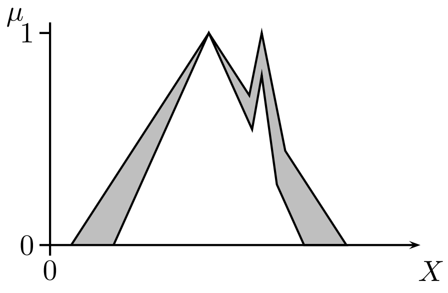

Figure 11: The FOU of the Geometric Type-2 Fuzzy Set $\widetilde{C} = \widetilde{A} \cup \widetilde{B}$.

4.3 The Geometric Or Operator

The *or* of two type-2 fuzzy sets is defined as the result of taking the join of the secondary membership functions of the two sets at each point in the domain of the sets. Again, the geometric clipping operation is used to give the join, not at every discrete point, but a every point along the continuous domain of the two geometric type-2 fuzzy sets.

Definition 3 *Let $\widetilde{A}$ and $\widetilde{B}$ be two geometric type-2 fuzzy sets, each with membership functions defined by lower and upper surface over the continuous domain X. Let the logical* and *of $\widetilde{A}$ and $\widetilde{B}$ be a third geometric type-2 fuzzy set $\widetilde{C}$.*

The lower surface of $\widetilde{C}$ = the maximum, as given by the surface clipping algorithm, of the lower surfaces of $\widetilde{A}$ and $\widetilde{B}$.

The upper surface of $\widetilde{C}$ = the maximum, as given by the surface clipping algorithm, of the upper surfaces of $\widetilde{A}$ and $\widetilde{B}$.

This performs the join across the entire domain of $\widetilde{A}$ and $\widetilde{B}$ giving the logical or.

The logical *or* of the example geometric type-2 fuzzy sets $\widetilde{A}$ and $\widetilde{B}$ is depicted in full in Figure 10 and just the FOU in Figure 11

4.4 The Geometric Implication Operator

The Mamdani implication of an antecedent on a consequent geometric type-2 fuzzy set can be found using the geometric *and* operation. A geometric type-2 fuzzy set must be constructed from a rule antecedent, which gives the surface to the antecedent across the domain of the consequent fuzzy set. Implication can be found from the logical *and* of these two sets. Due to space constraints we leave a worked example of geometric implication to the reader.

5 Conclusions

This paper has presented techniques which allow type-2 fuzzy logic inference to be performed over a continuous domain. There are both advantages and disadvantages to this approach. Accuracy is a clear advantage, geometric type-2 fuzzy sets are modelled over a continuous domain, giving more accurate computer representations of type-2 fuzzy sets. Geometric defuzzification [1] is faster than type-reduction, typically by several orders of magnitude. Only geometric type-2 fuzzy sets can be defuzzified using this technique, although it is possible to convert a discrete type-2 fuzzy set to a geometric one. However, geometric inference is somewhat slower than discrete inference suggesting that a hybrid discrete-geometric type-2 fuzzy system will have the smallest computational footprint, although not the highest level of accuracy.

Further work on the geometric approach is likely to look at hardware implementation, including potential exploitation of modern hardware including Graphics Processing Units and Field Programmable Gate Arrays.

References

[1] S. Coupland. Type-2 Fuzzy Sets: Geometric Defuzzification and Type-Reduction. In *Proc. IEEE Symposium on Foundations of Computational Intelligence*, pages 622 – 629, Honolulu, Hawaii, April 2007.

[2] S. Coupland and R. John. Fuzzy Logic and Computational Geometry. In *Proc. RASC 2004*, pages 3 – 8, Nottingham, England, December 2004.

[3] S. Coupland and R. John. Geometric Type-1 and Type-2 Fuzzy Logic Systems. *IEEE Transactions on Fuzzy Systems*, 15(1):3–15, Feburary 2007.

[4] D. Dubois and H. Prade. *Fuzzy Sets and Systems: Theory and Applications*. Academic Press, New York, 1980.

[5] P Guigue and O Devillers. Fast and Robust Triangle-Triangle Overlap Test Using Orientation Predicates. *Journal of Graphics Tools*, 8(1):25 – 32, 2003.

[6] N. N. Karnik and J. M. Mendel. Operations on Type-2 Fuzzy Sets. *Fuzzy Sets and Systems*, 122:327–348, 2001.

[7] M. Mizumoto and K. Tanaka. Fuzzy Sets of Type 2 Under Algebraic Product and Algebraic Sum. *Fuzzy Sets and Systems*, 5:277–290, 1981.

[8] T Möller. A Fast Triangle-Triangle Intersection Test. *Journal of Graphics Tools*, 2(2):25 – 30, 1997.

[9] K. Weiler and P. Atherton. Hidden surface removal using polygon area sorting. In *Proc. Siggraph 1977*, pages 214 – 222, San Jose, California, USA, 1977.

[10] L. A. Zadeh. The Concept of a Linguistic Variable and its Application to Approximate Reasoning – II. *Information Sciences*, 8:301–357, 1975.