

# Geometric Interval Type-2 Fuzzy Systems

Simon Coupland and Robert John

Centre for Computational Intelligence,

De Monfort University,

The Gateway,

Leicester,

LE1 9BH,

United Kingdom

e-mail: simonc@dmu.ac.uk, rij@dmu.ac.uk

## Abstract

In this paper we give a complete description of an interval type-2 fuzzy logic system based on geometry. We compare our novel defuzzification technique with type-reduction.

**Keywords:** Type-2 Interval Fuzzy Logic, Computational Geometry, Geometric Fuzzy Sets and Systems.

## 1 Introduction

Interval type-2 fuzzy logic has a solid theoretical basis [7, 9, 10] and is increasingly being applied to real world applications [5, 11, 8]. All of these systems are based around discrete fuzzy sets. We have previously shown that it is not necessary to discretise the sets in a type-1 fuzzy system [3]. In this work we demonstrate that interval type-2 systems can also operate over a continuous domain, giving fuzzy systems greater accuracy.

The paper is organised as follows. Section 2 discusses the relationship between computational geometry and fuzzy logic. Section 3 gives a complete description of how geometry can be used to perform interval type-2 fuzzy logic operations. A comparison is given between our novel defuzzification technique and type reduction. The final section draws conclusions from this work.

## 2 Fuzzy Logic and Computational Geometry

Fuzzy sets are typically represented in computer memory as a set of equidistant points along a domain each with a membership grade between zero and one. These discrete fuzzy sets can be easily manipulated

and operations easily coded. Fuzzy sets may also be viewed as geometric objects [2, 3, 4] that can be manipulated using geometrical operations. This section shows how fuzzy sets may be represented and manipulated as geometric objects.

### 2.1 Geometric Fuzzy Sets

A geometric fuzzy set is a fuzzy set that is represented by a ordered collection vertices called a polyline over a continuous domain. The vertices are stored in ascending order according to their  $x$  component. Each vertex is numbered from 0 to  $N$ . The geometric fuzzy set  $A$  is given in equation 1 along with a discretized version  $B$  with 10 discrete points in equation 2. These sets are depicted Figures 1 (a) and 1 (b) respectively.

$$A = \{(0.12, 0), (0.4, 0.3), (0.6, 0.3), (0.7, 1), (0.85, 0)\} \quad (1)$$

$$B = 0.70/0.186 + 0.14/0.253 + 0.21/0.319 + 0.28/0.385 + 0.30/0.452 + 0.30/0.518 + 0.30/0.585 + 0.57/0.651 + 0.90/0.717 + 0.43/0.784 \quad (2)$$

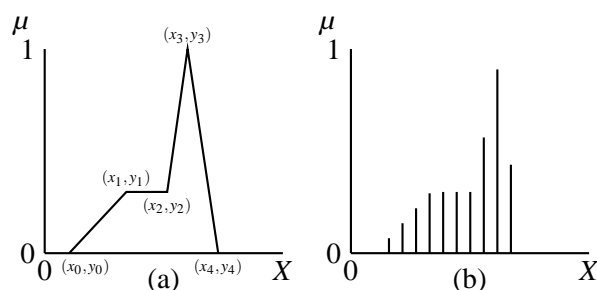


Figure 1: (a) The Geometric Fuzzy Set A. (b) The Discrete Fuzzy Set B.

Geometric fuzzy sets have been shown to provide greater modelling accuracy [3]. Having defined the geometric fuzzy set, we revisit how to apply operations from computational geometry to give the fuzzy AND and OR.

**2.2 The Geometric Fuzzy AND**

In computer graphics the ability to clip one polygon against another is often needed. For example when rendering a scene one way of ensuring that objects at back of the scene are not drawn is to clip those objects against objects in the foreground. This is known as back face culling or hidden surface removal. One of the methods used for two dimensional back face culling is the Weiler-Atherton [12] clipping algorithm. This algorithm can be used to give the fuzzy AND of two geometric fuzzy sets. The algorithm requires two vertex lists to be maintained, one for each set. All points where the sets intersect must be found beforehand, we use the Bentley-Ottmann [1] plane sweep algorithm to do this. The algorithm is given below.

1. Find all the points where the two polylines intersect.
2. Populate *A* and *B* vertex lists including intersection points.
3. Let the start point be the head of the list with highest x co-ordinate, output this point.
4. Traverse the list outputting points until an intersection point is come upon, then switch lists.
5. Repeat step 4 until off one of the lists.

Consider the two geometric fuzzy sets  $\alpha$  and  $\beta$  depicted in Figure 2 (a). Step 1 of the algorithm will return the vertex  $(x_{i0}, y_{i0})$ . The populated vertex lists from step 2 are given below.

Vertex List $\alpha$	Vertex List $\beta$
$(x_0, y_0)$	
$(x_1, y_1)$	$(x_0, y_0)$
$(x_{i0}, y_{i0})$	$(x_{i0}, y_{i0})$
$(x_2, y_2)$	$(x_1, y_1)$
	$(x_2, y_2)$

Step 3 requires the head of the list with the highest x component to be selected and output. This vertex

$(x_0, y_0)$  from  $\beta$  is output. Repeating step 3 adds the intersection point  $(x_{i0}, y_{i0})$  to the output and causes the lists to be switched. Step 3 then finds the vertex  $(x_2, y_2)$  from  $\alpha$ , which is added to the output. The end of  $\alpha$  has been reached so the algorithm terminates, returning the output vertices. The vertices form the resultant fuzzy set  $\lambda$  as depicted in Figure 2 (b).

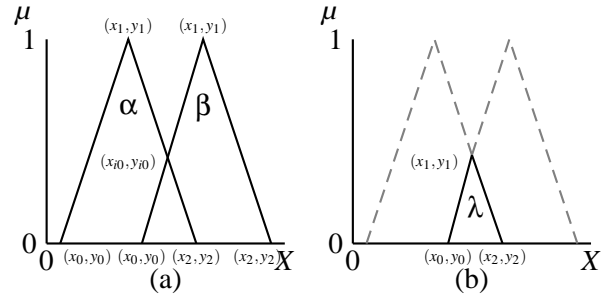


Figure 2: (a) the Fuzzy Sets  $\alpha$  and  $\beta$ . (b) The Fuzzy Set  $\alpha$  OR  $\beta$ .

**2.3 The Geometric Fuzzy OR**

The geometric fuzzy OR works in much the same way as the AND. The OR uses the modified Weiler-Atherton clipping [3] given below.

1. Find all the points where the two polylines intersect.
2. Populate *A* and *B* vertex lists including intersection points.
3. Let the start point be the head of the list with lowest x co-ordinate, output this point.
4. Traverse the list outputting points until an intersection point is come upon, then switch lists.
5. Repeat step 4 until off one of the lists.

The difference between the OR algorithm and AND algorithm is the vertex list that algorithm starts at. Starting at the vertex with lower x gives the OR and and higher x yields the AND.

This section has given a geometric representation of a fuzzy and shown how computational geometry can be used to find the AND and OR of such sets. The next section uses these concepts to give a geometric model of interval type-2 fuzzy sets and systems.

### 3 Geometric Interval Type-2 Fuzzy Systems

This section utilises work on type-1 geometric fuzzy sets to give representations and methods for interval type-2 fuzzy sets.

#### 3.1 Geometric Interval Type-2 Fuzzy Sets

A interval type-2 fuzzy set has membership grades, called secondary membership functions (SMF) that are type-1 interval sets, a fuzzy set where all points are at unity. Definitions of the interval type-2 fuzzy set  $\tilde{A}$  over discrete and continuous domains are given in equations 3 and 4 respectively. Figure 3 (a) depicts an interval type-2 fuzzy set along with a membership grade from that set in Figure 3 (b).

$$\tilde{A} = \sum_{x \in X} \sum_{u \in J_x} 1/(x, u); J_x \subseteq [0, 1] \quad (3)$$

$$\tilde{A} = \int_{x \in X} \int_{u \in J_x} 1/(x, u); J_x \subseteq [0, 1] \quad (4)$$

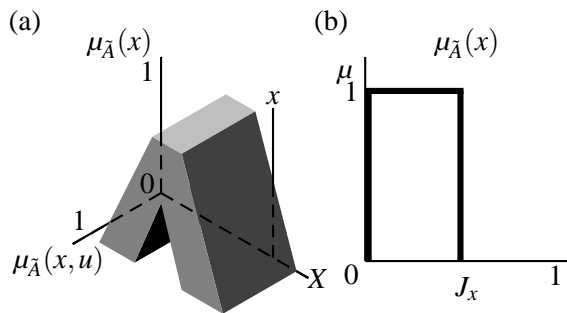


Figure 3: (a) The Type-2 Interval Fuzzy Set  $\tilde{A}$ . (b) The SMF of  $x$  in  $\tilde{A}$ .

Since all points in the third dimension of an interval type-2 fuzzy set are at unity this dimension can be ignored for modelling and computation purposes [10]. We can then define a geometric interval fuzzy set as two polylines, one representing the upper boundary of the set and one representing the lower boundary of the set. A example set  $\tilde{A}$  is given in equation 5 where  $\bar{x}$  denotes the upper bound and  $\underline{x}$  denotes the lower bound.  $\tilde{A}$  is depicted in Figure 4 (b) along with its discrete equivalent in Figure 4 (a). The shaded areas represent each sets footprint of uncertainty.

$$\tilde{A} = \overline{(0.3, 0), (1.4, 1), (3.3, 0), (0.3, 0), (1.4, 0.45), (3.3, 0)} \quad (5)$$

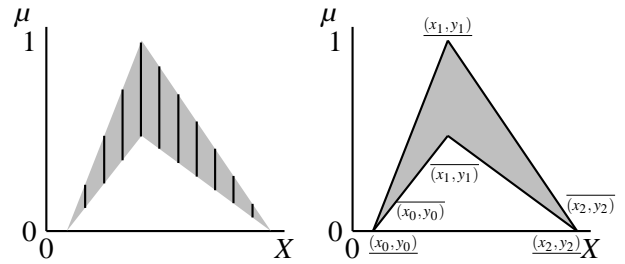


Figure 4: (a) A Discrete Type-2 Interval Fuzzy Set (b) A Geometric Type-2 Interval Fuzzy Set

The membership grade of a geometric interval type-2 fuzzy set is given by equations 6 - 8.

$$\mu_{\tilde{A}}(x) = [\overline{\mu_A(x)}, \underline{\mu_A(x)}] \quad (6)$$

$$\overline{\mu_A(x)} = \begin{cases} 0 & ; x < \bar{x}_0 \text{ OR } x > \bar{x}_n \\ \bar{y}_k & ; x = \bar{x}_k \\ \bar{y}_k + \frac{x - \bar{x}_k}{\bar{x}_{k+1} - \bar{x}_k} (\bar{y}_{k+1} - \bar{y}_k) & ; \bar{x}_k < x < \bar{x}_{k+1} \end{cases} \quad (7)$$

$$\underline{\mu_A(x)} = \begin{cases} 0 & ; x < \underline{x}_0 \text{ OR } x > \underline{x}_n \\ \underline{y}_k & ; x = \underline{x}_k \\ \underline{y}_k + \frac{x - \underline{x}_k}{\underline{x}_{k+1} - \underline{x}_k} (\underline{y}_{k+1} - \underline{y}_k) & ; \underline{x}_k < x < \underline{x}_{k+1} \end{cases} \quad (8)$$

Any fuzzy logic system has a rule base and an inferring system. A geometric interval type-2 fuzzy system has a set of rules where the fuzzy sets in the rules are geometric interval type-2 fuzzy sets. Such a rule base is given in equation 9. All rules in the rule base are combined with the OR operation.

$$\begin{aligned} R_1 &: \text{IF } a \text{ is } A_1 \text{ and } b \text{ is } B_1 \text{ or } c \text{ is } C_1 \text{ THEN } d \text{ is } D_1 \\ R_2 &: \text{IF } a \text{ is } A_2 \text{ and } b \text{ is } B_2 \text{ or } c \text{ is } C_2 \text{ THEN } d \text{ is } D_2 \end{aligned} \quad (9)$$

Where  $a, b$  and  $c$  are crisp inputs,  $d$  is a crisp output.  $A_1, B_1, C_1, D_1, A_2, B_2, C_2$  and  $D_2$  are all fuzzy sets.

#### 3.2 Computing Rule Antecedents

Since the membership grade of a geometric interval type-2 fuzzy set is a crisp interval  $[l, r]$  we can use standard definitions [7, 9] for the join (OR,  $\sqcup$ ) and meet (AND,  $\sqcap$ ). These are given in equations 10 and 11 respectively, where  $\vee$  denotes maximum and  $\star$  denotes a t-norm.

$$\sqcup_{i=1}^n F_i = [l_1 \vee l_2 \vee \dots \vee l_n, r_1 \vee r_2 \vee \dots \vee r_n] \quad (10)$$

$$\sqcap_{i=1}^n F_i = [l_1 \star l_2 \star \dots \star l_n, r_1 \star r_2 \star \dots \star r_n] \quad (11)$$

Consider the rule  $R_1$  from equation 9. Given a set of inputs  $a = 14$ ,  $b = 45$  and  $c = 7$ . the first step is to fuzzify each input in its respective set. Suppose this yields the following crisp intervals  $\mu_{\tilde{A}_1}(a) = [0.4, 0.8]$ ,  $\mu_{\tilde{B}_1}(b) = [0.5, 0.6]$  and  $\mu_{\tilde{C}_1}(c) = [0.3, 0.7]$ . First we calculate  $\mu_{\tilde{A}_1}(a) \sqcap \mu_{\tilde{B}_1}(b)$ , which is  $[0.4 \vee 0.5, 0.8 \vee 0.6] = [0.4, 0.6]$ . Then calculate  $(\mu_{\tilde{A}_1}(a) \sqcap \mu_{\tilde{B}_1}(b)) \sqcap \mu_{\tilde{C}_1}(c)$ , which using the maximum t-norm is  $[0.4 \wedge 0.5, 0.6 \wedge 0.8] = [0.5, 0.8]$ , giving the final antecedent value  $[0.5, 0.8]$ .

### 3.3 Computing Rule Consequents

Having arrived at an antecedent value  $[a_l, a_r]$ , the next stage in the fuzzy logic system is implication. In a discrete system this is done by taking the meet of every point in the consequent set with the antecedent value. To find the consequent set in the geometric system we utilise the type-1 operations defined in section 2. Our geometric interval type-2 fuzzy set consists of two geometric fuzzy sets, one giving the lower and one giving the upper bound. To perform implication we perform type-1 implication on the lower bound with the lower value from the antecedent and do the same with the upper bound and the higher value from the antecedent. Let  $x_{-1}$  be a value  $< x_0$  and let  $x_{n+1}$  be a value  $> x_n$ . Let  $i_l$  be a line segment from  $(x_{-1}, a_l)$  to  $(x_{n-1}, a_l)$  and let  $i_u$  be a line segment from  $(x_{-1}, a_u)$  to  $(x_{n-1}, a_u)$ . An example lines  $i_l$  and  $i_u$  along with a consequent set  $C_1$  are depicted in Figure 5 (a). To find the consequent set we perform the fuzzy AND from section 2.2 on  $\overline{C_1}$  and  $i_u$ , and also on  $\underline{C_1}$  and  $i_l$ . Note that these two operations are completely independent and can be performed in parallel. The resultant consequent is depicted in Figure 5 (b).

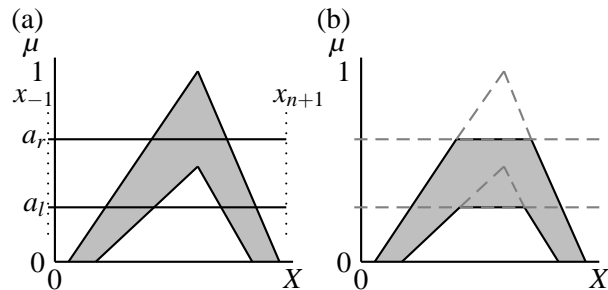


Figure 5: (a) The Computation of a Consequent Set. (b) The Resultant Consequent.

### 3.4 Combining the Rule Outputs

Having fired all the rules in the logic system we are left with a number of geometric interval fuzzy sets. We need to combine these set into one for the final step, defuzzification. In a discrete system this would be done by taking the join at every point in all the sets. For a geometric system we use the fuzzy OR operation given in section 2.3. Consider the two consequent sets  $C_1$  and  $C_2$  depicted in Figures 6 (a) and (b).

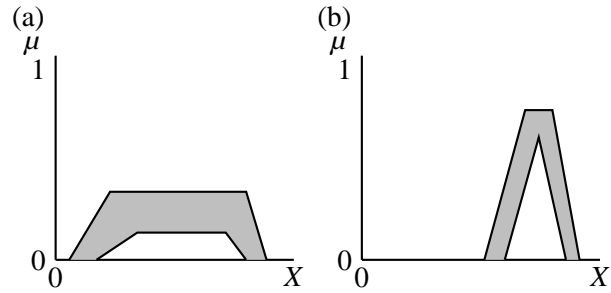


Figure 6: (a) The Consequent Set  $C_1$ . (b) The Consequent Set  $C_2$ .

To find the OR of the sets  $C_1$  and  $C_2$  we perform the fuzzy OR on  $\overline{C_1}$  and  $\overline{C_2}$  and also on  $\underline{C_1}$  and  $\underline{C_2}$ . Again these operations may be performed in parallel. The resultant geometric interval type-2 fuzzy set  $C_R$  is depicted in Figure 7.

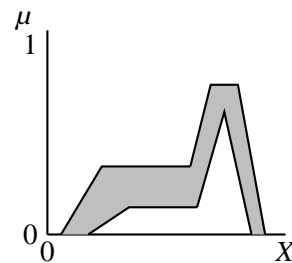


Figure 7: The Resultant Consequent Set  $C_R$ .

### 3.5 Defuzzification

Having computed a final consequent set we now must defuzzify that fuzzy set to give a usable crisp value. Discrete systems use type-reduction [6, 9] to give a crisp interval set and take the centre of this interval as the final output from the system. Here we depart from type-reduction instead favouring a geometric interpretation of the centre of the FOU of the consequent set. To find the geometric centre of the  $C_R$ 's FOU we convert the polylines of the lower and upper bound to a closed non-intersecting polygon  $P_{C_r}$ .

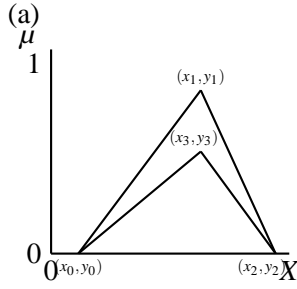


Figure 8: (a) The Polygon  $P_{C_R}$ .

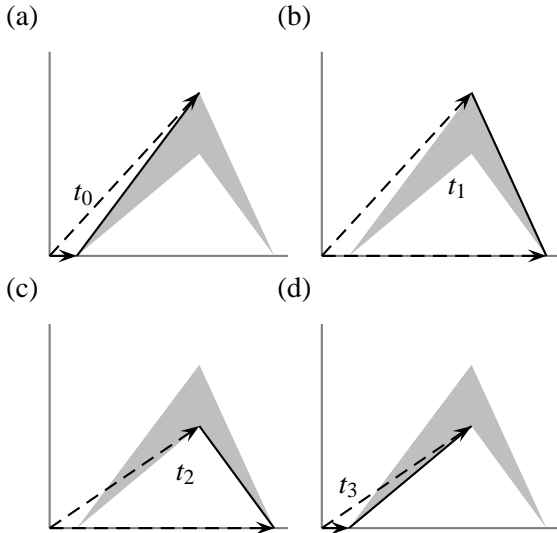


Figure 9: The Triangles  $t_0, t_1, t_2$  and  $t_3$ .

Let  $\overline{C_R} = (\overline{x_0, y_0}, \overline{x_1, y_1}, \dots, \overline{x_m, y_m})$  and let  $C_R = (x_0, y_0), (x_1, y_1), \dots, (x_n, y_n)$ . The polygon  $P_{C_r} = (x_0, y_0), (x_1, y_1), \dots, (x_m, y_m), (x_n, y_n), \dots, (x_1, y_1), (x_0, y_0)$ . Finally renumbering the indices for simplicity  $P_{C_r} = (x_0, y_0), (x_1, y_1), \dots, (x_n, y_n)$ . An example polygon  $P_{C_R}$  is depicted in Figure 8 (a). We then deconstruct  $P_{C_R}$  into a collection of  $n$  triangles<sup>1</sup>, each triangle consisting of the point  $(0,0)$  and two consecutive points from  $P_{C_R}$ . These are depicted in Figures 9 (a) to (d).

To find the centroid of  $P_{C_R}$  we take the weighted average of the centre against the area of the four triangles  $t_0$  to  $t_4$ . The centroids and areas of the triangles are given below.

$$Area\ t_0 = (x_0y_1 - x_1y_0)/2 \quad (12)$$

<sup>1</sup>Here we let  $(x_0, y_0)$  be used the  $n^{th}$  vertex that forms the closed polygon, thus our example has 4 vertices  $0 \dots 3$ ,  $(x_0, y_0)$  being the first and last vertex.

$$Area\ t_1 = (x_1y_2 - x_2y_1)/2 \quad (13)$$

$$Area\ t_2 = (x_2y_3 - x_3y_2)/2 \quad (14)$$

$$Area\ t_3 = (x_3y_0 - x_0y_3)/2 \quad (15)$$

$$Centroid\ t_0 = (x_0 + x_1)/3 \quad (16)$$

$$Centroid\ t_1 = (x_1 + x_2)/3 \quad (17)$$

$$Centroid\ t_2 = (x_2 + x_3)/3 \quad (18)$$

$$Centroid\ t_3 = (x_3 + x_0)/3 \quad (19)$$

Each triangles area is calculated as half the cross product of its edge vectors depicted with dashed and arrowheaded lines. The centroid of a triangle is the sum of the three vertex  $x$  components (one will always be zero) over three. We can generalize these giving the area and centroid of any triangle  $t_i$  in equations 20 and 21 respectively.

$$Area\ t_i = (x_iy_{i+1} - x_{i+1}y_i)/2 \quad (20)$$

$$Centroid\ t_i = (x_i + x_{i+1})/3 \quad (21)$$

The weighted average of all the triangles is given by equation 22 which reduces to equation 23.

$$C_x = \frac{\sum_{i=0}^n \left( \frac{x_i + x_{i+1}}{3} \frac{x_iy_{i+1} - x_{i+1}y_i}{2} \right)}{\sum_{i=0}^n \frac{x_iy_{i+1} - x_{i+1}y_i}{2}} \quad (22)$$

$$C_x = \frac{\sum_{i=0}^n (x_i + x_{i+1})(x_iy_{i+1} - x_{i+1}y_i)}{3 \left( \sum_{i=0}^n x_iy_{i+1} - x_{i+1}y_i \right)} \quad (23)$$

### 3.6 Defuzzification Comparison

Having shown an alternative method for defuzzifying geometric interval type-2 fuzzy set we now compare the outcome of that method with that of type-reduction. Consider the four sets  $\tilde{A}, \tilde{B}, \tilde{C}$  and  $\tilde{D}$  depicted in Figures 10 (a) to (d).

To look at the difference between type-reduction and our geometric centroid we discretised each set  $\tilde{A}$  to  $\tilde{D}$  into ten equidistant points covering each sets domain. We computed the centroid of each of these sets using centre of area type-reduction and took the geometric centroid of the original set. These centroids are presented in Table 1 where TR means type-reduced and centroid means the geometric centroid.

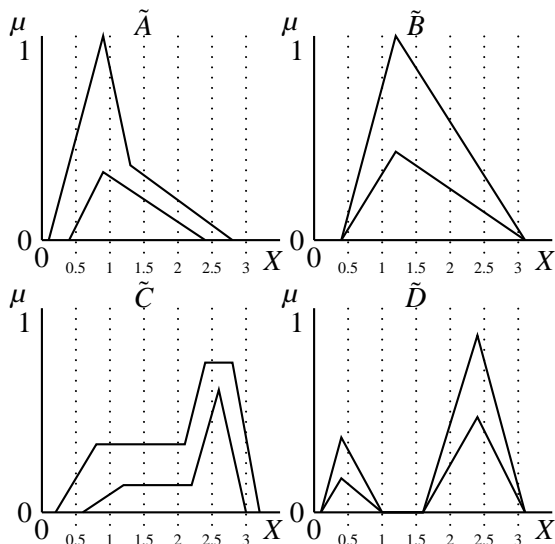


Figure 10: Four Geometric Interval Type-2 Fuzzy Sets  $\tilde{A}$ ,  $\tilde{B}$ ,  $\tilde{C}$  and  $\tilde{D}$ .

Set	TR Interval	TR Centroid	Centroid
$\tilde{A}$	[0.30,1.50]	0.90	1.02
$\tilde{B}$	[1.39,1.77]	1.58	1.57
$\tilde{C}$	[1.72,2.35]	2.03	2.10
$\tilde{D}$	[1.76,2.19]	1.98	1.94

Table 1: A Comparison of Defuzzification Techniques.

All the geometric centroid values fall well inside the type-reduced interval. With the exception of set  $\tilde{A}$  all geometric centroid values are close to the type-reduced centroid.

## 4 Conclusions

In this paper we have shown how type-2 fuzzy sets may be modelled as geometric objects. We have defined the complete set of operations needed to build a fuzzy system. We have given an alternative defuzzification method to type-reduction. The geometric centroid gave good results that were close to the type-reduced centroid in three of the examples. The other example, set  $\tilde{A}$  was the least regular of the sets and this may account for the greater difference between centroids. This work has given methods that we know from [3] improve the accuracy of fuzzy set models. Future work should consider what impact our methods have on performance.

## Acknowledgments

Simon Coupland is an EPSRC doctoral training account student.

## References

- [1] J. L. Bentley and T. A. Ottmann. Algorithms for reporting and counting geometric intersections. *IEEE Trans. Comput.*, pages 643–647, 1979.
- [2] S. Coupland and R. John. A New and Efficient Method for the Type-2 Meet Operation. In *Proc. FUZZ-IEEE 2004*, pages 959 – 964, Budapest, Hungary, July 2004.
- [3] S. Coupland and R. John. Fuzzy Logic and Computational Geometry. In *Proc. RASC 2004*, pages 3 – 8, Nottingham, England, December 2004.
- [4] S. Coupland and R. John. Towards More Efficient Type-2 Fuzzy Logic Systems. In *Proc. FUZZ-IEEE05*, pages 236 – 241, Reno, AZ, USA, May 2005.
- [5] H. Hagras. A hierarchical type-2 fuzzy logic control architecture for autonomous mobile robots. *IEEE Trans. Fuzzy Systems*, 12:524–539, 2004.
- [6] N. N. Karnik and J. M. Mendel. Centroid of a type-2 fuzzy Set. *Information Sciences*, 132:195–220, 2001.
- [7] Q. Liang and J. M. Mendel. Interval Type-2 Fuzzy Logic Systems: Theory and Design. *IEEE Transactions on Fuzzy Systems*, 8:535–549, 2000.
- [8] M. Melgarejo, A. Garcio, and C. Pena-Reyes. Pro-Two: A Hardware Based Platform For Real Time Type-2 Fuzzy Inference. In *Proc. FUZZ-IEEE 2004*, pages 977 – 982, Budapest, Hungary, July 2004.
- [9] J. M. Mendel. *Uncertain Rule-Based Fuzzy Logic Systems: Introduction and New Directions*. Prentice-Hall, Upper Saddle River, NJ, 2001.
- [10] J. M. Mendel, R. I. John, and F. Liu. Interval Type-2 Fuzzy Sets Made Simple. *IEEE Transaction on Fuzzy Systems*, 2005. In review process.
- [11] T. Ozen and J. M. Garibaldi. Effect of Type-2 Fuzzy Membership Function Shape on Modelling Variation in Human Decision Making. In *Proc. FUZZ-IEEE 2004*, pages 971 – 976, Budapest, Hungary, July 2004.
- [12] K. Weiler and P. Atherton. Hidden surface removal using polygon area sorting. In *Proc. Siggraph 77*, pages 214 – 222, San Jose, California, USA, 1977.