12th International Conference on Information Fusion
Seattle, WA, USA, July 6-9, 2009

Applying the PCR6 Rule of Combination in Real Time Classification Systems

Krispijn A. Scholte[1,2]
k.a.scholte@forcevision.nl

Willem L. van Norden[1,2]
w.l.van.norden@forcevision.nl

[1] Force Vision, Defence Material Organisation, P.O.Box 10.000, 1780 AC, Den Helder, The Netherlands
[2] MMI Group, Delft University of Technology, Mekelweg 4, 2628 CD, Delft, The Netherlands

Abstract – *Modern navies face the introduction of new ships with reduced and less trained manning and new, more complex, sensor systems. These trends could result in operator overload which might be solved by introducing a higher level of automation. In case of classification, a sub process of picture compilation, the problem of conflicting sensor data can be solved by proportional redistribution of conflict according to the* PCR6 *algorithm. This method is computationally demanding to such an extend that the applicability is limited in practice. The methodology in this paper adapts the* PCR6 *algorithm by simplification of the solution space of the algorithm by preprocessing the input data. Comparison tests show that this simplification reduces computing time by at least a factor hundred while the effects on the output are limited. We conclude that by applying preprocessing the* PCR6 *algorithm can be applied in real time classifications systems using common computer hardware.*

Keywords: Decision support, Dezert-Smarandache theory, PCR6, voting, classification.

1 Introduction

Three factors are the primary drive behind the need for automating Command and Control (C2) systems in the military maritime environment. The first one is the decreasing amount of training time for personnel due to budget cuts. This means the Royal Netherlands Navy (RNLN) is faced with a decreasing amount of knowledge in the operators on board a ship. The effects caused by this decreasing amount of knowledge are amplified by the second factor: the increase in complexity of the systems that operators need to use. The third factor is the increasing complexity of the missions undertaken by the RNLN. This is caused by the shift from "blue-water" (deep water region) operations, to "brown-water" (coastal region) operations. This results in operations that are more diverse, e.g., embargo enforcement, counter drug operations and humanitarian help. It also means that the identity of objects are less clearly defined and missions are executed closer to land which makes the sensor performance harder to predict and interpret.

These developments have led to research in automation of C2 systems, such as [1] and [2]. However, most of this work has been done on a conceptual level. In order to build a system for use in an operational environment, e.g., in the combat management system (CMS) on board navy ships, these concepts must be translated into software that runs on currently available (computer)hardware. Operators and systems on board military vessels sometimes only have minutes or even seconds to classify and identify a fast incoming object and decide what actions to take. These decisions are influenced by the amount and the quality of information they have at their disposal. It is therefore critical that a CMS functions near real time. As [3] points out, these factors will inevitably lead to challenges in practical implementation caused by limited computing power.

The research presented here focuses on the computational complexity of the naval classification system from [4]. The Proportional Conflict Redistribution (PCR6) rule used by this system to combine information has proven to be the computational bottleneck, effectively preventing it from running (near) real time. In this paper we present a solution where the combined information is (about) the same as with the PCR6 algorithm, but which is computationally less demanding.

Section 2 of this paper elaborates on the naval aspects of the classification problem. In Section 3 the algorithms as they are currently used are discussed. In section 4 a method for input filtering is presented that reduces computational complexity while having minimal impact on the combined classification solution. In Section 5 this method is verified by applying it to a simulated dataset with real-world characteristics. Thoughts on future work are offered in Section 6. Finally, the conclusions are presented in Section 7.

2 Classification

Following RNLN practice, we use the term classification for the process that determines the type of a detected object on various levels of specificity. This definition of classification also comprises the last step of recognition (determining the instance of a class), e.g., the object is the HNLMS Tromp; one of the four Dutch Air Defence and Command Frigates. We reserve the term identification for the process that determines one of the standard identities of a detected object from STANAG 1241 [5], e.g., *Friendly*, *Neutral*, or *Hostile*.

Many classification systems are based on a classification tree. This tree is either constructed on the basis of knowledge elicitation, [6, 7], or the structure is obtained by machine learning, [8]. The downside of the tree approach is the chance of getting stuck in a high level node while information might be available for lower nodes. In [9], a different model is proposed using a set notation analogy for the classification solution space. This approach enables us to have overlapping class labels and dynamically reduce the possible classification solutions. Furthermore, this set analogy allows the use of Dezert-Smarandache Theory (DSmT), [10], to combine conflicting, uncertain and imprecise classification information.

Where classification trees are based on a certain hierarchy in which one branches through the solution space, the set analogy is based on different specificity levels. In this work we use four specificity levels. On the first level the different domains are modelled, i.e., *Surface*, *Air*, and *Subsurface*.

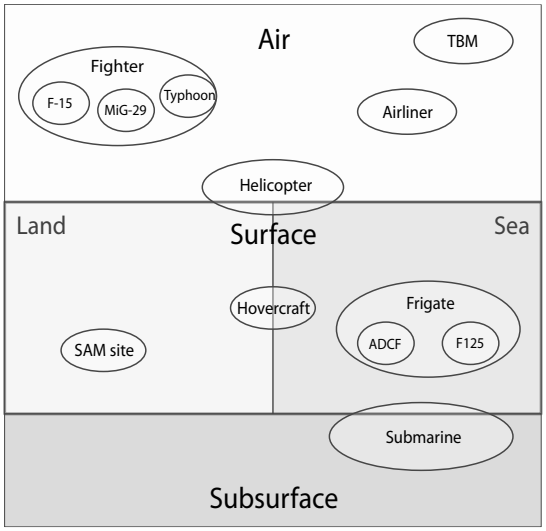


Figure 1: Set notation of the classification solution space

The second level gives a subdivision of the surface domain into *Land* and *Sea*. At one specificity level higher, the generic object types are represented such as, *Frigate* and *Helicopter*. Finally, more specific object types like *Kilo*-class submarine and *Apache* helicopter are represented at the fourth specificity level. A more generic description for models using the set analogy can be found in [11].

An example of such a classification solution space is given in Figure 1. This figure shows how different elements at the different specificity levels are related to one and another. Note that elements at the same specificity level are mutually exclusive whereas elements at different levels may not be.

3 Combining classifiers

Using different classifiers to assign belief on labels in a common frame of discernment, where each label represents a classification from the solution space mentioned above, requires a combination rule to find a single classification solution. Here, we use Dezert-Smarandache theory for the combination of classifier belief as in [11, 12]. More specifically, we use the Proportional Conflict Redistribution 6 rule (PCR6) for dealing with conflicting evidence on contraints given the fusion model that we use.

3.1 PCR6

For the combination of beliefs we start by assuming a totally free fusion model. In this model, a belief mass $m_i(X_i)$ is assigned by source S_i to a class label X_i from the hyper-power set based on the frame of discernment Θ, $X_i \in D^\Theta$ with $i \in \{1, 2, \ldots, n\}$. The construction of the hyper-power set is explained in detail in [10]. In short it is the set generated from Θ and $\varnothing$ using the $\cap$ and $\cup$ operators. The combined belief mass under the free model for class label X_i, denoted $m_c^f(X_i)$, is determined by equation (1).

$$m_c^f(X_i) = \sum_{\substack{\{X_1, X_2, \ldots, X_n\} \in D^\Theta \\ X_1 \cap X_2 \cap \ldots \cap X_n = X_i}} \prod_{\ell=1}^{n} m_\ell(X_\ell) \qquad (1)$$

In equation (1) we see that all intersections may be assigned combined belief mass due to the free model assumption. Some intersections however are impossible given a real world fusion model like classification. The set containing all these combinations is denoted $\varnothing_\mathcal{M}$. The combined mass that is assigned to these elements is proportionally redistributed to valid elements using PCR6, which is given in equation (2) and explained in detail in [13].

$$m_c^{pcr6}(X_i) = m_c^f(X_i) + \sum_{\ell=1}^{n} \Bigg(m_\ell(X_\ell)^2 \cdot \sum_{\substack{\bigcup_{u=1}^{n-1} X_{\varphi_\ell(u)} \cap X_i \in \varnothing_{\mathcal{M}} \\ X_{\varphi_\ell(1)},\ldots,X_{\varphi_\ell(n-1)} \in \left(D^\Theta\right)^{n-1}}} \frac{\prod_{w=1}^{n-1} m_{\varphi_\ell(w)}\left(X_{\varphi_\ell(w)}\right)}{m_\ell(X_i) + \sum_{w=1}^{n-1} m_{\varphi_\ell(w)}\left(X_{\varphi_\ell(w)}\right)} \Bigg) \quad (2)$$

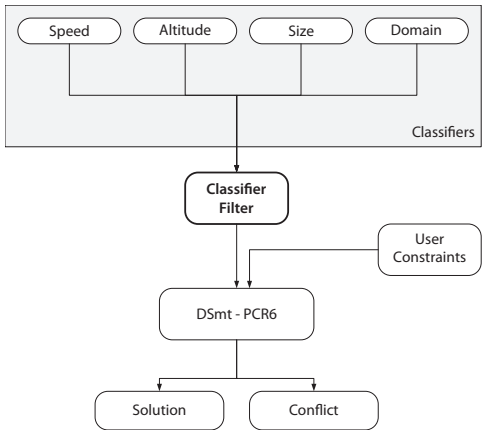


Figure 2: Classification System

3.2 Voting

Besides the computational complex PCR6 combination rule, we also look at a simple voting algorithm. This method takes the mean value of belief that the different sources assign to a label, as in equation (3). The downside of this approach is that the knowledge of the model is not used to combine evidence. Nor is this knowledge used to assign belief at a more precise class label. The obvious advantage of voting is that the required computation time is less than that required by PCR6.

$$m_c^{vot}(X_i) = \frac{1}{n} \cdot \sum_{\ell=1}^{n} m_\ell(X_i) \quad (3)$$

4 Applying PCR6 in real time

The computationally most demanding part of the PCR6 algorithm is calculating the intersecting belief assignments of the different sources. Its complexity is increases with the size of the hyper-power set and number of sources following Dedekind numbers [14], as was noted in [10]. The number of classes used in an operational real time system can be quite large, e.g., the number of specific helicopter classes that is used in practice is larger than the total of 32 classes used in this study, making the use of PCR6 difficult.

4.1 Input Simplification

The computational complexity of the algorithm could be limited by reducing the number of intersections that need to be calculated between the different sources. We aim to achieve this by placing a filtering algorithm between the classifier output and the input of the fusion algorithm. The resulting classification system, based on the system proposed in [11], is given in Figure 2.

The filtering may be of any type of algorithm that reduces the number of non-zero elements. In this case a simple threshold filtering is used. The belief $m_i(X_i)$ under a certain threshold λ (relative to the maximum value of the belief assignments of the source S_i and $0 < \lambda < 1$) is assumed to have an insignificant contribution to the output and is therefore discarded. The total discarded mass is then proportionally redistributed across the remaining beliefs. The pseudo code of such a filter is given in algorithm 1.

Algorithm 1: Simple classifier filtering

Data : sources $S : S[i], \ldots, S[n]$
threshold λ
Result : processed sources $S : S[i], \ldots, S[n]$

```
foreach S[i] do
    foreach θ in S[i] do
        if θ_j < max(S[i])*λ then
            θ_j ← 0
    if sum(S[i]) ≠ 0 then
        S[i] ← S[i] ./ sum(S[i])
```

By assigning zero to labels deemed insignificant, the number of intersections between the different sources will be reduced, which will lead to the desired reduction in computational complexity. To explore the implications of this concept we will first apply it to Zadeh's example. Secondly, we will experimentally verify the concept by means of a simulation.

4.2 Effects on Zadeh's example

It is expected that discarding believe assignments will affect the solution of the PCR6 algorithm. To get some insight into the impact of the filtering on the validity of the solution beforehand, the effects of the filtering applied to Zadeh's example [15] are studied. Table 1 shows Zadeh's example for n sources, with a frame of discernment size of $n + 1$.

The sources represented are by S_i, the classes in the frame of discernment denoted by θ_i and the value of the belief masses are a function of the variable ε_i. Given

Table 1: Zadeh's example for n-sources

	θ_1	θ_2	...	θ_n	θ_{n+1}
S_1	$1-\varepsilon$	0	...	0	ε_1
S_2	0	$1-\varepsilon$	...	0	ε_2
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
S_n	0	0	...	$1-\varepsilon$	ε_n

that all ε_i have equal values, equation (4) can be deduced for the amount of conflict (k). Conflict is an indication of how much the sources are in agreement, as explained in [10]. Here $0 \leq k \leq 1$, where 0 means all sources are in full agreement and 1 means all sources are in full disagreement.

$$k = \sum_{i=0}^{n-1} \binom{n}{i} \cdot (1-\varepsilon)^{n-i} \cdot \varepsilon^i \tag{4}$$

Because all mass in Zadeh's example is either in full conflict or full agreement, we can also write equation (4) as equation (5).

$$k = 1 - \varepsilon^n \tag{5}$$

The term ε^n represents the product of all beliefs assigned to θ_{n+1}. equation (5) can be generalised to include different values for ε_i for each source, which results in equation (6).

$$k = 1 - \prod_{i=1}^{n} \varepsilon_i \tag{6}$$

In Figure 3, conflicting mass k is plotted for different values of ε and n in accordance to equation (5). Assuming that all ε_i are equal, we denote ε_i as ε. The

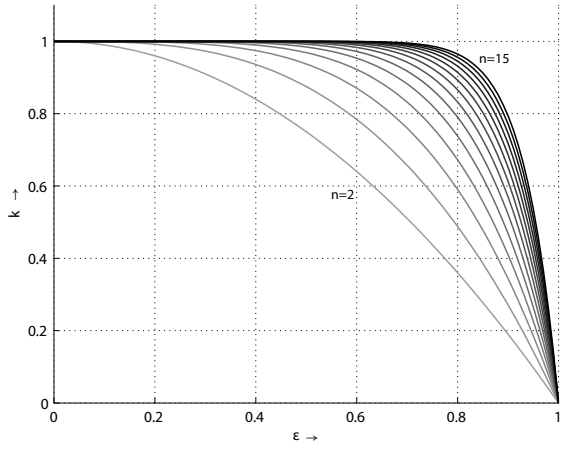


Figure 3: Conflicting mass in Zadeh's example for $n \in \{2, \ldots, 15\}$

proposed filtering entails that if threshold λ is bigger than the difference (in percent) between ε and $1-\varepsilon$, the filtered sources have the same belief assignments as would be the case if $\varepsilon = 0$. From the curves plotted in Figure 3 can be seen that this will result in an increase in k. This increase in conflicting mass follows from equation (6) and is quantified in equation (7).

$$k_{added} = \frac{\prod_{i=1}^{n} \varepsilon_i}{1 - \prod_{i=1}^{n} \varepsilon_i} \tag{7}$$

For example, when applying threshold $\lambda \geq 0.25$ to the sources and given that $n = 4$ and all $\varepsilon_i = 0.4$, k_{added} becomes 2.63% of the original conflicting mass.

From equation (7) can be seen that the impact of the input filtering decreases as n increases. The frame of discernment of the test classification system has 32 classes (although not all mutually exclusive) and though these equations only hold for Zadeh's example, it is nonetheless expected that the effects on test data will be similar.

5 Test results

The method proposed in Section 4.1 is experimentaly verified by applying it to a (pseudo random) dataset. The solutions of the sources filtered with different thresholds is compared to that of the unfiltered sources ($\lambda = 0$) and the effects of applying the filtering are evaluated using four criteria:

- confusion between class labels,
- distribution of this confusion,
- conflicting mass,
- and processing time.

These evaluation criteria will be explained in section 5.2.1 and further.

5.1 Dataset

The set of data used to verify the proposed filtering concept comprises 1080 (pseudo randomly) generated objects, 40 objects for each of the generic and specific classes in the frame of discernment. These objects have speed and altitude characteristics corresponding with those of real world objects. These generated objects were classified using the *soft* classifier output of:

- a 3-nearest neighbour classifier,
- a linear distance classifier,
- and a dissimilarity classifier.

These classifiers are described in [16]. The frame of discernment is composed of 32 classes that are not mutually exclusive but comprise four specificity levels as noted above in Section 2. Note that the dataset does not represent an actual tactical situation, but rather a set of objects that cover most of possible characteristics that could occur in the real world.

5.2 Results and discussion

All simulations were executed in a Matlab 2008b environment running on a Windows PC. All processing was done on an dual-core processor (Intel T5550), capable of 11.76 GFLOPS[1], with 2 Gb of RAM. All tests were performed without the use of parallel processing.

5.2.1 Confusion

In order to get an idea of how accurate the solution is, the classifications from the solution are compared to the labels used to generate the data set. This results in a *confusion* matrix. How the processing affects the confusion is shown in Figure 4. A comparison of the classification performance is given in Table 2. Each value represents the mean value in the confusion matrix; in other words: the percentage of objects that are correctly classified for a certain level of specificity.

Table 2: Confusion

Method	Exact	Branch	Domain	Wrong
Voting	0.266	0.110	0.180	0.444
PCR6 ($\lambda = 0$)	0.429	0.150	0.158	0.263
PCR6 ($\lambda = 0.1$)	0.434	0.151	0.161	0.254

Figure 4 shows that for $0 < \lambda \leq 0.25$ the changes in confusion are small: wrong classifications decrease by 4% while correct exact, branch and domain classifications increase by 1%. This can also be seen in Table 2. When the filtering is applied, the classification performance is slightly improved. Also note that the voting algorithm yields significantly worse results.

When $0.25 < \lambda \leq 0.35$, the classification performance increases considerably (exact +5%, branch +2.5%, domain +1.5% and wrong -10%). For these values of λ, the system has the highest number of correct exact classifications. For values of $\lambda > 0.35$, the amount of wrong classifications is still reduced but at the cost of correct exact classifications. When λ approaches 1, the amount of wrong classifications increases, which is expected as most of the beliefs are discarded.

This observation indicates that the filtering enhances the classification performance of the system with these classifiers which is caused by *noise* in the classifier output. These are (small) beliefs assigned to almost all classes by the classifiers. These beliefs lead to an increase in conflicting mass or even wrong classifications. The filtering reduces this noise though it should be noted that higher values for λ have an effect on the amount of correct exact classifications. This should be taken into account when selecting a value for λ.

5.2.2 Confusion distribution

The quality of the solution is not only represented by the amount of correct classification, as discussed in 5.2.1, but also in what way the belief assignments are distributed among the labels on the specificity level of the correct classification. In Figure 1 e.g., when an object of true classification *F-15* we determine how uniformly the belief is distributed among the labels *MiG-29* and *Typhoon*. This is expressed by the (RMS) deviation of the classifications to the mean confusion at that specificity level. A lower value is considered better, as it represents a more uniform distribution of the belief assignments. A comparison of the RMS deviation in

[1]Giga FLoating point Operations Per Second, benchmarked with SiSoft Sandra, see [17]

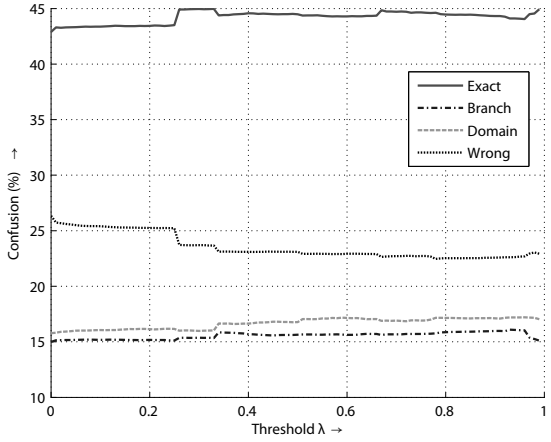


Figure 4: Confusion dependent on threshold λ

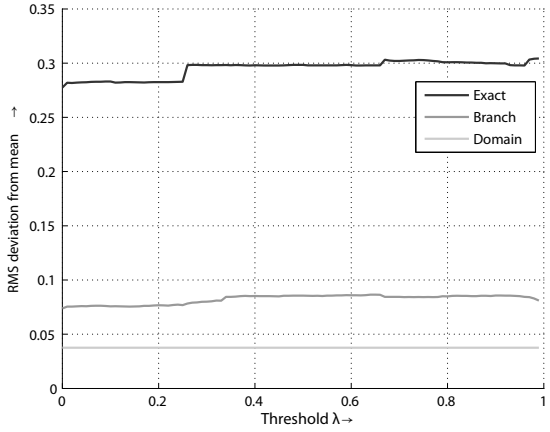


Figure 5: RMS deviation dependent on threshold λ

confusion is given in Table 3. Please note that *Wrong* classifications are omitted from this table because we are only interested in the quality of the correct classification solutions.

Table 3: RMS deviation in confusion

Method	Exact	Branch	Domain
Voting	0.169	0.039	0.038
PCR6 ($\lambda = 0$)	0.278	0.074	0.038
PCR6 ($\lambda = 0.1$)	0.283	0.076	0.038

From Table 3 can be seen that the voting algorithm yields the lowest deviation from its mean confusion. It also shows that filtering the data increases the distance marginally. This can also be seen in Figure 5 which shows the deviation dependent on λ. For $\lambda \leq 0.25$ the effects on all degrees of specificity are small. At $\lambda = 0.25$ the largest change in RMS deviation occurs (+10% with the classes in the highest degree of specificity). The classes with lower degrees of specificity change only marginally or not at all as is the case with the classes that are at *Domain* level.

5.2.3 Conflicting mass

In Figure 6 the impact of threshold-based filtering on the average conflicting mass per object of the dataset is displayed. For $\lambda \leq 0.25$ the effects of the filtering on the conflicting mass are minimal. After this threshold value, the average conflicting mass is significantly reduced.

This supports the notion that the majority of belief assignments removed by the filter are *classifier noise*, which is specific for this set of classifiers. For about a third of the objects the conflicting mass does increase, as was theorised in Section 4.2, but the total conflicting mass added by these objects is small (1% of total conflicting mass).

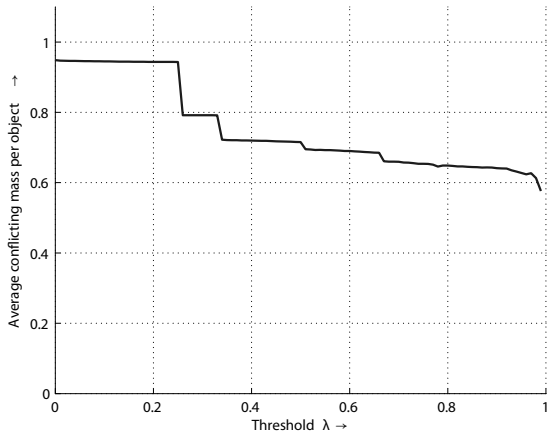
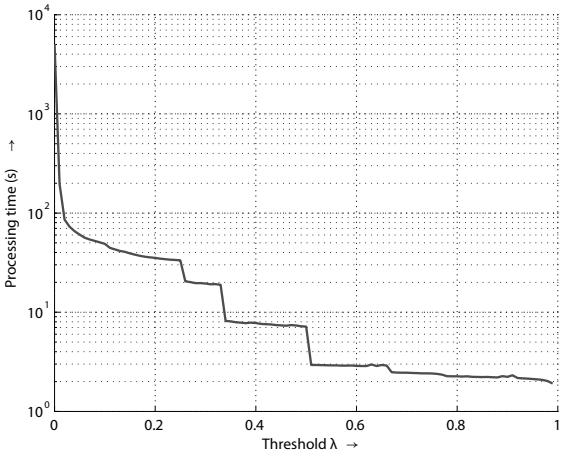
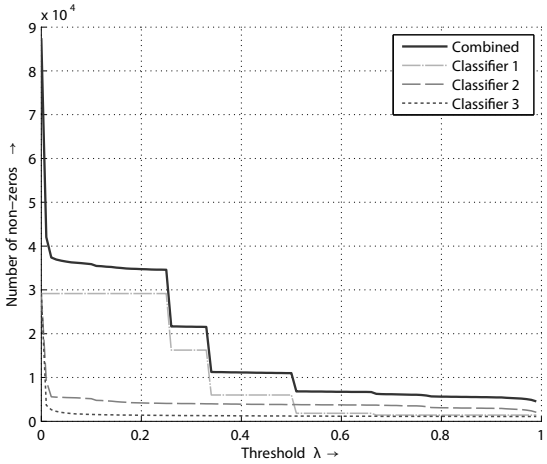


Figure 6: Average conflicting mass k per object



(a) Time required to combine the classifier information



(b) Number of non-zeros

Figure 7: Processing time and number of non-zeros dependent on threshold λ

5.2.4 Computing time

Lastly, the effect of input filtering on the required processing time is studied. The processing time as a function of λ is given in Figure 7(a) and a comparison is given in Table 4.

Table 4: Processing times for the complete dataset

Method	Processing time (s)
Voting	$5.37 \cdot 10^{-2}$
PCR6 ($\lambda = 0$)	$4.90 \cdot 10^{3}$
PCR6 ($\lambda = 0.1$)	$4.76 \cdot 10^{1}$

These measurements show that the filtering has the intended effect of reducing the required processing time by several orders of magnitude. The majority of this reduction takes place for low values of the filter threshold

$(0 < \lambda \leq 0.1)$. As theorised in Section 4.1, this gain in computing performance is caused by a reduction in non-zero elements in the classifier space, which occurs specifically in the data from classifier two and three. This can be seen from Figure 7(b) where the number of non-zero elements for each individual classifier is shown. The number of non-zero elements in the data from classifier one is reduced only for higher values of the threshold, e.g., $\lambda = 0.25$ or $\lambda = 0.5$. It is also at these values that the only other significant gains in computing performance occur. This corresponds with the notion that the number of intersecting belief assignments to be calculated is the cause for the computational complexity in the PCR6 algorithm.

5.3 Selecting a fusion method

Taking the four evaluation criteria into account, the preferred method for combining the information from the classifiers is PCR6. Voting, though fast, yields poor classification performance compared to PCR6. This becomes apparent from Table 2. Using voting, 44% of the objects were incorrectly classified in the solution. Using PCR6 gave better results: the solution is wrong for 26% of the objects. The drawback of PCR6 is its computational complexity, but as shown in Section 5.2.4, filtering the classifier output reduces this while changes in confusion, confusion distribution, and conflicting mass are insignificant when the data is filtered with a low threshold $(0 < \lambda \leq 0.1)$.

Therefore it is important to select a proper value for threshold λ. This selection depends on:

- desired reduction in processing time,
- classifier characteristics, and
- acceptable impact on the quality of the classification solution.

Choosing a low threshold reduces the processing time considerably while its effects on the output are insignificant. In most cases this would be a safe choice. In some cases it beneficial to select a higher threshold because classification performance increases due to the filtering. Figure 4 illustrates this: for $0.25 \leq \lambda \leq 0.35$ the number of wrong classifications is decreased by 10% and the number of exact classifications is increased by 5%. Because a higher threshold can potentially have a negative effect on the classification performance of the system, such a choice should always be made after analysing the output of the classifiers used.

6 Future work

This research has focused on a single domain: the classification of objects in a military maritime domain. However, the classifiers used to classify the objects are of a common type, as described [16], and are found in other domains as well. It is therefore expected that the solution proposed in this paper will also work in other domains with a large frame of discernment and overlapping classes (towards a free fusion model) where DSmT might be applied, e.g., financial market analysis, crisis response, and medical sciences. This should be experimentally verified.

Another expected benefit of *classifier noise* suppression is a temporal more stable combined classification solution. Future work should prove this hypothesis.

Choosing one threshold to apply to the data of all classifiers does not appear to be an optimal solution. Figure 7(b) shows how the number of non-zero elements in the data from a classifier is reduced. The non-zero elements in the data from classifiers two and three are reduced by the filter in the same manner. The data from classifier one, however, reacts differently to filtering. It is therefore expected that better classification performance can be attained by filtering the data from each classifier with a threshold appropriate for that classifier.

The expected size of the frame of discernment in an operational system will be much larger than the one used in this paper. Furthermore, it is expected that for the military maritime classification domain the number of objects will increase too. This means that although a substantial reduction in processing time is already achieved, more performance oriented work on this subject has to be done. This paper does not take parallel processing into account. As our processing measurements comprise the whole dataset and no dependencies between objects processed exist, it is expected that the computing performance for the entire dataset scales linear with the number of cpu cores available.

When parallel processing of the dataset is not enough, the PCR6 algorithm could be applied iteratively to each of the objects. Each iteration, the frame of discernment is expanded to a deeper level of specificity, e.g., in this case first determining an objects domain, then generic class and finally specific class. This would add an *anytime* feature to the system as the classification at some specificity level of any object will be known and the classification solution becomes more exact in time.

7 Conclusions

Using PCR6 for information fusion on currently available hardware makes the real time application difficult due to the computational complexity of the algorithm. Instead a voting algorithm could be used to combine the information. This method is very fast but it yields a considerably less accurate solution: it is desirable to use PCR6.

Despite the computational complexity of the algorithm, we have shown that it is possible to use PCR6 in a real time environment. The proposed input filter substantially reduces the required computing time

while the quality of the solution does not suffer or is even improved in some cases. How the filtering is applied is very much dependent on the characteristics of the classifiers used, which should always be analysed when building this kind of system.

Our method is suited for domains with a large frame of discernment containing overlapping classes. It would also allow domains that have a time-critical aspect, which currently restricts the use of PCR6 or DSmT in general, to use PCR6 for information fusion.

References

[1] Fok Bolderheij. *Mission Driven Sensor Management: Analysis, design, implementation and simulation*. PhD thesis, Delft University of Technology, November 2007.

[2] P.R. Smart, A. Russell, N.R. Shadbolt, M.C. Schraefel, and L.A. Carr. Aktivesa: A technical demonstrator system for enhanced situation awareness. *The Computer Journal*, 50(6):703–716, 2007.

[3] Eric J. Horvits. Reasoning about beliefs and actions under computational resource constraints. In *Proceedings of the Third Workshop on Uncertainty in Artificial Intelligence*, pages 429–444. AAAI and Association for Uncertainty in Artificial Intelligence, Mountain View, CA., July 1987.

[4] Wilbert L. van Norden and Catholijn Jonker. *Advances and Applications of DSmT for Information Fusion, Collected Works, volume 3*, chapter Utilizing classifier conflict for sensor management and user interaction. American Research Press, Rehoboth (MA), to appear 2009.

[5] NATO. Nato standard identity description structure for tactical use (maritime), October 16 1996 (promulgation).

[6] Bionda Mertens. Reasoning with uncertainty in the situational awareness of air targets. Master's thesis, Delft University of Technology, Delft, 2004.

[7] Krispijn A. Scholte. Dynamic bayesian networks for reasoning about noisy target data. Master's thesis, Royal Netherlands Naval College, Den Helder, 2005.

[8] J. Ross Quinlan. *C4.5: Programs for machine learning*. Morgan Kaufmann Publishers Inc., San Francisco (CA) USA, 1993.

[9] Willem L. van Norden, Fok Bolderheij, and Catholijn M. Jonker. Classification support using confidence intervals. In *Proc. of the 11th International Conference on Information Fusion*, pages 295–301, June 30 – July 3 2008.

[10] Florentin Smarandache and Jean Dezert. An introduction to the DSm theory for the combination of paradoxical, uncertain and imprecise sources of information. In *Proceedings of the 13th International Congress of Cybernetics and Systems*, July 6–10 2005.

[11] Willem L. van Norden, Fok Bolderheij, and Catholijn M. Jonker. Combining system and user belief on classification using the DSmT. In *Proc. of the 11th International Conference on Information Fusion*, pages 768–775, June 30 – July 3 2008.

[12] Arnaud Martin and Christophe Oswald. *Advances and Applications of DSmT for Information Fusion (collected works), volume 2*, chapter 11. Generalized proportional conflict redistribution rule applied to Somar imagery and Radar target classification. American Research Press, Rehoboth, USA, 2006. ISBN 1-59973-000-6.

[13] Arnaud Martin and Christophe Oswald. *Advances and Applications of DSmT for Information Fusion (collected works), volume 2*, chapter 2. A new generalization of the proportional conflict redistribution rule stable in terms of decision, pages 69–88. American Research Press, Rehoboth (MA), 2006.

[14] M. Tombak, A. Isotamm, and T. Tamme. On logical method for counting dedekind numbers. *Lecture notes on Computer Science*, 2138:424–427, 2001.

[15] L.A. Zadeh. On the validity of Dempster's rule of combination of evidence. *Memo M*, 79:24, 1979.

[16] F. van der Heijden, R.P.W. Duin, D. de Ridder, and D.M.J. Tax. *Classification, parameter estimation and state estimation - an engineering approach using Matlab*. John Wiley & Sons, Chichester, England, 2004. ISBN 0-47009-013-8.

[17] Tom's Hardware Guide Desktop CPU Chart Q1 '09 SiSoftware Sandra XI Arithmetic MFLOPS. http://www.tomshardware.com/charts/cpu-charts-2008-q1-2008/sisoftware-sandra-xi,392.html. Internet, 20th Feb 2009.