Star Identification using Neural Networks

Thomas Lindblad, Clark S. Lindsey
Department of Physics, Royal Institute of Technology
S-104 05 Stockholm, Sweden

Åge Eide, Öystein Solberg and Andrew Bolseth
Østfold College
N-1757 Halden, Norway

Abstract: Star trackers provide spacecraft with the most precise estimate of their orientation,or attitude, with respect to a fixed celestial coordinate system. The star tracker camera views a patch of the celestial sphere and attempts to recognize the stars contained within. Then from the known star positions it will calculate the attitude. A number of pattern recognition methods, each with various strengths and weaknesses, have been implemented in star trackers. The most challenging situation involves onboard autonomous identification. The limits on memory, power, weight, etc. place severe constraints on the processing available. We discuss here some neural algorithms and the kind of devices in which it might be implemememented.

1. Introduction

Star trackers are devices for satellites and space probes designed to determine with high precision the *attitude* of the vehicle. The star tracker identifies the stars in a camera image and by obtaining their position from a database determines the pointing direction of the camera and thereby the attitude of the spacecraft. In many spacecraft a coarse attitude estimate is first provided by a sun sensor or earth horizon detector [1]. Ideally, however, for the new generation of light-weight, low cost space craft a lone star tracker would provide the initial attitude determination with little or no *a-priori* knowledge. The star tracker is composed of optics, a charge-coupled device (CCD) detector, read-out electronics and some digital processors with pertinent auxiliary devices.

The attitude information may just be used for reference (coordinate mark) in connection with other instruments or serve as a part of an attitude stabilising system. Controlling the attitude implies that there is a "desired situation", and some force or forces are present in the system to change the attitude. Both reactions to the "disturbing forces" and a reference for the desired situation must be supplied to the control system. The objective of the star tracker is to provide the reference.

A typical star tracker [2] may have a field of view (FoV) of 20 x 20 degrees, and have a 512 pixel/ row 512 rows MMP CCD. The sensitivity is typically in a range of magnitude M = +0.1 to + 4.5. These characteristics yields a > 50% probability of up to ten stars in the FoV. The star catalogue in the onboard computer typically involves a few thousands stars whose positions are very accurately known and make it possible to determine the attitude to within few arcseconds.

Generally, spinning satellites have not used star trackers because of the streaking of the star images in the exposure time of the cameras. Instead, a star sensor measures crossing times of a star over a slit above a light detector. The pattern of light pulses provides the signature for a given star constellation. However, if processing and exposure speeds were increased sufficiently, a star tracker could be feasible for a spinning satellite, although one may have to give up accuracy in the case of large angular velocities. Sensitivity, precision and the number of stars seen during a single revolution are essential inputs to optimize the system.

The star tracker processor is generally a conventional von Neumann device. However, the fact that the recorded star intensity varies, noise pickup, etc. calls for a certain desired slack or tolerance in operation. This implies that an artificial neural network alogorithm would be beneficial. Although neural networks are highly parallel in their architectures they must be executed sequentially, and thus slowly, on a von Neumann processor. Today, however, there are several implementations directly in hardware. These devices yield processing times from a few hundred nanoseconds to a few microseconds and may thus be used as coprocessors to the conventional CPU. Alternatively, the star tracker may be implemented in a SIMD architecture. If star tracker NNW processing was thus implemented in hardware one would probably benefit both in performance, e.g. perhaps fast enough to do star tracking on a spinning satellite, and in weight.

2. The star tracker

The star tracker operates in two modes: (1) the inital acquisition mode where it must identify the current star pattern without any prior information, such as when the spacecraft first reaches orbit or after a temporary loss of control; (2) the tracking mode where it merely updates the current attitude based on the known position from the previous measurement. Here we examine only the case of the initial acquisition.

Star identification algorithms must use the given invariants of a star constellation: angular distances between the star and its nearest stars and the opening angles between the vectors from the star to the nearest stars. The brightness is not a true invariant due to both the natural variation of some stars and the noise and sensitivity variations of the CCD. Nevertheless, some algorithms take advantage of the general consistency of star brightness.

A common technique is the *triplets* method that combines the unknown star with its two nearest neighbors [3,4]. For each triplet the two angular distances plus the opening angle are compared to a triplets database. The closest match is taken as the star ID. This is repeated for each star in the FOV. Finally, the positions of the stars are compared for consistency (e.g. if nine stars are identified as belonging to a section in the northern celestial hemisphere and one is identified in the south, throw out the latter) and the pointing angle of the camera is calculated.

There are several difficulties with this method. The distances and opening angles are usually quantized in a coarse integer format, say 8 or 10 bits. Because of noise fluctuations, the measured values for a given star triplet will vary somewhat. So the database must include cases where each feature value varies by one or two increments. Furthermore, one or both of the nearest stars could be near the sensitivity threshold and disappear from view. Or a star just below the threshold may come into view and be closer than the nominal closest stars. Taking into account all these possibilites results in 100-200 entries for *each* star in the triplets feature database. So a given measured triplet usually matches several possible stars. Only in the finally consistency check with all the other stars in the FOV can a valid identification be determined.

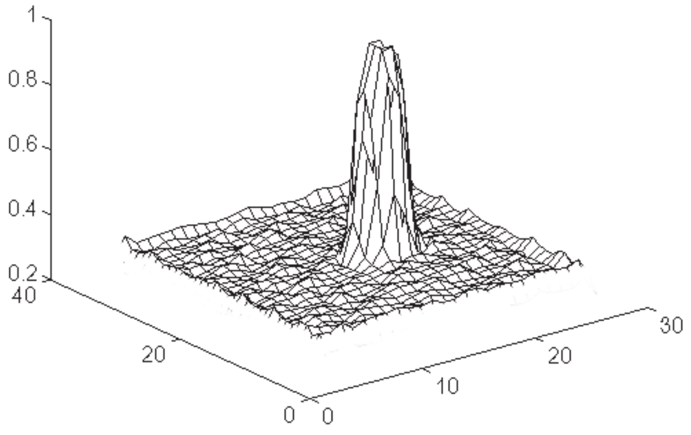

Fig. 1. Example of the intensity of a star observed with a CCD-camera. The intensity is distributed over several pixels.

Another method places the star pattern into a matrix or grid with the unknown star at the center [3]. The stars are rotated until the nearest star lies along the horizontal row and on the right hand side. The grid pattern is then compared to a database of similar patterns. This method is less sensitive to position fluctiations but does require multiple database entries for all the cases of different nearest stars.

The image on the CCD is defocused so that the starlight is spread over several pixels (figure 1). Averaging the position then provides subpixel resolution. For a CCD of a given number of elements, to obtain greater accuracy, the FOV must be made smaller and the magnification increased to spread the light of a given star over more pixels. To insure that there are enough stars in the smaller FOV in every possible direction, the brightness thresholds must be made lower, thus increasing the total mumber of stars in the database. The databases grow quickly and nonlinearly. For deep space probes that desire high precision, autonomous star tracking, the database with the above methods can become prohibitively large due to brightness fluctuations.

3. The implementations

Several Neural Network (NNW) based algorithms have been proposed for star tracking. For example, Alveda *et al*. [5] used a distribution of star brightness vs distance from the desired star as input to a back-propagation network. Bardwell [6] used distances and cosine squared of the angular separations as inputs to a Kohonen feature map. Domeika *et al*. [7] used brightness and angular separation as inputs to a Hopfield network. However, these type of inputs patterns have similar problems as the methods mentioned in section 2. The brightness of the stars can vary naturally for some stars and noise can move stars near the CCD threshold in and out of detection. This means the network must be capable of recognising the star for all possible cases of missing and extra stars. The angular separations then to the nearest stars must be included for cases where one or both of those stars are not detected. The network sizes must then grow accordingly.

These neural network approaches, however, were not considered in the context of hardware implementation. Today there are a several commercial neural network chips available. These include the Zero Instruction Set Computer (ZISC036) from IBM, which has a Radial Basis Function (RBF) neural network as well as a *K* nearest neighbour (non neural net) algorithm implemented, and the CNAPS chip from Adaptive Solutions. The latter is a general SIMD architecture processor, particularly suitable for neural network implementations.

3. 1 The RBF Neural Network Appraoch

One study involved a simplified task of identifying only the brightest star in the cell of a grid covering the celestial sphere. In the investigation described in the diploma work of ref. [13], the RCE or ROI method of training RBF type networks was used. This network is implemented in the IBM ZISC chip discussed in section 5 and the ZISC was used here to execute the network. The training data was generated by: (i) finding the brightest star in a cell, (ii) calculating the distance between this star and the three closest stars, (iii) calculating the opening angles between vectors to these stars from the brightest star and (iv) transforming the data to the ZISC input format. The training vector consisted of two pairs of distances and the intensity of the two brightest of the nearest stars, followed by the three angles and the identification of the star.

The results using the IBM ZISC036 hardware in RBF mode were not very rewarding when tested with respect to noise on distance and intensity. Figure 2 includes four cases of noisy data. It may be concluded that the ZISC036 is not performing very well for any case. It is particularly sensitive to noise on the intensities. Tests showed that while noise on the distance (5% as in group 1) yielded mediocre results, even worse results were obtained when noise was added to the intensity values. Since the RBF as well as

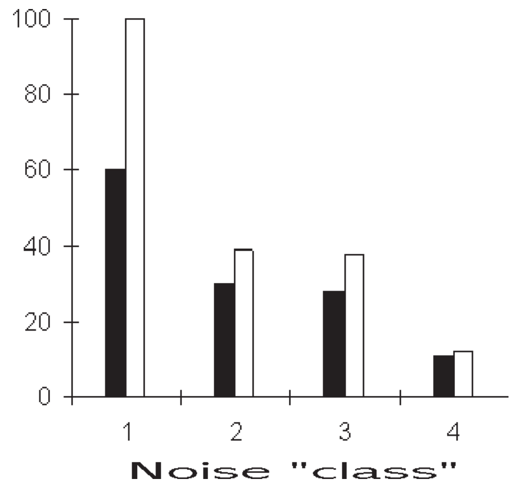

Figure 2. Success rate versus noisy distance/intensity data as obtained from the ZISC036 hardware. The results for four noise classes with noise on distance and intensity: class 1 = 5% and 0%, 2 = 0% and 5%, 3 = 5% and 5% and 4 = 10% and 10%, respectively, The black and white bars corresponds to two different ZISC configurations, the black bars are for no reduction of the NAIF and the white bars are for a 15% reduction.

the ZISC has been shown to be very sensitive to noisy data in connection with character recognition [10,11], this is perhaps what was to be expected. In the case of character recognition, it was shown that the Dynamic Decay Adjustment (DDA) algorithm (see section 5) was much more noise resistant.

Several other exploratory tests with the ZISC036 were performed. Generally speaking, fair results could only be obtained when using a large number of prototypes. The results presented in fig. 3 are obtained for the case when the L1 norm (the Manhattan block distance as discussed in section 5) is used when calculating the distances between the vector and prototypes. Also, a reduction of the so-called NAIF of 15% was used from the trained values. NAIF stands for Neuron Actual Influence Field and is the distance threshold for fire/no fire decision. In the ZISC chip, this is a 16-bit register. It will first be set equal to the MAF, the Maximum Influence Field (there is also a Minumum Influence Field, MIF, and clearly MAF > MIF). As more templates are presented during the training, MAF will be reduced. The ZISC chip holds the accumulated distances between the testvector and the prototype in another register called DIST. During execution, the actual prototype will fire if DIST < NAIF. What we have done here is to artificially reduce the “firing distance” with 15%. The results obtained for seven different noise classes shown in fig. 3 is not very good. Close to 100% can be obtained for up to 10% noise on the distance (class 1 and 2 in fig. 3), but for 5% noise in the intensity (class 3) the value drops well below 50%. The value is still the same when 5% noise is present on both distance and intensity. When the noise is 10% on the intensity, we only get proper classifications in one case out of ten (class 6 and 7).

Other NNW star ID techniques were also investigated. We tried, for example, input patterns of intensity ordered according to distance for feed-forward networks trained with back-propagation or cascade correlation, and also a radial basis function network [8]. This avoided the problems of including angular information, e.g. determining which stars to use for the opening angles. However, the classifications were nevertheless very sensitive to noise on the magnitude and somewhat sensitive to distance fluctuations.

3.2 The KNN Approach

Since the IBM ZISC036 also can be operated in the *K* nearest neighbour mode, it was reasonable to investigate this approach. The ZISC036 chip is very easy to implement in hardware both as a stand alone device (it is also highly cascadeable if 36 processing elements are not enough) and as a coprocessor to Intel and Motorola CPUs.

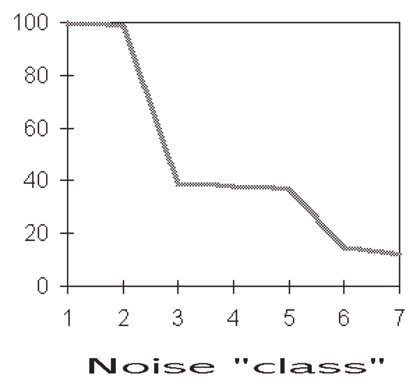

Figure 3. Noise tolerance of the ZISC036 for seven different noise classes discussed in the text for the case of a 15% reduction of the NAIF.

In a first test of the KNN approach we used the same data as discribed above. Although, the system because much more resistant to noise, there was still a problem with the low degree of correctly classified stars. When 5% (10) noise was added to the intensity, the success rate dropped from 99.5% to 62% (35%). The effect of noise on the distance values were even less than in the RBF case. Thus there seem to be a general problem with noise on intensity, while noise on distance is OK.

We decided to avoid the use of the brightness and angles. Instead we use a histogram of distances to all stars around the given star out to fixed angular radius from the star to be identified (*c.f.* fig. 4). If the histogram of *M* bins is sufficiently populated, the loss or gain of a star or two will not affect the pattern significantly. So the lower the brightness threshold, the more stars are included and the more robust the input patterns to fluctuations. All stars in the FOV do not have to be identified. This means that the database or NNW size can grow slowly since you only need one pattern per star.

Then patterns for the *N* bright stars becomes a network of *N* prototype vectors of dimension M. For each star we used only non-noisy prototypes. The number of variations of noise, extra stars, etc, was too great to make a practical training set. An input pattern is presented to the network and each prototype calculates its distance

$$d = |(\mathbf{P}_i - \mathbf{P}_0)| \qquad (1)$$

from this pattern and the resulting basis function value. The input is then said to belong to the class with the largest functional value. Alternatively, the input pattern could be said to belong to the nearest prototype,

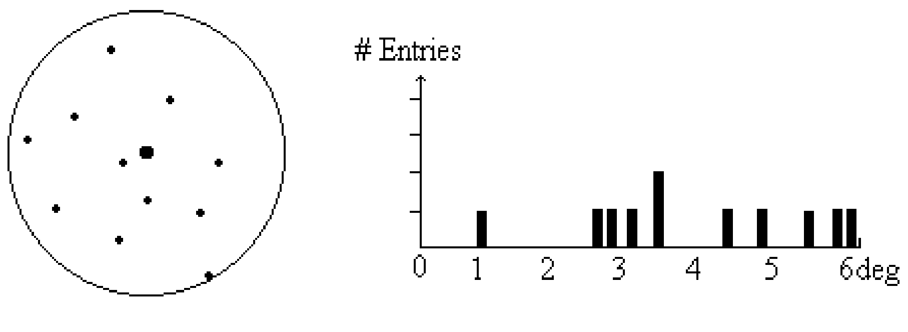

Figure 4. The star pattern on the left shows stars within 6° of the centre star that is to be identified. The diagram on the right shows a histogram of the distances (in degrees) of the stars from the centre star.

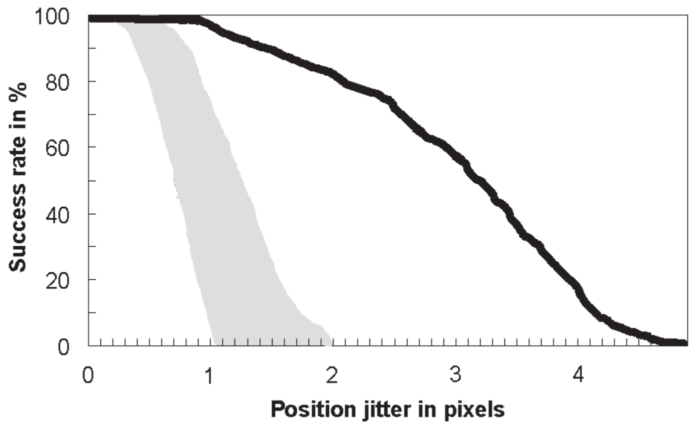

Figure 5. The ID efficiency *versus* noise on the star positions. The solid (black) curve represents the present approach, while the shaded area represents results obtained with various "triangle" methods

i.e. the *nearest neighbour* method. This is not a true neural network method but it works naturally with neural network hardware, especially the radial basis type as discussed in Section 5. (A probabilistic neural network, where every input pattern becomes a prototype, could be used perhaps to give a probability output.)

4. Results

It has been shown above that the implemetation of a star tracker using a RBF type of neural network architecture will require a system with many neuron. Specifically, one ZISC036 chip with only 36 neurons will not be enough, but rather 20-50 times that number is requires. It is also concluded that the ZISC036 on-chip learning paradigm does not yield a very noise resistant system. The ZISC036 is, however, cascadeable and yields results in 4.5µsec. A larger ZISC with several hundred neurons and a noise resistant paradigm like the DDA would be an ideal situation. Alternatively, the DDA will have to be implemented in a SIMD architecture like the CNAPS as will be discussed below and in appendix.

However, the ZISC can also be operated in a a non-neural mode, the KNN approach. As presented in reference [9], we found this method to provide high tolerance against flutuations in position and brightness. Figure 5 and 6 show that the histogram method provides good identification for large position and brightness jitter and falls gracefully at very large values. The network size grows more slowly than the triplets method (no comparison was made to the grid method but it can be expected to grow faster

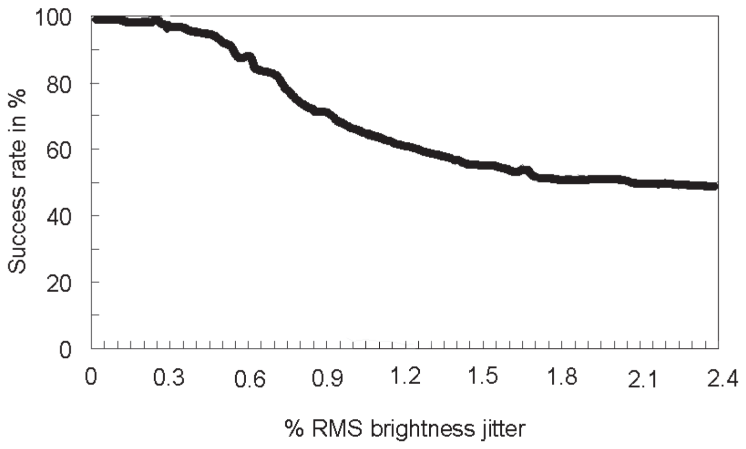

Figure 6. The ID efficiency *versus* noise on the star intensities. Here the triangle and match group algorithms yielded worse values or roughly 80% at 0.5 and 55% at 1 pixel

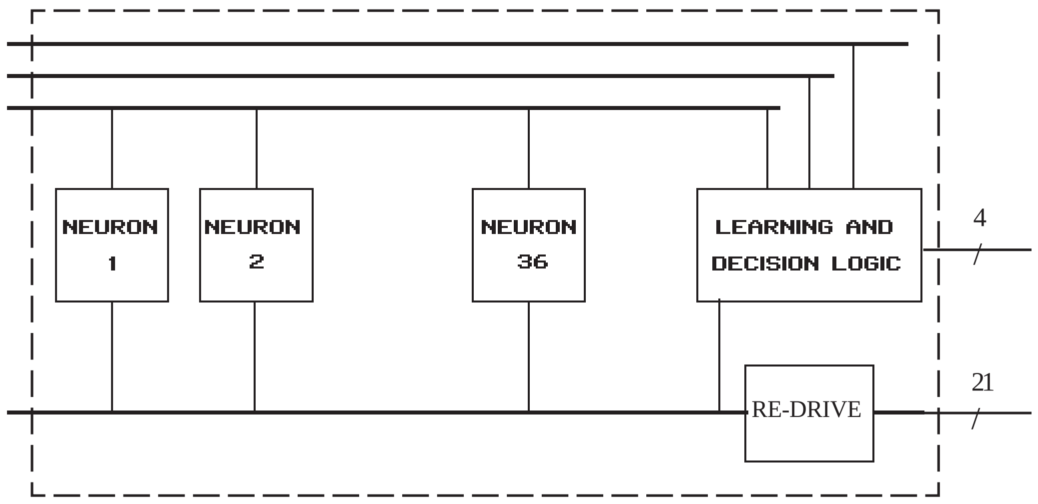

Figure 7. The IBM ZISC036 block diagram. The top part shows the address (6-bit), control (9-bit) and I/O data (16-bit) buses; to the right is shown the decision bus (4-bit) and the inter-ZISC communication bus (21-bit). This circuit can operate in RBF mode as well as KNN mode as discussed in the text.

as well). The speed of computation was slower due to the sequential processing in the simulations. The next section discusses possible solutions for this problem.

5. Hardware Designs.

There are several ways to implement the NNW star identification techniques in hardware. A conventional von Neumann computer could be used stand-alone or with a neural network hardware coprocessor. As mentioned above the IBM Zero Instruction Set Computer, ZISC036 (cf. fig. 7 for an general overview of the architecture) is the first one in a series of building blocks[10]. It has been implemented in a fairly standard technology, 1 μ CMOS and is available in a 144 pin surface mount package. It is a highly cascadable chip, which means that systems with more than 36 neurons are easily obtained by adding more chips. Also, the software is independent of whether or not you are using one or several chips. Each neuron has a register file for prototype storage, as well as a unit for evaluation of distances. The upper limit of the number of *categories* is equal to 16 K. The ZISC has a Radial Basis Function (RBF) architecture and an on-chip learning algorithm of the Region-of-Interest type. In the ZISC036, the hidden or prototype neurons use simple stepfunctions (other hardware implementations like the Ni 1000 uses Gaussians). When a new vector is presented to the network, each RBF neuron calculates the distance *d* between its prototype vector and the input vector. If the neuron fires, it will activate the output or category neuron to which it is connected. In the ZISC architecture, each hidden neuron is only connected to one output neuron (category). In other RBF topologies all neurons in these layers may be interconnected.

The *distances* (*d*) can be calculated according to two norms, L_1 or L_{sup}

$$L_1 = \Sigma \ |V_i - P_{\mathrm{i}}|, \tag{2}$$

where V_{i} and P_{i} represents the input vector and the stored prototype elements respectively. The summation runs from 1 to 64 and each component of the vector is coded as an 8-bit number. This is referred to as the Manhattan block distance and the distance calculations are carried out using a 14-bit accumulator. Alternatively, we can have

$$L_{\mathrm{sup}} = \max |V_i - P_{\mathrm{i}}|, \tag{3}$$

with $i = 1, n,$ for an n element vector.

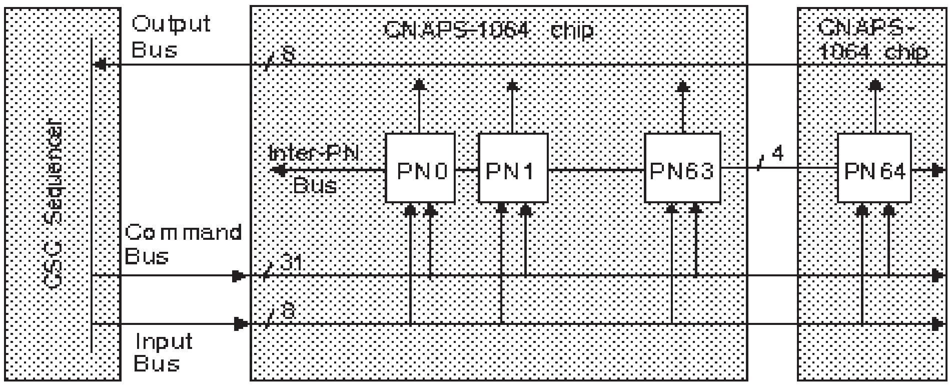

Figure 8. The architecture of the CNAPS SIMD chip from Adaptive Solutions, Inc.

The ZISC036 can also be used in *K* nearest neighbor (KNN) mode to carry out the star tracker algorithm discussed above. It allows the distance between an input vector and the stored prototypes to be calculated, associated with the corresponding category, and delivered by ascending order of distance. Distance evaluation is performed the same way as in the RBF-case, i.e. either corresponding to the polyhedral volume influence field, L_1, or the hyper-cube influence field, L_{sup}.

If the ZISC036 is operated at 20 MHz, 64 components can be fed and processed in 3.2 µs. The evaluation is obtained in 0.5 µs (or ten clock cycles) after the feeding of the last component, which corresponds to a quarter of a million evaluations per second on a 2000 MIPS von Neumann processor. The long execution time of the algorithm when run on a von Neumann computer reduces to a more reasonable one, when using dedicated hardware.

The implementation of a neural star tracker in ZISC hardware has been discussed by Solberg and Bolseth [13]. They used an ISA board from IBM with 16 ZISC036 chips implemented. They do get significantly better results for the ZISC-KNN than for the ZISC-RBF, in particular for very noisy data. It has been shown that the Dynamic Decay Adjustment alogorithm [14] is much more noise resistant [15] than the paradigm which resides on-chip the ZISC. Hence, a ZISC chip with this paradigm implemented would probably be a very nice alternative. Future versions of the ZISC will also probably contain larger numbers of neurons so that fewer chips would be needed to contain the star prototype patterns for at least 2K stars.

The other approach is based on a SIMD architecture (fig. 8). The CNAPS chip has been used to implement several neural network architectures. These architectures include the RBF architecture using the Dynamic Decay Adjustment (DDA) alogorithm [14,15] for learning, in order to get a better noise tolerant behavious. This algorithm is indeed simple and very efficient. Like several other paradigms for RBF networks, the DDA uses Gaussian response functions. This algorithm resists the growth in neurons, i.e. prototypes, needed and thus would reduce the number of hardware neurons required.

We have tested this and other implementaions on a CNAPS PCI-board with a total of 128 processors, and found that one generally gains a factor of 20 to 100 in speed as compared to a 90 MHz Pentium PC. This would again bring down the execution time for the present alogoritm to a reasonable value.

6. Summary

Unlike many problems attacked with neural network methods, the star identification method requires a very large number of classes, around 2 to 3 thousand at a minimum, since it requires one class per star. Usually, we deal with a few classes and hundreds to thousands of training vectors. Generating that many training vectors, e.g. from noise, for each star could in theory be done as well. A feedforward network with so many outputs would involve a very large network and be very difficult to train. A simple RBF network trained with RBF would require several thousand neurons. A more sophisticated algorithm like the DDA should ameliorate this “Curse of Dimensionality”, but we have not yet proved this. So for the sake of simplicity and practicality, we tried the basic nearest neighbor method, using only one prototype per star, and it worked quite well. If implemented in neural network type architecture, i.e. distributed parallel processing elements with each holding one prototype to compare to the input, such a method would provide a fast and robust solution for small FOV star trackers.

A number of possible hardware implementations were discussed based on what is either currently available or a short extrapolation from available hardware. At this time there seem to be two possible ways for implementation, the IBM ZISC036 or the CNAPS. While the latter performs at least 20 times faster than a modern pentium computer, the former produces results in a few microseconds. In either case, the star trackers could be fast enough even for spinnng satellites.

Acknowledgments

One of us (CSL) wishes to thank the Nordic Research Academy, NorFA, for the fellowship that allowed this research project to take place. The present work is also supported in part by the Swedish Research Council for Engineering Sciences (TFR) and by the Magn Bergvall Foundation (MBS).

References

[1] TRW Space Log - Winter 68-69 and 1970-71, TRW Systems Group, TRW Inc

[2] CT-632 and CT-633 Data Sheet. Ball Aerospace Systems Divistion, Boulder, CO

[3] Curtis Padgett, Kenneth Kreutz-Delgado and Surapohol Undomkesmalee, Evaluation of Star Identification Techniques (Dec, 1994).

[4] E. Groth, A pattern-matching algorithm for two-dimensional coordinate lists, Astronomical Journal, 91:1244-1248, 1996

[5] P. Alveda, A. Miguel San Martin, *Neural Network Star Pattern Recognition for Spacecraft Attitude Determination and Control*, Advances in Neural Information Processing Systems I, pp.314-322, ed. D. S. Touretzky, Sorgan Kaufmann, 1994.

[6] G. Bardwell, *On-Board Artificial Neural Network Mulit-Star Identification System for 3-Axis Attitude Determination*, Acta Astronautica Vol. 35, Suppl., pp. 753-761, 1995.

[7] M. J. Domeika, E. W. Page, G. A. Tagiarini, *Neural Network Approach to Star Field Recog nition*, Applications and Science of Artificial Neural Networks, edt. S. K. Rogers, D. W. Ruck, SPIE Vol. 2492, Part Two, pp.1007-1017 (1995).

[8] Th. Lindblad, C. S. Lindsey, Å . Eide, *Celestal Star Determination using Artificial Neural Networks*, Proc. of Int. Federation of Automatic Control, San Francisco, Ca. USA, June 30 - July 7, 1996.

[9] C. S. Lindsey, Th. Lindblad, Åge Eide, *A Method for Star Identification using Neural Net works*, to be presented at the Applications and Science of Artificial Neural Networks Session, SPIE Orlando, Fla. USA, April 1997.

[10] The ZISC036 Data Book, IBM Essonnes, France and J-P LeBouquin, *IBM Microelectronics ZISC, Zero Instruction Set Computer, Preliminary Information,* Poster Show WCNN, San Diego, CA, 1994 and addendum to conference proc. and J-P LeBouquin and M. Grandguillot, IBM Microelectronics, Essonnes.

[11] Th. Lindblad, C. S. Lindsey and Å. Eide, *The IBM Zero Instruction Set Computer ZISC036 A Hardware Implemented Radial Basis Function Neural Network*, CRC Industrial Engineering Handbook, Sect 7.23

[12] The CNAPS Data Sheets, Adaptive Solutions, Inc, Beaverton, CO

[13] Ö. Solberg and A. Bolseth, Stars and satellite navigation 2, Diploma Work, Ostfold College, Halden, Norway

[14] M.R. Berthold and J. Diamond: *Boosting the Performance of the RBF Networks with Dy namic Decay Adjustment* in G. Tesauro, D.S. Touretsky and T.K. Leen (eds) *Advances in Neural Information Processing Systems* 7, MIT Press, Cambridge, MA, 1995

[15] Th. Lindblad, G. Székely, M. L. Padgett, Å . Eide, C.S. Lindsey, *Implementing the Dynamic Decay Adjustment Algorithm in a CNAPS Parallel Computer System*, Nucl. Instr. Meth. A., to be published.

Appendix

The key to the Dynamic Decay Algorithm (DDA) is the use of *two thresholds* denoted q_{pos} and q_{neg}, such that for every pattern **x** of class *c*, we have

$$\theta_{pos} \leq R_i^c(\mathbf{x}) \quad \text{and} \quad \theta_{neg} > R_j^k(\mathbf{x}),$$

for at least one *i* in the range $1 \leq i \leq m_c$, for all $k \neq c$ and for all *j* in the range $1 \leq j \leq m_k$. Here *R* is the response function and are the number of prototypes of class *c* and *k* , respectively. Only when the activation of proper classes are all below θ_{pos} , a new prototype will be added. The following pseudo code shows what the training for one new pattern **x** of class *c* looks like:

```
1:      find i : $1 \leq i \leq m_c \wedge R_i^c(\mathbf{x}) = \max\{ R_j^c(\mathbf{x}), 1 \leq j \leq m_c \}$
2:      if $R_i^c(\mathbf{x}) \geq \theta_{pos}$
3:      [ $A_i^c + = 1.0$
4:  else
5:      [ add new prototype $p^c m_c+1$ with:
6:      [ $\mathbf{r}^c m_c+1 = \mathbf{x}$
7:      [ $\sigma^c m_c+1 = \max\{ \sigma : k \neq c \wedge 1 \leq j \leq m_k \wedge R^c m_c+1 (\mathbf{r}_j^k) < \theta_{neg} \}$
8:      [ $A^c m_c+1 = 1.0$
9:      $\forall k \neq c, \quad 1 \leq j \leq m_k :$   if $R_j^k(\mathbf{x}) > \theta_{neg}, \quad \sigma_j^k = \max\{ \sigma : R_j^k(\mathbf{x}) < \theta_{neg} \}$
```

The choice of the two new parameters does not appear to be critical, but some default values can be used (they need to be different or one ends up with the P-RCE like situation). In contrast to probabalistic neural networks, we have here a situation where the hidden layer can include more training patterns. This calls for the use of an initial weight referred to as A_i^c .