We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

4,800
Open access books available

122,000
International authors and editors

135M
Downloads

Our authors are among the

154
Countries delivered to

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities

CLARIVATE ANALYTICS BOOK CITATION INDEX INDEXED

WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com

7

Spatial Voting With Data Modeling

Holger Marcel Jaenisch, Ph.D., D.Sc.
Licht Strahl Engineering INC (LSEI)
United States of America
James Cook University
Australia

1. Introduction

Our detailed problem is one of having multiple orthogonal sensors that are each able to observe different objects, but none can see the whole assembly comprised of such objects. Further, our sensors are moving so we have positional uncertainties located with each observed object. The problem of associating multiple time based observations of a single object by fusing future estimate covariance updates with our best estimate to date and knowing which estimate to assign to which current location estimate becomes a problem in bias elimination or compensation. Once we have established which objects to fuse together, next we must determine which objects are close enough together to be related into a possible assembly. This requires a decision to determine which objects to group together. But if a group of nearby objects is found, how are their spatial locations best used? Simply doing a covariance update and combining their state estimates yields a biased answer, since the objects are "not" the same and "not" superimposed. Therefore, naive covariance updating yields estimates of assemblies within which we find no objects. Our proposed spatial correlation and voting algorithm solves this spatial object fusion problem.

The spatial voting (SV) concept for the object to assembly aggregation problem is based on the well-known principles of voting, geometry, and image processing using 2D convolution (Jaenisch et.al., 2008). Our concept is an adaptation of the subjects as covered in Hall and McCullen (2004) which are limited to multiple sensors, single assembly cases. Our concept is an extension to multiple orthogonal sensors and multiple assemblies (or aspects of the same assembly). Hall and McCullen describe general voting as a democratic process. Hard decisions from M sensors are counted as votes with a majority or plurality decision rule. For example, if M sensors observe a phenomena and make an identity declaration by ranking n different hypothesis, summing the number of sensors that declare each hypothesis to be true and taking the largest sum as the winner forms an overall declaration of identity. From this, it is easy to see that voting many times reduces to probabilities or confidences and their efficient mathematical combination. Typically, this is where either Bayes' rule or other covariance combining methods are used such as covariance updating and Klein's Boolean voting logic (Klein, 2004). However, all of these methods are still probabilistic.

SV reduces object location uncertainty using an analog method by stacking and tallying, which results in a vote. It is common with probabilistic methods to assume the assembly is at the center of the estimate with some uncertainty. In our spatial approach, we don't make that assumption. Rather, SV states that the object is located with confidence somewhere in the area. The larger the area, the higher the confidence that one or more objects will be contained within its boundary, which is the opposite approach taken in traditional covariance confidence updating.

2. Approach

In SV, the sensor report ellipses are stacked and the aggregation becomes a tally or vote. This results in a growing landscape of overlapping ellipses. By collecting assembly object estimates throughout one full epoch, a full landscape of all the best available assembly sensor reports and locations are obtained. By convolving this array with a 2-D spatial kernel, it is possible to achieve correlation based on a mission controlled setting for choosing a 2-D kernel of the necessary spatial extent. For our application, the grid exists as a 128 x 128 element array that is a total size of 128 x 128 meters. Each unit on the grid is 1 square meter, and we wish to fuse elements up to 5 meters apart using a suitable 5m x 5m kernel. Spatial averaging combines by blending high regions in the array that are approximately 5 meters apart or less into a continuous blob. The blob is then isolated by calculating an adaptive threshold from the frame and zeroing everything below the threshold. The resultant hills are projected down to the x and y-axis independently and the local sub pixel regions are extracted. The pixel regions are used to calculate spatial extent in the form of a covariance for the extracted region for the newly discerned assembly. Finally, each located assembly is evaluated to estimate total confidence of the assembly being significant or anomalous and the assembly labeled accordingly.

3. Algorithm Description

A flowchart for the SV algorithm is given in Fig. 1 on the top of the next page. SV has been implemented in MathCAD 14 and this implementation is included in Fig. 8 – 13 as a SV simulation shown later in this chapter. The SV simulation allows Monte Carlo cases to be generated (explained in Section 5). The example case described in this section is the first Monte Carlo case (FRAME=0) generated by the MathCAD simulation. To initialize the SV process, first define the dimensions of the detection space. For our example case, the size of the grid is 128 x 128 grid units. Each grid unit is 1 meter by 1 meter in size. Next, define the spatial extent over which objects are to be fused together as assemblies. This defines the size (or spatial extent) of the spatial convolution kernel (spatial correlator) that we apply to the detection space once the centroid and covariance defined ellipse representations (derived encompassing rectangles) are stacked. The kernel size is given by

$$Kernel\,Size = \frac{Spatial\,Extent}{Grid\,Resolution} \tag{1}$$

where kernel size is in number of grid units, and spatial extent and grid resolution are in meters.

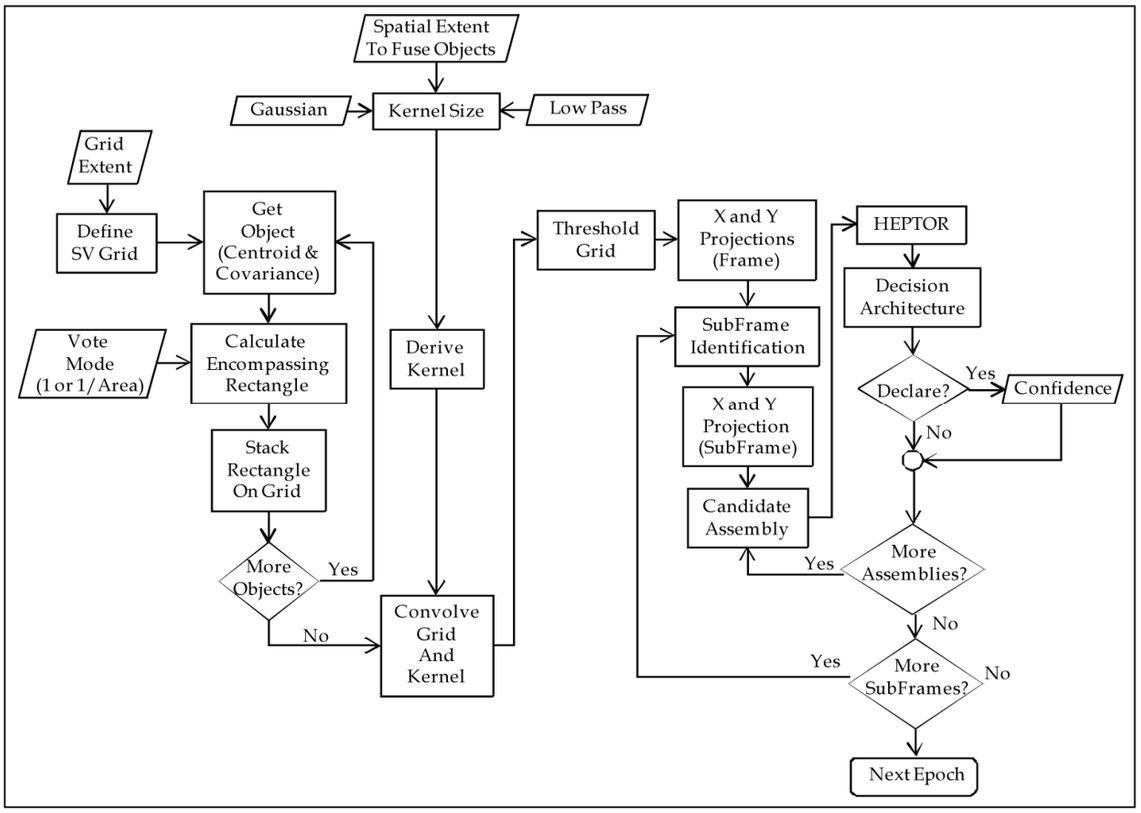


Fig. 1. Spatial Voting (SV) process flowchart.

The spatial convolution kernel (Jain, 1989) (NASA, 1962) is the equivalent kernel shown in Equation (2c), which is a 5m x 5m (with very little effect past 2m) matrix resulting from the convolution of the two low-pass or smoothing kernels used in image processing given in Equations (2a) and (2b).

$$Kernel_{LowPass} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 3 & 1 \\ 1 & 1 & 1 \end{bmatrix} \text{(a)} \quad Kernel_{Gaussian} = \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \text{(b)} \tag{2}$$

(3m x 3m) (3m x 3m)

$$Kernel_{SpatialConvolutionKernel} = Kernel_{LowPass} ** Kernel_{Gaussian} = \begin{bmatrix} 1 & 3 & 4 & 3 & 1 \\ 3 & 11 & 16 & 11 & 3 \\ 4 & 16 & 24 & 16 & 4 \\ 3 & 11 & 16 & 11 & 3 \\ 1 & 3 & 4 & 3 & 1 \end{bmatrix} \text{(c) (5m x 5m)}$$

The spatial convolution kernel in Equation (2c) defines the shape of a Gaussian distribution, and by convolving the spatial convolution kernel with the detection space, it is converted into a correlation map describing how each pixel neighborhood in the detection space is correlated with the spatial convolution kernel.

To enlarge the kernel to match the spatial extent, the spatial convolution kernel in Equation (2c) is convolved with itself until the equivalent kernel size corresponds with the extent. The number of times that the kernel in Equation (2c) is convolved with itself is given in Equation (3), and the final equivalent kernel requiring convolving the original spatial convolution kernel *n* times with itself is given in Equation (4) as

$$N_{comb} = floor(\tfrac{1}{2}(Kernel\ Size - 3)) \tag{3}$$

$$Kernel_{SC} = Kernel_{SCK} ** Kernel_{n-1} \tag{4}$$

where $Kernel_{n-1}$ is the result of convolving the spatial convolution kernel with itself *n-1* times.

The estimated object's position is described by the sensor report using position centroid and covariance (which defines the location and size of the uncertainty region). The centroid and covariance are given by

$$X = \begin{bmatrix} \mu_x \\ \mu_y \end{bmatrix} \qquad \Lambda = \begin{bmatrix} \sigma_{xx} & \sigma_{xy} \\ \sigma_{yx} & \sigma_{yy} \end{bmatrix} \qquad \mu_x = \frac{1}{N}\sum_i x_i \qquad \mu_y = \frac{1}{N}\sum_i y_i$$

$$\sigma_{xx} = \frac{1}{N}\sum_{i=1}^{N}(x_i - \mu_x)^2 \qquad \sigma_{xy} = \frac{1}{N}\sum_{i=1}^{N}(x_i - \mu_x)(y_i - \mu_y) \qquad \sigma_{yy} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \mu_y)^2 \tag{5}$$

where x_i and y_i are the individual sensor reports of position estimates used to derive the centroid and covariance. We begin with the sensor report consisting of the position centroid X and the covariance Λ. From the covariance, the one sigma distance from the centroid along the semi-major (a) and semi-minor (b) axes of the ellipse are given by

$$a^2 = \frac{1}{2}\left[\sigma_{xx} + \sigma_{yy} + \sqrt{[\sigma_{yy} - \sigma_{xx}]^2 + 4\sigma_{xy}^2}\right] \qquad b^2 = \frac{1}{2}\left[\sigma_{xx} + \sigma_{yy} - \sqrt{[\sigma_{yy} - \sigma_{xx}]^2 + 4\sigma_{xy}^2}\right] \tag{6}$$

(along Semi-major axis) (along Semi-minor axis)

and the angle of rotation θ of the semi-major axis is given in Equation (7) and the lengths *a* and *b* and the rotation angle θ are used to define a rotated ellipse of the form given in Equation (7).

$$\theta = \frac{1}{2}\arctan\left(\frac{2\sigma_{xy}}{\sigma_{yy} - \sigma_{xx}}\right)$$

$$\frac{[(x-h)\cos(\theta) - (y-k)\sin(\theta)]^2}{a^2} + \frac{[(y-k)\sin(\theta) + (x-h)\sin(\theta)]^2}{b^2} = 1 \qquad \sigma_{xx} > \sigma_{yy}$$

$$\frac{[(x-h)\cos(\theta) - (y-k)\sin(\theta)]^2}{b^2} + \frac{[(y-k)\sin(\theta) + (x-h)\sin(\theta)]^2}{a^2} = 1 \qquad \sigma_{xx} < \sigma_{yy} \tag{7}$$

where h is the centroid x value, k is the centroid y value, and a and b are defined in Equation (6). The ellipse in Equation (7) defines the perimeter of the elliptical region, and to define the entire region encompassed by the ellipse, we simply change the equality (=) in Equation (7) to the less than or equal to (≤) inequality so that the function includes not only the boundary, but also the locations contained within the boundary.

Because the actual location of each object is unknown, the only information that is available is contained in the sensor report in the form of a centroid and covariance. It is an incorrect assumption that the object is located at the center of the ellipse; because if this were true then the covariance information would not be needed since the true position would be defined by the centroid alone.

The semi-major axis length, semi-minor axis length, and rotation angle are converted into covariance using

$$\sigma_{xx} = a \cdot \cos(\theta)^2 + b \cdot \sin(\theta)^2 \quad \sigma_{yy} = a \cdot \sin(\theta)^2 + b \cdot \cos(\theta)^2 \quad \sigma_{xy} = \sigma_{yx} = (b - a) \cdot \cos(\theta)\sin(\theta) \tag{8}$$

Fig. 2 shows 2 examples of starting with the rotation angle and semi-major and semi-minor axis lengths deriving the covariance matrix and corresponding ellipse.

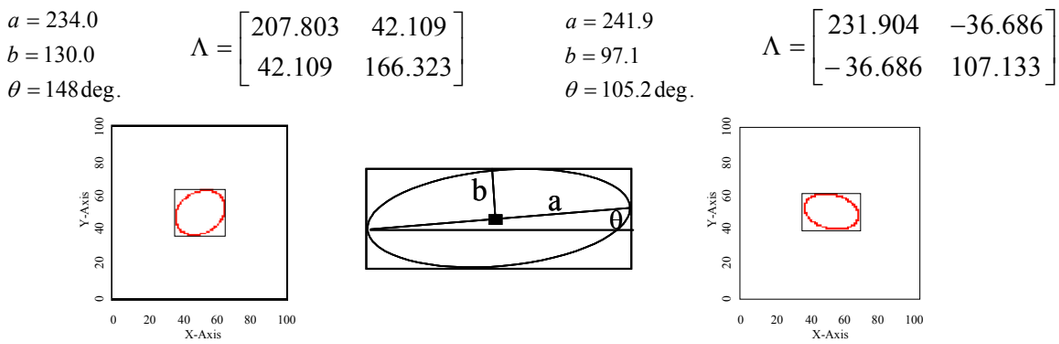


Fig. 2. Starting with the semi-major axis length, semi-minor axis length, and rotation angle, the covariance matrices and ellipses above are derived using Equation (8).

If the ellipses are placed into the detection grid directly, artifacts are introduced by aliasing and pixelization from the boundary of the ellipse. Also, as the size of an ellipse to place becomes small relative to the detection grid size, the overall shape approaches the rectangle. Therefore, to minimize scale dependent artifacts, encompassing rectangles with well-defined boundaries replace each ellipse (Press et.al., 2007). The semi-major axis of the ellipse is the hypotenuse of the triangle, and completing the rectangle yields the first approximation to an equivalent rectangle. Finally, the width of the rectangle is scaled to the semi-minor axis length to preserve the highest spatial confidence extent reported in the covariance. The length of the sides of the rectangle that are placed instead of the ellipse are given by

$$\left.\begin{array}{c} x = 2a\cos(\theta) \\ y = Max\,(2a\sin(\theta), 2b\sin(\theta)) \end{array}\right\} if\ \sigma_{xx} > \sigma_{yy}$$
$$\left.\begin{array}{c} x = Max\,(2a\sin(\theta), 2b\sin(\theta)) \\ y = 2a\cos(\theta) \end{array}\right\} if\ \sigma_{yy} > \sigma_{xx} \quad (9)$$

where θ again is the rotation angle of the semi-major axis given in Equation (7) above. The length of the semi-minor axis modifies the size of the rectangle subject to the conditions given in Equation (9), which preserves the axis with the greatest spatial location confidence. The value at each grid location inside of the rectangle is

$$Value = \frac{1}{N} = \frac{1}{x \cdot y} \quad (10)$$

where x and y are the sides of the rectangle, and if the analysis is done under the assumption that a smaller area increases the confidence that it contains an object. If on the other hand the converse is true, then N = 1, which implies that the confidence of an object being contained in a larger area is weighted higher than the confidence in a smaller area when the spatial vote or stacking occurs. Both forms may be used to determine how many pedigree covariance reports are associated with each respective assembly by using the sum of the values mapped into the assembly location as a checksum threshold.

Now that the rectangle extent and value is defined, the rectangles are stacked into the detection grid one at a time. This is accomplished by adding their value (1 or 1/Area depending on how scoring is done for testing sensor report overlap. If 1's are placed the number of overlaps is max value in subarray, if 1/Area is used sum>Area indicates overlap) in each rectangle to the current location in the grid where the rectangle is being stacked. Fig. 3 (left) shows as an example (obtained from the MathCAD 14 implementation in Fig. 8-13) 38 original sensor reports along a stretch of road, and (left center) is the detection grid after stacking the 38 rectangles that represent each of the sensor reports and applying the spatial convolution kernel.

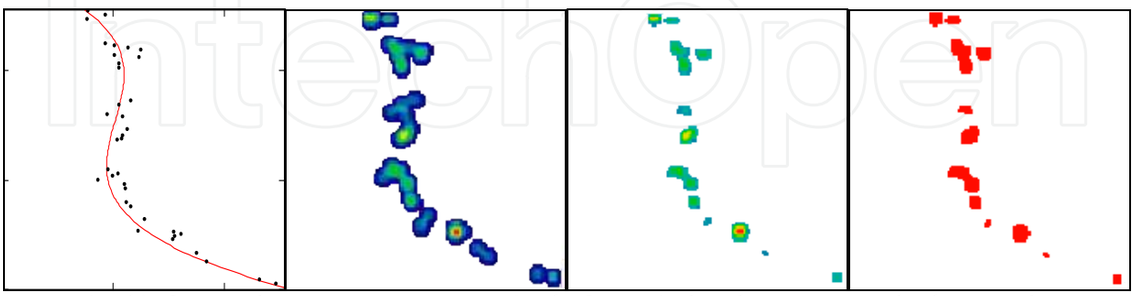


Fig. 3. (Left) Original 38 sensor reports, (Left Center) the detection grid after stacking and applying the spatial convolution kernel, (Right Center) After applying the threshold, and (Right) locations in the graph on the left converted into a 0/1 mask.

Next, automatically calculate a threshold to separate the background values in the detection grid from those that represent the assemblies (anomaly detection). This threshold is

calculated as the minimum value of the non-zero grid locations plus a scale factor times the range of the values in the detection grid (set in the MathCAD implementation as 0.3 times the range (maximum minus minimum) of the non-zero values) in Fig. 3 (left center). Values that occur below this threshold are set to zero, while those that are above the threshold retain their values. Fig. 3 (right center) shows an example of applying the threshold to the detection grid in Fig. 3 (left center), resulting assembly blobs (fused objects) are shown. Also shown in Fig. 3 (right) is an example of the mask that is formed by setting all values above the threshold to one and all those below the threshold to zero.

In order to isolate the assemblies that now are simply blobs in the detection grid, we compute blob projections for both the x-axis and the y-axis of the grid by summing the mask in Fig. 3 (right) both across the rows (y-axis projection) and down the columns (x-axis projection). Fig. 4 shows examples of these assembly shadow projections, which are calculated using

$$DX_j = \sum_{k=0}^{ngrid-1} D_{k,j} \quad \text{and} \quad DY_i = \sum_{k=0}^{ngrid-1} D_{i,k} \tag{11}$$

where D is the array shown in Fig. 3 (right), $ngrid$ is the grid size (128 for our example), DX_j is the x-axis projection for the j^{th} column, and DY_i is the y-axis projection for the i^{th} row.

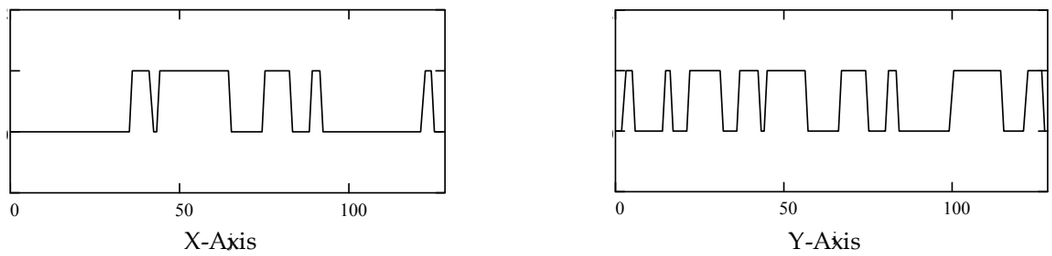


Fig. 4. Example assembly shadow projections for the x-axis (left) and y-axis (right) for the mask shown in Fig. 4 (right).

Once these assembly shadow projections are calculated, they are renormalized so that all non-zero locations have a value of one while zero locations remain zero. Using the assembly shadow projections, horizontal and vertical lines are placed across the detection grid corresponding to the transition points from zero to one and from one to zero in the graphs in Fig. 4. Regions on the grid that are formed by intersections of these lines are labeled 1 through 45 in Fig. 5, and are the candidate assemblies that are identified (including zero frames).

Each of the 45 candidate assembly subframes are processed to remove zero frames by determining if any non-zero locations exist within its boundary. This is done by extracting the assembly subframe into its own separate array and calculating the maximum value of the array. If the maximum value is non-zero, then at least one grid unit in the array is a part of an object assembly and is labeled kept.

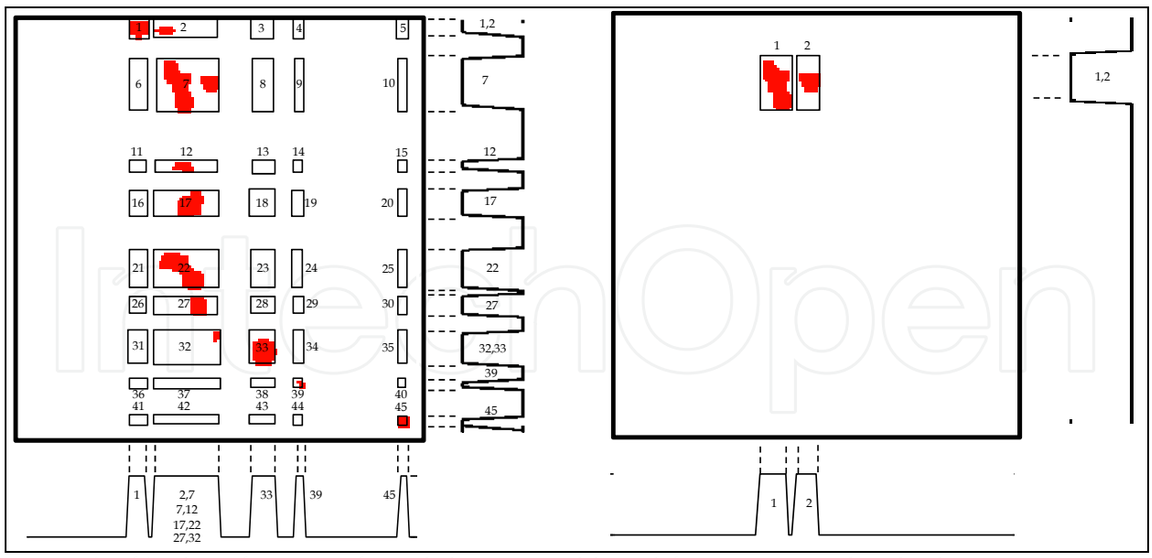


Fig. 5. (Left) Identification of candidate assembly subframes using the shadow projection in Equation (11) and Fig. 3, (Center) Applying the shadow projection algorithm a second time to Region 7 to further isolate assemblies. After processing the subframes a second time, a total of 12 candidate assemblies have been located.

This is repeated for each of the subframes that are identified. Once the subframes have been processed, we repeat this process on each subframe one at a time to further isolate regions within each subframe into its own candidate assembly. This results in subframes being broken up into smaller subframes, and for all subframes found, the centroid and covariance is calculated. This information is used by the object and assembly tracking routines to improve object and assembly position estimates as a function of motion and time. As a result of this processing, the number of candidate assembly subframes in Fig. 5 is reduced from 45 to 12 (the number of isolated red regions in the grid). The final results after applying the SV algorithm to the stacked rectangles in Fig. 6 (center) results in the graphic shown in Fig. 6 (right).

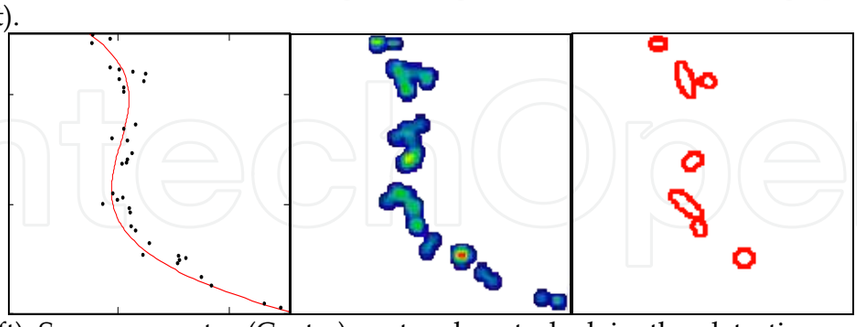


Fig. 6. (Left) Sensor reports, (Center) rectangles stacked in the detection grid with SV smoothing applied, (Right) final result after applying spatial voting.

4. Heptor

From an isolated SV target, we have available the geospatial distribution attributes (X and Y or latitude and longitude components characterized independently, including derivatives across cluster size transitions), and if physics based features exist, Brightness (including

derivatives across cluster size transitions), Amplitude, Frequency, Damping, and Phase. Each of these attributes is characterized with a fixed template of descriptive parametric and non-parametric (fractal) features collectively termed a Heptor (vector of seven (7) features) and defined in Equations (12) – (18) as:

$$\sigma = \sqrt{\tfrac{1}{N-1}\sum_{j=1}^{N}(x_j - \bar{x})^2} \quad (12) \qquad Skew = \tfrac{1}{N}\sum_{j=1}^{N}\left[\frac{x_j - \bar{x}}{\sigma}\right]^3 \quad (13)$$

$$Kurt = \left\{\tfrac{1}{N}\sum_{j=1}^{N}\left[\frac{x_j - \bar{x}}{\sigma}\right]^4\right\} - 3 \quad (14) \qquad M6 = \left\{\tfrac{1}{N}\sum_{j=1}^{N}\left[\frac{x_j - \bar{x}}{\sigma}\right]^6\right\} - 15 \quad (15)$$

$$M8 = \left\{\tfrac{1}{N}\sum_{j=1}^{N}\left[\frac{x_j - \bar{x}}{\sigma}\right]^8\right\} - 105 \quad (16) \qquad \mathrm{Re}(J_{i+1}) = \min\left(\lim_{\Delta J \to 0}\left[\frac{\log\left(\frac{Range}{\sigma N J_i}\right)}{\log\left(\frac{1}{N}\right)}\right]\right) \quad (17)$$

$$DH = 1 + \log\left(\tfrac{1}{N}\sum_{j=1}^{N-1}\sqrt{1 + \left(\tfrac{x_{j+1} - x_j}{Range}\right)^2}\right) \quad (18)$$

The basic Heptor can be augmented by additional features that may exist, but in their absence, the Heptor represents an excellent starting point. Equations (19) to (23) lists some additional features that can be used to augment the Heptor.

$$\bar{x} = \tfrac{1}{N}\sum_{j=1}^{N} x_j \quad (19) \qquad Min = \min(x) \quad (20)$$

$$Max = \max(x) \quad (21) \qquad Range = Max - Min \quad (22)$$

$$ChiSq = N - (N-1)\sigma^2 \quad (23)$$

The features are associated with a category variable (class) for 1=no target or nominal, and 2=target or off-nominal. The next step is to drive a classifier to associate the features with class in a practical fashion. Here we use the Data Model since very few examples are available, and not enough to discern statistical behavior. For this, we want to populate a knowledge base which encodes all examples encountered as observed and identified species. This mandates a scalable, bottom-up approach that is well suited to the Group Method of Data Handling (GMDH) approach to polynomial network representation.

5. SV Simulation in MathCAD 14

SV has been implemented in MathCAD 14, C, and Java. Included in this work (Figs. 8-13) is the MathCAD 14 implementation, which facilitates the ability to generate Monte Carlo ensemble cases used in deriving the Data Model decision architecture described in later sections of this chapter.

Spatial Voting Algorithm Demonstration

A. SV Initialization

Vote mode = 0 for 1/N filling, and no single component targets

$votemode := 0$

Vote mode = 1 for filling with 1's

$snglcmptgt \equiv 0$ Display Single Componet Targets? (0 = single components not recognized as targets)

$maxseparation := 5$ max obj separation $nsize := 128$ Grid Size $res := 1$ m

$isigma := 1$ For placing ellipses $ntgt := 3$ Number of targets per epoch $sx := 6$ Max number of components per target

B. Kernel Initialization

$i := 0..nsize - 1$ $j := 0..nsize - 1$ $t := nsize \cdot res$ $t = 128$ meters by $t = 128$ meters grid ($res = 1$ meter resolution)

$ksize1 := \frac{maxseparation}{res}$ $ksize1 = 5$ $ksize := floor\left(\frac{ksize1}{2}\right) \cdot 2 + 1$ $ksize = 5$ kernel size to detect the selected max separation

$ncomb := floor\left(\frac{ksize - 3}{2}\right) - 1$ $ncomb = 0$ number of LP/G passes required to build ksize kernel starting with initial LP/G

Initial LowPass and Gaussian kernels

$$LP := \begin{pmatrix} 1 & 1 & 1 \\ 1 & 3 & 1 \\ 1 & 1 & 1 \end{pmatrix} \quad G := \begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix}$$

C. Calculate final correlator kernel

$LPK_{nsize-1,\,nsize-1} := 0$ $GK_{nsize-1,\,nsize-1} := 0$ $ij := 0..2$ $ik := 0..2$

$LPK_{floor[(nsize+1)\cdot 0.5]-1+ij,\,floor[(nsize+1)\cdot 0.5]-1+ik} := \frac{LP_{ij,\,ik}}{11}$ $GK_{floor[(nsize+1)\cdot 0.5]-1+ij,\,floor[(nsize+1)\cdot 0.5]-1+ik} := \frac{G_{ij,\,ik}}{16}$

$KCOMB := FFTCONV(LPK, GK)$ $K1 := KCOMB$ $K2 :=$ if $ncomb \geq 1$ $K2 := if(ksize1 \leq 2, LPK, K2)$

$K2 := if(ksize1 > 2 \wedge ksize1 \leq 4, GK, K2)$

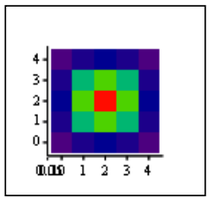
for $i \in 1..ncomb$ $ij := 0..ksize - 1$ $ik := 0..ksize - 1$

$K1 \leftarrow FFTCONV(K1, KCOMB)$

$abc \leftarrow 0$

$K1$

$KF_{ij,\,ik} := K2_{ij+\frac{nsize}{2}-2,\,ik+\frac{nsize}{2}-2}$



KF

D. Initialize detection grid and road

$A_{i,\,j} := 0$ $ir := 0..11$ $rx_{ir} := ir \cdot \frac{nsize - 1}{11}$ $ry_{ir} := nsize \cdot RD(kkk1 + ir)$

$rdx^{\langle 0 \rangle} := rx$ $rdx^{\langle 1 \rangle} := rx^2$ $rdx^{\langle 2 \rangle} := rx^3$ $norder := 3$ $MM := FIT(12, 3, rdx, ry, 8)$

$ir1 := 0..nsize - 1$ $rpoly_{ir1} := \sum_{i=0}^{norder} \left(MM_i \cdot ir1^i\right)$ $rpoly1_{ir} := \sum_{i=0}^{norder} \left[MM_i \cdot \left(rx_{ir}\right)^i\right]$

E. Initialize truth locations

$sy := \frac{nsize - 1}{ntgt}$ $jtgt := 0..ntgt - 1$ $k := 0..ntgt - 1$ $ty_{jtgt} := floor[[(jtgt + 1) \cdot sy + jtgt \cdot sy + 1] \cdot 0.5]$ $tx_{jtgt} := rpoly_{ty_{jtgt}}$

$tx1_{jtgt} := tx_{jtgt} + sx \cdot RD(kkk1 + jtgt)$ $ty1_{jtgt} := ty_{jtgt} + sx \cdot RD(kkk1 + jtgt \cdot 2)$ $rottheta := 180$

$\beta_k := RD[kkk1 + (k + 1) \cdot 5] \cdot \frac{\pi}{180} \cdot rottheta$ $sval1_k := 9 \cdot RD[kkk1 + (k + 1) \cdot 3]$

$sval2_k := 9 \cdot RD[kkk1 + (k + 1) \cdot 4]$ $txx_k := sval1_k \cdot \cos\left(\beta_k\right)^2 + sval2_k \cdot \sin\left(\beta_k\right)^2$ $ncomp_k := 2 + 2 \cdot RD(kkk1 + 2 + k)$

$tyy_k := sval1_k \cdot \sin\left(\beta_k\right)^2 + sval2_k \cdot \cos\left(\beta_k\right)^2$ $txy_k := \left(sval2_k - sval1_k\right) \cdot \cos\left(\beta_k\right) \cdot \sin\left(\beta_k\right)$ $CT_k := \begin{pmatrix} txx_k & txy_k \\ txy_k & tyy_k \end{pmatrix}$

$jx_k :=$ $A \leftarrow 0$; for $j \in 0..ncomp_k - 1$; $A_j \leftarrow tx1_k + txx_k \cdot [2 \cdot RD[kkk1 + (k + 1 + j) \cdot 6] - 1]$; A

$jy_k :=$ $A \leftarrow 0$; for $j \in 0..ncomp_k - 1$; $A_j \leftarrow ty1_k + tyy_k \cdot [2 \cdot RD[kkk1 + (k + 1 + j) \cdot 7] - 1]$; A

$ix := MAPCOMP(jx, jy)_0$ $iy := MAPCOMP(jx, jy)_1$ $n := rows(ix)$ $RD(x) \equiv \frac{\sin\left[(x)^{exp(1)}\right] + \cos\left[(x)^{exp(1)}\right] + 2}{4}$

Fig. 8. Part 1 of MathCAD 14 implementation of SV.

A. Map in false alarm detections

```
FA := | m ← 0
      | for p ∈ 0..nsize − 1
      |   if RD[kkk1 + (p)·8] > 0.75
      |     | tmp ← [RD[kkk1 + (p)·8] − 0.8]·nsize
      |     | FA_{m,0} ← rpoly_p + tmp
      |     | FA_{m,1} ← p
      |     | m ← m + 1
      | FA
```

k := 0..rows(FA) − 1 ix_{n+k} := FA_{k,0} iy_{n+k} := FA_{k,1}

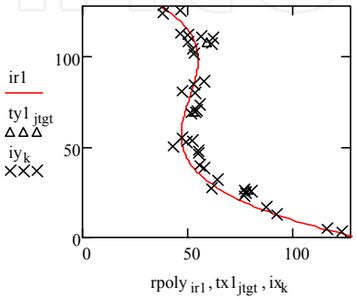
n := rows(ix) k := 0..n − 1 **ix and iy are the detection centroids**

roadx_{ir1} := rpoly_{ir1} roady_{ir1} := ir1

```
min2(A,z) ≡ | a ← A
            | a ← z if z < a
            | a

max2(A,z) ≡ | a ← A
            | a ← z if z > a
            | a
```

```
MAPCOMP(jx,jy) ≡ | m ← 0
                 | for i ∈ 0..rows(jx) − 1
                 |   | n ← rows(jx_i)
                 |   | for j ∈ 0..n − 1
                 |   |   | ix_m ← (jx_i)_j
                 |   |   | iy_m ← (jy_i)_j
                 |   |   | m ← m + 1
                 | A_0 ← ix
                 | A_1 ← iy
                 | A
```



Triangles are truth targets, X's are detections

B. Assign detection covariances

$\beta_k := RD[kkk1 + (k+1)\cdot 9]\cdot\frac{\pi}{180}\cdot rottheta$ $sval1_k := 4\cdot RD[kkk1 + (k+1)\cdot 10]$ $sval2_k := 4\cdot RD[kkk1 + (k+1)\cdot 11]$

$txx_k := sval1_k\cdot\cos(\beta_k)^2 + sval2_k\cdot\sin(\beta_k)^2$ $tyy_k := sval1_k\cdot\sin(\beta_k)^2 + sval2_k\cdot\cos(\beta_k)^2$ $XA_k := \begin{pmatrix} ix_k \\ iy_k \end{pmatrix}$ Centroids

$txy_k := (sval2_k - sval1_k)\cdot\cos(\beta_k)\cdot\sin(\beta_k)$ $ix_k := floor(ix_k + 0.5)$ $iy_k := floor(iy_k + 0.5)$ $C_k := \begin{pmatrix} txx_k & txy_k \\ txy_k & tyy_k \end{pmatrix}$ Covariances

A1 := PLCC(A, ix, iy, C, 2, isigma, 0) $C_k := C_k\cdot isigma^2$ A1 is for display only

$I2_0 := A$ $I2_1 := ix$ $I2_2 := iy$ $I2_3 := C$ $I2_4 := votemode$ $I2_5 := maxseparation$

C. Calculate Equivalent Rectangles and Place on SV Grid AA := PLCR(I2)

D. Convolve SV Kernel and Grid B := FFTCONV(AA_0, K2)

E. Auto calculate and apply threshold (D is now the 0/1 mask)

cmin := MIN(B) cmax := MAX(B) crng := cmax − cmin fac := 0.3 $D_{i,j} := if(B_{i,j} < cmin + fac\cdot crng, 0, 1)$ $B1_{i,j} := B_{i,j}\cdot D_{i,j}$

F. Compute Shadowing For Isolation

$DX_j := \sum_{k=0}^{nsize-1} D_{k,j}$ $DY_i := \sum_{k=0}^{nsize-1} D_{i,k}$ $DX_j := if(DX_j > 0, 1, 0)$ $DY_i := if(DY_i > 0, 1, 0)$

G. Extract Candidate Assemblies

$II_0 := AA_0$ $II_1 := DX$ $II_2 := DY$ $II_3 := B$ $II_4 := D$ $II_5 := snglcmptgt$ $II_6 := isigma$ $II_7 := votemode$ IFRAMES := FRMS(II)

NTargets := rows(IFRAMES) − 1 $D_{i,j} := 0$ $G_{i,j} := B_{i,j}\cdot D_{i,j}$ D := G icolor := max(D) isigma := 2 isigma here is for the target covariance

H. Draw Results (A1 and D)

Covar := CALCOV(IFRAMES⟨8⟩) icolor := if(icolor = 0, 1, icolor) xcent := IFRAMES⟨1⟩ ycent := IFRAMES⟨2⟩

A1 := PLCC(A1, xcent, ycent, Covar, icolor, isigma, 1) D := PLCC(D, xcent, ycent, Covar, icolor, isigma, 1)

I. Heptor Candidate Assemblies for Decision Architecture

AAA := HEPFRAME(IFRAMES, tx1, ty1) kkk1 ≡ FRAME + 1 kkk1 is the Monte Carlo parameter to vary (increment by +1) FRAME increments using the Animation tool under Tools/Animation/Record

Fig. 9. Part 2 of MathCAD 14 implementation of SV.

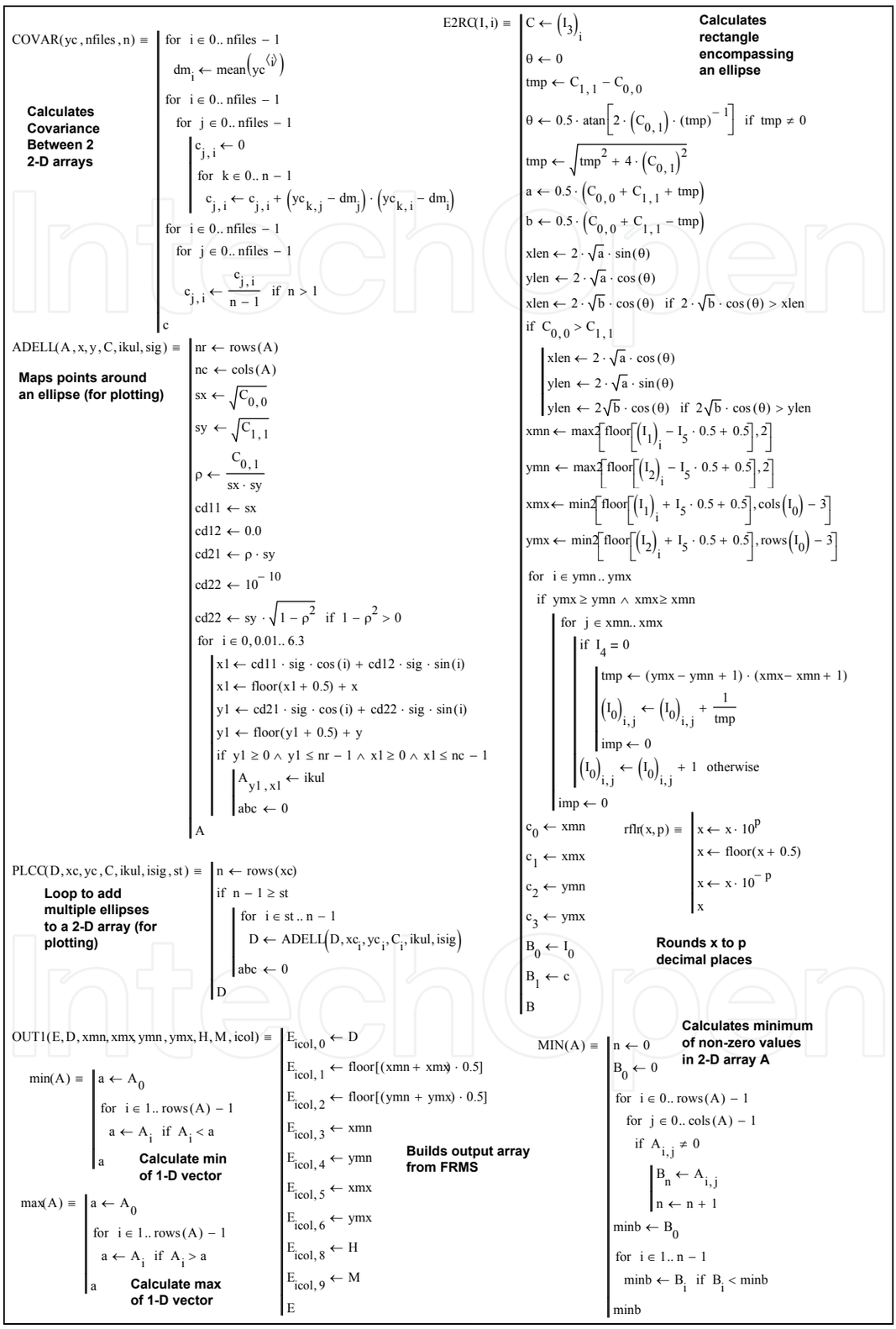


Fig. 10. Part 3 of MathCAD 14 implementation of SV.

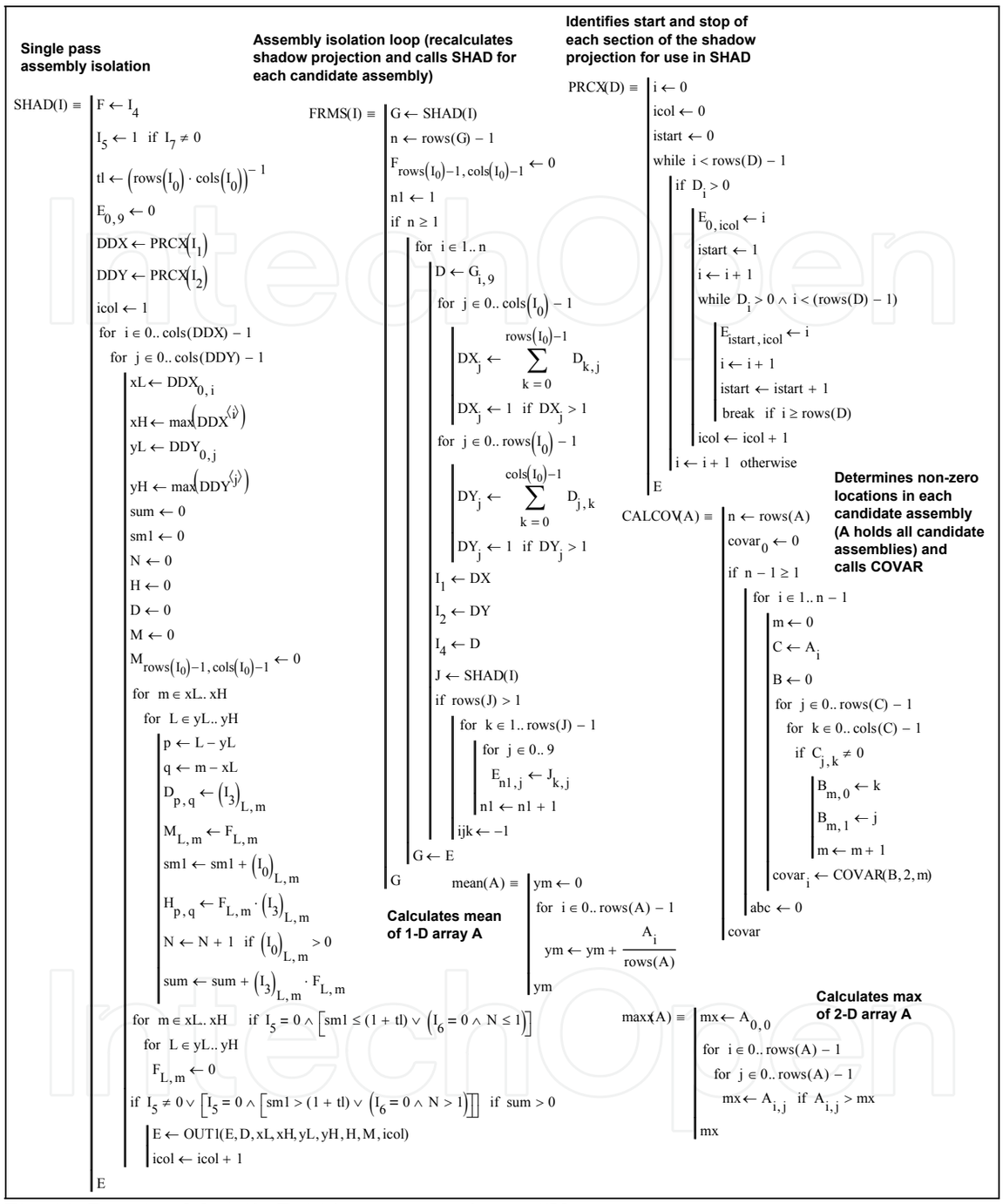


Fig. 11. Part 4 of MathCAD 14 implementation of SV.

Heptor value J in Equation 17

```
J(A) ≡ n ← rows(A)
       ystat ← 1
       yold ← 2
       j ← 0
       if (max(A) − min(A)) ≠ 0
         while |yold − Re(ystat)| > .0001
           yold ← Re(ystat)
           if Im(ystat) = 0
             lgr ← ln[|(Re(ystat))^−1|]
             lgi ← 1 − sign[(Re(ystat))^−1]
             lgi ← logi · π · 0.5
           otherwise
             xx ← Re(ystat) · (|ystat|)^−2
             yy ← −Im(ystat) · (|ystat|)^−2
             if xx · (|yy|)^−1 > 0
               acot ← atan(|yy| · xx^−1)
               abc ← 0
             if xx · (|yy|)^−1 = 0
               acot ← 0.5 · π
               abc ← 0
             if xx · (|yy|)^−1 < 0
               acot ← π + atan(|yy| · xx^−1)
               abc ← 0
             logr ← ln(xx^2 + yy^2) · 0.5
             logi ← sign(yy) · acot
           ystat ← max(A) − min(A)
           ystat ← ystat · n^−1 · Stdev(A)^−1
           ystat ← ln(ystat) · (ln(n^−1))^−1
           ystat ← ystat + lgr · (ln(n^−1))^−1
           ystat ← ystat + √−1 · lgi · (ln(n^−1))^−1
           j ← j + 1
         abc ← 0
       h ← Re(ystat)
       h ← 1 if h > 1
       h ← 0 if h < 0
       h ← h^−1 if h ≥ .5
       h ← (1 − h)^−1 otherwise
       h
```

PLCR places rectangles Calls E2RC (ellipse to rectangle) for multiple covariance ellipses Returns array with rectangles stacked along with min and max x and y locations for each rectangle extent

Heptor (Equations 12 - 18) in this chapter

$$HEP(A) \equiv \left| \begin{array}{l} ym \leftarrow \frac{1}{rows(A)} \cdot \sum_{i=0}^{rows(A)-1} A_i \\ h_0 \leftarrow \sqrt{\frac{1}{rows(A)-1} \cdot \sum_{i=0}^{rows(A)-1} (A_i - ym)^2} \\ h_1 \leftarrow \frac{1}{rows(A)} \cdot \sum_{i=0}^{rows(A)-1} \left[(A_i - ym) \cdot (h_0)^{-1} \right]^3 \\ h_2 \leftarrow \frac{1}{rows(A)} \cdot \sum_{i=0}^{rows(A)-1} \left[(A_i - ym) \cdot (h_0)^{-1} \right]^4 - 3 \\ h_3 \leftarrow \frac{1}{rows(A)} \cdot \sum_{i=0}^{rows(A)-1} \left[(A_i - ym) \cdot (h_0)^{-1} \right]^6 - 15 \\ h_4 \leftarrow \frac{1}{rows(A)} \cdot \sum_{i=0}^{rows(A)-1} \left[(A_i - ym) \cdot (h_0)^{-1} \right]^8 - 105 \\ h_5 \leftarrow J(A) \\ A \leftarrow (A - min(A)) \cdot (max(A) - min(A))^{-1} \\ M \leftarrow \sum_{i=0}^{rows(A)-2} \sqrt{1 + \left[A_{i+1} \cdot (rows(A) - 1) - A_i \cdot (rows(A) - 1) \right]^2} \\ h_6 \leftarrow 1 + ln\left(\frac{M}{rows(A) - 1} \right) \cdot (ln(M))^{-1} \\ h \end{array} \right.$$

```
PLCR(I) ≡ n ← rows(I_2)
          if n − 1 ≥ 0
            for i ∈ 0.. n − 1
              D1 ← E2RC(I, i)
              I_0 ← D1_0
              xm_i ← (D1_1)_0
              xx_i ← (D1_1)_1
              ym_i ← (D1_1)_2
              yx_i ← (D1_1)_3
            imp ← 0
          E_0 ← I_0
          E_1 ← xm
          E_2 ← xx
          E_3 ← ym
          E_4 ← yx
          E
```

```
HEPFRAME(I, x, y) ≡ for i ∈ 1.. rows(I) − 1
                      A ← 0
                      m ← 0
                      for j ∈ 0.. rows(I_i,8) − 1
                        for k ∈ 0.. cols(I_i,8) − 1
                          A_m ← (I_i,8)_j,k
                          m ← m + 1
                      H^⟨i−1⟩ ← HEP(A)
                      icls ← 2
                      for j ∈ 0.. rows(x) − 1
                        if x_j ≥ I_i,3 ∧ x_j ≤ I_i,5 ∧ y_j ≥ I_i,4 ∧ y_j ≤ I_i,6
                          icls ← 1
                          ijk ← −1
                      H_7,i−1 ← icls
                    H ← H^T
                    w ← APPENDPRN("feats.out" , H)
```

Calls Heptor for each candidate assembly frame and determines if a truth location is within the assembly (sets output variable for use in Data Model K-G algorithm used to derive the Decision Architecture)

Fig. 12. Part 5 of MathCAD 14 implementation of SV.

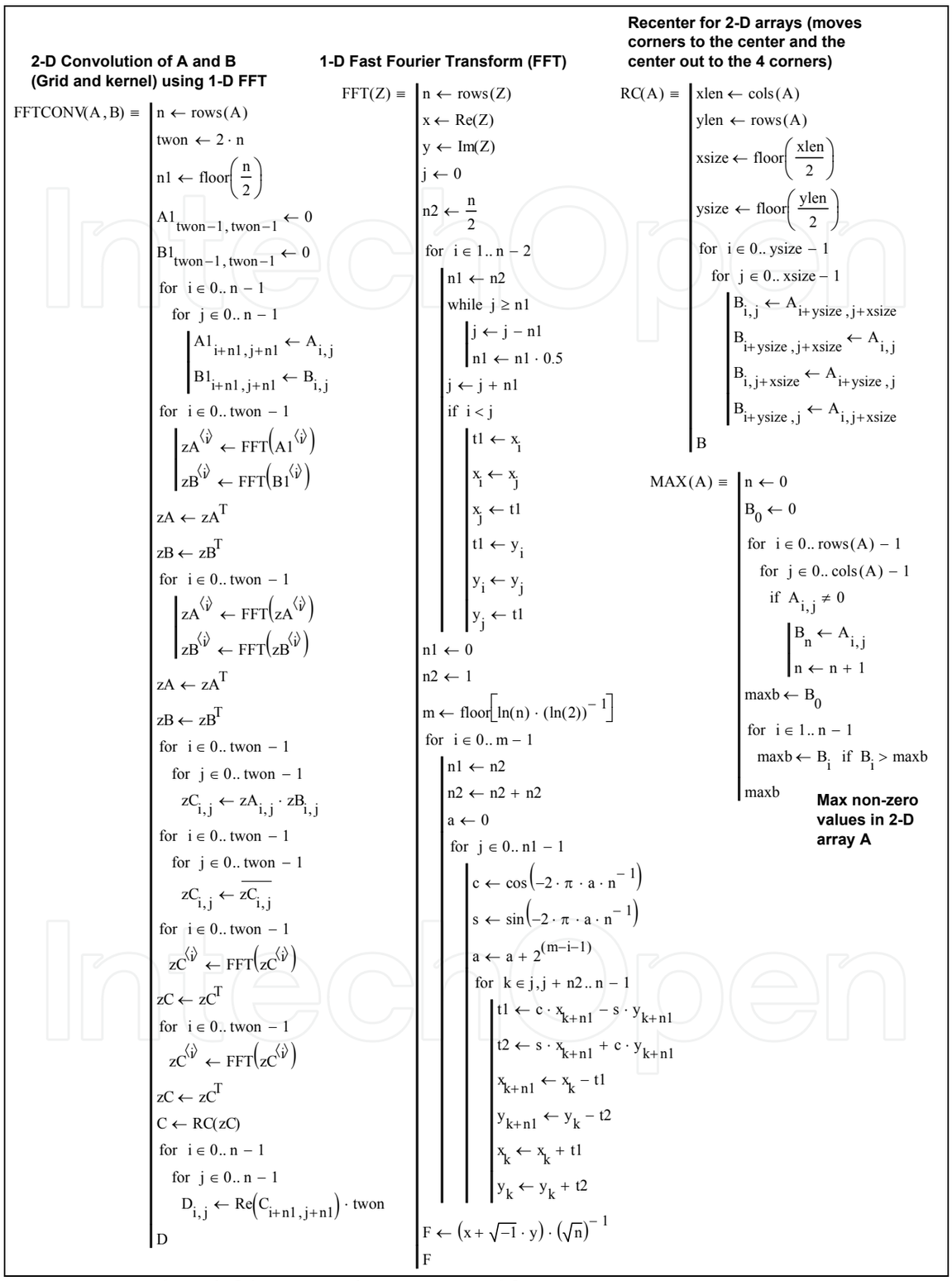


Fig. 13. Part 6 of MathCAD 14 implementation of SV.

The first 2 pages (Fig. 8 and 9) list the overall structure of the SV algorithm implementation (the main program body), and each of these 2 pages has been broken up into lettered sections with brief descriptions of each section. The remaining 4 pages (Fig. 10-13) are individual MathCAD programs that implement each of the specific functions used in SV, along with a general description of each function. When the MathCAD 14 document is loaded, a single case is generated. In order to vary the road and object placements, new individual cases can be generated by increasing the value of kkk1 (Fig. 9, Section I at the bottom of the figure) in integer steps. Alternatively, Monte Carlo cases can be generated using the Tool/Animation/Record pull down menu to load the movie recording capability in MathCAD 14. Place a fence around the kkk1 equation and set the FRAME variable to range from 0 to the number of Monte Carlos desired and set the time step to 1. The resultant HEPTOR features for each Monte Carlo are written into the file feats.out in the HEPFRAME function (note, delete this file from the directory containing the MathCAD 14 document before starting this process so that only the selected Monte Carlos are written into the file).

6. Classifier KG algorithm

To derive a general mathematical Data Model (Jaenisch and Handley, 2003), it is necessary to combine multiple input measurement variables to provide a classifier in the form of an analytical math model. Multivariate linear regression is used to derive an $O(3^n)$ Data Model fusing multiple input measurement sources or data sets and associated target label definitions. This is accomplished using a fast algorithm (flowchart in Fig. 14) that derives the coefficients of the approximation to the Kolmogorov-Gabor (KG) polynomial (which they proved to be a universal function or mathematical model for any dynamic process)

$$y(x_1, x_2, \ldots, x_L) = a_0 + \sum_i a_i x_i + \sum_i \sum_j a_{ij} x_i x_j + \sum_i \sum_j \sum_k a_{ijk} x_i x_j x_k + \cdots \quad (24)$$

which takes all available inputs in all possible combinations raised to all possible powers (orders).

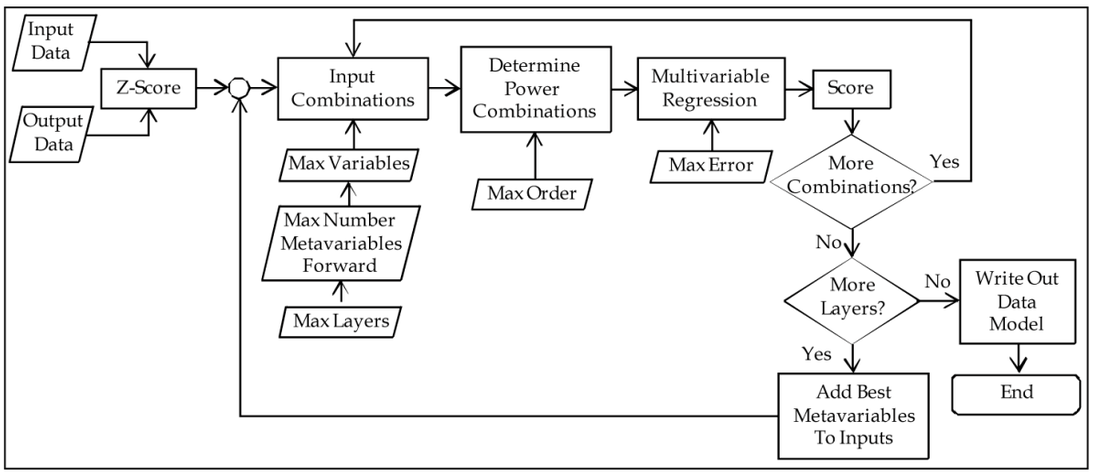


Fig. 14. Multivariable Data Model algorithm flowchart.

The full KG multinomial is impractical to derive directly. One method for approximating the KG polynomial is the Group Method of Data Handling (GMDH) algorithm (Madala and Ivakhnenko, 1994), which has been improved upon by the author into Data Modeling. Data Modeling uses multivariable linear regression to fit combinations of input variables (up to a user specified number at a time) to find the minimum error using either correlation or root sum square (RSS) differences between the regression output and the objective function. The best of these combinations (user specified number) are retained and used as metavariables (new inputs) and the process repeated at the next layer. Layering is terminated when overall desired RSS difference is minimized (Jaenisch and Handley, 2009). Figs. 16-20 on the following pages contain a MathCAD 14 implementation of Data Model K-G algorithm that was used to build the decision architecture in Section 7, and as before, Fig. 16 is broken up into sections for explanation.

7. Decision Architecture

It is possible to identify an optimal subset of the exemplars using available support vector finding machines; however, a good rule of thumb is to use 10% of the available exemplars. The SV algorithm in Figs. 8-13 was run for 50 epochs (FRAME ranging from 0 to 49), generating a total of 320 exemplars. The first 1/3 of these points (107 exemplars) was used as input into the MathCAD 14 document in Figs. 16-20. Fig. 16 shows the output results from this Data Model graphically at the bottom of the page. Two thresholds were set (lower threshold at 0.89 and an upper threshold at 1.92), and the exemplars cases which fell between the two thresholds were pulled out as the support vectors (87 of the 107 original cases were selected as support vectors) using the EXTR function provided.

Starting with these 87 exemplars, a new Data Model was generated using the decision architecture construction/execution flowchart in Fig. 15. Each node was constructed using the exemplars siphoned from the previous node (using EXTUP in the MathCAD document). The number of layers (nlay) was changed to 2 to make the Data Models shorter for publication in this work. A total of 3 nodes (bulk filter plus 2 resolvers) were required to learn these 87 support vector exemplars (with care taken to preserve each Data Model BASIC source code written out from the MathCAD 14 document at each siphon point along with the exemplar data).

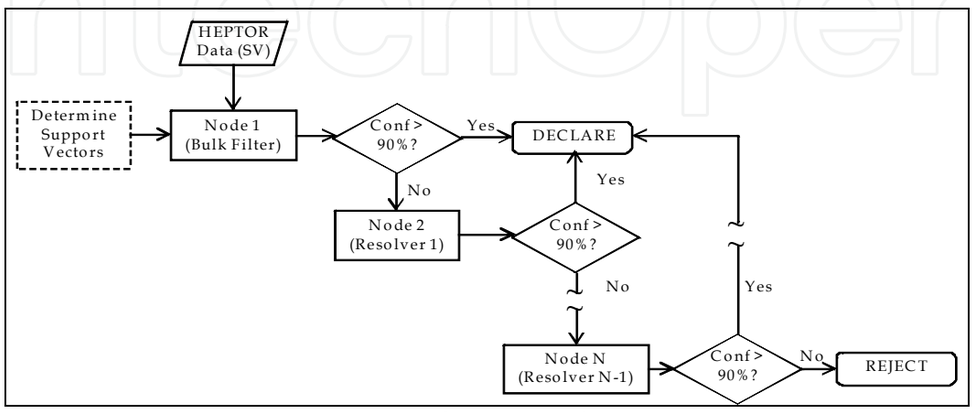


Fig. 15. Decision architecture construction/execution flowchart.

Derive Multi-Variable Data Model

A. Initialization

prec := 2 exitc := 0.0001 nfwd := 3 nlay := 4 maxvar := 3 maxorder := 3 X1 := READPRN("feats.out") nins := 7

Calc Prec **Exit Criteria** **# metvar forward** **# Layers** **Max # ins per bldg blk** **Max bldg blk Order** **Data file from SV** **# inputs**

outcol := 7 nsamp := 107

Output Data Col **# samps to build DM**

B. Pull out samples and Sort into ascending order on output (for visualization)

$i := 0..nsamp - 1$ $j := 0..nins - 1$ $X2_{i,j} := X1_{i,j}$ $X2_{i,outcol} := X1_{i,outcol}$ $X2 := CSORT(X2, outcol)$ $X_{i,j} := X2_{i,j}$

$Y_i := X2_{i,outcol}$ $j := 0..cols(X) - 1$ $Xtmp := X2$ $npts := rows(X)$ $i := 0..npts - 1$

C. Supply names for inputs and output $InNam_j := concat("z", num2str(j + 1))$ **Input names (z1 to z7)** $OutName := "y"$

D. Z-score inputs and output

$xavg_j := rflr(mean(X^{\langle j \rangle}), prec)$ $xdev_j := rflr(ADEV(X^{\langle j \rangle}, npts), prec)$ $yavg := rflr(mean(Y), prec)$

$ydev := rflr(ADEV(Y, npts), prec)$ $X_{i,j} := \frac{X_{i,j} - xavg_j}{xdev_j}$ $Y_i := \frac{Y_i - yavg}{ydev}$

E. Perform Multivariable linear regression (K-G) algorithm process

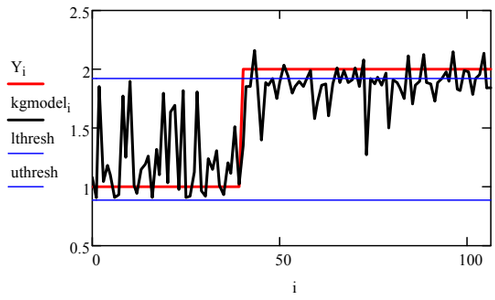
$I_0 := X$ $I_1 := InNam$ $I_2 := Y$ $I_3 := nlay$ $I_4 := nfwd$ $I_5 := exitc$ $I_6 := maxvar$ $I_7 := maxorder$ $I_8 := prec$ $I_9 := nins$

$kgmodel := DM(I)$ Derive KG Data Model

F. Undo Z-scoring on output to score

$kgmodel_i := kgmodel_i \cdot ydev + yavg$ $Y_i := Y_i \cdot ydev + yavg$

$rss := \sqrt{\sum_i (kgmodel_i - Y_i)^2}$ $rss = 2.99$



$lthresh \equiv 0.89$

$uthresh \equiv 1.92$

G. Tools for siphoning (used in decision architecture construction)

$B := EXTR(Xtmp, kgmodel, lthresh, uthresh)$ Used to select support vectors (between thresholds)

$B := EXTUP(Xtmp, kgmodel, uthresh)$ Enable for siphoning

H. Write out Data Model BASIC code

$f := "outbas.prn"$ $iscale_i := 1$ $jscale := 1$ KG output Data Model file $C1 := READPRN("fnldm.prn")$

$I1_0 := C1$ $I1_1 := OutName$ $I1_2 := yavg$ $I1_3 := ydev$ $I1_4 := f$ $I1_5 := prec$ $I1_6 := InNam$ $I1_7 := xavg$ $I1_8 := xdev$

$I1_9 := iscale$ $I1_{10} := jscale$ $COD := WHD(f, InNam)$ $WPG(I1)$

$mean(A) \equiv$ ym ← 0; for $i \in 0..rows(A) - 1$: $ym \leftarrow ym + \frac{A_i}{rows(A)}$; ym

Calculates the mean of 1-D vector A

$rflr(x, p) \equiv$ $x \leftarrow x \cdot 10^p$; $x \leftarrow floor(x + 0.5)$; $x \leftarrow x \cdot 10^{-p}$; x

Rounds x to p decimal places precision

B

Enable to write out siphoned data or support vectors

COD

Writes out the Data Model BASIC code

Fig. 16. Part 1 of MathCAD 14 implementation to derive a multivariable Data Model.

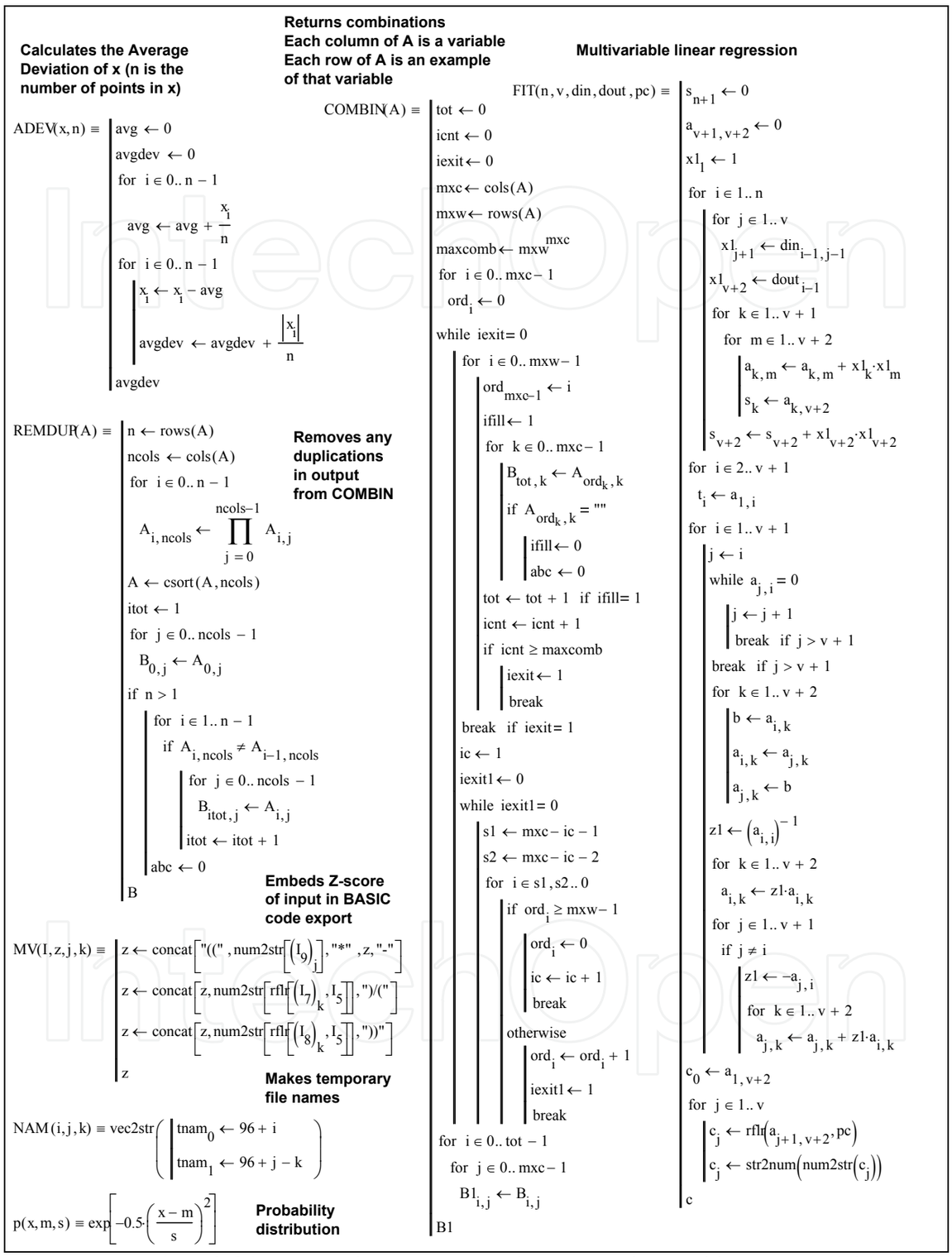


Fig. 17. Part 2 of MathCAD 14 implementation to derive a multivariable Data Model.

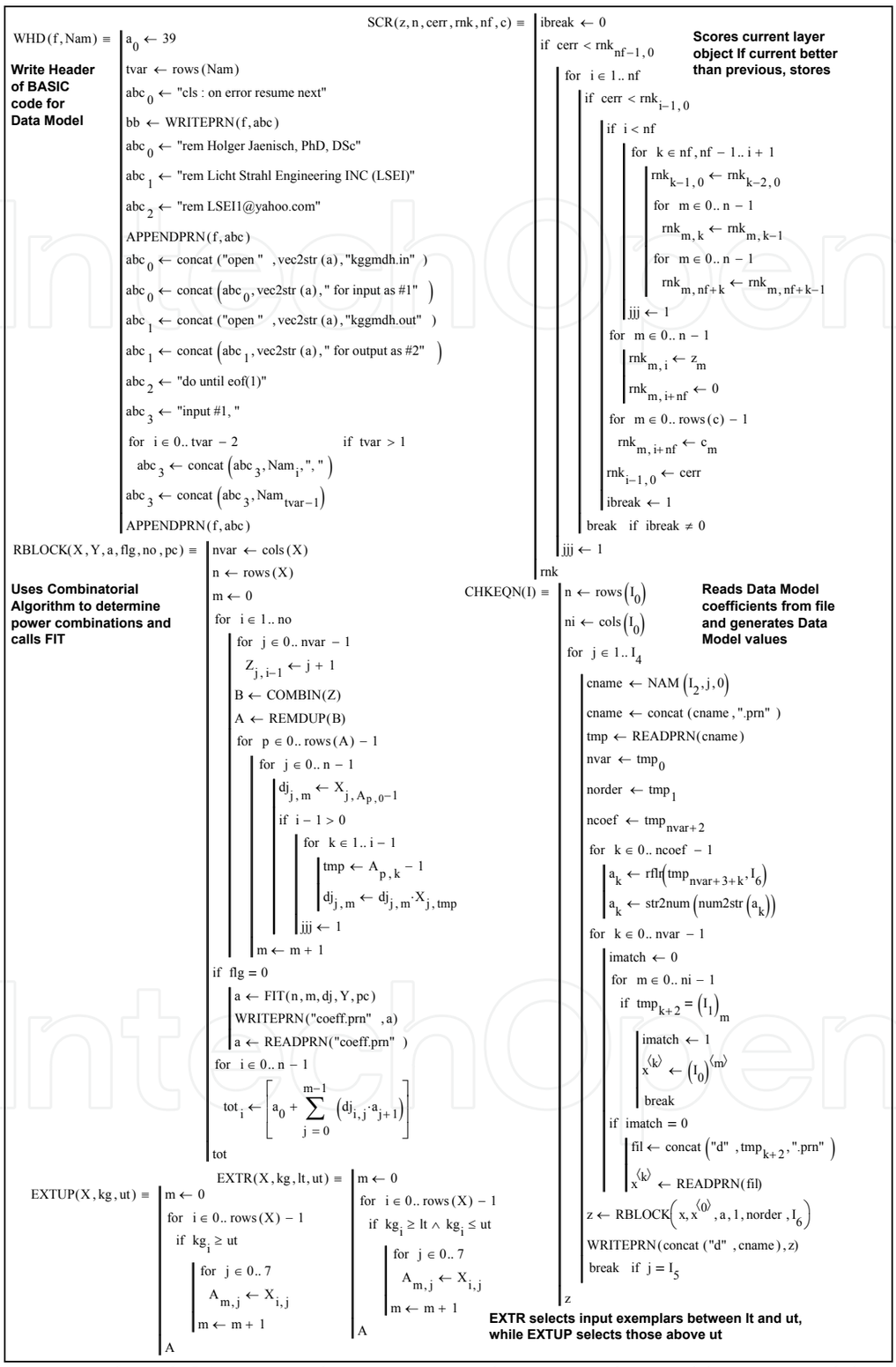


Fig. 18. Part 3 of MathCAD 14 implementation to derive a multivariable Data Model.

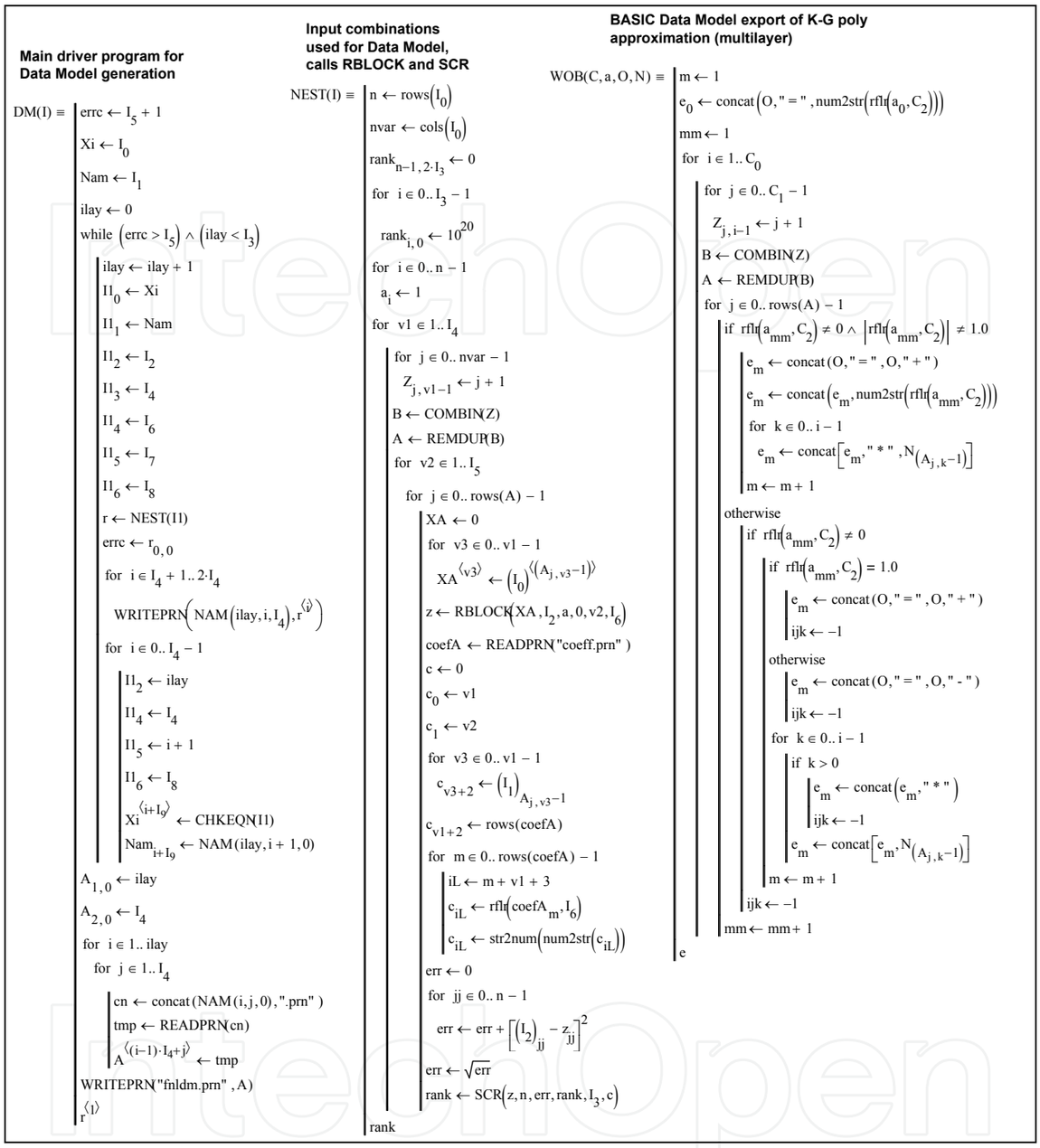


Fig. 19. Part 4 of MathCAD 14 implementation to derive a multivariable Data Model.

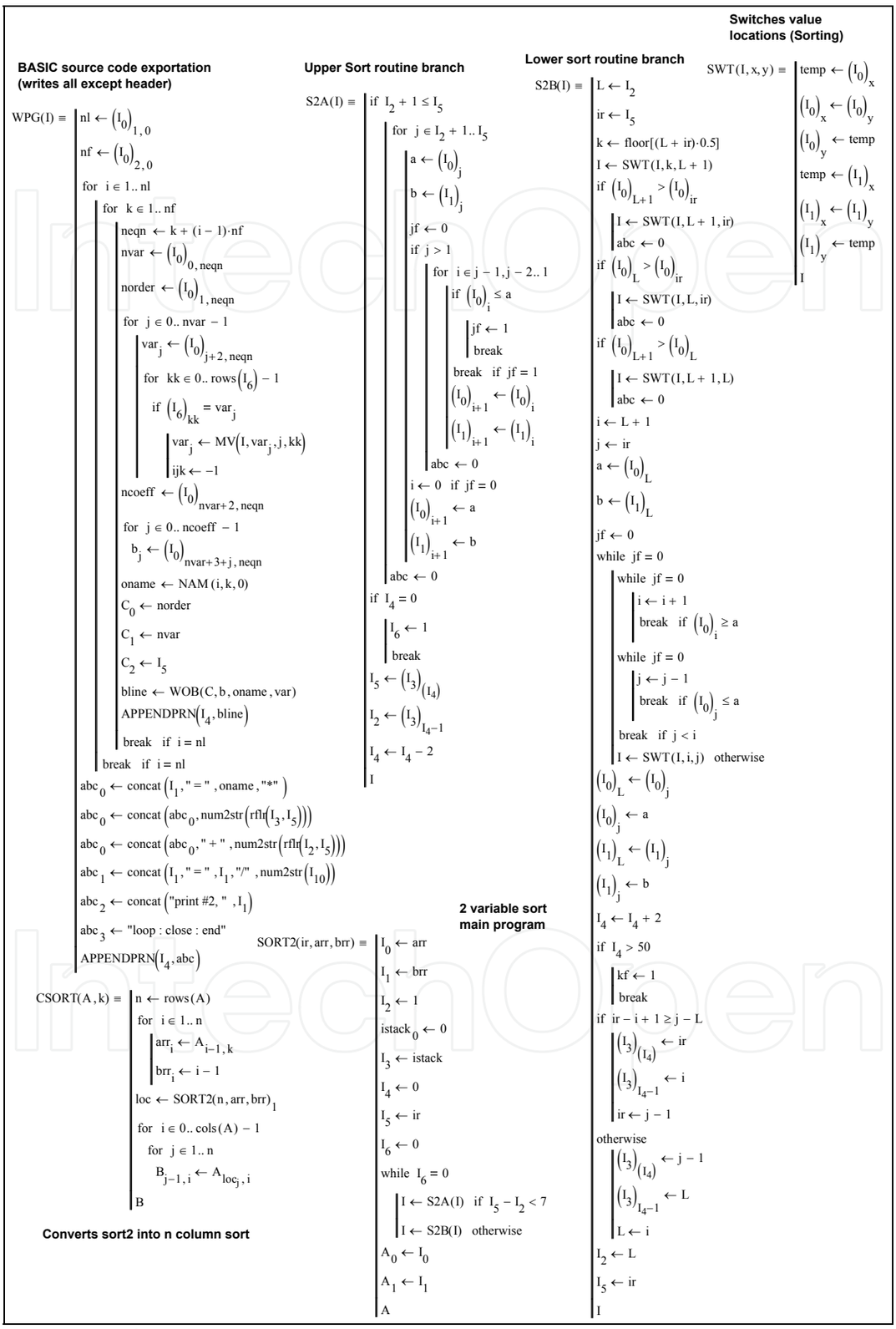


Fig. 20. Part 5 of MathCAD 14 implementation to derive a multivariable Data Model.

Fig. 21 shows the results of processing all of the 87 training exemplars thru the bulk filter and 2 resolver Data Models in this process (Jaenisch et.al., 2002)(2010). All examples for which the Data Model returns a value inside the lower and upper thresholds (labeled Declare on each graph) are declared targets, while those outside the upper and lower thresholds are deferred until the last resolver, where a reject decision is made. Rollup equations for each node in the decision architecture are also provided under each graph in Fig. 21. The coefficients in front of each variable are derived by first determining how many times the variable occurs in the actual multinomial, normalizing each by dividing by the number of occurrences of the least frequently occurring variable, summing the result, and dividing each result by the sum. By normalizing by the least frequently occurring variable first and then turning the number into a percentage by dividing by the result sum, the coefficients describe the key and critical feature contribution in the full Data Model.

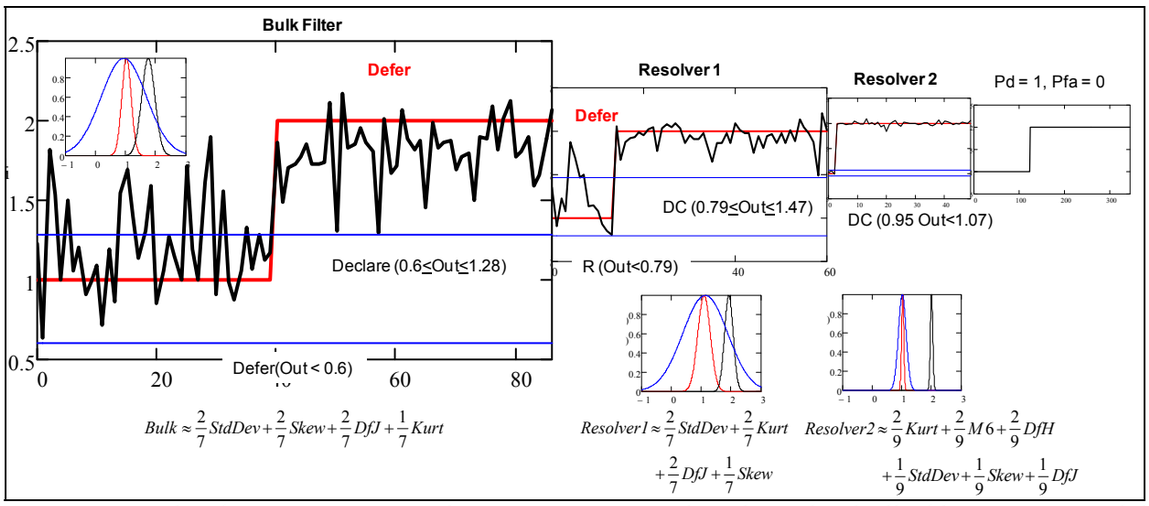


Fig. 21. Results from processing the training examples thru the bulk filter Data Model classifier and the ambiguity resolver. The entire decision architecture flows thru the bulk filter and, if required, as many of the ambiguity resolvers until either a reject or declare is determined.

The 3 BASIC files saved from deriving each of the nodes were combined together into the single decision architecture BASIC program given in Figs. 22 and 23. The value for each node in the decision architecture was converted into a confidence using the normal probability distribution defined by

$$Conf = \exp[-0.5(((Val - m)/s)^2)] \tag{25}$$

where Val is the value returned by the individual node in the decision architecture, m is the average between the upper and lower declare thresholds, and s (normally in the distribution the standard deviation) the value required so that Equation 25 returned a value of 0.9 (90% confidence) at the declaration thresholds. At the upper declaration threshold, no potential targets with a confidence of less than 90% are ever allowed to be declared, since they are labeled as defer by the decision architecture. All of the 320 examples were processed thru the decision architecture, yielding a probability of detection (Pd) of 0.65 and a probability of false alarm (Pfa) of 0.16.

```
CLS
ON ERROR RESUME NEXT
OPEN "feats.out" FOR INPUT AS #1
OPEN "dearch.out" FOR OUTPUT AS #2
DO UNTIL EOF(1)
INPUT #1, z1, z2, z3, z4, z5, z6, z7, trth
'z1 to z7 are heptor elements
'trth=truth class from SV file for comparison
GOSUB node1
IF p1 >= .9 THEN
 class=1
ELSE
 GOSUB node2
 IF p2 >= .9 THEN
  class=1
 ELSE
  GOSUB node3
  IF p3 >= .9 THEN
   class=1
  ELSE
   class=2
  END IF
 END IF
END IF
PRINT #2, class, trth
LOOP
CLOSE
END
node1:
aa=.35+.47*(z6-1.52)/(.14)
aa=aa-.52*(z1-370.42)/(73.83)
aa=aa+.4*(z2+.06)/(.56)
aa=aa-.07*(z6-1.52)/(.14)*(z6-1.52)/(.14)
aa=aa+.37*(z1-370.42)/(73.83)*(z6-1.52)/(.14)
aa=aa+.17*(z6-1.52)/(.14)*(z2+.06)/(.56)
aa=aa-.09*(z1-370.42)/(73.83)*(z1-370.42)/(73.83)
aa=aa+.33*(z1-370.42)/(73.83)*(z2+.06)/(.56)
aa=aa+.07*(z2+.06)/(.56)*(z2+.06)/(.56)
aa=aa-.04*(z6-1.52)/(.14)*(z6-1.52)/(.14)*(z6-1.52)/(.14)
aa=aa-.02*(z6-1.52)/(.14)*(z1-370.42)/(73.83)*(z6-1.52)/(.14)
aa=aa+.01*(z2+.06)/(.56)*(z6-1.52)/(.14)*(z6-1.52)/(.14)
aa=aa-.02*(z6-1.52)/(.14)*(z1-370.42)/(73.83)*(z1-370.42)/(73.83)
aa=aa-.05*(z6-1.52)/(.14)*(z1-370.42)/(73.83)*(z2+.06)/(.56)
aa=aa+.16*(z2+.06)/(.56)*(z2+.06)/(.56)*(z6-1.52)/(.14)
aa=aa-.02*(z1-370.42)/(73.83)*(z1-370.42)/(73.83)*(z2+.06)/(.56)
aa=aa+.09*(z2+.06)/(.56)*(z2+.06)/(.56)*(z1-370.42)/(73.83)
aa=aa-.02*(z2+.06)/(.56)*(z2+.06)/(.56)*(z2+.06)/(.56)
ab=.19-.14*(z3+.91)/(.53)
ab=ab-.39*(z1-370.42)/(73.83)
ab=ab+.41*(z6-1.52)/(.14)
ab=ab+.02*(z3+.91)/(.53)*(z3+.91)/(.53)
ab=ab-.05*(z1-370.42)/(73.83)*(z3+.91)/(.53)
ab=ab-.05*(z3+.91)/(.53)*(z6-1.52)/(.14)
ab=ab-.06*(z1-370.42)/(73.83)*(z1-370.42)/(73.83)
ab=ab-.01*(z1-370.42)/(73.83)*(z6-1.52)/(.14)
ab=ab-.1*(z6-1.52)/(.14)*(z6-1.52)/(.14)
ab=ab-.03*(z3+.91)/(.53)*(z3+.91)/(.53)*(z3+.91)/(.53)
ab=ab-.09*(z3+.91)/(.53)*(z1-370.42)/(73.83)*(z3+.91)/(.53)
ab=ab-.04*(z6-1.52)/(.14)*(z3+.91)/(.53)*(z3+.91)/(.53)
ab=ab+.08*(z3+.91)/(.53)*(z1-370.42)/(73.83)*(z1-370.42)/(73.83)
ab=ab-.16*(z3+.91)/(.53)*(z1-370.42)/(73.83)*(z6-1.52)/(.14)
ab=ab+.02*(z1-370.42)/(73.83)*(z1-370.42)/(73.83)*(z1-370.42)/(73.83)
ab=ab-.06*(z6-1.52)/(.14)*(z6-1.52)/(.14)*(z3+.91)/(.53)
ab=ab-.01*(z1-370.42)/(73.83)*(z1-370.42)/(73.83)*(z6-1.52)/(.14)
ab=ab+.08*(z6-1.52)/(.14)*(z6-1.52)/(.14)*(z1-370.42)/(73.83)
ab=ab-.01*(z6-1.52)/(.14)*(z6-1.52)/(.14)*(z6-1.52)/(.14)
ba=-.04+2*ab
ba=ba+.11*aa
ba=ba+.39*(z2+.06)/(.56)
ba=ba+1.3*ab*ab
ba=ba-2.28*aa*ab
ba=ba-.43*ab*(z2+.06)/(.56)
ba=ba+.24*aa*aa
ba=ba+.06*aa*(z2+.06)/(.56)
ba=ba-.03*(z2+.06)/(.56)*(z2+.06)/(.56)
ba=ba-.55*ab*ab*ab
ba=ba-.24*ab*aa*ab
ba=ba+.46*(z2+.06)/(.56)*ab*ab
ba=ba+.38*ab*aa*aa
ba=ba-2.29*ab*aa*(z2+.06)/(.56)
ba=ba-.67*aa*aa*aa
ba=ba-.79*(z2+.06)/(.56)*(z2+.06)/(.56)*ab
ba=ba+1.34*aa*aa*(z2+.06)/(.56)
ba=ba+.55*(z2+.06)/(.56)*(z2+.06)/(.56)*aa
ba=ba-.1*(z2+.06)/(.56)*(z2+.06)/(.56)*(z2+.06)/(.56)
da1=ba*.5+1.54
p1=EXP(-.5*((da1-.94)/.75) ^ 2)
RETURN
node2:
aa=-.58-.27*(z3+.98)/(.5)
aa=aa-.55*(z1-343.23)/(50.85)
aa=aa+1.12*(z6-1.54)/(.14)
aa=aa+1.16*(z3+.98)/(.5)*(z3+.98)/(.5)
aa=aa+.72*(z1-343.23)/(50.85)*(z3+.98)/(.5)
aa=aa-.53*(z3+.98)/(.5)*(z6-1.54)/(.14)
aa=aa+.32*(z1-343.23)/(50.85)*(z1-343.23)/(50.85)
aa=aa-.57*(z1-343.23)/(50.85)*(z6-1.54)/(.14)
aa=aa+.1*(z6-1.54)/(.14)*(z6-1.54)/(.14)
aa=aa+.19*(z3+.98)/(.5)*(z3+.98)/(.5)*(z3+.98)/(.5)
aa=aa+1.15*(z3+.98)/(.5)*(z1-343.23)/(50.85)*(z3+.98)/(.5)
aa=aa-.39*(z6-1.54)/(.14)*(z3+.98)/(.5)*(z3+.98)/(.5)
aa=aa+.48*(z3+.98)/(.5)*(z1-343.23)/(50.85)*(z1-343.23)/(50.85)
aa=aa-1.34*(z3+.98)/(.5)*(z1-343.23)/(50.85)*(z6-1.54)/(.14)
aa=aa-.02*(z1-343.23)/(50.85)*(z1-343.23)/(50.85)*(z1-343.23)/(50.85)
aa=aa+.21*(z6-1.54)/(.14)*(z6-1.54)/(.14)*(z3+.98)/(.5)
aa=aa-.34*(z1-343.23)/(50.85)*(z1-343.23)/(50.85)*(z6-1.54)/(.14)
aa=aa+.75*(z6-1.54)/(.14)*(z6-1.54)/(.14)*(z1-343.23)/(50.85)
aa=aa-.29*(z6-1.54)/(.14)*(z6-1.54)/(.14)*(z6-1.54)/(.14)
ab=.29+.3*(z6-1.54)/(.14)
ab=ab-.24*(z1-343.23)/(50.85)
ab=ab+.15*(z2+.23)/(.5)
ab=ab-.45*(z6-1.54)/(.14)*(z6-1.54)/(.14)
ab=ab+.69*(z1-343.23)/(50.85)*(z6-1.54)/(.14)
ab=ab-.85*(z6-1.54)/(.14)*(z2+.23)/(.5)
ab=ab-.31*(z1-343.23)/(50.85)*(z1-343.23)/(50.85)
ab=ab+1.02*(z1-343.23)/(50.85)*(z2+.23)/(.5)
ab=ab-.3*(z2+.23)/(.5)*(z2+.23)/(.5)
ab=ab-.3*(z6-1.54)/(.14)*(z6-1.54)/(.14)*(z6-1.54)/(.14)
ab=ab+.39*(z6-1.54)/(.14)*(z1-343.23)/(50.85)*(z6-1.54)/(.14)
ab=ab-.5*(z2+.23)/(.5)*(z6-1.54)/(.14)*(z6-1.54)/(.14)
ab=ab+.15*(z6-1.54)/(.14)*(z1-343.23)/(50.85)*(z1-343.23)/(50.85)
ab=ab-.04*(z6-1.54)/(.14)*(z1-343.23)/(50.85)*(z2+.23)/(.5)
ab=ab-.16*(z1-343.23)/(50.85)*(z1-343.23)/(50.85)*(z1-343.23)/(50.85)
ab=ab-.03*(z2+.23)/(.5)*(z2+.23)/(.5)*(z6-1.54)/(.14)
ab=ab+.46*(z1-343.23)/(50.85)*(z1-343.23)/(50.85)*(z2+.23)/(.5)
ab=ab-.13*(z2+.23)/(.5)*(z2+.23)/(.5)*(z1-343.23)/(50.85)
ab=ab-.05*(z2+.23)/(.5)*(z2+.23)/(.5)*(z2+.23)/(.5)
ba=.45-.88*ab
ba=ba+.46*aa
ba=ba+.11*(z3+.98)/(.5)
ba=ba+.74*ab*ab
ba=ba-.47*aa*ab
ba=ba-.47*ab*(z3+.98)/(.5)
ba=ba-.07*aa*aa
ba=ba-.99*aa*(z3+.98)/(.5)
ba=ba-.09*(z3+.98)/(.5)*(z3+.98)/(.5)
ba=ba+.86*ab*ab*ab
ba=ba-.14*ab*aa*ab
ba=ba-.64*(z3+.98)/(.5)*ab*ab
ba=ba+.19*ab*aa*aa
ba=ba-.3*ab*aa*(z3+.98)/(.5)
ba=ba-.13*aa*aa*aa
ba=ba+.49*(z3+.98)/(.5)*(z3+.98)/(.5)*ab
ba=ba+1.1*aa*aa*(z3+.98)/(.5)
ba=ba+.12*(z3+.98)/(.5)*(z3+.98)/(.5)*aa
ba=ba-.03*(z3+.98)/(.5)*(z3+.98)/(.5)*(z3+.98)/(.5)
da2=ba*.35+1.77
p2=EXP(-.5*((da2-1.13)/.75) ^ 2)
RETURN
node3:
aa=1.94+14.85*(z3+.96)/(.55)
aa=aa-12.06*(z4+8.56)/(4.29)
aa=aa+2.26*(z7-1.36)/(.02)
aa=aa-35.27*(z3+.96)/(.55)*(z3+.96)/(.55)
aa=aa+91.74*(z4+8.56)/(4.29)*(z3+.96)/(.55)
aa=aa-.31*(z3+.96)/(.55)*(z7-1.36)/(.02)
aa=aa-57.86*(z4+8.56)/(4.29)*(z4+8.56)/(4.29)
aa=aa+.8*(z4+8.56)/(4.29)*(z7-1.36)/(.02)
aa=aa-.09*(z7-1.36)/(.02)*(z7-1.36)/(.02)
aa=aa-9.12*(z3+.96)/(.55)*(z3+.96)/(.55)*(z3+.96)/(.55)
aa=aa+11.75*(z3+.96)/(.55)*(z4+8.56)/(4.29)*(z3+.96)/(.55)
aa=aa-6.98*(z7-1.36)/(.02)*(z3+.96)/(.55)*(z3+.96)/(.55)
aa=aa-2.62*(z3+.96)/(.55)*(z4+8.56)/(4.29)*(z4+8.56)/(4.29)
aa=aa+12.73*(z3+.96)/(.55)*(z4+8.56)/(4.29)*(z7-1.36)/(.02)
```

Fig. 22. BASIC source code for the decision architecture (Part 1 of 2).

```
aa=aa+.16*(z4+8.56)/(4.29)*(z4+8.56)/(4.29)*(z4+8.56)/(4.29)
aa=aa+.69*(z7-1.36)/(.02)*(z7-1.36)/(.02)*(z3+.96)/(.55)
aa=aa-6.08*(z4+8.56)/(4.29)*(z4+8.56)/(4.29)*(z7-1.36)/(.02)
aa=aa-1.16*(z7-1.36)/(.02)*(z7-1.36)/(.02)*(z4+8.56)/(4.29)
aa=aa-.14*(z7-1.36)/(.02)*(z7-1.36)/(.02)*(z7-1.36)/(.02)
ab=1.15+11.53*(z3+.96)/(.55)
ab=ab-11.27*(z4+8.56)/(4.29)
ab=ab+.72*(z6-1.55)/(.13)
ab=ab-28.13*(z3+.96)/(.55)*(z3+.96)/(.55)
ab=ab+73.45*(z4+8.56)/(4.29)*(z3+.96)/(.55)
ab=ab+.21*(z3+.96)/(.55)*(z6-1.55)/(.13)
ab=ab-47.14*(z4+8.56)/(4.29)*(z4+8.56)/(4.29)
ab=ab-1.2*(z4+8.56)/(4.29)*(z6-1.55)/(.13)
ab=ab+.6*(z6-1.55)/(.13)*(z6-1.55)/(.13)
ab=ab-.05*(z3+.96)/(.55)*(z3+.96)/(.55)*(z3+.96)/(.55)
ab=ab-7.74*(z3+.96)/(.55)*(z4+8.56)/(4.29)*(z3+.96)/(.55)
ab=ab+10.68*(z6-1.55)/(.13)*(z3+.96)/(.55)*(z3+.96)/(.55)
ab=ab+11.2*(z3+.96)/(.55)*(z4+8.56)/(4.29)*(z4+8.56)/(4.29)
ab=ab-23*(z3+.96)/(.55)*(z4+8.56)/(4.29)*(z6-1.55)/(.13)
ab=ab-2.98*(z4+8.56)/(4.29)*(z4+8.56)/(4.29)*(z4+8.56)/(4.29)
ab=ab-7.06*(z6-1.55)/(.13)*(z6-1.55)/(.13)*(z3+.96)/(.55)
ab=ab+12.47*(z4+8.56)/(4.29)*(z4+8.56)/(4.29)*(z6-1.55)/(.13)
ab=ab+7.43*(z6-1.55)/(.13)*(z6-1.55)/(.13)*(z4+8.56)/(4.29)
ab=ab+.15*(z6-1.55)/(.13)*(z6-1.55)/(.13)*(z6-1.55)/(.13)
ac=-2.24+2.01*(z7-1.36)/(.02)
ac=ac-1.26*(z1-341.35)/(48.93)
ac=ac-.55*(z2+.31)/(.48)
ac=ac+.68*(z7-1.36)/(.02)*(z7-1.36)/(.02)
ac=ac+.31*(z1-341.35)/(48.93)*(z7-1.36)/(.02)
ac=ac+.39*(z7-1.36)/(.02)*(z2+.31)/(.48)
ac=ac+.4*(z1-341.35)/(48.93)*(z1-341.35)/(48.93)
ac=ac-.77*(z1-341.35)/(48.93)*(z2+.31)/(.48)
ac=ac+.75*(z2+.31)/(.48)*(z2+.31)/(.48)
ac=ac-.4*(z7-1.36)/(.02)*(z7-1.36)/(.02)*(z7-1.36)/(.02)
ac=ac+.22*(z7-1.36)/(.02)*(z1-341.35)/(48.93)*(z7-1.36)/(.02)
ac=ac+.26*(z2+.31)/(.48)*(z7-1.36)/(.02)*(z7-1.36)/(.02)
ac=ac-.12*(z7-1.36)/(.02)*(z1-341.35)/(48.93)*(z1-341.35)/(48.93)
ac=ac+.75*(z7-1.36)/(.02)*(z1-341.35)/(48.93)*(z2+.31)/(.48)
ac=ac+.32*(z1-341.35)/(48.93)*(z1-341.35)/(48.93)*(z1-341.35)/(48.93)
ac=ac-.44*(z2+.31)/(.48)*(z2+.31)/(.48)*(z7-1.36)/(.02)
ac=ac-.24*(z1-341.35)/(48.93)*(z1-341.35)/(48.93)*(z2+.31)/(.48)
ac=ac+.01*(z2+.31)/(.48)*(z2+.31)/(.48)*(z1-341.35)/(48.93)
ac=ac+.14*(z2+.31)/(.48)*(z2+.31)/(.48)*(z2+.31)/(.48)
ba=.53-.33*ac
ba=ba+.01*ab
ba=ba-.24*aa
ba=ba+.04*ac*ac
ba=ba-.16*ab*ac
ba=ba-.17*ac*aa
ba=ba-.12*ab*ab
ba=ba-.1*ab*aa
ba=ba+.14*aa*aa
ba=ba-.06*ac*ac*ac
ba=ba+.13*ac*ab*ac
ba=ba+.53*aa*ac*ac
ba=ba+.31*ac*ab*ab
ba=ba+.15*ac*ab*aa
ba=ba-.02*ab*ab*ab
ba=ba-.06*aa*aa*ac
ba=ba-.15*ab*ab*aa
ba=ba-.08*aa*aa*ab
ba=ba+.01*aa*aa*aa
da3=ba*.11+1.94
p3=EXP(-.5*((da3-1.01)/.13) ^ 2)
RETURN
```

Fig. 23. BASIC source code for the decision architecture (Part 2 of 2).

8. Summary

We use the Spatial Voting (SV) process for fusing spatial positions in a 2-D grid. This process yields a centroid and covariance estimate as the basis of robust cluster identification. We calculate a series of geospatial features unique to the identified cluster and attempt to identify unique and consistent features to enable automated target recognition. We define the geospatial features and outline our process of deriving a decision architecture populated with Data Models. We attempt to identify the support vectors of the feature space and enable the smallest subsample of available exemplars to be used for extracting the analytical rule equations. We present details of the decision architecture derivation process. We construct ambiguity resolvers to further sieve and classify mislabeled sensor hits by deriving a new resolver Data Model that further processes the output from the previous layer. In this fashion through a cascade filter, we are able to demonstrate unique classification and full assignment of all available examples even in high dimensional spaces.

9. Acknowledgements

The author would like to thank James Handley (LSEI) for programming support and proofreading this document; and Dr. William "Bud" Albritton, Jr., Dr. Nat Albritton, Robert Caspers, and Randel Burnett (Amtec Corporation) for their assistance with applications development, as well as their sponsorship of and technical discussions with the author.

10. References

Hall, D.L., & McMullen, S.A.H. (2004), *Mathematical Techniques in Multisensor Data Fusion*, Artech House, ISBN 0890065586, Boston, MA, USA.

Jaenisch, H.M., Albritton, N.G., Handley, J.W., Burnett, R.B., Caspers, R.W., & Albritton Jr., W.P. (2008), "A Simple Algorithm For Sensor Fusion Using Spatial Voting (Unsupervised Object Grouping)", Proceedings of SPIE, Vol. 6968, pp. 696804-696804-12, ISBN 0819471593, 17-19 March 2008, Orlando, FL, USA.

Jaenisch, H.M., & Handley, J.W. (2009), "Analytical Formulation of Cellular Automata Rules Using Data Models", Proceeding of SPIE, Vol. 7347, pp. 734715-734715-13, ISBN 0819476137, 14-15 April 2009, Orlando, FL, USA.

Jaenisch, H.M., & Handley, J.W. (2003), "Data Modeling for Radar Applications", Proceedings of IEEE Radar Conference 2003, ISBN 0780379209, 18-19 May 2003, Huntsville, AL, USA.

Jaenisch, H.M., Handley, J.W., Albritton, N.G., Koegler, J., Murray, S., Maddox, W., Moren, S., Alexander, T., Fieselman, W., & Caspers, R.T., (2010), "Geospatial Feature Based Automatic Target Recognition (ATR) Using Data Models", Proceedings of SPIE, Vol. 7697, 5-9 April 2010, Orlando, FL, USA.

Jaenisch, H.M., Handley, J.W., Massey, S., Case, C.T., & Songy, C.G. (2002), "Network Centric Decision Architecture for Financial or 1/f Data Models", Proceedings of SPIE, Vol. 4787, pp. 86-97, ISBN 0819445541, 9-10 July 2002, Seattle, WA, USA.

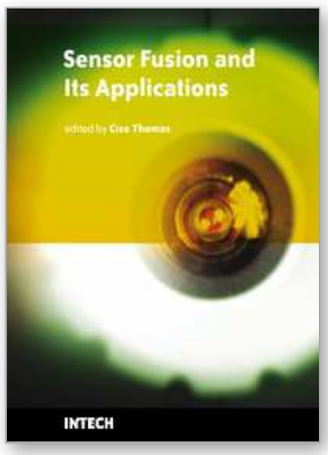
Jain, A.K. (1989), *Fundamentals of Digital Image Processing,* Prentice-Hall, ISBN 0133361659, Englewood Cliffs, NJ.

Klein, L. (2004), *Sensor and Data Fusion,* SPIE Press, ISBN 0819454354, Bellingham, WA.

Madala, H, & Ivakhnenko, A. (1994), *Inductive Learning Algorithms for Complex Systems Modeling*, CRC Press, ISBN 0849344387, Boca Raton, FL.

National Aeronautics and Space Administration (NASA) (1962), *Celestial Mechanics and Space Flight Analysis*, Office of Scientific and Technical Information, Washington, DC.

Press, W.H., Teukolsky, S.A., Vetterling, W.T., & Flannery, B.P. (2007), *Numerical Recipes: The Art of Scientific Computing, 3rd Edition,* Cambridge University Press, ISBN 0521880688, Cambridge, UK.



Sensor Fusion and its Applications

Edited by Ciza Thomas

ISBN 978-953-307-101-5
Hard cover, 488 pages
Publisher Sciyo
Published online 16, August, 2010
Published in print edition August, 2010

This book aims to explore the latest practices and research works in the area of sensor fusion. The book intends to provide a collection of novel ideas, theories, and solutions related to the research areas in the field of sensor fusion. This book is a unique, comprehensive, and up-to-date resource for sensor fusion systems designers. This book is appropriate for use as an upper division undergraduate or graduate level text book. It should also be of interest to researchers, who need to process and interpret the sensor data in most scientific and engineering fields. The initial chapters in this book provide a general overview of sensor fusion. The later chapters focus mostly on the applications of sensor fusion. Much of this work has been published in refereed journals and conference proceedings and these papers have been modified and edited for content and style. With contributions from the world's leading fusion researchers and academicians, this book has 22 chapters covering the fundamental theory and cutting-edge developments that are driving this field.

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen