As charged nuclear particles move through the liquid in the chamber, they leave small tracks of bubbles. A nuclear event occurs when an elementary particle interacts with an atomic nucleus in the liquid and produces a new group of particles moving in different directions.

Numbers have been assigned to the track images in the different views in the figure. The set of image-number triplets that corresponds to the correct track-image matching is

$$S = \{(1,1,3), (2,3,2), (3,2,1)\}$$

where, for example, the triplet (1,1,3) means that track image 1 in view 1, track image 1 in view 2, and track image 3 in view 3 are images of the same physical track.

**Producing the Lattice**

The computer's procedure for matching the track images begins with the construction of the correlation lattice, $E$. Each dimension of this lattice corresponds to a set of track images from one of the stereo views, and the indices along a dimension are the identifying numbers of the track images in that view. The value, one or zero, of an element $e_{ijk}$ in $E$ will indicate the possibility or impossibility, respectively, of matching image $i$ in view 1 with image $j$ in view 2 with image $k$ in view 3. Since all image combinations must initially be assumed to be possible, the procedure begins with all lattice elements equal to one. Figure 2 shows the initial lattice for our sample event.

Often the correlation lattice will not be cubic, as it was for our sample event. It is possible that spurious images may be seen or real images missed in some views. With the bubble chamber film, it may be difficult to dissociate the background tracks from the interesting interaction in some view, thus producing spurious images in that view. However, the measuring devices seldom miss track images. The strategy for dealing with the noncubic lattice for track matching is then rather simple: if $L$ is the number of images the code “expects” to match, then every view must have at least $L$ measured track images, and the solution set must contain $L$ triplets. Other applications may require other strategies.

The remainder of the matching procedure is an iterative process of applying successively more difficult tests of physical consistency to the image combinations in an attempt to eliminate matching possibilities from the correlation lattice, and, after each test sequence, scanning the lattice to determine the number of possible solutions sets that remain.
If there is no such set, or only one, the procedure ends. If there is more than one set, the lattice is still ambiguous, and it is necessary to go into the next test sequence.

**Testing Image Combinations**

A test may be concerned with an image from each of the three views (a triplet test) or it may compare images from only two views (a pair test). The pair test is very effective in the early stages of the procedure since, if an image pair is shown to be inconsistent, any triplet involving that pair must also be inconsistent. A single test can therefore eliminate many elements at a time.

The tests may be expressed as binary-valued functions. A pair test is written as the binary function $F_{i,j}(p,q)$ of the pair of image indices $p$ and $q$ from views $P$ and $Q$. When the image pair is not consistent with physical constraints, the value of the function is zero; otherwise its value is one. A similar definition is made of the triplet test $F_{i,j,k}(i,j,k)$.

For efficiency, we do not apply a test to any image combination that has already been eliminated by a previous test sequence. The determination of triplets to be tested with a triplet test is straightforward: A triplet is tested only if its corresponding element in $E$ has the value of one. For the set of pair tests $(F_{i,j}; F_{j,k}; F_{k,i})$, one view at a time must be systematically eliminated from the image combinations by forming the “projections” of $E$ along each of its three dimensions, one at a time. This produces the three two-dimensional-projection lattices $xE$, $yE$, and $zE$, where the elements of these lattices are given by

\[
xe = (xe_{ijk}) = \left( \begin{array}{c} \text{OR}_{i=1}^{n_1} e_{ijk} \\ \text{OR}_{j=1}^{n_2} e_{ijk} \\ \text{OR}_{k=1}^{n_3} e_{ijk} \end{array} \right)
\]

where $\text{OR}$ is the logical sum operator, i.e.,

\[
\begin{array}{c|ccc} \text{OR} & 0 & 1 \\ \hline 0 & 0 & 1 \\ 1 & 1 & 1 \end{array}
\]

The image pairs to be tested are determined by scanning the three projection lattices for elements with a value of one and testing only the corresponding pairs. The projections of the initial lattice shown in Fig. 2 will obviously produce projection lattices that have elements all equal to one.

**Recording Test Results**

Now that the image combinations to be tested are determined, it is now necessary to record the results of the test sequence in the correlation lattice $E$. For a triplet test this simply involves setting the lattice element corresponding to the triplet to zero if the value of the test function is zero. The procedure for recording the results of a pair-test sequence is to record the value of the test function in the two-dimensional-projection lattices that indicated the image pairs to be tested. After all image pairs have been tested, the resulting projection lattices are used to “mask” the correlation lattice $E$. This masking operation is written as

\[
E' = iE . \text{AND} . E
\]

or expressed at the element level for the view $\frac{2}{3}$ projection as

\[
e'_{ijk} = (e_{ijk}) . \text{AND} . (e_{ijk})
\]

for all $i$, $j$, and $k$. Here, $\text{AND}$ is the logical product operator

\[
\begin{array}{c|cc} \text{AND} & 0 & 1 \\ \hline 0 & 0 & 1 \\ 1 & 0 & 1 \end{array}
\]

The masking is done for all three projections, and the final lattice $E'$ replaces $E$ as the correlation lattice.

Returning to our sample event, suppose that the first test sequence was a pair test and eliminated five image pairs: $(1,2)$ and $(2,2)$ were inconsistent in views 1 and 2, and $(1,1)$, $(2,1)$, and $(2,3)$ were inconsistent in views 1 and 3. No pairs could be eliminated from views 2 and 3. Figure 3 shows the projection lattices containing the test results ready to be masked into the correlation lattice $E$.

**Scanning for Solutions**

After masking into the correlation lattice, we now scan the three-dimensional lattice for possible solutions. A solution in the lattice is defined as a set of lattice elements that all have a value of one and are chosen in such a way that no image in any view is used more than once. The length $L$ of the solution

---

From the collection of the Computer History Museum (www.computerhistory.org)
Figure 3. The projection lattices containing the test results ready to be masked into the correlation lattice.

set is defined as the number of elements in the solution set; in our sample event, the length is three.

Figure 4 shows the results of the masking operation on our sample event. The dashed lines represent the two possible solution sets that will be found by the scanning procedure.

The search for solutions in the three-dimensional lattice is based on the following proposition: If there exists a solution set consisting of $L$ triplets — $S(E)$ — in the three-dimensional lattice $E$, then at least one two-dimensional-solution set consisting of $L$ doublets — $S_k(E)$ — must also exist in each of the projection lattices of $E$; i.e., a solution in $E$ will project as a solution in any projection of $E$. A corollary to this proposition is that if no solution set of length $L$ is found in some projection of $E$, then no solution set of length $L$ exists in $E$.

Figure 4. The correlation lattice after making. Dashed lines show the two remaining solutions.

The scanning procedure is a four-step process. In the first step the lattice is reprojected along some axis to produce $3E$ ($3E$ in our case). In the second step this two-dimensional lattice is scanned for two-dimensional solutions of the required length by the technique of backtrack programming. When a solution is found, the index pairs forming the solution set are used to construct a two-dimensional “post-projection” lattice $P$ (step three) as described later. In step four this lattice is then scanned for solutions in order to determine the third index of the solution triplets. This type of scan fixes the indices of the triplets one view at a time — i.e., $i$ is fixed for view 1, then corresponding $j$'s are chosen from view 2, and this information is used to pick out the corresponding $k$'s from view 3. These steps are illustrated in Fig. 5.

As the three-dimensional solutions are found, they are used to construct a new three-dimensional lattice, $E_S$, that contains only elements known to be a part of some solution. This lattice will be used later to discard unused elements on the basis of logical inconsistency. (If it is found that an element is not now a part of some solution in $E$, it can never be a part of any solution, and it can then therefore be ignored in any further processing.)

The first step — to produce $3E$ — is the projection procedure of Eq. (2c). By the proposition stated above, any solution in $E$ will project as a solution in $3E$.

The second step — to scan $3E$ for a two-dimensional solution — is done with backtrack program-
The modified criterion function used in the scan of a projection lattice at the
$k$th row is

$$f_k(r) = \sum_{i=1}^{k} r_{i1} \geq L - (n - k) \quad (6)$$

i.e., there must be enough rows left for the scan to reach the final value of $L$. When $n$ and $L$ are equal (as in our sample event), this function reduces to

$$f_k(r) = k \quad (7)$$

In Fig. 5, the backtrack program begins by selecting $r_{11}$ from set $R_1$ and testing it with the modified criterion function $f_1(r)$. In this case

$$f_1(r_{11}) = 1$$

and the criterion is met. Having found the first component of our sample vector, $(r_{11}, -)$, the program now begins a search in $R_2$. The first choice is $r_{21}$ and by applying Eq. (7) we see that

$$f_2(r_{11}, r_{21}) = 2$$

However, the constraint that a column may be used only once eliminates this path from consideration.

The component $r_{21}$ is discarded, and we move to $r_{22}$. Here again we meet failure because

$$f_2(r_{11}, r_{22}) < 2$$

Continuing, we find that $r_{31}$ meets the requirements, and the sample vector is extended to $(r_{11}, r_{23}, -)$. The search now goes to $R_3$. As seen in Fig. 5, $r_{31}$ is the only member meeting the constraints, and so completes the vector $(r_{11}, r_{23}, r_{31})$. The solution set we have found in $\mathcal{E}$ is then

$$S(\mathcal{E}) = \{(1,1), (2,3), (3,2)\} \quad (8)$$

At this point in the scan procedure, the position of the scan in $\mathcal{E}$ is saved, and step three is entered.

The third step is to construct the post projection lattice, $P$, defined by the solution set just found in $\mathcal{E}$ [Eq. (8)]. The index pairs in this set prescribe the posts (parallel to view 3) in the correlation lattice $P$ that are to be used as rows in $P$. This projection is shown on the left of Fig. 5.

The fourth step—-to fix the third index of the solution triplets—is done by a scan of $P$ in the same manner as the scan at $\mathcal{E}$. As shown in the figure, the solution vector $(r_{13}, r_{22}, r_{31})$ is the only one possible, and a solution set has been found in the correlation lattice $E$—

$$S(E) = \{(1,1,2), (2,3,2), (3,2,1)\} \quad (9)$$

Elements corresponding to this set are made equal to 1 in the "solution lattice" $E_S$.

After the solution is found in $P$, the scan of $P$ will continue in an effort to find any more that might be present. If other solutions had been found, they also would have been entered into $E_S$.

When the scan for solution sets in the post projection has been exhausted, the program returns to step two to continue the scan of $\mathcal{E}$ from the point where the last solution set was found. If another solution set is found, steps three and four are again used to determine if the solution in $\mathcal{E}$ is the projection of a solution in the three-dimensional lattice. These procedures continue until the scan of $\mathcal{E}$ is exhausted, at which time the solution lattice, $E_S$, will contain the elements of all solutions found in the scan. There is a second possible solution for our sample event, with the resulting solution lattice shown in Fig. 6.

The solution lattice is used as a replacement for the original correlation lattice in order to eliminate all elements that do not contribute to some solution set. This lattice is now checked for elements common to all solutions in the lattice. These triplets are
fully determined and need no further testing. Their elements are set equal to zero, their indices are saved to become a part of the final solution, and the required solution length in $E$ is reduced accordingly. In Fig. 6 we see that the triplet $(3,2,1)$ is of this type and can be eliminated from the lattice. This leaves only four elements to be tested by the second test sequence, thus illustrating the power of what seemed to be a rather weak first test that could eliminate only 5 image pairs out of the 27 tested.

THE MULTIDIMENSIONAL CORRELATION LATTICE

The properties and procedures described thus far for the correlation lattice can be extended to the case of $n$ stereo views requiring an $n$-dimensional correlation lattice. If $m_i$ topological characteristics are seen in view $i$, the lattice will have dimensions of $m_1$ by $m_2$ by ... by $m_n$. Solutions in this lattice will be made of $L$ $n$-tuplets chosen with the same constraints as those used in the three-view case.

In this general case, the tests of image combinations can compare any number of views from 2 to $n$, and the concept of the pair and triplet test is extended to the more general $k$-tuplet test, a binary function of image indices from $k$ views.

It is possible to form various “orders” of projections of an $n$-dimensional lattice. The first-order projection, $iE$, that we used in the three-dimensional case still eliminates only one view, but it produces an $(n-1)$-dimensional projection lattice. The elements of the $iE$ projection are now written as

$$i_{E_{j-k}} = \sum_{l=1}^{m_1} e_{ij-k}$$

(10)

The second-order projection, $jkE$, eliminates two of the views and is formed by projection from the appropriate first-order projection. In this procedure, the projection operation is commutative, i.e.,

$$jkE = jE = iE$$

(11)

In general, it is then possible to discuss the $k$th-order projection in which $k$ of the views have been eliminated by successive projections.

An $(n-k)$-tuplet test will use the $k$th-order projection lattices to determine which image combinations it is to test and to record the results. However, the masking of the projections back into the lattice may be done in different ways. Storage requirements may prevent the keeping of all intervening projections to allow successively higher order masking back onto the $n$-dimensional lattice. It may be more efficient to record the test results directly into the lattice $E$; i.e., if the value of the test function is zero, then all elements in $E$ that use the same indices used by the test function are set to zero.

The scanning procedure for the $n$-dimensional lattice is simply a continuation of the three-dimensional scan. A solution set in the two-dimensional projection lattice, $3_{-2}E$, defines the posts in the three-dimensional lattice, $4_{-3}E$, to be used in the construction of the post-projection lattice. A solution set in this lattice then defines posts in the four-dimensional lattice, and a second two-dimensional post projection is constructed. This continues as far as the selection of posts from the $n$-dimensional lattice $E$ in order to determine the final index of the $n$-tuplets forming the solution set.

SUMMARY

The correlation lattice and the iterative backtrack scheme of scanning for solution sets of matching triplets were used as the basis for a code written in the FORTRAN IV language to match track images of nuclear-particle events. This code has been running on the IBM 7094 for over a year, and timing studies show that it can match images of events with seven images in each view in less than one half second, with most of that time devoted to computation for the various image-combination tests.

As developed, the program is divided into two logically separate sections (test and logic) with a
simple interface. The tests are represented as FORTRAN logical functions which have values ".TRUE." or ".FALSE." corresponding to the possibility or impossibility, respectively, of matching the image combination being tested. This division allows the tests to be developed independently of the logic section of the code and permits easy development of new programs using the same solution-finding technique.

The literature offers several examples of the use of backtrack programming. As Golomb and Baumert so candidly state in their excellent summary of the technique, "Backtrack has been independently 'discovered' and applied by many people." We regret that we are a member of those ranks. However, the notion of using backtrack in an iterative sense as we have done seems to be new. We feel that backtrack in any form may offer other people a very useful tool and hope that wider publication of the method will result in fewer independent discoveries.

ACKNOWLEDGMENTS

We thank Dr. Frank Solmitz and Dr. Margaret Alston for their suggestions and encouragement during the development of the concepts and methods of the correlation lattice.

REFERENCES


A PATTERN RECOGNITION TECHNIQUE AND ITS APPLICATION TO HIGH-RESOLUTION IMAGERY

R. D. Joseph and S. S. Viglione
Astropower Laboratory, Missile & Space Systems Division, Douglas Aircraft Company, Inc.
Newport Beach, California

SCOPE OF THE PROGRAM

The purpose of this project was to extend the study of the feasibility of automatic TIROS photograph analysis. A specific objective was to arrive at the design specification for a feasibility model of an automatic vortex recognition system.

The study had four main phases: 1) the development of logical design techniques applicable to the analysis of TIROS photographs, 2) experimental investigations of selected design parameters on a simplified problem, 3) the simulated design of a vortex recognition system using actual TIROS photographs, and 4) the consideration of suitable hardware for implementing the system.

In the study of the design approach, a variation of the discriminant analysis-iterative design technique was developed. A description of this technique is given in the next section along with a discussion on discriminant analysis, in which alternative pattern differences are assumed. Only one of these, one in which differences in covariance matrices of the pattern distributions are exploited, is suitable for the cloud pattern analysis. This discriminant analysis yields a quadratic discriminant surface and requires complex hardware for mechanization of the resulting first layer logic units. To simplify the implementation of the system, an approximation to the quadratic unit was developed. The approximate unit is derived from a principal axis solution, and substitutes a pair of parallel hyperplanes for the quadratic switching surface of the more complex unit. Methods have been developed to make the system invariant to changes in the brightness and contrast of the input patterns. This invariance is considerably more effective than a simple normalization of the input pattern, as it is achieved by making each logic unit invariant to such changes. The iterative design process itself is a means for assigning output weights to the logic units, and for emphasizing the difficult patterns. In common with the popular error correction methods, iterative design will find a solution whenever it is possible to assign these output weights to give perfect performance on the sample patterns. Unlike error correction, iterative design provides an "optimum" set of weights when no solution exists, and maximizes the switching surface to pattern distances when one does exist.

A portion of the experimental program was performed on a simplified problem using low-resolution, hand-printed alphabetic characters. These studies were used to investigate selected aspects of the design technique, rather than its application. The simplified problem permitted a very much more extensive investigation than would be possible on the cloud patterns. The average computer simulation time (IBM 7094) to design a recognition network for cloud patterns was five hours—for the alphabetic characters, five minutes. Two sets of